5136-PFB-VME
Software Guide

Version 2.04

SST

50 Northland Road

Waterloo, Ontario, CANADA

N2V 1N3

Tel: (519) 725-5136   Fax: (519) 725-1515

www.sstech.on.ca

Publication Revision: 2.04

Publication Date: October 16, 2001

This document applies to the 5136-PFB-VMEInterface Card.

Contents

A

viii

1

Introduction

This chapter describes the following:

- the purpose of the manual
- the style conventions used in the manual
- an overview of the card
- reference material
- warranty and technical support information

1.1 Purpose

This document is a software user's guide for the SST 5136-PFB-VME interface card. This card makes it possible for an application running on a VMEbus host computer to communicate with ProfiBus networks using Profibus DP and FDL.

2

1.2 Conventions

1.2.1 Style

The following conventions are used throughout the manual:

- Listed items, where order is of no significance, are preceded by bullets.
- Listed items, to be performed in the order in which they appear, are preceded by a number.
- References to commands, or dialog boxes are *italicized.*
- User entry text is in `Courier 9 pt` font.
- Buttons that the user may press are in SMALL CAPS.

1.2.2 Special Notation

The following special notations are used throughout the manual:

Warning messages alert the reader to situations where personal injury may result. Warnings are accompanied by the symbol shown, and precede the topic to which they refer.

Caution messages alert the reader to situations where equipment damage may result. Cautions are accompanied by the symbol shown, and precede the topic to which they refer.

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.

3

1.3 Card Overview

The 5136-PFB-VME card can:

- act as a DP slave
- act as a DP master
- send and receive FDL (layer 2) messages

The card supports simultaneous operation in all these modes.

The card supports the standard ProfiBus baud rates of 9.6K, 19.2K, 93.75K, 187.5K, 500K, 750K, 1.5M, 3M, 6M and 12M baud.

The card has an onboard Intel i960 processor with 512 Kbytes of local RAM, that handles the communication protocol and data formatting into the shared RAM.

The 5136-PFB-VME consists of:

A VMEbus interface, with capabilities as outlined in the Specifications section of this manual.

Five registers in VME short address space for control of the card.

The card has an additional 256 Kbytes of RAM which is shared with the host as either a single block of 256 Kbytes (linear addressing mode) or as one of sixteen, 16Kbyte pages (paged mode). In paged mode, the host determines which page of this shared RAM is mapped into the host memory by writing to a register on the card. This block of memory contains all the tables and buffers that are used to pass information between the interface card and the application software running in the host computer. This approach ensures a fast and simple connection between the host computer and the card software.

In addition, the card has 512 Kbytes of sectored flash memory, for storage of programs and configuration data. ProfiBus configuration information may also be stored in flash.

4

1.4   Reference Documents

For information on ProfiBus, refer to one of the following:

- ProfiBus standard DIN 19 245 parts 1, 2 and 3. Part 1 describes the low level protocol and electrical characteristics, part 2 describes FMS, part 3 describes DP
- European standard EN 50170
- ET 200 Distributed I/O System Manual, 6ES5 998-3ES22

5

1.5   Technical Support

1.5.1   Before you call for help...

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

- Card type and serial number
- Computer make and model and hardware configuration (other cards installed)
- Operating system type and version
- Details of the problem; application module type and version, target network, circumstances that caused the problem

1.5.2   Getting Help

Technical support is available during regular business hours by telephone, fax or email from any SST office, or from the company Web site at www.sstech.on.ca.

Documentation and software updates are available on our Web site.

North America

Telephone: 519-725-5136, Fax: 519-725-1515

Email: techsupport@sstech.on.ca

Europe

Telephone: +49/(0)7252/9496-30, Fax: +49/(0)7252/9496-39

Email: sst@woodhead.de

Asia

Telephone: +81-4-5224-3560, Fax: +81-4-5224-3561

Email: techsupport@woodhead.co.jp

6

2

Quick Start

This chapter provides step by step instuctions on:

- bringing the card online as a DP slave
- bringing the card online as a DP master

7

2.1   Quick Start

The following sections describe briefly the steps required to bring the 5136-PFB-VME online as a DP master or a DP slave, using the sample programs provided with the card.

These steps assume that the card is being installed using the default port and memory addresses. They also assume familiarity with the SST ProfiBus Configuration Tool or Siemens COM PROFIBUS configuration software.

2.1.1   DP Master

> ⚠️ Prior to performing this procedure, refer to the *Hardware User's Guide* for more detailed information on installing the card.

Use the following steps to get the 5136-PFB-VME scanning I/O using ProfiBus DP:

1.   Install the card. (Refer to the *Hardware User's Guide0*.

     Check the switch setting on the card and install the card in the computer.

2.   Install the Software, choosing one of the three options described here. Refer to the *Hardware User's Guide,* for more detailed information.

     - Run the batch file *install.bat* on the distribution disk to install the DOS distribution software. This batch file install the software in the directory *sspfbdos*.

     - Run the *setup.exe* program from the appropriate Windows disk to install the SST ProfiBus Configuration Tool.

     - Run the batch file *comet\updcomet* to install the files for configuring the 5136-PFB-VME under COM PROFIBUS.

3.   Create an I/O configuration.

     Use the SST ProfiBus Configuration Tool or Siemens COM PROFIBUS to create a configuration for the I/O you are scanning.

     Refer to the *software's online help* for complete details on creating a configuration with one of these tools.

8

4. Download the ProfiBus software module. Refer to the *Hardware User's Guide.*

   Use the loader program *pfbinst* to download the ProfiBus module to the card. For example, if the card is installed at the default port and memory addresses, the command is:

   ```
   vmeinst vmeprofi
   ```

5. Configure the card as a DP Master

   Run the sample program *cometcnf* to configure the card as a DP master using the binary file you exported from the SST ProfiBus Configuration Tool or COM PROFIBUS. If the card is installed at port address 250 and the file is called *test.2bf*, the command is:

   ```
   cometcnf 250 test
   ```

6. Put the card online.

   Run the sample program *vmecmd* to put the card online. If the card is installed at port address 250, the command is:

   ```
   vmecmd 250 online
   ```

7. Scan I/O.

   Run the sample program *dpmon* to display and edit data. If the card is installed at port address 250, the command is:

   ```
   dpmon 250
   ```

   From the program menu:

   - Press *A* to view the active station list, then press *P* to see what passive stations (in this case, I/O slaves) are present on the network.

   - Press *M* to view configured slaves. Press *R* to put the card in run mode. Select a slave and press *Enter* to view and edit its I/O data.

   - Press *E* from the main menu to view diagnostic counters.

9

2.1.2 DP Slave

> ⚠️ Prior to performing this procedure, refer to the *Hardware User's Guide* for more detailed information on installing the card.

Use the following steps to bring a 5136-PFB-VME online as a DP slave.

1. Install the card. (Refer to the *Hardware user's Guide).*

   Check the switch settings on the card and install the card in the computer.

2. Install the software, choosing one of the three options described here.

   • Run the batch file *install.bat* on the distribution disk to install the distribution software. This batch file install the software in the directory ssvmedos.

   • Run the *setup.exe* program from the appropriate Windows disk to install the SST ProfiBus Configuration Tool.

   • Run the batch file comet\updcomet to install the files for configuring the 5136-PFB-VME under COM PROFIBUS.

3. Download the ProfiBus software module.

   Use the loader program *vmeinst* to download the ProfiBus module to the card. For example:

   ```
   vmeinst vmeprofi
   ```

4. Create a network configuration.

   Edit the sample file *netslv.ncf* file to set the network parameters. At minimum, you need to set:

   - baud rate
   - station address

5. Create a slave configuration.

   Edit the sample file *slave.scf* to set the slave parameters. At minimum, you need to set:

   - number of inputs (bytes)
   - number of outputs (bytes)

10

6. Configure the card's network parameters.

    Run the sample program *pbnetcnf* to configure the card's network parameters. If the card is installed at port address 250 and the file is called *netslave.ncf*, the command is:

    ```
    pbnetcnf 250 netslave
    ```

7. Configure the card as a slave

    Run the sample program *dpslvcnf* to configure the card as a DP slave. If the card is installed at port address 250 and the file is called *slave.scf*, the command is:

    ```
    dpslvcnf 250 slave
    ```

8. Put the card online

    Run the sample program *pbcmd* to put the card online. If the card is installed at port address 250, the command is:

    ```
    vmecmd 250 online
    ```

9. Access the data.

    Run the sample program *dpmon* to access data. If the card is installed at port address 250, the command is:

    ```
    dpmon 250
    ```

    From the program menu:

    - Press *S* to view the slave data.
    - Press *D* to view slave diagnostics
    - Press *E* to view diagnostic counters.

11

3

The VMEPROFI Module

This section describes the following:

- an overview of the module
- an overview of the software
- command, ID and status registers
- network parameters
- using the card as a DP master, a slave and for FDL
- diagnostic counters
- active station list
- putting the card online
- events and interrupts
- station IDs
- using the host watchdog
- VMEPROFI LED usage

13

3.1 VMEPROFI Module Overview

This section is a guide for writing applications to interface to the SST VMEPROFI module (*vmeprofi.ss1, vmeprofi.ssf*) using the 5136-PFB-VME card.

Using the VMEPROFI software module, the 5136-PFB-VME can:

- act as a DP master
- act as a DP slave
- send and receive FDL (layer 2) messages
- send and receive FMS messages

The module can be used for all these operations at the same time.

In addition, the module:

- 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
- supports all standard baud rates and can detect the network baud rate
- maintains an event queue where the host can be notified of various events occurring in the DP master, DP slave. These events can be disabled or enabled independently.

As a DP master, the card:

- can control up to 126 slaves
- supports up to 244 bytes of input data per slave, 16 Kbytes total input data for all slaves
- supports up to 244 bytes of output data per slave, 16 Kbytes total output data for all slaves
- can be configured using binary files exported from the SST ProfiBus Configuration Tool or Siemens COM PROFIBUS software
- can generate an event and an optional interrupt on:
- received data change from any slave
- update from any slave
- scan done
- transition to error
- transition to OK

14

As a DP slave, the card:

> • 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

The layer 2 (FDL) interface:

> • allows configuration of up to 64 SAPs
>
> • has an optional timeout watchdog on any SAP
>
> • supports cyclic, periodic and one-shot requests
>
> • 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

15

3.2 VMEPROFI Software Overview

3.2.1 Memory Overview

The host uses a memory mapped interface to the program running on the card. The host interface is defined in the structure PROFI_USR in the file *profictl.h*. The user interface for the VMEPROFI module is completely open. There is no need to link to any special libraries, making the card independent of the compiler and operating system being used.

In paged mode, the amount of data on a ProfiBus network is more than can be directly mapped into the memory of the host computer at one time, so the shared memory on the card is paged with only one 16 Kbybe page mapped into host memory at one time. The PROFI_USR structure is on the home page. Other pages of card memory contain the DP master input data, DP master output data, DP master control blocks, and so on. Your application swaps these pages into the host computer memory whenever it needs to access the data on those pages.

In linear mode, pages of memory occupy successive 16 Kbyte areas in standard memory. Whenever the card data is identified by page and offset, use the page to locate the appropriate 16 Kbyte area, and the offset to locate the data within the 16 Kbyte area.

The following table shows where the data is found in linear and paged modes.

Data	Linear Mode Offset	Paged Mod Offset
Home page	08000h	2
MasCntrlblocks	0C000h	3
FMS CRL	10000h	4
DP mas Tx data	14000h	5
DP mas Rx data	18000h	6
MPI blocks	1C000h	7

The following table briefly describes the layout of this structure. Refer to *profictl.h* for detailed information.

Offset	Description
0	Command register
1	Status register
2-7	ID registers
08h-0Dh	Event and interrupt control
0Eh-25h	ASPC2 control
26h-27h	Trigger queue control
28h-2Fh	DP master status and control
30h-32h	FMS status and control
40h-7Dh	Diagnostic counters
7Eh-FFh	Active station list
100h-103h	Master configuration parameters
	Watchdog parameters
10Ch	Global event register
110h-17Fh	ID response text
180h-3FFh	DP slave control, status and data
400h-5FFh	Event queue
600h-7FFh	Trigger queue
800h-8f9h	ID response fields
FE0h-3FFFh	Layer 2 control and data

17

3.2.2 Programming Notes

The interface between application and card module requires that the application software write into and read from specific memory locations. The sample programs supplied with the card show how to do this, using pointers and structures in the C programming language. The sample programs were developed using Borland C++ version 3.1. The sample programs are written under DOS for a paged mode ISA card.

The file *profictl.h* contains structure declarations and constant definitions. Include this file in any C program that uses the same structures, names, etc. as the sample programs. The file *profiswp.h* contains the same information but for environments where bytes are swapped.

SST recommends that you use defined constants and masks rather than hard-coded constants. Any changes in future versions of the card software will be easier to accommodate.

18

3.3   Command Register

The host uses the command register, vmeCommand, to issue commands to the VMEPROFI module.

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

The following table lists available commands. These commands are described in more detail in other sections of this manual, as appropriate. The card sets the command register to the following values to indicate the card state.

Command	Value	Description
CMD_OFF	E0h	Card is offline ready to take commands
CMD_ON	E1h	Card is online ready to take offline command
CMD_COM_CFG	E2h	On versions of the card with a serial port, this value indicates that the card is being configured from the serial port.
CMD_ERROR	EFh	Card is in error, status contains error code

The host can issue the following commands to the VMEPROFI module by writing to the command register.

Command	Value	Description
CMD_GO_ON	01h	Card should now go online
CMD_GO_OFF	02h	Card should now go offline, or abort config
CMD_REINIT	03h	Card should reinitialize memory and all parameters
CMD_CLR_CFG_BUF	04h	Card should clear vmeBinCfgPage
CMD_CHK_NET_CFG	05h	Card should check network parameters, and assign defaults
CMD_CPY_MAS_CFG	06h	Card should copy a page of cfg from vmeBinCfgPage

19

Command	Value	Description
CMD_AUTO_BAUD_DET	07h	Card should do automatic baud detect
CMD_MAS_ASIGN_ADDR	08h	Assign and fill in data addr (page/offset) for DP master
CMD_CFG_2BF_SHRAM	10h	Configure DP master from ET200 binary previously copied
CMD_CFG_FROM_FLASH	11h	Configure card from configuration in flash
CMD_FLASH_GO_ON	14h	Configure card from flash, then go online
CMD_PGM_TO_FLASH	21h	Burn card configuration into flash

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

If there is an error in executing a command, the host should check the status register (refer to section 3.4, *Status Register*) to determine what the error was. If the error is a non-fatal error, the host should clear the error by issuing the following command.

Command	Value	Description
CMD_ERR_ACK	FFh	Acknowledge offline error (online errors cannot be acked)

This returns the card to the startup state, just after you ran the loader. You must then reconfigure parameters.

If the card returns a fatal error, you must reload the card.

The sample program *vmecmd.c* shows how to use the command register.

3.4   Status Register

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

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

If the card 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 card sets the status register to STS_BAD_CMD (01h).

Refer to section 3.3, *Command Register*, for information on what to do when there is an error.

Network Parameter Errors

The following status errors may occur when the host sets the network parameters. See section 3.6, *Network Parameters*, for allowed values for the network parameters.

Command	Value
STS_BAD_BAUD	02h
STS_BAD_STN_ADR	03h
STS_BAD_HI_STN_ADR	04h
STS_BAD_TOK_ROT	05h
STS_BAD_SLOT_TME	06h
STS_BAD_IDLE_1	07h
STS_BAD_IDLE_2	08h
STS_BAD_RDY_TME	09h
STS_BAD_QUI_TME	0Ah
STS_BAD_GAP_UPD	0Bh
STS_BAD_TOK_RETRY	0Ch
STS_BAD_MSG_RETRY	0Dh
STS_BAD_TOK_ERR_LIM	0Eh
STS_BAD_RSP_ERR_LIM	0Fh
STS_BAUD_DET_ERROR	10h

21

COM PROFIBUS Configuration Errors

The following errors occur when the host is configuring the card as a DP master using a binary file generated by the Siemens COM PROFIBUS software. These errors also occur when using the SST Config tool. See section 3.7, *Using the Card as a DP Master*.

Error Message	Value
STS_CFG_BAD_CHK_PATTERN	20h
STS_CFG_BIN_TOO_SHORT	21h
STS_CFG_BIN_TOO_LONG	22h
STS_CFG_BAD_CHKSUM	23h
STS_CFG_INVALID_CPU_HDR	24h
STS_CFG_INVALID_SLV_REC_TYP	25h
STS_CFG_RX_OVERFLOW	26h
STS_CFG_TX_OVERFLOW	27h

The following errors occur when you are using the Siemens COM PROFIBUS software or the SST Config tool to configure the card as a DP master and you use the slave designation fields to set card options. If there are errors in what you enter, you get the following status errors. Refer to section 3.7, *Using the Card as a DP Master*, for more information.

Error Message	Value	Description
STS_CFG_DESIG_NAME_TOO_LONG	28h	Nm= parameter, name too long (12 chars max)
STS_CFG_DESIG_BAD_ARG	29h	Unrecognized argument (Nm=, Tx=, Rx=, Ch)
STS_CFG_DESIG_INV_RX_OFS	2Ah	Rx= parameter, invalid offset (0000-3ff8)
STS_CFG_DESIG_INV_TX_OFS	2Bh	Tx= parameter, invalid offset (0000-3ff8)
STS_CFG_DESIG_OFS_NOT_SPEC	2Ch	Rx,Tx Ofs has been spec'd for one, but not all slaves

Error Message	Value	Description
STS_CFG_RX_OVERLAP	2Dh	Rx data for one block overlaps another
STS_CFG_TX_OVERLAP	2Eh	Tx data for one block overlaps another
STS_CFG_INV_LEN	2Fh	Invalid parameter or check data length
STS_CFG_MAS_EXT_ALLOC_ERR	35h	Out of master extension memory
STS_CFG_ADDR_OUT_OF_RANGE	36h	Rx or Tx offset was out of range
STS_CFG_COPY_TABLE_OVERUN	37h	

Flash Programming Errors

The following errors may occur when the host issues a command to program flash memory.

Error Message	Value	Description
STS_CFG_NO_CONFIG	30h	No configuration present to program into flash
STS_FLASH_BAD_ID	31h	Internal flash error
STS_FLASH_ERASE_ERR	32h	Internal flash error
STS_FLASH_PROG_ERR	33h	Internal flash error
STS_FLASH_VRFY_ERR	34h	Internal flash error

If any of the internal flash errors occur, make sure the flash write enable jumper (JP6) is installed. Refer to the *Hardware User's Guide* for more information on switches and jumpers.

Fatal Errors

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

If the status register contains STS_CFG_INTERNAL_ERROR (80h) there is an internal error on the card. Record the contents of the errInternal and errArg registers.

If the status register contains STS_OUT_OF_APBS (81h), the card 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. If you get this error, you must reduce the number of application blocks you are using.

If the status register contains STS_HOST_WD_BITE (82h), the host watchdog has timed out. Refer to section 3.16, *Using the Host Watchdog*.

If the status register contains STS_HEAP_ALLOC_FAIL (83h), this indicates that the card has run out of local RAM. This error should never occur. If the status register contains STS_SH_HEAP_ALLOC_FAIL (84h), this indicates that the card has run out of shared memory. Reduce the number of message blocks, or SAPs.

If the status register contains `STS_NET_ERROR` (90h), there has been a network error and the `OPTION_STAY_OFF_ERR` bit is set in the pfbOptions register. The card is offline. Refer to section 3.6.1, *Basic Parameters*.

25

3.5 ID Registers

The host uses the ID registers to identify the card, the VMEPROFI module, and the module revision.

3.5.1 Card ID

Applications can use the card ID, pfbCardId, to verify that the card is present. It contains a value of AAD0h or 43728 decimal.

3.5.2 Module ID

Applications can use the DP module ID, vmeModId, to verify that the correct module is loaded on the card. For the VMEPROFI module, this register contains BB01h or 47873 decimal.

3.5.3 Module Version

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

3.6   Network Parameters

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

You must set the network parameters before you put the card online. To change the network parameters, you must take the card offline.

The following sections describe the various network parameters. See also section 3.6.4, *Setting the Network Parameters*.

If you configure the card as a DP master using a binary file exported from the SST ProfiBus Configuration Tool or Siemens COM PROFIBUS software, the network parameters are included in the configuration file and you do not need to set them. You can still use the card registers to see the values set by the SST ProfiBus Configuration Tool or COM PROFIBUS.

3.6.1   Basic Parameters

The basic network parameters include:

- the local station address
- the network baud rate
- whether the station is active or passive
- the high station address for the network
- some network options

The station address is required; the other quantities have default values and may not need to be explicitly set if the default matches your application.

Station Address

The host uses the pfbStnAddr register to set the station address of the card on the ProfiBus network. The value can range from 0 to 126. The default station address is FFh, which is an invalid station address. Set the station address before putting the card online.

Note

Do not use station address 126 if you are using DP on the network. This is the default address for slaves whose station address must be set while the slave is online.

27

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 card sees an active station with a higher address and the `OPTION_STAY_OFF_ERR` bit in the pfbOptions register is set, the card goes offline with a status of `STS_NET_ERROR`. If this bit is not set, the card increments the errHsa error counter and goes back online. The default high station address is 126.

The high station address affects how much time is spent soliciting for 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 3.6.1, *Basic 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 can have station addresses higher that the high station address.

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 on the network by writing to the vmeActive register. Set this register to 1 for an active station or 0 for a passive station. The default is for the station to be passive (vmeActive=0).

A station must be an active station to act as a DP master or to initiate FDL or FMS messages.

28

Baud Rate

The host uses the vmeBaud 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
93.75 K	BAUD_93K75	2h
187.5K	BAUD_187K5	3h
500K	BAUD_500K	4h
750K	BAUD_750K	5h
1.5M	BAUD_1M5	6h
3M	BAUD_3M	7h
6M	BAUD_6M	8h
12M	BAUD_12M	9h

Note

The default baud rate is 9600 baud. The baud rate determines the default values for most of the network bus parameters.

The card can automatically detect the network baud rate. To detect the baud rate, issue the CMD_AUTO_BAUD_DET command. The card then listens to the network (but does not go online) at each possible baud rate for a period determined by the baud rate and shown in the following table. In the worst case, this can take up to 6 seconds. If the card detects 10 good messages in a row, it decides it has found the baud rate and enters the corresponding value in vmeBaud and sets vmeCommand back to CMD_OFF. If the card fails to detect the network baud rate, it returns a status of STS_BAUD_DET_ERROR (10h) in the vmeStatus register and sets the baud rate to the default.

29

Baud rate	Time, μs
9600	3000
19200	1500
93.75 K	513
187.5K	257
500K	86
750K	65
1.5M	33
3M	17
6M	9
12M	5

Network Options

The host sets several options related to how the card operates on the network by setting bits in the vmeOptions register.

The host uses bit 0, `OPTION_REPEATER`, to tell the card whether there are any repeaters on the network. Set this bit to 1 if there is at least one repeater on the network. Set this bit to 0 if there are no repeaters on the network. The card checks this bit when it assigns the default bus parameters (except vmeTokRotTime). The default is 0 (no repeaters).

The host uses bit 1, `OPTION_FMS`, to tell the card whether there are any FMS devices on the network. Set this bit to 1 if there are any FMS devices on the network. Set it to 0 if the network consists of only DP devices. The card checks this bit when it assigns the default bus parameters (except vmeTokRotTime). The default is 0 (DP only)

The host uses bit 2, `OPTION_STAY_OFF_ERR`, to tell the card what to do when the token error limit, vmeTokErrLimit**,** (see section 3.6.3, *Error Handling Parameters*) or the message error limit, vmeRespErrLimit**,** is exceeded within 256 token cycles. If this bit is 0 and either of these error conditions occurs, the card increments the corresponding error counter and goes offline, then goes back online immediately. If the bit is 1, the card goes offline with a fatal error and you must reload the card before putting it back online. The default is 0 and the card goes back online.

30

3.6.2 Bus Parameters

> **Note**
>
> Quantities of time are referred to as Tbits (bit times), and are the inverse of the baud rate. For example, if the baud rate is 1.5 Mbaud, 1 Tbit would equal 0.667 µs.

The bus parameters relate to quantities for example, times between messages. The card 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 vmeOptions register. You usually do not change parameters from their default values.

Target Token Rotation Time

The target token rotation time, vmeTokRotTime, is the target maximum token rotation time for the network, in Tbits. The allowed range is 256 to 16,777,215. 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. Set it to the sum of the target token rotation time required for each of the masters. If you are have problems configuring, especially at lower baud rates, increase this time.

When using the SST ProfiBus Configuration Tool or COM PROFIBUS and configuring multiple masters in the same project, the configuration tool adds the target token rotation times for the masters.

Slot Time

The slot time or frame timeout, vmeSlotTime, is how long the card waits for a reply to a message, in Tbits. The allowed values are 37 to 16,383. If the card does not receive the reply within this time, the card retries the message up to the maximum number of times specified in vmeMsgRetryLimit. If the card has not successfully sent a message to the destination previously, it does not retry the message.

The slot time is also the time the card waits for a reply when it polls for new nodes coming on the network.

31

Idle Times

Idle time 1, vmeIdleTime1, is the time, in Tbits, that the card waits after it receives a reply, an acknowledge or a token message before sending another message. The allowed range is 35 to 1023.

Idle time 2, vmeIdleTime2, is the time, in Tbits, that the card waits after sending an SDN (send data with no acknowledge) message before it sends another message. The allowed range is 35 to 1023.

Ready Time

The ready time, vmeReadyTime, is the time in Tbits that the card, after sending a command, waits before sending an ACK or response and is also the time the card waits after receiving a command, before it sends a reply. The allowed range is 11 to 1023.

Gap Update Factor

The gap update factor, vmeGapUpdFact, is the number of token rotations between solicits for a new node. The process of soliciting for 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 allowed range is 0 to 255. The default is 128.

Note

The gap update factor sets the number of token rotations between solicits; the actual time depends on the baud rate. At lower baud rates, reduce the gap update factor so that nodes can come online quickly. At higher baud rates, use a higher gap update factor to reduce soliciting.

Quiet Time for Modulator

The modulator quiet time (Tqui), vmeQuiTime, is the time, in Tbits, that the card waits after it turns on its transmitter before it begins to send data and also the time the card keeps its transmitter on after it has finished transmitting a message. The allowed range is 0 to 127.

32

Network Parameter Defaults

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

No repeater, no FMS					
Baud rate	**Slot time**	**Idle time 1**	**Idle time 2**	**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
750K	300	37	140	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

No repeater, FMS					
Baud rate	**Slot time**	**Idle time 1**	**Idle time 2**	**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
750K	3000	251	990	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

33

Repeater, no FMS

Baud rate	Slot time	Idle time 1	Idle time 2	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
750K	300	37	140	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

Repeater, FMS

Baud rate	Slot time	Idle time 1	Idle time 2	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
750K	3000	251	990	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

34

3.6.3 Error Handling Parameters

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

Token Retry Limit

The token retry limit, vmeTokRetryLimit, is the number of times the card retries passing the token before deciding the station is not there. If the card decides the station is not there, it takes the station out of the active station list and passes the token to the next station in the active station list. Allowed values are 0 to 15. The default is 4.

Message Retry Limit

The message retry limit, vmeMsgRetryLimit, is the maximum number of times the card retries a message when the slot time expires. For example, if it is 4, the card tries the message a total of 5 times. If the card 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 in detail on the card function being performed. For example, if the message is an I/O update with the card as a DP master, the slave would fail and would have to be reinitialized. Allowed values are 0 to 15. The default is 4.

If the card has not successfully sent a message to a station previously, it will not retry messages.

Token Error Limit

The token error limit, vmeTokErrLimit, is the maximum number of errors in 256 token cycles. Allowed values are 0 to 255 The default is 255.

If the OPTION_STAY_OFF_ERR bit in vmeOptions is 1, the card goes offline with a fatal error and you must reload the card before putting it back online.

If this bit is 0, the card increments the corresponding error counter and goes offline, then goes back online immediately.

Response Error Limit

The response error limit, vmeRespErrLimit, is the maximum number of message failures (e.g., retry limit is exceeded) in 16 successive messages. Allowed values are 1 to 15 The default is 15.

35

If the `OPTION_STAY_OFF_ERR` bit in vmeOptions is 1, the card goes offline with a fatal error and you must reload the card before putting it back online.

If the `OPTION_STAY_OFF_ERR` bit is 0, the card increments the corresponding error counter and goes offline, then goes back online immediately.

3.6.4 Setting the Network Parameters

1. Make sure the card is offline.

2. Write appropriate values for the various network parameters as described in section 3.6.2, *Bus Parameters*. In most cases, do not change the default values assigned by the card.

3. Issue the command `CMD_CHK_NET_CFG` or `CMD_GO_ONLINE` and wait up to 7 seconds for the command to execute.

4. Monitor the command register for either `CMD_OFF`, which indicates that the card has returned to the offline state, or `CMD_ERROR` which indicates that the card encountered an error in processing the parameters.

    If the command register contains CMD_ERROR, check the status register for an error status that indicates what the problem was. Refer to section 3.4, *Status Register*, for information on possible status values. Any other value in the command register is invalid.

The card resets the network parameters when it goes offline or when a module is loaded. That is how the card knows which values have changed and how to assign defaults. When told to set the network parameters, it assumes that any value that is different from the invalid value it assigned was set by the host application. If any parameter still has the initial value assigned by the card, the card sets the value of that parameter to the default for the baud rate.

Therefore, if the network parameters are configured for one baud rate but the card is not put online, then the baud rate is changed, reload the card to reset the network parameters to the startup invalid values. Similarly, if you take the card offline, reconfigure the network parameters before you put the card back online.

36

3.6.5 Network Parameters in Flash Memory

The host can write the current network configuration data to flash memory
and read the network configuration from flash. Refer to section 3.9, *Network,
DP Master and DP Slave Data in Flash*, for information on how to do this.

37

3.7 Using the Card as a DP Master

The 5136-PFB-VME can communicate as a DP master to control up to 126 DP slaves. Each slave can have up to 244 bytes of input data and 244 bytes of output data. The supported total for all slaves is 16 Kbytes of input data and 16 Kbytes of output data.

The host application can configure the slaves using a configuration file exported from the SST ProfiBus Configuration Tool or the Siemens COM PROFIBUS software or it can configure the slaves itself.

We recommend that you configure the card using the SST ProfiBus Configuration Tool or COM PROFIBUS. Otherwise, you need detailed knowledge of all the slaves you are scanning, of the parameters that apply to the entire scan, and of parameters that apply to the network itself.

One page of shared memory on the card reserved for the master control blocks. A second page is reserved for input data from the slaves and a third page is reserved for output data to the slaves. The card writes the page numbers for these pages in the PROFI_USR structure. This structure also contains global control and status registers for the master.

For each configured slave, there must be a master control block for that slave on the card. This structure is defined as MAS_CNTRL in *profictl.h*. If you configure using a binary file exported from the SST ProfiBus Configuration Tool or COM PROFIBUS, the card software parses the binary file and creates these master control blocks. If you configure the slaves from the host, your application must create the master control blocks.

The card scans the slaves in master control block number order, not in station number order. The master control blocks must be consecutive.

Before putting the card online as a DP master, configure the network parameters. If configuring the card as a DP master using a binary configuration file exported from the SST ProfiBus Configuration Tool or COM PROFIBUS software, the network parameters are part of the configuration file and get set automatically. Refer to section 3.6, *Network Parameters*, for information on what these parameters are and how to set them.

The card can generate events in the event queue on various conditions occurring in the operation of the DP master. It can also optionally generate an interrupt on these conditions.

The card maintains several diagnostic counters that relate to operation as a DP master. Refer to section 3.11.2, *Master Block Statistics*, for a description of these counters.

3.7.1 DP Master Scanning Modes

When scanning as a DP master, the 5136-PFB-VME has two basic modes of operation:

- fully asynchronous mode (default)
- synchronous mode.

The SST ProfiBus master products have very consistent and repeatable DP scan times when operated in asynchronous mode.

> **Caution**
>
> Using the synchronous scanning mode may be detrimental to the deterministic nature of DP scan cycles.

In synchronous mode, the host application is in control of the mechanism that starts DP scan cycles. Therefore, the DP scan cycle is only as repeatable as the host application and/or the host operating system. It is recommended that this feature only be used on real-time operating system platforms.

DP Master Synchronous Mode

This basic operation of this mode has three parts:

1. Enable the synchronous operation mode and decide how `END_OF _SCAN` notification is done. For example, an `END_OF_SCAN` event can be detected by polling the vmeMasGlbEvt register for the `VME_MAS_SCAN_DONE` bit and/or by enabling interrupt operation.

2. Create a procedure to look for `END_OF_SCAN` notification, to read the DP input image memory (RxData) and to update the DP output image memory (TxData). The host application must clear the last `VME_MAS_SCAN_DONE` event or no further DP scan done events are processed.

3. Start a new DP scan.

Enabling Synchronous Mode

The host application must enable the synchronous mode by setting the option `VME_MAS_CTRL_SYNC_SCAN` in the pfbMasCntrlCfg register of the `PROFI_USR` structure (refer to *profictl.h* for more information). As with all configurable options, this must be done prior to issuing `CMD_GO_ON` command.

This is a bit set option and should be done as a read-modify-write operation. In this way, other options contained in the vmeMasCntrlCfg register are not disturbed.

End of DP Scan Notification

To determine when the VME card has finished a DP scan, the `VME_MAS_CTRL_EVT_SCAN_DONE` option must be set in the vmeMasCntrlCfg 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 vmeIntEna register.

The vmeMasGlbEvt register contains the value `VME_MAS_SCAN_DONE` after a DP scan. To recognize any further `VME_MAS_SCAN_DONE` events and/or receive any further interrupts from this event, the `VME_MAS_SCAN_DONE` bit must be cleared by the application. The error counter errEventOverrun increments for every DP scan completed with the `VME_MAS_SCAN_DONE` bit still set.

Your application is now free to read the DP slave input data and update the DP slave output data tables.

Restarting a DP Scan Cycle

The 5136-PFB-VME card waits at the end of each DP scan cycle for the host application to restart a new scan. Restart the DP scan by setting the I960_INT bit in the BCR (base port address).

Scan Determinism

The scan cycle is completely under the control of the host application. Therefore, the repeatability of the scan cycle time is completely dependent on the host application. For example, running your host application under an operating system which does not have real-time response, such as Microsoft Win 95/98/NT, can result in DP scan times that fluctuate as much as 200 per cent.

40

Processing VME_MAS_SCAN_DONE Events

Processing of these events should be done using hardware interrupts. Most real-time operating systems provide a mechanism by which a task is tied to a hardware event. In this way, DP scan time determinism is addressed and the host application will be able to process the input and output data in the most responsive way.

DP Slave Watchdog Timeout Values

Most DP Master configuration utilities provide a means of adjusting the response monitoring or slave watchdog timeout values. This DP master configured parameter determines how long it takes for the slave devices to fault if no master data exchange occurs. Since the host application's performance affects how often the 5136-PFB-VME executes a data exchange cycle with the assigned slaves (DP scan), it may be necessary to adjust the slave watchdog timeout value.

Most configuration utilities set this timeout to 4 to 6 times the estimated response time. Therefore, the default value should not be a problem. Be aware, however, that anomalous DP slave faults may result from the host application's performance.

3.7.2 DP Master Data Coherency Mode

This section explains the software method which provides data coherency across an entire DP slave device. It is implemented by the firmware modules for the 5136-PFB-VME interface card.

> **Note**
>
> The minimum firmware revision that provides this function is v1.22. The card's shared RAM interface specifications and structure are documented in the *profictl.h* header file included with your DOS installation disk.

Enabling Coherency

The interface card's home page contains the main card control structure `PROFI_USR`. The register vmeMasCoherFlags in this structure contains all the control and configuration flags for the data coherency mode of operation.

Specifically, setting the `VME_MAS_COHER_ENA` bit in this register, **prior to putting the card online**, puts the card in data coherency mode.

41

Dealing with DP Input Data

Normally, the `PROFI_USR` register vmeMasRxPage contains a page reference to the DP input data image and this number never changes subsequent to putting the card online. In data coherency mode, this number is set by the card's embedded processor at the end of each scan and is one of two possible values. This is done to ensure 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 `VME_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 `VME_MAS_RXPG_LOCK` bit must be cleared by the host application or the host never gets access to new input data from the network.

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 contained in the vmeMasTxPage until the host processor instructs the card that output data was updated.

Therefore, the host should write all of the DP output data to be updated to the page contained in the vmeMasTxPage register. The embedded processor sends this data on the next bus scan as soon as the `VME_MAS_TXPG_UPDATE` flag is set in the vmeMasCoherFlags register. When the new output data is processed and the vmeMasTxPage is changed to the alternative page number, the `VME_MAS_TXPG_UPDATE` bit is cleared **by the card's embedded processor**. The host should never clear the `VME_MAS_TXPG_UPDATE` flag, nor should the host write data to the DP output data page when this bit is set.

Example Code

The VME Windows 95/98/NT installation (*vmewin32.zip*) contains example applications which use data coherency mode or the default mode of operation. Specifically, the DP MONITOR application (*dpmon.c*) has code which checks for the `VME_MAS_COHER_ENA` flag and implements the Rx lock and Tx update flag control.

Algorithm

This algorithm provides a software solution which prevents the 5136-PFB-VME's embedded processor from accessing (reading or writing) the same memory locations the host processor is currently addressing. This problem only occurs for data ranges larger than 16 bits. The card hardware guarantees exclusive access to data sizes that are 16 bits or less.
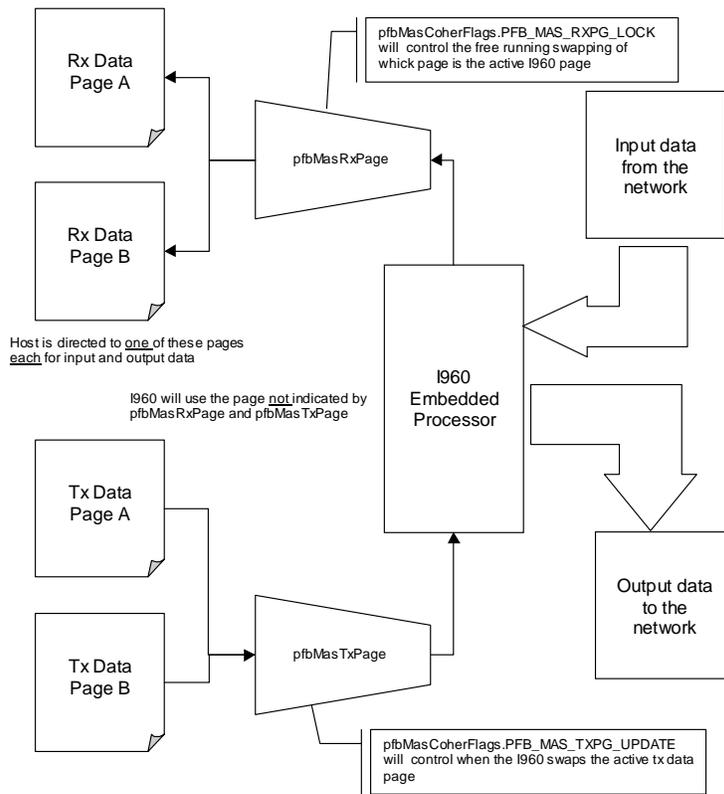
Figure 2 - Figure shows graphical representation of algorithm.

43

3.7.3 DP Master Page Registers

There are three registers in the `PROFI_USR` structure that tell the host where the master control blocks, DP master input data and DP master output data are located.

> • The vmeMasCntrlPage 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 vmeMasRxPage register contains the page number of the memory page that contains master received data (from slaves).
>
> • The vmeMasTxPage register contains the page number of the memory page that contains master transmit data (to slaves).

Read the values from these page registers and write them directly to the Memory Page register (MPR) to set the current page when you need to access the data on that page.

3.7.4 Configuring the DP Master

Use either the SST ProfiBus Configuration Tool or Siemens COM PROFIBUS software to create a binary file to configure a master system on the 5136-PFB-VME. There are several advantages to using either tool to configure the I/O:

> • The software assigns and calculates all the details for each slave. For example, the configuration check data, extended user parameter data, the watchdog factors, etc.
>
> • The software calculates the details for the entire collection of slaves, for example, the target token rotation time
>
> • The configuration software assigns appropriate values for all the network parameters and includes them in the binary file

44

Configuring the DP Master using the SST ProfiBus Configuration Tool

The SST ProfiBus Configuration Tool consists of a main view (the network view) and two dockable frames: the *ProfiBus Devices* frame and the *Online Browse* frame. Both of these are floating frames; drag and drop them to move them anywhere on the desktop.

To configure the 5136-PFB-VME card as a DP master:

1. Choose *Start/Programs/5136-PFB-32/Configuration Tools/SST ProfiBus Configuration* to run the SST ProfiBus Configuration Tool.

2. Choose *File/New* to create a new configuration.

3. If the *ProfiBus Devices* frame is closed, choose *View/Library* to open it. This frame appears on the upper left side of the window by default.

4. If the *Online Browse* frame is closed, choose *View/Online* to open it. This frame appears at the bottom of the window by default.

5. Find and select your SST ProfiBus master device in the *ProfiBus Devices* frame. To add this master device to the DP Network:

   Drag and drop it into the network view.

   or

   Click on the *Add to Network* button on the main toolbar.

   A configuration dialog box appears for the selected device.

6. Specify the master station number in the *General* tab of the dialog box.

7. Click *OK*.

   The master device is added to the network view.

   You can edit a device's properties at any time by right clicking on the device and choosing *Properties* or by selecting the device from the network list and clicking on the *Properties* button on the main toolbar.

8. Find and select your slave device in the *ProfiBus Devices* frame. To add this slave device to the DP Network:

    Drag and drop it into the network view under the master device.

    or

    Click on the *Add to Network* button on the main toolbar.

    If your slave device is not listed, click on the *Add Device* button on the *ProfiBus Devices* frame to add the GSD file for your device. The *Add PROFIBUS Devices* dialog box opens. Find the GSD file and click *Open*. You can now add the device to the network view as described above.

    A configuration dialog box appears for the selected slave.

9. Set the station address of the slave in the *General* tab.

10. If your device is a modular device, insert the module(s) your slave device uses. Select the *Modules* tab of the dialog box. If your slave is not a modular device (compact device), the module(s) are already configured.

11. Repeats steps 8-11 for each slave device on your network.

12. To set the parameters for your network, right click on the highest level of the network list (PROFIBUS_DP by default) and choose *Properties*.

    The *Network* dialog box opens.

13. Set the baud rate for your network.

14. Chose *File/Save As...* to save your configuration.

    The *Save As* dialog box opens.

15. Specify a file name for your configuration and click *Save*.

16. Choose the *Edit/Load Configuration* command to load the configuration to the card or choose *Edit/Export Binary...* to export your configuration to a binary (*.bss*) file.

17. Choose the *Edit/Online* command to put the card online.

You can now use the DP Monitor to view your network, look at your slaves and make sure the network is operating correctly.

Configuring the DP Master using COM PROFIBUS

Use Siemens COM PROFIBUS software to create a binary file to configure a master system on the 5136-PFB-VME. There are several advantages of using COM PROFIBUS to configure the I/O:

> • COM PROFIBUS calculates all the details for each slave. For example, the card calculates the configuration check data and the watchdog factors.
>
> • COM PROFIBUS calculates the details for the entire collection of slaves. For example, the card calculates the target token rotation time.
>
> • COM PROFIBUS software assigns appropriate values for all the network parameters and includes them in the binary file.

Use the following steps to create a binary file to configure the 5136-PFB-VME as a DP master. Refer to the documentation for the COM PROFIBUS software for details on how to carry out each step.

1. Create a new file *(File/New)*.

2. Select the master station number. This will be the station number of the scanner on the ProfiBus network.

3. For the master station type select *5136-PFB-VME* Master.

4. Select *Configure/bus parameters...* Set the baud rate for the network. If there is a repeater on the bus, check the *Repeater on bus* checkbox. If there are FMS devices on the network, change the Bus profile to *DP/FMS*. If the network has only ProfiBus DP devices, leave the bus profile as *PROFIBUS-DP*. COM PROFIBUS takes care of assigning appropriate default values for all the network parameters. Click *OK*.

5. Select *Configure/slave parameters...* to create and configure each slave:

    - select the slave station number

    - select the slave device family, such as ET 200, Simatic, etc.

    - select the slave station type

    - select the slave module type

6. Click *Configure...* to assign the number of inputs, number of outputs, data types, etc.

47

7. Edit the slave designation field (refer to3.8, *Using the Card as a DP Slave*) to set options for the slave

8. When you have configured all the slaves, save the configuration file (*File/Save)*.

9. Export the configuration to a binary file (*File/ Export/Binary File)*. This generates a *.2bf* file.

Using the Slave Designation Fields

Use the slave designation field for each slave to enter text that the card uses to set some features and options specific to the 5136-PFB-VME as a DP master. The text must begin with a backslash (\) character.

\nm=(up to 12 characters for a station name)

\rx=receive data offset (hex), if you are setting them manually

\tx=transmit data offset (hex), if you are setting them manually

\Ch to enable receive data change events

\RxSwp to swap bytes of received data

If assigning offsets, assign both transmit and receive data offsets. If assigning offsets for one slave, assign offsets for all slaves. These offsets must be in the range 0 to 3FF8h and must be on 8-byte boundaries. The values must be hexadecimal.

Example

\nm=Winder_IO rx=1000 tx=1000 ch

Networks with Multiple Masters

On networks with multiple masters, extend the target token rotation time for each one.

When configuring with the SST ProfiBus Configuration Tool or COM PROFIBUS and the masters are on the same network, include both masters in the same SST ProfiBus Configuration Tool or COM PROFIBUS file. Select the master system to export and the configuration tool takes care of increasing the target token rotation times, and watchdog times.

Make sure the correct master system is highlighted when exporting.

Configuring the DP Master with the Binary File

The card must be offline before it can be configured. Check the command register for the `CMD_OFF` state.

The card uses vmeBinCfgPage in the `PFB_PROFI` structure to tell the host on which page to write the configuration file. 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 vmeBinCfgOfs and the block length to vmeBinCfgLen. 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 card to process the block and set the command register to return to `CMD_OFF` to indicate that it has finished processing.

When the host has written the entire file to the card and the card has processed them all, the host issues the `CMD_CFG_2BF_SHRAM` command and waits up to 2 seconds for the command to complete. The card 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. Acknowledge the error. If the card successfully processes the file, it sets the command register to `CMD_OFF` to indicate that it is done.

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

The sample program *cometcnf.c* shows how to configure the 5136-PFB-VME as a DP master using a binary file exported from the SST ProfiBus Configuration Tool or COM PROFIBUS.

3.7.5 Configuring the DP Master from the Host

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

> • set any required global options, in the DP master global control register, vmeMasCntrlCfg
>
> • set minimum and maximum I/O scan time limits
>
> • create a master control block for each slave it is scanning

49

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

Information on how to set these quantities is found in the following sections.

3.7.6 DP Master Global Control Register

The host uses the DP master global control register, vmeMasCntrlCfg, in the PROFI_USR structure to set options for and control the overall operation of the card 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, VME_MAS_CTRL_RUN_MODE, while it is online, to set the scanning mode of the DP master. If the bit is 1, the card scans I/O in run mode. In run mode, the card reads inputs and updates outputs. If the bit is 0, the card is in program (stop) mode. In program mode, the card reads inputs and sends all 0s for outputs.

The host sets bit 2, VME_MAS_CTRL_USR_OFS, while it is configuring the card as a master, to tell the card that the host is going to assign offsets for the received and transmitted data and that the card should not assign these offsets. The host must do this before it puts the card online as a master. If using this feature, your application is responsible for assigning offsets to data in the input and output pages. This feature may be useful in applications such as embedded applications where it may be difficult if the card reassigns offsets each time the I/O configuration changes.

If you enable this option, assign offsets for all slaves you are scanning. The card checks the offsets assigned for overlaps and does not go online if it finds any.

50

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

The host sets bit 4, `VME_MAS_CTRL_DIS_LED`, while it is configuring the card as a DP master, to disable showing the master status on the system status LED.

The host sets bit 5, `VME_MAS_CTRL_HOLD_INTR`, while it is configuring the card as a DP master, to tell the card 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, `VME_MAS_CTRL_EVT_SCAN_DONE`, while it is configuring the card as a DP master, to tell the card to generate an event every time the I/O scan is complete.

The card sets bit 7, `VME_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 putting the card online, the card checks this bit and, if it is set, the card does not reassign the offsets. The host should not change the state of this bit.

3.7.7 Scan Time Limits

The host can set a minimum I/O scan time by writing to the vmeMasMinIoCycTme register. The value can range from 0 to 65535. The units are 100 µs increments so the time can range from 0 to 6.6635 seconds. This feature is sometimes required because some 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 you are scanning.

The host can set the maximum I/O scan time by writing to the vmeMasMaxIoCycTme register. The value can range 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, 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.

3.7.8 Master Control Blocks

For each slave there must be a master control block. Each master control block occupies 128 bytes. The contents of the master control block are given in the MAS_CNTRL structure in *profictl.h*.

Some elements of the master control block are used for configuration data; others are used by the card for status information about the slave.

If you are configuring the card as a DP master using a binary file exported from the SST ProfiBus Configuration Tool or COM PROFIBUS, the card software builds the master control block for each slave from the data in the *.2bf* binary file.

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

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

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 can range from 0 to 125.

Slave Control/Config Register

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

The host sets bit 0, MAS_CTL_IGNORE_STS, to tell the card to ignore the status of this master block in the display of the DP master status on the LED and in the global status register, vmeMasSts.

The host sets bit 1, MAS_CTL_EVT_RX_CHG, to tell the card to generate an event in the event queue (and optionally an interrupt) when input data for this slave changes.

The host sets bit 2, MAS_CTL_RX_BYTE_SWAP, to tell the card to swap the order of bytes in received data.

The host sets bit 3, MAS_CTL_EVT_UPDTE, to generate an event in the event queue (and optionally an interrupt) when the slave has been updated.

52

The host sets bit 7, `MAS_CTL_ENABLE`, to tell the card to enable this slave. The card does not actually scan the slave until you put the card online. Setting this bit just tells the card that you want to scan this slave when the card goes online.

Note

Do not set both `MAS_CTL_EVT_RX_CHG` and `MAS_CTL_EVT_UPDTE`.

Data Length and Location

One page of card shared memory is reserved for input data (from slaves) and a second page 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 to masTxDataLen.

Normally the card assigns the offsets where the data is stored when you put the card 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 card 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 card as a DP master. If you choose to assign data offsets rather than having the card assign them, set the `VME_MAS_CTRL_USR_OFS` bit in the vmeMasCntrlCfg 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 3FF8h and must be on 8-byte boundaries. If assigning offsets, assign the offset for ALL slaves.

The card checks for overlaps in the data for different slaves before it goes online.

53

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 ident (2 bytes)
- group ID

In addition, the master can send up to 25 bytes of extended parameter data (up to 237 bytes with extensions). Refer to the *ProfiBus Specification* for detailed information on the parameter data.

Master Status Byte

The master uses bits in the master station status byte, masParmMasSts, to lock/unlock access to the slave from other masters, to request sync/freeze support and to enable/disable the master watchdog. Refer to the *ProfiBus Specification* for further details.

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.

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 and corresponds to vmeReadyTime 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.

54

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.

Group ID

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

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 vmeMasCntrlExtFree 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; the card does not use this location.

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 startup sequence for the slave. 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, etc.

55

The identifier byte has the following format (from the ProfiBus specification):

```
        MSB                     LSB

        +--+--+--+--+--+--+--+--+
Bit-No. !7 !6 !5 !4 !3 !2 !1 !0 !

        +--+--+--+--+--+--+--+--+
        !  !  !  !  !  !  !  !  !    meaning:

        !  !  !  !  !  !  !  !
        !  !  !  !  +--+--+--+---- length of data
        !  !  !  !                 00 =  1 byte/word
        !  !  !  !                 .
        !  !  !  !                 .
        !  !  !  !                 15 = 16 byte/words
        !  !  !  !
        !  !  !  !
        !  !  +--+---------------- input/output
        !  !                00= specific identifier formats
        !  !                01= input
        !  !                10= output
        !  !                11= input-output
        !  !
        !  +--------------------- length      format
        !                      0  byte    byte structure
        !                      1  word     word structure
        !
        +----------------------- consistency over
                                0  byte or word
                                1  whole length
```
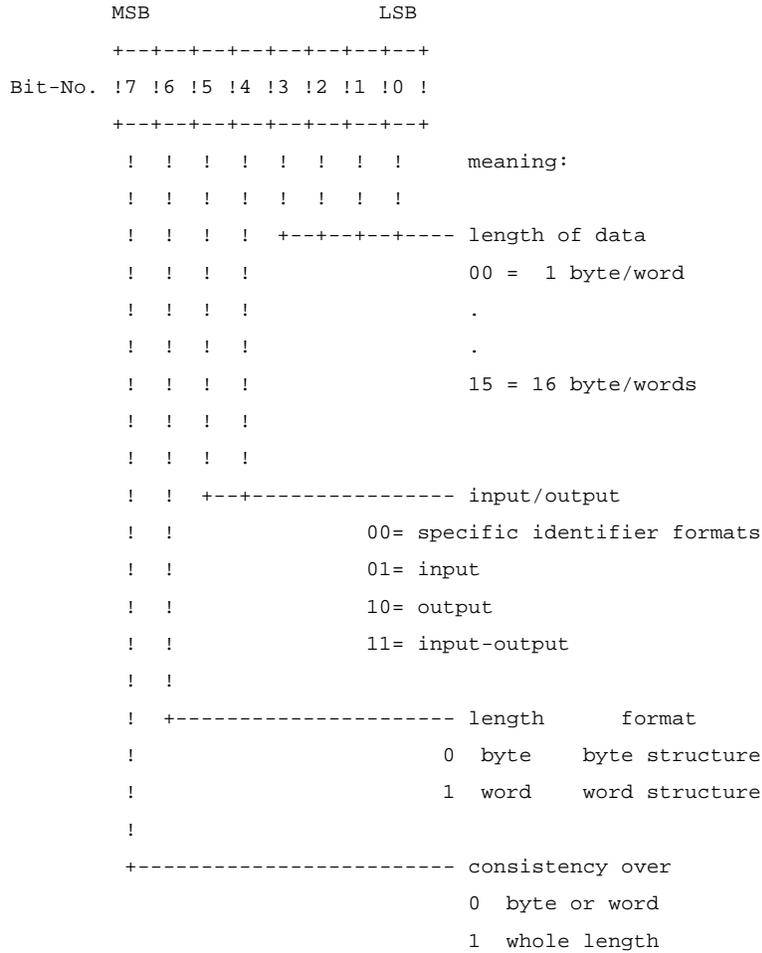
For more information about specific identifier formats or other details, refer to the *ProfiBus Specification*.

The host writes the length (in bytes) of the configuration check data to masChkLen in the master control block for the slave and the configuration check data to the array masChkData.

The check data can be extended up to 244 bytes using the masCntrlExt area in the masCntrl page. This normally happens only when configuring with COM PROFIBUS. 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 vmeMasCntrlExtFree in the home page to keep track of the last offset used. This location is reserved for that purpose but it is up to you to update it; the card does not use this location.

Configuration Check Example

In simple cases:

> • the contents of the lower 4 bits + 1 equals the length of the data in bytes or words, i.e., if the length is 7, the lower 4 bits should be set to 6 or 0110

> • bits 4 and 5 indicate the data type (01=inputs, 10=outputs, 11=input/output)

> • bit 6 indicates whether the length is in bytes or words (0=bytes, 1=words)

> • bit 7 indicates whether the data is consistent over the byte/word (0) or over the whole length (1). This bit should be set to 0.

For an input module with a length of 4 bytes, the configuration check byte would therefore be 0010 0011 = 23h = 35 decimal.

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
> • the slave Ident number
> • the length of the extended diagnostics
> • up to 25 bytes of vendor defined extended diagnostic data

57

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

The card stores the station status bytes in masDiagSts1, masDiagSts2 and masDiagSts3. The card software maintains the station status bytes. Refer to the *ProfiBus Specification* for information on the meaning of bits within these status bytes (from which the following was obtained).

```
Octet 1: Station_status_1

        MSB                   LSB

        +--+--+--+--+--+--+--+--+
 Bit-No. !7 !6 !5 !4 !3 !2 !1 !0 !
        +--+--+--+--+--+--+--+--+
```

The individual bits have the following meaning:

```
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 different from the 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, e. g. wrong length, wrong Ident_Number, invalid parameters.

```
Bit 5: Diag.Invalid_Slave_Response
```

This bit is set by the DP master as soon as receiving 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 was requested which is not supported from this DP slave.

```
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).

58

```
Bit 2: Diag.Cfg_Fault
```

This bit is set by the DP slave as soon as the last received configuration data from the master are different from these which the DP slave has determined.

```
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 respective DP slave can not be reached over the line. 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
        MSB                   LSB
        +--+--+--+--+--+--+--+--+
 Bit-No. !7 !6 !5 !4 !3 !2 !1 !0 !
        +--+--+--+--+--+--+--+--+
```

The individual bits have the following meaning:

```
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 sets this bit always to zero.

```
Bit 6: Reserved
Bit 5: Diag.Sync_Mode
```

This bit is set by the DP slave as soon as the respective DP slave has received the Sync command.

```
Bit 4: Diag.Freeze_Mode
```

This bit is set by the DP slave as soon as the respective DP slave has received the Freeze command.

```
Bit 3: Diag.WD_On (Watchdog on)
```

59

This bit is set by the DP slave. If this bit is set to 1, the watchdog control at 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

        MSB                     LSB

        +--+--+--+--+--+--+--+--+
 Bit-No. !7 !6 !5 !4 !3 !2 !1 !0 !
        +--+--+--+--+--+--+--+--+
```

The individual bits have the following meaning:

```
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; or the DP-Master sets this bit if the DP slave sends more diagnostic information than the DP master can enter in its diagnostic buffer.

```
Bit 0-6: reserved
```

The card stores the station number of the master that parameterized the slave in masDiagMasStn.

The card stores the slave Ident number returned by the slave in masDiagID_hi (high byte) and masDiagID_lo (low byte).

60

The card stores the total current length of the diagnostics in masDiagLen and the maximum possible diagnostics length in masDiagMaxLen. If the maximum diagnostic length is less than 32, the card stores the extended diagnostic data in the masDiagData array with the final two bytes stored 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 since the length of the diagnostic data sent by the slave may change.

Slave Designation Data

The host can assign a designation string of up to 12 characters to each slave. The host stores it in the masDesig array in the master control block 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.

3.7.9 Accessing Input Data

The vmeMasRxPage register, in the `PROFI_USR` structure, contains the page number of the input data memory page.

Within that page, the host uses two elements in the control block for the slave to locate the input data. masRxDataOfs contains the offset to the input data for the slave. masRxDataLen contains the input data length for the slave, in bytes.

All 16-bit values are sent on ProfiBus in high-byte low-byte order. If you set the `MAS_CTL_RX_BYTE_SWAP` option bit in the masCntCfg register in the master control block for the slave, the card swaps bytes before it writes them to the data area for the slave.

If word values are kept on even boundaries and accessed as words, word consistency is guaranteed.

If you have some byte modules and some word modules, you are probably better off doing the swapping in the host since the card swaps either all values or none.

3.7.10 Accessing Output Data

The vmeMasTxPage register contains the page number of the output data memory page.

Within that page, the host uses two elements in the control block for the slave to locate the output data. masTxDataOfs contains the offset to the output data for the slave. masTxDataLen contains the output data length, in bytes.

If the host needs to swap bytes for transmit data, it must do so before writing the data to the card since all 16-bit values are sent on ProfiBus in high-byte low-byte order.

If word values are kept on even boundaries and written as words, word consistency is guaranteed.

3.7.11 Monitoring Slave Status

Your application can monitor the status of all configured slaves by reading the DP master global status register.

The application can use the master status cross reference table to quickly check the status of each slave and to determine the master control block associated with each slave.

The application can check the status of an individual slave by reading the slave status register in the master control block for the slave. It can use the slave error register in the master control block for the slave to determine the cause of slave errors.

DP Master Global Status Register

The DP master global status register, vmeMasSts in the PROFI_USR structure, shows the status of the overall operation of the card as a DP master.

The card sets bit 0, VME_MAS_STS_ALL_OK, to 1 to indicate that all configured slaves are being scanned with no problems. If this bit is 0, there are problems with one or more slaves. Check the status of each slave to determine where the problems are.

Master Status Cross Reference Table

The master status cross reference table, vmeMasStsTab, 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.

Slave Status Register

The Slave Status register, masStatus, in the master control block for each slave, shows the status of that slave. This register should not be changed by the host.

The card sets bit 7, MAS_STS_OK, to 1 if the current status of the slave is OK. If this bit is zero, there is a problem with the slave.

63

Slave Error Register

The card sets various values in the error register, masError in the master control block for the slave, to indicate the cause of any problems with the slave. Some errors occur during parameterizing the slave, 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	Cause
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

64

Extended Error Register

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

```
masError = MAS_ERR_CFG_FAILURE
```

Value	Description
1	No response or NAK after sending the first diagnostic status request to the slave
2	No response or NAK after sending parameter data to the slave
3	No response or NAK after sending configuration check data to the slave
4	No response or NAK after sending the second diagnostic status request to the slave
5	Invalid response after sending the first diagnostic status request to the slave
6	Invalid response after sending parameter data to the slave
7	Invalid response after sending configuration check data to the slave
8	Response to configuration check packet was non-zero length (slave should never return anything)
9	Invalid response after sending the second diagnostic status request to the slave

```
masError = MAS_ERR_DATA_UPD_FAILURE
```

Value	Description
1	Error in data update during configuration
2	No response or NAK when updating data while online

65

```
masError = MAS_ERR_CFG_DIAG_READ_FAILURE
```

Value	Description
1	Invalid response when reading slave diagnostics while online
2	No response or NAK when reading slave diagnostics while online

```
masError = MAS_ERR_CFG_DIAG_STS1_ERR
```

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	Meaning
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

```
masError = MAS_ERR_UPD_DIAG_STS1_ERR
```

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 and any bits that are set indicate the error. The bits in station status 1 are shown in the above table.

```
masError = MAS_ERR_CFG_DIAG_STS2_ERR
```

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 and any bits that are set in the result indicate the error. Mask the value in masExtErrInfo with 04h and bit 2 should be set.

The bits in station status byte 2 are:

Bit	Meaning
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

```
masError = MAS_ERR_UPD_DIAG_STS2_ERR
```

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 and any bits that are set should not be set. 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.

3.7.12 Diagnostic Event Register

An online slave can request that the master read its diagnostics. The software on the card takes care of reading the diagnostics from the slave. The card then sets bit 0, MAS_DEVT_DIAG_UPD, in the diagnostic event register, masDiagEvent, in the master control block for the slave, to indicate that it has read the slave diagnostics. The host acknowledges by clearing the register so that it can detect when another diagnostic update occurs.

3.7.13 DP Master Events

> **Note**
>
> The event queue and event processing are described in detail in section 3.14, *Events and Interrupts*.

The card can generate events in the event queue to notify the host of various DP master events. You enable these events by setting bits in the DP master global control register, vmeMasCntrlCfg, and in the control/config register, masCntCfg, in the master control block for individual slaves.

The upper byte of the entry in the event queue contains the event type; for the receive data change, slave update events, and the status change events, the lower byte contains the block number of the master control block that generated the event.

Command	Value
EVT_MAS_RX_DATA_CHG	30NNh
EVT_MAS_UPDTE	31NNh
EVT_MAS_SCAN_DONE	3200h
EVT_MAS_OK	33NNh
EVT_MAS_ERROR	34NNh

To enable the EVT_MAS_RX_DATA_CHG event and generate an event when the received data from the slave changes, set bit 1, MAS_CTL_EVT_RX_CHG, in the master control block for the slave.

68

To enable the `EVT_MAS_UPDTE` event and generate an event when the slave update is complete, set bit 3, `MAS_CTL_EVT_UPDTE` in the master control block for the slave.

> **Note**
>
> Do not enable both the `EVT_MAS_RX_DATA_CHG` and `EVT_MAS_UPDTE` events at the same time.

When an `EVT_MAS_RX_DATA_CHG` or `EVT_MAS_UPDTE` event occurs, the card uses bits in the event register, masEvent, in the master control block for the slave. It sets bit 0, `MAS_EVT_UPD`, to indicate that this slave has been updated. It 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, you never get more than one event in this register. The host acknowledges by clearing the bit.

If you do not clear the masEvent register, the card does not generate any further data change or update events for this slave and instead increments the event overrun counter.

To enable the end-of-scan event, `EVT_MAS_SCAN_DONE`, set the `VME_MAS_CTRL_EVT_SCAN_DONE` bit in the vmeMasCntrlCfg register. The card generates an event in the event queue and sets the `VME_MAS_SCAN_DONE` bit in the vmeMasGlbEvt register when the scan is done. The host application must clear the vmeMasGlbEvt register to acknowledge the event or the card will not generate any more end-of-scan events and will instead increment the event overrun counter.

You can disable the `EVT_MAS_OK` and `EVT_MAS_ERROR` events only by setting the `MAS_CTL_IGNORE_STS` in the masCntCfg register of the master control block for the slave.

3.7.14 Using Flash Memory

The host can write the current DP master configuration data to flash memory and read the DP master configuration from flash. Refer to section 3.9, *Network, DP Master and DP Slave Data in Flash*, for information on how to do this.

3.7.15 Sample Programs

dpmascfg.c shows how to configure the DP master functions of the 5136-PFB-VME.

dpmasdmp.c shows how to dump a configuration to a text file that can be read by *dpmascfg.c*.

ssbincfg.c shows how to configure the card as a DP master using a file exported from the SST ProfiBus Configuration Tool or the Siemens COM PROFIBUS software.

dpmon.c shows how to access data, and diagnostics for both the DP master and the DP slave.

3.7.16 What Happens When the Master Brings a Slave Online...

The following sequence takes place when a master puts a slave online:

1. The master sends a diagnostic read (SRD).

2. The slave responds with its diagnostic data (DL, data low).

3. The master, if everything in the diagnostic data is OK, then sends parameterization data to the slave (SRD).

4. The slave responds with a DL or ACK.

5. The master sends check data packet. See the *ProfiBus Specification*.

6. The slave replies with a DL or ACK.

7. The master sends a diagnostic read (SRD).

8. The slave replies with its diagnostic data (DL). If there are any problems with anything up to this point, the slave says so now in its reply. Otherwise, cyclic data transfer begins between the master and the slave and the slave is considered to be online.

70

3.7.17 DP Auto-configuration

The ProfiBus DP protocol requires the master device be given a configuration for each of the slave devices it controls. When the master device is put online, it first "configures" the list of slave devices for which it has configuration information. The configuration cycle includes checking the master's configuration information against what the slave devices actually expect. An incorrect configuration of a master device is the most common error with ProfiBus DP networks. The configuration information is provided to the master device via a configuration tool.

SST ProfiBus products have implemented a solution to this common problem. The card has the ability to scan the currently attached network of slave devices for the configuration information normally supplied to the master device by the user.

The ProfiBus DP protocol does not explicitly support any auto-identification or auto-configuration services and therefore the success of the SST ProfiBus auto-configuration feature may not work successfullywith all slave devices.

What SST ProfiBus Card Settings Are Required?

The card must go online to get the current list of slave devices and their configuration spaces. Therefore, the SST ProfiBus card needs a valid network configuration. The only settings required to do a DP auto-configuration command are baud rate, station address and active station setting. The remaining network settings (see the *Hardware User's Guide* for details) can use the default settings, unless your network has specific needs or deviates from the default ProfiBus DP network set up.

Once the DP auto-configure command is issued and completed, the card goes offline again and contains the configuration of any DP slave devices found on the network.

71

Issuing an Auto-configuration Command

Using the Win32 Example Applications

The *vmewin32* software installation includes a set of example applications that exercise the features of the SST ProfiBus card. Additionally, this software's source code is included with the executable programs to provide software developers a set of working examples on the use of the SST ProfiBus products. For help using these applications, a Quick Start guide, in Windows Help File format, is also included with the software.

To auto-configure a 5136-PFB-VME DP Master:

1. Run the VME Network Configuration program (*5136-PFB-32/ Configuration Tools/PFB Network Configuration*) and select the VME card you wish to use from the *Open Card* dialog.

2. Set the station address, baud rate and check the active station option.

3. Click *OK* to close the program

4. Run the VME Command utility (*5136-PFB-32/Configuration Tools/ VME Command*) and again pick the VME card you used in steps 1 through 3.

5. Choose the *auto_dp_config* option from the pull-down menu and click *Execute*. The SST ProfiBus card goes online and searches for configurable DP slave devices. The operation is complete when the command status returns as completed.

To test the results of an *auto_dp_cfg* command:

Still using the VME Command utility, choose the *on_line* command.

Open the DP Monitor program (*5136-PFB-32/Monitor Utilities/DP Monitor*) and choose the card used for the *auto_dp_cfg* command.

6. Move to the *Master Directory* tab to view the state of the configured DP slaves. This list of slaves is derived from the auto-configured slaves found on the network during the configuration command.

72

Application Developer's Notes on the DP Auto-configuration Command

If you are an application developer using the SST ProfiBus card family, you can access the AUTO_DP_CFG command through the vmeCommand register found in the PROFI_USR structure (see *ProfiCtl.h* for details). The minimum setup requirements for the network parameters remain the same as defined earlier in this application note. Namely, the baud rate, station address and active station settings must be initialized prior to issuing the DP auto-configuration command.

The shared RAM interface layout contained in the C header file *ProfiCtl.h* (version 1.10 or greater) defines the CMD_AUTODP_CFG (HEX 09) command. Issue this command to the vmeCommand register to start the auto-configuration cycle. The cycle is complete when the vmeCommand register returns to the CMD_OFF state or, if an error occurs, the CMD_ERROR state. In the event of an error, the vmeStatus register contains an error code. These codes are also defined in the *ProfiCtl.h* file.

Why Does A DP Slave Not Auto-configure?

As stated previously, the ProfiBus DP protocol does not explicitly support a set of auto-configuration services. Therefore, it is possible that some slave devices do not provide enough information to the SST ProfiBus card to allow for a successful auto-configuration. There are two common states for a DP slave after an auto-configuration cycle executes and the DP master goes online:

1. The DP slave appears in the *Master Directory* (DP Monitor application) but reports an error condition. This is generally because the DP slave uses extended user parameters and the information about these parameters cannot be interrogated by a DP auto-configuration operation.

2. The DP slave does not appear as part of the *Master Directory* list (configured scan list). There are two reasons for this result:

   a) The station in question is not a DP slave device and/or has a bus wiring fault/cable fault.

   b) The DP slave device is currently locked by another master or indicates it is not free to be scanned by a DP Master device.

73

There is no solution to the problem noted by item (1). These types of slave devices must be configured in the traditional manner – with a DP Configuration Tool. The DP auto-configuration cycle has still provided useful information about the amount of I/O contained in the slave device and these settings should be noted for future use.

In the case of problem 2a, the DP Monitor program can be used to determine if the slave device appears in the station list. Move to the *DP Network* tab of the DP Monitor application and click on *Update Passive*. If the DP slave in question does not appear at its configured station address, check the wiring of the bus to this slave device. If the DP slave appears in the station address list, ensure no other DP Masters are scanning this station address.

74

3.8   Using the Card as a DP Slave

The card can transmit and receive up to 244 bytes of data as a slave on a DP network. All variables and tables for using the card as a DP slave are contained in the PROFI_USR structure and are all found on the home page of card memory.

If you are using the card only as a DP slave, set the network parameters before putting the card online. Refer to section 3.6, *Network Parameters*, for information on how to set network parameters.

3.8.1   What the Host has to Configure...

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, including station number, baud rate, high station address.
>
> • Slave Ident if it is different from the default (the default is 6715h)
>
> • Receive data length, default is 0
>
> • Transmit data length, default is 0
>
> • Any required slave options, in the slvCntCfg register

The card assigns the slave defaults when you load the VMEPROFI module. If reconfiguring the slave, be sure to restore the default values before assigning any new values.

3.8.2   DP Slave Data Coherency Mode

The DP Slave Services currently ensure 16-bit data integrity on the 5136-PFB-VME card. The Host application can ensure wider data integrity if required. A custom algorithm must be implemented to support entire DP slave data consistency.

3.8.3   DP Slave Control/Config Register

The host controls and sets options for the operation of the card as a 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 card online. The host also sets bits in this register to perform DP slave operations while the card is online.

75

The host sets bit 0, `SLV_CTL_DIAG_UPD`, while it is online, to request a diagnostic read from the master. Setting this bit causes the card to send its next reply to the master as a data high priority packet. When the master sees the high priority packet, it does a diagnostic status read from the slave.

The host sets bit 1, `SLV_CTL_EVT_RX_CHG`, when it is configuring the slave, to tell the card to generate an event in the event queue when the received data changes. See section 3.14, *Events and Interrupts*, for more information on processing events. Do not set both this bit and the `SLV_CTL_EVT_UPDTE` bit on.

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

The host sets bit 3, `SLV_CTL_IGN_SYNC_FRZ_ERR`, while configuring the slave, to tell the card how to respond when the master requests sync or freeze while it is parameterizing the slave. If the bit is 0, the card 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 card does not report that it does not support sync or freeze. The card then ignores any subsequent sync or freeze commands from the master and updates outputs when they are received. The default is 0.

The host sets bit 4, `SLV_CTL_RX_BYTE_SWAP`, when configuring the slave, to tell the card to swap the upper and lower bytes of received data. Note that transmitted data goes out directly. If you need to swap the bytes on transmit data, your application must swap the bytes before it writes them to the card. If you enable byte-swapping on received data, there is a performance penalty since the processor on the card must constantly swap bytes before it writes them to the data table. The default is 0 and the bytes are not swapped. If you have some byte and some word values, you are probably better off swapping bytes in your application.

The host sets bit 5, `SLV_CTL_EVT_UPDTE`, while configuring the slave, to tell the card to generate an event whenever the slave is updated by the master. Do not set both this bit and the `SLV_CTL_EVT_RX_CHG` bit on.

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

76

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

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

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

3.8.4 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.

3.8.5 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.

3.8.6 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 card 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)
> • length of 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.

77

Station Status Bytes

The three station status bytes are maintained by the card and should not be written to by the host. Refer to the ProfiBus specification for detailed information on what these bytes contain. The station status bytes are:

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

Master that Parameterized Slave

The card stores the station number of the master that set the parameters of the slave in slvMasStn. The card writes the value when the master set the parameters of the card as a DP slave and this register should not be written to by the host application.

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 75h). The Ident number is unique for each ProfiBus I/O device. 0875 is registered for the 5136-PFB-VME and will not conflict with any other registered device.

Change the Ident number on the 5136-PFB-VME from the default if you want to emulate some other device.

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 used 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 array slvDiag.

78

Sending Diagnostic Data While Online

When the card 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. When the master has read the diagnostic data, the card sets bit 0, `SLV_DEVT_DIAG_UPD`, in the DP slave diagnostic event register, slvDiagEvent to indicate that the diagnostics have been read by the master.

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 card can recognize a new request from the host to update the diagnostics) and by clearing the slvDiagEvent register to 0.

3.8.7 Master Parameter Data

As part of the startup process, the master sets the parameters of the slave. 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

Master Station Status

The card stores the master station status register in slvMasSts. Refer to the *ProfiBus Specification* for detailed information on what this register contains.

Watchdog Factors

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

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
```

79

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. Allowed values are 1 to 255. The card defaults to the network setting.

The host can force the card 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. Refer to section 3.8.3, *DP Slave Control/Config Register*, for details.

Generally, you use the same value everywhere on the network, and use the same value in the slave as in the master.

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 card 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.

Group Ident

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

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. 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.

3.8.8 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 on such things as the number of inputs and outputs, etc. Typically, each byte of configuration check data represents one slot. Refer to the *ProfiBus Specification* for detailed information on what the configuration check data contains.

80

The VMEPROFI 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 card 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, it indicates an error in its final diagnostic reply to the master and also sends an error to the host application.

3.8.9 Configuring the Slave

To complete the configuration after the host has written all the required configuration data to the card, the host sets the `SLV_CTL_ENABLE` bit in the slave Control/config register, slvCntCfg. The card is then ready to be put online. The card does not check any of the parameters entered until you put the card online.

If you are using the card as a DP master or for some other purpose in addition to using it as a DP slave, configure these functions before putting the card online.

The host should then put the card online and check the slave status register for any problems.

3.8.10 Accessing I/O Data

Received Data

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

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

Transmit Data

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

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

81

3.8.11 DP Slave Status Register

The card reports the status of the DP slave operation by setting bits in the slvStatus register.

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

The card sets bit 7, SLV_STS_OK, if the current slave status is OK. This means setting the parameters was successful and the slave watchdog has not timed out.

3.8.12 DP Slave Error Register

The card sets the slave error register, slvError, to the following values to report various error conditions. 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 us is incorrect
SLV_ERR_TX_LEN_MISM	05h	Length of data from us to master is incorrect
SLV_ERR_WD_FACT_INV	06h	slvWdFact1 or slvWdFact2 from master was 0
SLV_ERR_TIME_OUT	07h	Slave watchdog time out (check response timeout)
SLV_ERR_WARN_WD_DIS	08h	Slave timeout watchdog disabled from master

All these errors except 07h happen when the slave parameters are set up by the 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 card is different from the value configured in the master. To determine the value configured in the master, read slvReadyTime. Override the value sent by the master by setting the SLV_CTL_FORCE_READY_TIME bit in the slvCntCfg register while you are configuring the slave. The card can communicate as a slave even if the times are different but you may experience network errors.

83

If the value is `SLV_ERR_UNSUP_REQ`, the master has requested Sync or Freeze during parameterization, which the card does not support.

If the value is `SLV_ERR_RX_LEN_MISM,` the data received from the master has a length different from the length configured on the card. Read slvReqRxDataLen to determine the Receive data length requested by the master. If there is a receive length mismatch, the card 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 card 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. See section 3.8.7, *Master Parameter Data*.

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.

3.8.13 Slave Events

The card can generate the following events in the event queue related to the operation of the card as DP slave.

Event	Value
EVT_SLV_RX_DATA_CHG	4000h
EVT_SLV_UPDTE	4100h
EVT_SLV_OK	4200h
EVT_SLV_ERROR	4300h
EVT_SLV_CHG_TO_RUN	4400h
EVT_SLV_CHG_TO_STOP	4500h

The low byte of a DP slave event is always zero.

Note

The event queue and event processing are described in detail in section 3.14, *Events and Interrupts*.

Enabling Slave Events

To enable the `EVT_SLV_RX_DATA_CHG` event and generate an event in the event queue when the received data changes, set bit 1, `SLV_CTL_EVT_RX_CHG`, in the slvCntCfg register.

To enable the `EVT_SLV_UPDTE` event and generate an event whenever the slave is updated by the master, set bit 5, `SLV_CTL_EVT_UPDTE`, in the slvCntCfg register.

Caution

Do not enable both the `EVT_SLV_RX_DATA_CHG` and `EVT_SLV_UPDTE` events at the same time.

The `EVT_SLV_OK` and `EVT_SLV_ERROR` events are always enabled unless you set bit 7, `SLV_CTL_IGNORE_STS` bit in the slvCntCfg register.

To enable the `EVT_SLV_CHG_TO_RUN` and `EVT_SLV_CHG_TO_STOP` events and generate an event whenever the master mode changes, set bit 6, `SLV_CTL_EVT_MODE_CHANGE`, in the slvCntCfg register.

DP Slave Event Register

The card sets bits in the DP slave event register, slvEvent, whenever it generates a DP slave event.

The card sets bit 0, `SLV_EVT_UPD`, when the master sends an I/O update to the slave.

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

When an event occurs, the card sets the appropriate bit, then puts the event into the event queue. The host acknowledges by processing the event and then clearing this byte. The card will not generate a new received data change or update event until this byte has been cleared.

3.8.14 Master Control Commands

The master can send special control commands to one slave or to a group of slaves. These commands include the sync and freeze commands and a clear data command. The slave accepts these commands only from the master that has parameterized it. Refer to the *ProfiBus Specification* for details.

The card stores any such command it receives in the slvGlbCntrl[2] array.

3.8.15 Using Flash Memory with the DP Slave

The host can write the current slave configuration data to flash memory and read the slave configuration from flash. Refer to section 3.9, *Network, DP Master and DP Slave Data in Flash*, for information on how to do this.

3.8.16 Updating COM PROFIBUS to include the 5136-PFB-VME as a Slave

To update Siemens COM PROFIBUS, install the files for the card using the batch file provided on the installation disk. The path to the file is:

dlink32/5136-pfb/comet/updcomet.bat

The card then appears in the OTHERS family when configuring slaves. If configuring a master system using something other than COM PROFIBUS, and what you are using does not support the 5136-PFB-VME as a DP slave, change the card's Ident so that it emulates an IM 318B and then configure the card as an IM 318B in the configuration software.

3.8.17 Sample Programs

dpslvcfg.c shows how to configure the slave parameters.

dpslvdmp.c shows how to dump the slave parameters to a text file that *dpslvcfg* can read.

dpmon.c shows how to access DP master and slave data and diagnostics.

pbnetcfg.c shows how to configure network parameters.

pbcmd.c shows how to use the command register.

86

3.8.18 What Happens When the Slave Goes Online...

The following sequence takes place when a master puts a slave online:

1. The master sends a diagnostic status read (SRD).

2. The slave responds with its diagnostic status data (DL).

3. The master, if everything in the diagnostic data is OK, sends parameterization data to the slave (SRD).

4. The slave responds with an ACK.

5. The master sends a configuration check data packet, see the *ProfiBus Specification*.

6. The slave replies with an ACK.

7. The master sends a diagnostic status read (SRD).

8. The slave replies with its diagnostic status data (DL). If there are any problems with anything up to this point, the slave says so now in its reply and the master goes back to the beginning and tries to bring the slave online the next time it scans it. Otherwise, cyclic data transfer begins between the master and the slave.

87

3.9 Network, DP Master and DP Slave Data in Flash

The host can write the current network, DP master, and DP slave configuration data to flash memory and configure the card from configuration data stored in flash.

The following commands relate to the use of flash memory.

Command	Value	Description
CMD_CFG_FROM_FLASH	11h	Configure card from configuration in flash
CMD_CFG_FLASH_GO_ON	14h	Configure card from flash, then go online
CMD_PGM_TO_FLASH	21h	Burn card configuration into flash

To write the network, DP master, and DP slave configuration to flash, issue the CMD_PGM_TO_FLASH command (21h).

To configure the card using the network, DP master, and DP slave configuration data already stored in flash, issue the CMD_CFG_FROM_FLASH command (11h).

To configure card using the network, DP master, and DP slave configuration data already stored in flash, then go online, issue the CMD_CFG_FLASH_GO_ON command (14h).

Refer to section 3.3, *Command Register*, for information on issuing commands.

88

3.10 Using the Card for FDL (Layer 2) Messaging

The 5136-PFB-VME can send and receive FDL (layer 2) messages. Features include:

- configure up to 64 Service Access Points (SAPs)
- update timeout on any SAP
- supports up to 128 request blocks at one time
- supports periodic and one shot requests
- generates events and optional interrupts on received data change

The card must be offline to configure layer 2 SAPs and message blocks.

Ensure that the network parameters have been configured before putting the card online. Refer to section 3.6, *Network Parameters* for information on what the network parameters are and how to set them.

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

3.10.1 FDL Global Control Register

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

The host sets bit 12, `LAY2_CTL_MSG_DIS_LED`, to disable the display of the layer 2 SAP status on the system status LED.

The host sets bit 13, `LAY2_CTL_SAP_DIS_LED`, to disable the display of the layer 2 SAP status on the system status LED.

The host sets bit 14, `LAY2_CTL_SAP_ENABLE`, to enable the Layer 2 SAP service. The host must set this bit before it issues the online command.

The host sets bit 15, `LAY2_CTL_MSG_ENABLE`, to enable the Layer 2 Message service. The host must set this bit before it issues the online command.

3.10.2 FDL Global Status Register

The card sets bits in the FDL global status register, lay2Status, in the *profi_usr* structure, to show the overall status of layer 2 operations.

89

The card sets bit 14, `LAY2_STS_ALL_SAP_OK`, to indicate that all configured layer 2 SAPs are OK.

The card sets bit 15, `LAY2_STS_ALL_MSG_OK`, to indicate that all configured layer 2 messages are OK.

3.10.3 The Trigger Queue

The host controls message transmission and online changes to SAPs through the trigger queue, vmeTrigQueue[256], in the VMEPROFI structure. The host accesses the trigger queue by means of the trigger queue head pointer, vmeTrgHead, in the VMEPROFI structure. To insert triggers, the host puts the appropriate trigger value in the trigger queue at the head position, then increments the head pointer.

The card writes to the queue tail pointer, vmeTrgTail. 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.

3.10.4 FDL (Layer 2) SAPs

The 5136-PFB-VME supports up to 64 service access points. You configure a SAP by creating a SAP control block.

SAP Control Blocks

The Lay2sCntrl[64] array in the *profi_usr* structure contains the SAP control blocks. The SAP control block structures are defined as `LAY2S_CNTRL` in *profictl.h*. Each SAP control block corresponds to a specific SAP number, for example, block 3 corresponds to SAP 3.

90

Creating a 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 card operations (DP master, DP slave, FDL messages) then puts the card online. When the card goes online, it allocates the memory locations for the receive and transmit data for the SAP. The host can then access the data, using the page and offset values provided by the card.

91

SAP Control and Configuration Register

The host sets various options for each SAP by setting bits in the control and configuration register, lay2sCntCfg. These bits must be set when the card is offline and you are configuring the SAP, with the exception of the `LAY2S_CTL_IGNORE_STS` bit, which can be changed while you are online.

The host sets bit 2, `LAY2S_CTL_RX_BYTE_SWAP`, to tell the card to swap 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 the data to the card.

The host sets bit 3, `LAY2S_CTL_EVT_RX_CHG`, to tell the card to generate an event in the event queue when the received data for this SAP changes. Do not enable both received data change and SAP update events.

The host sets bit 4, `LAY2S_CTL_IGNORE_STS`, to tell the card to ignore 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.

The host sets bit 5, `LAY2S_CTL_EVT_UPDTE`, to tell the card to generate an event in the event queue whenever this SAP is updated. Do not enable both received data change and SAP update events.

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 or FMS function configured on this card. Possible SAP types are:

Type	Value
LAY2S_TYP_NOT_DEF	00h
LAY2S_TYP_DP_SLAVE	01h
LAY2S_TYP_LAYER2_SAP	02h
LAY2S_TYP_FMS	03h
LAY2S_TYP_MPI_SAP	04h
LAY2S_TYP_DP_MAS_MC2	05h

92

Strict Station Checking

To accept messages only from a specific station, set lay2sStn to the source station number. Otherwise, set it FFh to disable strict station checking. The valid range of stations is 0 to 126.

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

Strict Source SAP Checking

To accept messages only from a specific source SAP, set lay2sSap to the source SAP number. To disable strict source SAP checking, set it 0FFh. The valid range of source SAPs is 0 to 63.

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

Strict Frame Control Checking

To accept only a specific message type, set the lay2sFrmCntrl register to the message type from the following table:

Name	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_HI	03h	Accept only SDN hi or lo priority
LAY2S_FC_SDA_LO	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	
LAY2S_FC_DDB_REQ_LO	0Dh	
LAY2S_FC_DDB_REQ_HI	0Eh	
LAY2S_FC_DDB_REQ_LO_HI	0Fh	

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

93

Received Data

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

When putting the card online, the card assigns the location where it stores the received data for this SAP. The card writes the data page for the data 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 card writes the length of the actual received data for this SAP in lay2sRxLen.

If the card receives a message with a data length greater than the value in lay2sRxMaxLen, it replies with a NAK (RR) and discards the data.

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. Change the value in lay2sTxLen to vary the reply length while online. If you try to set the reply length to a value greater than the maximum reply length, the card generates an error and does not change the current length.

The allowed range of values for lay2sTxMaxLen and lay2sTxLen is 0 to 244.

When putting the card online, the card assigns the location for the response data. The card 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.

Timeout

The host sets the timeout for this SAP by writing to lay2sTimeOut. The timeout is in units of 10 ms and must be in the range 1 to 8190. The maximum timeout is therefore 81900 ms, or 81.9 seconds. To disable the timeout for this SAP, write 0 to this register.

If the SAP is not updated within the timeout period, the card generates an error.

94

SAP Updates

Whenever a SAP is updated, the card 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.

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.

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, 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 3.10.3, *The Trigger Queue* for information on using the trigger queue.

Status Register

The card uses the lay2sStatus register to indicate the operating status of the SAP. If bit 7, LAY2S_STS_OK, is set, this SAP is operating with no problems. If this bit is 0, there is a problem with this SAP.

95

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	Meaning
0h	OK
1h	User error, SAP locked
2h	No resource for send data, tried to send to SAP that was not configured
3h	No service available (SAP does not exist)
4h	Access point blocked
80h	Short character, problems with wiring, termination.
9Fh	No access
AFh	Double token detected, problems with wiring, termination.
BFh	Response buffer too small
8Fh	Noise at SM command, problems with wiring, termination.

Error Register

The error register, lay2sError, indicates the cause of a problem with this SAP 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.

96

SAP Events

The VMEPROFI module can generate various events in the event queue related to FDL SAPs. The following table shows the upper byte of the event queue entry for the event. The lower byte contains the SAP 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

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

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

The card sets bit 1, LAY2S_EVT_RX_DATA_CHG, to indicate that data for this SAP has been received and it is different from the previous data.

The host acknowledges the event by clearing the bit. The card will not generate another received data change or update event for this SAP unless the lay2sEvent register is 0 and will instead increment the event overrun counter.

SAP Diagnostic Counters

The card also maintains diagnostic counters that show the operation of the layer 2 SAPs. Refer to section 3.11.4, *FDL (Layer 2) Statistics* for a description of these counters.

97

3.10.5 FDL (Layer 2) Messages

You can use the card to send FDL messages. To send a message you must set up a control message block for the message before you put the card online. The VMEPROFI module supports up to 128 message blocks.

You send messages by putting them in the trigger queue (vmeTrigQueue). You can set up messages to be:

- one-shot
- periodic, with periods from 1 µs to 16.4 seconds

The card takes care of the details of rescheduling and sending periodic messages.

The card must be an active station on the network to send messages.

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 defined as `LAY2M_CNTRL` in *profictl.h*.

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

- 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 received data change or message confirmation
- whether the message is periodic
- whether the message should be retried on errors
- whether the card should give the message high priority processing in its internal queue

98

- whether the received data should be byte swapped

- whether the card 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 you must set before you put the card online are the maximum transmit data length and the maximum receive data length. The card needs these two values to allocate memory for the message. However, 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 card operations (FDL SAPs, DP master, DP slave), it puts the card online. The card then allocates the transmit and receive data locations for the message block. The host can then fill in any missing components of the message block and put the message in the message queue.

Message Control Blocks

Message control blocks contain the following elements:

Control and Configuration Register

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

The host sets bit 0, `LAY2M_CTL_PERIODIC`, to make this a periodic message. If the message is periodic, the host must also write the message update time to lay2mUpdTime (see below).

The host sets bit 1, `LAY2M_CTL_RETRY_PERIODIC`, to tell the card to retry this message if there are problems. If the card is to retry the message, the host must also write the error retry time to lay2mErrTime (see below).

The host sets bit 2, `LAY2M_CTL_HI_PRI`, to send the message to the high priority queue. The card maintains two message queues. The card can send one message from the high priority queue when it gets the token, even if the token hold time has expired. This high priority refers to how the message is processed by the card in getting it out on the network and is different from a high priority frame control value.

99

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

The host sets bit 3, `LAY2M_CTL_EVT_RX_CHG`, to enable generation of an event in the event queue on received data change.

> ⚠️ **Caution**
>
> Do not enable both receive data change and message confirmation events

The host sets bit 4, `LAY2M_CTL_IGNORE_STS`, to tell the card not to include this message block in the LED status display for FDL messages and also in the global message bit in the FDL status register, lay2Status.

The host sets bit 5, `LAY2M_CTL_EVT_CONFIRM`, to tell the card to generate an event in the event queue whenever this message is confirmed by the receiver. Do not enable both receive data change and message confirmation events.

The host sets bit 6, `LAY2M_CTL_RX_BYTE_SWAP`, to tell the card to swap the bytes in received data. If the host needs to swap bytes in transmitted data, it must swap the data before writing to the card.

Destination Station

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

The card can send multicast messages by setting the destination station to 127 and the destination SAP to whatever SAP is configured in the receivers to process multicast messages.

The card can send a broadcast message by setting the destination station to 127 and the destination SAP to 63.

Destination SAP

The host writes the destination SAP (if required) to lay2mDstSap. Valid entries are 0 to 63. To disable sending the destination SAP, write 255 (FFh). For example, disable sending the destination SAP if sending the message to a default SAP on the destination station.

When sending a message with a destination SAP, the card automatically sets the high bit in the destination station number when it sends the message. Do not set the high bit when you set the destination station number in lay2mDstStn.

100

Source SAP

The host may set the source SAP in the message by writing the source SAP number to lay2mSrcSap. Valid entries are 0 to 63. To disable sending the source SAP, write 255 (FFh).

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	SM Send Data No ack
FC_SmSRD	01h	SM 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

Transmit Data

The host sets the maximum transmit data (request) length in bytes for this message by writing to the lay2mTxMaxLen register and the current length in bytes in lay2mTxLen. Usually these lengths are equal. However the value in lay2mTxLen can be less than the value in lay2mTxmaxLen.

101

Possible lengths are 0 to 244.

Change the value in lay2mTxLen and vary the message length while you are online. However, for periodic messages, stop and restart the message for the change to take effect.

When the card goes online, it assigns the location of the transmit data for the message. The card 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.

Receive Data

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

When the card goes online, the card assigns the location of the received data for this message. The card 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 card receives response data for this message, it writes the actual received data length in lay2mRxLen.

Message Update Time

To make the message periodic, set the `LAY2M_CTL_PERIODIC` bit in the lay2mCntCfg register and set the message update time by writing to lay2mUpdTime. The update time is the time the card waits after completing one message until it starts another message. The update time is measured in units of 1 μs. The value can range from 1 to 16380. The maximum update time is 16.38 seconds

.

> **Note**
>
> The update time does not include the time taken to send the message. For example, to send messages of the maximum length of 244 bytes with a reply length that is also the maximum length of 244 bytes at 500 kbaud takes about 11 μs. If you set the update time to 1 μs, the card will send the message and wait 1 μs before it sends the message again, so the time from the start of one message to the start of the next will be 12 μs.

Message Error Retry Time

To make the card retry failed messages, set the
`LAY2M_CTL_RETRY_PERIODIC` bit in the lay2mCntCfg register and set
the retry time for periodic messages by writing to lay2mErrTime. 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 you do not specify a retry time and there is an error with a periodic or
cyclic message, the message stops until you restart it.

Initiating Messages

The host controls message transmission through the trigger queue,
vmeTrigQueue[256], in the VMEPROFI structure. The host accesses the
trigger queue by means of the trigger queue head pointer, vmeTrgHead, in the
VMEPROFI structure. To insert triggers, the host puts the appropriate trigger
value in the trigger queue at the head position, then increments the head
pointer.

The card writes to the queue tail pointer, vmeTrgTail. 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.

The card must be online to send messages. Only an active station can send
messages.

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

The host sets lay2mState to `LAY2M_STE_IN_CONFIGURE` to indicate that host is configuring the block. The card will ignore any block in this state. In multitasking environments, the host can use this state to "hold" the message block.

The host sets lay2mState to `LAY2M_STE_ENABLE` to tell the card to send the message.

The card sets lay2mState to `LAY2M_STE_ACTIVE` while processing the message.

The card sets lay2mState to `LAY2M_STE_DONE` to tell the host that 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 card 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, for the message block is set and that the message update time has been set.

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 pointed to by the head pointer. The value the host writes is formed by ORing 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.

Stopping Periodic Messages

To stop a periodic message, the host simply sets the `LAY2M_CTL_PERIODIC` bit in the lay2mCntCfg register in the message control block to 0. When the message completes, the card will not resend it until you put it back in the trigger queue.

104

Sending Broadcast/Multicast Messages

Broadcast and multicast messages are sent to destination station 127. Broadcast messages are sent to destination SAP 63. The frame control must be SDNl or SDNh as all broadcast/multicast messages are unconfirmed.

In order for the destination station to receive a broadcast or multicast message, the station must have strict station checking disabled. To receive a broadcast message, the station must have a SAP configured on SAP 63.

Monitoring Message Status

The card maintains several registers to show the message status.

Status Register

The card sets bits in the status register, lay2mStatus, to show whether the message block is operating correctly. If bit 7, LAY2M_STS_OK, is set to 1, there are no problems with this message block. If bit 7 is 0, there has been an error.

Error Register

The card writes values to the error register, lay2mError, to indicate various errors in operation.

Message	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.

105

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	Meaning
0h	OK
1h	User error, SAP locked, destination did not take the message because the SAP was locked
2h	No resource for send data, SAP could not accept the message
3h	No service available (SAP does not exist)
4h	Access point blocked
80h	Short character, problems with wiring, termination, etc.
9Fh	No access, destination station was not there
AhF	Double token detected, problems with wiring, termination, etc.
BFh	Response buffer too small
8Fh	Noise at SM command, problems with wiring, termination, etc.

Message Events

These events occur when you are using the card 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.

The card sets bits in the event register, lay2mEvent, to notify the host that various events related to this message block have occurred.

The card sets bit 0, `LAY2M_EVT_INDICATION`, to notify the host that this message has been updated.

The card 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. Since only one of these options can be enabled, you will never get more than one of these events. If the host does not clear lay2mEvent, the card will not generate any further received data change or message confirmation events and will instead increment the overrun error counter.

3.10.6 Sample Programs

l2sapcfg.c shows how to configure FDL SAP control blocks.

l2msgcfg.c shows how to configure FDL messages.

l2mon.c shows how to access FDL data.

pbnetcfg.c shows how to configure network parameters.

vmecmd.c shows how to issue commands to the card.

107

3.11 Diagnostic Counters

The card 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 vmeInitCtrs to 1. The card then clears the counters to 0 and clears vmeInitCtrs to 0 to indicate that the counters have been cleared.

In the following sections, counters whose names begin with "diag" roll over to zero when they reach their maximum value. Counters with names beginning with "err" hold their maximum value.

3.11.1 General Statistics

These counters relate to the overall operation of the card on ProfiBus.

The errLanOffline counter increments when the LAN encountered errors and went offline.

If the OPTION_STAY_OFF_ERR bit in pfbOptions is 0, the card increments the errLanOffline error counter and goes offline, then goes back online immediately. If the bit is 1, the card goes offline with a fatal error and you must reload the card before putting it back online.

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

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

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, FDL messages and FMS messages.

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

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

3.11.2 Master Block Statistics

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

The diagMasterUpdate counter is the number of Master I/O update cycles completed, the number of complete I/O scans completed by the master.

The errMasErr counter is the number of DP master to DP slave communication errors. It increments any time a message failed because of retries exceeded, etc.

The errReConfig counter is the number of times a DP slave went offline and had to be reconfigured, the master was actively updating a slave and got a faulty message. The card increments this counter after it has retried the message the number of times specified in vmeMsgRetryLimit.

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

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

3.11.3 DP Slave Statistics

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

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

The card increments the errSlvErr counter when there are errors while the master is parameterizing the slave.

The card increments the errSlvTout counter when the slave has not received a message from the master within the master timeout period (see section 3.8.7, *Master Parameter Data*).

109

3.11.4 FDL (Layer 2) Statistics

Message Block Statistics

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

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

SAP Statistics

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

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

The card increments errLay2SapTout when you have set a timeout on a SAP and the timer times out.

3.11.5 ASPC2 ProfiBus Controller Statistics

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

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

The errFifo counter counts FIFO overflow errors. These errors occur when the LAN controller could not write to memory fast enough. This error is an internal card error and should never occur.

The errRxOverun counter counts receive overrun errors. This error is an internal card error and should never occur.

The errDblTok counter counts double token errors. These errors may occur when more than one node thinks it has the token or they may occur due to wiring errors, duplicate nodes, etc. The card withdraws to the "not hold token" state (decides it does not have the token) and waits until it gets the token passed to it again.

The errRespErr counter counts response errors, when a message failed or there was no response from the destination. This error may be due to bad hardware or faulty wiring.

110

If the `OPTION_STAY_OFF_ERR` bit in vmeOptions is 0, the card increments the errRespErr error counter and goes offline, then goes back online immediately. If the option bit is 1, the card goes offline with a fatal error and you must reload the card before putting it back online.

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

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

The errHsa counter occurs when a station with a station number higher than the high station address set on the card was heard. If the card is going online when this happens, it increments the counter and stays offline.

If this error happens when the card is online and the `OPTION_STAY_OFF_ERR` bit in vmeOptions is 0, the card increments the errHsa error counter and goes offline, then goes back online immediately. If the option bit is 1, the card goes offline with a fatal error and you must reload the card before putting it back online.

The errStn counter increments when a duplicate station is detected. If this error occurs when the card is going online, the card increments the errStn counter and stays offline.

If the error occurs when the card is online and the `OPTION_STAY_OFF_ERR` bit in vmeOptions is 0, the card increments the errStn error counter and goes offline, then goes back online immediately. If the bit is 1, the card goes offline with a fatal error and you must reload the card before putting it back online.

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

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

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

111

3.11.6 Event Statistics

The card increments errEventOverun when a new event occurred before the last one was processed. Refer to section 3.14, *Events and Interrupts*, for information on event processing.

3.12 Active Station List

The card maintains a list of active stations on the network in the vmeActStnList[128] table, one byte per station. The first byte corresponds to station 0.

The active station list is valid only if the card is an active station, not a passive station.

The card sets bit 2, `LAS_ACTIVE` bit, in the byte for a station if the station is active.

The card sets bit 3, `LAS_CHANGED`, when the status of the station changes, that is, when the station goes online or offline.

To clear the changed bits for all stations, set bit 0 in vmeAckLasChnge to 1. The card indicates that it has cleared the changed bits for all the stations by clearing vmeAckLasChnge to 0.

The card can determine which passive stations are present on the network. To do this, set bit 0 in the vmeUpdPasv register to 1. The card then sends an FDL status request to any stations not already in the active station list. If the station replies, the card sets the `LAS_PASSIVE` bit (bit 0) for that station in the active station list. When the card has tried all missing stations, it clears pfbUpdPasv to tell the host that it is finished scanning all stations.

If you update passive stations, the errNotOk diagnostic counter increments once for each station that does not reply.

The card also updates the changed bit for passive stations when you update the passive stations in the list.

113

3.12.1 Active Station List Events

The card can generate an event in the event queue and optionally an interrupt when the active station list changes.

To enable active station list events, set the `EVT_ENA_LAS_CHANGE` bit in the vmeEvtEna register.

Possible events include:

Event	Value
EVT_LAS_ACT_STN_ON	20NNh
EVT_LAS_ACT_STN_OFF	21NNh
EVT_LAS_PSV_STN_ON	22NNh
EVT_LAS_PSV_STN_OFF	23NNh

The low byte of the event contains the station number.

The card generates new events for active stations even if you do not clear the changed bit for the station. However, the card does not generate new events for passive stations if the changed bit is already set.

See section 3.14, *Events and Interrupts*, for more information on processing events and interrupts.

3.13 Putting the Card Online

Once the host has finished configuring the various card operations (DP master, DP slave, FDL messages, FDL SAPs), it can put the card online.

Issue the `CMD_GO_ON` command to the command register and wait up to 7 seconds for the command to execute.

When the card executes the command successfully, it sets the command register to the `CMD_ON` state. If the command fails, the card sets the command register to `CMD_ERROR`. If the command fails, check the status register for the cause of the failure.

When the card goes online, it checks the configurations for all the operations (DP master, DP slave) that have been configured. Before proceeding, check the various error and status registers for these operations to make sure that there are no problems.

115

3.14 Events and Interrupts

The various card operations can notify you of when various events take place. For example, if using the card as a DP master, configure the slaves to generate an event when they receive data changes.

The card indicates that an event has occurred by writing to the event queue, vmeEventQueue[256], in the *profi_usr* structure.

Your application then extracts events from the queue and processes them. You can design your application to be event driven. Refer to section 3.1, *VMEPROFI Module Overview* which describes the various card functions.

Each event in the event queue is a 16-bit integer. The high 4 bits, bits 12-15, indicate the event class (DP slave, DP master, etc.) 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 the event.

Event Class	Definition	Value
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

ProfiBus Events

These relate to the overall operation of the card on the ProfiBus network. To enable ProfiBus events, set the `EVT_ENA_BUS_ERROR` bit in the pfbEvtEna register.

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

116

Event	Value
EVT_PFB_FAT_RUN_ERR	0100h
EVT_PFB_OFFLINE_STAY	0200h
EVT_PFB_FATAL_INTERNAL	0300h

LAN Events

These relate to the operation of the ASPC2 LAN controller. Refer to section 3.6, *Network Parameters*, for more information about the causes of these events. LAN events cannot be disabled. The table shows possible LAN events. The low byte is always 0.

Event	Value
EVT_LAN_INV_REQ_LEN	1000h
EVT_LAN_FIFO_ERR	1100h
EVT_LAN_RX_OVERUN	1200h
EVT_LAN_DBL_TOK	1300h
EVT_LAN_RSP_ERR	1400h
EVT_LAN_SYNI_ERR	1500h
EVT_LAN_NET_TOUT	1600h
EVT_LAN_BAD_HSA	1700h
EVT_LAN_BAD_STN	1800h
EVT_LAN_TOK_PASS_ERR	1900h
EVT_LAN_LAS_BAD	1A00h
EVT_LAN_OFFLINE	1B00h

Active Station List Events

These occur when there are changes in the active station list. Refer to section 3.12.1, *Active Station List Events*, for information on what causes these events.

117

To enable active station list events, you must set the
`EVT_ENA_LAS_CHANGE` bit in the vmeEvtEna register.

The table shows possible active station list events. The low byte contains the
station that changed.

Event	Value
EVT_LAS_ACT_STN_ON	20NNh
EVT_LAS_ACT_STN_OFF	21NNh
EVT_LAS_PSV_STN_ON	22NNh
EVT_LAS_PSV_STN_OFF	23NNh

DP Master Events

These events occur when you are using the card as a DP master and you have
enabled these events. Refer to section 3.7.13, *DP Master Events*, for
information on what these events mean and how to enable/disable these
events.

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 the low byte is zero.

Event	Value
EVT_MAS_RX_DATA_CHG	30NNh
EVT_MAS_UPDTE	31NNh
EVT_MAS_SCAN_DONE	3200h
EVT_MAS_OK	33NNh
EVT_MAS_ERROR	34NNh

DP Slave Events

These events occur when you are using the card as a DP slave and you have
enabled these events. Refer to section 3.8.13, *Slave Events*, for information
on what these events mean and how to enable or disable these events.

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

118

Event	Value
EVT_SLV_RX_DATA_CHG	4000h
EVT_SLV_UPDTE	4100h
EVT_SLV_OK	4200h
EVT_SLV_ERROR	4300h
EVT_SLV_CHG_TO_RUN	4400h
EVT_SLV_CHG_TO_STOP	4500h

Layer 2 Message Events

These events occur when you are using the card to send FDL (layer 2) messages. The low byte contains the message block number.

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

119

Layer 2 SAP Events

These events occur when you have configured FDL (layer 2) SAPs on the card. The low byte contains the SAP number.

Event	Value
EVT_LY2S_RX_DATA_CHG	60NNh
EVT_LY2S_UPDATE	61NNh
EVT_LY2S_ERROR	62NNh

Event Queue Events

Event	Value
EVT_QUE_OVERUN	000Fh
EVT_QUE_BAD_TRIG	100Fh

The EVT_QUE_OVERUN event occurs if the event queue fills up. If it occurs, the EVT_QUE_OVERUN event replaces the last event in the queue. This event occurs when the host cannot process events quickly enough.

The EVT_QUE_BAD_TRIG event occurs if you enter an invalid entry in the trigger queue.

Event queue events cannot be disabled.

3.14.1 Accessing the Event Queue

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

Whenever the card adds an event to the queue, it increments the head pointer, vmeEvtHead.

After the host removes an event from the queue, it increments the tail pointer, vmeEvtTail. 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 more specifically 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 card will not enter the new event in the queue. Instead, it increments the errEventOverun counter.

3.14.2 Using Interrupts

The VMEPROFI module can generate host interrupts on the same conditions that generate events.

You must be familiar with the host interrupt system and how to set them up and process them on you system.

You can selectively enable interrupts for any or all of the card funcions by setting bits in the pfbIntEna register. Enable the event classes in the VMEPROFI module that you want to generate interrupts.

> • To generate interrupts on active station list events, set the INT_ENA_LAS_CHANGE bit.
>
> • To enable intrrupts on ProfiBus error events, set the INT_ENA_BUS_ERROR.
>
> • To enable interrupts on DP master events, set the INT_ENA_DP_MAS_EVENT bit. You can tell the card to hold the interrupt until the end of the scan.
>
> • To enable interrupts on DP slave events, set the INT_ENA_DP_SLV_EVENT bit.

121

• To enable interrupts on FDL (layer 2) message events, set the INT_ENA_LY2M_EVENT bit.

• To enable interrupts on FDL (layer 2) SAP events, set the INT_ENA_LY2S_EVENT bit.

3.14.3 Sample Programs

The sample programs are written under DOS for a paged mode ISA card.

vmeevent.c shows how to process events in both polled and interrupt modes. The details of the interupt handling are specific to the ISA versions of the card.

122

3.15 Station IDs

The card can send a request Ident with reply message to read the identification of a remote station. The card can also respond to such a request from another station.

To read the ID of a remote station, write the remote station number to vmeIdStn, then set bit 0 in vmeIdReq to 1 and wait for the card to clear the bit. If there has been an error in reading the station ID, the card reports the error in vmeIdRspSts. Otherwise, the card writes the total length of the returned data in vmeIdRspLen, the lengths of each of the four fields returned in the array vmeIdFldLen[4] and the text of the response in vmeIdText[242]. Use the lengths to decode the text into the four fields.

To set the text to be returned when a remote station requests the card's ID, write the string to the array vmeLocIdUsrStr[112]. By default, if you do not enter any text in this array, the card returns

```
Copyright (c) 1995-1996 S-S Technologies Inc.

5136-PFB-VME

Ver x.xx
```

in response to an ID request, where x.xx is the version number of the VMEPROFI module. Write this text before you put the card online.

123

3.16 Using the Host Watchdog

Use the host watchdog feature to ensure that the card takes itself offline if the host application fails.

When enabling the host watchdog, the host must check in with the card within the time set as the watchdog period. Otherwise the watchdog times out and the card takes itself offline.

To enable the host watchdog, write a non-zero value to the vmeWdTime register in the VME_PROFI structure. This value sets the host watchdog time in milliseconds. The watchdog time can range from 1 ms to 65.535 seconds. Write the value as a word, not as two bytes.

The host checks in with the watchdog by resetting vmeWdKick to 0. Every millisecond, the card checks to see if vmeWdKick is 0. If it is, the card restarts its internal watchdog timer and sets vmeWdKick to 1. If it is not 0, the card increments the internal watchdog timer. If the value in the card's internal watchdog timer ever exceeds vmeWdTime, the watchdog bites and the card takes itself offline. Reload or rerun the VMEPROFI module before putting the card back online.

> **Note**
>
> Once you enable the host watchdog, you cannot disable it.

124

3.17 VMEPROFI LED Usage

The communication status (lower) LED shows the health of the network. It is green when the card has the token. It is never green for a passive station. It is red whenever there is a network error (token pass failure, communication failure, etc.) If there is an error, it is on for a minimum of 1 second.

The system status (upper) LED on the bracket of the 5136-PFB- shows the current state of the various operations configured on the 5136-PFB-. The LED flashes sequentially the state of the DP master, DP slave, Layer 2 messages, layer 2 SAPs, and FMS. Only those operations configured on the card are shown. You have 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 VMEPROFI operations and green if the operation is OK. For DP master, amber means all slaves are OK but we are scanning in program mode. For DP slave, amber means that the slave is being scanned by a master in program mode.

The card also uses the LEDs to signal internal errors. If an internal error occurs, the card 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 the bit is zero, green indicates the bit is 1. Then the cycle repeats. Record the sequence before calling for technical support.

125

126

4

Memory Locations and Constants

This section provides tables which show:

- PFBPROFI control structure
- FDL SAP control blocks
- FDL message control blocks
- DP master control blocks

4.1 Summary of Memory Locations and Constants

4.1.1 PFBPROFI Control Structure

5136-PFB-VME Basic Control/Status Registers

Size	Name	Offset	Description
BYTE	pfbCommand	00h	Card command register
BYTE	pfbStatus	01h	Command status register

Constants related to pfbCommand

Name in profictl.h	Value	Description
CMD_OFF	E0h	PFB - card is offline ready to take commands
CMD_ON	E1h	PFB - card is online ready to take offline command
CMD_COM_CFG	E2h	PFB - card is being configured from serial port
CMD_ERROR	Efh	PFB - card is in error, PFB status contains error code
CMD_GO_ON	01h	HOST - card should now go online
CMD_GO_OFF	02h	HOST - card should now go offline, or abort config
CMD_REINIT	03h	HOST - card should reinitialize memory and all parameters
CMD_CLR_CFG_BUF	04h	HOST - card should clear pfbBinCfgPage
CMD_CHK_NET_CFG	05h	HOST - card should check network parameters, and assign defaults
CMD_CPY_MAS_CFG	06h	HOST - card should copy a page of cfg from pfbBinCfgPage

128

Name in profictl.h	Value	Description
CMD_AUTO_BAUD_DET	07h	HOST - card should do an automatic baud detect
CMD_MAS_ASIGN_ADDR	08h	HOST - assign and fill in IO data addresses (page/offset) for DP master
CMD_AUTODP_CFG	09h	HOST - configure DP master using online auto-configuration functionality
CMD_CFG_ABF_SHRAM	0Fh	HOST - configure DP master from binary file, copied to the beginning of the shared memory
CMD_CFG_2BF_SHRAM	10h	HOST - configure DP master from COM PROFIBUS binary file, copied to the beginning of the shared memory
CMD_CFG_FROM_FLASH	11h	HOST - configure card from configuration stored in flash
CMD_CFG_FLASH_GO_ON	14h	HOST - configure card from flash then go online
CMD_PGM_TO_FLASH	21h	HOST - burn card configuration into flash
CMD_PGM_SID_FLASH	22h	HOST - burn special ID (SST only interface)
CMD_GET_SID_FLASH	23h	HOST - get special ID (SST only interface)
CMD_ERR_ACK	FFh	HOST - acknowledge offline error (online errors cannot be acked)

129

Constants related to pfbStatus

Name in profictl.h	Value	Description
STS_NO_ERROR	00h	Command finished without error
STS_BAD_CMD	01h	Unrecognized command
STS_BAD_BAUD	02h	Unsupported baud rate
STS_BAD_STN_ADR	03h	Unsupported station address
STS_BAD_HI_STN_ADR	04h	HSA is not valid
STS_BAD_TOK_ROT	05h	Token rotation time out of range
STS_BAD_SLOT_TME	06h	Slot time out of range
STS_BAD_IDLE_1	07h	Idle time 1 out of range
STS_BAD_IDLE_2	08h	Idle time 2 out of range
STS_BAD_RDY_TME	09h	Ready time out of range
STS_BAD_QUI_TME	0Ah	Quiet time out of range
STS_BAD_GAP_UPD	0Bh	Gap update time out of range
STS_BAD_TOK_RETRY	0Ch	Token retry limit out of range
STS_BAD_MSG_RETRY	0Dh	Message retry limit out of range
STS_BAD_TOK_ERR_LIM	0Eh	Token error limit out of range
STS_BAD_RSP_ERR_LIM	0Fh	Bad response limit out of range
STS_BAUD_DET_ERROR	10h	Unable to detect baud rate (no activity detected)
STS_CFG_BAD_CHK_PATTERN	20h	Binary configuration file corrupted
STS_CFG_BIN_TOO_SHORT	21h	Binary configuration file corrupted
STS_CFG_BIN_TOO_LONG	22h	Binary configuration file corrupted
STS_CFG_BAD_CHKSUM	23h	Binary configuration file corrupted
STS_CFG_INVALID_CPU_HDR	24h	Binary configuration file corrupted
STS_CFG_INVALID_SLV_REC_TYP	25h	Binary configuration file corrupted
STS_CFG_RX_OVERFLOW	26h	Too much Master RX data has been configured (16KB max)
STS_CFG_TX_OVERFLOW	27h	Too much Master TX data has been configured (16KB max)

130

Name in profictl.h	Value	Description
STS_CFG_DESIG_NAME_TOO_LONG	28h	Nm= parameter, designation name too long (12 chars max) using COM PROFIBUS
STS_CFG_DESIG_BAD_ARG	29h	Unrecognized argument (Nm=, Tx=, Rx=, Ch) using COM PROFIBUS
STS_CFG_INV_RX_OFS	2Ah	RX= parameter, invalid offset (0000-3FF8h) using COM PROFIBUS
STS_CFG_INV_TX_OFS	2Bh	TX= parameter, invalid offset (0000-3FF8h) using COM PROFIBUS
STS_CFG_DESIG_OFS_NOT_SPEC	2Ch	RX and TX offsets have been specified for one, but not all, slaves
STS_CFG_RX_OVERLAP	2Dh	RX data of one block overlaps another
STS_CFG_TX_OVERLAP	2Eh	TX data of one block overlaps another
STS_CFG_INV_LEN	2Fh	Invalid parameter or check data length
STS_CFG_NO_CONFIG	30h	No configuration present to program into flash
STS_FLASH_BAD_ID	31h	Internal flash error, bad ID
STS_FLASH_ERASE_ERR	32h	Internal flash error, erase
STS_FLASH_PROG_ERR	33h	Internal flash error, program
STS_FLASH_VRFY_ERR	34h	Internal flash error, verification
STS_CFG_MAS_EXT_ALLOC_ERR	35h	Out of master extension memory
STS_LAY2M_INV_MAX_LEN	36h	Max length is greater than 244
STS_CFG_ADDR_OUT_OF_RANGE	37h	Address out of range
STS_CFG_COPY_TABLE_OVERRUN	38h	Configuration buffer is full
STS_CFG_AUTOCFG_DONE	39h	The DP master auto-configuration command succeeded
STS_ACFG_STATE_ERR	40h	The DP master auto-configuration command failed

Name in profictl.h	Value	Description
STS_CFG_INTERNAL_ERROR	80h	Internal card error - see errInternal and err Arg registers and call SST support
STS_OUT_OF_APBS	81h	Out of application blocks
STS_HOST_WD_BITE	82h	Host did not kick PFB watchdog within watchdog period
STS_HEAP_ALLOC_FAIL	83h	Internal error
STS_SH_HEAP_ALLOC_FAIL	84h	Internal error
STS_NET_ERROR	90h	Network error, card taken offline due to OPTION_STAT_OFF_ERR set

Card and Module Identification Registers

Size	Name in profictl.h	Offset	Description
WORD	pfbCardId	02h	5136-PFB-VME Card ID (AAD0h)
WORD	pfbModId	04h	PFBPROFI Module ID (BB01h)
WORD	pfbModVer	06h	PFBPROFI Module Version (ex. 0102h = 1.02)

Constants related to pfbCardId, pfbModId

Name in profictl.h	Value	Description
CRD_5136_PFB	AAD0h	Applications can use this to verify that 5136-PFB-VME card is present
MOD_PROFI	BB01h	Applications can use this to verify that PFBPROFI module is present

Host Event and Interrupt ControL Registers

Size	Name	Offset	Description
WORD	pfbEvtEna	08h	Event enable mask
WORD	pfbIntEna	0Ah	Event interrupt enable mask
BYTE	pfbEvtHead	0Ch	Event queue head pointer (changed by host)
BYTE	pfbEvtTail	0Dh	Event queue tail pointer (changed by card)

132

Constants related to pfbEvtEna

Name in profictl.h	Value	Description
EVT_ENA_LAS_CHANGE	0001h	Active station list change
EVT_ENA_BUS_ERROR	0002h	PROFIBUS error

Constants related to pfbIntEna

Name in profictl.h	Value	Description
INT_ENA_LAS_CHANGE	0001h	Active station list change
INT_ENA_BUS_ERROR	0002h	PROFIBUS error
INT_ENA_DP_MAS_EVENT	0004h	DP master event has occurred
INT_ENA_DP_SLV_EVENT	0008h	DP slave event has occurred
INT_ENA_LY2M_EVENT	0010h	Layer 2 message event has occurred
INT_ENA_LY2S_EVENT	0020h	Layer 2 SAP event has occurred
INT_ENA_FMS_EVENT	0040h	FMS event has occurred
INT_ENA_MPI_EVENT	0080h	MPI event has occurred

ASPC2 ProfiBus Controller Basic Parameters

Size	Name	Offset	Description
BYTE	pfbStnAddr	0Eh	PROFIBUS local station address
BYTE	pfbHiStnAddr	0Fh	Highest active station address on network
BYTE	pfbActive	10h	1=active station, 0=passive station
BYTE	pfbBaud	11h	Network baud rate
WORD	pfbOptions	12h	Network options

133

Constants related to pfbBaud

name in profictl.h	Value
BAUD_9K6	0h
BAUD_19K2	1h
BAUD_93K75	2h
BAUD_187K5	3h
BAUD_500K	4h
BAUD_750K	5h
BAUD_1M5	6h
BAUD_3M	7h
BAUD_6M	8h
BAUD_12M	9h

Constants related to pfbOptions

name in profictl.h	Value	Description
OPTION_REPEATER	0001h	1=repeater on network, 0=no repeater on network
OPTION_FMS	0002h	1=fms devices on network, 0=dp only on network
OPTION_STAY_OFF_ERR	0004h	Stay offline if token error limit or msg error limit exceeded

134

ASPC2 ProfiBus Controller Bus Parameters

Size	Name	Offset	Description
DWORD	pfbTokRotTime	14h	Target token rotation time
WORD	pfbSlotTime	18h	Slot time
WORD	pfbIdleTime1	1Ah	Idle time 1
WORD	pfbIdleTime2	1Ch	Idle time 2
WORD	pfbReadyTime	1Eh	Ready time (MinTsr)
BYTE	pfbGapUpdFact	20h	Gap update factor
BYTE	pfbQuiTime	21h	Quiet Time

ASPC2 ProfiBus Controller Error Handling Parameters

Size	Name	Offset	Description
BYTE	pfbTokRetryLimit	22h	Token retry limit
BYTE	pfbMsgRetryLimit	23h	Message retry limit
BYTE	pfbTokErrLimit	24h	Token error limit
BYTE	pfbRespErrLimit	25h	Response error limit

Host Trigger Queue Control

Size	Name	Offset	Description
BYTE	pfbTrgHead	26h	pfb function trigger event head pointer (changed by host)
BYTE	pfbTrgTail	27h	pfb function trigger event tail pointer (changed by card)

135

DP Master Global Status and Control Table

Size	Name	Offset	Description
BYTE	pfbMasCntrlCfg	28h	Global control for all master blocks
BYTE	pfbMasSts	29h	Global status for all master blocks
BYTE	pfbMasCntrlPage	2Ah	Memory page which contains master block control table
BYTE	pfbMasRxPage	2Bh	Memory page which contains master RX data (from slaves)
BYTE	pfbMasTxPage	2Ch	Memory page which contains master TX data (to slaves)
BYTE	pfbMasCoherFlags	2Dh	Master TX and RX coherent data flags
WORD	pfbMasMaxIoCycTme	2Eh	Maximum I/O cycle time in 10µs increments (0.01s-655.35s)
WORD	pfbMasMinIoCycTme	106h	Minimum I/O cycle time in 100µs increments (200µs-6,553.5µs)

136

Constants related to pfbMasCoherFlags

Name in profictl.h	Value	Description
PFB_MAS_TXPG_UPDATE	01h	The TX data for at least one block is ready to send
PFB_MAS_RXPG_LOCK	02h	The host is currently reading at least one block's TX data
PFB_MAS_COHER_ENA	80h	Enable DP master data coherency

Constants related to pfbMasCntrlCfg

Name in profictl.h	Value	Description
PFB_MAS_CTRL_SYNC_SCAN	01h	Enable synchronized scan
PFB_MAS_CTRL_RUN_MODE	02h	Scan I/O in RUN mode
PFB_MAS_CTRL_USR_OFS	04h	User defined rx and tx data offsets (no pfb auto assign)
PFB_MAS_CTRL_ENABLE	08h	Enable master mode on PFB card
PFB_MAS_CTRL_DIS_LED	10h	Disable status led for dp master function
PFB_MAS_CTRL_HOLD_INTR	20h	Hold dp master event interrupt(s) until end of scan
PFB_MAS_CTRL_EVT_SCAN_DONE	40h	generate event (intr) at end of scan
PFB_MAS_CTRL_ADDR_ASIGNED	80h	DP master data addresses have been assigned

Constants related to pfbMasSts

Name in profictl.h	Value	Description
PFB_MAS_STS_ALL_OK	01h	All I/O configured for master to scan is being scanned ok

FMS global status and control table pointer

Size	Name	Offset	Description
BYTE	pfbFmsCntrlCfg	30h	FMS control and configuration
BYTE	pfbFmsSts	31h	Global status for all FMS blocks
BYTE	pfbFmsCRLPage	32h	FMS Comm Rel List (CRL) page

Diagnostic Counters and Control

Size	Name	Offset	Description
BYTE	pfbInitCtrs	40h	If non-zero PFB will initialize counters then set back to 0

Constants related to pfbFmsCntrlCfg

Name in profictl.h	Value	Description
PFB_FMS_CNTRL_ENABLE	80h	Enable FMS services on the cont

Constants related to pfbFmsSts

Name in profictl.h	Value	Description
PFB_FMS_STS_CRL_ALL_OK	01h	No errors on any CRL entries

MPI global status and control table

Size	Name	Offset	Description
BYTE	pfbMpiCntrlCfg	33h	MPI control and configuration
BYTE	pfbMpiSts	34h	Global status for all MPI blocks
BYTE	pfbMpiCrlPage	35h	MPI user interface page
BYTE	pfbMpiReqTime	36h	Last request execution time

Constants related to pfbMPICntrlCfg

Name in profictl.h	Value	Description
PFB_MPI_CNTRL_ENABLE	80h	Enable MPI services on the card

138

Constants related to pfbMPISts

Name in profictl.h	Value	Description
PFB_MPI_STS_ALL_OK	01h	No errors on any MPI blocks

DP master Sync and Freeze support

Size	Name	Offset	Description
WORD	pfbMasSyncFrzCmd	37h	Sync and Freeze command MSB = command, LSB = group select
BYTE	pfbMasScanEmpty	39h	Non-zero value = no slaves in data exchange mode

Extended date configuration mode

Size	Name	Offset	Description
BYTE	pfbExtDateCfg	3Ah	Sets extended date configuration mode
BYTE	pfbExtDatePage	3Bh	Page of the extended configuration date

Constants related to pfbMasSyncFrzCntrl

Name in profictl.h	Value	Description
PFB_SYNC_CMD	2000h	Enable Sync mode
PFB_UNSYNC_CMD	1000h	Disable Sync mode
PFB_FREEZE_CMD	800h	Enable Freeze mode
PFB_UNFREEZE_CMD	400h	Disable Freeze mode

Constants related to pfbExtDataCfg

Name in profictl.h	Value	Description
PFB_EXTDATA_PAGE_EN	80h	Allocates extended data to a separate page

139

General Statistics

Size	Name	Offset	Description
BYTE	errLanOffline	41h	LAN encountered errors and went into offline state
WORD	diagConf	42h	Total confirmations (requests from us) (MAS,LAY2,FMS)
WORD	diagInd	44h	Total indications (requests to us) (MAS,LAY2,FMS)
WORD	errNotOk	46h	Total not OK confirmations and/or indications (MAS,LAY2,FMS)
DWORD	diagTokHldTime	48h	Actual instantaneous token hold time in Tbit
DWORD	diagMinTokHldTime	4Ch	Minimum actual token hold time in Tbit

DP Master Block Statistics

Size	Name	Offset	Description
WORD	diagMasterUpdate	50h	DP Master I/O update cycles completed
BYTE	errMasErr	52h	DP Master to slave communication errors
BYTE	errReConfig	53h	DP Master went offline and had to be reconfigured
DWORD	diagMasScanTime	54h	Instantaneous master scan time in µs
DWORD	diagMasMaxScanTime	58h	Maximum master scan time in µs

140

DP Slave Statistics

Size	Name	Offset	Description
WORD	diagSlaveUpdate	5Ch	Slave updates
BYTE	errSlvErr	5Eh	Slave configuration failures
BYTE	errSlvTout	5Fh	Slave watchdog timeouts

Layer 2 Message Statistics

Size	Name	Offset	Description
WORD	diagLay2MsgOk	60h	Layer 2 messages sent ok
BYTE	errLay2MsgNotOk	62h	Layer 2 message errors
BYTE	errLay2MsgProcOvrn	63h	Layer 2 message overran

Layer 2 SAP Statistics

Size	Name	Offset	Description
WORD	diagLay2SapOk	64h	Layer 2 SAP requests processed OK
BYTE	errLay2SapNotOk	66h	Layer 2 SAP errors
BYTE	errLay2SapTout	67h	Layer 2 SAP update timeouts

141

ASPC2 ProfiBus Controller Statistics

Size	Name	Offset	Description
BYTE	errInvReqLen	70h	Invalid request length errors
BYTE	errFifo	71h	FIFO overflow errors
BYTE	errRxOverun	72h	Receive overrun errors
BYTE	errDblTok	73h	Double token errors (bad wiring or hardware)
BYTE	errRespErr	74h	Response errors (bad wiring or hardware)
BYTE	errSyniErr	75h	syni errors (bad wiring or hardware)
BYTE	errNetTout	76h	Network timeout errors
BYTE	errHsa	77h	Station higher than HSA was heard
BYTE	errStn	78h	Duplicate station detected
BYTE	errPasTok	79h	Unable to pass token (bad wiring or hardware)
BYTE	errLasBad	7Ah	Active station list invalid (bad wiring or hardware)

Miscellaneous Counters

Size	Name	Offset	Description
BYTE	errInternal	7Bh	When PFB status register =80h, this register provide detailed error information to SST Technical Support
BYTE	errArg	7Ch	When PFB status register =80h, this register provide detailed error information to SST Technical Support
BYTE	errEventOverun	7Dh	A new event occurred before the last one was cleared

142

Active Station List

Size	Name	Offset	Description
BYTE	pfbAckLasChnge	7Eh	Bit 0=1 to acknowledge station list change (card clears)
BYTE	pfbUpdPasv	7Fh	Bit 0=1 to update status of passive stations (card clears)
BYTE	pfbActStnList	80-FFh	Active Station list, 1 byte per station

Constants related to pfbActStnList

Name in profictl.h	Value	Description
LAS_PASSIVE	01h	This station is passive
LAS_ACTIVE	04h	This station is active
LAS_CHANGED	08h	Status of this station has changed

DP Master Configuration

Size	Name	Offset	Description
BYTE	pfbBinCfgPage	100h	Master binary configuration memory page (filled in by PFB)
BYTE	pfbBinCfgOfs	101h	Master binary config block offset (in 16K pages)
WORD	pfbBinCfgLen	102h	master binary configuration length of data in current page
WORD	pfbMc2PollTout	104h	poll timeout for reception of master class 2 commands

Host Watchdog

Size	Name	Offset	Description
WORD	pfbWdTime	108h	Host Watchdog Time in ms. Non-Zero to enable
WORD	pfbWdKick	10Ah	Host Watchdog Kick. Card writes 1, host writes 0 to kick watchdog

DP Master Global Event Register

Size	Name	Offset	Description
BYTE	pfbMasGlbEvt	10Ch	DP master global event

Constants related to pfbMasGlbEvt

Name in profictl.h	Value	Description
PFB_MAS_SCAN_DONE	01h	DP Master finished scanning all slaves

DP Master Blocks Configured

Size	Name	Offset	Description
BYTE	pfbMasNumBlks	10Dh	DP master - number of master blocks configured

Free Extension Memory

Size	Name	Offset	Description
WORD	pfbMasCntrlExtFree	10Eh	The free extended data area for storing extended DP parameters, configurations and diagnostic data on master configuration page

ID Text

Size	Name	Offset	Description
BYTE	pfbLocIdUsrStr	110h-17Fh	Text for the 4 fields to return in the ID response

144

DP Slave Controls

Size	Name	Offset	Description
WORD	slvCntCfg	180h	Control/config for DP slave function
BYTE	slvStatus	182h	Status for DP slave function (Host Read Only)
BYTE	slvError	183h	Error indication for DP slave function
BYTE	slvEvent	184h	Event Flags for DP slave function
BYTE	slvDiagEvent	185h	Diagnostic Event Flags for DP slave function
BYTE	slvRxDataLen	188h	Receive data length (data from master)
BYTE	slvReqRxDataLen	189h	Receive data length requested by master
BYTE	slvTxDataLen	18Ch	Transmit data length (data to master)
BYTE	slvReqTxDataLen	18Dh	Transmit data length requested by master
BYTE	slvChkLen	18Eh	Length of configuration check data
BYTE	slvParmLen	18Fh	Length of parameters from master
BYTE	slvGlbCntrl	190-191h	Global controls from master
BYTE	slvSts1	1A0h	Status byte 1 to master
BYTE	slvSts2	1A1h	Status byte 2 to master
BYTE	slvSts3	1A2h	Status byte 3 to master
BYTE	slvMasStn	1A3h	Station that configured Slave
BYTE	slvID_hi	1A4h	ID value to master hi byte (default 08h)
BYTE	slvID_lo	1A5h	ID value to master lo byte (default 55h)
BYTE	slvDiagLen	1A6h	Length of diagnostics to master
BYTE	slvDiag	1A7-1BFh	Diagnostics to master
BYTE	slvMasSts	1C0h	Status byte from master
BYTE	slvWdFact1	1C1h	Watchdog factor 1 from master
BYTE	slvWdFact2	1C2h	Watchdog factor 2 from master
BYTE	slvReadyTime	1C3h	Response delay time (tbit) from master
BYTE	slvMasID_hi	1C4h	ID value from master (hi byte)
BYTE	slvMasID_lo	1C5h	ID value from master (lo byte)
BYTE	slvGrpId	1C6h	Group ID value from master

145

Size	Name	Offset	Description
BYTE	slvParm	1C7h-1DF	Parameters from master
BYTE	slvChk	1E0h-1FF	Configuration Check values from master
BYTE	slvRxData	200-2FFh	Slave data received from master
BYTE	slvTxData	300-3FFh	Slave data to be sent to master

Constants related to slvCntCfg

Name in profictl.h	Value	Description
SLV_CTL_DIAG_UPD	0001h	Request diagnostic read from master
SLV_CTL_EVT_RX_CHG	0002h	Generate event (interrupt) when RX data to slave changes
SLV_CTL_FORCE_READY_TIME	0004h	Force response time (pfbReadyTime), and ignore what the master sends
SLV_CTL_IGN_SYNC_FRZ_ERR	0008h	Don't generate error if master request sync and/or freeze
SLV_CTL_RX_BYTE_SWAP	0010h	Swap upper and lower bytes of RX data
SLV_CTL_EVT_UPDTE	0020h	Generate event (interrupt) when slave is updated
SLV_CTL_EVT_MODE_CHANGE	0040h	Generate event (interrupt) when slave mode changes (master RUN/STOP)
SLV_CTL_IGNORE_STS	0080h	Ignore status of slave (no event)
SLV_CTL_DIS_LED	4000h	Disable status LED for DP slave function
SLV_CTL_ENABLE	8000h	Enable PFB to act as DP slave

146

Constants related to slvStatus

Name in profictl.h	Value	Description
SLV_STS_RUN_MODE	40h	DP slave is scanned in run mode
SLV_STS_OK	80h	Current slave status is OK

Constants related to slvError

Name in profictl.h	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	Master is requesting Freeze or Sync, which is not supported
SLV_ERR_RX_LEN_MISM	04h	Length of data from master to us is incorrect
SLV_ERR_TX_LEN_MISM	05h	Length of data from us to master is incorrect
SLV_ERR_WD_FACT_INV	06h	slvWdFact1 or slvWdFact2 from master was 0
SLV_ERR_TIME_OUT	07h	Slave watchdog time out (check response timeout)
SLV_ERR_WARN_WD_DIS	08h	Slave timeout watch dog disabled from master

Constants related to slvEvent

Name in profictl.h	Value	Description
SLV_EVT_UPD	01h	Master has updated us
SLV_EVT_RX_DATA_CHG	02h	RX data from master has changed

Constants related to slvDiagEvent

Name in profictl.h	Value	Description
SLV_DEVT_DIAG_UPD	01h	Master has read diagnostic info

147

Event and Trigger Queues

Size	Name	Offset	Description
WORD	pbfEventQueue	400-5FFh	Event queue
WORD	pfbTrigQueue	600-7FFh	Trigger queue

Constants related to pbfEventQueue

Name in profictl.h	Value
EVT_PFB	0h
EVT_PFB_FAT_RUN_ERR	0100h
EVT_PFB_OFFLINE_STAY	0200h
EVT_PFB_FATAL_INTERNAL	0300h
EVT_LAN	1h
EVT_LAN_INV_REQ_LEN	1000h
EVT_LAN_FIFO_ERR	1100h
EVT_LAN_RX_OVERRUN	1200h
EVT_LAN_DBL_TOK	1300h
EVT_LAN_RSP_ERR	1400h
EVT_LAN_SYNI_ERR	1500h
EVT_LAN_NET_TOUT	1600h
EVT_LAN_BAD_HSA	1700h
EVT_LAN_BAD_STN	1800h
EVT_LAN_TOK_PASS_ERR	1900h
EVT_LAN_LAS_BAD	1A00h
EVT_LAN_OFFLINE	1B00h
EVT_LAS	2h
EVT_LAS_ACT_STN_ON	2000h
EVT_LAS_ACT_STN_OFF	2100h
EVT_LAS_PSV_STN_ON	2200h
EVT_LAS_PSV_STN_OFF	2300h
EVT_MAS	3h
EVT_MAS_RX_DATA_CHG	3000h

148

Name in profictl.h	Value
EVT_MAS_UPDTE	3100h
EVT_MAS_SCAN_DONE	3200h
EVT_MAS_OK	3300h
EVT_MAS_ERROR	3400h
EVT_SLV	4h
EVT_SLV_RX_DATA_CHG	4000h
EVT_SLV_UPDTE	4100h
EVT_SLV_OK	4200h
EVT_SLV_ERROR	4300h
EVT_SLV_CHG_TO_RUN	4400h
EVT_SLV_CHG_TO_STOP	4500h
EVT_LY2M	5h
EVT_LY2M_RX_DATA_CHG	5000h
EVT_LY2M_CONFIRM	5100h
EVT_LY2M_ERROR	5200h
EVT_LY2M_BAD_MSG_NUM	5300h
EVT_MC2_CONFIRM	5400h
EVT_MC2_ERROR	5500h
EVT_LY2S	6h
EVT_LY2S_RX_DATA_CHG	6000h
EVT_LY2S_UPDATE	6100h
EVT_LY2S_ERROR	6200h
EVT_FMS	7h
EVT_FMS_CON_CHG	7000h
EVT_FMS_RX_DATA_CHG	7100h
EVT_FMS_UPDATE	7200h
EVT_FMS_ERROR	7300h
EVT_MPI	8h
EVT_MPI_CON_CHG	8000h
EVT_MPI_RX_DATA_CHG	8100h

149

Name in profictl.h	Value
EVT_MPI_UPDATE	8200h
EVT_MPI_ERROR	8300h
EVT_QUE	Fh
EVT_QUE_OVERUN	F000h
EVT_QUE_BAD_TRIG	F100h

Constants related to pfbTrigQueue

Name in profictl.h	Value
TRG_LAY2M	1000h
TRG_MAS_CLSS2	1100h
TRG_SAP_ACYC_ACK	1200h
TRG_SAP_UPDATE	1800h
TRG_FMS_CRL_OPEN	2000h
TRG_FMS_CRL_INIT	2100h
TRG_FMS_CRL_ABORT	2200h
TRG_FMS_PROC_REQ	3000h
TRG_MPI_PROC_REQ	4000h
TRG_DDSC_ENABLE	5000h
TRG_DDSC_DISABLE	5100h
TRG_DDSC_MAXDATA	5200h

Station ID

Size	Name	Offset	Description
BYTE	pfbIdReq	800h	Bit 0=1 to request ID for station in pfbIdStn (card clears)
BYTE	pfbIdStn	801h	Station to send ID request
BYTE	pfbIdRspSts	802h	If non-zero, error getting ID
BYTE	pfbIdRspLen	803h	Total length of ID response
BYTE	pfbIdFldLen	804-807h	Length of the 4 fields returned in the ID response
BYTE	pfbIdText	808-8f9h	Text for the 4 fields returned in the ID response

150

DP Master Cross-reference Table

Size	Name	Offset	Description
WORD	pfbMasStsTab	900-9FFh	Master status/xref table arranged by station number

Constants related to pfbMasStsTab

Name in profictl.h	Value	Description
MAS_STS_TAB_OK	8000h	Current status of slave OK. Low byte contains master control block number

151

DP Master Class 2

Size	Name	Offset	Description
BYTE	mc2CntCfg	A00h	Master Class 2 control and configuration
BYTE	mc2State	A01h	State of the Master Class 2
BYTE	mc2Status	A02h	Status of the Master Class 2
BYTE	mc2Error	A03h	Master Class 2 error
BYTE	mc2Event	A04h	Master Class 2 event
BYTE	mc2RspSts	A05h	Response status
BYTE	ms2TxLen	A06h	TX length
BYTE	mc2RxLen	A07h	RX length
WORD	mc2PollTime	A08h	Polling time in ms
WORD	mc2PollLimit	A0Ah	Polling limit
BYTE	mc2DstStn	A0Ch	Destination station

Constants related to mc2CntCfg

Name in profictl.h	Value	Description
MC2_CTL_UNCONF	01h	Unconfirmed messages
MC2_CTL_HI_PRI	04h	High priority messages
MC2_CTL_EVT_CONFIRM	20h	Generate event (interrupt) when message is confirmed

Constants related to mc2State

Name in profictl.h	Value	Description
MC2_STE_DISABLE	00h	Master Class 2 disabled
MC2_STE_BUSY	01h	Master Class 2 message is processing
MC2_STE_ENABLE	02h	Master Class 2 is enabled
MC2_STE_ACTIVE	03h	Master Class 2 is active
MC2_STE_DONE	FFh	Message is processed

152

Constants related to mc2Status

Name in profictl.h	Value	Description
MC2_STS_OK	80h	Master Class 2 status is OK

Constants related to mc2Error

Name in profictl.h	Value	Description
MC2_ERR_NOT_OK	01h	Master Class 2 is in error
MC2_ERR_RX_LEN	02h	Invalid RX length
MC2_ERR_TX_LEN	03h	Invalid TX length
MC2_ERR_POLL_TOUT	04h	Polling timed out

Constants related to mc2Event

Name in profictl.h	Value	Description
MC2_EVT_CONFIRM	01h	Event generated when confirmation received

FDL (Layer 2) Global Control and Status

Size	Name	Offset	Description
WORD	lay2Cntrl	FE0h	Global control for all Layer 2 SAP's and MSG's
WORD	lay2Status	FE2h	Global status for all Layer 2 SAP's and MSG's

Constants related to lay2Cntrl

Name in profictl.h	Value	Description
LAY2_CTL_SAP_DIS_LED	1000h	Disable status LED for layer 2 SAP function
LAY2_CTL_MSG_DIS_LED	2000h	Disable status LED for layer 2 message function
LAY2_CTL_SAP_ENABLE	4000h	Enable PFB Layer 2 SAP service
LAY2_CTL_MSG_ENABLE	8000h	Enable PFB Layer 2 Message service

153

Constants related to lay2Status

Name in profictl.h	Value	Description
LAY2_STS_ALL_SAP_OK	4000h	All configured layer 2 SAPs are OK
LAY2_STS_ALL_MSG_OK	8000h	All configured layer 2 messages are OK

FDL (Layer 2) Control Blocks

Offsets 1000h-1FFFh contain the layer 2 message control blocks, Lay2mCntrl. There are 128 blocks; each block occupies 32 bytes. The contents of each block are described in section 4.1.3, *FDL (Layer 2) Message Control Blocks*.

Offsets 2000h-27FFh contain the layer 2 SAP control blocks, Lay2sCntrl. There are 64 blocks; each block occupies 32 bytes. The contents of each block are described in section 4.1.2, *Layer 2 SAP Control Blocks*.

4.1.2 Layer 2 SAP Control Blocks

Size	Name	Offset	Description
BYTE	lay2sType	00h	Type of SAP (DP slave, Layer 2, or FMS)
BYTE	lay2sStn	01h	If strict station, accept updates only from this station
BYTE	lay2sSap	02h	If 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 strict FC, accept updates only 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 within mem page
WORD	lay2sTxDataOfs	14h	Transmit data (response) offset within mem page
BYTE	lay2sRxDataPage	16h	Receive data (from request) data mem page
BYTE	lay2sTxDataPage	17h	Transmit data (response) data mem page
BYTE	lay2sTxMaxLen	18h	Transmit data (response) length in bytes
BYTE	lay2sNumAcyBufs	19h	Number of acyclic buffers

155

Constants related to lay2sType

Name in profictl.h	Value	Description
LAY2S_TYP_NOT_DEF	00h	Type not defined
LAY2S_TYP_DP_SLAVE	01h	DP Slave SAP
LAY2S_TYP_LAYER2_SAP	02h	Layer 2 SAP
LAY2S_TYP_FMS	03h	FMS SAP
LAY2S_TYP_MPI_SAP	04h	MPI SAP
LAY2S_TYP_DP_MAC_MC2	05h	Master Class 2 SAP

Constants related to lay2sCntCfg

Name in profictl.h	Value	Description
LAY2S_CTL_RX_BYTE_SWAP	0004h	Swap hi and low bytes of RX data to this SAP
LAY2S_CTL_EVT_RX_CHG	0008h	Generate event (interrupt) when RX data to this SAP changes
LAY2S_CTL_IGNORE_STS	0010h	Ignore status of this SAP block (all_ok flag, LED and event)
LAY2S_CTL_EVT_UPDTE	0020h	Generate event (interrupt) when this SAP is updated (Indication)

Constants related to lay2sStatus

Name in profictl.h	Value	Description
LAY2S_STS_OK	80h	SAPs OK

Constants related to lay2sError

Name in profictl.h	Value	Description
LAY2S_ERR_NOT_OK	01h	Other error
LAY2S_ERR_TIME_OUT	02h	Time out
LAY2S_ERR_RX_LEN	03h	Invalid RX length
LAY2S_ERR_TX_LEN	04h	Invalid TX length
LAY2S_ERR_BAD_PARAM	05h	Bad parameter

Constants related to lay2sEvent

Name in profictl.h	Value	Description
LAY2S_EVT_UPDATE	01h	Update of date is done
LAY2S_EVT_RX_DATA_CHG	02h	RX data has been changed

Constants related to lay2sFrmCntrl

Name in profictl.h	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_HI	03h	Accept only SDN hi or lo priority
LAY2S_FC_SDA_LO	05h	Accept only SDA lo priority
LAY2S_FC_SDA_HI	06h	Accept only SDA hi priority
LAY2S_FC_SDA_LO_HI	07h	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	DDB file service request
LAY2S_FC_DDB_REQ_LO	0Dh	DDB file service low priority
LAY2S_FC_DDB_REQ_HI	0Eh	DDB file service high priority
LAY2S_FC_DDB_REQ_LO_HI	0Fh	DDB file servie low/high priority

4.1.3 FDL (Layer 2) Message Control Blocks

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	Event register
WORD	lay2mUpdTime	06h	Update interval for periodic in ms (1ms-16380ms)
WORD	lay2mErrTime	08h	Retry interval for periodic if error occurs * 1ms
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

158

Constants related to lay2mCntCfg

Name in profictl.h	Value	Description
LAY2M_CTL_PERIODIC	01h	Send message periodically
LAY2M_CTL_RETRY_PERIODIC	02h	Retry message periodically
LAY2M_CTL_HI_PRI	04h	Use high priority queue for messages
LAY2M_CTL_EVT_RX_CHG	08h	Generate event (interrupt) when RX data to this message changes
LAY2M_CTL_IGNORE_STS	10h	Ignore status of this message block (all_ok, LED, and event)
LAY2M_CTL_EVT_CONFIRM	20h	Generate event (intr) when this message is confirmed
LAY2M_CTL_RX_BYTE_SWAP	40h	Swap bytes of the RX data

Constants related to lay2mState

Name in profictl.h	Value	Description
LAY2M_STE_DISABLE	00h	Message is disabled
LAY2M_STE_IN_CONFIGURE	01h	Message is being configured
LAY2M_STE_ENABLE	02h	Message is enabled
LAY2M_STE_ACTIVE	03h	Message is inactive
LAY2M_STE_DONE	FFh	Message is processed

Constants related to lay2mStatus

Name in profictl.h	Value	Description
LAY2M_STS_OK	80h	Message is OK

Constants related to lay2mError

Name in profictl.h	Value	Description
LAY2M_ERR_NOT_OK	01h	Other error
LAY2M_ERR_RX_LEN	02h	Invalid RX length
LAY2M_ERR_TX_LEN	03h	Invalid TX length

Constants related to lay2mEvent

Name in profictl.h	Value	Description
LAY2M_EVT_CONFIRM	01h	Message has been confirmed
LAY2M_EVT_RX_DATA_CHG	02h	RX data has been changed

Constants related to lay2mFrmCntrl

Name in profictl.h	Value
FC_SDAl	03h
FC_SDAh	05h
FC_SDNl	04h
FC_SDNh	06h
FC_SRDl	0Ch
FC_SRDh	0Dh
FC_SmTime1	00h
FC_SmTime2	80h
FC_SmSDN	02h
FC_SmSRD	01h
FC_SmSRDSltDel	0Ah
FC_SmSRDSltKeep	0Bh
FC_DdbSRD	07h
FC_DiagSRD	08h
FC_ReqFDL	09h
FC_ReqId	0Eh
FC_ReqLSAPSts	0Fh

160

Constants related to lay2mRspStatus

Name in profictl.h	Value
RSP_STS_ACK	00h
RSP_STS_DL	08h
RSP_STS_DH	0Ah

4.1.4 DP Master Control Blocks

One page of card memory is reserved for DP master control blocks. Each block is used to configure one slave. The following table lists the elements of each block and the offsets within the block.

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
BYTE	masDiagEvent	05h	Diagnostic event
BYTE	masParmLen	06h	Parameters to slave length in bytes
BYTE	masChkLen	07h	Configuration check to slave length in bytes
WORD	masRxDataOfs	08h	RX data from slave offset within memory page
WORD	masTxDataOfs	0Ah	TX 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

161

Size	Name	Offset	Description
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
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

162

Constants related to masCntCfg

Name in profictl.h	Value	Description
MAS_CTL_IGNORE_STS	01h	Ignore status of this master block
MAS_CTL_EVT_RX_CHG	02h	Generate event (interrupt) when RX data from slave changes
MAS_CTL_RX_BYTE_SWAP	04h	Byte swap the RX data
MAS_CTL_EVT_UPDTE	08h	Generate event (interrupt) when slave update is complete
MAS_CTL_FAIL_SAFE	10h	Slave is a fail safe slave
MAS_CTL_DDSC_EN	20h	DDSC enable bit, must be set prior to setting the card online
MAS_CTL_TX_BYTE_SWAP	40h	Byte swap the TX data
MAS_CTL_ENABLE	80h	Enable this block

Constants related to masStatus

Name in profictl.h	Value	Description
MAS_STS_OK	80h	Current status of this master block is OK

Constants related to masError

Name in profictl.h	Value	Description
MAS_ERR_CFG_FAILURE	01h	Failure while trying to configure slave
MAS_ERR_SLV_ID_MISMATCH	02h	Slave 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 diags
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

163

Name in profictl.h	Value	Description
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
MAS_ERR_DDSC_RESRC_ALLOC	0Ch	Out of resources
MAS_ERR_DDSC_ENABLE	0Dh	General failure in the DDSC enable trigger

Constants related to masEvent

Name in profictl.h	Value	Description
MAS_EVT_UPD	01h	Slave has been updated
MAS_EVT_RX_DATA_CHG	02h	RX data has changed
MAS_EVT_DDSC_MAXSET	03h	The maximum size of 10 data parameters and diagnostics space is set

Constants related to masDiagEvent

Name in profictl.h	Value	Description
MAS_DEVT_DIAG_UPD	01h	Slave diagnostics have been updated

Constants related to masSiemType

Name in profictl.h	Value	Description
MAS_SIEM_TYP_OLD_DP	80h	Old Siemens style slave

A

Technical Data

Part number	5136-PFB-VME
Function	Interface card for ProfiBus DP, FMS and FDL (layer 2) networks
Size	IEEE 1014, 6U height, P1 compatible
Capabilities	memory SD16, SADO24
Addressing	(standard) 256 Kbytes on any even 256 Kbyte boundary or 16 Kbytes on any 16 Kbyte boundary (short) 8 bytes on any 1 Kbyte boundary
Interrupts	switch selected level 1-7 software set 8-bit status/ID release on acknowledge.
Current Consumed	maximum 950 mA at 5V. This must be from a supply delivering Separated Extra Low voltage.
Environmental Storage Temp	operating temperature 0-60 degrees Celsius -40 - +85 degrees Celcius
Card connectors	standard ProfiBus DB-9 connector Phoenix Combicon connector, part number MSTB 2.5/5-ST-5.08

166