Mask Set Errata for Mask 3M17W

This report applies to mask 3M17W for these products:

- MPC5673F
- MPC5674F

Mask Specific Information

Major mask revision number	0x2
Minor mask revision number	0x0
JTAG identifier	0x1827_401D

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e3521	DECFIL: Soft reset failures at the end of filtering
e8251	DECFIL: timestamp may be lost in edge trigger mode
e9976	DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode
e6026	DSPI: Incorrect SPI Frame Generated in Combined Serial Interface Configuration
e7352	DSPI: reserved bits in slave CTAR are writable
e10755	DSPI: Transmit and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured
e4396	e200z7: Erroneous Address Fetch
e3419	e200z: Exceptions generated on speculative prefetch
e10799	EBI: Address 31 signal is not available in non-multiplexed mode
e6966	eDMA: Possible misbehavior of a preempted channel when using continuous link mode
e818	EMIOS/ETPU: Global timebases not synchronized
e11235	EMIOS: Any UC running in OPWMB or OPWMCB mode may function improperly if the source counter bus is used in another UC in MC or MCB mode
e11293	EMIOS: For any UC operating in OPWFMB mode the Channel Count register should not be written with a value greater than Channel B Data register value
e11295	EMIOS: In OPWFMB mode, A1/B1 registers do not get reloaded with A2/B2 register values if counter value returns 0x1 after counter wrap condition
e11294	EMIOS: OPWFMB and MCB mode counter rollover resets the counter to 0x0 instead of 0x1 as mentioned in the specification

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e9978	eMIOS: Unexpected channel flag assertion during GPIO to MCB mode transition
e4480	eQADC: Differential conversions with 4x gain may halt command processing
e3378	EQADC: Pull devices on differential pins may be enabled for a short period of time during and just after POR
e5086	eQADC: unexpected result may be pushed when Immediate Conversion Command is enabled
e2386	eSCI : No LIN frame reception after leaving stop mode
e1171	eSCI : DMA stalled after return from stop or doze mode
e1297	eSCI : reads of the SCI Data Register, which clears the RDRF flag, may cause loss of frame if read occurs during reception of the STOP bit
e2396	eSCI : Stop Mode not entered in LIN mode
e9344	eSCI: Late assertion of Transmit Data Ready Interrupt Flag (TXRDY) for Local Interconnect Network (LIN) frame receive (RX) operation
e9001	eSCI: Incorrect behavior while in LIN Standard Bit error detection mode
e1221	eSCI: LIN bit error indicated at start of transmission after LIN reset
e1381	eSCI: LIN Wakeup flag set after aborted LIN frame transmission
e7536	ESCI: Registers are writable in supervisor mode only
e9361	eSCI: Timing of TXRDY interrupt flag assertion is incorrect for LIN TX Frame
e9797	eSCI: Unable to send next frame after timeout in LIN mode
e5642	ETPU2: Limitations of forced instructions executed via the debug interface
e6309	ETPU2: STAC bus timebase export to peripherals does not work if the ratio of eTPU clock to peripheral clock is 2:1.
e2740	ETPU2: Watchdog Status Register (WDSR) may fail to update on channel timeout
e5640	ETPU2: Watchdog timeout may fail in busy length mode
e8194	eTPU: EAC may detect double teeth in a single input transition
e8252	eTPU: ETPU Angle Counter (EAC) Tooth Program Register (TPR) register write may fail
e9090	eTPU: Incorrect eTPU angle counter function under certain conditions
e9809	eTPU: MDU flags(Overflow/Carry) may be set incorrectly
e2157	FLASH: 4 wait states are required to run above 250MHz
e2382	FLASH: Flash Array Integrity Check
e1312	FLASH: MCR[DONE] bit may be set before high voltage operation completes when executing a suspend sequence
e3659	FLASH: Resuming after a suspend during an Erase may prevent the erase from completing.
e7322	FlexCAN: Bus Off Interrupt bit is erroneously asserted when soft reset is performed while FlexCAN is in Bus Off state
e3407	FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1
e2424	FlexCAN: switching CAN protocol interface (CPI) to system clock has very small chance of causing the CPI to enter an indeterminate state
e1364	FlexRay : Message Buffer Slot Status corrupted after system memory access timeout or illegal address access
e1369	FlexRay : Message Buffer Status, Slot Status, and Data not updated after system memory access timeout or illegal address access
e2302	FlexRay: Message Buffer can not be disabled and not locked after CHI command FREEZE

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e6726	NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled
e3553	NXFR: Flexray databus translates into unexpected data format on the Nexus interface
e1279	NZ7C3:Core Nexus Read/Write Access registers cleared by system reset
e7120	NZxC3: DQTAG implemented as variable length field in DQM message
e3377	Pad Ring:Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge
e9109	PAD_RING: Output pulse may occur on analog inputs during power on reset
e2696	PBRIDGE: Write buffer may cause overflow/underflow of DMA transfers
e11321	PIT_RTI: Generates false RTI interrupt on re-enabling
e2996	PIT_RTI: RTI timer corruption when debugging
e10783	PMC: BGTS (BandGap Temperature Sensor) may not assert at high temperature
e2322	PMC: LVREH/LVREA/LVRE50 may exit LVI triggered reset with LVI condition still existing
e1419	SIU: Reverting ENGCLK source to the system clock has a very small chance of causing the ENGCLK generator to enter an indeterminate state
e9658	SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event

Table 2. Revision History

Revision	Changes
July 2015	The following errata were added: 9001 9361
December 2016	The following errata were added. • e9797 • e9658 • e9344 • e9978 • e9976 • e9809 • e10542
September 2018	The following errata were removed. • e10542: (replaced by e10755) The following errata were added. • e11293 • e11295 • e11294 • e11235 • e2157 • e10799 • e11321

Table 2. Revision History

Revision	Changes
	• e10755 • e10783

e3521: DECFIL: Soft reset failures at the end of filtering

Description: If a software reset of a decimation filter is made exactly at the time it finishes filtering, several registers reset for one clock, but have their values updated by the filtering on the next clock,

including (but not limited to) the integrator current value register DECFIL_CINTVAL and

the tap registers DECFILTER_TAPn.

Workaround: Before making the soft reset write (DECFIL_MCR bit SRES=1), perform the procedure below:

1- disable filter inputs, writing DECFIL_MCR bit IDIS = 1.

2- read the register DECFIL_MSR and repeat the read until the bit BSY is 0.

3- repeat the loop of step 2; this is necessary to cover the case when a sample is left in the input buffer.

e8251: DECFIL: timestamp may be lost in edge trigger mode

Description: The Enhanced Queued Analog-to-Digital Converter (eQADC) supports a conversion command that configures it to send a timestamp along with the specified ADC conversion data to the Decimation Filter (DECFIL) for digital processing. The DECFIL receives the data and the timestamp, and updates internal registers with these two values. When the DECFIL is configured for edge triggered output by setting the Triggered Output Result Enable bit in the Module Configuration Register (DECFIL_MCR[TORE]) and setting the Trigger Mode bitfield to either 2b00 or 2b10, and a trigger edge is detected, the DECFILT loads an Internal Output Buffer register (DECFILT_IOB) with the conversion data, and then the timestamp data. This register is intended to hold data to be returned on one of the two Parallel Side Interfaces (PSI0 or PSI1). In the case where a trigger edge occurs and DECFILT_IOB is loaded with the conversion and timestamp data, and then a second trigger edge occurs before any new conversion and timestamp data has been received by the DECFILT, the DECFILT will provide only the initial conversion data, and will not provide the initial timestamp data.

The level triggered mode is not affected by this issue.

Workaround: When the DECFILT has been configured for edge triggered output buffer mode, ensure that the trigger edge rate is slower than the input data rate of the decimation filter. That is, be sure that there is always a new conversion arriving at the DECFILT before any new output trigger edge is detected. If the DECFILT is not receiving timestamps from the eQADC, this limitation is not required.

e9976: DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode

Description: When the Deserial Serial Peripheral Interface (DSPI) module is configured as follows:

1. Master mode is enabled (Master/Slave Mode Select bit in Module Configuration Register is set (DSPI_MCR [MSTR] = 0b1))

2. Modified transfer format is enabled (Modified Transfer Format Enable bit in Module Configuration Register is set (DSPI_MCR [MTFE] = 0b1))

3. Continuous serial communication clock mode is enabled (Continuous SCK Enable bit in Module Configuration Register is set (DSPI_MCR [CONT_SCKE] = 0b1))

In this configuration if the frame size of the current frame is greater than the frame size of the next received frame, corrupt frames are received in two scenarios:

a) Continuous Peripheral Chip Select Enable bit in PUSH TX FIFO Register is set (DSPI_PUSHR [CONT] = 0b1)

b) DSPI_PUSHR [CONT] = 0b0 and lower significant bit of the frame is transferred first (LSB first bit in Clock and Transfer Attributes Register is set (DSPI_CTAR [LSBFE] =0b1))

Workaround: To receive correct frames:

a) When DSPI_PUSHR [CONT] = 0b1, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).

b) When DSPI_PUSHR [CONT] = 0b0, configure DSPI_CTAR [LSBFE] = 0b0. Alternatively, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).

Make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

e6026: DSPI: Incorrect SPI Frame Generated in Combined Serial Interface Configuration

Description: In the Combined Serial Interface (CSI) configuration of the Deserial Serial Peripheral Interface (DSPI) where data frames are periodically being sent (Deserial Serial Interface, DSI), a Serial Peripheral Interface (SPI) frame may be transmitted with incorrect framing.

The incorrect frame may occur in this configuration if the user application writes SPI data to the DSPI Push TX FIFO Register (DSPI_PUSHR) during the last two peripheral clock cycles of the Delay-after-Transfer (DT) phase. In this case, the SPI frame is corrupted.

Workaround: Workaround 1: Perform SPI FIFO writes after halting the DSPI.

To prevent writing to the FIFO during the last two clock cycles of DT, perform the following steps every time a SPI frame is required to be transmitted:

Step 1: Halt the DSPI by setting the HALT control bit in the Module Configuration Register (DSPI_MCR[HALT]).

Step 2: Poll the Status Register's Transmit and Receive Status bit (DSPI_SR[TXRXS]) to ensure the DSPI has entered the HALT state and completed any in-progress transmission. Alternatively, if continuous polling is undesirable in the application, wait for a fixed time interval such as 35 baud clocks to ensure completion of any in-progress transmission and then check once for DSPI_SR[TXRXS].

Step 3: Perform the write to DSPI_PUSHR for the SPI frame.

Step 4: Clear bit DSPI_MCR[HALT] to bring the DSPI out of the HALT state and return to normal operation.

Workaround 2: Do not use the CSI configuration. Use the DSPI in either DSI-only mode or SPI-only mode.

Workaround 3: Use the DSPI's Transfer Complete Flag (TCF) interrupt to reduce worst-case wait time of Workaround 1.

Step 1: When a SPI frame is required to be sent, halt the DSPI as in Step 1 of Workaround 1 above.

Step 2: Enable the TCF interrupt by setting the DSPI DMA/Interrupt Request Select and Enable Register's Transmission Complete Request Enable bit (DSPI_RSER[TCF_RE])

Step 3: In the TCF interrupt service routine, clear the interrupt status (DSPI_SR[TCF]) and the interrupt request enable (DSPI_RSER[TCF_RE]). Confirm that DSPI is halted by checking DSPI_SR[TXRXS] and then write data to DSPI_PUSHR for the SPI frame. Finally, clear bit DSPI_MCR[HALT] to bring the DSPI out of the HALT state and return to normal operation.

e7352: DSPI: reserved bits in slave CTAR are writable

Description: When the Deserial/Serial Peripheral Interface (DSPI) module is operating in slave mode (the Master [MSTR] bit of the DSPI Module Configuration Register [DSPIx_MCR] is cleared), bits 10 to 31 (31 = least significant bit) of the Clock and Transfer Attributes Registers (DSPIx_CTARx) should be read only (and always read 0). However, these bits are writable, but setting any of these bits to a 1 does not change the operation of the module.

Workaround: There are two possible workarounds.

Workaround 1: Always write zeros to the reserved bits of the DSPIx_CTARn_SLAVE (when operating in slave mode).

Workaround 2: Mask the reserved bits of DSPIx_CTARn_SLAVE when reading the register in slave mode.

e10755: DSPI: Transmit and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured

Description: The Deserial/Serial Peripheral Interface Transmit and Receive First In/First Out (FIFO) buffers can request additional information to be transferred via the Direct Memory Access (DMA) module when either the Transmit or Receive FIFO Fill/Drain Flags are set in the DSPI Status Register (SR[TFFF/RFDF]). However, the Transmit Fill Flag indicates that at least 1 location each (2 bytes each) in the Transmit and Command FIFOs is available to be written. It does not indicate that the FIFO is empty. Similarly, Receive FIFO fill flag only indicates at least 1 location (2 bytes) of the FIFO is available to be read. It does not indicate that the FIFO is full. If the DMA is configured to transfer more than 1 FIFO location size of data, the FIFO Fill/Drain

Flags may not be properly cleared indicating that the FIFO is not full even when the FIFO is actually full (for Transmit FIFO) and not empty when the FIFO is actually empty (for Receive FIFO).

Workaround: Properly configure the DMA to fill/drain only 2 bytes to Transmit, Command and Receive FIFOs. Use the DMA loop to transfer more data if needed.

e4396: e200z7: Erroneous Address Fetch

Description: Under certain conditions, if a static branch prediction and a dynamic return prediction (which uses the subroutine return address stack) occur simultaneously in the Branch Target Buffer (BTB), the e200z7 core can issue an errant fetch address to the memory system (instruction fetched from wrong address).

This can only happen when the static branch prediction is “taken” but the branch actually resolves to “not taken”. If the branch resolves to taken, correct fetching occurs for this branch path and no issue is seen.

If Branch Unit Control and Status Register (BUCSR) Branch Prediction Control Static (BPRED) = 0b00, 0b01, or 0b10, then static branch prediction is configured as “taken”. The issue can occur with these settings.

If BUSCR[BPRED] = 0b11, then static branch prediction is configured as “not taken”. The issue does not occur with this setting.

Workaround: To prevent the issue from occurring, configure static branch prediction to “not taken” by setting the Branch Unit Control and Status Register (BUCSR) Branch Prediction Control Static (BPRED) to 0b11.

e3419: e200z: Exceptions generated on speculative prefetch

Description: The e200z7 core can prefetch up to 2 cache lines (64 bytes total) beyond the current instruction execution point. If a bus error occurs when reading any of these prefetch locations, the machine check exception will be taken. For example, executing code within the last 64 bytes of a memory region such as internal SRAM, may cause a bus error when the core pre-fetches past the end of memory. An ECC exception can occur if the core prefetches locations that are valid, but not yet initialized for ECC.

The Boot Assist Monitor (BAM) is located at the end of the address space and so may cause instruction pre-fetches to wrap-around to address 0 in internal flash memory. If this first block of flash memory contains ECC errors, such as from an aborted program or erase operation, a machine-check exception will be asserted. At this point in the boot procedure, exceptions are disabled, but the machine-check will remain pending and the exception vector will be taken if user application code subsequently enables the machine check interrupt.

Workaround: Do not place code to be executed within the last 64 bytes of a memory region. When executing code from internal ECC SRAM, initialize memory beyond the end of the code until the next 32-byte aligned address and then an additional 64 bytes to ensure that prefetches cannot land in uninitialized SRAM.

To guard against the possibility of the BAM causing a machine-check exception to be taken, as noted in the errata description, user application code should check and clear the Machine Check Syndrome Register (MCSR) in the core before enabling the machine check interrupt. This can be done by writing all 1s to MCSR.

e10799: EBI: Address 31 signal is not available in non-multiplexed mode

Description: The reference manual specifies that the address 31 signal of the External Bus Interface (EBI) should be available on pin D_CS2 in non-multiplexed mode. Instead, Data 31 is available on this pin.

Address 31 is needed for non-chip-select accesses that require byte addressing.

Workaround: Do not perform non-chip-select accesses in non-multiplexed mode to external devices that require byte addressing.

e6966: eDMA: Possible misbehavior of a preempted channel when using continuous link mode

Description: When using Direct Memory Access (DMA) continuous link mode Control Register Continuous Link Mode (DMA_CR[CLM]) = 1) with a high priority channel linking to itself, if the high priority channel preempts a lower priority channel on the cycle before its last read/write sequence, the counters for the preempted channel (the lower priority channel) are corrupted. When the preempted channel is restored, it continues to transfer data past its “done” point (that is the byte transfer counter wraps past zero and it transfers more data than indicated by the byte transfer count (NBYTES)) instead of performing a single read/write sequence and retiring.

The preempting channel (the higher priority channel) will execute as expected.

Workaround: Disable continuous link mode (DMA_CR[CLM]=0) if a high priority channel is using minor loop channel linking to itself and preemption is enabled. The second activation of the preempting channel will experience the normal startup latency (one read/write sequence + startup) instead of the shortened latency (startup only) provided by continuous link mode.

e818: EMIOS/ETPU: Global timebases not synchronized

Description: The eTPU and eMIOS timebases can be started synchronously by asserting the Global Time Base Enable (GTBE) bit in either the eTPU Module Configuration Register (eTPUMCR) or the eMIOS Module Configuration Register (eMIOS_MCR). GTBE bits from the eMIOS and eTPU are ORed together such that asserting either of them allows both eMIOS and eTPU timebases to start running simultaneously.

When the timebases are running and GTBE is cleared, the timebase and eTPU Angle Counter (EAC) prescalers can stop at any count. When GTBE is reasserted, the prescalers must have a determined initialization value so that the timebases will remain in sync with each other.

Workaround: Perform the following steps in sequence to synchronize the time base between the eTPU and the eMIOS:

1- Clear the Global Time Base Enable (GTBE) bit in the eTPU Module Configuration Register (ETPUMCR[GTBE] = 0).

2- Clear the Global Time Base Enable (GTBE) and Global Prescaler Enable (GPREN) bits in the eMIOS Module Configuration Register (EMIOS_MCR[GTBE] = 0, EMIOS_MCR[GPREN] = 0) to stop the Global Clock Prescaler (GCP) from counting.

3- Write the desired counter values to the eTPU Time Counter Registers 1 and 2 (TCR1 and TCR2) with eTPU microcode (these registers cannot be written by the CPU). This step is required even if the values in TCR1 and TCR2 are not changed.

4- Clear the Unified Channel Prescaler Enable (UCPREN) bit in the eMIOS Channel Control Register (EMIOS_CCRn) for each of the eMIOS channels (EMIOS_CCRn[UCPREN] = 0).

5- If it is necessary to set the eMIOS counters to known values, configure each eMIOS channel for General Purpose Input (GPI) mode (EMIOS_CCRn[MODE] = 0000000), write the desired counter values, and restore the desired operational mode (EMIOS_CCRn[MODE] = xxxxxxx).

6- Set the UCPREN bit in the EMIOS_CCRn register for each of the eMIOS channels (EMIOS_CCRn[UCPREN] = 1) to re-enable the unified channel prescaler.

7- Set the GTBE and GPREN bits in the eMIOS MCR simultaneously with a single write operation(EMIOS_MCR = EMIOS_MCR | 0x14000000).

Note: This works only for eTPU and single eMIOS module synchronization. For more than one eMIOS module there is no workaround, since their GPREN bits cannot be written at the same time.

e11235: EMIOS: Any UC running in OPWMB or OPWMCB mode may function improperly if the source counter bus is used in another UC in MC or MCB mode

Description: If a user configures any Unified Channel (UC) in Modulus Counter (MC) or Modulus Counter Buffered (MCB) mode by setting the eMIOS Channel Control Register MODE bitfield to 7'h10 or 7'h11 and if pre-scalers are enabled to increment the counter (GPREN bit of the Mode Control Register = 1'b1 or UCPREN bit of the Channel Control Register = 1'b1), then the UC does not trigger the counter bus reload event. The counter bus reload event is propagated as "counter bus sync" at UC output. If this particular UC is driving local or global buses of EMIOS then the bus_sync_signal is affected. This will manifest at least on any UC channel set in Center Aligned Output Pulse Width Modulation Buffered (OPWMCB) or Output Pulse Width Modulation Buffered (OPWMB) modes, and driven using the local or global bus by faulty UC channel.

Workaround: If any local or global bus in EMIOS is driven by Unified channels and the mode configuration of such unified channel control register is set to MC or MCB mode (Channel Control Register MODE bitfield is 7'h10 or 7'h11) and if the global prescaler is enabled in the module configuration register (GPREN bit of the Mode Control Register = 1'b1 or UCPREN bit of the Channel Control Register = 1'b1), then that bus should not be used by another UC with mode configuration register using OPWMCB or OPWMB mode (Channel Control Register MODE bitfield set to 7'b11000X0 or 7'b10111XX (X = don't care)).

e11293: EMIOS: For any UC operating in OPWFMB mode the Channel Count register should not be written with a value greater than Channel B Data register value

Description: For any Unified Channel (UC) running in Output Pulse-Width and Frequency Modulation Buffered (OPWFMB) mode, Channel Control Register MODE bitfield = 7'h1011000 or 7'h1011010, the internal counter runs from 0x1 to Channel B Data register value.

The internal counter can be overwritten by software using the Chanel Count register during 'freeze' operation.

If a counter wrap occurs due to overwriting of the counter with a value greater than its expiry value (B Data Register value); than the output signal behavior cannot be guaranteed.

Workaround: For any UC operating in OPWFMB mode the Channel Count register should not be written with a value greater than Channel B Data register value.

e11295: EMIOS: In OPWFMB mode, A1/B1 registers do not get reloaded with A2/B2 register values if counter value returns 0x1 after counter wrap condition

Description: In Output Pulse-Width and Frequency Modulation Buffered (OPWFMB) mode, A1/B1 registers do not get reloaded with A2/B2 register values if counter value returns 0x1 after counter wrap condition.

In order to avoid the counter wrap condition make sure internal counter value is within the 0x1 to B1 register value range when the OPWFMB mode is entered. Also overwriting of Channel Count register by forcing ‘freeze’ in OPWFMB mode should not take internal counter outside 0x1 to B register value.

Workaround: In order to avoid the counter wrap condition:

1. Make sure internal counter value is within the 0x1 to (B1 register) value range when the OPWFMB mode is entered.

2. Overwrite of Channel Count register by forcing ‘freeze’ in OPWFMB mode should not be outside the range of 0x1 to (B register) value.

e11294: EMIOS: OPWFMB and MCB mode counter rollover resets the counter to 0x0 instead of 0x1 as mentioned in the specification

Description: When the enhanced Modular Input/Output System (eMIOS) is used in Output Pulse-Width and Frequency Modulation Buffered (OPWFMB) or Modulus Counter Buffered (MCB) modes, when the counter rolls over, the counter returns to 0x0 instead of 0x1 as specified in the reference manual.

Workaround: In order to avoid the counter wrap condition:

1. Make sure internal counter value is within the 0x1 to (B1 register) value range when the OPWFMB mode is entered.

2. Overwrite of Channel Count register by forcing ‘freeze’ in OPWFMB mode should not be outside the range of 0x1 to (B register) value.

e9978: eMIOS: Unexpected channel flag assertion during GPIO to MCB mode transition

Description: When changing an Enhanced Modular IO Subsystem (eMIOS) channel mode from General Purpose Input/Output (GPIO) to Modulus Counter Buffered (MCB) mode, the channel flag in the eMIOS Channel Status register (eMIOS_Sn[FLAG]) may incorrectly be asserted. This will cause an unexpected interrupt or DMA request if enabled for that channel.

Workaround: In order to change the channel mode from GPIO to MCB without causing an unexpected interrupt or DMA request, perform the following steps:

(1) Clear the FLAG enable bit in the eMIOS Control register (eMIOS_Cn[FEN] = 0).

(2) Change the channel mode (eMIOS_Cn[MODE]) to the desired MCB mode.

(3) Clear the channel FLAG bit by writing ‘1’ to the eMIOS Channel Status register FLAG field (eMIOS_Sn[FLAG] = 1).

(4) Set the FLAG enable bit (eMIOS_Cn[FEN] = 1) to re-enable the channel interrupt or DMA request reaction.

e4480: eQADC: Differential conversions with 4x gain may halt command processing

Description: If the four times amplifier is enabled for a differential analog-to-digital conversion in the Enhanced Queued Analog to Digital Converter (eQADC) and the ADC clock prescaler is set to divide by 12 or greater, then the ADC will stop processing commands if a conversion command is executed immediately after a differential, gain 4x conversion.

Workaround: 1) Do not use a prescaler divide factor greater than or equal to 12 (11 can be used on devices that support odd prescalers).

2) Insert a dummy write command to any internal ADC register after every 4x conversion command.

Note 1: If the command FIFO preemption feature is used and it is possible to preempt a FIFO which contains the 4x conversion + dummy write workaround, then the preempting command FIFO must be loaded FIRST with a dummy write command and then the desired preempting conversion command in order to avoid the possibility of following a 4x conversion command with another conversion command in the same ADC.

Note 2: The level sensitive triggers (when in Low/High Level Gated External Trigger, Single/ Continuous Scan modes) can interrupt the command sequence at any point in time, potentially breaking the safe sequence 4x conversion command -> dummy write command.

Note 3: When using an odd prescaler (ADCx_CLK_ODD = 1), the duty cycle setting (ADCxCLK_DTY) must be kept at the default setting of 0.

e3378: EQADC: Pull devices on differential pins may be enabled for a short period of time during and just after POR

Description: The programmable pull devices (up and down) on the analog differential inputs of the eQADC may randomly be enabled during the internal Power On Reset (POR) and until the 1st clock edge propagates through the device. After the first clock edge, the pull resistors will be disabled until software enables them.

Workaround: Protect any external devices connected to the differential analog inputs. The worst case condition is with a 1.4K ohm resistor to VDDA (5K pull-up enabled) or VSSA (5K pull-down enabled). This may also cause temporary additional current requirements on the VDDA supply of each eQADC module, up to 15 mA on each eQADC if both the pull up and pull down resistors are enabled simultaneously on all of the differential analog pins.

e5086: eQADC: unexpected result may be pushed when Immediate Conversion Command is enabled

Description: In the enhanced Queued Analog to Digital Converter (eQADC), when the Immediate Conversion Command is enabled (ICEAn=1) in the eQADC_MCR (Module Configuration Register), if a conversion from Command First-In-First Out (CFIFO0, conv0) is requested concurrently with the end-of-conversion from another, lower priority conversion (convx), the result of the convx may be lost or duplicated causing an unexpected number of results in the FIFO (too few or too many).

Workaround: Workaround 1: Do not use the abort feature (ICEAn=0).

Workaround 2: Arrange the timing of the CFIFO0 trigger such that it does not assert the trigger at the end of another, lower priority conversion.

Workaround 3: Detect the extra or missing conversion result by checking the EQADC_CFTCRx (EQADC CFIFO Transfer Counter Register x). This register records how many commands were issued, so it can be used to check that the expected number of results have been received.

e2386: eSCI : No LIN frame reception after leaving stop mode

Description: When the eSCI module is in LIN mode and transmits or receives an LIN frame, if the CPU requests Stop Mode, and the Stop Mode is left, an subsequent triggered LIN RX Frame reception may hang. The module will never assert the eSCI_IFSR2[RXRDY] and eSCI_IFSR2[TXRDY] flags.

Workaround: The application should ensure that no LIN transmission is running before it requests Stop Mode by checking the status of the eSCI_IFSR1[TACT] and eSCI_IFSR1[RACT] status flags.

e1171: eSCI : DMA stalled after return from stop or doze mode

Description: If the eSCI module enters stop or doze mode while the transmit DMA is enabled and messages are being transmitted, when the CPU exits stop or doze mode, it is possible that DMA requests will not be generated by the eSCI module.

Workaround: The application should ensure that the eSCI module is idle before entering the stop mode.

The eSCI module is idle when both transmitter and receiver active status bits in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) are not set.

The application should not trigger a new transmission on the eSCI module if the application is preparing for the stop mode.

e1297: eSCI : reads of the SCI Data Register, which clears the RDRF flag, may cause loss of frame if read occurs during reception of the STOP bit

Description: A received SCI frame is not written into the SCI Data Registers and the Overrun (OR) flag is not set in the SCI Status Register 1 (SCISR1), if:

1.) The eSCI has received the last data bit of an SCI frame n

2.) and the Receive Data Register Full (RDRF) flag is still set in the SCISR1 after the reception of SCI frame n-1

3.) and during the reception of the STOP bit of frame n the host reads the SCI Data Registers, and clears the RDRF flag

In this case the RDRF flag is erroneously set again by the controller instead of the OR flag. Thus, the host reads the data of frame n-1 a second time, and the data of frame n is lost.

Workaround: The application should ensure that the data of the foregoing frame is read out from the SCI Data Registers before the last data bit of the actual frame is received.

e2396: eSCI : Stop Mode not entered in LIN mode

Description: When the eSCI module is in LIN mode and transmits the Header of an LIN RX frame, if the CPU requests Stop Mode, the eSCI module may not acknowledge the Stop Mode request and will stay in Normal Operating Mode (not in lower power stop mode).

Workaround: The application should ensure that no LIN transmission is running before it requests Stop Mode by checking the Transmit Active and Receive Active status bits in the eSCI Interrupt Flag and Status Register (eSCI_IFSR1[TACT] and eSCI_IFSR1[RACT]).

e9344: eSCI: Late assertion of Transmit Data Ready Interrupt Flag (TXRDY) for Local Interconnect Network (LIN) frame receive (RX) operation

Description: Assertion of the Transmit Data Ready Interrupt Flag (TXRDY) in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) indicates that data written to the LIN Transmit Register (eSCI_LTR) has been processed by the eSCI module. For the first three data writes to the eSCI_LTR during LIN frame generation, the TXRDY flag is asserted one clock cycle after the write access. During LIN RX operation, assertion of the TXRDY flag that coincides with the fourth data write to the eSCI_LTR is delayed. The TXRDY flag is not asserted until the LIN RX frame has been completely received from the slave device. The TXRDY flag is asserted when the Frame Complete Interrupt (FRC) flag of the eSCI_IFSR2 register is asserted.

Workaround: Application software should expect a delay in the assertion of the TXRDY flag after the fourth data write to the eSCI_LTR. Instead of expecting TXRDY assertion within one clock cycle of the fourth data write to the eSCI_LTR, application software should expect assertion of the TXRDY flag after the LIN RX frame has been completely received from the slave device.

e9001: eSCI: Incorrect behavior while in LIN Standard Bit error detection mode

Description: After a Local Interconnect Network (LIN) wake-up signal frame is transmitted from a master device while in Standard Bit error detection mode (eSCI_CR2[FBR] = 0), a bit error is detected in any subsequent LIN Transmit (TX) or Receive (RX) frames sent from the master device. After the bit error is detected, the Bit Error Interrupt Flag (eSCI_IFSR1[BERR]) is asserted, and the LIN controller will not generate TX or RX frames.

Workaround: Workaround 1: Reset the LIN Protocol Engine of the eSCI controller by writing 1 and then a 0 to the LIN Protocol Engine Stop and Reset bit in LIN Control Register 1 (eSCI_LCR1[LRES]) after a complete wake-up frame is sent.

Workaround 2: Use the LIN module in Fast Bit error detection mode, and do not use the Standard Bit error detection mode. Fast Bit Error detection mode can be enabled by writing 1 to the Fast Bit Error Detection bit in Control Register 2 (eSCI_CR2[FBR] =1).

e1221: eSCI: LIN bit error indicated at start of transmission after LIN reset

Description: If the eSCI module is in LIN mode and is transmitting a LIN frame, and the application sets and subsequently clears the LIN reset bit (LRES) in the LIN Control register 1 (ESCI_LCR1), the next LIN frame transmission might incorrectly signal the occurrence of bit errors (ESCI_IFSR1[BERR]) and frame error (ESCI_IFSR1[FE]), and the transmitted frame might be incorrect.

Workaround: There is no generic work around. The implementation of a suitable workaround is highly dependent on the application and a workaround may not be possible for all applications.

e1381: eSCI: LIN Wakeup flag set after aborted LIN frame transmission

Description: If the eSCI module is transmitting a LIN frame and the application sets and clears the LIN Finite State Machine Resync bit in the LIN Control Register 1 (eSCI_LCR1[LRES]) to abort the transmission, the LIN Wakeup Receive Flag in the LIN Status Register may be set (LWAKE=1).

Workaround: If the application has triggered LIN Protocol Engine Reset via the eSCI_LCR1[LRES], it should wait for the duration of a frame and clear the eSCI_IFSR2[LWAKE] flag before waiting for a wakeup.

e7536: ESCI: Registers are writable in supervisor mode only

Description: Write access to all Enhanced Serial Communication Interface (ESCI) registers is restricted to supervisor mode of the core performing the write.

Workaround: Ensure that the core which requires write access to ESCI registers is in supervisor mode.

e9361: eSCI: Timing of TXRDY interrupt flag assertion is incorrect for LIN TX Frame

Description: When generating a Local Interconnect Network (LIN) Transmit (TX) Frame, the Transmit Data Ready Interrupt flag (eSCI_IFSR2[TXRDY]) should assert after the transmission of the Identifier (ID) field. In the TX frame generation, however, the eSCI_IFSR2[TXRDY] asserts after the Sync field. All subsequent TXRDY Interrupt flags in the current frame assert after each subsequent byte field has been transmitted except for the final TXRDY Interrupt flag. The last TXRDY Interrupt flag asserts after the transmission of the checksum field.

Workaround: The timing of the TXRDY Interrupt flag cannot be changed from the incorrect behavior. The incorrect TXRDY Interrupt flag behavior does not affect LIN functionality. Even though the TXRDY Interrupt flag asserts earlier than expected, the TXRDY Interrupt flag still signals that the content of the LIN Transmit Register (eSCI_LTR) was processed by the LIN Protocol Engine.

e9797: eSCI: Unable to send next frame after timeout in LIN mode

Description: When generating a Local Interconnect Network (LIN) Transmit (Tx) and Receiver (Rx) frame, the Enhanced Serial Communication Interface (eSCI) module should first send the Header as per the LIN protocol. However, after the Slave Timeout Interrupt Flag (STO) in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) for the previous LIN Rx Frame is asserted (eSCI_IFSR2[STO]=1), the eSCI module is not able to generate the next Header, it remains in a suspended state.

Workaround: Perform the following actions in this order:

(1) Set the LIN Protocol Engine Stop and Reset (LRES) control bit to '1' in the LIN Control Register 1 (eSCI_LCR1).

(2) Wait until the status bits DACT, WACT, LACT, TACT,and RACT in the Interrupt Flag and Status Register (eSCI_IFSR1) are cleared to '0'.

(3) Clear LRES in eSCI_LCR1 to '0'.

(4) Begin transmission of the LIN Header for the next frame.

e5642: ETPU2: Limitations of forced instructions executed via the debug interface

Description: The following limitations apply to forced instructions executed through the Nexus debug interface on the Enhanced Time Processing Unit (ETPU):

1- When a branch or dispatch call instruction with the pipeline flush enabled (field FLS=0) is forced (through the debug port), the Return Address Register (RAR) is updated with the current program counter (PC) value, instead of PC value + 1.

2- The Channel Interrupt and Data Transfer Requests (CIRC) instruction field is not operational.

Workaround: Workaround for limitation #1 (branch or dispatch call instruction):

Increment the PC value stored in the RAR by executing a forced Arithmetic Logic Unit (ALU) instruction after the execution of the branch or dispatch call instruction.

Workaround for limitation #2 (CIRC):

To force an interrupt or DMA request from the debugger:

1- Program a Shared Code Memory (SCM) location with an instruction that issues the interrupt and/or DMA request. Note: Save the original value at the SCM location.

2- Save the address of the next instruction to be executed.

3- Force a jump with flush to the instruction position.

4- Single-step the execution.

5- Restore the saved value to the SCM location (saved in step 1).

6- Force a jump with flush to the address of the next instruction to be executed (saved in step 2).

NOTE: This workaround cannot be executed when the eTPU is in HALT_IDLE state.

e6309: ETPU2: STAC bus timebase export to peripherals does not work if the ratio of eTPU clock to peripheral clock is 2:1.

Description: The Shared Time Angle Counter (STAC) bus allows an Enhanced Time Processing Unit (eTPU) to export its timebase or angle counters to another eTPU as well as to other peripherals such as the Enhanced Modular Input/Output Subsystem (eMIOS) and Enhanced Queued Analog-to-Digital Converter (eQADC). If the eTPU clock is configured to be twice the frequency of those peripherals, the STAC bus will not be able to transfer timebase or angle information from the eTPUs to the slower peripherals. The timebase/angle export between eTPUs, however, is still operational in this configuration.

Workaround: Configure the eTPU clock to the same frequency as peripherals if timebase/angle export to them is required.

e2740: ETPU2: Watchdog Status Register (WDSR) may fail to update on channel timeout

Description: The Watchdog Status Register (WDSR) contains a single watchdog status bit for each of the 32 eTPU channels per engine. When this bit is set, it indicates that the corresponding channel encountered a watchdog timeout and was aborted. Under certain conditions the corresponding bit is not set due to a watchdog timeout, and therefore no indication is available as to which channel timed out. However, the global exception is indicated correctly on a per engine basis, and the correct exception is issued to the interrupt controller and may be serviced.

Workaround: The application software should treat any watchdog event as a global eTPU exception and handle it in the eTPU global exception handler. Additionally, during the global exception handler the application should check the WDSR and clear any bits that may be set by writing ‘1’ to that bit.

e5640: ETPU2: Watchdog timeout may fail in busy length mode

Description: When the Enhanced Time Processing Unit (eTPU) watchdog is programmed for busy length mode (eTPU Watchdog Timer Register (ETPU_WDTR) Watchdog Mode field (WDM) = 3), a watchdog timeout will not be detected if all of the conditions below are met:

1- The watchdog timeout occurs at the time slot transition, at the first instruction of a thread, or at the thread gap. (a thread gap is a 1 microcycle period between threads that service the same channel).

2- The thread has only one instruction.

3- The eTPU goes idle right after the timed-out thread, or after consecutive single-instruction threads.

Workaround: Insert a NOP instruction in threads which have only one instruction.

e8194: eTPU: EAC may detect double teeth in a single input transition

Description: The eTPU Enhanced Angle Counter (EAC) may detect two consecutive teeth in a single tooth input transition, when the microengine Tooth Program Register (TPR) register bit HOLD=1.

As a consequence of the input transition, the EAC:

(1) resets HOLD, which is correct, then

(2) detects another tooth (incorrect), so that if it is in normal mode, it goes to high-rate mode, and if it is in halt mode, it goes to normal mode.

No problem occurs if the EAC was in high-rate mode when HOLD=1.

The problem occurs only if both of these configuration conditions are true:

(a) EAC is configured with the eTPU Time Base Configuration Register (ETPU_TBCR_ENGx) Angle Mode Selection (AM) field = 2 (channel 1) or AM = 3 (channel 2).

(b) Channel filter configuration with etpu Engine Control Register (ETPU_ECR_ENGx) Channel Digital Filter Control (CDFC) field = 1 (bypass) or ETPU_ECR_ENGx Filter Clock Source Selection (FCSS) field = 1 (eTPU clock as filter clock).

Workaround: Configure the channel filters to use any mode except bypass (ETPU_ECR_ENGx field CDFC ! = 0b01) and configure ETPU_ECR_ENGx field FCSS = 0. (CDFC should be set to 0b00, 0b10, or 0b11.)

e8252: eTPU: ETPU Angle Counter (EAC) Tooth Program Register (TPR) register write may fail

Description: When the TPR is written with the Insert Physical Tooth (IPH) bit set to 1, and a physical tooth arrives at near the same time, the buffering of a second write to the TPR may fail, even if the required wait for one microcycle after the IPH write is observed.

Workaround: Wait at least two microcycles between consecutive writes to the TPR register, if the first write sets the IPH bit.

e9090: eTPU: Incorrect eTPU angle counter function under certain conditions

Description: The eTPU Angle Counter (EAC) can function incorrectly in some scenarios when all of the following conditions apply:

- EAC Tooth Program Register (TPR), Angle Ticks Number in the Current Tooth field (TICKS) = 0 [TPR.TICKS = 0]

and

- Tick Rate Register (TRR) and the eTPU Engine Time Base Configuration Register prescaler field [eTPU_TBR_TBCR_ENGn.TCRnP] satisfy the following condition:

(TRR – 1)*(TCRnP + 1) < 3, where TRR is the non-zero 15-bit integer part (the 15 most significant bits).

When the above conditions are met, three possible scenarios can cause the EAC to function incorrectly:

Scenario 1:

1. The EAC is in High Rate Mode, TRR = 1, and TPR Missing Tooth Counter field = 0 [TPR.MISSCNT = 0]

2. On an EAC transition from High Rate Mode to Normal mode, a positive value is written to TPR.MISSCNT

3. The first microcycle in Normal Mode coincides with a tick timing and either

a. A tooth does not arrive

or

b. A tooth arrives

Expected EAC behavior:

a. Nothing happens

or

b. The EAC transitions back to High Rate Mode

Actual (incorrect) EAC behavior:

a. The EAC transitions to Halt Mode, even though TPR.MISSCNT > 0

or

b. The EAC stays in Normal Mode, even though a tooth arrived before expected and TPR.MISSCNT > 0. The values of TPR.MISSCNT and TPR.LAST are reset, even though the EAC does not transition to High Rate Mode.

Scenario 2:

TCRnP = 0, TRR = 1 (integer part) and a new value is written to TPR.MISSCNT when the EAC transitions from High Rate Mode to Normal Mode. In this scenario, TPR.MISSCNT decrements on every microcycle, but the time the EAC takes to transition to Halt Mode is determined by the previous

TPR.MISSCNT value, so that one of the following unique situations is observed:

a. TPR.MISSCNT reaches zero, but the EAC transitions to Halt Mode only after a number of microcycles equal to the TPR.MISSCNT value before the write.

b. EAC transitions to Halt Mode with TPR.MISSCNT > 0 while, decrementing MISSCNT one more time. If TPR.MISSCNT > 1 during the mode transition, the EAC will stay in Halt mode with a non-zero value of TPR.MISSCNT.

Scenario 3:

1. The EAC transitions to Normal mode from High Rate or Halt Mode

2. The EAC enters Normal mode with TPR.LAST = 1

3. A tooth is received on the second or third microcycle after the EAC transitions to Normal mode. The tooth may be either a physical tooth or a dummy physical tooth generated by setting the Insert Physical Tooth (IPH) field of the TPR register (TPR.IPH = 1).

Observed result:

The EAC resets the values of TPR.LAST, TPR.IPH and the eTPU Engine Time Base2 (TCR2) register, but the EAC goes to Halt mode.

If a new TPR.TICKS value is written with the EAC in Normal mode, the value is effective after a new tooth is received in Halt mode, with TCR2 counting from 0.

Workaround: Limit the angle tick period to a minimum value that satisfies the condition (TRR – 1)* (TCRnP + 1) > 2, where TRR is the non-zero 15-bit integer part (the 15 most significant bits).

e9809: eTPU: MDU flags(Overflow/Carry) may be set incorrectly

Description: The MAC Carry (MC) & MAC Overflow (MV) flags can be incorrectly set on a MAC instruction if it is the first MDU operation in a thread and the last MDU operation in previous thread was aborted/terminated (thread ended before the operation finished).

Workaround: There are 2 workarounds:

(1) Do not abort/terminate a MDU operation

or

(2) Do not use a MAC instruction as the first MDU operation in a thread

e2157: FLASH: 4 wait states are required to run above 250MHz

Description: The access time for the flash requires 4 wait states when the core system clock (Core Fsys) is above 250MHz

Workaround: Program the Flash Bus Interface Platform Flash Configuration Register 1 (PFCR1) In the flash module A, Read Wait State Control field to 4 (FLASHA_PFCR1[RWSC]=4).

e2382: FLASH: Flash Array Integrity Check

Description: The Flash Array Integrity Check (AIC) which may be enabled during the flash user test (UTest) mode does not return the expected UMn[MISR] values for some flash PFCRPn[RWSC] read wait state configurations. For PFCRPn[RWSC] values of 3-6, the UMn[MISR] signature computation during AIC does not include the data read from the very last address in the selected address sequence and thus the UMn[MISR] value is not as expected. For PFCRPn[RWSC] values of 7, the UMn[MISR] signature computation during AIC will not be correct as well.

Workaround: The Flash Array Integrity Check is correct for PFCRPn[RWSC] values of 0-2. For PFCRPn[RWSC] values of 3-6, the expected UMn[MISR] values will not include the data read from the very last address and thus the value expected should be for the data read up to the 2nd-last address in the selected address sequence. For a PFCRPn[RWSC] value of 7, the Array Integrity Check should not be used at all.

e1312: FLASH: MCR[DONE] bit may be set before high voltage operation completes when executing a suspend sequence

Description: The program and erase sequence of the flash may be suspended to allow read and program access to the flash core. An suspend operation is initiated by setting the Erase Suspend (ESUS) bit or Program Suspend (PSUS) bit in the flash Module Configuration Register (MCR). Setting a suspend bit causes the flash module to start the sequence which places it in the suspended state. The user must then wait until the MCR[DONE] bit is set before a read or program to the flash is initiated, as the high voltage operation needs to be complete to avoid errors.

However, during normal read to the same partition, following a suspend sequence, (setting MCR bit and waiting for MCR[DONE] bit to be set) can result in read fails that will return multiple bit ECC errors. The error is due to the MCR[DONE] bit being set before the internal high voltage operation is complete.

Workaround: Because the MCR[DONE] flag can be set too soon, a delay needs to be inserted between setting the MCR[ESUS] or MCR[PSUS] and reading the same flash partition. The minimum duration of the delay should be 40us to guarantee correct operation. The Freescale flash programming driver includes this workaround.

e3659: FLASH: Resuming after a suspend during an Erase may prevent the erase from completing.

Description: If an erase suspend (including the flash put into sleep or disabled mode) is done on any block in the low Address Space (LAS) or the Mid-Address Space (MAS) except the 16 KB blocks, or if a suspend is done with multiple non-adjacent blocks (including the High Address Space [HAS]), the flash state machine may not set the FLASH_MCR[DONE] bit in the flash Module Control Register. This condition only occurs if the suspend occurs during certain internal flash erase operations. The likelihood of an issue occurring is reduced by limiting the frequency of suspending the erase operation.

Workaround: If the suspend feature (including disable and sleep modes) of the flash is used, then software should ensure that if the maximum time allowed for an erase operation occurs without a valid completion flag from the flash (FLASH_MCR[DONE] = 1), the software should abort the erase operation (by first clearing the Enable High Voltage (FLASH_MCR[EHV]) bit, then clearing the Erase read/Write bit (FLASH_MCR[ERS] bit) and the erase operation should be restarted.

Note: The cycle count of the sector is increased by this abort and restart operation.

e7322: FlexCAN: Bus Off Interrupt bit is erroneously asserted when soft reset is performed while FlexCAN is in Bus Off state

Description: Under normal operation, when FlexCAN enters in Bus Off state, a Bus Off Interrupt is issued to the CPU if the Bus Off Mask bit (CTRL[BOFF_MSK]) in the Control Register is set. In consequence, the CPU services the interrupt and clears the ESR[BOFF_INT] flag in the Error and Status Register to turn off the Bus Off Interrupt.

In continuation, if the CPU performs a soft reset after servicing the bus off interrupt request, by either requesting a global soft reset or by asserting the MCR[SOFT_RST] bit in the Module Configuration Register, once MCR[SOFT_RST] bit transitions from 1 to 0 to acknowledge the soft reset completion, the ESR[BOFF_INT] flag (and therefore the Bus Off Interrupt) is re-asserted.

The defect under consideration is the erroneous value of Bus Off flag after soft reset under the scenario described in the previous paragraph.

The Fault Confinement State (ESR[FLT_CONF] bit field in the Error and Status Register) changes from 0b11 to 0b00 by the soft reset, but gets back to 0b11 again for a short period, resuming after certain time to the expected Error Active state (0b00). However, this late correct state does not reflect the correct ESR[BOFF_INT] flag which stays in a wrong value and in consequence may trigger a new interrupt service.

Workaround: To prevent the occurrence of the erroneous Bus Off flag (and eventual Bus Off Interrupt) the following soft reset procedure must be used:

1. Clear CTRL[BOFF_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).

2. Set MCR[SOFT_RST] bit in the Module Configuration Register.

3. Poll MCR[SOFT_RST] bit in the Module Configuration Register until this bit is cleared.

4. Wait for 4 peripheral clocks.

5. Poll ESR[FLTCONF] bit in the Error and Status Register until this field is equal to 0b00.

6. Write “1” to clear the ESR[BOFF_INT] bit in the Error and Status Register.

7. Set CTRL[BOFF_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).

e3407: FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1

Description: FlexCAN does not transmit an expected message when the same node detects an incoming Remote Request message asking for any remote answer.

The issue happens when two specific conditions occur:

1) The Message Buffer (MB) configured for remote answer (with code “a”) is the last MB. The last MB is specified by Maximum MB field in the Module Configuration Register (MCR[MAXMB]).

2) The incoming Remote Request message does not match its ID against the last MB ID.

While an incoming Remote Request message is being received, the FlexCAN also scans the transmit (Tx) MBs to select the one with the higher priority for the next bus arbitration. It is expected that by the Intermission field it ends up with a selected candidate (winner). The coincidence of conditions (1) and (2) above creates an internal corner case that cancels the Tx winner and therefore no message will be selected for transmission in the next frame. This gives the appearance that the FlexCAN transmitter is stalled or “stops transmitting”.

The problem can be detectable only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is NO ISSUE if any of the conditions below holds:

a) The incoming message matches the remote answer MB with code “a”.

b) The MB configured as remote answer with code “a” is not the last one.

c) Any MB (despite of being Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.

d) A new incoming message sent by any external node starts just after the Intermission field.

Workaround: Do not configure the last MB as a Remote Answer (with code “a”).

e2424: FlexCAN: switching CAN protocol interface (CPI) to system clock has very small chance of causing the CPI to enter an indeterminate state

Description: The reset value for the clock source of the CAN protocol interface (CPI) is the oscillator clock. If the CPI clock source is switched to the system clock while the FlexCAN is not in freeze mode, then the CPI has a very small chance of entering an indeterminate state.

Workaround: Switch the clock source while the FlexCAN is in a halted state by setting HALT bit in the FlexCAN Module Configuration Register (CANx_MCR[HALT]=1). If the write to the CAN Control Register to change the clock source (CANx_CR[CLK_SRC]=1) is done in the same oscillator clock period as the write to CANx_MCR[HALT], then chance of the CPI entering an indeterminate state is extremely small. If those writes are done on different oscillator clock periods, then the corruption is impossible. Even if the writes happen back-to-back, as long as the system clock to oscillator clock frequency ratio is less than three, then the writes will happen on different oscillator clock periods.

e1364: FlexRay : Message Buffer Slot Status corrupted after system memory access timeout or illegal address access

Description: If the system memory read access that retrieves the first message buffer header data from a FlexRay transmit buffer fails due to a system memory access timeout or illegal address access, it is possible that the slot status information for the previous slot is written into the currently used transmit message buffer. In this case, the slot status information is not written into the message buffer assigned to the last slot.

Thus, both the message buffer assigned to the last slot, and the currently used transmit message buffer contain incorrect slot status information.

However, if this occurs, either the System Bus Communication Failure Error Flag (SBCF_EF) or the Illegal System Bus Address Error Flag (ILSA_EF) will be set in the Controller Host Interface Error Flag Register (CHIERFR).

Workaround: The FlexRay module and the system memory subsystem should be configured to avoid the occurrence of system memory access timeouts and illegal address accesses.

In case that one of the error flags CHIERFR[SBCF_EF] or CHIERFR[ILSA_EF] is set, the application should not use the slot status information of the message buffers.

e1369: FlexRay : Message Buffer Status, Slot Status, and Data not updated after system memory access timeout or illegal address access

Description: If a message buffer is assigned to the last slot in a FlexyRay communication cycle and a system memory access timeout or illegal address access occurs during the system memory access in this slot, it is possible that for all future communication 1) no slot status information will be written, 2) the message buffer status will not be updated, and 3) no message frames will be received. If this happens, several message buffers can never be locked by the application.

However, if this occurs, either the System Bus Communication Failure Error Flag (SBCF_EF) or the Illegal System Bus Address Error Flag (ILSA_EF) will be set in the Controller Host Interface Error Flag Register (CHIERFR).

Workaround: The FlexRay module and the system memory subsystem should be configured to avoid the occurrence of system memory access timeouts and illegal address accesses.

In case that one of the error flags CHIERFR[SBCF_EF] or CHIERFR[ILSA_EF] is set, the application should stop the FlexRay controller via a FREEZE or HALT command and subsequently restart the controller.

e2302: FlexRay: Message Buffer can not be disabled and not locked after CHI command FREEZE

Description: If a complete message was transmitted from a transmit message buffer or received into a message buffer and the controller host interface (CHI) command FREEZE is issued by the application before the end of the current slot, then this message buffer can not be disabled and locked until the module has entered the protocol state normal active.

Consequently, this message buffer can not be disabled and locked by the application in the protocol config state, which prevents the application from clearing the commit bit CMT and the module from clearing the status bits. The configuration bits in the Message Buffer Configuration, Control, Status Registers (MBCCSRn) and the message buffer configuration registers MBCCFRn, MBFIDRn, and MBIDXRn are not affected.

At most one message buffer per channel is affected.

Workaround: There are two types of workaround.

1) The application should not send the CHI command FREEZE and use the CHI command HALT instead.

2) Before sending the CHI command FREEZE the application should repeatedly try to disable all message buffers until all message buffers are disabled. This maximum duration of this task is three static or three dynamic slots.

e6726: NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled

Description: The Nexus auxiliary message clock (MCKO) may be gated one clock period early when the MCKO frequency is programmed as SYS_CLK/8 in the Nexus Port Controller Port Configuration Register (NPC_PCR[MCKO_DIV]=111) and the MCKO gating function is enabled (NPC_PCR[MCKO_GT]=1). In this case, the last MCKO received by the tool prior to the gating will correspond to the END_MESSAGE state. The tool will not receive an MCKO to indicate the transition to the IDLE state, even though the NPC will transition to the IDLE state internally. Upon re-enabling of MCKO, the first MCKO edge will drive the Message Start/End Output (MSEO=11) and move the tool's state to IDLE.

Workaround: Expect to receive the MCKO edge corresponding to the IDLE state upon re-enabling of MCKO after MCKO has been gated.

e3553: NXFR: Flexray databus translates into unexpected data format on the Nexus interface

Description: The data format for Nexus Flexray messages is in little-endian (least significant byte first) order. This is not currently documented and may be unexpected for users of Power Architecture devices.

For example, in the case of a Flexray System Memory write within the address space determined by Data Trace Start and Data Trace End Addresses (DTSAx/DTEAx) with the data: 0x1122, the Flexray Nexus interface generates Data Trace Messages (DTM) containing the data: 0x2211.

Workaround: The user must be aware of the data format. This will be documented in a future release of the device reference manual.

e1279: NZ7C3:Core Nexus Read/Write Access registers cleared by system reset

Description: The e200z7 Nexus Read/Write Access registers are cleared when system reset is asserted. This affects the Read/Write Access Data register (RWD), the Read/Write Access Address register (RWA), and the Read/write Access Control/Status register (RWCS).

Workaround: Do not expect RWD, RWCS, and RWA to retain values after reset. After reset reload any values required for a transfer.

e7120: NZxC3: DQTAG implemented as variable length field in DQM message

Description: The e200zx core implements the Data Tag (DQTAG) field of the Nexus Data Acquisition Message (DQM) as a variable length packet instead of an 8-bit fixed length packet. This may result in an extra clock (“beat”) in the DQM trace message depending on the Nexus port width selected for the device.

Workaround: Tools should decode the DQTAG field as a variable length packet instead of a fixed length packet.

e3377: Pad Ring:Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge

Description: The Nexus Output pins (Message Data outputs 0:15 [MDO] and Message Start/End outputs 0:1 [MSEO]) may drive an unknown value (high or low) immediately after power up but before the 1st clock edge propagates through the device (instead of being weakly pulled low). This may cause high currents if the pins are tied directly to a supply/ground or any low resistance driver (when used as a general purpose input [GPI] in the application).

Workaround: 1. Do not tie the Nexus output pins directly to ground or a power supply.

2. If these pins are used as GPI, limit the current to the ability of the regulator supply to guarantee correct start up of the power supply. Each pin may draw upwards of 150mA.

If not used, the pins may be left unconnected.

e9109: PAD_RING: Output pulse may occur on analog inputs during power on reset

Description: If the 1.2V core supply voltage (VDD) power supply is the last power supply to ramp into operating voltage (after the Analog to Digital [ADC] converter [VDDA] and other supplies [VDDEn, VDDEHn] are powered) or is the first supply to go out of the operating specified voltage, it is possible that an output pulse will occur on an analog pin (ANx) of the device. This pulse could be a maximum of 3.8 volts (unloaded). If the ANx pin is grounded, a current of up to 2 mA could be seen on the ANx pin.

This pulse could occur on up to 2 pins per enhanced Queued Analog to Digital Converter (eQADC) or 2 pairs of differential analog inputs. The pulse occurs if one of the ADC channels is selected to be connected to one of the two ADCs (ADC_0 or ADC_1) in the eQADC.

The two ANx pins selected could be random over the complete specified device processing, temperature and voltage ranges, and VDD voltage ramp speed. The same channel could be selected to both ADCs for a total current of 4 mA (unloaded voltage remains a maximum of 3.8V).

On devices with ANx channels shared between multiple eQADC modules (currently, the maximum number of eQADC modules on a device is 2), it is theoretically possible for a channel to be selected to all ADC's (up to 4) on the device for a total current of 8 mA.

The pulse may start (rising edge) when VDD reaches 700 mv and go to a high impedance when VDD reaches 1.166 volts. The pulse width could be the time that VDD ramps between these voltages.

Workaround: Design circuitry connected to the analog pins to withstand a possibility of up to 4 mA (or 8 mA for shared analog pins) during power up and power down.

e2696: PBRIDGE: Write buffer may cause overflow/underflow of DMA transfers

Description: Peripheral-paced DMA transfers are controlled by a hardware handshake protocol: when the peripheral requires a data transfer, it asserts a request to the DMA. The DMA recognizes the request, activates the corresponding channel and performs the data transfer, reading from the source and writing to the destination. As the write is being processed, the DMA sends an acknowledge back to the peripheral so it can negate its request.

If buffered writes are enabled in the PBRIDGE, there are certain conditions where the DMA's acknowledge is asserted before the actual write operation to the peripheral has occurred. The net effect is the DMA request is not negated properly, causing the DMA to reactivate the channel, overwriting the last transferred data value before it is transferred to the peripheral.

Workaround: Do not enable buffered writes for the eDMA in the Peripheral Bridge Master Privilege Control registers (PBRIDGE_A_MPCR and PBRIDGE_B_MPCR), by leaving the MBW4 (eDMA A master) and MBW5 (eDMA B master) bits cleared. This is the default state of these bits, which disables buffered writes to the peripherals from both eDMA masters.

e11321: PIT_RTI: Generates false RTI interrupt on re-enabling

Description: A false Real-Time Interrupt (RTI) may be observed when the RTI module is re-enabled if, after servicing an RTI interrupt (by clearing TFLGn[TIF]), the clocks to the RTI module are disabled.

This occurs only if the RTI module clock is disabled within four RTI clock cycles of an RTI Interrupt being cleared.

Workaround: Option 1: The user should check the RTI interrupt flag,TFLGn[TIF] before servicing the interrupt, this flag won't be set for the false/spurious interrupts.

Option 2: Ensure that the module clock to the RTI module is not disabled within four RTI clock cycles after servicing an RTI interrupt. Consult the chip-specific documentation to determine the clock period of the RTI module and implement a time delay of at least five times this period before disabling the RTI module clock.

e2996: PIT_RTI: RTI timer corruption when debugging

Description: In rare cases, due to a synchronization issue, the Real-Time Interrupt (RTI) timer value may become corrupted when a breakpoint occurs and the freeze bit is set to pause the timers in debug mode (PIT_RTI_MCR[FRZ]=0b1).

None of the other timers in the PIT_RTI module are affected. During normal operation (without debugger attached) there is no impact to the application.

Workaround: When debugging code utilizing the RTI, do not depend on the value of the RTI timer being correct.

e10783: PMC: BGTS (BandGap Temperature Sensor) may not assert at high temperature

Description: BGTS (BandGap Temperature Sensor) may not assert at high temperature.

Workaround: Do not rely on PMC_CR[BGTS] for high temperature indication. If a secondary high temperature indication is required for the application, the application must provide another temperature sensor outside the device (typically inside the external power management IC).

e2322: PMC: LVREH/LVREA/LVRE50 may exit LVI triggered reset with LVI condition still existing

Description: After asserting the Low Voltage Reset Enables PMC_CFGR[LVREH], PMC_CFGR[LVREA], PMC_CFGR[LVRE50] bits in the PMC, if the voltage ramps down below the LVI voltage on VDDEHx, VDDA or VDDREG respectively, the part will go to a short power on reset (POR). After the reset counter has expired, the device will go into normal operation, even though one or more of the affected supplies may still be below the specified LVI voltage.

Workaround: The part will recover into normal operation, however software should check the status of the LVIs for those segments that are required for further operation.

e1419: SIU: Reverting ENGCLK source to the system clock has a very small chance of causing the ENGCLK generator to enter an indeterminate state

Description: The reset value for the Engineering Clock (ENGCLK) source is the system clock. If the clock source is switched to the external clock (buffered crystal frequency or external clock input) by setting Engineering Clock Source Select bit in the External Clock Control Register (SIU_ECCR[ECSS]=1) and then reverted to the system clock, then the ENGCLK generator has a very small chance of entering an indeterminate state.

Workaround: Do not change the ENGCLK source back to the System clock after changing it to the external clock.

e9658: SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event

Description: In the Serial Peripheral Interface (SPI) module, when both the receive FIFO and shift register are full (Receive FIFO Overflow Flag bit in Status Register is set (SR [RFOF] = 0b1)) and then the Clear Rx FIFO bit in Module Configuration Register (MCR [CLR_RXF]) is asserted to clear the receive FIFO, shift register data is sometimes loaded into the receive FIFO after the clear operation completes.

Workaround: 1. Avoid a receive FIFO overflow condition (SR[RFOF] should never be 0b1). To do this, monitor the RX FIFO Counter field of the Status Register (SR[RXCTR]) which indicates the number of entries in receive FIFO and clear before the counter equals the FIFO depth.

2. Alternatively, after every receive FIFO clear operation (MCR[CLR_RXF] = 0b1) following a receive FIFO overflow (SR[RFOF] = 0b1) scenario, perform a single read from receive FIFO and discard the read data.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.