molex®
one company › a world of innovation

SST Profibus Scanner Module

Firmware Reference Guide

Document Edition: 2.1

Document #: 717-0032

Document Edition: 2.1

Date: February 15, 2008

This document applies to the SST Profibus Scanner Module.

At Woodhead, we strive to ensure accuracy in our documentation. However, due to rapidly evolving products, software or hardware changes occasionally may not be reflected in our documents. If you notice any inaccuracies, please contact us (see Appendix A of this document).

Written and designed at Woodhead Software & Electronics, 50 Northland Road, Waterloo, Ontario, Canada N2V 1N3.

Hardcopies are not controlled.

Contents

Preface

Preface Sections:

- Purpose of this Guide
- Conventions

Purpose of this Guide

This document is a reference guide for the Profibus Scanner Module firmware (PFBSCAN), an application module for the SST-PFB3 family of interface cards. For DLL-related information, refer to The SST Profibus Scanner Module DLL Reference Guide.

Conventions

This guide uses stylistic conventions, special terms and special notation to help enhance your understanding.

Style

The following stylistic conventions are used throughout this guide:

Bold	indicates field names, button names, tab names, and options or selections
Underlining	indicates a hyperlink
Italics	indicates keywords (indexed) or instances of new terms and/or specialized words that need emphasis
CAPS	indicates a specific key selection, such as ENTER, TAB, CTRL, ALT, DELETE
`Code Font`	indicates command line entries or text that you'd type into a field
">" delimiter	indicates how to navigate through a hierarchy of menu selections/options

Special Terms

The following special terms are used throughout this guide:

Card	The SST-PB3-PCI-2 network interface card
Channel	Profibus network interface on the card
DWORD	Little Endian 32-bit value, unless otherwise stated
Firmware Module	The embedded software module that gets loaded to the card's memory and runs on the card. This is the operating system of the card, enabling it to respond to commands from the host and manage network communications.
Host	The computer system in which the card is installed
Outputs	Data that originate in the process controller and are transmitted to the field devices
PFBSCAN	The Profibus Scanner Module, which encapsulates both the pfb3.ss3 and pfb3-2.ss3 firmware modules
Shared Memory	On-card memory mapped to the host computer and shared between the firmware module and the host application
.ss3	An encoded firmware module for the card
WORD	Little Endian 16-bit value, unless otherwise stated

Special Notation

The following special notation is used throughout this guide:

Note

A note provides additional information, emphasizes a point, or gives a tip for easier operation. Notes are accompanied by the symbol shown, and follow the text to which they refer.

1 Introduction

Chapter Sections:

- PFBSCAN Overview
- Terminology

1.1 PFBSCAN Overview

PFBSCAN refers to two different firmware modules: pfb3.ss3, and pfb3-2.ss3. Pfb3-2.ss3 will be used on the new expanded I/O Profibus interface card, the SST-PFB3-PCI-2. Pfb3-2.ss3 has the same characteristics and performance as the original pfb3.ss3 module, except for where specified in this document.

General Module Capabilities

The Profibus Scanner Module allows an SST-PFB3 interface card to:

- Function as a DP Master
- Function as a DP Slave
- Send and receive FDL (Layer 2) messages
- Support simultaneous operation in all of the above modes
- Support the standard Profibus baud rates of 9.6K, 19.2K, 31.25K, 45.45K, 93.75K, 187.5K, 500K, 1.5M, 3M, 6M and 12M

Additional Module Capabilities

- Maintains an active station list for the network and can update a list of passive stations, on demand
- Maintains diagnostic counters for all the card operations
- Maintains an event queue where the host can be notified of various events occurring in the DP master, DP slave, and so on. These events can be disabled or enabled independently.

Card as Master

- Can control up to 125 slaves
- Supports up to 244 bytes of input data per slave, 16 Kbytes total input data for all slaves for the original cards, and 32 Kbytes for the expanded I/O card
- Supports up to 244 bytes of output data per slave, 16 Kbytes total output data for all slaves for the original cards, and 32 Kbytes for the expanded I/O card
- Can generate an event and an optional interrupt on:
    - Receive data change from any slave
    - Updates from any slave
    - Scan done
    - Transition to error
    - Transition to OK

Card as Slave

- Supports up to 244 bytes of input data and up to 244 bytes of output data
- Can generate an event and an optional interrupt on:
    - Received data change
    - Slave update by master
    - State change to run
    - State change to stop
    - Transition to slave error
    - Transition to slave OK

Layer 2 (FDL) Interface

- Allows configuration of up to 64 SAPs
- Has an optional timeout watchdog on any SAP
- Supports up to 128 message blocks at one time
- Can generate an event and an optional interrupt on:
    - Received data change
    - Message or SAP update
    - Message or SAP error

2

PFBSCAN.SS3 Operation

Chapter Sections:

- Module Overview
- PROFI_USR Structure
- Host Interface Description

2.1 Module Overview

The host uses a memory-mapped interface to access the PFBSCAN module. The user interface for PFBSCAN is completely open. As there is no need to link to any special libraries, the module is independent of the compiler and operating system.

The Profibus network data is divided in 16K logical pages. On some hardware platforms, the Profibus network data is larger than what can be directly mapped into the memory of the host computer at one time. On such systems, one 16K page is mapped into the host memory space at a time. On other hardware platforms, the entire amount of Profibus network data can be directly mapped into host memory. For details on how the Profibus network data is mapped into a particular hardware platform, refer to the relevant Hardware Reference Guide.

Note

Regardless of the hardware platform that the PFBSCAN Module is running on, memory is organized into 16K bytes pages.

Logical page 2 (offset 8000h) always contains the PROFI_USR structure. Other pages of card memory contain the DP master input data, DP master output data, DP master control blocks, and so on. Locations in the PROFI_USR structure contain the page numbers for these pages. The following sections describe the layout of this structure.

2.2 PROFI_USR Structure

The host system interfaces with the PROFI_USR structure for configuration and runtime control of the Profibus system. The following table describes the PROFI_USR structure in detail.

Group	Size	Name	Offset
Command register	UINT8	pfbCommand	8000h
Status register	UINT8	pfbStatus	8001h
ID registers 8002h-8007h	UINT16	pfbCardId	8002h
	UINT16	pfbModId	8004h
	UINT16	pfbModVer	8006h
Event and interrupt control 8008h-800Dh	UINT16	pfbEvtEna	8008h
	UINT16	pfbIntEna	800Ah
	UINT8	pfbEvtHead	800Ch
	UINT8	pfbEvtTail	800Dh
		Reserved	800Eh-8025h
ASPC2 profibus controller basic parameters 800Eh-8013h	UINT8	pfbStnAddr	800Eh
	UINT8	pfbHiStnAddr	800Fh
	UINT8	pfbActive	8010h
	UINT8	pfbBaud	8011h
	UINT16	pfbOptions	8012h
ASPC2 Profibus controller bus parameters (PFB will put in defaults if left unchanged) 8014h-8021h	UINT32	pfbTokRotTime	8014h
	UINT16	pfbSlotTime	8018h
	UINT16	pfbIdleTime1	801Ah
	UINT16	pfbIdleTime2	801Ch
	UINT16	pfbReadyTime	801Eh
	UINT8	pfbGapUpdFact	8020h
	UINT8	pfbQuiTime	8021h
ASPC2 Profibus controller error handling Parameters 8022h-8025h	UINT8	pfbTokRetryLimit	8022h
	UINT8	pfbMsgRetryLimit	8023h
	UINT8	pfbTokErrLimit	8024h
	UINT8	pfbRespErrLimit	8025h
Host trigger queue control 8026h-8027h	UINT8	pfbTrgHead	8026h
	UINT8	pfbTrgTail	8027h
DP Master global status and control table 8028h-802Fh	UINT8	pfbMasCntrlCfg	8028h
	UINT8	pfbMasSts	8029h
	UINT8	pfbMasCntrlPage	802Ah
	UINT8	pfbMasRxPage	802Bh
	UINT8	pfbMasTxPage	802Ch
	UINT8	pfbMasCoherFlags	802Dh
	UINT16	pfbMasMaxIoCycTme	802Eh
		Reserved	8030h-803Fh

Group	Size	Name	Offset
DP Master sync and freeze support 8037h-8039h	UINT16	pfbMasSyncFrzCmd	8037h
	UINT8	pfbMasScanEmpty	8039h
Diagnostic Counters counters and control 8040h	UINT8	pfbInitCtrs	8040h
General statistics 8041h-804Fh	UINT8	errLanOffline	8041h
	UINT16	diagConf	8042h
	UINT16	diagInd	8044h
	UINT16	errNotOk	8046h
	UINT32	diagTokHldTime	8048h
	UINT32	diagMinTokHldTime	804Ch
DP Master Block statistics 8050h-805Bh	UINT16	diagMasterUpdate	8050h
	UINT8	errMasErr	8052h
	UINT8	errReConfig	8053h
	UINT32	diagMasScanTime	8054h
	UINT32	diagMasMaxScanTime	8058h
DP Slave statistics 805Ch-805Fh	UINT16	diagSlaveUpdate	805Ch
	UINT8	errSlvErr	805Eh
	UINT8	errSlvTout	805Fh
Layer 2 Message Statistics 8060h-8063h	UINT16	diagLay2MsgOk	8060h
	UINT8	errLay2MsgNotOk	8062h
	UINT8	errLay2MsgProcOvrn	8063h
Layer 2 SAP Statistics 8064h-8067h	UINT16	diagLay2SapOk	8064h
	UINT8	errLay2SapNotOk	8066h
	UINT8	errLay2SapTout	8067h
		Reserved	8068h-806F
ASPC2 Profibus controller Statistics 8070h-807Ah	UINT8	errInvReqLen	8070h
	UINT8	errFifo	8071h
	UINT8	errRxOverun	8072h
	UINT8	errDblTok	8073h
	UINT8	errRespErr	8074h
	UINT8	errSyniErr	8075h
	UINT8	errNetTout	8076h
	UINT8	errHsa	8077h
	UINT8	errStn	8078h
	UINT8	errPasTok	8079h
	UINT8	errLasBad	807Ah
	UINT8	errInternal	807Bh
	UINT8	errArg	807Ch
	UINT8	errEventOverun	807Dh
Active station list 807Eh-80FFh	UINT8	pfbAckLasChnge	807Eh
	UINT8	pfbUpdPasv	807Fh
	UINT8[128]	pfbActStnList[128]	8080-80FFh

Group	Size	Name	Offset
Master configuration parameters 8100h-8103h	UINT8	pfbBinCfgPage	8100h
	UINT8	pfbBinCfgOfs	8101h
	UINT16	pfbBinCfgLen	8102h
Master Class 2 reception poll timeout	UINT16	pfbMc2PollTout	8104h
Master minimum I/O cycle time in microseconds	UINT16	pfbMasMinIoCycTme	8106h
Watchdog parameters 8108h-810Bh	UINT16	pfbWdTime	8108h
	UINT16	pfbWdKick	810Ah
Global event register	UINT8	pfbMasGlbEvt	810Ch
Number of DP Master blocks configured	UINT8	pfbMasNumBlks	810Dh
Free extended area offset	UINT16	pfbMasCntrlExtFree	810Eh
ID response text	UINT8[112]	pfbLocIdUsrStr[112]	8110h-817Fh
DP slave control, status and data 8180h-83FFh	UINT16	slvCntCfg	8180h
	UINT8	slvStatus	8182h
	UINT8	slvError	8183h
	UINT8	slvEvent	8184h
	UINT8	slvDiagEvent	8185h
		Reserved	8186h-8187h
	UINT8	slvRxDataLen	8188h
	UINT8	slvReqRxDataLen	8189h
		Reserved	818Ah-818Bh
	UINT8	slvTxDataLen	818Ch
	UINT8	slvReqTxDataLen	818Dh
	UINT8	slvChkLen	818Eh
	UINT8	slvParmLen	818Fh
	UINT8[2]	slvGlbCntrl[2]	8190h
		Reserved	8191h-819Fh
	UINT8	slvSts1	81A0h
	UINT8	slvSts2	81A1h
	UINT8	slvSts3	81A2h
	UINT8	slvMasStn	81A3h
	UINT8	slvID_hi	81A4h
	UINT8	slvID_lo	81A5h
	UINT8	slvDiagLen	81A6h
	UINT8[25]	slvDiag[25]	81A7h
	UINT8	slvMasSts	81C0h
	UINT8	slvWdFact1	81C1h
	UINT8	slvWdFact2	81C2h
	UINT8	slvReadyTime	81C3h
	UINT8	slvMasID_hi	81C4h
	UINT8	slvMasID_lo	81C5h

Group	Size	Name	Offset
	UINT8	slvGrpId	81C6h
	UINT8[25]	slvParm[25]	81C7h
	UINT8[32]	slvChk[32]	81E0h
	UINT8[256]	slvRxData[256]	8200h
	UINT8[256]	slvTxData[256]	8300h
Event queue	UINT16[256]	pfbEventQueue[256]	8400h
Trigger queue	UINT16[256]	pfbTrigQueue[256]	8600h
ID response fields 8800h-88F9h	UINT8	pfbIdReq	8800h
	UINT8	pfbIdStn	8801h
	UINT8	pfbIdRspSts	8802h
	UINT8	pfbIdRspLen	8803h
	UINT8[4]	pfbIdFldLen[4]	8804h
	UINT8[242]	pfbIdText[242]	8808h
		Reserved	88FAh-88FFh
Master Status/Block table	UINT16[128]	pfbMasStsTab[128]	8900h
Master Class 2 control status and data	UINT8	mc2CntCfg	8A00h
	UINT8	mc2State	8A01h
	UINT8	mc2Status	8A02h
	UINT8	mc2Error	8A03h
	UINT8	mc2Event	8A04h
	UINT8	mc2RspSts	8A05h
	UINT8	mc2TxLen	8A06h
	UINT8	mc2RxLen	8A07h
	UINT16	mc2PollTime	8A08h
	UINT16	mc2PollLimit	8A0Ah
	UINT8	mc2DstStn	8A0Ch
		Reserved	8A0Dh-8A0Fh
	UINT8[244]	mc2TxData[244]	8A10h
	UINT8[244]	mc2RxData[244]	8B04h
		Reserved	8BF8h-8FDFh
Layer 2 control, status and data	UINT16	lay2Cntrl	8FE0h
	UINT16	lay2Status	8FE2h
		Reserved	8FEAh-8FFFh
	LAY2M_CNTRL	lay2mCntrl[128]	9000h
	LAY2S_CNTRL	lay2sCntrl[64]	A000h

2.2.1 Command Register

The host uses the command register, pfbCommand, to issue commands to the PFBSCAN module.

The module accepts commands only when it is offline. The exception to this is the command to go offline. If you issue any other command while the module is online, the module sets the command register to CMD_ERROR and the status register to STS_BAD_CMD, and then it goes offline.

The module sets the command register to the following values to indicate the module state.

Command	Value	Description
CMD_OFF	E0h	Module is offline ready to take commands
CMD_ON	E1h	Module is online ready to take offline command
CMD_COM_CFG	E2h	On versions of the module with a serial port
CMD_ERROR	EFh	Module is in error, status contains error code

By writing to the command register, the host can issue the following commands to the module.

Command	Value	Description
CMD_GO_ON	01h	Module should now go online
CMD_GO_OFF	02h	Module should now go offline, or abort config
CMD_REINIT	03h	Module should reinitialize memory and all parameters
CMD_CLR_CFG_BUF	04h	Module should clear pfbBinCfgPage
CMD_CHK_NET_CFG	05h	Module should check network parameters, and assign defaults
CMD_CPY_MAS_CFG	06h	Module should copy a page of cfg from pfbBinCfgPage
CMD_AUTO_BAUD_DET	07h	Module should do automatic baud detect
CMD_MAS_ASIGN_ADDR	08h	Assign and fill in data addr (page/offset) for dp master
CMD_CFG_ABF_SHRAM	0Fh	Configure dp master from SST config tool binary previously copied
CMD_CFG_2BF_SHRAM	10h	Configure dp master from ET200 binary previously copied
CMD_CFG_FROM_FLASH	11h	Configure module from configuration in flash
CMD_FLASH_GO_ON	14h	Configure module from flash, then go online
CMD_PGM_TO_FLASH	21h	Burn module configuration into flash
CMD_CFG_ABF_FLASH	30h	Configure module with config.bss file in flash
CMD_CFG_2BF_FLASH	31h	Configure module with config.2bf file in flash
CMD_CFG_ABF_FLASH_ GO_ON	32h	Configure module with config.bss file in flash, then go online
CMD_CFG_2BF_FLASH_ GO_ON	33h	Configure module with config.2bf file in flash, then go online

When the module successfully executes a command, it sets the command register to CMD_OFF. The exception is the CMD_GO_ON command, when it sets the command register to CMD_ON. If the command fails, the module sets the command register to CMD_ERROR.

If an error occurs while a command is executing, the module writes the error code into the status register (refer to Section 2.2.2, Status Register). If the error is a non-fatal, the host can clear the error by issuing the following command.

Command	Value	Description
CMD_ERR_ACK	FFh	Acknowledge offline error (online errors can't be acked)

Issuing CMD_ERR_ACK returns the module to the startup state. The host must then reconfigure parameters. If the module returns a fatal error, the host must reinitialize the card.

2.2.2 Status Register

When the host tells the module to execute a command, the module returns the result of the command in the status register, pfbStatus.

If the command executes successfully, the value in the status register is STS_NO_ERROR (0).

If the module encounters a problem executing a command, it writes the value CMD_ERROR (EFh) in the command register and returns a value in the status register that indicates the nature of the error.

If the host issues an invalid command, the module sets the status register to STS_BAD_CMD (01h).

2.2.2.1 Network Parameter Errors

The following status errors may occur when the host sets the network parameters. Refer to Section 2.2.6, ASPC2 Profibus Controller Basic Parameters, for allowed network parameter values.

All of the network parameter errors occur when the module executes CMD_GO_ON or CMD_CHK_NET_CFG.

The status register displays the following values when the network parameters are configured with invalid data.

Status	Value	Invalid Parameter	Reference
STS_BAD_BAUD	02h	pfbBaud	2.2.6.1, Basic Parameters
STS_BAD_STN_ADR	03h	pfbStnAddr	2.2.6.1, Basic Parameters
STS_BAD_HI_STN_ADR	04h	pfbHiStnAddr	2.2.6.1, Basic Parameters
STS_BAD_TOK_ROT	05h	pfbTokRotTime	2.2.6.2, Bus Parameters
STS_BAD_SLOT_TME	06h	pfbSlotTime	2.2.6.2, Bus Parameters
STS_BAD_IDLE_1	07h	pfbIdleTime1	2.2.6.2, Bus Parameters
STS_BAD_IDLE_2	08h	pfbIdleTime2	2.2.6.2, Bus Parameters
STS_BAD_RDY_TME	09h	pfbReadyTime	2.2.6.2, Bus Parameters
STS_BAD_QUI_TME	0Ah	pfbQuiTime	2.2.6.2, Bus Parameters
STS_BAD_GAP_UPD	0Bh	pfbGapUpdFact	2.2.6.2, Bus Parameters
STS_BAD_TOK_RETRY	0Ch	pfbTokRetryLimit	2.2.6.2, Bus Parameters
STS_BAD_MSG_RETRY	0Dh	pfbMsgRetryLimit	2.2.6.2, Bus Parameters
STS_BAD_TOK_ERR_LIM	0Eh	pfbTokErrLimit	2.2.6.2, Bus Parameters
STS_BAD_RSP_ERR_LIM	0Fh	pfbRespErrLimit	2.2.6.2, Bus Parameters

2.2.2.1.1 Autobaud Detect Error

The following error occurs when the host tries to detect the network baud rate with the CMD_AUTO_BAUD_DETECT command and the command fails.

Status	Value	Description	Reference
STS_BAUD_DET_ERROR	10h	Error when trying to autobaud detect	2.2.6.1, Basic Parameters

2.2.2.1.2 Configuration Errors

The following errors occur when the host is configuring the module as a DP master using a binary file generated by the SST Config tool. The status register will show the following values for configuration errors.

Error Message	Value	Description	Applicable Commands
STS_CFG_BAD_CHK_PATTERN	20h	The version code, abfCtrlWareVer, in the binary file was corrupted	CMD_CFG_FROM_FLASH, CMD_CFG_FROM_FLASH_GO_ON, CMD_CFG_ABF_SHRAM
STS_CFG_BIN_TOO_SHORT	21h	The binary file is shorter than 100 bytes	CMD_CFG_ABF_SHRAM, CMD_CFG_2BF_SHRAM
STS_CFG_BIN_TOO_LONG	22h	The binary file is longer than the 256 Kbytes	CMD_CFG_ABF_SHRAM, CMD_CFG_2BF_SHRAM
STS_CFG_BAD_CHKSUM	23h	The binary file has a bad checksum	CMD_CFG_FROM_FLASH, CMD_CFG_FROM_FLASH_GO_ON
STS_CFG_INVALID_CPU_HDR	24h	The binary file has an invalid CPU header	Not used
STS_CFG_INVALID_SLV_REC_TYP	25h	The binary file has a slave with and invalid record type	CMD_CFG_2BF_SHRAM
STS_CFG_RX_OVERFLOW	26h	There is more than 4000h bytes of RX data configured	CMD_MAS_ASIGN_ADDR, CMD_GO_ON
STS_CFG_TX_OVERFLOW	27h	There is more than 4000h bytes of TX data configured	CMD_MAS_ASIGN_ADDR, CMD_GO_ON

2.2.2.1.3 Configuration Software Errors

The following errors occur when the SST Config tool is used to configure the module as a DP master and the slave designation fields are used to set module options. If invalid data is entered, the following errors will appear in the status register.

The status register will show the following values for configuration software errors.

Error Message	Value	Description	Applicable Commands
STS_CFG_DESIG_NAME_TOO_LONG	28h	Nm= parameter, name too long (12 chars max)	CMD_CFG_2BF_SHRAM
STS_CFG_DESIG_BAD_ARG	29h	Unrecognized argument (Nm=, Tx=, Rx=, Ch)	CMD_CFG_2BF_SHRAM
STS_CFG_DESIG_INV_RX_OFS	2Ah	Rx= parameter, invalid offset (00003ff8)	Not used
STS_CFG_DESIG_INV_TX_OFS	2Bh	Tx= parameter, invalid offset (00003ff8)	Not used
STS_CFG_DESIG_OFS_NOT_SPEC	2Ch	Rx,Tx Ofs has been spec'd for one, but not all slaves	CMD_CFG_2BF_SHRAM
STS_CFG_RX_OVERLAP	2Dh	Rx data for one block overlaps another	CMD_MAS_ASIGN_ADDR, CMD_GO_ON
STS_CFG_TX_OVERLAP	2Eh	Tx data for one block overlaps another	CMD_MAS_ASIGN_ADDR, CMD_GO_ON
STS_CFG_INV_LEN	2Fh	Invalid parameter or check data length	CMD_CFG_ABF_SHRAM, CMD_CFG_2BF_SHRAM
STS_CFG_MAS_EXT_ALLOC_ERR	35h	Out of master extension memory	CMD_CFG_ABF_SHRAM, CMD_CFG_2BF_SHRAM
STS_CFG_ADDR_OUT_OF_RANGE	36h	Rx or Tx offset was out of range	Not used
STS_CFG_COPY_TABLE_OVERUN	37h	Not enough room in the copy table(backplanes only)	CMD_CFG_ABF_SHRAM, CMD_CFG_2BF_SHRAM

2.2.2.1.4 Flash Programming Errors

The following errors may occur when the host issues a command to program flash memory. The status register will show the following values for the programming errors.

Error Message	Value	Description	Applicable Commands
STS_CFG_NO_CONFIG	30h	No configuration available in flash or no configuration available in shared memory to program flash	CMD_PGM_TO_FLASH, CMD_CFG_FLASH_GO_ON, CMD_CFG_FROM_FLASH
STS_FLASH_BAD_ID	31h	Flash ID is not recognized	CMD_PGM_TO_FLASH
STS_FLASH_ERASE_ERR	32h	There has been an error when erasing the flash	CMD_PGM_TO_FLASH
STS_FLASH_PROG_ERR	33h	There has been an error when programming the flash	CMD_PGM_TO_FLASH
STS_FLASH_VRFY_ERR	34h	There has been an error when verifying the flash	CMD_PGM_TO_FLASH

2.2.2.1.5 Fatal Errors

The following errors are fatal errors. The module software must be rerun, or reloaded.

Error Message	Value	Description
STS_CFG_INTERNAL_ERROR	80h	There is an internal error on the card
STS_OUT_OF_APBS	81h	The module has run out of application blocks. Each configured SAP uses 3 application blocks. The DP slave uses 2 application blocks. The DP master uses 2 application blocks per configured slave. There are a total of 835 application blocks.
STS_HOST_WD_BITE	82h	The host watchdog has timed out
STS_HEAP_ALLOC_FAIL	83h	The module has run out of local RAM
STS_SH_HEAP_ALLOC_FAIL	84h	The module has run out of shared memory
STS_NET_ERROR	90h	There has been a network error and the OPTION_STAY_OFF_ERR bit is set in the pfbOptions register

2.2.3 ID Registers

The host uses the ID registers to identify the card, the PFBSCAN module, and the module revision. Applications can use the card ID, pfbCardId, to verify that the card is present.

2.2.3.1 Card ID

The Card ID register contains a value of AAD0h or 43728 decimal.

2.2.3.2 Module ID

Applications can use the DP module ID, pfbModId, to verify that the correct module is loaded on the card. The Module ID register shall contain a value of BB01h or 47873 decimal for the original cards and BB03h or 47875 decimal for the expanded I/O card.

2.2.3.3 Module Version

The version number of the module is stored in pfbModVer, with the major version in the low byte and the minor version in the high byte. For example, if the value read is 0102h, the version number is 1.02.

The Module Version register shall contain the module version.

2.2.4 Event Control

The various module operations can notify the host when certain events take place. The host's application can be event driven, if so desired. To control events, the host uses the event registers.

Register	Description
PfbEvtEna	Event enable register
PfbIntEna	Event Interrupt register
PfbEvtHead	Event queue head pointer
PfbEvtTail	Event queue tail pointer

2.2.4.1 Accessing the Event Queue

The module indicates that an event has occurred by writing to the event queue, pfbEventQueue[256].

The module and the host use two pointers into the event queue to control queue access. If the head and tail pointers are different, the host determines that there are unprocessed events in the queue.

Whenever the module adds an event to the queue, it increments the head pointer (pfbEvtHead).

Once the host removes an event from the queue, it increments the tail pointer (pfbEvtTail). Since the pointers are unsigned chars, they wrap around to 0 when they are incremented past 255. The host should remove and process any events in the queue. The high byte of the event contains the event type. The low byte may indicate the source of the event; for example, if the event is a received data change on a Layer 2 message, the low byte indicates the message block number. For some event types, if a new event occurs before the host removes the last one from the queue, the module does not queue the new event. Instead, it increments the errEventOverun counter.

2.2.4.2 Event Format

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Event Class				Event Type				Event Info							

Each event in the event queue is a 16-bit integer. The 4 high bits, bits 12-15, indicate the event class (DP slave, DP master, and so on). Bits 8-11 indicate the event type and are listed in the tables below. Bits 0-7 may be used to indicate further information about the event. For example, if the event is a DP master event, bits 0-7 indicate which master control block caused it.

2.2.4.3 Event Classes

Event Class	Definition	Value (Bits12-15)
PROFIBUS event	EVT_PFB	0h
LAN event	EVT_LAN	1h
Active station list event	EVT_LAS	2h
DP master event	EVT_MAS	3h
DP slave event	EVT_SLV	4h
Layer 2 message event	EVT_LY2M	5h
Layer 2 SAP event	EVT_LY2S	6h
Event queue event	EVT_QUE	Fh

2.2.4.3.1 Profibus Events

These relate to the overall operation of the module on the Profibus network.

When the EVT_ENA_BUS_ERROR bit is set in the pfbEvtEna register, Profibus events are enabled.

The table shows possible Profibus events. The low byte is always 0.

Event Type	Value	Description
EVT_PFB_FAT_RUN_ERR	0100h	There has been a command error
EVT_PFB_OFFLINE_STAY	0200h	There has been a LAN error and the module will stay offline
EVT_PFB_FATAL_INTERNAL	0300h	Fatal internal error

2.2.4.3.2 LAN Events

LAN events relate to the operation of the ASPC2 LAN controller. For more information about the causes of these events, refer to Section 2.2.6, ASPC2 Profibus Controller Basic Parameters.

Note

LAN events are never disabled.

The low byte is always 0. The firmware puts the value of the errNetTout in the lower byte.

The table shows possible LAN events.

Event Type	Value	Description
EVT_LAN_INV_REQ_LEN	1000h	Invalid request length for the given slave
EVT_LAN_FIFO_ERR	1100h	The LAN FIFO has overflowed
EVT_LAN_RX_OVERUN	1200h	The LAN receive buffer has overrun
EVT_LAN_DBL_TOK	1300h	The LAN has detected two tokens
EVT_LAN_RSP_ERR	1400h	There has been a response error from a slave
EVT_LAN_SYNI_ERR	1500h	The SYNI timer on the LAN has expired
EVT_LAN_NET_TOUT	1600h	There has been a timeout on the network
EVT_LAN_BAD_HSA	1700h	A station outside of the highest station address has been detected
EVT_LAN_BAD_STN	1800h	The station already exists on the network
EVT_LAN_TOK_PASS_ERR	1900h	There has been a token passing error
EVT_LAN_LAS_BAD	1A00h	The list of active stations is expired
EVT_LAN_OFFLINE	1B00h	The LAN has gone offline

2.2.4.3.3 Active Station List Events

These occur when there are changes in the active station list.

When the EVT_ENA_LAS_CHANGE bit is set in the pfbEvtEna register, the active station list events are enabled. The low byte contains the station that changed.

The table shows possible active station list events.

Event Type	Value	Description
EVT_LAS_ACT_STN_ON	20NNh	An active station has come online
EVT_LAS_ACT_STN_OFF	21NNh	An active station has gone offline
EVT_LAS_PSV_STN_ON	22NNh	A passive station has come online
EVT_LAS_ PSV_STN_OFF	23NNh	A passive station has gone offline

The module generates new events for active stations even if the changed bit is not cleared. However, if the changed bit is already set, the module does not generate new events for passive stations.

2.2.4.3.4 DP Master Events

These events occur when they are enabled and the module is being used as a DP master.

The table shows possible DP master events. The low byte contains the number of the master control block corresponding to the event, except for the EVT_MAS_SCAN_DONE event, where he low byte is zero.

Event Type	Value	Description
EVT_MAS_RX_DATA_CHG	3000h	There has been a change of data in the receive buffer
EVT_MAS_UPDTE	3100h	There has been an update to the master control block
EVT_MAS_SCAN_DONE	3200h	The scan has been completed
EVT_MAS_OK	3300h	The master control block now has no errors
EVT_MAS_ERROR	3400h	The master control block has an error

When the host sets bit 1 in the masCntCfg register in the master control block for the slave, the EVT_MAS_RX_DATA_CHG event is enabled.

When the host sets bit 3 in the masCntCfg register in the master control block for the slave, the EVT_MAS_UPDTE event is enabled by the module.

When an EVT_MAS_UPDTE event occurs, the module sets bit 0, MAS_EVT_UPD, to indicate that this slave has been updated.

When an EVT_MAS_RX_DATA_CHG event occurs, the module sets bit 1, MAS_EVT_RX_DATA_CHG, to indicate that the received data from this slave has changed. Since only one or the other can be enabled, there is never more than one event in this register. The host acknowledges by clearing the bit. If the host does not clear the masEvent register, the module does not generate any further data change or update events for this slave and instead increments the event overrun counter.

If the host sets the PFB_MAS_CTRL_EVT_SCAN_DONE bit in the pfbMasCntrlCfg register, the end-of-scan event, EVT_MAS_SCAN_DONE, is enabled. When the scan is done and the EVT_MAS_SCAN_DONE event is enabled, the module generates an event in the event queue and sets the PFB_MAS_SCAN_DONE bit in the pfbMasGlbEvt register

If the host application does not clear the pfbMasGlbEvt register to acknowledge the event, the module does not generate any more end-of-scan events and instead increments the event overrun counter.

To disable the EVT_MAS_OK and EVT_MAS_ERROR events, the host sets the MAS_CTL_IGNORE_STS in the masCntCfg register of the master control block for the slave.

2.2.4.3.5 DP Slave Events

These events occur when they are enabled and the module is being used as a DP slave. For information on how to enable and disable slave events with the slvCntCfg register, refer to Section 2.2.14.2, Slave Control/Config Register, and for details on accessing DP slave events, refer to Section 2.2.16.4, DP Slave Event Register.

The low byte is always 0. The table shows possible DP slave events.

Event Type	Value	Description
EVT_SLV_ RX_DATA_CHG	4000h	There has been a change in data in the slave's receive buffer
EVT_SLV_ UPDTE	4100h	There has been a slave update
EVT_SLV_OK	4200h	The slave is now has no errors
EVT_SLV_ERROR	4300h	The slave has an error
EVT_SLV_CHG_TO_RUN	4400h	The slave has been put into RUN mode
EVT_SLV_CHG_TO_STOP	4500h	The slave has gone into PROGRAM mode

2.2.4.3.6 Layer 2 Message Events

These events occur when the module is being used to send FDL (Layer 2) messages. The low byte contains the message block number. For information on what these events mean and how to enable or disable these events, refer to Section 2.2.20.5, FDL (Layer 2) Messages.

Event Type	Value	Description
EVT_LY2M_RX_DATA_CHG	50NNh	There has been a change in the FDL data
EVT_LY2M_CONFIRM	51NNh	An FDL message has arrived
EVT_LY2M_ERROR	52NNh	There is an error with the FDL message
EVT_LY2M_BAD_MSG_NUM	53NNh	The message number does not exist

2.2.4.3.7 Layer 2 SAP Events

These events occur when you have configured FDL (Layer 2) SAPs on the module. The low byte contains the SAP number. For information on what these events mean and how to enable or disable these events, refer to Section 2.2.20.4, FDL (Layer 2) SAPs.

Event Type	Value	Description
EVT_LY2S_ RX_DATA_CHG	60NNh	There has been a in SAP data
EVT_LY2S_UPDATE	61NNh	There has been a SAP update
EVT_LY2S_ ERROR	62NNh	There is a SAP error

2.2.4.3.8 Event Queue Events

These events occur when an error happens in the event queue. Event queue events cannot be disabled.

Event	Value
EVT_QUE_OVERUN	000Fh
EVT_QUE_BAD_TRIG	F100h. The firmware always sets the low byte to the low byte in the bad trigger

If the host cannot process events quickly enough, the event queue fills up, generating the EVT_QUE_OVERUN event. This event replaces the last event in the queue. If you enter an invalid entry in the trigger queue, the EVT_QUE_BAD_TRIG event occurs.

2.2.5 Interrupt Control

The various module operations can notify the host when certain events take place. The host application can be interrupt driven, if so desired. To control interrupts, the host uses the event and interrupts registers.

2.2.5.1 Using Interrupts

The PFBSCAN module can generate host interrupts on the same conditions that generate events. For details on how to configure interrupts on the card hardware, refer to the relevant Hardware Reference Guide. The following sections describe the general steps for initializing and processing interrupts.

2.2.5.2 Setting up Interrupts

The host can selectively enable interrupts for any or all of the module functions by setting specific bits in the pfbIntEna Register.

The module enables interrupts for the following events when the associated bit is set in the pfbIntEna register and the event has been enabled.

Event Class	Set Bit in pfbIntEna	Enabling Event Class
Active Station List Event	INT_ENA_LAS_CHANGE	Refer to Section 2.2.4.3.3, Active Station List Events
Profibus Error Event	INT_ENA_BUS_ERROR	Refer to Section 2.2.4.3.1, Profibus Events
DP Master Event	INT_ENA_DP_MAS_EVENT	Refer to Section 2.2.4.3.4, DP Master Events
DP Slave Event	INT_ENA_DP_SLV_EVENT	Refer to Section 2.2.4.3.5, DP Slave Events
FDL Message Event	INT_ENA_LY2M_EVENT	Refer to Section 2.2.4.3.6, Layer 2 Message Events
FDL SAP Event	INT_ENA_LY2S_EVENT	Refer to Section 2.2.4.3.7, Layer 2 SAP Events

After configuring the pfblntEna register, the host must enable card interrupts. For more details, refer to the relevant Hardware Reference Guide.

2.2.5.3 The PFB_WaitIrq API Function

In Windows environments, the actual interrupt routine is contained in the SST Kmode driver. The user mode application must ask the driver for an indication that it has serviced an interrupt. The PFB_WaitIrq API call should be made from a separate thread in the user mode process, because the thread or process execution will stop until the driver receives an interrupt.

To start using interrupts:

1. Call the PFB_WaitIrq function and test for a TRUE return value.

2. Process all events that have occurred in the module by updating the event queue tail pointer and clearing any event registers.

2.2.5.4 Ending Interrupts

To stop using interrupts:

1. Clear pfbIntEna. For details, refer to Section 2.2.5.2, Setting up Interupts.

2. Disable card interrupts. For details, refer to the relevant Hardware Reference Guide.

2.2.6 ASPC2 Profibus Controller Basic Parameters

Network parameters include the card station address, the network baud rate, various timing parameters and limits on the numbers of retries.

The host needs to set the network parameters before the module is put online. To change the network parameters, the host must take the module offline. If you configure the module as a DP master using a binary file exported from the SST Profibus Configuration Tool, the network parameters are included in the configuration file and you do not need to set them. You can still use the module registers to see the values set by the SST Profibus Configuration Tool.

The following sections describe the various network parameters.

2.2.6.1 Basic Parameters

The basic network parameters include:

- The local station address
- The high station address for the network
- Whether the station is active or passive
- The network baud rate
- Some network options

2.2.6.1.1 Local Station Address

The host uses the pfbStnAddr register to set the station address of the card on the Profibus network. The default station address is FFh, which is invalid.

If the station address is not from 0 to 125, the module reports the error, STS_BAD_STN_ADR, in pfbStatus. For details, refer to Section 2.2.2, Status Register.

The host must set the station address before putting the module online.

2.2.6.1.2 High Station Address

The high station address register, pfbHiStnAddr, is the highest allowed station address for any active station on the network. All active stations on the network should use the same value for the high station address.

If the module sees an active station with a higher address and the OPTION_STAY_OFF_ERR bit in the pfbOptions register is set, the module will go offline with a status of STS_NET_ERROR. If this bit is not set, the module will increment the errHsa error counter and go back online.

The default high station address is 126. If the high station address is greater than 126, the module reports the error, STS_BAD_HI_STN_ADR, in pfbStatus. For details, refer to Section 2.2.2, Status Register.

The high station address affects how much time is spent soliciting new nodes. If the nodes are assigned consecutive addresses and the high station address is set to the address of the highest node, no soliciting will take place and all network time will be used for messages. However, this also means that no new nodes can come on the network.

The gap update factor (see Section 2.2.6.2, Bus Parameters) also affects how often soliciting takes place. When assigning station numbers, leave as few gaps as possible so that fewer stations spend time soliciting.

The high station address applies only to active stations. Passive stations are able to have station addresses higher than the high station address.

2.2.6.1.3 Active/Passive

A passive station can only reply to messages; an active station can initiate messages and reply to messages. Only active stations participate in the token rotation. Adding passive stations does not affect the token rotation time.

The host sets whether the card is an active or a passive station by writing to the pfbActive register. This register is set to 1 for an active station, or 0 for a passive station. The default shall be for the station to be passive (pfbActive=0). To act as a DP master or to initiate FDL, a station must be active.

2.2.6.1.4 Baud Rate

The host uses the pfbBaud register to set the network baud rate, according to the values in the following table:

Baud Rate	Profictl.h	Value
9600	BAUD_9K6	0h
19200	BAUD_19K2	1h
93.75K	BAUD_93K75	2h
187.5K	BAUD_187K5	3h
500K	BAUD_500K	4h
1.5M	BAUD_1M5	6h
3M	BAUD_3M	7h
6M	BAUD_6M	8h
12M	BAUD_12M	9h
31.25K	BAUD_31k25	0Ah
45.45k	BAUD_45k45	0Bh

The default baud rate is 9600. The module can automatically detect the network baud rate.

If the baud rate does not match one of the values in the above table, it is invalid, and the module reports the error, STS_BAD_BAUD, in pfbStatus. For details, see Section 2.2.2, Status Register.

To detect the baud rate, the host issues the CMD_AUTO_BAUD_DET command. The module then listens to the network (but does not go online) at each possible baud rate, for a period determined by the rate. This can take up to 6 seconds.

If the module detects 10 valid messages in a row, it decides that it has found the baud rate and it enters the corresponding value in pfbBaud, setting pfbCommand back to CMD_OFF. If the module fails to detect the network baud rate, it returns a status of STS_BAUD_DET_ERROR (10h) in the pfbStatus register and sets the baud rate to the default.

2.2.6.1.5 Network Options

By setting bits in the pfbOptions register, the host sets several options related to how the module operates on the network The host uses bit 0, OPTION_REPEATER, to tell the module whether there are repeaters on the network. If there is at least one repeater, the host sets this bit to 1; if there are no repeaters, the bit is set to 0. The module checks this bit when it assigns the default bus parameters (except pfbTokRotTime). The default for bit 0 of the pfbOptions register is 0 (no repeaters).

The host uses bit 1, OPTION_FMS, to tell the module whether there are FMS devices on the network. Set this bit to 1 if there are FMS devices on the network, and set it to 0 if the network consists of only DP devices. The module checks this bit when it assigns the default bus parameters (except pfbTokRotTime). The default for bit 1 of the pfbOptions register is 0 (DP only).

The host uses bit 2, OPTION_STAY_OFF_ERR, to tell the module what to do when the token error limit, pfbTokErrLimit (see Section 2.2.6.3, Error Handling Parameters), or the message error limit, pfbRespErrLimit, is exceeded within 256 token cycles. If bit 2 of the pfbOptions register is 0 and either of these error conditions occurs, the module increments the corresponding error counter and goes offline, then goes back online immediately. If the bit is 1, the module goes offline with a fatal error and must be restarted before it is put back online. The default for bit 2 of the pfbOptions register is 0 (the module goes back online).

2.2.6.2 Bus Parameters

The bus parameters relate to quantities, for example, times between messages. The module assigns defaults based on the baud rate for bus parameters not explicitly set. The defaults also depend on whether the OPTION_REPEATER and OPTION_FMS bits are set in the pfbOptions register. These parameters do not usually need to be changed from their default values.

2.2.6.2.1 Target Token Rotation Time (pfbTokRotTime)

The target token rotation time is the target maximum token rotation time for the network, in Tbits. If a station gets the token and the target token rotation time has expired, it sends only one high priority message, then passes the token. The target rotation time applies only to active nodes and should be set to the same value for all active nodes. If there are multiple DP masters on the network, increase the target token rotation time for each master, setting it to the sum of the target token rotation time required for each master. If you have configuration problems, especially at lower baud rates, increase this time.

If the target token rotation time is not between 256 and 16,777,215, the STS_BAD_TOK_ROT error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

2.2.6.2.2 Slot Time (pfbSlotTime)

The slot time or frame timeout is how long the module waits for a reply to a message, in Tbits. If the module does not receive the reply within this time, it retries the message up to the maximum number of times specified in pfbMsgRetryLimit. If the module has not successfully sent a message to the destination previously, it does not retry the message. The slot time is also the time the module waits for a reply when it polls for new nodes on the network.

If the slot time is not in the range of 37 to 16,383, the STS_BAD_SLOT_TME error is reported in pfbStatus. For details, see Section 2.2.2, Status Register.

2.2.6.2.3 Idle Times (pfbIdleTime1 and pfbIdleTime2)

Idle time 1 is the time, in Tbits, that the module waits after it receives a reply, an acknowledge or a token message before sending another message. If idle time 1 is not in the range of 35 to 1023, the STS_BAD_IDLE_1 error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

Idle time 2 is the time, in Tbits, that the module waits after sending an SDN (send data with no acknowledge) message before it sends another message. If idle time 2 is not in the range of 35 to 1023, the STS_BAD_IDLE_2 error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

2.2.6.2.4 Ready Time (pfbReadyTime)

The ready time is the time in Tbits that the module, after sending a command, waits before sending an ACK or response; it is also the time the module waits after receiving a command, before it sends a reply.

If the ready time is not in the range of 11 to 1023, the STS_BAD_RDY_TME error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

2.2.6.2.5 Gap Update Factor (pfbGapUpdFact)

The gap update factor is the number of token rotations between solicits for a new node. The process of soliciting new nodes is called gap update. If the gap update factor is large, nodes spend less time soliciting for new nodes but it takes longer for new nodes to come on the network.

The default for the gap update time is 128. If the gap update factor is not in the range of 1 to 255, the STS_BAD_GAP_UPD error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register

2.2.6.2.6 Quiet Time for Modulator (pfbQuiTime)

The modulator quiet time is the time, in Tbits, that the module waits after it turns on its transmitter before it begins to send data; it is also the time the module keeps its transmitter on after it has finished transmitting a message.

If the quiet time is not in the range of 0 to 127, the STS_BAD_QUI_TME error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

2.2.6.2.7 Network Parameter Defaults

The following tables show the default values assigned by the module.

No repeater, no FMS:

Baud Rate	Slot Time	Idle Time1	Idle Time2	Ready Time	Qui Time
9600	100	37	60	11	0
19200	100	37	60	11	0
93.75 K	100	37	60	11	0
187.5K	100	37	60	11	0
500K	200	37	100	11	0
1.5M	300	37	150	11	0
3M	400	45	250	11	3
6M	500	55	350	11	6
12M	750	75	550	11	9
31.25K	100	37	60	11	0
45.45k	100	37	60	11	0

No repeater, FMS:

Baud Rate	Slot Time	Idle Time1	Idle Time2	Ready Time	Qui Time
9600	125	37	60	30	0
19200	250	61	120	60	0
93.75 K	600	126	250	125	0
187.5K	1500	251	500	250	0
500K	3500	251	1000	250	0
1.5M	3000	151	980	150	0
3M	400	45	250	11	3
6M	500	55	350	11	6
12M	750	75	550	11	9
31.25K	300	61	120	60	0
45.45k	400	61	120	60	0

Repeater, no FMS:

Baud Rate	Slot Time	Idle Time1	Idle Time2	Ready Time	Qui Time
9600	100	37	60	11	0
19200	100	37	60	11	0
93.75 K	100	37	60	11	0
187.5K	100	37	60	11	0
500K	200	37	100	11	0
1.5M	300	37	150	11	0
3M	400	45	250	11	3
6M	500	55	350	11	6
12M	750	75	550	11	9
31.25K	100	37	60	11	0
45.45k	100	37	60	11	0

Repeater, FMS:

Baud Rate	Slot Time	Idle Time1	Idle Time2	Ready Time	Qui Time
9600	125	37	60	30	0
19200	250	61	120	60	0
93.75 K	600	126	250	125	0
187.5K	1500	251	500	250	0
500K	3500	251	1000	250	0
1.5M	3000	151	980	150	0
3M	400	45	250	11	3
6M	500	55	350	11	6
12M	750	75	550	11	9
31.25K	300	61	120	60	0
45.45k	400	61	120	60	0

2.2.6.3 Error Handling Parameters

The error handling parameters determine how the ASPC2 ASIC handles errors on the network; for example, how many times it retries messages.

2.2.6.3.1 Token Retry Limit (pfbTokRetryLimit)

The token retry limit is the number of times the module retries to pass the token before deciding that the station is not present. If the module comes to this conclusion, it takes the station out of the active station list and passes the token to the next station in the active station list.

The default token retry limit is 4. If the token retry time is not in the range of 0 to 15, the STS_BAD_TOK_RETRY is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

2.2.6.3.2 Message Retry Limit (pfbMsgRetryLimit)

The message retry limit is the maximum number of times the module retries a message after the slot time expires. For example, if it is set to 4, the module tries the message a total of 5 times. If the module still has not received a reply after the maximum number of retries expires, the message is aborted and returned with an error. What happens next depends on the function being performed. For example, if the message is an I/O update with the module as a DP master, the slave will fail and will need to be reinitialized. If the module has not successfully sent a message to a station previously, it will not retry messages.

The default shall be 4. If the message retry time is not in the range of 0 to 15, the STS_BAD_MSG_RETRY error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

2.2.6.3.3 Token Error Limit (pfbTokErrLimit)

The token error limit is the maximum number of errors in 256 token cycles. If the OPTION_STAY_OFF_ERR bit in pfbOptions is 1, the module goes offline with a fatal error and it must be restarted before it is put back online. If the bit is 0, the module increments the corresponding error counter, goes offline, and then goes back online immediately.

The default value is 255. If the token error limit is not in the range of 0 to 255, the STS_BAD_TOK_ERR_LIM error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

2.2.6.3.4 Response Error Limit (pfbRespErrLimit)

The response error limit is the maximum number of message failures (e.g., retry limit is exceeded) in 16 successive messages.

The default value is 15. If the response error limit is not in the range of 1 to 15, the STS_BAD_RSP_ERR_LIM error is reported in pfbStatus. For details, refer to Section 2.2.2, Status Register.

If the OPTION_STAY_OFF_ERR bit in pfbOptions is 1, the module goes offline with a fatal error and it must be restarted before it is put back online. If the OPTION_STAY_OFF_ERR bit is 0, the module increments the corresponding error counter and goes offline, and then goes back online immediately.

2.2.7 Trigger Queue Control

For more details on the trigger queue, refer to Section 2.2.20.3, The Trigger Queue.

2.2.8 DP Master Status and Control

2.2.8.1 DP master Global Control Register (pfbMasCntrlCfg)

The host uses the DP master global control register in the PROFI_USR structure to set options for and control the overall operation of the module as a DP master. The host sets some of these bits when it is configuring the master; it sets others when it is online.

The host sets bit 1, PFB_MAS_CTRL_RUN_MODE, while it is online, to set the DP master's scanning mode.

If the bit is 1, the module scans I/O in run mode. In run mode, the module reads inputs and updates outputs. If the bit is 0, the module is in program (stop) mode. In program mode, the module reads inputs and sends all 0s for outputs.

The host sets bit 2, PFB_MAS_CTRL_USR_OFS, while it is configuring the module as a master. This tells the module that the host is going to assign offsets for received and transmitted data, and that these offsets should not be assigned. The host must do this before it puts the module online as a master. When using this feature, the application is responsible for assigning data offsets in the input and output pages. This feature may be useful in embedded applications, where it may be difficult to change the embedded application to use new I/O offsets. If you enable this option, assign offsets for all slaves being scanned.

The module will not go online if any overlaps are found in the offsets.

The host sets bit 3, PFB_MAS_CTRL_ENABLE, before the online command to tell the module that you are using the DP master function.

The host sets bit 4, PFB_MAS_CTRL_DIS_LED, while it is configuring the module as a DP master to disable the master status on the system status LED. If this bit is set, the master status will not be shown.

The host sets bit 5, PFB_MAS_CTRL_HOLD_INTR, while it is configuring the module as a DP master to tell the module not to generate any DP master interrupts until the end of the scan. For example, if you are scanning at a high baud rate and there are several stations with changing data, set this bit to generate a single interrupt at the end of the scan, then process all pending events, rather than handling each one as it occurs.

The host sets bit 6, PFB_MAS_CTRL_EVT_SCAN_DONE, while it is configuring the module as a DP master to tell the module to generate an event every time the I/O scan is complete.

The module sets bit 7, PFB_MAS_CTRL_ADDR_ASIGNED, to indicate that it has assigned slave data offsets in response to a CMD_MAS_ASIGN_ADDR or ON_LINE command. When the module goes online, it checks this bit and, if it is set, the offsets are not reassigned. The host should not change the state of this bit.

2.2.8.2 DP Master Global Status Register (pfbMasSts)

The DP master global status register, in the PROFI_USR structure, shows the status of the overall operation of the module as a DP master. To indicate that all configured slaves are being scanned with no problems, the module sets bit 0, PFB_MAS_STS_ALL_OK, to 1. If this bit is 0, there are problems with one or more slaves. Check the status of each slave to determine where the problems exist.

The module sets bit 0 to 1 if all slaves are being scanned with no problems and bit 0 to 0 if any slave has a problem.

2.2.8.3 DP Master Page Registers

In the PROFI_USR structure, there are three registers that tell the host where the master control blocks, DP master input data and DP master output data are located:

- The pfbMasCntrlPage register contains the page number of the memory page that contains the master block control table, the table with the configuration data for each slave.
- The pfbMasRxPage register contains the page number of the memory page that contains master received data (from slaves).
- The pfbMasTxPage register contains the page number of the memory page that contains master transmit data (to slaves).

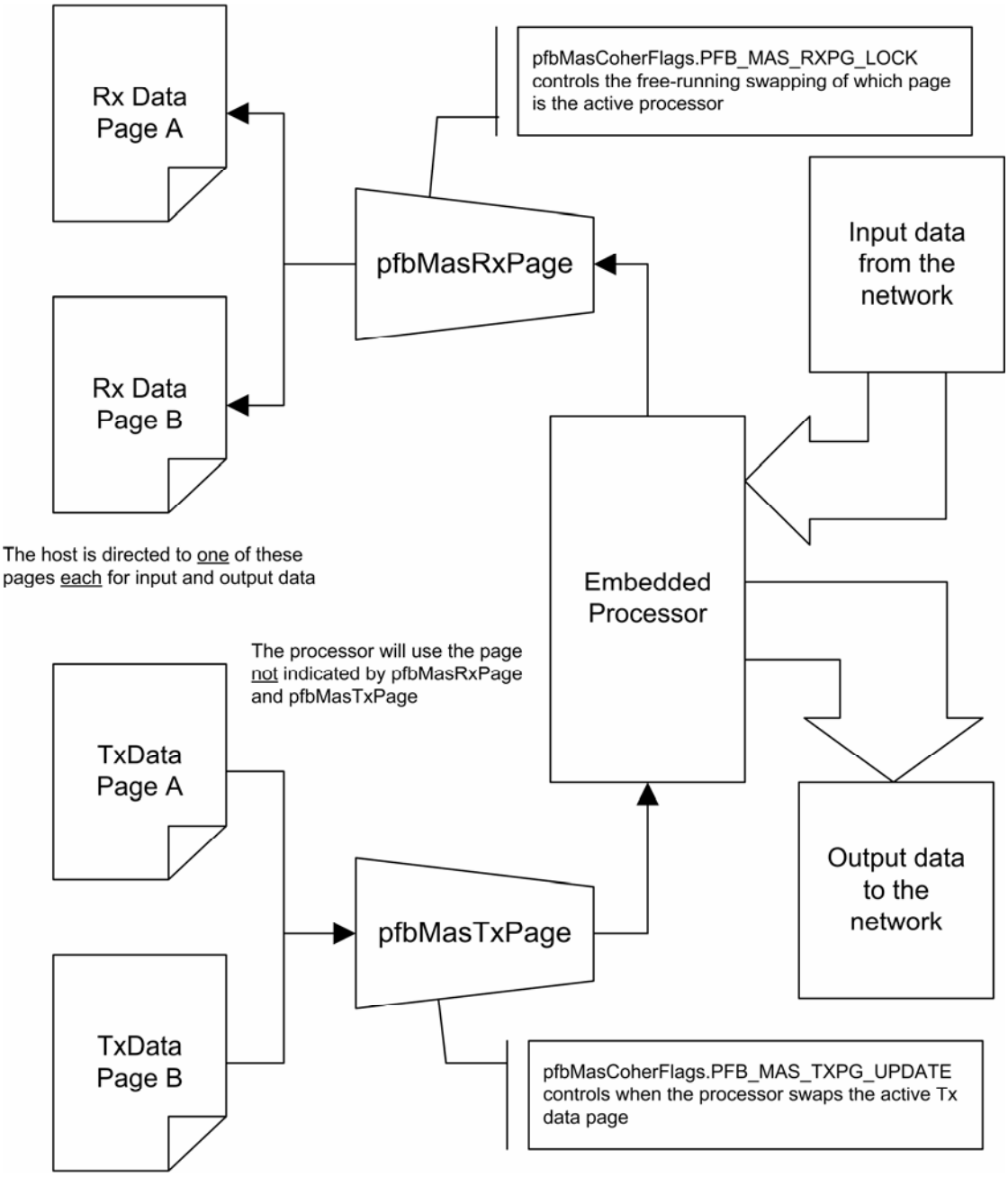
For information on accessing shared memory pages, refer to the relevant Hardware Reference Guide.

2.2.8.4 DP Master Data Coherency Mode

This section explains the software method that provides data coherency across an entire DP slave device. The pfbMasCoherFlags register contains all the control and configuration flags for the data coherency mode of operation. If the host sets the PFB_MAS_COHER_ENA bit in this register, prior to putting the module online, puts the module in data coherency mode.

```
Profictl.h

/* definitions related to pfbMasCoherFlags */

#define PFB_MAS_TXPG_UPDATE     0x01    /* The tx data for at least one mas block has
                                           been updated */

#define PFB_MAS_RXPG_LOCK       0x02    /* The host is currently reading the at least
                                           one mas block's rx pg */

#define PFB_MAS_COHER_ENA       0x80    /* Enable the master block data coherency
                                           option */
```

Figure 1: Data Coherency Mode



2.2.8.4.1 Dealing with DP Input Data

Normally, the PROFI_USR register, pfbMasRxPage, contains a page reference to the DP input data image that never changes after the module is put online. In data coherency mode, this number is set by the module's embedded processor at the end of each scan and is one of two possible values. This is done to ensure that the embedded processor and the host processor never access the same page at the same time. Before reading any DP input data from the Rx data page, the host must assert the PFB_MAS_RXPG_LOCK bit in the pfbMasCoherFlags register. This prevents the embedded processor from changing the host's valid Rx page in the middle of a host read operation. When the host finishes reading the DP input data, the PFB_MAS_RXPG_LOCK bit must be cleared by the host application or the host will never obtain access to new input data from the network.

The module will not switch valid Rx pages when the PFB_MAS_RXPG_LOCK bit is set in the pfbMasCoherFlags register.

2.2.8.4.2 Updating DP Output Data

The DP output data image works almost the same as the input data, with one major difference: the embedded processor does not change the number in the pfbMasTxPage until the host processor instructs the module that output data was updated. Therefore, the host should write all DP output data to be updated to the page in the pfbMasTxPage register. The embedded processor sends this data on the next bus scan as soon as the PFB_MAS_TXPG_UPDATE flag is set in the pfbMasCoherFlags register. When the new output data is processed and the pfbMasTxPage is changed to the alternative page number, the module clears the PFB_MAS_TXPG_UPDATE bit. The host should never clear the PFB_MAS_TXPG_UPDATE flag, nor write data to the DP output data page, when this bit is set.

The module will not change the number in the pfbMasTxPage until the host processor sets the PFB_MAS_TXPG_UPDATE bit in the pfbMasCoherFlags register.

2.2.8.5 Scan Time Limits

The host can set a minimum I/O scan time by writing to the pfbMasMinIoCycTme register. The value ranges from 0 to 65535. The units are in 100 µs increments, so the time ranges from 0 to 6.6635 seconds. This feature is sometimes required because certain I/O modules have restrictions on how often they can be scanned. Set this value based on the minimum scan times of all the slaves being scanned.

The host can set the maximum I/O scan time by writing to the pfbMasMaxIoCycTme register. The value ranges from 1 to 65535. The units are 10 µs increments, so the time ranges from 0.01 seconds to 655.35 seconds. If the scan time exceeds the value in the maximum scan time register, the master faults all the slaves, and then they get reinitialized and come back online. The minimum and maximum I/O scan times apply to the entire I/O scan for all slaves, not to the scan for an individual slave.

2.2.8.6 Fail-Safe Slaves

The DP Master scans all Profibus conforming fail-safe slaves.

2.2.9 Diagnostic Counters and Control

The module maintains a variety of diagnostic counters to indicate:

- General statistics on messages sent and received
- The state of the master
- The state of the slave
- FDL (Layer 2) message statistics
- Network statistics

To clear the counters, set pfbInitCtrs to 1. The module then clears the counters to 0 and pfbInitCtrs to 0 to indicate that the counters have been cleared. In the following section, counters whose names begin with “diag” roll over to zero when they reach their maximum value, and those with names beginning with “err” hold their maximum value.

2.2.9.1 General Statistics

These counters relate to the overall operation of the module on Profibus.

2.2.9.1.1 errLanOffline

If the OPTION_STAY_OFF_ERR bit in pfbOptions is 0, when the LAN encounters errors and goes offline, the module increments the errLanOffline error counter and then goes back online immediately. If the OPTION_STAY_OFF_ERR bit is 1, when the LAN encounters errors and goes offline, the module reports a fatal error and the host must then restart it before putting it back online.

2.2.9.1.2 diagConf

The diagConf counter counts total confirmations (good replies to messages that this station has generated). This is the total for DP master and FDL messages.

2.2.9.1.3 diagInd

The diagInd counter counts total indications (unsolicited messages to this station). This is the total for DP master and FDL messages.

2.2.9.1.4 errNotOK

The errNotOk counter counts the Total Not OK confirmations and indications (total bad replies and bad unsolicited messages (indications)). This is the total for DP master and FDL messages.

2.2.9.1.5 diagTokHldTime

The module stores the instantaneous token hold time, in Tbits, in diagTokHldTime. This time is the time available to send messages when the module gets the token.

2.2.9.1.6 diagMinTokHldTime

The module stores the minimum token hold time, in Tbits, in diagMinTokHldTime. This time is the minimum value of diagTokHldTime. If this number is 0, the host may need to increase the target token rotation time (delta TTR in COM PROFIBUS).

2.2.9.2 Master Block Statistics

These counters relate to the operation of the module as a DP master.

2.2.9.2.1 diagMasterUpdate

The diagMasterUpdate counter holds the number of Master I/O update cycles completed and the number of I/O scans completed by the master.

2.2.9.2.2 errMasErr

The errMasErr counter holds the number of DP master-to-DP slave communication errors. It increments anytime a message fails.

2.2.9.2.3 errReConfig

The errReConfig counter holds the number of times a DP slave went offline and had to be reconfigured by the master. The module increments this counter after it has retried the message the number of times specified in pfbMsgRetryLimit.

2.2.9.2.4 diagMasScanTime

The diagMasScanTime register contains the instantaneous master scan time in µs, that is, the time to scan all the slaves assigned to this master plus 100 µs to allow for overhead in starting the timer.

2.2.9.2.5 diagMasMaxScanTime

The diagMasMaxScanTime register contains the maximum value that diagMasScanTime reached since it was last cleared.

2.2.9.3 DP Slave Statistics

These counters relate to the operation of the module as a DP slave.

2.2.9.3.1 diagSlaveUpdate

The module increments the diagSlaveUpdate counter when it receives an I/O data update from the master.

2.2.9.3.2 errSlvErr

The module increments the errSlvErr counter when errors occur while the master is parameterizing the slave.

2.2.9.3.3 errSlvTout

The module increments the errSlvTout counter when the slave has not received a message from the master within the master timeout period.

2.2.9.4 FDL (Layer 2) Statistics

2.2.9.4.1 Message Block Statistics

diagLay2MsgOk

The module increments diagLay2MsgOk whenever it sends an FDL message and receives the appropriate acknowledge or response data.

errLay2MsgNotOk

The module increments errLay2MsgNotOk whenever it sends an FDL message and does not receive an appropriate acknowledge or response data.

2.2.9.4.2 SAP Statistics

diagLay2SapOk

The module increments diagLay2SapOk when it processes an FDL SAP request with no errors.

errLay2SapNotOk

The module increments errLay2SapNotOk when there is an error receiving a SAP request. For example, if strict SAP checking is enabled and a SAP request is received from a different source SAP, the module increments this counter.

errLay2SapTout

The module increments errLay2SapTout when a timeout has been set on a SAP and the timer times out.

2.2.9.5 ASPC2 Profibus Controller Statistics

The ASPC2 LAN controller maintains the following counters. They are all 1 byte long, and when they reach 255, they hold at 255 until cleared.

2.2.9.5.1 errInvReqLen

The errInvReqLen counter counts invalid request length errors. These errors occur when the module gives the LAN controller a message that is too long. This error is an internal module error and should never occur.

2.2.9.5.2 errFifo

The errFifo counter counts FIFO overflow errors, which occur when the LAN controller cannot write to memory fast enough. These are internal module errors and should never occur.

2.2.9.5.3 errRxOverun

The errRxOverun counter counts receive overrun errors. These are internal module errors and should never occur.

2.2.9.5.4 errDblTok

The errDblTok counter counts double token errors. These errors may occur when more than one node thinks it has the token, or due to wiring errors, duplicate nodes, and so on. The module withdraws to the “not hold token” state and waits until the token is passed to it again.

2.2.9.5.5 errRespErr

The errRespErr counter is incremented when a message fails or there was no response from the destination. If the errRespErr counter is incremented and the OPTION_STAY_OFF_ERR bit in the pfbOptions register is set, the module will stay offline, otherwise, it will go offline and then come back online immediately.

2.2.9.5.6 errSyniErr

The errSyniErr counter counts general network errors. These errors occur when there are problems on the network that are not severe enough to cause a network timeout error.

2.2.9.5.7 errNetTout

The errNetTout counter counts network timeout errors. These errors occur when the network is dead. If a timeout occurs, the module enters the claim token state.

2.2.9.5.8 errHsa

The errHsa counter is incremented when a station number higher than the high station address set on the master is detected.

If the errHsa counter is incremented and the OPTION_STAY_OFF_ERR bit in the pfbOptions register is set, the module will stay offline, otherwise it will go offline, and then come back online immediately.

2.2.9.5.9 ErrStn

The errStn counter is incremented when a duplicate station is detected. If the errStn counter is incremented and the OPTION_STAY_OFF_ERR bit in the pfbOptions register is set, the module will stay offline, otherwise it will go offline, and then come back online immediately.

2.2.9.5.10 errPasTok

The errPasTok counter is incremented when the module is unable to pass the token. This is usually caused by bad wiring (e.g., shorted) or other hardware problems. The module tries to pass the token, fails to hear its own token pass message, and puts itself offline.

2.2.9.5.11 errLasBad

The errLasBad counter is incremented when the active station list is invalid because of multiple network errors. This error is caused by bad wiring or hardware.

ErrInternal and errArg are reserved. If a fatal error occurs, the values in these registers may indicate the source of the problem. However, the module uses these locations for other purposes, so if there is a value in one of them, it does not necessarily indicate that a fatal error has occurred.

2.2.9.6 Event Statistics

2.2.9.6.1 errEventOverun

The module increments errEventOverun when a new event occurred before the last one was processed.

2.2.10 Active Station List

The module maintains a list of active stations on the network in the pfbActStnList[128] table, one byte per station. The first byte corresponds to station 0. The active station list is valid only if the module is an active, and not a passive, station.

If a station is active, the module sets bit 2, the LAS_ACTIVE bit, in the corresponding byte in the active station list.

If the status of a station changes, that is, when the station goes online or offline, the module sets bit 3, LAS_CHANGED.

If bit 0 in the pfbAckLasChnge register is set, the LAS_CHANGED the module clears the bits for all stations.

To indicate that the LAS_CHANGED bits for all of the stations have been cleared, the module clears pfbAckLasChnge.

The module can determine which passive stations are present on the network. To do this, set bit 0 in the pfbUpdPasv register to 1. The module then sends an FDL status request to any stations not already in the active station list. If the station replies, the module sets the LAS_PASSIVE bit (bit 0) for that station in the active station list. When the module has tried all missing stations, it clears pfbUpdPasv to tell the host that it has finished scanning all stations. If the passive stations list is updated, the errNotOk diagnostic counter increments once for each station that does not reply. The module also updates the changed bit for passive stations when the passive stations in the list are updated.

2.2.11 Master Configuration Parameters

2.2.11.1 Configuring the DP Master with a Binary File

Before it can be configured, the module must be offline. Check the command register for the CMD_OFF state. The module uses pfbBinCfgPage in the PROFI_USR structure to tell the host which page to write the configuration file on. Since the binary file may be large, the host must write it one 16 Kbyte block at a time. After the host writes each block, it writes the block number to pfbBinCfgOfs and the block length to pfbBinCfgLen. For all blocks but the last, the length is 16384 (16 Kbytes). When the host has finished writing each block, it issues the CMD_CPY_MAS_CFG command to the command register. The host should then wait for up to 1 second for the module to process the block and set the command register to return to CMD_OFF. This indicates that it has finished processing.

When the host has written the entire file and the module has processed all the blocks, the host issues the CMD_CFG_2BF_SHRAM command and waits up to 2 seconds for the command to complete. The module processes the file and validates the data it contains. If it encounters problems, it sets the command register to CMD_ERROR and indicates the cause in the status register. The host should acknowledge the error by writing 0xFF (CMD_ERR_ACK) to the Comand register. If the module successfully processes the file, it sets the command register to CMD_OFF.

The network and DP master parameters are now configured on the module. The host may now configure any other required operations, such as DP slave, and then put the module online.

2.2.12 Watchdog Parameters

Use the host watchdog feature to ensure that the module takes itself offline if the host application fails. When enabling the host watchdog, the host must check in with the module within the time set as the watchdog period. Otherwise, the watchdog times out and the module takes itself offline.

2.2.12.1 pfbWdTime

To enable the host watchdog, write a non-zero value to the pfbWdTime register in the PROFI_USR structure. This value sets the host watchdog time in milliseconds. The register pfbWdTime has a valid range from 1 to 65535 and is written as a word.

2.2.12.2 pfbWdKick

The host checks in with the watchdog by resetting the pfbWdKick register to 0. Every millisecond, the module checks to see if pfbWdKick is 0. If it is, the module restarts its internal watchdog timer and sets pfbWdKick to 1. If it is not, the module increments the internal watchdog timer.

If the value in the module's internal watchdog timer ever exceeds pfbWdTime, the watchdog bites and the module takes itself offline.

The PFBSCAN module must be restarted before putting the card back online.

2.2.13 Global Event Register pfbMasGlbEvt

2.2.13.1 DP Master Scanning Mode

When scanning as a DP master, the module scans in fully asynchronous mode.

2.2.13.1.1 End of DP Scan Notification

To determine when the module has finished a DP scan, PFB_MAS_CTRL_EVT_SCAN_DONE must be set in the pfbMasCntrlCfg register. Additionally, the host application can use a hardware interrupt to process the end of the DP scan. Enable an interrupt for this event by setting the INT_ENA_DP_MAS_EVENT in the pfbIntEna register. The pfbMasGlbEvt register contains the PFB_MAS_SCAN_DONE value after a DP scan. To recognize any further DP SCAN_DONE events and/or receive any further interrupts from this event, the application must clear the PFB_MAS_SCAN_DONE bit. Your application is now free to read the DP slave input data and update the DP slave output data tables.

If the PFB_MAS_CTRL_EVT_SCAN_DONE option is set in the pfbMasCntrlCfg register, the module writes the PFB_MAS_SCAN_DONE value into the pfbMasGlbEvt register after a DP scan.

The module increments the errEventOverun register for every DP scan completed with the PFB_MAS_SCAN_DONE bit still set.

2.2.14 Master Control Blocks

For each slave, there must be a master control block. Each master control block occupies 128 bytes. Some elements of the master control block are used for configuration data, while others are used by the module for slave status information.

If configuring the module as a DP master from the host, your application must build consecutive master control blocks for each slave.

When putting the module online as a DP master or issuing the CMD_MAS_ASIGN_ADDR command to assign data addresses without going online, the module writes the number of configured master control blocks to pfbMasNumBlks.

Size	Name	Offset	Description
BYTE	masStn	00h	Station address of slave
BYTE	MasCntCfg	01h	Control and configuration register
BYTE	MasStatus	02h	Status Register (Host only Reads)
BYTE	MasError	03h	Error indication
BYTE	MasEvent	04h	Event Flags
BYTE	MasDiagEvent	05h	Diagnostic Event Flags
BYTE	MasParmLen	06h	Parameters to slave length, in bytes
BYTE	MasChkLen	07h	Configuration check to slave length, in bytes
WORD	MasRxDataOfs	08h	Data received from slave offset within memory page
WORD	MasTxDataOfs	0Ah	Data to be sent to slave offset within memory page
BYTE	MasRxDataLen	0Ch	Data received from slave length, in bytes
BYTE	MasTxDataLen	0Dh	Data to be sent to slave length, in bytes
BYTE	MasSiemType	0Eh	Siemens Device Type
BYTE	MasExtErrInfo	0Fh	Extended error info
BYTE	masDesig[13]	10-1Ch	Slave designation (text)
BYTE	MasDiagMaxLen	1Eh	Maximum length of diagnostic status response
BYTE	MasDiagLen	1Fh	Diagnostic from slave length, in bytes
BYTE	masDiagSts1	20h	Status byte 1 from slave
BYTE	masDiagSts2	21h	Status byte 2 from slave
BYTE	masDiagSts3	22h	Status byte 3 from slave
BYTE	MasDiagMasStn	23h	Station that configured Slave
BYTE	masDiagID_hi	24h	ID hi byte sent back from slave
BYTE	masDiagID_lo	25h	ID lo byte sent back from slave
BYTE	masDiagData[24]	26-3Dh	Vendor-defined diagnostic info from slave
WORD	MasDiagExtOfs	3Eh	Offset within extension area if greater than 26 bytes; otherwise, last two bytes of diagnostics
BYTE	MasParmMasSts	40h	Status byte to slave
BYTE	masParmWdFact1	41h	Watchdog factor 1 to slave
BYTE	masParmWdFact2	42h	Watchdog factor 2 to slave
BYTE	MasParmRdyTme	43h	Response delay time (tbit) to slave

Size	Name	Offset	Description
BYTE	masParmID_hi	44h	ID value to slave hi
BYTE	masParmID_lo	45h	ID value to slave lo
BYTE	MasParmGrpId	46h	Group ID value for slave (not supported, always 0)
BYTE	masParmData[23]	47-5Dh	Parameters to slave
WORD	MasParmExtOfs	5Eh	Offset within extension area, if greater than 25 bytes
BYTE	masChkData[30]	60-7Dh	Configuration check values to slave
WORD	MasChkDataExtOfs	7Eh	Offset within extension area, if greater than 32 bytes

2.2.14.1 Slave Station Address

The host sets the slave station address by writing to masStn in the master control block for the slave. The station address range is 0 to 125.

2.2.14.2 Slave Control/Config Register (masCntCfg)

The host uses bits in the control and configuration register, located in the master control block for each slave, to set options for that slave. These bits must be set before going online, when configuring the slave. The only bit you can change online is MAS_CTL_IGNORE_STS.

When the host sets bit 0, MAS_CTL_IGNORE_STS, the module ignores the status of this master block in the DP master status display on the LED and in the global status register, pfbMasSts.

When the host sets bit 1, MAS_CTL_EVT_RX_CHG, the module generates an event in the event queue (and optionally an interrupt) when input data for this slave changes.

When the host sets bit 2, MAS_CTL_RX_BYTE_SWAP, the module swaps the order of bytes in received data.

When the host sets bit 3, MAS_CTL_EVT_UPDTE, the module generates an event in the event queue (and optionally an interrupt) when the slave has been updated.

When the host sets bit 4, MAS_CTL_FAIL_SAFE, the module communicates to this slave as a fail-safe slave.

Bit 5 is reserved.

When the host sets bit 6, MAS_CTL_TX_BYTE_SWAP, the module swaps the order of transmitted bytes.

When the host sets bit 7, MAS_CTL_ENABLE, the module enables this slave.

The module does not actually scan the slave until the module is put online. Setting this bit just tells the module that this slave is to be scanned when the card goes online.

2.2.14.3 Slave Status Register (masStatus)

The Slave Status register, located in the master control block for each slave, shows the slave's status. This register should not be changed by the host.

When the current status of the slave is OK, the module sets bit 7, MAS_STS_OK, otherwise, the module clears this bit.

2.2.14.4 Slave Error Register (masError)

The module sets various values in the error register, located in the master control block for the slave, to indicate the cause of any problems with the slave. Some errors occur while the slave is being parameterized; others occur at runtime. If there are multiple errors, only the last one is shown. The host acknowledges these errors by clearing the register.

Error	Value	Description
MAS_ERR_CFG_FAILURE	01h	Failure while trying to configure slave
MAS_ERR_SLV_ID_MISMATCH	02h	Slave's real ID does not match slave's configured ID
MAS_ERR_DATA_UPD_FAILURE	03h	Frame delivery problem while updating slave data
MAS_ERR_CFG_DIAG_READ_FAILURE	04h	Frame delivery problem while reading slave diag's
MAS_ERR_CFG_DIAG_STS1_ERR	05h	Error in diagnostic status byte #1 during configure
MAS_ERR_CFG_DIAG_STS2_ERR	06h	Error in diagnostic status byte #2 during configure
MAS_ERR_UPD_DIAG_STS1_ERR	07h	Error in diagnostic status byte #1 during diag read
MAS_ERR_UPD_DIAG_STS2_ERR	08h	Error in diagnostic status byte #2 during diag read
MAS_ERR_CFG_STN_MISMATCH	09h	Station address from diag read does not match
MAS_ERR_IO_CYC_TOUT	0Ah	Timeout waiting for I/O update
MAS_ERR_SLV_WD_OFF	0Bh	Warning: slave watchdog is not enabled

2.2.14.5 Extended Error Register (masExtErrInfo)

For some values in the error register, the module provides additional information in the extended error register to help pinpoint the cause of the problem.

If masError has the MAS_ERR_CFG_FAILURE value, the module puts one of the following values into masExtErrInfo.

Value	Description
01h	No response or NAK after sending the first diagnostic status request to the slave
02h	No response or NAK after sending parameter data to the slave
03h	No response or NAK after sending configuration check data to the slave
04h	No response or NAK after sending the second diagnostic status request to the slave
05h	Invalid response after sending the first diagnostic status request to the slave
06h	Invalid response after sending parameter data to the slave
07h	Invalid response after sending configuration check data to the slave
08h	Response to configuration check packet was non-zero length (slave should never return anything)
09h	Invalid response after sending the second diagnostic status request to the slave
0Ah	Timeout waiting for I/O update

If masError has the MAS_ERR_DATA_UPD_FAILURE value, the module puts one of the following values into masExtErrInfo.

Value	Description
01h	Error in data update during configuration
02h	No response or NAK when updating data while online

If masError has the MAS_ERR_CFG_DIAG_READ_FAILURE value, the module puts one of the following values into masExtErrInfo.

Value	Description
01h	Invalid response when reading slave diagnostics while online
02h	No response or NAK when reading slave diagnostics while online

masError = MAS_ERR_CFG_DIAG_STS1_ERR

If masError has the MAS_ERR_CFG_DIAG_STS1_ERR value, the value in masExtErrInfo depends on the value returned by the slave in the first station status byte when the master reads diagnostics during configuration. Mask the value with F5h, and any bits that are set in the result should not be set. Mask the value in masExtErrInfo with 02h, and bit 1 should be set.

The bits in station status byte 1 are:

Bit	Description
7	DP slave has been parameterized by another master
6	Slave received an invalid parameter frame, wrong Ident, wrong length, invalid parameters, etc.
5	Invalid response from the slave
4	Master requested a function that the slave does not support
3	An entry exists in the slave specific diagnostic area.
2	Configuration check data for the slave was incorrect
1	Slave is not ready for data transfer
0	DP slave non-existent

If masError has the MAS_ERR_UPD_DIAG_STS1_ERR value, the value in masExtErrInfo depends on the value returned by the slave in the first station status byte when the master reads diagnostics while online. Mask the value with F7h. The result should be 00h. The bits in station status 1 are shown in the above table.

If masError has the MAS_ERR_CFG_DIAG_STS2_ERR value, the value in masExtErrInfo depends on the value returned by the slave in the second station status byte when the master reads diagnostics during configuration. Mask the value with 80h. The result should be 00h. Mask the value in masExtErrInfo with 04h, and bit 2 should be set.

The bits in station status byte 2 are:

Bit	Description
7	Slave has been marked inactive by the master
6	Reserved
5	The slave has received a Sync command
4	The slave has received a freeze command
3	The slave watchdog has been activated
2	The slave sets this bit to 1
1	Slave is requesting a diagnostic read.
0	Slave is requesting reparameterization

If masError has the MAS_ERR_UPD_DIAG_STS2_ERR value, the value in masExtErrInfo depends on the value returned by the slave in the second station status byte when the master reads diagnostics while online. Mask the value with 80h. The result should be 00h. Mask the value in masExtErrInfo with 04h, and bit 2 should be set. The bits in station status 2 are shown in the above table.

The bits in station status byte 3 are:

Bit	Description
7	If this bit is set there exists more diagnostic information than specified in Ext_Diag_Data
6	Reserved
5	Reserved
4	Reserved
3	Reserved
2	Reserved
1	Reserved
0	Reserved

2.2.14.6 Diagnostic Event Register (masDiagEvent)

An online slave can request that the master read its diagnostics, which the module takes care of. The module then sets bit 0, MAS_DEVT_DIAG_UPD, in the diagnostic event register, which is located in the master control block for the slave. This indicates that the slave diagnostics have been read. The host acknowledges by clearing the register so that it can detect when another diagnostic update occurs.

2.2.14.7 Data Length and Location

One page of the module's shared memory is reserved for input data (from slaves) and a second is reserved for output data (to slaves). The host must set the lengths of input and output data for each slave. It writes the length of data, in bytes, to be received from the slave to masRxDataLen, and the length of data, in bytes, to be sent to the slave from masTxDataLen. Normally, the module assigns the offsets where the data is stored when the module is put online. The offset to the received data is stored in masRxDataOfs. The offset to the transmit data is stored in masTxDataOfs. Use the contents of these registers to access the data.

To force the module to assign the offsets without putting the card online, issue the CMD_MAS_ASIGN_ADDR command. You might, for example, want to do this if setting initial values for the data before putting the card online. Before issuing this command, set up all the master control blocks for all the slaves you will be scanning.

You can also manually assign the data offsets when configuring the module as a DP master. If you choose to assign data offsets rather than having the module assign them, set the PFB_MAS_CTRL_USR_OFS bit in the pfbMasCntrlCfg register and write the offset to the received data in masRxDataOfs, and the offset to the transmit data in masTxDataOfs for each slave. These offsets must be in the range 0 to 0x3ff8 and must be on 8-byte boundaries. If assigning offsets, assign the offset for ALL slaves. The module checks for overlaps in the data for different slaves before it goes online.

2.2.14.8 Parameter Data

The master sends parameterization data to the slave when bringing the slave online. This parameter data can be up to 32 bytes long (up to 244 bytes with extensions). The first 7 bytes are always sent. They consist of:

- Master status byte
- Slave watchdog factors (2 bytes)
- Slave response delay time
- Slave ID (2 bytes)
- Group ID

In addition, the master can send up to 25 bytes of extended parameter data (up to 237 bytes with extensions). For detailed information on the parameter data, refer to the Profibus Specification.

2.2.14.8.1 Master Status Byte (masParmMasSts)

The master uses bits in the master station status byte to lock/unlock access to the slave from other masters, to request sync/freeze support and to enable/disable the master watchdog. For furher details, refer to the Profibus Specification.

2.2.14.8.2 Slave Watchdog Factors

The master can set two parameters, masParmWdFact1 and masParmWdFact2, to set the master watchdog time for the slave. The watchdog time in milliseconds is calculated as:

wdtime = 10 * masParmWdFact1 * masParmWdFact2

where the two factors can range from 1 to 255.

If the slave does not receive any communication from the master within the watchdog time, it faults, the outputs are set to 0 and the slave gets reinitialized by the master.

2.2.14.8.3 Slave Response Delay

The master sends the minimum station response delay in masParmRdyTme. This is the minimum time, in Tbits, that the slave waits before it sends a reply to the master. It corresponds to pfbReadyTime in the local network parameters. If the value sent by the master in the parameter data is 0, the slave uses the value configured in its network parameters.

2.2.14.8.4 Slave Ident

Each Profibus DP slave module has a unique Ident assigned by the Profibus Trade Organization. The master sets the slave ID (IDENT number) using masParmID_hi and masParmID_lo. These are the high byte and the low byte, respectively, of the slave Ident number. If this does not match the Ident number of the slave, the slave will not communicate with the master.

2.2.14.8.5 Group ID (masParmGrpId)

The group ID is used in Profibus for functions such as sync and freeze. These functions are not supported by the module software and this byte should always be set to 0.

2.2.14.8.6 Extended Parameter Data

The master can send up to 25 bytes of extended parameter data to a slave. It must set this data when it is configuring the slave. What this parameter data is used for depends on what kind of device the slave is. To send the parameter data, the host writes the length to masParmLen and the parameter data itself to the array masParmData in the master control block for the slave.

The parameter data can be extended up to 244 bytes using the masCntrlExt area in the masCntrl page (the page that contains the master control blocks). This extended parameter data only appears when the slave has extended parameters. If the length is greater than 23, the first 23 bytes are stored in masParmData and masParmExtOfs contains the offset into the masCntrlExt area (3000h-3FFFh in the masCntrl page), where the remainder of the parameter data is found.

If you are manually entering extended parameter data, use pfbMasCntrlExtFree in the home page to keep track of the last offset used by any slave. This location is reserved for that purpose but it is up to you to update it, as the module does not use the location.

2.2.14.8.7 Configuration Check Data

The master sends up to 32 bytes (up to 244 bytes with extensions) of configuration check data to each slave during the slave's startup sequence. The configuration check data contains information about the number of inputs and outputs in each slot of the slave, whether the data is word data or byte data, and so on.

The check data can be extended up to 244 bytes using the masCntrlExt area in the masCntrl page. If the length is greater than 30, the first 30 bytes in masChkData contain configuration check data and masChkDataExtOfs contains the offset into the masCntrlExt area (3000h-3FFFh in the masCntrl page), where the remainder of the configuration check data is found.

If you are manually entering extended configuration check data, use pfbMasCntrlExtFree in the home page to keep track of the last offset used. Although this location is reserved for keeping track of the most recent offset, it must be updated manually, as the module does not use it.

2.2.14.8.8 Slave Diagnostic Data

Each slave returns the following diagnostic data to the master during the startup sequence. The slave can also request that the master read the diagnostic data from the slave while the slave is online and being scanned by the master.

The diagnostic data consists of:

- 3 station status bytes
- The station ID of the master that parameterized the slave (1 byte)
- The slave Ident number (2 bytes)
- The length of the extended diagnostics (1 byte)
- Up to 25 bytes of vendor-defined extended diagnostic data

The first 7 bytes of diagnostic data are always sent; the extended diagnostics are sent if the length byte is non-zero.

The module stores the station status bytes in masDiagSts1, masDiagSts2 and masDiagSts3. The module software maintains the station status bytes. For information on the meaning of bits within these status bytes (from which the following was obtained), refer to the Profibus Specification.

Octet 1: Station_status_1

- Bit 7: Diag.Master_Lock. The DP-Slave has been parameterized from another master. This bit is set by the DP master (class 1) if the address in octet 4 is different from FFh and from its own address. The DP Slave sets this bit to zero.
- Bit 6: Diag.Prm_Fault. This bit is set by the DP slave if the last parameter frame was faulty; for example, wrong length, wrong Ident_Number or invalid parameters.
- Bit 5: Diag.Invalid_Slave_Response. This bit is set by the DP master as soon as it receives a not plausible response from an addressed DP slave. The DP slave sets this bit to zero.
- Bit 4: Diag.Not_Supported. This bit is set by the DP slave as soon as a function that it does not support is requested.
- Bit 3: Diag.Ext_Diag. This bit is set by the DP slave. It indicates that a diagnostic entry exists in the slave-specific diagnostic area (Ext_Diag_Data).
- Bit 2: Diag.Cfg_Fault. This bit is set by the DP slave. If I/O configuration data doesn't match what the master sent it, this fault occurs.
- Bit 1: Diag.Station_Not_Ready. This bit is set by the DP slave if the DP slave is not yet ready for data transfer.
- Bit 0: Diag.Station_Non_Existent. This bit is set by the DP master if the corresponding DP slave cannot be reached. If this bit is set, the diagnostic bits contain the state of the last diagnostic message or the initial value. The DP slave sets this bit to zero.

Octet 2: Station_status_2

- Bit 7: Diag.Deactivated. This bit is set by the DP master as soon as the DP slave has been marked inactive within the DP slave parameter set and has been removed from cyclic processing. The DP slave always sets this bit to zero.
- Bit 6: Reserved
- Bit 5: Diag.Sync_Mode. This bit is set by the DP slave as soon as the corresponding DP slave has received the Sync command.
- Bit 4: Diag.Freeze_Mode. This bit is set by the DP slave as soon as the corresponding DP slave has received the Freeze command.
- Bit 3: Diag.WD_On (Watchdog on). This bit is set by the DP slave. If this bit is set to 1, the watchdog control on the DP slave has been activated.
- Bit 2: This bit is set to 1 by the DP-Slave.
- Bit 1: Diag.Stat_Diag (static diagnostics). If the DP slave sets this bit, the DP master fetches diagnostic data as long as this bit is reset again. For example, the DP slave sets this bit if it is not able to provide valid user data.
- Bit 0: Diag.Prm_Req. If the DP slave sets this bit, the respective DP slave is reparameterized and reconfigured. The bit remains set until parameterization is finished. If bit 1 and bit 0 are set, bit 0 has the higher priority.

Octet 3: Station_status_3

- Bit 7: Diag.Ext_Diag_Overflow. If this bit is set, more diagnostic information exists than specified in Ext_Diag_Data. For example, the DP slave sets this bit if there are more channel diagnostics than the DP slave can enter in its send buffer; the DP master sets it if the DP slave sends more diagnostic information than the DP master can enter in its diagnostic buffer.

- Bits 0-6: Reserved. In masDiagMasStn, the module stores the station number of the master that parameterized the slave.

  In masDiagID_hi (high byte) and masDiagID_lo (low byte), the module stores the slave Ident number returned by the slave.

  In masDiagLen, the module stores the total current length of the diagnostics, and in masDiagMaxLen, it stores the maximum possible diagnostics length. If the maximum diagnostic length is less than 32, the module stores the extended diagnostic data in the masDiagData array, with the final two bytes in masDiagExtOfs. If the maximum length is greater than 32, the first 24 bytes of extended data are stored in masDiagData, and masDiagExtOfs contains the offset into the masCntrlExt area (3000h-3FFFh in the masCntrl page) where the remainder of the diagnostic data is found.

  Always check the current length in masDiagLen before reading the diagnostics, as the length of the diagnostic data sent by the slave may change.

2.2.14.8.9 Slave Designation Data

The host can assign a designation string of up to 12 characters to each slave, which it stores in the master control block's masDesig array for the slave. This data is not sent out on the network to the slave. The host application can use it to associate a specific slave with a configuration block, then access the slave by name rather than by block number. For example, even if block numbers or station numbers changed, your application could still scan through the block list to find the block associated with the name in the slave designation data area.

2.2.15 ID Response Text

For a description, refer to Section 2.2.19, ID Response Fields.

2.2.16 DP Slave Control, Status and Data

As a slave on a DP network, the module can transmit and receive up to 244 bytes of data. All variables and tables for using the module as a DP slave are contained in the PROFI_USR structure and are found on the module memory homepage. If you are using the module as a DP slave only, set the network parameters before putting the card online.

2.2.16.1 DP Slave Control and Configuration

The host controls and sets options for the DP slave by setting bits in the Config/Status register, slvCntCfg. The host must set the bits that configure the DP slave operation before it puts the module online. The host also sets bits in this register to perform DP slave operations while the module is online.

To request a diagnostic read from the master, the host sets bit 0, SLV_CTL_DIAG_UPD, while it is online. Setting this bit causes the module to send its next reply to the master as a high-priority data packet. When the master sees this packet, it reads the slave's diagnostic status.

To tell the module to generate an event in the event queue when received data changes, the host sets bit 1, SLV_CTL_EVT_RX_CHG, when it is configuring the slave. For more information on processing events, refer to Sections 2.2.4, Event Control, and 2.2.5, Interrupt Control. Do not set both this bit and the SLV_CTL_EVT_UPDTE bit in the slvCntCfg register.

To force the module to use pfbReadyTime from the module network parameters and ignore the value the master sends (slvReadyTime), the host sets bit 2, SLV_CTL_FORCE_READY_TIME, when configuring the slave. If this bit is 0 (the default), and there is a mismatch in the ready times, the module returns an error and does not try to communicate any further with the slave.

To tell the module how to respond when the master requests sync or freeze during slave parameterization, the host sets bit 3, SLV_CTL_IGN_SYNC_FRZ_ERR, while configuring the slave. If the bit is 0 (the default), the module replies to the master that sync and freeze are not supported and the master does not try to communicate further with the slave. If the bit is set to 1, the module does not report that it does not support sync or freeze. It then ignores any subsequent sync or freeze commands from the master and updates outputs when they are received.

To tell the module to swap the upper and lower bytes of received data, the host sets bit 4, SLV_CTL_RX_BYTE_SWAP, when configuring the slave. Note that transmitted data goes out directly. If you need to swap the bytes of transmit data, your application must swap the bytes before it writes them to the module. If you enable byte-swapping on received data, there is a performance penalty, as the module's processor must constantly swap bytes before it writes them to the data table. As a result, the default is 0 and the bytes are not swapped. If you have byte and word values, it is advisable to swap bytes in your application.

To tell the module to generate an event whenever the slave is updated by the master, the host sets bit 5, SLV_CTL_EVT_UPDTE, while configuring the slave. Do not set both this bit and the SLV_CTL_EVT_RX_CHG bit in the slvCntCfg register.

To tell the module to generate an event and optionally an interrupt when the master mode changes, for example from run to stop or from stop to run, the host sets bit 6, SLV_CTL_EVT_MODE_CHANGE, when configuring the slave.

To tell the module not to generate an event if there is an error with the slave, the host sets bit 7, SLV_CTL_IGNORE_STS, when configuring the slave. Setting this bit forces the LED status display for the DP slave to indicate that there are no problems.

To disable the status LED for the DP slave function, the host sets bit 14, SLV_CTL_DIS_LED, when configuring the slave.

To tell the module that the DP slave function is to be enabled, the host sets bit 15, SLV_CTL_ENABLE. When you put the module online, this bit tells the module that you are using the DP slave function.

2.2.16.2 DP Slave Status Register

By setting bits in the slvStatus register, the module reports the status of the DP slave operation.

The module sets bit 6, SLV_STS_RUN_MODE, if it is being scanned by a DP master in run mode.

If the current slave status is OK, the module sets bit 7, SLV_STS_OK. This means that the parameters were set successfully and the slave watchdog has not timed out.

2.2.16.3 DP Slave Error Register

To report various error conditions, the module sets the slave error register, slvError, to the following values. If there are multiple errors, the register holds the value for the last error encountered.

Error	Value	Description
SLV_ERR_ID_MISM	01h	ID from master does not match configured ID
SLV_ERR_READY_TIME_MISM	02h	pfbReadyTime does not match what master sent
SLV_ERR_UNSUP_REQ	03h	pfb is requesting Freeze or Sync, which is not supported
SLV_ERR_RX_LEN_MISM	04h	Length of data from master to slave is incorrect
SLV_ERR_TX_LEN_MISM	05h	Length of data from slave to master is incorrect
SLV_ERR_WD_FACT_INV	06h	slvWdFact1 or slvWdFact2 from master was 0
SLV_ERR_TIME_OUT	07h	Slave watchdog timeout (check response timeout)
SLV_ERR_WARN_WD_DIS	08h	Slave timeout watchdog disabled from master

If the value is SLV_ERR_ID_MISM, the slave ID set in slvID_hi and slvID_lo does not match the slave ID configured in the master. Check slvMasID_hi and slvMasID_lo to determine what values the master expects. If there is a mismatch, the slave will not communicate with the master.

If the value is SLV_ERR_READY_TIME_MISM, the ready time for the module is different from the value configured in the master. To determine the value configured in the master, read slvReadyTime. While you are configuring the slave, set the SLV_CTL_FORCE_READY_TIME bit in the slvCntCfg register to override the value sent by the master. The module can communicate as a slave even if the times are different; however, you may experience network errors.

If the value is SLV_ERR_UNSUP_REQ, the master requested Sync or Freeze during parameterization, which the module does not support. If it is SLV_ERR_RX_LEN_MISM, the data received from the master has a length different from the length configured on the module. Read slvReqRxDataLen to determine the Receive data length requested by the master. If there is a receive length mismatch, the module will not communicate as a slave.

If the value is SLV_ERR_TX_LEN_MISM, the master has requested data from the slave with a length different from the length configured for the slave. Read slvReqTxDataLen to determine the Transmit data length requested by the master. If there is a transmit length mismatch, the module will not communicate as a slave.

If the value is SLV_ERR_WD_FACT_INV, one of the two slave watchdog factors is zero, which is not allowed.

If the value is SLV_ERR_TIME_OUT, the slave's watchdog timed out. The slave goes offline and must be reinitialized by the master.

If the value is SLV_ERR_WARN_WD_DIS, the master has disabled the slave watchdog.

2.2.16.4 DP Slave Event Register

Whenever it generates a DP slave event, the module sets bits in the DP slave event register, slvEvent.

When the master sends an I/O update to the slave, the module sets bit 0, SLV_EVT_UPD.

The module sets bit 1, SLV_EVT_RX_DATA_CHG, when the data received from the master has changed.

When an event occurs, the module sets the appropriate bit, then puts the event into the event queue. The host acknowledges by processing the event and then clearing the slvEvent register.

The module will not generate a new received data change or update the event until this byte has been cleared.

2.2.16.5 Sending Diagnostic Data

When the module is online as a DP slave, it can request that the master read its slave diagnostic data. To enable this feature, set the SLV_CTL_DIAG_UPD bit in the slvCntCfg register.

When the master has read the diagnostic data, the module sets bit 0, SLV_DEVT_DIAG_UPD, in the DP slave diagnostic event register, slvDiagEvent.

The host application acknowledges that the master has read the diagnostic data by clearing the SLV_CTL_DIAG_UPD bit in the slvCntCfg register (so that the module can recognize a new request from the host to update the diagnostics), and by clearing the slvDiagEvent register to 0.

2.2.16.6 Received Data Length

To set the received data length, the host writes to the slvRxDataLen register. This is the length of the data from the master, in bytes. The value can range from 0 to 244. The default length is 0.

2.2.16.7 Requested Receive Data Length

This register holds the value of the receive length requested by the master.

2.2.16.8 Transmitted Data Length

To set the transmit data length, the host writes to the slvTxDataLen register. This is the length of the data to the master, in bytes. The value can range from 0 to 244. The default length is 0.

2.2.16.9 Requested Transmit Data Length

The slvReqTxDataLen register holds the value of the transmit length requested by the master.

2.2.16.10 Check Configuration Length

The slvChkLen register holds the length of the configuration check data.

2.2.16.11 Parameter Data Length

The slvParmLen register holds the length of the parameter data from the master.

2.2.16.12 Master Control Commands

The master can send special control commands to one slave or to a group of slaves. These commands include sync, freeze and a clear data command. The slave accepts these commands from only the master that has parameterized it. For details, refer to the Profibus Specification. The module stores any such received command in the slvGlbCntrl[2] array.

2.2.16.13 Slave Diagnostic Data

The slave can return up to 32 bytes of diagnostic data to the master. The diagnostic data is sent to the master as part of the initial startup. The diagnostic data can also be sent when the module is online, but only when the slave requests that the master read it. The first 7 bytes of diagnostic data are always sent to the master and consist of:

- The three station status bytes
- The ID of the master that parameterized the slave
- The slave Ident value (two bytes)
- The length of the extended diagnostics

The remainder is extended diagnostics and is user defined. The extended diagnostics are sent if the extended diagnostic length is greater than 1.

2.2.16.13.1 Station Status Bytes

The three station status bytes are maintained by the module and should not be written to by the host. For detailed information on what these bytes contain, refer to the Profibus Specification. The station status bytes are:

- Station status 1, slvSts1
- Station status 2, slvSts2
- Station status 3, slvSts3

2.2.16.13.2 Master of the Parameterized Slave

The module stores the station number of the master that set the slave's parameters in slvMasStn. When the master configures the module as a DP slave, the module writes the value and this register should not be written to by the host application.

2.2.16.13.3 Slave Ident Number

The slave ID is the Ident value returned to the master by the slave. It is a 4-digit hexadecimal number and is stored in two consecutive bytes, slvID_hi (default 08h) and slvID_lo (default 55h). The Ident number is unique to each Profibus I/O device. If you want to emulate another device, change the Ident number on the module from the default.

2.2.16.13.4 Extended Diagnostics

The slave can send up to 25 bytes of extended diagnostic data to the master. This data is all user defined. The extended diagnostic data is typically used for device-specific fault information, but can be employed for other purposes as well.

The host must write the length of the extended diagnostic data to slvDiagLen. The diagnostic length can be set only when configuring the slave. The slave always sends the byte that contains the length of the extended diagnostics to the master.

The value of the length includes the length byte itself. For example, if you want to send 10 bytes of diagnostic data to the master, set slvDiagLen to 11. If the length is 1, the slave sends no extended diagnostic data to the master, just the length itself.

The host writes the extended diagnostic data to the slvDiag[25] array.

2.2.16.13.5 Sending Diagnostic Data While Online

When the module is online as a DP slave, it can request that the master read its slave diagnostic data. To send this request, set the SLV_CTL_DIAG_UPD bit in the slvCntCfg register.

2.2.16.14 Master Parameter Data

As part of the startup process, the master sets the slave's parameters. The parameter data sent by the master includes:

- The master station status byte
- Two factors for setting the watchdog control time
- The station delay time configured in the master
- The slave station Ident configured in the master
- The group ID value
- Additional, module-specific parameter information

2.2.16.14.1 Master Station Status

The module stores the master station status register in slvMasSts. For detailed information on what this register contains, refer to the Profibus Specification.

2.2.16.14.2 Watchdog Factors

The slave watchdog ensures that if the master fails or the slave loses communication with the master, the slave goes into to a safe state.

If the slave times out, the module clears the inputs to zero. When the master can again communicate with the slave, it reconfigures the slave.

The slave watchdog is enabled/disabled by the master and the slave watchdog time is determined by slvWdFact1 and slvWdFact2. The values can each range from 1 to 255.

The slave watchdog time in milliseconds is calculated as:

Twd = 10 * slvWdFact1 * slvWdFact2

2.2.16.14.3 Station Delay

The slvReadyTime is the value configured in the master and is the minimum time the slave waits until it sends a response to the master. If the value is 0, the previous value remains unchanged. Values are in Tbits, and allowed values are 1 to 255. The module defaults to the network setting.

The host can force the module to override the value from the master and use the value from the card's own network parameters by setting the SLV_CTL_FORCE_READY_TIME bit in the slvCntCfg register. For details, refer to Section 2.2.16.1, DP Slave Control and Configuration. Generally, you would use the same value everywhere on the network, and the same value on the slave as on the master.

2.2.16.14.4 Slave Ident Number from Master

The slave accepts parameter frames (messages) only if the Ident number from the master is equal to the slave's own Ident number. The module stores the Ident number configured in the master in slvMasID_hi and slvMasID_lo. If there is a mismatch between the slave Ident number and the value configured in the master, the slave will not communicate with the master.

2.2.16.14.5 Group Ident

The master sends a group ID value, which the module stores in slvGrpId. However, the module does not support group Idents.

2.2.16.14.6 Extended Parameter Data

The master can send up to 25 additional bytes of parameter data to the slave. The length of parameter data is stored in slvParmLen, and the additional parameter data itself is stored in the array slvParm[25]. The use of this parameter data depends on the particular I/O module.

2.2.16.15 Configuration Check Values

As part of the startup sequence, the master sends up to 32 bytes of configuration check data to the slave. This check data contains information such as the number of inputs and outputs. Typically, each byte of configuration check data represents one slot. For detailed information on what the configuration check data contains, refer to the Profibus Specification.

The module stores the length of the configuration check data in the slvChkLen byte and the configuration check data itself in the slvChk array.

When the module receives the configuration check data, it compares the total of the lengths in the configuration check data with the expected transmit and receive data lengths configured for the slave. If they do not match, the module indicates an error in its final diagnostic reply to the master and sends an error to the host application.

2.2.16.16 Received Data

The data received from the master is stored in the PROFI_USR structure's slvRxData table.

If you have set the SLV_CTL_RX_BYTE_SWAP bit in the slvCntCfg register, the module swaps high and low bytes of received data before it writes them to the data table.

2.2.16.17 Transmit Data

The data transmitted to the master is stored in the PROFI_USR structure's slvTxData table.

If you need to swap data bytes in transmitted data, your application must do the swapping before it writes the data to the module.

2.2.17 Event Queue

For a description, refer to Section 2.2.4, Event Control.

2.2.18 Trigger Queue

For a description, refer to Section 2.2.20.3, The Trigger Queue.

2.2.19 ID Response Fields

The module can send a request Ident with the reply message to read a remote station's ID. The module can also respond to such a request from another station.

To read a remote station's ID, write the remote station number to pfbIdStn, set bit 0 in pfbIdReq to 1 and wait for the module to clear the bit. If there was an error reading the station ID, the module reports the error in pfbIdRspSts. Otherwise, the module writes the total length of the returned data in pfbIdRspLen, the lengths of each of the four fields returned in the array pfbIdFldLen[4] and the text of the response in pfbIdText[242]. Use the lengths to decode the text into the four fields.

When a remote station requests the module's ID, the module returns the string in pfbLocIdUsrStr[112], unless it is null. If it is null, the module shall return

```
Copyright (c) 1995-2002 SST/Woodhead Canada Ltd.
For SST-PFB3-PCI Card
Version x.xx
```

in response to an ID request, where x.xx is the PFBSCAN module's version number. This text must be written before the module is put online.

2.2.19.1 Master Status Cross-Reference Table

The Master Status Cross-Reference table, pfbMasStsTab, is organized by station number and shows the status and master control block number for each station number. There is one word for each station. If the upper byte in the word is set, the station is being scanned with no errors. The lower byte of the word contains the master control block associated with that station number. If the low byte is FFh, the slave is not configured.

2.2.20 FDL (Layer 2) Messaging

The module can send and receive FDL (Layer 2) messages. Specifically, it is able to:

- Configure up to 64 Service Access Points (SAPs)
- Update timeout on any SAP
- Support up to 128 request blocks at one time
- Support periodic and one-shot requests
- Generate events and optional interrupts on received data changes

To configure Layer 2 SAPs and message blocks, the module must be offline. Ensure that the network parameters have been configured before putting the module online For information on what the network parameters are and how to set them, refer to Section 2.2.6, ASPC2 Profibus Controller Basic Parameters.

To send FDL messages, the module must be an active station on the network. You can set up and access SAPs even if the module is a passive station.

2.2.20.1 FDL Global Control Register

The global control register, lay2Cntrl, contains bits that enable the FDL functions. These bits must be set when the module is offline.

When the host sets bit 12, LAY2_CTL_MSG_DIS_LED, the module disables the display of the Layer 2 message status on the system status LED.

When the host sets bit 13, LAY2_CTL_SAP_DIS_LED, the module disables the display of the Layer 2 SAP status on the system status LED.

When the host sets bit 14, LAY2_CTL_SAP_ENABLE, the module enables the Layer 2 SAP service. The host must set this bit before it issues the online command.

When the host sets bit 15, LAY2_CTL_MSG_ENABLE, the module enables the Layer 2 Message service. The host must set this bit before it issues the online command.

2.2.20.2 FDL Global Status Register

To show the overall status of Layer 2 operations, the module sets bits in the FDL global status register, lay2Status.

The module sets bit 14, LAY2_STS_ALL_SAP_OK, to indicate that all configured Layer 2 SAPs are OK.

The module sets bit 15, LAY2_STS_ALL_MSG_OK, to indicate that all configured Layer 2 messages are OK.

2.2.20.3 The Trigger Queue

The host controls message transmission and online changes to SAPs through the trigger queue, pfbTrigQueue[256], in the PROFI_USR structure.

2.2.20.3.1 pfbTrgHead

The host accesses the trigger queue via the trigger queue head pointer, pfbTrgHead. To insert triggers, the host puts the appropriate trigger value in the trigger queue at the head position, and then increments the head pointer.

2.2.20.3.2 pfbTrgTail

The module removes triggers from the queue at the tail position and increments the tail pointer, pfbTrgTail. The host can use the tail pointer to determine if the queue is full. If the head pointer + 1 = the tail pointer, the queue is full.

2.2.20.4 FDL (Layer 2) SAPs

The module supports up to 64 service access points. You configure an SAP by creating an SAP control block.

2.2.20.4.1 SAP Control Blocks

The Lay2sCntrl[64] array contains the SAP control blocks. Each SAP control block corresponds to a specific SAP number, for example, block 3 corresponds to SAP 3. The SAP control block structures are defined in the following table.

Size	Name	Offset	Description
BYTE	lay2sType	00h	Type of SAP (DP slave, Layer 2, or FMS)
BYTE	lay2sStn	01h	If it is a strict station, accept updates only from this station
BYTE	lay2sSap	02h	If it is a strict SAP, accept updates only from this SAP
BYTE	lay2sSrcSap	03h	Source SAP from request
WORD	lay2sCntCfg	04h	Control and Configuration register
BYTE	lay2sStatus	06h	Status register (Host Reads Only)
BYTE	lay2sError	07h	Error register
BYTE	lay2sEvent	08h	Events
BYTE	lay2sSrcStn	09h	Source station from request
WORD	lay2sTimeOut	0Ah	Timeout for SAP * 10µs (1- 8190, 0=disable)
BYTE	lay2sFrmCntrl	0Ch	If it is a strict FC, accept updates only from specified Frame Control values
BYTE	lay2sRspStatus	0Dh	Response Status if NOT_OK
BYTE	lay2sRxLen	0fh	Actual receive data length, in bytes
BYTE	lay2sRxMaxLen	10h	Receive data (from request) max length, in bytes
BYTE	lay2sTxLen	11h	Transmit data (response) length, in bytes
WORD	lay2sRxDataOfs	12h	Receive data (from request) offset on the mem page
WORD	lay2sTxDataOfs	14h	Transmit data (response) offset on the mem page
BYTE	lay2sRxDataPage	16h	Receive data (from request) on the data mem page
BYTE	lay2sTxDataPage	17h	Transmit data (response) from the data mem page
BYTE	lay2sTxMaxLen	18h	Transmit the data (response) length, in bytes
BYTE	lay2sNumAcyBufs	19h	Maximum transmit data (response) length, in bytes

2.2.20.4.2 Creating an SAP

To configure an FDL SAP, the host must perform the following steps. The various registers in the SAP control block are described in the sections that follow.

1. Select the SAP to be configured.
2. Set the SAP type to LAYER2_SAP by writing to lay2sType.
3. Set the maximum receive data length in lay2sRxMaxLen.
4. Set the maximum transmit (reply) data length in lay2sTxmaxLen and the current transmit length in lay2sTxLen.
5. Set the SAP timeout, if required, in lay2sTimeOut.
6. Set whether the SAP should accept only:
    - Messages from a specific station (lay2sStn)
    - Messages from a specific source SAP (lay2sSap)
    - Messages of a specific type (lay2sFrmCntrl)
7. Set any required options for the SAP by setting bits in lay2sCntCfg:
    - Byte swapping on received data
    - Event on received data change
    - Event on SAP update
    - Ignore status of this SAP

The host configures all required SAPs and any other module operations (DP master, DP slave, FDL messages) then puts the module online. When the module goes online, it allocates the memory locations for the SAP's receive and transmit data. The host can then access the data, using the page and offset values provided by the module.

2.2.20.4.3 SAP Control and Configuration Register

The host configures various options for each SAP by setting bits in the control and configuration register, lay2sCntCfg. These bits must be set when you are configuring the module offline, except for the LAY2S_CTL_IGNORE_STS bit, which can be changed while you are online.

When the host sets bit 2, LAY2S_CTL_RX_BYTE_SWAP, the module swaps the bytes in words of received data. If the host needs to swap the byte order in transmitted data, it must swap the bytes itself before it writes data to the module.

If the host sets bit 3, LAY2S_CTL_EVT_RX_CHG, the module generates an event in the event queue when the SAP's received data changes.

Note

Do not enable both received data change and SAP update events.

When the host sets bit 4, LAY2S_CTL_IGNORE_STS, the module ignores the status of this SAP block in the LED status display for Layer 2 SAPs, and also in the global SAP bit in the Layer 2 status register, lay2Status.

If the host sets bit 5, LAY2S_CTL_EVT_UPDTE, the module generates an event in the event queue whenever the SAP is updated. Do not enable both received data change and SAP update events.

2.2.20.4.4 SAP Type

The host allocates an FDL SAP by setting the type to LAY2S_TYP_LAYER2_SAP in the lay2sType register. The host should first check that the SAP has not already been used by a DP slave on this module.

Possible SAP types are:

Type	Value
LAY2S_TYP_NOT_DEF	00h
LAY2S_TYP_DP_SLAVE	01h
LAY2S_TYP_LAYER2_SAP	02h
LAY2S_TYP_DP_MAS_MC2	05h

2.2.20.4.5 Strict Station Checking

If the host sets lay2sStn to a valid station number, the module accepts messages from that specific station only. Otherwise, if it is set to FFh, the module disables strict station checking. The valid station range is 0 to 126.

If you also select source SAP checking (see next section), set the upper bit in lay2sStn.

2.2.20.4.6 Strict Source SAP Checking

If the host sets lay2sSap to a valid SAP number, the module accepts messages from that specific source SAP only. Otherwise, if it is set of FFh, the module disables strict source SAP checking. The valid range of source SAPs is 0 to 63.

If you are also using strict station checking, set the upper bit in lay2sStn.

2.2.20.4.7 Strict Frame Control Checking

If the host sets lay2sFrmCntrl to a valid message type, the module accepts that specific message type only.

The default is LAY2S_FC_ALL (0) and the SAP accepts all messages.

Type	Value	Description
LAY2S_FC_ALL	00h	Accept all requests
LAY2S_FC_SDN_LO	01h	Accept only SDN lo priority
LAY2S_FC_SDN_HI	02h	Accept only SDN hi priority
LAY2S_FC_SDN_LO	03h	Accept only SDN hi or lo priority
LAY2S_FC_SDA_LO	04h	04h Accept only SDA lo priority
LAY2S_FC_SDA_HI	05h	Accept only SDA hi priority
LAY2S_FC_SDA_LO_HI	06h	Accept only SDA hi or lo priority
LAY2S_FC_SRD_LO	09h	Accept only SRD lo priority
LAY2S_FC_SRD_HI	0Ah	Accept only SRD hi priority
LAY2S_FC_SRD_LO_HI	0Bh	Accept only SRD hi or lo priority
LAY2S_FC_DDB_REQ	0Ch	N/A
LAY2S_FC_DDB_REQ_LO	0Dh	N/A
LAY2S_FC_DDB_REQ_HI	0Eh	N/A
LAY2S_FC_DDB_REQ_LO_HI	0Fh	N/A

2.2.20.4.8 Received Data

The host sets the maximum received data length in bytes for the SAP by writing to lay2sRxMaxLen.

When going online, the module assigns a location for storing the SAP's received data. The module writes the data page in lay2sRxDataPage and the offset within that page in lay2sRxDataOfs. The offset is always on an 8-byte boundary.

When the SAP is updated, the module writes the length of the actual received SAP data in lay2sRxLen. If the module receives a message with a data length greater than the value in lay2sRxMaxLen, it replies with a NAK (RR) and discards the data.

2.2.20.4.9 Transmit Data

The host sets the maximum response data length in bytes for this SAP by writing to lay2sTxMaxLen, and the current reply length in lay2sTxLen. Usually these are equal. However, the value in lay2sTxLen can be less than lay2sTxMaxLen. While online, change the value in lay2sTxLen to vary the reply length.

If the host tries to set the reply length to a value greater than the maximum length, the module generates an error and does not change the current length. The allowed range of values for lay2sTxMaxLen and lay2sTxLen is 0 to 244.

When going online, the module assigns the location for response data. The module writes the data page for the response data in lay2sTxDataPage and the offset within that page in lay2sTxDataOfs. The offset is always on an 8-byte boundary. Transmit data will be sent only in response to an SRD command.

2.2.20.4.10 Timeout

The host sets the timeout for the SAP by writing to lay2sTimeOut. The timeout is in units of 10 ms and must be in the range of 1 to 8190. The maximum timeout is therefore 81900 ms, or 81.9 seconds. If the SAP is not updated within the timeout period, the module generates an error. To disable the timeout for this SAP, write 0 to this register.

2.2.20.4.11 SAP Updates

Whenever a SAP is updated, the module writes the station number of the station that updated the SAP in lay2sSrcStn, and the source SAP from the station that updated the SAP in lay2sSrcSap.

2.2.20.4.12 Receiving Broadcast/Multicast Messages

To receive broadcast or multicast messages, turn off strict station checking. To receive broadcast messages, configure a SAP on SAP number 63.

2.2.20.4.13 Online Changes

Change the following SAP elements when online:

- Reply length
- Strict station checking
- Strict source SAP checking
- Strict frame control checking

When making online changes, the host must enter a SAP update trigger (TRG_SAP_UPDATE, 1800h) in the trigger queue. The lower byte of this event should contain the SAP number. Refer to Section 2.2.20.3, TheTrigger Queue, for information on using the trigger queue.

2.2.20.4.14 Status Register

The module uses the lay2sStatus register to indicate the operating status of the SAP.

If the SAP has no operating problems, the module sets bit 7, LAY2S_STS_OK, of lay2sStatus. If this bit is 0, there is a problem with the SAP.

2.2.20.4.15 Response Status

The response status, lay2sRspStatus, is returned by the LAN controller when it encounters an error. Mask the value with BFh and use the following table to determine the meaning of the resulting value. X indicates “do not care” values in the upper half of the byte.

Value	Description
00h	OK
01h	User error, SAP locked
02h	No resource for send data, tried to send to SAP that was not configured
03h	No service available (SAP does not exist)
04h	Access point blocked
80h	Short character, problems with wiring, termination, etc.
9Fh	No access
Afh	Double token detected, problems with wiring, termination, etc.
BFh	Response buffer too small
8Fh	Noise at SM command, problems with wiring, termination, etc.

2.2.20.4.16 Error Register

The error register, lay2sError, indicates the cause of the SAP problem and may contain one of the following values:

Error Message	Value
LAY2S_ERR_NOT_OK	01h
LAY2S_ERR_TIME_OUT	02h
LAY2S_ERR_RX_LEN	03h
LAY2S_ERR_TX_LEN	04h
LAY2S_ERR_BAD_PARAM	05h

If there are multiple errors, this register contains the last error that occurred.

2.2.20.4.17 SAP Events

In the event queue, the module can generate various events related to FDL SAPs. The following table shows the upper byte of the event queue entry for the event. The lower byte contains the number of the SAP that generated the event.

Error Message	Value
EVT_LY2S_RX_DATA_CHG	60NNh
EVT_LY2S_UPDATE	61NNh
EVT_LY2S_ERROR	62NNh

To notify the host that events related to this SAP have occurred, the module sets bits in the event register, lay2sEvent.

The module sets bit 0, LAY2S_EVT_INDICATION, to indicate that data for the SAP has been received.

To indicate that data for the SAP has been received and that it is different from the previous data, the module sets bit 1, LAY2S_EVT_RX_DATA_CHG. The host acknowledges the event by clearing the bit. The module does not generate another received data change or update event for the SAP unless the lay2sEvent register is 0. Instead, increments the event overrun counter.

2.2.20.4.18 SAP Diagnostic Counters

The module also maintains diagnostic counters that show the operation Layer 2 SAPs. Refer to Section 2.2.9.4, FDL (Layer 2) Statistics, for a description of these counters.

2.2.20.5 FDL (Layer 2) Messages

The module can be used to send FDL messages. To send a message, set up a control message block for the message before putting the module online. The module supports up to 128 message blocks. Messages are sent by putting them in the trigger queue (pfbTrigQueue). Messages can be set up as:

- One-shot
- Periodic, with periods from 1 µs to 16.4 seconds

The module handles the details of rescheduling and sending periodic messages. It must be an active station on the network to send messages.

2.2.20.5.1 Message Control Blocks

The Lay2mCntrl array in the PROFI_USR structure contains the FDL (Layer 2) message control blocks. The message control block structures are in the table below.

Size	Name	Offset	Description
BYTE	lay2mCntCfg	00h	Control and configuration register
BYTE	lay2mState	01h	Current state of message block
BYTE	lay2mStatus	02h	Status register (Host only Reads)
BYTE	lay2mError	03h	Error register
BYTE	lay2mEvent	04h	Events
WORD	lay2mUpdTime	06h	Update interval for periodic * 1ms (1-16380)
WORD	lay2mErrTime	08h	Retry interval for periodic if error occurs * 1µs
BYTE	lay2mDstStn	0Ah	Destination station address (or with 80h to enable DstSap)
BYTE	lay2mDstSap	0Bh	Destination SAP (FFh to disable)
BYTE	lay2mSrcSap	0Ch	Source SAP (FFh to disable)
BYTE	lay2mFrmCntrl	0Dh	Frame control byte
BYTE	lay2mRspStatus	0Eh	Response Status if not OK
BYTE	lay2mRxLen	0Fh	Actual receive data length in bytes
BYTE	lay2mRxMaxLen	10h	Max receive data (from response) max length in bytes
BYTE	lay2mTxLen	11h	Transmit data (request) length in bytes
WORD	lay2mRxDataOfs	12h	Receive data (from response) offset within Layer 2 data mem page
WORD	lay2mTxDataOfs	14h	Transmit data (request) offset within Layer 2 data mem page
BYTE	lay2mRxDataPage	16h	Receive data (from response) data mem page
BYTE	lay2mTxDataPage	17h	Transmit data (request) data mem page
BYTE	lay2mTxMaxLen	18h	Max transmit data (request) length in bytes

The message control block consists of the following components that must be set by the host before the message is sent:

- The destination station in lay2mDstStn
- The destination SAP in lay2mDstSap (or 255 to lay2mDstSap to disable sending the destination SAP)
- Source SAP in lay2mSrcSap (or 255 to lay2mSrcSap to disable sending the source SAP)
- The message type in lay2mFrmCntrl, the maximum transmit data length in lay2mTxMaxLen and the current length in lay2mTxLen
- The maximum receive data length in lay2mRxMaxLen
- Any required options for the message, by setting bits in the lay2mCntCfg register
- Whether the message should generate an event on a received data change or message confirmation
- Whether the message is periodic
- Whether the message should be retried on errors
- Whether the module should give the message high priority processing in its internal queue
- Whether the received data should be byte swapped
- Whether the module should ignore the status of this message
- For periodic messages, the message update time in lay2mUpdTime
- For messages that are to be retried on errors, the message retry time, in lay2mErrTime

The only elements that must be set before the host puts the module online are the maximum transmit data length and the maximum receive data length. The module needs these two values to allocate memory for the message. However, the host must set the other components before putting the message in the queue to be sent. The various elements of the message control blocks are described in the following sections.

After the host configures all required message blocks and any other required module operations (FDL SAPs, DP master, DP slave), it puts the module online. The module then allocates the transmit and receive data locations for the message block. The host can fill in any missing components of the message block and put the message in the message queue.

2.2.20.5.2 Control and Configuration Register

The host configures options for the message block by setting various bits in the control and configuration register, lay2mCntCfg. These bits can be changed while the module is online. The module checks the state of this register each time it sends a message from the block.

If the host sets bit 0, LAY2M_CTL_PERIODIC, the module makes the message periodic. If the message is periodic, the host must also write the message update time to lay2mUpdTime (see below).

If the host sets bit 1, LAY2M_CTL_RETRY_PERIODIC, the module retries the message if there are problems. If the module is to retry the message, the host must also write the error retry time to lay2mErrTime (see below).

If the host sets bit 2, LAY2M_CTL_HI_PRI, the module sends the message to the high priority queue.

The module maintains two message queues. The module can send one message from the high priority queue when it gets the token, even if the token hold time has expired. This kind of high priority refers to how the module processes the message in getting it out on the network and is different from a high priority frame control value.

If the message type is SRDL, the message is processed as a low priority message at the destination.

If the host sets bit 3, LAY2M_CTL_EVT_RX_CHG, the module enables generation of an event in the event queue on received data change.

If the host sets bit 4, the firmware will ignore the status of the message block. Its status will not be included in the global status for all configured message blocks.

If the host sets bit 5, LAY2M_CTL_EVT_CONFIRM, the module generates an event in the event queue whenever the message is confirmed by the receiver.

Note

Do not enable both receive data change and message confirmation events.

If the host sets bit 6, LAY2M_CTL_RX_BYTE_SWAP, the module swaps the bytes in received data. If the host needs to swap bytes in transmitted data, it must swap the data before writing to the module.

2.2.20.5.3 Destination Station

The host sets the destination station for the message by writing to the lay2mDstStn register in the message control block. The range for the lay2mDstStn register is 0 to 126.

When the destination station is set to 127 and the destination SAP is set to whatever SAP is configured to process multicast messages, the module sends multicast messages. The module sends a broadcast message when the host sets the destination station to 127 and the destination SAP to 63.

2.2.20.5.4 Destination SAP

The host writes the destination SAP (if required) to lay2mDstSap. Valid entries for the destination SAP are 0 to 63. The destination SAP is disabled when 255 is written to lay2mDstSap.

When sending a message with a destination SAP, the module automatically sets the high bit in the destination station. The host should not set the high bit when it sets the destination station number in lay2mDstStn.

2.2.20.5.5 Source SAP

The host may set the source SAP in the message by writing the source SAP number to lay2mSrcSap. Valid entries for the source SAP are 0 to 63. The source SAP is disabled when 255 (FFh) is written to lay2mSrcSap.

2.2.20.5.6 Message Type

The host sets the message type by writing to lay2mFrmCntrl. The message type can be one of the following commands:

Command	Value	Description
FC_SDAl	03h	Send Data with Ack Lo Pri (SDAL)
FC_SDAh	05h	Send Data with Ack Hi Pri (SDAH)
FC_SDNl	04h	Send Data No ack Lo Pri (SDNL)
FC_SDNh	06h	Send Data No ack Hi Pri (SDNH)
FC_SRDl	0Ch	Send and Request Data Lo Pri (SRDL)
FC_SRDh	0Dh	Send and Request Data Hi Pri (SRDH)
FC_SmTime1	00h	First SM time message
FC_SmTime2	80h	Second SM time message
FC_SmSDN	02h	Send Data No ack
FC_SmSRD	01h	Send and Request Data
FC_SmSRDSltDel	0Ah	SM Send and Request Data Slot Del
FC_SmSRDSltKeep	0Bh	SM Send and Request Data Slot Keep
FC_DdbSRD	07h	DDB Send and Request Data
FC_DiagSRD	08h	Diagnosis Send and Request Data
FC_ReqFDL	09h	Request FDL status
FC_ReqId	0Eh	Request ID
FC_ReqLSAPSts	0Fh	Request LSAP Status

2.2.20.5.7 Transmit Data

The host sets the maximum transmit data (request) length in bytes for the message by writing to the lay2mTxMaxLen register, and the current length in bytes by writing to lay2mTxLen. Although these lengths are usually equal, the value in lay2mTxLen can be less than the value in lay2mTxmaxLen. Valid lengths for lay2mTxMaxLen and lay2mTxLen are 0 to 244.

While online, the host may change the value in lay2mTxLen and vary the message length. However, for periodic messages, the host must stop and restart the message for the change to take effect.

When the module goes online, it assigns the location of the message's transmit data. The module writes the page number of the transmit data for this message in lay2mTxDataPage and the offset within that page in lay2mTxDataOfs. The offsets are always on 8-byte boundaries.

2.2.20.5.8 Receive Data

The host sets the maximum receive data length in bytes (from the response) by writing to the lay2mRxMaxLen register. Valid lengths for lay2mRxMaxLen are 0 to 244.

When the module goes online, it assigns the location of the message's received data. The module writes the page number of the received data in lay2mRxDataPage and the offset within that page in lay2mRxDataOfs. The offsets are always on 8-byte boundaries.

When the module receives response data for the message, it writes the actual received data length in lay2mRxLen.

2.2.20.5.9 Message Update Time

When the host sets LAY2M_CTL_PERIODIC bit in the lay2mCntCfg register and sets the message update time by writing to lay2mUpdTime, the module makes the message periodic.

The update time is the time the module waits after completing one message until it starts another one.

The update time is measured in units of 1 µs. The value ranges from 1 to 16380; therefore, the maximum update time is 16.38 seconds.

2.2.20.5.10 Message Error Retry Time

If the host sets the LAY2M_CTL_RETRY_PERIODIC bit in the lay2mCntCfg register and sets the retry time for periodic messages by writing to lay2mErrTime, the module retries failed messages.

The retry time is the time from when the error occurs until the message is retried and is measured in ms. The range is 1 to 16380, for a maximum retry time of 16.38 seconds. If the retry time is 0, the message is not retried.

If the host does not specify a retry time and there is an error with a periodic or cyclic message, the message stops until the host restarts it.

2.2.20.5.11 Initiating Messages

The host controls message transmission through the trigger queue, pfbTrigQueue[256], in the PROFI_USR structure. The host accesses the trigger queue via the head pointer, pfbTrgHead, in the PROFI_USR structure. To insert triggers, the host puts the appropriate trigger value in the queue at the head position, then increments the head pointer.

The module then writes to the queue tail pointer, pfbTrgTail. The host can use the tail pointer to determine if the queue is full. If the head pointer + 1 = the tail pointer, the queue is full.

To send messages, the module must be online and the stations must be active.

Access to the trigger queue is controlled by the state register, lay2mState, in the message control block. Possible values are:

Message	Value
LAY2M_STE_DISABLE	00h
LAY2M_STE_IN_CONFIGURE	01h
LAY2M_STE_ENABLE	02h
LAY2M_STE_ACTIVE	03h
LAY2M_STE_DONE	FFh

The LAY2M_STE_DISABLE value indicates that the block is not being used.

To indicate that the host is configuring the block, the module ignores any block in this state. The host sets lay2mState to LAY2M_STE_IN_CONFIGURE. In multitasking environments, the host can use this state to “hold” the message block.

The module sends the message when the host sets lay2mState to LAY2M_STE_ENABLE. The module sets lay2mState to LAY2M_STE_ACTIVE while processing the message, and then to LAY2M_STE_DONE, once the message is complete.

To start a message, the host should first check the state register and confirm that the state is LAY2M_STE_DONE or LAY2M_STE_DISABLE. The host should then set lay2mState to LAY2M_STE_IN_CONFIGURE. The module ignores this state but other tasks on the host will know that this block is busy. The host should then write any necessary parameters and values to the block.

If the message is to be periodic, the host should verify that the LAY2M_CTL_PERIODIC bit in the control and config register, lay2mCntCfg, is set, and that the message update time has been configured. The host should then set the state register, lay2mState, in the message control block to LAY2M_STE_ENABLE. Finally, the host should put the message in the trigger queue, by writing to the location in the queue indicated by the head pointer. The value the host writes is formed by OR’ing together TRG_LAY2M (1000h) and the message block number; for example, the upper byte is 10h and the lower byte is the message block number. The host should then increment the trigger queue head pointer.

2.2.20.5.12 Stopping Periodic Messages

When the host clears the LAY2M_CTL_PERIODIC bit in the message control block’s lay2mCntCfg register, the module stops the periodic message. When the message completes, the module does not resend it until the host puts it back in the trigger queue.

2.2.20.5.13 Sending Broadcast/Multicast Messages

Broadcast and multicast messages are sent to destination station 127, and broadcast messages are sent to destination SAP 63. The frame control must be SDNl or SDNh, as all broadcast/multicast messages are unconfirmed. To receive a broadcast or multicast message, the destination station must have strict station checking disabled. To receive a broadcast message, the station must have a SAP configured on SAP 63.

2.2.20.5.14 Monitoring Message Status

To indicate the message status, the module maintains several registers.

2.2.20.5.15 Status Register

To show whether the message block is operating correctly, the module sets bits in the status register, lay2mStatus. If the block is problem-free, the module sets bit 7, LAY2M_STS_OK.

2.2.20.5.16 Error Register

To indicate various errors operating errors, the module writes to the error register, lay2mError.

Error	Value
LAY2M_ERR_NOT_OK	01h
LAY2M_ERR_RX_LEN	02h
LAY2M_ERR_TX_LEN	03h

If there are multiple errors, this register contains the value for the last error that occurred.

2.2.20.5.17 Response Status

The response status, lay2mRspStatus, is returned by the LAN controller when it encounters an error. Mask the value with BFh and use the following table to determine the meaning. X indicates “do not care” values in the upper half of the byte.

Value	Error
00h	OK
01h	User error, SAP locked, destination did not take the message because the SAP was locked
02h	No resource for send data. SAP could not accept the message.
03h	No service available (SAP does not exist)
04h	Access point blocked
80h	Short character, problems with wiring, termination, etc.
9Fh	No access. Destination station was not present.
Afh	Double token detected, problems with wiring, termination, etc.
BFh	Response buffer too small
8Fh	Noise at SM command, problems with wiring, termination, etc.

2.2.20.5.18 Message Events

These events occur when the module is used to send FDL (Layer 2) messages.

Message	Value
EVT_LY2M_RX_DATA_CHG	50NNh
EVT_LY2M_CONFIRM	51NNh
EVT_LY2M_ERROR	52NNh
EVT_LY2M_BAD_MSG_NUM	53NNh

The lower byte in the event word contains the block number of the message that generated the event.

To notify the host that various events related to this message block have occurred, the module sets bits in the event register, lay2mEvent. To notify the host that this message has been updated, the module sets bit 0, LAY2M_EVT_INDICATION.

The module sets bit 1, LAY2M_EVT_RX_DATA_CHG, to notify the host that the received data for this message block has changed.

The host acknowledges these events by clearing the bits. As only one of these options can be enabled, no more than one will occur. If the host does not clear lay2mEvent, the module does not generate any further received data change or message confirmation events. Instead, it increments the overrun error counter.

2.2.21 LED Usage

2.2.21.1.1 Communication Status LED

The communication status (lower) LED shows the health of the network.

When the local master has the token, the communication status LED is green. If there is a network error, the module sets the communication status LED to red for a minimum of 1 second. If the local station is passive, the communication status LED is off.

2.2.21.1.2 System Status LED

The system status LED on the module bracket shows the current state of the various operations configured on the module. For more information on LEDs, refer to the Hardware Reference Guide.

The system status LED flashes the state of the DP master, DP slave, Layer 2 messages and Layer 2 SAPs sequentially. Only those operations configured on the module are shown. The host has the option of disabling the LED display for a particular operation, even if you are using that operation.

The system status LED flashes red if there is a problem with one of the configured PFBSCAN operations, and green if the operation is OK. For DP master, amber means that all slaves are OK but the scan is being performed in clear mode. For a DP slave, amber means that the slave is being scanned by a master in clear mode.

The module also uses the LEDs to signal internal errors. If an internal error occurs, the module flashes the system status LED once red, then flashes an 8-bit error code sequentially on the communication status LED, from low bit to high bit. Red indicates that the bit is zero; green that it is 1. Then the cycle repeats. Record the sequence before calling Technical Support.

2.2.22 Master Class 2

In a PROFIBUS-DP system, a DP Master Class 2 is a programmer or a management device. The following register allows the host to communicate with a Master Class 1 device, using the module as a Master Class 2 device. The mechanism for sending a message is the same as sending an FDL message: the host will add data to the appropriate registers and then trigger the send of the message. For more information, refer to Sections 2.2.20.3, The Trigger Queue, and 2.2.20.5, FDL (Layer 2) Messages.

2.2.22.1 Master Class2 Control and Configuration Register (mc2CntCfg)

The mc2CntCfg register contains bits that control Master Class 2 functions.

If bit 0 in mc2CntCfg is set, the Master Class 2 message that the module sends is Send Data with No Acknowledge. Otherwise, it is Send Data with Acknowledge.

If bit 2 in mc2CntCfg is set, the Master Class 2 message is sent with high priority. Otherwise, it is low priority.

If bit 5 in mc2CntCfg is set, the module generates an event, EVT_MC2_CONFIRM, when a Master Class 2 message is confirmed.

2.2.22.2 Master Class 2 State Register (mc2State)

The mc2State register shows the current state of Master Class 2 operations on the module.

State	Value	Description
MC2_STE_DISABLE	00h	Master Class 2 operation is disabled on the module
MC2_STE_ENABLE	02h	Master Class 2 operation is enabled on the module. This bit must be set before sending a Master Class 2 message.
MC2_STE_ACTIVE	03h	A Master Class 2 message is being sent
MC2_STE_DONE	FFh	A Master Class 2 message has been confirmed

2.2.22.3 Master Class 2 Status Register (mc2Status)

The mc2Status register shows the status of Master Class 2 operations on the module. If the Master Class 2 operations are OK, the module writes 80h to the mc2Status register. Otherwise, the module writes 0h to the mc2Status register.

2.2.22.4 Master Class 2 Error Register (mc2Error)

The mc2Error register shows the last error that occurred in Master Class 2 operations.

State	Value	Description
MC2_ERR_NOT_OK	01h	An error has been returned from the ASPC2
MC2_ERR_RX_LEN	02h	The RX length is greater than 244
MC2_ERR_TX_LEN	03h	The TX length is greater than 244
MC2_ERR_POLL_TOUT	04h	There has been a poll timeout

2.2.22.5 Master Class 2 Event Register

The following event can happen when using Master Class 2 functionality. The module sets bit 0, MC2_EVT_CONFIRM, to notify the host that the message has been confirmed.

2.2.22.6 Master Class 2 TX Length (mc2TxLen)

The mc2TxLen register is the length of data to transmit in the Master Class 2 message. If the value in the mc2TxLen register is greater than 244, the module reports the MC2_ERR_TX_LEN error in the mc2Error register.

2.2.22.7 Master Class 2 RX Length (mc2RxLen)

The mc2RxLen register is the length of data to receive from the Master Class 2 message. If the value in the mc2RxLen register is greater than 244, the module reports the MC2_ERR_RX_LEN error in the mc2Error register.

2.2.22.8 Master Class 2 Poll Time (mc2PollTime)

The mc2PollTime register contains the length of time to wait for the Master Class 2 message to be confirmed.

2.2.22.9 Master Class 2 Poll Limit (mc2PollLimit)

The module receives periodic responses from the ASPC2 ASIC with regards to the Master Class 2 message. If the module receives more responses than the mc2PollLimit with no confirmation, the module reports a MC2_ERR_POLL_TOUT error in the mc2Error register.

2.2.22.10 Master Class 2 Destination Station

The mc2DstStn register holds the number of the station that the message is destined for.

2.2.22.11 Master Class 2 TX Data

The mc2TxData[244] array holds the data that will be transmitted to the remote station. When the host triggers a Master Class 2 message to be sent, the module transmits the data in the mc2TxData buffer.

2.2.22.12 Master Class 2 RX Data

The mc2TxData[244] array holds the data that will be received from the remote station. When the host triggers a Master Class 2 message to be sent, the module receives the data in the mc2RxData buffer.

2.3 Host Interface Summary

This section summarizes the steps required by the host system to bring the module online as a DP master and a DP slave. For more detailed information, refer to Section 2.2, PROFI_USR Structure.

2.3.1 DP Master Summary

To configure the card as a DP master from the host:

- Set any required global options in the DP master global control register, pfbMasCntrlCfg
- Set minimum and maximum I/O scan time limits
- Create a master control block for each slave being scanned

To configure each slave, the host must set the following in the master control block for each slave:

- Station address
- Slave options
- Receive data length
- Receive data offset, if your application is assigning offsets
- Transmit data length
- Transmit data offset, if your application is assigning offsets
- Slave Ident number
- Slave watchdog factors
- Any additional slave parameter data (if required)
- Slave configuration check data

2.3.2 DP Slave Summary

Before the card can go online as a DP slave, the host has to set the following:

- Network parameters, if they have not already been set. These include the station number, baud rate, high station address and so on.
- Slave Ident, if it is different from the default
- Receive data length (default is 0)
- Transmit data length (default is 0)
- Any required slave options, in the slvCntCfg register

A

Warranty and Support

Appendix Contents:

- Warranty
- Technical Support

A.1 Warranty

For warranty information, refer to http://www.mysst.com/warranty.asp.

A.2 Technical Support

Please ensure that you have the following information readily available before calling for technical support:

- Card type and serial number
- Computer's make, model, CPU speed and hardware configuration (other cards installed)
- Operating system type and version
- Details of the problem you are experiencing: application module type and version, target Network, and circumstances that may have caused the problem

A.2.1 Getting Help

Technical support is available during regular business hours by telephone, fax or email from any Woodhead Software & Electronics office, or from http//www.woodhead.com. Documentation and software updates are also available on the Web site.

Note

If you are using the card with a third-party application, refer to the documentation for that package for information on configuring the software for the card.

North America

Canada:
Tel: +1-519-725-5136
Fax: +1-519-725-1515
Email: WoodheadSupportNA@molex.com

Europe

France:
Tel: +33 2 32 96 04 22
Fax: +33 2 32 96 04 21
Email: WoodheadIC.SupportEU@molex.com

Germany:
Tel: +49 7252 9496 555
Fax: +49 7252 9496 99
Email: WoodheadIC.SupportDE@molex.com

Italy:
Tel: +39 010 5954 052
Fax: +39 02 664 00334
Email: WoodheadIC.SupportIT@molex.com

Other countries:
Tel: +33 2 32 96 04 23
Fax: +33 2 32 96 04 21
Email: WoodheadIC.SupportEU@molex.com

Asia-Pacific

Japan:
Tel: +81 52 221 5950
Fax: +81 46 265 2429
Email: WoodheadIC.SupportAP@molex.com

Singapore:
Tel: +65 6268 6868
Fax: +65 6261 3588
Email: WoodheadIC.SupportAP@molex.com

China:
Tel: +86 21 5835 9885
Fax: +86 21 5835 9980
Email: WoodheadIC.SupportAP@molex.com

For the most current contact details, please visit http://www.woodhead.com.