

SST Profibus Scanner Module

DLL Reference Guide

Document Edition: 2.2.0.0

Document #: 717-0033

Document Edition: 2.2.0.0

Date: October 11, 2011

This document applies to the SST Profibus Scanner Module.

©2011 Molex Inc. All rights reserved.

This document and its contents are the proprietary and confidential property of Molex Inc. and/or its related companies and may not be used or disclosed to others without the express prior written consent of Molex Inc. and/or its related companies.

SST is a trademark of Molex Inc. All other trademarks belong to their respective companies.

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

Written and designed at:

**Molex Incorporated
216 Bathurst Drive
Waterloo, Ontario, Canada N2V 2L7**

Hardcopies are not controlled.

Contents

Contents	iii
Preface	ix
Purpose of this Guide	x
Conventions	x
Style.....	x
Special Terms	xi
Special Notation	xi
Introduction	13
1.1 PFBSCAN Overview.....	14
PFBMAN32 DLL API	17
2.1 Introduction	18
2.2 Overview	18
2.3 Services.....	18
2.4 Application Stack	19
2.5 Quick Start Using Sample Utilities	19
2.5.1 Configuring the DP Master	19
2.5.2 Configuring the DP Slave	20
2.5.3 PROFIBUS Command	21
2.5.4 DP Monitor.....	21
2.6 Operating the Application	22
2.7 Card Access Types	23
2.8 General Resource Access	25

2.8.1 PFB_CancelWaitIrq	25
2.8.2 PFB_CloseCard	26
2.8.3 PFB_DriverVersion	27
2.8.4 PFB_EnumDrivers	28
2.8.5 PFB_FreeCardPtr	29
2.8.6 PFB_FreeDriver	30
2.8.7 PFB_GetBusType	31
2.8.8 PFB_GetCardPtr	32
2.8.9 PFB_GetHomePage	33
2.8.10 PFB_GetMem	34
2.8.11 PFB_LoadDriver	35
2.8.12 PFB_OpenCard	36
2.8.13 PFB_PageReadBit	40
2.8.14 PFB_PageReadBlock	41
2.8.15 PFB_PageReadByte	42
2.8.16 PFB_PageReadWord	43
2.8.17 PFB_PageToggleBit	44
2.8.18 PFB_PageWriteBit	45
2.8.19 PFB_PageWriteBlock	46
2.8.20 PFB_PageWriteByte	47
2.8.21 PFB_PageWriteWord	48
2.8.22 PFB_PutMem	49
2.8.23 PFB_ReadBit	50
2.8.24 PFB_ReadBlock	51
2.8.25 PFB_ReadByte	52
2.8.26 PFB_ReadHC	53
2.8.27 PFB_ReadWord	57
2.8.28 PFB_SetAccessTimeout	58
2.8.29 PFB_ToggleBit	59
2.8.30 PFB_Version	60
2.8.31 PFB_WaitIrq	61
2.8.32 PFB_WriteBit	62
2.8.33 PFB_WriteBlock	63
2.8.34 PFB_WriteByte	64
2.8.35 PFB_WriteHC	65
2.8.36 PFB_WriteWord	69
2.9 Profibus Network	70
2.9.1 PFB_AckStnChange	70
2.9.2 PFB_Command	71
2.9.3 PFB_GetCardStatus	73
2.9.4 PFB_GetCountersSts	74
2.9.5 PFB_GetDiagCounters	75
2.9.6 PFB_GetEvent	76
2.9.7 PFB_GetL2GlbCntrl	77

2.9.8 PFB_GetL2GlbSts	78
2.9.9 PFB_GetNetCfg	79
2.9.10 PFB_GetStationList	80
2.9.11 PFB_InitCounters	81
2.9.12 PFB_PutNetCfg	82
2.9.13 PFB_UpdatePassiveStn	84
2.10 DP Master API	85
2.10.1 PFB_AckMasDiagEvent	85
2.10.2 PFB_AckMasError	86
2.10.3 PFB_AckMasEvent	87
2.10.4 PFB_AckMasGlbEvent	88
2.10.5 PFB_Config2bf	89
2.10.6 PFB_ConfigAbf	90
2.10.7 PFB_GetMasCfg	91
2.10.8 PFB_GetMasCfgChk	92
2.10.9 PFB_GetMasCntCfg	93
2.10.10 PFB_GetMasDiagInfo	94
2.10.11 PFB_GetMasGlbCfg	95
2.10.12 PFB_GetMasGlbSts	96
2.10.13 PFB_GetMasPage	97
2.10.14 PFB_GetMasParmData	98
2.10.15 PFB_GetMasRxData	99
2.10.16 PFB_GetMasStatusTable	100
2.10.17 PFB_GetMasSts	101
2.10.18 PFB_GetMasTxData	102
2.10.19 PFB_PutMasCfg	103
2.10.20 PFB_PutMasCfgChk	104
2.10.21 PFB_PutMasCntCfg	106
2.10.22 PFB_PutMasGlbCfg	107
2.10.23 PFB_PutMasGlbCntrlCfg	108
2.10.24 PFB_PutMasParmData	109
2.10.25 PFB_PutMasTxData	110
2.11 DP Slave API	111
2.11.1 PFB_AckSlvDiagEvent	111
2.11.2 PFB_AckSlvError	112
2.11.3 PFB_AckSlvEvent	113
2.11.4 PFB_GetSlvCfg	114
2.11.5 PFB_GetSlvCfgChk	115
2.11.6 PFB_GetSlvDiagInfo	116
2.11.7 PFB_GetSlvParmData	117
2.11.8 PFB_GetSlvRxData	118
2.11.9 PFB_GetSlvStatus	119
2.11.10 PFB_GetSlvTxData	120
2.11.11 PFB_PutSlvCfg	121

2.11.12 PFB_PutSlvCntCfg	122
2.11.13 PFB_PutSlvDiagInfo.....	123
2.11.14 PFB_PutSlvTxData	124
2.12 Layer 2 Message API	125
2.12.1 PFB_AckL2MsgError	125
2.12.2 PFB_AckL2MsgEvent	126
2.12.3 PFB_ActivateL2Msg.....	127
2.12.4 PFB_CloseL2MsgBlk	128
2.12.5 PFB_GetL2MsgCfg	129
2.12.6 PFB_GetL2MsgCntCfg.....	130
2.12.7 PFB_GetL2MsgInfo.....	131
2.12.8 PFB_GetL2MsgMaxRxTxLen.....	132
2.12.9 PFB_GetL2MsgReply	133
2.12.10 PFB_GetL2MsgRxData	134
2.12.11 PFB_GetL2MsgState	135
2.12.12 PFB_GetL2MsgSts.....	136
2.12.13 PFB_GetL2MsgTxData	137
2.12.14 PFB_InitL2MsgBlk.....	138
2.12.15 PFB_InitL2MsgGlb.....	139
2.12.16 PFB_OpenL2MsgBlk.....	140
2.12.17 PFB_PutL2MsgCfg.....	141
2.12.18 PFB_PutL2MsgCntCfg	142
2.12.19 PFB_PutL2MsgInfo	143
2.12.20 PFB_PutL2MsgTxData.....	144
2.12.21 PFB_SendL2Msg	145
2.12.22 PFB_SendL2MsgWaitForReply	146
2.12.23 PFB_TrigL2Msg	148
2.13 Layer 2 SAP API	149
2.13.1 PFB_AckL2SapError	149
2.13.2 PFB_AckL2SapEvent	150
2.13.3 PFB_GetL2SapCfg.....	151
2.13.4 PFB_GetL2SapCntCfg.....	152
2.13.5 PFB_GetL2SapMaxRxTxLen	153
2.13.6 PFB_GetL2SapRequest.....	154
2.13.7 PFB_GetL2SapRxData	155
2.13.8 PFB_GetL2SapSts.....	156
2.13.9 PFB_GetL2SapTxData	157
2.13.10 PFB_GetL2SapType	158
2.13.11 PFB_InitL2SapBlk.....	159
2.13.12 PFB_InitL2SapGlb.....	160
2.13.13 PFB_PutL2SapCfg	161
2.13.14 PFB_PutL2SapCntCfg	162
2.13.15 PFB_PutL2SapResponse.....	163
2.13.16 PFB_PutL2SapTxData.....	164

2.14 DP Master Class II API	165
2.14.1 PFB_AckMC2MasterError	165
2.14.2 PFB_AckMC2MasterEvent	166
2.14.3 PFB_AckMC2SlaveError.....	167
2.14.4 PFB_AckMC2SlaveEvent.....	168
2.14.5 PFB_GetMC2MasterBlock	169
2.14.6 PFB_GetMC2MasterCfg.....	170
2.14.7 PFB_GetMC2MasterRxData	171
2.14.8 PFB_GetMC2SlaveCfg	172
2.14.9 PFB_GetMC2SlaveRd	173
2.14.10 PFB_PutMC2MasterCfg	174
2.14.11 PFB_PutMC2MasterTxData	175
2.14.12 PFB_PutMC2SlaveCfg	176
2.14.13 PFB_SetMC2SlaveAddr	177
PFBMAN32 DLL DPV1 API.....	179
3.1 Introduction	180
3.1.1 DPV1 API Features	180
3.2 Configuration.....	180
3.3 API.....	180
3.3.1 Initialization	180
3.3.2 DPV1 Services	181
3.3.3 DPV1_Init	184
3.3.4 DPV1_Exit	185
3.3.5 MSAC1_Read (PFB_MSAC1_Read_req).....	186
3.3.6 MSAC1_Read (PFB_MSAC1_Read_con)	187
3.3.7 MSAC1_Write (PFB_MSAC1_Write_req)	189
3.3.8 MSAC1_Write (PFB_MSAC1_Write_con)	190
3.3.9 MSAC2_Initiate (PFB_MSAC2_Initiate_req).....	192
3.3.10 MSAC2_Initiate (PFB_MSAC2_Initiate_con)	194
3.3.11 MSAC2_Abort (PFB_MSAC2_Abort_req).....	196
3.3.12 MSAC2Read (PFB_MSAC2_Read_req).....	197
3.3.13 MSAC2Read (PFB_MSAC2_Read_con)	199
3.3.14 MSAC2_Write (PFB_MSAC2_Write_req)	200
3.3.15 MSAC2_Write (PFB_MSAC2_Write_con)	202
3.3.16 MSAC2_Idle (PFB_MSAC2_Idle_req).....	203
3.3.17 MSAC2_Idle (PFB_MSAC2_Idle_con)	205
3.3.18 Structure Definition (PFB_DPV1_ADDR_PARAM)	207
3.3.19 Structure Definition (PFB_DPV1_STS)	208
3.3.20 Meaning of FDL_ErrorCode	209
3.3.21 Meaning of DPV1 Error	209
PFBMAN32 API Structure Definitions.....	211
4.1 Profibus Network.....	212

4.1.1 Card Status Structure.....	212
4.1.2 Diagnostic Counters Structure	215
4.2 Station List Structure	217
4.2.2 Network Configuration Structure.....	218
4.3 DP Master	220
4.3.1 Master Global Configuration Structure.....	220
4.3.2 Master Status Structure	221
4.4 Master Configuration Structure	223
4.4.2 DP Configuration Check Structure	224
4.5 DP Diagnostic Information Structure	224
4.5.2 DP Parameter Data Structure	225
4.6 Layer 2 Message.....	226
4.6.1 Layer 2 Message Configuration Structure	226
4.6.2 Layer 2 Message Status Structure	227
4.7 Layer 2 Message Transmit Structure.....	228
4.8 Layer 2 SAP	229
4.8.1 Layer 2 SAP Configuration Structure	229
4.9 Layer 2 SAP Status Structure	230
4.10 DP Slave	231
4.10.1 Slave Configuration Structure	231
4.11 Slave Status Structure.....	232
4.12 Master Class II.....	234
4.12.1 Master Class II Slave Configuration Structure	234
4.13 Master Class II Slave Read Structure	235
4.13.2 Master Class II Set Slave Address Structure.....	235
4.14 Master Class II Master Configuration Structure.....	236
4.15 Master Class II Read Master Data Structure	237
4.16 Master Class II Read Block Structure	238
4.16.2 Master Class II Master Write Structure.....	238
Warranty and Support.....	239
A.1 Warranty	240
A.2 Technical Support.....	240
A.2.1 Getting Help	240

Preface

Preface Sections:

- Purpose of this Guide
- Conventions

Purpose of this Guide

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

Conventions

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

Style

The following stylistic conventions are used throughout this guide:

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

Special Terms

The following special terms are used throughout this guide:

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

Special Notation

The following special notation is used throughout this guide:



Note

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

1

Introduction

Chapter Sections:

- PFBSCAN Overview
- Terminology

1.1 PFBSCAN Overview

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

General Module Capabilities

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

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

Additional Module Capabilities

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

Card as Master

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

Card as Slave

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

Layer 2 (FDL) Interface

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

2

PFBMAN32 DLL API

Chapter Sections:

- Application Programming Interface (API) for the PFBMAN32 DLL

2.1 Introduction

This section defines the API (*Application Programming Interface*) for the PFBMAN32 DLL. All functions and data types are listed alphabetically.

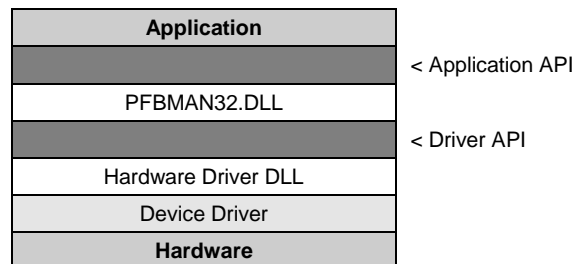
2.2 Overview

PFBMAN32.DLL is a non-thread-safe 32-bit DLL. It provides a common interface that abstracts physical hardware to allow one application or DLL to interface with the Profibus Scanner module on different hardware platforms. It is for use with Windows XP/Server 2003/Vista/Server 2008/Windows 7 (32-bit and 64-bit).

2.3 Services

- PFBMAN32.DLL provides the following services:
- DLL revision details
- Hardware driver management
- Card management (multiple cards, multiple clients)
- Card abstraction using handles eliminates the need for the application to be aware of memory or I/O addresses
- Interrupt support (notification messages)
- Read/Write services at the block, word, byte and bit level

2.4 Application Stack



2.5 Quick Start Using Sample Utilities

2.5.1 Configuring the DP Master

The following steps explain how to configure the DP master using the SST PROFIBUS-exported **BSS** master configuration file or the Siemens COM PROFIBUS exported **2BF** master configuration file.

2.5.1.1 Configuring with SST PROFIBUS / Siemens COM PROFIBUS Master

1. Run **ssbincfg.exe**.
2. Open a connection to the card.
3. From the dialog box, select your BSS (SST) or 2BF (Siemens) file.
4. Click Open.
5. If the selected file was transferred successfully, a confirmation dialog box appears.

2.5.1.2 Configuring Manually

1. Run **dpmascfg.exe**.
2. Open the card. The DP Master Configuration dialog box appears.
3. Set the global parameters: Min Cycle Time and Max Cycle Time.
4. For each slave, set the DP Master block parameters: Slave Station, RX/TX length, ParamMasSts, ParamWdFact1, ParamWdFact2, ParamRdyTime, ParamID and ChkData.

2.5.1.3 Configuring the PFB Network

1. Run **pbnetcfg.exe**.
2. Open the card. The PFB Network Configuration box appears.
3. Set the following fields: Local Station, BAUD, and Option-Active. Additional fields use default values.

2.5.2 Configuring the DP Slave

Configure the slave's network parameters using PFB Network Configuration (see above).

1. Run **dpslvcfg.exe**.
2. Open the card. The DP Slave Dialog Box appears.
3. Set the RX/TX Length field. The other fields are optional.

2.5.3 PROFIBUS Command

1. Run **pfbcmd.exe**.
2. Open the card. The PFB Command dialog box appears.
3. Select the ONLINE command.

2.5.4 DP Monitor

1. Run **dpmon.exe**.
2. Open the card.
3. Select one of the tabs:
 - DP Network - displays all stations on the network
 - Diagnostics - monitors card local diagnostics
 - Master Directory - monitors all configured slaves
 - Master Data - monitors a specific slave's data
 - Master Diagnostics - monitors a specific slave's diagnostics
 - Slave Data - monitors data of the SST-PB3 slave services
 - Slave Diagnostics - monitors diagnostics of the SST-PB3 slave services

2.6 Operating the Application

The typical sequence of events for an application using the PROFIBUS Manager Interface DLL is:

1. If a specific version is required, check the PFBMAN32.DLL version (PFB_Version).
2. Load the hardware driver (PFB_LoadDriver, with the “sspfb32.dll” parameter).
If a specific version is required, check the Hardware Driver DLL version (PFB_DriverVersion).
3. Open a card connection (PFB_OpenCard), where sharing type and firmware module load options are specified.
4. Clear the configuration buffers using PFB_Command (REINIT).
5. Configure the card to act as:

A DP Master

- Use a binary configuration image generated by the SST PROFIBUS Configuration tool (BSS)
- Using various DLL APIs, such as PFB_PutMasGlbCfg, PFB_PutMasCfg, PFB_PutMasParmData and PFB_PutMasCfgChk to configure master parameters, and PFB_PutNetCfg to configure network parameters

A DP Slave:

- Using PFB_PutSlvCfg to configure slave parameters and PFB_PutNetCfg to configure network parameters
6. Put the card online (PFB_Command).
 7. Put the DP Master into RUN mode by setting the PFB_MAS_CTRL_RUN_MODE bit in the pfbMasCntrlCfg register. Use PFB_GetMasGlbCfg to retrieve the current register value, then OR it with the PFB_MAS_CTRL_RUN_MODE bit in MasGlbCfg.pfbMasCntrlCfg. Write this register back to the card using PFB_PutMasGlbCntrlCfg.
 8. Monitor RX/TX Data (PFB_GetMasRx/TxData, PFB_PutMasTxData, PFB_GetSlvRx/TxData, PFB_PutSlvTxData).

9. Go offline (PFB_Command).
10. Close the card connection (PFB_CloseCard).
11. Unload the driver (PFB_FreeDriver).

2.7 Card Access Types

2.7.1.1 Using POINTERS

- Performance LEVEL 1 (fastest): the SS_HCONTROL load flag is required to control page switching and to collect knowledge of the card's structure.
- To get high-speed memory access, the application can point to the card. The PFB_GetCardPtr call maps card memory with a pointer. Using a pointer locks other applications out of using card memory.



Note

To unlock card memory once accessing is complete, the pointer needs to be freed (via the PFB_FreeCardPtr command).

2.7.1.2 Using Direct I/O Calls (e.g., PFB_ReadByte, PFB_WriteByte, PFB_ReadWord)

- Performance LEVEL 2: the SS_HCONTROL load flag is required to control page switching. Access to memory offsets is also required.
- The application can access the card's memory via direct read and write function calls. It must control the page using Hardware Control calls (PFB_ReadHC, PFB_WriteHC).

2.7.1.3 Using Direct Page I/O calls (e.g., PFB_PageReadByte, PFB_PageWriteByte, PFB_PageReadWord)

- Performance LEVEL 3: access to memory offsets is required
- Used for shared direct access to card memory. Page control is achieved through function calls.

2.7.1.4 Using BUFFERS

- Performance LEVEL 4: no shared access limitations or offset knowledge

To share access to the card between applications without limitations, use standard PFBMAN32.DLL calls, such as PFB_Get/PutMasRx/TxData and PFB_GetSlvRx/TxData to access the card's memory.



Note

When a card is opened, it is locked, and cannot be opened by any other application. If an application fails to close a card connection, that card is unavailable to all applications until the computer is restarted.

2.8 General Resource Access

2.8.1 PFB_CancelWaitIrq

2.8.1.1 Description

Cancels a pending WaitIrq request.

2.8.1.2 Prototype

```
BOOL WINAPI PFB_CancelWaitIrq (DWORD CardHandle)
```

2.8.1.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.8.1.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

2.8.1.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0105	Application Interrupt access denied (sharing violation)

2.8.2 PFB_CloseCard

2.8.2.1 Description

Closes the previously opened card connection.

2.8.2.2 Prototype

```
BOOL WINAPI PFB_CloseCard (DWORD CardHandle)
```

2.8.2.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.8.2.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

2.8.2.5 Error Codes

Value (hex)	Description
2000 0100	General failure error code
2000 0102	Invalid CardHandle
2000 0003	Driver not loaded

2.8.3 PFB_DriverVersion

2.8.3.1 Description

Returns the Hardware DLL version information.

2.8.3.2 Prototype

```
BOOL WINAPI PFB_Driver (TCHAR* Buffer, WORD* Version, DWORD Size)
```

2.8.3.3 Arguments

Argument	Description
Buffer	Driver identification string buffer
Version	Minor revision (LSB) Major revision (MSB)

2.8.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.3.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0114	Invalid pointer - Buffer and/or Version

2.8.4 PFB_EnumDrivers

2.8.4.1 Description

Returns the available driver name.

2.8.4.2 Prototype

```
BOOL WINAPI PFB_EnumDrivers (DWORD Index, char* Name, DWORD* Size)
```

2.8.4.3 Arguments

Argument	Description
Index	Index of the card
Name	Buffer for storing the card name
Size	Size of the buffer for the card name. Returns the number of characters stored in the buffer.

2.8.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.4.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure

2.8.5 PFB_FreeCardPtr

2.8.5.1 Description

Frees a pointer retrieved from a previous PFB_GetCardPtr call.

2.8.5.2 Prototype

```
BOOL WINAPI PFB_FreeCardPtr (DWORD CardHandle, volatile void* *ThePointer)
```

2.8.5.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
ThePointer	Pointer to be freed

2.8.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.5.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 011a	Card access timeout
2000 0114	Invalid pointer - ThePointer

2.8.6 PFB_FreeDriver

2.8.6.1 Description

Unloads the Hardware Driver DLL. The link-count for the driver is decremented and, if no other applications are using the driver, it is unloaded.

2.8.6.2 Prototype

```
BOOL WINAPI PFB_FreeDriver (void)
```

2.8.6.3 Arguments

This function has no arguments.

2.8.6.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.6.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded

2.8.7 PFB_GetBusType

2.8.7.1 Description

Gets the card's bus type.

2.8.7.2 Prototype

```
BOOL WINAPI PFB_GetBusType (DWORD CardHandle, DWORD* BusType)
```

2.8.7.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BusType	Pointer to a DWORD to receive the bus type

2.8.7.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.7.5 Supported Bus Types

Value	Description
1	ISA
5	PCI
8	PCMCIA

2.8.7.6 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – BusType

2.8.8 PFB_GetCardPtr

2.8.8.1 Description

Gets a pointer to the start of a card's host interface memory.

2.8.8.2 Prototype

```
BOOL WINAPI PFB_GetCardPtr (DWORD CardHandle, volatile void* *ThePointer)
```

2.8.8.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
ThePointer	Pointer to be modified

2.8.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.8.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 0114	Invalid pointer - ThePointer
2000 011a	Card access timeout

2.8.9 PFB_GetHomePage

2.8.9.1 Description

Gets the Home Page.

2.8.9.2 Prototype

```
BOOL WINAPI PFB_GetHomePage (DWORD CardHandle, BYTE* HomePage)
```

2.8.9.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
HomePage	Destination of Home Page

2.8.9.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.9.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0114	Invalid pointer - HomePage

2.8.10 PFB_GetMem

2.8.10.1 Description

Gets data from card-mapped memory.

2.8.10.2 Prototype

```
BOOL WINAPI PFB_GetMem (DWORD CardHandle, BYTE Page, WORD Ofs, WORD Len, BYTE* DataBuf)
```

2.8.10.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Page	Page number of the data to be read
Ofs	Offset within the page (0-3fff) for original cards. Offset within 2 pages (0-7fff) for the expanded I/O card.
Len	Length of a block to be read (1-4000) for original cards. Length of a block to be read (1-8000) for the expanded I/O card.
DataBuf	Destination of data to be read

2.8.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.10.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DataBuf
2000 011a	Card access timeout

2.8.11 PFB_LoadDriver

2.8.11.1 Description

Loads the Hardware Driver DLL. The standard driver name is “sspfb32.dll”.

2.8.11.2 Prototype

```
BOOL WINAPI PFB_LoadDriver (TCHAR* DriverName )
```

2.8.11.3 Arguments

Argument	Description
DriverName	Name and optional path to the hardware driver DLL

2.8.11.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code

2.8.11.5 Error Codes

Value (hex)	Description
2000 0000	Driver loaded
2000 0001	Driver not found
2000 0002	Invalid driver

2.8.12 PFB_OpenCard

2.8.12.1 Description

Opens a card connection. The card handle returned by this function is used by the DLL client to identify the card to all other services. You need to call PFB_CloseCard when your application does not require access it anymore. The limit of open connections to all PROFIBUS cards is 16.

2.8.12.2 Prototype

```
BOOL WINAPI PFB_OpenCard (DWORD* CardHandle, TCHAR* CardName, void* Module,  
DWORD ModuleId, WORD ShareFlags, WORD LoadFlags)
```

2.8.12.3 Arguments

Argument	Description
CardHandle	Upon successful completion of PFB_OpenCard, CardHandle is used for subsequent API calls
CardName	The name assigned to the interface card during hardware installation (stored in registry).
Module	Set to NULL - reserved
ModuleId	The ID of the required application module (0xBB01 for original cards, and 0xBB03 for the expanded I/O card).
ShareFlags	Connection share flags (see below)
LoadFlags	Application module loader flags (see below)

2.8.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.12.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0101	CardHandle not available
2000 0103	Write access denied (sharing violation). Card must be opened with SS_WRITE flag.
2000 0104	Card Interrupt access denied (sharing violation)
2000 0105	Application Interrupt access denied (sharing violation)
2000 0106	Hardware Control access denied (sharing violation). Card is already opened with SS_HCONTROL flag.
2000 0107	Overlap conflict
2000 0109	Card not found (incorrect hardware configuration)
2000 010b	Invalid / damaged application module
2000 010d	Card diagnostic failure
2000 0114	Invalid pointer - CardHandle
2000 0115	Invalid interrupt level
2000 0116	Invalid I/O address
2000 0117	Invalid I/O length
2000 0118	Invalid memory address
2000 0119	Invalid memory length
2000 011a	Card access timeout
2000 011b	Driver access denied
2000 011d	Module load request denied
2000 011f	CPU not running. Module must be reloaded.
2000 0121	Timeout waiting for alive signal from BOOT program.
2000 0122	Timeout waiting for module diagnostics, or module failure.

2.8.12.6 ShareFlags

This argument controls multiple application arbitration for a single card.

Bit	Name	Description
0	SS_WRITE	Card is opened for shared write access
1	SS_XWRITE	Card is opened for exclusive write access
2	SS_HCONTROL	Card is opened for exclusive hardware control access
3	SS_CARDIRQ	Card is opened for exclusive card interrupt access (interrupts from the application to the card)
4	SS_APPIRQ	Card is opened for exclusive application interrupt access (interrupts from the card to the application)
5-7	Reserved	Reserved. Set to 0.
8	SS_USER1**	Exclusive user access 1
9	SS_USER2**	Exclusive user access 2
10	SS_USER3**	Exclusive user access 3
11	SS_USER4**	Exclusive user access 4
12-13	Reserved	Reserved. Set to 0.
14	SS_OVERLAP	Obsolete.
15	Reserved	Reserved. Set to 0.

** Exclusive user access flags are used to avoid conflicts between applications sharing access to the same card. The meaning of these flags is application specific.

2.8.12.7 LoadFlags

This argument controls the application module loader and also selects some connection modes of operation.



Note

The SS_GO_OFF_ONCLOSE and SS_STAY_RUN_ONCLOSE flags take effect only when the application is closing and there are no other applications connected to the card. If neither of these flags are set, the card is put in PROGRAM mode (shuts down the outputs).

Bit	Name	Description
0	SS_REPLACE	Do not replace module if it's already loaded. If the firmware has never been loaded, the application opening the connection must be the only one with write access.
1	SS_RELOAD	If no other applications have write access, reload the specified module. Opening more than one connection at the same time with this flag set is not permitted and will result in the MODULE_LOAD_REQUEST_DENIED error.
3	SS_GO_OFF_ONCLOSE	Last application's option to take card OFFLINE when finished
4	SS_STAY_RUN_ONCLOSE	Last application's option to leave the card in unchanged state
6-15	Reserved	Always 0

2.8.13 PFB_PageReadBit

2.8.13.1 Description

Reads a bit from the card host interface memory of a specific page.

2.8.13.2 Prototype

```
BOOL PFB_PageReadBit (DWORD CardHandle, DWORD Offset, DWORD Bit, BYTE* Buffer,  
BYTE Page)
```

2.8.13.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the byte containing the target bit
Bit	The target bit within the byte to be read (0-7)
Buffer	Contains the value of bit read
Page	Page number to read from

2.8.13.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.13.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 010e	Invalid offset
2000 010f	Invalid Bit number (0-7)
2000 0114	Invalid pointer - Buffer
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.14 PFB_PageReadBlock

2.8.14.1 Description

Reads a block of data from the card host interface memory of a specific page.

2.8.14.2 Prototype

```
BOOL PFB_PageReadBlock (DWORD CardHandle, DWORD Offset, void* Block, DWORD  
Size, BYTE Page)
```

2.8.14.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the first byte of the target block (0x0000 - 0x3FFF) for original cards. The offset from the start of the host interface memory to the first byte of the target block (0x0000 – 0x7FFF) for expanded I/O cards.
Block	Pointer to the buffer for data read
Size	Number of bytes to read
Page	Page number to read from

2.8.14.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.14.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 010e	Invalid offset
2000 0110	Invalid Size
2000 0114	Invalid pointer - Block
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.15 PFB_PageReadByte

2.8.15.1 Description

Reads an 8-bit byte from the card host interface memory of a specific page.

2.8.15.2 Prototype

```
BOOL PFB_PageReadByte (DWORD CardHandle, DWORD Offset, BYTE* Buffer, BYTE Page)
```

2.8.15.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the target byte (0x0000 - 0x3FFF)
Buffer	Buffer for value read
Page	Page number to read from

2.8.15.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.15.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 010e	Invalid offset
2000 0114	Invalid pointer - Buffer
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.16 PFB_PageReadWord

2.8.16.1 Description

Reads a 16-bit word from the card host interface memory of a specific page.

2.8.16.2 Prototype

```
BOOL PFB_PageReadWord (DWORD CardHandle, DWORD Offset, WORD* Buffer, BYTE Page)
```

2.8.16.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the first byte (LSB) of the target word (0x0000 - 0x3FFE)
Buffer	Buffer to receive data word read
Page	Page number to read from

2.8.16.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.16.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 010e	Invalid offset
2000 0114	Invalid pointer - Buffer
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.17 PFB_PageToggleBit

2.8.17.1 Description

Toggles a bit in the card host interface memory on a specific page.

2.8.17.2 Prototype

```
BOOL PFB_PageToggleBit (DWORD CardHandle, DWORD Offset, DWORD Bit, BYTE Page)
```

2.8.17.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the byte containing the target bit (0x0000 - 0x3FFF)
Bit	The target bit within the byte to be toggled (0-7)
Page	Page number to toggle on

2.8.17.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.17.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag
2000 010e	Invalid offset
2000 010f	Invalid Bit number (0-7)
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.18 PFB_PageWriteBit

2.8.18.1 Description

Writes a bit in the card host interface memory to a specific page.

2.8.18.2 Prototype

```
BOOL PFB_PageWriteBit (DWORD CardHandle, DWORD Offset, DWORD Bit, DWORD Data,  
BYTE Page)
```

2.8.18.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the byte containing the target bit (0x0000 - 0x3FFF)
Bit	The target bit within the byte to be written (0-7)
Data	The data value to write (a non-zero value sets the bit)
Page	Page number to write to

2.8.18.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.18.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 010f	Invalid Bit number (0-7)
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.19 PFB_PageWriteBlock

2.8.19.1 Description

Writes a block of data to the card host interface memory of a specific page.

2.8.19.2 Prototype

```
BOOL PFB_PageWriteBlock (DWORD CardHandle, DWORD Offset, void* Block, DWORD  
Size, BYTE Page)
```

2.8.19.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the first byte of the target block (0x0000 - 0x3FFF) for the original cards. The offset from the start of the host interface memory to the first byte of the target block (0x0000 – 7FFF) for expanded I/O cards.
Block	Pointer to data to be written
Size	Number of bytes to write
Page	Page number to write to

2.8.19.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.19.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 0110	Invalid Size
2000 0114	Invalid pointer
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.20 PFB_PageWriteByte

2.8.20.1 Description

Writes an 8-bit byte to the card host interface memory of a specific page.

2.8.20.2 Prototype

```
BOOL PFB_PageWriteByte (DWORD CardHandle, DWORD Offset, BYTE Data, BYTE Page)
```

2.8.20.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the target byte (0x0000 - 0x3FFF)
Data	Data byte to write
Page	Page number to write to

2.8.20.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.20.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.21 PFB_PageWriteWord

2.8.21.1 Description

Writes a 16-bit word to the card host interface memory of a specific page.

2.8.21.2 Prototype

```
BOOL PFB_PageWriteWord (DWORD CardHandle, DWORD Offset, WORD Data, BYTE Page)
```

2.8.21.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the first byte (LSB) of the target word (0x0000 - 0x3FFE)
Data	Data word to write
Page	Page number to write to

2.8.21.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.21.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 011a	Card access timeout
2000 011e	Invalid Page (0-15)

2.8.22 PFB_PutMem

2.8.22.1 Description

Writes data in the card-mapped memory.

2.8.22.2 Prototype

```
BOOL WINAPI PFB_PutMem (DWORD CardHandle, BYTE Page, WORD ofs, WORD Len, BYTE* DataBuf)
```

2.8.22.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Page	Page number the data is to be written to
ofs	Offset within the page (0-3FFFh) for original cards. Offset within 2 pages (0-7FFFh) for the expanded I/O card.
Len	Length of a block to be written (1-4000h)
DataBuf	Source of data to be written

2.8.22.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.22.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.8.23 PFB_ReadBit

2.8.23.1 Description

Reads a bit from the card host interface memory.

2.8.23.2 Prototype

```
BOOL PFB_ReadBit (DWORD CardHandle, DWORD Offset, DWORD Bit, BYTE* Buffer)
```

2.8.23.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the byte containing the target bit (0-3FFFh)
Bit	The target bit within the byte to be read (0-7)
Buffer	Contains the value of the bit read

2.8.23.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.23.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 010e	Invalid offset
2000 010f	Invalid bit number (0-7)
2000 0114	Invalid pointer – Buffer
2000 011a	Card access timeout (Card is in OverLap Mode)

2.8.24 PFB_ReadBlock

2.8.24.1 Description

Reads a block of data from the card host interface memory.

2.8.24.2 Prototype

```
BOOL PFB_ReadBlock (DWORD CardHandle, DWORD Offset, void* Block, DWORD Size)
```

2.8.24.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the first byte of the target block (0 - 3FFFh)
Block	Pointer to buffer for data read
Size	Number of bytes to read

2.8.24.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.24.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 010e	Invalid offset
2000 0110	Invalid Size
2000 0114	Invalid pointer – Block
2000 011a	Card access timeout (Card is in OverLap Mode)

2.8.25 PFB_ReadByte

2.8.25.1 Description

Reads an 8-bit byte from the card host interface memory.

2.8.25.2 Prototype

```
BOOL PFB_ReadByte (DWORD CardHandle, DWORD Offset, BYTE* Buffer)
```

2.8.25.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the target byte (0 - 3FFFh)
Buffer	Buffer for value read

2.8.25.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.25.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 010e	Invalid offset
2000 0114	Invalid pointer – Buffer
2000 011a	Card access timeout (Card is in OverLap Mode)

2.8.26 PFB_ReadHC

2.8.26.1 Description

Reads a hardware control signal.

2.8.26.2 Prototype

```
BOOL PFB_ReadHC (DWORD CardHandle, WORD Signal, WORD* Buffer)
```

2.8.26.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Signal	Hardware control signal ID (see below)
Buffer	Value of control signal read

2.8.26.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

2.8.26.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0111	Invalid Signal ID
2000 0114	Invalid pointer – Buffer
2000 0120	PCI Card does not support port addressing

2.8.26.6 Signal ID

Signal ID	Name	Access	Description
4	PFB3_CTRL_OFS	R/W	Control Register. See below.
5	PFB3_ADDR_OFS	R	AddrMatch Register. See below.
6	PFB3_BANK_OFS	R	BankSelect Register. See below.
7	PFB3_WINSIZE_OFS	R	WinSize Register. See below.
8	PFB3_HOSTIRQ_OFS	R	HostIrq Register. See below.
9	PFB3_LED_OFS	R	LedReg Register. See below.
10	PFB3_DEBUG_OFS	R/W	Debug Register. See below.
11	PFB3_HDR_OFS	R/W	HDR Register. See below.
12-255	Reserved	-	-

2.8.26.7 Control Register (PFB3_CTRL_OFS)

Bit Offsets and Names							
7	6	5	4	2	3	1	0
CardRun	MemEn	IntEn	WdTout	HostIrq1	HostIrq0	CardIrq1	CardIrq0

The following table describes each Control Register Bit:

Bit Name	Description
CardRun	High (1) indicates that the card is in "Run" mode (CPU is out of reset and will attempt to execute firmware). Low (0) indicates that the card is in "Stop" mode (CPU is in reset) or a WdTout has occurred (also see WdTout bit).
MemEn	High (1) enables shared memory decoding of addresses in this board's range. This board's range is defined by the AddrMatch register.
IntEn	High (1) enables interrupts on IrqLevel when a HostIrq bit is high (1).
WdTout	High ('1') indicates that a watchdog timeout has occurred.
HostIrq1 HostIrq0	High ('1') indicates that the specified interrupt is in progress and has not yet been serviced. Writing HostIrq1, HostIrq0 high ('1') causes the pending Host interrupt to end (via the Interrupt service routine running on the host)
CardIrq1 CardIrq0	Writing CardIrq1, CardIrq0 high ('1') generates an interrupt to the card at the appropriate level

2.8.26.8 AddrMatch Register (PFB3_ADDR_OFS)

Bit Offsets and Names							
7	6	5	4	2	3	1	0
Enable	A18	A17	A16	A15	A14	-	-

This register always reads zero, and writing to it has no effect.

2.8.26.9 BankSelect Register (PFB3_BANK_OFS)

Bit Offsets and Names							
7	6	5	4	2	3	1	0
0	0	0	BA17	BA15	BA14	-	-

This register always reads zero, and writing to it has no effect.

2.8.26.10 WinSize Register (PFB3_WINSIZE_OFS)

Bit Offsets and Names							
7	6	5	4	3	2	1	0
WS19	WS18	WS17	WS16	WS15	WS14	WS13	WS12

This register always reads 0x3F, and writing to it has no effect.

2.8.26.11 HostIrq Register (PFB3_HOSTIRQ_OFS)

Bit Offsets and Names							
7	6	5	4	3	2	1	0
-	-	-	-	Int_sel_3	Int_sel_2	Int_sel_1	Int_sel_0

This register always reads zero, and writing to it has no effect.

2.8.26.12 LedReg Register (PFB3_LED_OFS)

The LedReg register represents the state of the card's LEDs. The state of this register is controlled by firmware.

Bit Offsets and Names							
7	6	5	4	3	2	1	0
-	-	-	-	CommRed	CommGrn	SysRed	SysGrn

The Communication LED conforms to the following standard.

CommGrn	CommRed	LED Color
0	0	Off
0	1	Red
1	0	Green
1	1	Amber

The System LED conforms to the following standard.

SysGrn	SysRed	LED Color
0	0	Off
0	1	Red
1	0	Green
1	1	Amber

2.8.26.13 Debug Register PFB3_DEBUG_OFS)

Bit Offsets and Names							
7	6	5	4	3	2	1	0
HWRReset	-	-	JTAGGEN	CPUTRST	CPUTMS	CPUTDI	CPUTCK

Reserved for future use.

2.8.26.14 HDR Register (PFB3_HDR_OFS)

Bit Offsets and Names							
7	6	5	4	3	2	1	0
HostDataReg (written by CPU)							

Reserved for future use.

2.8.27 PFB_ReadWord

Description Reads a 16-bit word from the card host interface memory.

2.8.27.1 Prototype

```
BOOL PFB_ReadWord (DWORD CardHandle, DWORD Offset, WORD* Buffer)
```

2.8.27.2 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the first byte (LSB) of the target word (0 - 3FFEh)
Buffer	Buffer to receive data word read

2.8.27.3 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.27.4 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 010e	Invalid offset
2000 0114	Invalid pointer – Buffer
2000 011a	Card access timeout (Card is in OverLap Mode)

2.8.28 PFB_SetAccessTimeout

2.8.28.1 Description

Sets the card access timeout delay.

2.8.28.2 Prototype

```
BOOL WINAPI PFB_SetAccessTimeout (DWORD CardHandle, DWORD Timeout)
```

2.8.28.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Timeout	The access timeout delay in milliseconds (default is 1 sec)

2.8.28.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.28.5 Errors

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle

2.8.29 PFB_ToggleBit

2.8.29.1 Description

Toggles a bit in the card host interface memory.

2.8.29.2 Prototype

```
BOOL PFB_ToggleBit( DWORD CardHandle, DWORD Offset, DWORD Bit )
```

2.8.29.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the byte containing the target bit (0 - 3FFFh)
Bit	The target bit within the byte to be toggled (0-7)

2.8.29.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.29.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 010f	Invalid Bit number (0-7)
2000 011a	Card access timeout (Card is in OverLap Mode)

2.8.30 PFB_Version

2.8.30.1 Description

Gets the Hardware Driver version information.

2.8.30.2 Prototype

```
BOOL PFB_Version (TCHAR* Buffer, WORD* Version, DWORD Size)
```

2.8.30.3 Arguments

Argument	Description
Buffer	Buffer to receive the DLL version string
Version	Word to contain the DLL versions number MSB - Major revision LSB - Minor revision
Size	The size of the buffer (version string is truncated to fit)

2.8.30.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.30.5 Error Codes

Value (hex)	Description
2000 0201	Invalid pointer - Buffer
2000 0202	Invalid pointer - Version

2.8.31 PFB_WaitIrq

2.8.31.1 Description

Waits for card interrupt.

2.8.31.2 Prototype

```
BOOL WINAPI PFB_lWaitIrq (DWORD CardHandle)
```

2.8.31.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.8.31.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.31.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Application Irq access denied

2.8.32 PFB_WriteBit

2.8.32.1 Description

Writes a bit in the card's host interface memory.

2.8.32.2 Prototype

```
BOOL PFB_WriteBit (DWORD CardHandle, DWORD Offset, DWORD Bit, DWORD Data)
```

2.8.32.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the byte containing the target bit (0x0000 - 0x3FFF)
Bit	The target bit within the byte to be written (0-7)
Data	The data value to write (a non-zero value sets the bit)

2.8.32.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.32.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 010f	Invalid Bit number (0-7)
2000 011a	Card access timeout (Card is in OverLap Mode)

2.8.33 PFB_WriteBlock

2.8.33.1 Description

Writes a block of data to the card's host interface memory.

2.8.33.2 Prototype

```
BOOL PFB_WriteBlock (DWORD CardHandle, DWORD Offset, void* Block, DWORD Size)
```

2.8.33.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the first byte of the target block (0 - 3FFFh)
Block	Pointer to data to be written
Size	Number of bytes to write

2.8.33.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.33.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 0110	Invalid Size
2000 011a	Card access timeout (Card is in OverLap Mode)

2.8.34 PFB_WriteByte

2.8.34.1 Description

Writes an 8-bit byte to the card's host interface memory.

2.8.34.2 Prototype

```
BOOL PFB_WriteByte (DWORD CardHandle, DWORD Offset, BYTE Data)
```

2.8.34.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the target byte (0 - 3FFFh)
Data	Data byte to write

2.8.34.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.34.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 011a	Card access timeout (card is in OverLap Mode)
2000 0102	Invalid card handle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset

2.8.35 PFB_WriteHC

2.8.35.1 Description

Writes a hardware control signal.

2.8.35.2 Prototype

```
BOOL PFB_WriteHC (DWORD CardHandle, DWORD Signal, WORD Data)
```

2.8.35.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Signal	Hardware control signal ID
Data	New signal value

2.8.35.4 Returns

Value	Description
TRUE	Success
FALSE	Error, use GetLastError to retrieve error code

2.8.35.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0106	Hardware Control access denied (sharing violation), card is already opened with SS_HCONTROL flag
2000 0111	Invalid Signal ID
2000 0113	Invalid Signal Value
2000 0120	PCI Card does not support port addressing.

2.8.35.6 Signal ID

Signal ID	Name	Access	Description
4	PFB3_CTRL_OFS	R/W	Control Register. See below.
5	PFB3_ADDR_OFS	R	AddrMatch Register. See below.
6	PFB3_BANK_OFS	R	BankSelect Register. See below.
7	PFB3_WINSIZE_OFS	R	WinSize Register. See below.
8	PFB3_HOSTIRQ_OFS	R	HostIrq Register. See below.
9	PFB3_LED_OFS	R	LedReg Register. See below.
10	PFB3_DEBUG_OFS	R/W	Debug Register. See below.
11	PFB3_HDR_OFS	R/W	HDR Register. See below.
12-255	Reserved	-	-

2.8.35.7 Control Register (PFB3_CTRL_OFS)

Bit Offsets and Names							
7	6	5	4	2	3	1	0
CardRun	MemEn	IntEn	WdTout	HostIrq1	HostIrq0	CardIrq1	CardIrq0

The following table describes each Control Register Bit:

Bit Name	Description
CardRun	High (1) indicates that the card is in "Run" mode (CPU is out of reset and will attempt to execute firmware). Low (0) indicates that the card is in "Stop" mode (CPU is in reset) or a WdTout has occurred (also see WdTout bit).
MemEn	High (1) enables shared memory decoding of addresses in this board's range. This board's range is defined by the AddrMatch register.
IntEn	High (1) enables interrupts on IrqLevel when a HostIrq bit is high (1).
WdTout	High ('1') indicates that a watchdog timeout has occurred.
HostIrq1 HostIrq0	High ('1') indicates that the specified interrupt is in progress and has not yet been serviced. Writing HostIrq1, HostIrq0 high ('1') causes the pending Host interrupt to end (via the Interrupt service routine running on the host)
CardIrq1 CardIrq0	Writing CardIrq1, CardIrq0 high ('1') generates an interrupt to the card at the appropriate level

2.8.35.8 AddrMatch Register (PFB3_ADDR_OFS)

Bit Offsets and Names							
7	6	5	4	2	3	1	0
Enable	A18	A17	A16	A15	A14	-	-

This register always reads zero, and writing to it has no effect.

2.8.35.9 BankSelect Register (PFB3_BANK_OFS)

Bit Offsets and Names							
7	6	5	4	2	3	1	0
0	0	0	BA17	BA15	BA14	-	-

This register always reads zero, and writing to it has no effect.

2.8.35.10 WinSize Register (PFB3_WINSIZE_OFS)

Bit Offsets and Names							
7	6	5	4	3	2	1	0
WS19	WS18	WS17	WS16	WS15	WS14	WS13	WS12

This register always reads 0x3F, and writing to it has no effect.

2.8.35.11 HostIrq Register (PFB3_HOSTIRQ_OFS)

Bit Offsets and Names							
7	6	5	4	3	2	1	0
-	-	-	-	Int_sel_3	Int_sel_2	Int_sel_1	Int_sel_0

This register always reads zero, and writing to it has no effect.

2.8.35.12 LedReg Register (PFB3_LED_OFS)

The LedReg register represents the state of the card's LEDs. The state of this register is controlled by firmware.

Bit Offsets and Names							
7	6	5	4	3	2	1	0
-	-	-	-	CommRed	CommGm	SysRed	SysGrn

The System LED conforms to the following standard.

SysGrn	SysRed	LED Color
0	0	Off
0	1	Red
1	0	Green
1	1	Amber

The Communication LED conforms to the following standard.

CommGrn	CommRed	LED Color
0	0	Off
0	1	Red
1	0	Green
1	1	Amber

2.8.35.13 Debug Register PFB3_DEBUG_OFS)

Bit Offsets and Names							
7	6	5	4	3	2	1	0
HWReset	-	-	JTAGGEN	CPUTRST	CPUTMS	CPUTDI	CPUTCK

Reserved for future use.

2.8.35.14 HDR Register (PFB3_HDR_OFS)

Bit Offsets and Names							
7	6	5	4	3	2	1	0
HostDataReg (written by CPU)							

Reserved for future use.



Note

Setting the memory page register for a PCI card is acceptable (the software will emulate the page setting). All other calls to PFB_WriteHC() are illegal for PCI cards.

2.8.36 PFB_WriteWord

2.8.36.1 Description

Writes a 16-bit word to the card host interface memory.

2.8.36.2 Prototype

```
BOOL PFB_WriteWord (DWORD CardHandle, DWORD Offset, WORD Data)
```

2.8.36.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Offset	The offset from the start of the host interface memory to the first byte (LSB) of the target word (0x0000 - 0x3FFE)
Data	Data word to write

2.8.36.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.8.36.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card has already been opened, with exclusive write access.
2000 010e	Invalid offset
2000 011a	Card access timeout (card is in OverLap Mode)

2.9 Profibus Network

2.9.1 PFB_AckStnChange

2.9.1.1 Description

Acknowledges a station change.

2.9.1.2 Prototype

```
BOOL PFB_AckStnChange (DWORD CardHandle)
```

2.9.1.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.9.1.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.1.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.9.2 PFB_Command

2.9.2.1 Description

Executes PFBSCAN module commands.

2.9.2.2 Prototype

```
BOOL WINAPI PFB_Command (DWORD CardHandle, TCHAR* Command)
```

2.9.2.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Command	Name of the command to be executed

2.9.2.4 Commands

Command	Description
"auto_dp_cfg"	Automatically builds master control blocks for DP slaves (requires version 1.55 of the firmware or greater)
"autobaud"	Automatic Baud Detect
"cfg_2bf_shram"	Configure card with a binary image (COM PROFIBUS format)
"cfg_abf_shram"	Configure card with a binary image (BSS format)
"cfg_flash_go_on"	Configure network and master parameters from flash, and then go on line
"cfg_from_flash"	Configure card with configuration in flash
"chk_net_cfg"	Check network parameters and assign defaults
"clear_err"	Clear card error
"clr_cfg_buf"	Clear pfbBinCfgPage
"cpy_mas_cfg"	Copy a page of configuration from pfbBinCfgPage
"mas_assign_addr"	Assign and fill in data address (page/offset) for DP master
"online"	Put card online
"offline"	Put card offline; clear network, master and slave configuration. Card must be reconfigured
"pgm_to_flash"	Program flash with current configuration
"reinit"	Reinitialize memory and all parameters

2.9.2.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.2.6 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0215	Invalid command
2000 0216	Not Online state (Command = "offline")
2000 0217	No error to clear (Command = "clear_err")
2000 0218	Not Offline state (all commands except : "offline" and "clear_err")
2000 0219	Timeout executing command
2000 03xx	Status Error (refer to definitions related to pfbStatus in Section 4.1.1.1).

2.9.3 PFB_GetCardStatus

2.9.3.1 Description

Gets the card status structure.

2.9.3.2 Prototype

```
BOOL WINAPI PFB_GetCardStatus (DWORD CardHandle, PFB_CARD_STS* CrdSts).  
For details, refer to PFB\_CARD\_STS, in Section 4.1.1.1.
```

2.9.3.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
CrdSts	Destination of card status

2.9.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.3.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - CrdSts
2000 011a	Card access timeout

2.9.4 PFB_GetCountersSts

2.9.4.1 Description

Gets the counter status. To initialize the counters, use the PFB_InitCounters function. The card then clears the counters to 0 and sets CntrsSts to 0.

2.9.4.2 Prototype

```
BOOL PFB_GetCountersSts (DWORD CardHandle, BYTE* CntrsSts)
```

2.9.4.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
CntrsSts	Destination of counters status

2.9.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.4.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - CtrsSts
2000 011a	Card access timeout

2.9.5 PFB_GetDiagCounters

2.9.5.1 Description

Gets a local diagnostic counters list.

2.9.5.2 Prototype

```
BOOL WINAPI PFB_GetDiagCounters (DWORD CardHandle, PFB_DIAG_CTRS* DiagCtrs).
```

For details, refer to [PFB_DIAG_CTRS](#), in Section 4.1.2.1.

2.9.5.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
DiagCtrs	Destination of local diagnostic counters

2.9.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.5.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DiagCtrs
2000 011a	Card access timeout

2.9.6 PFB_GetEvent

2.9.6.1 Description

Gets an event and its argument, and checks for more queue events.

2.9.6.2 Prototype

```
BOOL WINAPI PFB_GetEvent (DWORD CardHandle, WORD* Event, BOOL* MoreEvt)
```

2.9.6.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Event	Destination of event
MoreEvt	There are more events in queue (TRUE, FALSE)

2.9.6.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.6.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 021a	Invalid Event pointer
2000 021c	Invalid More Event pointer

2.9.7 PFB_GetL2GlbCntrl

2.9.7.1 Description

Gets the Layer 2 global control register.

2.9.7.2 Prototype

```
BOOL WINAPI PFB_GetL2GlbCntrl (DWORD CardHandle, WORD* L2GlbCntrl)
```

2.9.7.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
L2GlbCntrl	Destination of Layer 2 global control register

2.9.7.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.7.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2GlbCntrl
2000 011a	Card access timeout

2.9.8 PFB_GetL2GlbSts

2.9.8.1 Description

Gets the Layer 2 global status.

2.9.8.2 Prototype

```
BOOL WINAPI PFB_GetL2GlbSts (DWORD CardHandle, WORD* L2GlbSts)
```

2.9.8.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
L2GlbSts	Destination of Layer 2 global status

2.9.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.8.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2GlbSts
2000 011a	Card access timeout

2.9.9 PFB_GetNetCfg

2.9.9.1 Description

Gets the network configuration parameters.

2.9.9.2 Prototype

```
BOOL WINAPI PFB_GetNetCfg (DWORD CardHandle, PFB_NET_CFG* NetCfgTable).  
For details, refer to PFB\_NET\_CFG, in Section 4.2.2.1.
```

2.9.9.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
NetCfgTable	Destination of Network Configuration Data

2.9.9.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.9.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - NetCfgTable
2000 011a	Card access timeout

2.9.10 PFB_GetStationList

2.9.10.1 Description

Gets the active station list.

2.9.10.2 Prototype

```
BOOL WINAPI PFB_GetStationList (DWORD CardHandle, PFB_STN_LST* StnLst).
```

For details, refer to [PFB_STN_LST](#), in Section 4.2.1.1.

2.9.10.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
StnLst	Destination of Station List

2.9.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.10.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - StnLst
2000 011a	Card access timeout

2.9.11 PFB_InitCounters

2.9.11.1 Description

Initializes diagnostic counters. All counters are cleared to 0.

2.9.11.2 Prototype

```
BOOL PFB_InitCounters (DWORD CardHandle)
```

2.9.11.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.9.11.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.11.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.9.12 PFB_PutNetCfg

2.9.12.1 Description

Configures the network parameters.

2.9.12.2 Prototype

```
BOOL WINAPI PFB_PutNetCfg (DWORD CardHandle, PFB_NET_CFG* NetCfgTable).
```

For details, refer to [PFB_NET_CFG](#), in Section 4.2.2.1.

2.9.12.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
NetCfgTable	Source of Network Configuration Data

2.9.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.12.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0204	Invalid Event Enable state (refer to definitions related to pfbEvtEna , in Section 4.2.2.1).
2000 0205	Invalid Interrupt Enable state (refer to definitions related to pfbIntEna , in Section 4.2.2.1).
2000 0206	Invalid Station number (0-126)
2000 0207	Invalid Highest Station (0-126, 255 - disable)
2000 0208	Invalid Active state (0,1)
2000 0209	Invalid Baud rate (refer to definitions related to pfbBaud , in Section 4.2.2.1).
2000 020a	Invalid Options (refer to definitions related to pfbOptions , in Section 4.2.2.1).
2000 020b	Invalid Target Token Rotation Time (256-16,777,215 tBit, 0-card sets default value)
2000 020c	Invalid Slot Time (37-16,383 tBit, 0-card set default value)
2000 020d	Invalid Idle Time 1 (35-1,023 tBit, 0-card set default value)
2000 020e	Invalid Idle Time 2 (35-1,023 tBit, 0-card set default value)
2000 020f	Invalid Ready Time (11-1,023 tBit, 0-card set default value)
2000 0210	Invalid Quiet Time (0-127 tBit, 255-card set default value)
2000 0211	Invalid Token Retry Limit (0-15)
2000 0212	Invalid Message Retry Limit (0-15)
2000 0213	Invalid Token Error Limit (0-255)
2000 0214	Invalid Response Error Limit (0-15)
2000 021b	Network configuration error
2000 03xx	Status Error (refer to definitions related to pfbStatus in Section 4.1.1.1).
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.9.13 PFB_UpdatePassiveStn

2.9.13.1 Description

Updates a passive station in the Active Station List.

2.9.13.2 Prototype

```
BOOL PFB_UpdatePassiveStn (DWORD CardHandle)
```

2.9.13.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.9.13.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.9.13.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 025a	Timeout updating passive stations

2.10 DP Master API

2.10.1 PFB_AckMasDiagEvent

2.10.1.1 Description

Acknowledges a master diagnostic event so that the next master diagnostic block event can be processed.

2.10.1.2 Prototype

```
BOOL WINAPI PFB_AckMasDiagEvent (DWORD CardHandle, BYTE MasBlk)
```

2.10.1.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)

2.10.1.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.1.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.2 PFB_AckMasError

2.10.2.1 Description

Acknowledges a master error so that the next master block error can be processed.

2.10.2.2 Prototype

```
BOOL WINAPI PFB_AckMasError (DWORD CardHandle, BYTE MasBlk)
```

2.10.2.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)

2.10.2.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.2.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.3 PFB_AckMasEvent

2.10.3.1 Description

Acknowledges a master event so that the next master block event can be processed.

2.10.3.2 Prototype

```
BOOL WINAPI PFB_AckMasEvent (DWORD CardHandle, BYTE MasBlk)
```

2.10.3.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)

2.10.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.3.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.4 PFB_AckMasGlbEvent

2.10.4.1 Description

Acknowledges a master global event so that the next master global event can be processed.

2.10.4.2 Prototype

```
BOOL WINAPI PFB_AckMasGlbEvent (DWORD CardHandle)
```

2.10.4.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.10.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.4.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.10.5 PFB_Config2bf

2.10.5.1 Description

Configures the card's master services using a binary configuration file exported from the SIEMENS COM PROFIBUS Configuration tool (*.2BF).

2.10.5.2 Prototype

```
BOOL WINAPI PFB_Config2bf (DWORD CardHandle, TCHAR* sz2bfFileName)
```

2.10.5.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
sz2bfFileName	2BF file name including its path

2.10.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.5.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 021d	Invalid file name pointer
2000 021e	Invalid configuration file
2000 0254	Taking card offline failed
2000 0255	Copy master configuration failed
2000 0256	Configuration 2bf shared RAM failed
2000 0258	Reading file error
2000 03xx	Status Error (refer to definitions related to pfbStatus in Section 4.1.1.1).

2.10.6 PFB_ConfigAbf

2.10.6.1 Description

Configures the card's master services using a binary configuration file exported from the SST PROFIBUS Configuration tool (*.BSS).

2.10.6.2 Prototype

```
BOOL WINAPI PFB_ConfigAbf (DWORD CardHandle, TCHAR* szBssFileName)
```

2.10.6.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
szBssFileName	BSS file name, including its path

2.10.6.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.6.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation), card must be open with SS_WRITE flag
2000 011a	Card access time-out
2000 021d	Invalid file name pointer
2000 021e	Invalid configuration file
2000 0254	Taking card off line failed
2000 0255	Copy master configuration failed
2000 0256	Configuration 2bf shared RAM failed
2000 0258	Reading file error
2000 03xx	Status Error (refer to definitions related to pfbStatus in Section 4.1.1.1).

2.10.7 PFB_GetMasCfg

2.10.7.1 Description

Gets the master block configuration parameters.

2.10.7.2 Prototype

```
BOOL WINAPI PFB_GetMasCfg (DWORD CardHandle, BYTE MasBlk, PFB_MAS_CFG* MasCfg).
```

For details, refer to [PFB_MAS_CFG](#), in Section 4.4.1.1.

2.10.7.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasCfg	Destination of Master block configuration parameters

2.10.7.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.7.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – MasCfg
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.8 PFB_GetMasCfgChk

2.10.8.1 Description

Gets the master configuration check data.

2.10.8.2 Prototype

BOOL WINAPI PFB_GetMasCfgChk (DWORD CardHandle, BYTE MasBlk, PFB_DP_CHK_CFG* MasChkCfg). For details, refer to [PFB_DP_CHK_CFG](#), in Section 4.4.2.1.

2.10.8.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasChkCfg	Destination of Master Configuration Check data

2.10.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.8.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – MasChkCfg
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.9 PFB_GetMasCntCfg

2.10.9.1 Description

Gets the master control configuration register.

2.10.9.2 Prototype

```
BOOL WINAPI PFB_GetMasCntCfg (DWORD CardHandle, BYTE MasBlk, BYTE* MasCntCfg)
```

2.10.9.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasCntCfg	Destination of Master control configuration register

2.10.9.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.9.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – MasCntCfg
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.10 PFB_GetMasDiagInfo

2.10.10.1 Description

Gets the master diagnostic parameters.

2.10.10.2 Prototype

```
BOOL WINAPI PFB_GetMasDiagInfo (DWORD CardHandle, BYTE MasBlk,  
PFB_DP_DIAG_INFO* MasDiag). For details, refer to PFB\_DP\_DIAG\_INFO,  
in Section 4.5.1.1.
```

2.10.10.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasDiag	Destination of Master Diagnostic parameters

2.10.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.10.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – MasDiag
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.11 PFB_GetMasGlbCfg

2.10.11.1 Description

Gets the master global configuration parameters.

2.10.11.2 Prototype

BOOL WINAPI PFB_GetMasGlbCfg (DWORD CardHandle, PFB_MAS_GLB_CFG* MasGlbCfg).
For details, refer to [PFB_MAS_GLB_CFG](#), in Section 4.3.1.1.

2.10.11.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasGlbCfg	Destination of Master Global Configuration parameters

2.10.11.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.11.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – MasGlbCfg
2000 011a	Card access timeout

2.10.12 PFB_GetMasGlbSts

2.10.12.1 Description

Gets the master global status parameters.

2.10.12.2 Prototype

```
BOOL WINAPI PFB_GetMasGlbSts (DWORD CardHandle, BYTE* MasGlbSts)
```

2.10.12.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasGlbSts	Destination of Master Global Status parameters

2.10.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.12.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – MasGlbSts
2000 011a	Card access timeout

2.10.13 PFB_GetMasPage

2.10.13.1 Description

Gets the master page number.

2.10.13.2 Prototype

```
BOOL WINAPI PFB_GetMasPage (DWORD CardHandle, BYTE* MasPage)
```

2.10.13.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasPage	Destination of Master page number

2.10.13.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.13.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - MasPage
2000 011a	Card access timeout

2.10.14 PFB_GetMasParmData

2.10.14.1 Description

Gets the master parameter data.

2.10.14.2 Prototype

```
BOOL WINAPI PFB_GetMasParmData (DWORD CardHandle, BYTE MasBlk,  
PFB_DP_PARM_DATA* MasParm). For details, refer to PFB\_DP\_PARM\_DATA,  
in Section 4.5.2.1.
```

2.10.14.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasParm	Destination of Master Diagnostic parameters

2.10.14.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.14.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - MasParm
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.15 PFB_GetMasRxData

2.10.15.1 Description

Gets the master receive data.

2.10.15.2 Prototype

```
BOOL WINAPI PFB_GetMasRxData (DWORD CardHandle, BYTE MasBlk, BYTE Ofs, BYTE  
Len, BYTE* DataBuf)
```

2.10.15.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
Ofs	Offset of the RX Data
Len	Length of the data to be read
DataBuf	Destination of Master RX Data

2.10.15.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.15.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – DataBuf
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number
2000 0234	Receive data not configured
2000 0236	Invalid relative offset (Ofs >= RxDataLen)
2000 0237	Invalid relative length (Len > RxDataLen - Ofs)

2.10.16 PFB_GetMasStatusTable

2.10.16.1 Description

Gets the master status table.

2.10.16.2 Prototype

```
BOOL WINAPI PFB_GetMasStatusTable (DWORD CardHandle, WORD* MasStsTab)
```

2.10.16.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasStsTab	Destination of Master Status Table

2.10.16.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.16.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – MasStsTab
2000 011a	Card access timeout

2.10.17 PFB_GetMasSts

2.10.17.1 Description

Gets the master status parameters.

2.10.17.2 Prototype

```
BOOL WINAPI PFB_GetMasSts (DWORD CardHandle, BYTE MasBlk, PFB_MAS_STS* MasSts).
```

For details, refer to [PFB_MAS_STS](#), in Section 4.3.2.1.

2.10.17.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasSts	Destination of Master Status parameters

2.10.17.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.17.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – MasSts
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number

2.10.18 PFB_GetMasTxData

2.10.18.1 Description

Gets the master transmit data.

2.10.18.2 Prototype

```
BOOL WINAPI PFB_GetMasTxData (DWORD CardHandle, BYTE MasBlk, BYTE Ofs, BYTE  
Len, BYTE* DataBuf)
```

2.10.18.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
Ofs	Offset of the TX Data
Len	Length of the data to be read
DataBuf	Destination of Master TX Data

2.10.18.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.18.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – DataBuf
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number
2000 0238	Invalid relative offset (Ofs >= TxDataLen)
2000 0239	Invalid relative length (Len >= TxDataLen - Ofs)
2000 023a	Transmit data not configured (TxDataLen = 0)

2.10.19 PFB_PutMasCfg

2.10.19.1 Description

Writes the master configuration block parameters into the card's Master Control page.

2.10.19.2 Prototype

```
BOOL WINAPI PFB_PutMasCfg (DWORD CardHandle, BYTE MasBlk, PFB_MAS_CFG* MasCfg).
```

For details, refer to [PFB_MAS_CFG](#), in Section 4.4.1.1.

2.10.19.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasCfg	Source of Master Configuration Block parameters

2.10.19.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.19.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number
2000 0229	Invalid Station Number (0-126)
2000 022a	Invalid Master Configuration Control register (refer to definitions related to masCntCfg , in Section 4.4.1.1).
2000 022b	Invalid Master RX length (0-244)
2000 022c	Invalid Master TX length (0-244)
2000 022d	Invalid RX offset (0-3FF8h - 8 byte boundary)
2000 022e	Invalid TX offset (0-3FF8h - 8 byte boundary)
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.10.20 PFB_PutMasCfgChk

2.10.20.1 Description

Writes the master configuration check data.

2.10.20.2 Prototype

```
BOOL WINAPI PFB_PutMasCfgChk (DWORD CardHandle, BYTE MasBlk, PFB_DP_CHK_CFG* MasChkCfg). For details, refer to PFB\_DP\_CHK\_CFG, in Section 4.4.2.1.
```

2.10.20.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasChkCfg	Source of Master Configuration check data

2.10.20.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.20.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number
2000 0233	Invalid Master Check length (0-30)
2000 0259	No more resources for extended check data available
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.10.21 PFB_PutMasCntCfg

2.10.21.1 Description

Writes the master control configuration register.

2.10.21.2 Prototype

```
BOOL WINAPI PFB_PutMasCntCfg (DWORD CardHandle, BYTE MasBlk, BYTE MasCntCfg)
```

2.10.21.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
MasCntCfg	Source of Master Control Configuration register

2.10.21.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.21.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number
2000 022a	Invalid Master Configuration Control register (refer to definitions related to masCntCfg , in Section 4.4.1.1).

2.10.22 PFB_PutMasGlbCfg

2.10.22.1 Description

Writes the master global configuration parameters.

2.10.22.2 Prototype

```
BOOL WINAPI PFB_PutMasGlbCfg (DWORD CardHandle, PFB_MAS_GLB_CFG* MasGlbCfg).
```

For details, refer to [PFB_MAS_GLB_CFG](#), in Section 4.3.1.1.

2.10.22.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasGlbCfg	Source of Master Global Configuration parameters

2.10.22.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.22.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0224	Master already configured
2000 0225	Invalid Global Master Control Configuration register (refer to definitions related to pfbmasCntrlCfg , in Section 4.3.1.1).
2000 0226	Invalid Min Cycle Time (2-255 in 100us)
2000 0227	Invalid Max Cycle Time (1-65535 in 10ms)
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.10.23 PFB_PutMasGlbCntrlCfg

2.10.23.1 Description

Writes the master global control configuration register. Sets various options for the DP Master.

2.10.23.2 Prototype

```
BOOL PFB_PutMasGlbCntrlCfg (DWORD CardHandle, BYTE MasGlbCntrlCfg)
```

2.10.23.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasGlbCntrlCfg	Source of master global configuration register (see bit definitions for that register in pfbMasCntrlCfg register of PFB_MAS_GLB_CFG structure)

2.10.23.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.23.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 011a	Card access timeout
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 0102	Invalid CardHandle
2000 0225	Invalid Global Master Control Configuration register (refer to definitions related to pfbmasCntrlCfg , in Section 4.3.1.1).

2.10.24 PFB_PutMasParmData

2.10.24.1 Description

Writes the master parameter data.

2.10.24.2 Prototype

```
BOOL WINAPI PFB_PutMasParmData (DWORD CardHandle, BYTE MasBlk,  
PFB_DP_PARM_DATA* MasParm). For details, refer to PFB\_DP\_PARM\_DATA,  
in Section 4.5.2.1.
```

2.10.24.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Number of a master block (0-124)
MasGlbCfg	Source of Master Parameter data

2.10.24.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.24.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number
2000 022f	Invalid Master Parameter length (0-23)
2000 0230	Invalid Master Watch Dog Factor 1 (1-255)
2000 0231	Invalid Master Watch Dog Factor 2 (1-255)
2000 0232	Invalid Master Ready Time (1-255)
2000 0259	No more resources for extended parameters available
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.10.25 PFB_PutMasTxData

2.10.25.1 Description

Writes the master transmit data.

2.10.25.2 Prototype

```
BOOL WINAPI PFB_PutMasTxData (DWORD CardHandle, BYTE MasBlk, BYTE ofs, BYTE  
Len, BYTE* DataBuf)
```

2.10.25.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
MasBlk	Master Control Block number (0-124)
Offset	Offset of the TX Data
Length	Length of the data to be written
DataBuf	Source of Master TX Data

2.10.25.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.10.25.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0228	Invalid Master Block Number
2000 0238	Invalid relative offset (Ofs >= TxDataLen)
2000 0239	Invalid relative length (Len >= TxDataLen - Ofs)

2.11 DP Slave API

2.11.1 PFB_AckSlvDiagEvent

2.11.1.1 Description

Acknowledges a slave diagnostic event so that the next slave diagnostic event can be processed.

2.11.1.2 Prototype

```
BOOL WINAPI PFB_AckSlvDiagEvent (DWORD CardHandle)
```

2.11.1.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.11.1.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.1.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.11.2 PFB_AckSlvError

2.11.2.1 Description

Acknowledges a slave error so that the next slave error can be processed.

2.11.2.2 Prototype

```
BOOL WINAPI PFB_AckSlvError (DWORD CardHandle)
```

2.11.2.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.11.2.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.2.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.11.3 PFB_AckSlvEvent

2.11.3.1 Description

Acknowledges a slave event so that the next slave event can be processed.

2.11.3.2 Prototype

```
BOOL WINAPI PFB_AckSlvEvent (DWORD CardHandle)
```

2.11.3.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.11.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.3.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.11.4 PFB_GetSlvCfg

2.11.4.1 Description

Gets the slave configuration table.

2.11.4.2 Prototype

```
BOOL WINAPI PFB_GetSlvCfg (DWORD CardHandle, PFB_SLV_CFG* SlvCfg).
```

For details, refer to [PFB_SLV_CFG](#), in Section 4.10.1.1.

2.11.4.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
SlvCfg	Destination of Slave Configuration Table

2.11.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.4.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - SlvCfg
2000 011a	Card access timeout

2.11.5 PFB_GetSlvCfgChk

2.11.5.1 Description

Gets slave configuration check data.

2.11.5.2 Prototype

```
BOOL WINAPI PFB_GetSlvCfgChk (DWORD CardHandle, PFB_DP_CHK_CFG* SlvChkCfg).
```

For details, refer to [PFB_DP_CHK_CFG](#), in Section 4.4.2.1.

2.11.5.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
SlvChkCfg	Destination of Slave Configuration Check parameters

2.11.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.5.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - SlvChkCfg
2000 011a	Card access timeout

2.11.6 PFB_GetSlvDiagInfo

2.11.6.1 Description

Gets the slave diagnostic parameters.

2.11.6.2 Prototype

```
BOOL WINAPI PFB_GetSlvDiagInfo (DWORD CardHandle, PFB_DP_DIAG_INFO* SlvDiag).  
For details, refer to PFB\_DP\_DIAG\_INFO, in Section 4.5.1.1.
```

2.11.6.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
SlvDiag	Destination of Slave Diagnostic parameters

2.11.6.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.6.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - SlvDiag
2000 011a	Card access timeout

2.11.7 PFB_GetSlvParmData

2.11.7.1 Description

Gets slave parameter's data.

2.11.7.2 Prototype

```
BOOL WINAPI PFB_GetSlvParmData (DWORD CardHandle, PFB_DP_PARM_DATA* SlvParm).  
For details, refer to PFB\_DP\_PARM\_DATA, in Section 4.5.2.1.
```

2.11.7.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
SlvParm	Destination of Slave Parameter data

2.11.7.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.7.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - SlvParm
2000 011a	Card access timeout

2.11.8 PFB_GetSlvRxData

2.11.8.1 Description

Gets the slave's received data.

2.11.8.2 Prototype

```
BOOL WINAPI PFB_GetSlvRxData (DWORD CardHandle, BYTE ofs, BYTE Len, BYTE*
DataBuf)
```

2.11.8.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Ofs	Offset within configured data (0-243)
Len	Length of data (1-244)
DataBuf	Destination of Slave received data

2.11.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.8.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DataBuf
2000 011a	Card access timeout
2000 0236	Invalid relative offset (Ofs >= RxDataLen)
2000 0237	Invalid relative length (Len > RxDataLen - Ofs)

2.11.9 PFB_GetSlvStatus

2.11.9.1 Description

Gets a slave's status parameters.

2.11.9.2 Prototype

```
BOOL WINAPI PFB_GetSlvStatus (DWORD CardHandle, PFB_SLV_STS* SlvSts).  
For details, refer to PFB\_SLV\_STS, in Section 4.11.1.1.
```

2.11.9.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
SlvSts	Destination of Slave Status parameters

2.11.9.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.9.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - SlvSts
2000 011a	Card access timeout

2.11.10 PFB_GetSlvTxData

2.11.10.1 Description

Gets a slave's transmitted data.

2.11.10.2 Prototype

```
BOOL WINAPI PFB_GetSlvTxData (DWORD CardHandle, BYTE ofs, BYTE Len, BYTE*
DataBuf)
```

2.11.10.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Ofs	Offset within configured data (0-243)
Len	Length of data (1-244)
DataBuf	Destination of Slave transmitted data

2.11.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.10.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DataBuf
2000 011a	Card access timeout
2000 0238	Invalid relative offset (Ofs >= TxDataLen)
2000 0239	Invalid relative length (Len >= TxDataLen - Ofs)

2.11.11 PFB_PutSlvCfg

2.11.11.1 Description

Writes the slave configuration parameters.

2.11.11.2 Prototype

```
BOOL WINAPI PFB_PutSlvCfg (DWORD CardHandle, PFB_SLV_CFG* SlvCfg).
```

For details, refer to [PFB_SLV_CFG](#), in Section 4.10.1.1.

2.11.11.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
SlvCfg	Source of Slave Configuration parameters

2.11.11.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.11.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0220	Invalid Control Configuration register (refer to definitions related to slvCntlCfg , in Section 4.10.1.1).
2000 0221	Invalid slave receive length (0-244)
2000 0222	Invalid slave transmit length (0-244)
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.11.12 PFB_PutSlvCntCfg

2.11.12.1 Description

Writes the slave control configuration register.

2.11.12.2 Prototype

```
BOOL PFB_PutSlvCntCfg (DWORD CardHandle, WORD SlvCntCfg)
```

2.11.12.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
SlvCntCfg	Source of slave control configuration register

2.11.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.12.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0220	Invalid Control Configuration register (refer to definitions related to slvCntlCfg , in Section 4.10.1.1).

2.11.13 PFB_PutSlvDiagInfo

2.11.13.1 Description

Writes the slave diagnostic parameters.

2.11.13.2 Prototype

```
BOOL WINAPI PFB_PutSlvDiagInfo (DWORD CardHandle, PFB_DP_DIAG_INFO* SlvDiag).  
For details, refer to PFB\_DP\_DIAG\_INFO, in Section 4.5.1.1.
```

2.11.13.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
SlvDiag	Destination of Slave Diagnostic parameters

2.11.13.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.13.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0223	Invalid slave diagnostic length (0-26)
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.11.14 PFB_PutSlvTxData

2.11.14.1 Description

Writes the slave transmit data.

2.11.14.2 Prototype

```
BOOL WINAPI PFB_PutSlvTxData (DWORD CardHandle, BYTE ofs, BYTE Len, BYTE*
DataBuf)
```

2.11.14.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Ofs	Offset within configured data (0-243)
Len	Length of data (1-244)
DataBuf	Source of Slave transmitted data

2.11.14.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.11.14.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0238	Invalid relative offset (Ofs >= TxDataLen)
2000 0239	Invalid relative length (Len >= TxDataLen - Ofs)

2.12 Layer 2 Message API

2.12.1 PFB_AckL2MsgError

2.12.1.1 Description

Acknowledges a Layer 2 message error so that the next message error can be processed.

2.12.1.2 Prototype

```
BOOL WINAPI PFB_AckL2MsgError (DWORD CardHandle, BYTE BlkNum)
```

2.12.1.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)

2.12.1.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.1.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)

2.12.2 PFB_AckL2MsgEvent

2.12.2.1 Description

Acknowledges a Layer 2 message event so that the next message event can be processed.

2.12.2.2 Prototype

```
BOOL WINAPI PFB_AckL2MsgEvent (DWORD CardHandle, BYTE BlkNum)
```

2.12.2.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)

2.12.2.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.2.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)

2.12.3 PFB_ActivateL2Msg

2.12.3.1 Description

Activates (triggers) a Layer 2 message. The message must be in the LAY2M_STE_DONE or the LAY2M_STE_IN_CONFIGURE state. Set the message state to LAY2M_STE_ENABLE.

2.12.3.2 Prototype

```
BOOL WINAPI PFB_ActivateL2Msg (DWORD CardHandle, BYTE BlkNum)
```

2.12.3.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)

2.12.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.3.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 0244	Invalid Message state (LAY2M_STE_DONE or LAY2M_STE_IN_CONFIGURE)

2.12.4 PFB_CloseL2MsgBlk

2.12.4.1 Description

Closes the previously opened message block.

2.12.4.2 Prototype

```
BOOL WINAPI PFB_CloseL2MsgBlk (DWORD CardHandle, BYTE BlkNum)
```

2.12.4.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)

2.12.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.4.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 0245	Message already closed
2000 0246	Message not processed yet

2.12.5 PFB_GetL2MsgCfg

2.12.5.1 Description

Gets a Layer 2 message control block.

2.12.5.2 Prototype

BOOL WINAPI PFB_GetL2MsgCfg (DWORD CardHandle, BYTE BlkNum, PFB_L2MSG_CFG* L2MsgCfg). For details, refer to [PFB L2MSG_CFG](#), in Section 4.6.1.1.

2.12.5.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgCfg	Destination of Layer 2 message block

2.12.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.5.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2MsgCfg
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)

2.12.6 PFB_GetL2MsgCntCfg

2.12.6.1 Description

Gets a Layer 2 message control configuration register.

2.12.6.2 Prototype

```
BOOL PFB_GetL2MsgCntCfg (DWORD CardHandle, BYTE BlkNum, BYTE* L2MsgCntCfg)
```

2.12.6.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgCntCfg	Destination of message control configuration register

2.12.6.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.6.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2MsgCntCfg
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)

2.12.7 PFB_GetL2MsgInfo

2.12.7.1 Description

Gets a Layer 2 message information block.

2.12.7.2 Prototype

```
BOOL WINAPI PFB_GetL2MsgInfo (DWORD CardHandle, BYTE BlkNum, PFB_L2MSG_TX*  
L2MsgTx). For details, refer to PFB L2MSG\_TX, in Section 4.7.1.1.
```

2.12.7.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgTx	Destination of Layer 2 message information block

2.12.7.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.7.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2MsgTx
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)

2.12.8 PFB_GetL2MsgMaxRxTxLen

2.12.8.1 Description

Gets a Layer 2 message maximum RX and TX length that's configured on the card.

2.12.8.2 Prototype

```
BOOL PFB_GetL2MsgMaxRxTxLen (DWORD CardHandle, BYTE BlkNum, BYTE* MaxRxLen,  
BYTE* MaxTxLen)
```

2.12.8.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
MaxRxLen	Destination of message max. RX length
MaxTxLen	Destination of message max. TX length

2.12.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.8.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - MaxRxLen and/or MaxTxLen
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)

2.12.9 PFB_GetL2MsgReply

2.12.9.1 Description

Gets a Layer 2 message reply.

2.12.9.2 Prototype

```
BOOL WINAPI PFB_GetL2MsgReply (DWORD CardHandle, BYTE BlkNum, PFB_L2MSG_STS* L2MsgSts, BYTE* RxData, DWORD TimeOut). For details, refer to PFB\_L2MSG\_STS, in Section 4.6.2.1.
```

2.12.9.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgSts	Destination of Layer 2 message status parameters
RxData	Destination of data to be read
TimeOut	Time allowed for message reply

2.12.9.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.9.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2MsgSts and/or RxData
2000 011a	Card access timeout
2000 0257	Message replay timeout
2000 023b	Invalid Message block number (0-127)

2.12.10 PFB_GetL2MsgRxData

2.12.10.1 Description

Gets Layer 2 message RX data.

2.12.10.2 Prototype

```
BOOL WINAPI PFB_GetL2MsgRxData (DWORD CardHandle, BYTE BlkNum, BYTE Ofs, BYTE  
Len, BYTE* DataBuf)
```

2.12.10.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
Ofs	Offset of the RX Data
Len	Length of the data to be read
DataBuf	Destination of Message RX Data

2.12.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.10.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DataBuf
2000 011a	Card access timeout
2000 0235	Receive data not available
2000 0236	Invalid relative offset (Ofs >= RxDataLen)
2000 0237	Invalid relative length (Len > RxDataLen - Ofs)
2000 023b	Invalid Message block number (0-127)

2.12.11 PFB_GetL2MsgState

2.12.11.1 Description

Gets a Layer 2 message state.

2.12.11.2 Prototype

```
BOOL WINAPI PFB_GetL2MsgState (DWORD CardHandle, BYTE BlkNum, BYTE* L2MsgState)
```

2.12.11.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgState	Destination of Layer 2 message state

2.12.11.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.11.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2MsgState
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)

2.12.12 PFB_GetL2MsgSts

2.12.12.1 Description

Gets Layer 2 message status parameters.

2.12.12.2 Prototype

BOOL WINAPI PFB_GetL2MsgSts (DWORD CardHandle, BYTE BlkNum, PFB_L2MSG_STS* L2MsgSts). For details, refer to [PFB L2MSG_STS](#), in Section 4.6.2.1.

2.12.12.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgSts	Destination of Layer 2 message status parameters

2.12.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.12.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2MsgSts
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)

2.12.13 PFB_GetL2MsgTxData

2.12.13.1 Description

Gets Layer 2 message TX data.

2.12.13.2 Prototype

```
BOOL WINAPI PFB_GetL2MsgTxData (DWORD CardHandle, BYTE BlkNum, BYTE ofs, BYTE  
Len, BYTE* DataBuf)
```

2.12.13.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
Ofs	Offset of the TX Data
Len	Length of the data to be read
DataBuf	Destination of Message TX Data

2.12.13.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.13.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DataBuf
2000 011a	Card access timeout
2000 0238	Invalid relative offset (Ofs >= TxDataLen)
2000 0239	Invalid relative length (Len >= TxDataLen - Ofs)
2000 023a	Transmit data not configured (TxDataLen = 0)
2000 023b	Invalid Message block number (0-127)

2.12.14 PFB_InitL2MsgBlk

2.12.14.1 Description

Initializes a Layer 2 message control block.

2.12.14.2 Prototype

```
BOOL WINAPI PFB_InitL2MsgBlk (DWORD CardHandle, BYTE BlkNum, BYTE MaxRxLen,  
BYTE MaxTxLen)
```

2.12.14.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
MaxRxLen	Maximum length of received data
MaxTxLen	Maximum length of transmit data

2.12.14.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.14.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 023c	Invalid Message Receive Length (0-244)
2000 023d	Invalid Message Transmit Length (0-244)
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.12.15 PFB_InitL2MsgGlb

2.12.15.1 Description

Initializes the Layer 2 message global control register. This function enables Message Services on the card and updates the Message Disable LED bit.

2.12.15.2 Prototype

```
BOOL WINAPI PFB_InitL2MsgGlb (DWORD CardHandle, BOOL DisLed)
```

2.12.15.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
DisLed	TRUE - disable LED, FALSE - enable LED

2.12.15.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.15.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.12.16 PFB_OpenL2MsgBlk

2.12.16.1 Description

Opens a Layer 2 message control block.

2.12.16.2 Prototype

```
BOOL WINAPI PFB_OpenL2MsgBlk (DWORD CardHandle, BYTE BlkNum)
```

2.12.16.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)

2.12.16.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.16.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 0242	Message Block busy
2000 0243	Message Block used before

2.12.17 PFB_PutL2MsgCfg

2.12.17.1 Description

Writes a Layer 2 Message Configuration block.

2.12.17.2 Prototype

BOOL WINAPI PFB_PutL2MsgCfg (DWORD CardHandle, BYTE BlkNum, PFB_L2MSG_CFG* L2MsgCfg). For details, refer to [PFB_L2MSG_CFG](#), in Section 4.6.1.1.

2.12.17.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgCfg	Source of Layer 2 Message Control Block Data

2.12.17.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.17.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 023e	Invalid Message Control Configuration register (refer to definitions related to lay2mCntCfg , in Section 4.6.1.1).
2000 023f	Invalid Update Time (periodic updtme * 1ms (1-16380))
2000 0240	Invalid Error Time (retry per msg. on error after this time * 1ms (1-16380))

2.12.18 PFB_PutL2MsgCntCfg

2.12.18.1 Description

Writes a Layer 2 Message control configuration register.

2.12.18.2 Prototype

```
BOOL PFB_PutL2MsgCntCfg (DWORD CardHandle, BYTE BlkNum, BYTE L2MsgCntCfg)
```

2.12.18.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgCntCfg	Source of Layer 2 Message control configuration register

2.12.18.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.18.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 023e	Invalid Message Control Configuration register (refer to definitions related to lay2mCntCfg , in Section 4.6.1.1).

2.12.19 PFB_PutL2MsgInfo

2.12.19.1 Description

Writes the Layer 2 message information (L2MsgTx) structure on the card. The message must be in the LAY2M_STE_DONE or LAY2M_STE_IN_CONFIGURE state. Set the state of the message to LAY2M_STE_ENABLE.

2.12.19.2 Prototype

```
BOOL WINAPI PFB_PutL2MsgInfo (DWORD CardHandle, BYTE BlkNum, PFB_L2MSG_TX*  
L2MsgTx). For details, refer to PFB L2MSG\_TX, in Section 4.7.1.1.
```

2.12.19.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgTx	Source of Layer 2 message TX parameters

2.12.19.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.19.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 0244	Invalid Message state (LAY2M_STE_DONE or LAY2M_STE_IN_CONFIGURE)
2000 0247	Invalid Destination Station (0-126)
2000 0248	Invalid Destination SAP (0-63, 255-disable)
2000 0249	Invalid Source SAP (0-63, 255-disable)
2000 024a	Invalid Frame Control (refer to definitions related to lay2mFrmCntrl , in Section 4.7.1.1).
2000 024b	Invalid Message Transmit Length (0-244)

2.12.20 PFB_PutL2MsgTxData

2.12.20.1 Description

Writes the Layer 2 message transmit data.

2.12.20.2 Prototype

```
BOOL WINAPI PFB_PutL2MsgTxData (DWORD CardHandle, BYTE BlkNum, BYTE Ofs, BYTE  
Len, BYTE* DataBuf)
```

2.12.20.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
Ofs	Offset of the TX Data
Len	Length of the data to be written
DataBuf	Source of Message TX Data

2.12.20.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.20.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0238	Invalid relative offset (Ofs >= TxDataLen)
2000 0239	Invalid relative length (Len >= TxDataLen - Ofs)
2000 023b	Invalid Message block number (0-127)

2.12.21 PFB_SendL2Msg

2.12.21.1 Description

Sends a Layer 2 message.

2.12.21.2 Prototype

BOOL WINAPI PFB_SendL2Msg (DWORD CardHandle, BYTE BlkNum, PFB_L2MSG_TX* L2MsgTx, BYTE* TxData). For details, refer to [PFB_L2MSG_TX](#), in Section 4.7.1.1.

2.12.21.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgTx	Source of Layer 2 message TX parameters
TxData	Source of Data to be transmitted

2.12.21.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.21.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 0244	Invalid Message state (LAY2M_STE_DONE or LAY2M_STE_IN_CONFIGURE)
2000 0247	Invalid Destination Station (0-126)
2000 0248	Invalid Destination SAP (0-63, 255-disable)
2000 0249	Invalid Source SAP (0-63, 255-disable)
2000 024a	Invalid Frame Control (refer to definitions related to lay2mFrmCntrl , in Section 4.7.1.1).
2000 024b	Invalid Message Transmit Length (0-244)

2.12.22 PFB_SendL2MsgWaitForReply

2.12.22.1 Description

Sends a Layer 2 message and waits for a reply.

2.12.22.2 Prototype

```
BOOL WINAPI PFB_SendL2MsgWaitForReply (DWORD CardHandle, BYTE BlkNum,  
PFB_L2MSG_TX* L2MsgTx, BYTE* TxData, PFB_L2MSG_STS* L2MsgSts, BYTE* RxData,  
DWORD TimeOut). For details, refer to PFB L2MSG TX in Section 4.7.1.1,  
and PFB L2MSG STS, in Section 4.6.2.1.
```

2.12.22.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgTx	Source of Layer 2 message TX parameters
TxData	Source of Data to be transmitted
L2MsgSts	Destination of Layer 2 message status
RxData	Destination of received data
TimeOut	Time allowed for message reply

2.12.22.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.22.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 0114	Invalid pointer - L2MsgSts and/or RxData
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 0244	Invalid Message state (LAY2M_STE_DONE or LAY2M_STE_IN_CONFIGURE)
2000 0247	Invalid Destination Station (0-126)
2000 0248	Invalid Destination SAP (0-63, 255-disable)
2000 0249	Invalid Source SAP (0-63, 255-disable)
2000 024a	Invalid Frame Control (refer to definitions related to lay2mFrmCntrl , in Section 4.7.1.1).
2000 024b	Invalid Message Transmit Length (0-244)
2000 0257	Message replay timeout

2.12.23 PFB_TrigL2Msg

2.12.23.1 Description

Triggers a Layer 2 message and writes the PFB_L2MSG_TX structure contents to the card.

2.12.23.2 Prototype

```
BOOL WINAPI PFB_TrigL2Msg (DWORD CardHandle, BYTE BlkNum, PFB_L2MSG_TX*  
L2MsgTx). For details, refer to PFB\_L2MSG\_TX, in Section 4.7.1.1.
```

2.12.23.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	Message block number (0-127)
L2MsgTx	Source of Layer 2 message TX parameters

2.12.23.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.12.23.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 023b	Invalid Message block number (0-127)
2000 0244	Invalid Message state (LAY2M_STE_DONE or LAY2M_STE_IN_CONFIGURE)
2000 0247	Invalid Destination Station (0-126)
2000 0248	Invalid Destination SAP (0-63, 255-disable)
2000 0249	Invalid Source SAP (0-63, 255-disable)
2000 024a	Invalid Frame Control (refer to definitions related to lay2mFrmCntrl , in Section 4.7.1.1).
2000 024b	Invalid Message Transmit Length (0-244)

2.13 Layer 2 SAP API

2.13.1 PFB_AckL2SapError

2.13.1.1 Description

Acknowledges a Layer 2 SAP error so that the next SAP error can be processed.

2.13.1.2 Prototype

```
BOOL WINAPI PFB_AckL2SapError (DWORD CardHandle, BYTE BlkNum)
```

2.13.1.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)

2.13.1.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.1.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)

2.13.2 PFB_AckL2SapEvent

2.13.2.1 Description

Acknowledges a Layer 2 SAP event so that the next SAP event can be processed.

2.13.2.2 Prototype

```
BOOL WINAPI PFB_AckL2SapEvent (DWORD CardHandle, BYTE BlkNum)
```

2.13.2.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)

2.13.2.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.2.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)

2.13.3 PFB_GetL2SapCfg

2.13.3.1 Description

Gets a SAP control block for Layer 2.

2.13.3.2 Prototype

BOOL WINAPI PFB_GetL2SapCfg (DWORD CardHandle, BYTE BlkNum, PFB_L2SAP_CFG* L2SapCfg). For details, refer to [PFB L2SAP_CFG](#), in Section 4.8.1.1.

2.13.3.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
L2SapCfg	Destination of Layer 2 SAP block

2.13.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.3.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2SapCfg
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)

2.13.4 PFB_GetL2SapCntCfg

2.13.4.1 Description

Gets a Layer 2 SAP control configuration register.

2.13.4.2 Prototype

```
BOOL PFB_GetL2SapCntCfg (DWORD CardHandle, BYTE BlkNum, WORD* L2SapCntCfg)
```

2.13.4.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
L2SapCntCfg	Destination of SAP control configuration register

2.13.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.4.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2SapCntCfg
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)

2.13.5 PFB_GetL2SapMaxRxTxLen

2.13.5.1 Description

Gets the Layer 2 SAP maximum RX and TX length configured.

2.13.5.2 Prototype

```
BOOL PFB_GetL2SapMaxRxTxLen (DWORD CardHandle, BYTE BlkNum, BYTE* MaxRxLen,  
BYTE* MaxTxLen)
```

2.13.5.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
MaxRxLen	Destination of SAP max. RX length
MaxTxLen	Destination of SAP max. TX length

2.13.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.5.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - MaxRxLen and/or MaxTxLen
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)

2.13.6 PFB_GetL2SapRequest

2.13.6.1 Description

Gets the Layer 2 SAP request data.

2.13.6.2 Prototype

```
BOOL WINAPI PFB_GetL2SapRequest (DWORD CardHandle, BYTE BlkNum, PFB_L2SAP_STS* L2SapSts, BYTE* RxData). For details, refer to PFB L2SAP STS, in Section 4.9.1.1.
```

2.13.6.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
L2SapSts	Destination of Layer 2 SAP status
RxData	Destination of received data

2.13.6.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.6.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2SapSts and/or RxData
2000 011a	Card access timeout
2000 021f	Memory Page busy
2000 0235	Receive data not available
2000 0241	Invalid SAP block number (0-63)

2.13.7 PFB_GetL2SapRxData

2.13.7.1 Description

Gets the Layer 2 SAP receive data.

2.13.7.2 Prototype

```
BOOL WINAPI PFB_GetL2SapRxData (DWORD CardHandle, BYTE BlkNum, BYTE Ofs, BYTE  
Len, BYTE* DataBuf)
```

2.13.7.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
Ofs	Offset of the RX Data
Len	Length of the data to be read
DataBuf	Destination of SAP RX Data

2.13.7.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.7.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DataBuf
2000 011a	Card access timeout
2000 0235	Receive data not available
2000 0236	Invalid relative offset (Ofs >= RxDataLen)
2000 0237	Invalid relative length (Len > RxDataLen - Ofs)
2000 0241	Invalid SAP block number (0-63)

2.13.8 PFB_GetL2SapSts

2.13.8.1 Description

Gets the SAP status parameters for Layer 2.

2.13.8.2 Prototype

BOOL WINAPI PFB_GetL2SapSts (DWORD CardHandle, BYTE BlkNum, PFB_L2SAP_STS* L2SapSts). For details, refer to [PFB L2SAP_STS](#), in Section 4.9.1.1.

2.13.8.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
L2SapSts	Destination of Layer 2 SAP status

2.13.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.8.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2SapSts
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)

2.13.9 PFB_GetL2SapTxData

2.13.9.1 Description

Gets the Layer 2 SAP transmit data.

2.13.9.2 Prototype

```
BOOL WINAPI PFB_GetL2SapTxData (DWORD CardHandle, BYTE BlkNum, BYTE Ofs, BYTE  
Len, BYTE* DataBuf)
```

2.13.9.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
Ofs	Offset of the TX Data
Len	Length of the data to be read
DataBuf	Destination of SAP TX Data

2.13.9.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.9.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DataBuf
2000 011a	Card access timeout
2000 0238	Invalid relative offset (Ofs >= TxDataLen)
2000 0239	Invalid relative length (Len >= TxDataLen - Ofs)
2000 023a	Transmit data not configured (TxDataLen = 0)
2000 0241	Invalid SAP block number (0-63)

2.13.10 PFB_GetL2SapType

2.13.10.1 Description

Gets the Layer 2 SAP type.

2.13.10.2 Prototype

```
BOOL WINAPI PFB_GetL2SapType (DWORD CardHandle, BYTE BlkNum, BYTE* L2SapType)
```

2.13.10.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
L2SapType	Destination of Layer 2 SAP type

2.13.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.10.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - L2SapType
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)

2.13.11 PFB_InitL2SapBlk

2.13.11.1 Description

Initializes the Layer 2 SAP control block.

2.13.11.2 Prototype

```
BOOL WINAPI PFB_InitL2SapBlk (DWORD CardHandle, BYTE BlkNum, BYTE MaxRxLen,  
BYTE MaxTxLen)
```

2.13.11.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
MaxRxLen	Maximum length of received data
MaxTxLen	Maximum length of transmit data

2.13.11.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.11.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)
2000 024c	Invalid SAP Receive Length (0-244)
2000 024d	Invalid SAP Transmit Length (0-244)
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.13.12 PFB_InitL2SapGlb

2.13.12.1 Description

Initializes the Layer 2 SAP global control register. This function enables SAP Services on the card and updates the SAP Disable LED bit.

2.13.12.2 Prototype

```
BOOL WINAPI PFB_InitL2SapGlb (DWORD CardHandle, BOOL DisLed)
```

2.13.12.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
DisLed	TRUE - disable LED, FALSE - enable LED

2.13.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.12.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.13.13 PFB_PutL2SapCfg

2.13.13.1 Description

Writes the Layer 2 SAP configuration parameters.

2.13.13.2 Prototype

BOOL WINAPI PFB_PutL2SapCfg (DWORD CardHandle, BYTE BlkNum, PFB_L2SAP_CFG* L2SapCfg). For details, refer to [PFB L2SAP_CFG](#), in Section 4.8.1.1.

2.13.13.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
L2SapCfg	Source of Layer 2 SAP Control Block Data

2.13.13.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.13.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)
2000 024e	Invalid SAP Station (0-126)
2000 024f	Invalid SAP Number (0-63)
2000 0250	Invalid SAP Control Configuration register (refer to definitions related to lay2sCntCfg , in Section 4.8.1.1).
2000 0251	Invalid SAP Time-out Time * 10ms (1-8190, 0=disable)
2000 0252	Invalid SAP Frame Control Value (refer to definitions related to lay2sFrmCntrl , in Section 4.8.1.1).
2000 04xx	Not Offline state - cannot configure (xx contains current command)

2.13.14 PFB_PutL2SapCntCfg

2.13.14.1 Description

Writes the Layer 2 SAP control configuration register.

2.13.14.2 Prototype

```
BOOL PFB_PutL2SapCntCfg (DWORD CardHandle, BYTE BlkNum, BYTE L2SapCntCfg)
```

2.13.14.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
L2SapCntCfg	Source of Layer 2 SAP control configuration register

2.13.14.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.14.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)
2000 0250	Invalid SAP Control Configuration register (refer to definitions related to lay2sCntCfg , in Section 4.8.1.1).

2.13.15 PFB_PutL2SapResponse

2.13.15.1 Description

Writes the Layer 2 SAP response data.

2.13.15.2 Prototype

```
BOOL WINAPI PFB_PutL2SapResponse (DWORD CardHandle, BYTE BlkNum, BYTE TxLen,  
BYTE* TxData)
```

2.13.15.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
TxLen	Length of data to transmit
L2SapCfg	Source of SAP transmit data

2.13.15.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.15.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0241	Invalid SAP block number (0-63)
2000 0253	Invalid SAP Response Transmit Length (TX length must be less or equal to configured length)

2.13.16 PFB_PutL2SapTxData

2.13.16.1 Description

Writes the Layer 2 SAP transmit data.

2.13.16.2 Prototype

```
BOOL WINAPI PFB_PutL2SapTxData (DWORD CardHandle, BYTE BlkNum, BYTE Ofs, BYTE  
Len, BYTE* DataBuf)
```

2.13.16.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
BlkNum	SAP block number (0-63)
Ofs	Offset of the TX Data
Len	Length of the data to be written
DataBuf	Source of SAP TX Data

2.13.16.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.13.16.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout
2000 0238	Invalid relative offset (Ofs >= TxDataLen)
2000 0239	Invalid relative length (Len >= TxDataLen - Ofs)
2000 0241	Invalid SAP block number (0-63)

2.14 DP Master Class II API

2.14.1 PFB_AckMC2MasterError

2.14.1.1 Description

Acknowledges a Master Class II to Master Class I error so that the next error can be processed.

2.14.1.2 Prototype

```
BOOL WINAPI PFB_AckMC2MasterError (DWORD CardHandle)
```

2.14.1.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.14.1.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.1.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.2 PFB_AckMC2MasterEvent

2.14.2.1 Description

Acknowledges a Master Class II event so that the next event can be processed.

2.14.2.2 Prototype

```
BOOL WINAPI PFB_AckMC2MasterEvent (DWORD CardHandle)
```

2.14.2.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.14.2.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.2.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.3 PFB_AckMC2SlaveError

2.14.3.1 Description

Acknowledges a Master Class II event to a slave error so that the next error can be processed.

2.14.3.2 Prototype

```
BOOL WINAPI PFB_AckMC2SlaveError (DWORD CardHandle)
```

2.14.3.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.14.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.3.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.4 PFB_AckMC2SlaveEvent

2.14.4.1 Description

Acknowledges a Master Class II slave event so that the next event can be processed.

2.14.4.2 Prototype

```
BOOL WINAPI PFB_AckMC2SlaveEvent (DWORD CardHandle)
```

2.14.4.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

2.14.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.4.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.5 PFB_GetMC2MasterBlock

2.14.5.1 Description

Reads the PFB_MC2_MASTER_BLK structure from the card host interface memory.

2.14.5.2 Prototype

```
BOOL PFB_GetMC2MasterBlock (DWORD CardHandle, PFB_MC2_MASTER_BLK  
*Mc2MasterBlk). For details, refer to PFB\_MC2\_MASTER\_BLK, in  
Section 4.16.1.1.
```

2.14.5.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Mc2MasterBlk	Pointer to the structure that will receive the current block information

2.14.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.5.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.6 PFB_GetMC2MasterCfg

2.14.6.1 Description

Reads the PFB_MC2_MASTER_CFG structure from the card host interface memory.

2.14.6.2 Prototype

```
BOOL PFB_GetMC2MasterCfg (DWORD CardHandle, PFB_MC2_MASTER_CFG *Mc2MasterCfg),  
BYTE *GlbDPMsgBlk). For details, refer to PFB\_MC2\_MASTER\_CFG, in Section  
4.14.1.1.
```

2.14.6.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Mc2MasterCfg	Pointer to the structure that will receive the current configuration

2.14.6.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.6.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.7 PFB_GetMC2MasterRxData

2.14.7.1 Description

Reads the result of a Master Class II to Master Class I transaction.

2.14.7.2 Prototype

```
BOOL WINAPI PFB_GetMC2MasterRxData (DWORD CardHandle, PFB_MC2_MASTER_RD
*Mc2MasterRd, BYTE *RxData). For details, refer to PFB\_MC2\_MASTER\_RD,
in Section 4.15.1.1.
```

2.14.7.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Mc2MasterRd	Pointer to the structure that will hold the status information from the last Master Class II transaction
RxData	Pointer to the buffer that will hold the returned data, if any, from the last Master Class II transaction

2.14.7.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.7.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.8 PFB_GetMC2SlaveCfg

2.14.8.1 Description

Reads the PFB_MC2_SLV_CFG structure from the card host interface memory.

2.14.8.2 Prototype

```
BOOL PFB_GetMC2SlaveCfg (DWORD CardHandle, PFB_MC2_SLV_CFG *Mc2SlaveCfg, BYTE *GlbDPMsgBlk). For details, refer to PFB MC2 SLV CFG, in Section 4.12.1.1.
```

2.14.8.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Mc2SlaveCfg	Pointer to the structure that will receive the current configuration

2.14.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.8.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 011a	Card access timeout

2.14.9 PFB_GetMC2SlaveRd

2.14.9.1 Description

Reads the PFB_MC2_SLV_RD structure for status information and the data received from the card host interface memory.

2.14.9.2 Prototype

```
BOOL PFB_GetMC2SlaveRd (DWORD CardHandle, BYTE DstStn, BYTE RdArea,  
PFB_MC2_SLV_RD *Mc2SlaveRd, BYTE *RxData). For details, refer to  
PFB\_MC2\_SLV\_RD, in Section 4.13.1.1.
```

2.14.9.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
DstStn	The slave station address
RdArea	The area to read from the slave. For example, input table, output table and configuration.
Mc2SlaveRd	Pointer to the structure that will receive the current status of the last read
RxData	Pointer to the buffer that will receive any returned data

2.14.9.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.9.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.10 PFB_PutMC2MasterCfg

2.14.10.1 Description

Writes the PFB_MC2_MASTER_CFG structure to the card host interface memory.

2.14.10.2 Prototype

```
BOOL PFB_PutMC2MasterCfg (DWORD CardHandle, PFB_MC2_MASTER_CFG *Mc2MasterCfg).
```

For details, refer to [PFB_MC2_MASTER_CFG](#), in Section 4.14.1.1.

2.14.10.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Mc2MasterCfg	Pointer to the structure that contains configuration information.

2.14.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.10.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.11 PFB_PutMC2MasterTxData

2.14.11.1 Description

Writes the information for a Master Class II to Master Class I transaction.

2.14.11.2 Prototype

```
BOOL WINAPI PFB_PutMC2MasterTxData (DWORD CardHandle, BYTE DstStn,  
PFB_MC2_MASTER_WRT *Mc2MasterWrt, BYTE *TxData). For details, refer to  
PFB\_MC2\_MASTER\_WRT, in Section 4.16.2.1.
```

2.14.11.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
DstStn	The station address of the target Master
Mc2MasterWrt	Pointer to the structure that holds the information for the current Master Class II transaction
TxData	Pointer to the buffer that holds the command data, if any, for the current Master Class II transaction

2.14.11.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.11.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

2.14.12 PFB_PutMC2SlaveCfg

2.14.12.1 Description

Writes the PFB_MC2_SLV_CFG structure to the card host interface memory.

2.14.12.2 Prototype

```
BOOL PFB_PutMC2SlaveCfg (DWORD CardHandle, PFB_MC2_SLV_CFG *Mc2SlaveCfg).
```

For details, refer to [PFB_MC2_SLV_CFG](#), in Section 4.12.1.1.

2.14.12.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Mc2SlaveCfg	Pointer to the structure that contains configuration information.

2.14.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.12.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 010e	Invalid offset
2000 011a	Card access timeout

2.14.13 PFB_SetMC2SlaveAddr

2.14.13.1 Description

Writes the PFB_MC2_SETSLV_ADDR structure to the card host interface memory. This will send an SSA message to a slave and if the slave supports this service, it will change the slave's station address.

2.14.13.2 Prototype

```
BOOL PFB_SetMC2SlaveAddr (DWORD CardHandle, BYTE DstStn, BYTE RdArea,  
PFB_MC2_SETSLV_ADDR *Mc2SetSlaveAddr, BYTE *TxData). For details, refer  
to PFB MC2 SETSLV ADDR, in Section 4.13.2.1.
```

2.14.13.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
DstStn	The current slave station address
RdArea	The area to write to the slave. This should always be the Set_Slave_Addr Sap (55).
Mc2SetSlaveAddr	Pointer to the structure that receives the result of the set slave address message.
TxData	Pointer to the buffer that contains the set slave address message (see DIN 19 245, part 3).

2.14.13.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

2.14.13.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied (sharing violation). Card must be opened with the SS_WRITE flag.
2000 011a	Card access timeout

3

PFBMAN32 DLL DPV1 API

Chapter Sections:

- This chapter defines the API (*Application Programming Interface*) for the PFBMAN32 DLL's DPV1

3.1 Introduction

This chapter defines the API (*Application Programming Interface*) for the PFBMAN32 DLL's DPV1.

3.1.1 DPV1 API Features

The DPV1 API's features include:

- Master Class1 & Class2
- Mono-thread DLL

3.2 Configuration

The SST Profibus Configuration Tool for Windows (PbCfg.exe) can enable DPV1 Master Class 1 services for supporting slaves through the “DPV1” device configuration page.

3.3 API

The API provides:

- Functions for DLL initialization
- Functions for DPV1 services

3.3.1 Initialization

Before DPV1 functions can be used, DPV1_Init must be called, in CONFIGURED/OFFLINE mode. This function allocates resources for FDL blocks. DPV1_Exit releases these resources.

3.3.2 DPV1 Services

For all DPV1 services, the API provides 2 asynchronous functions (except MSAC2_Abort):

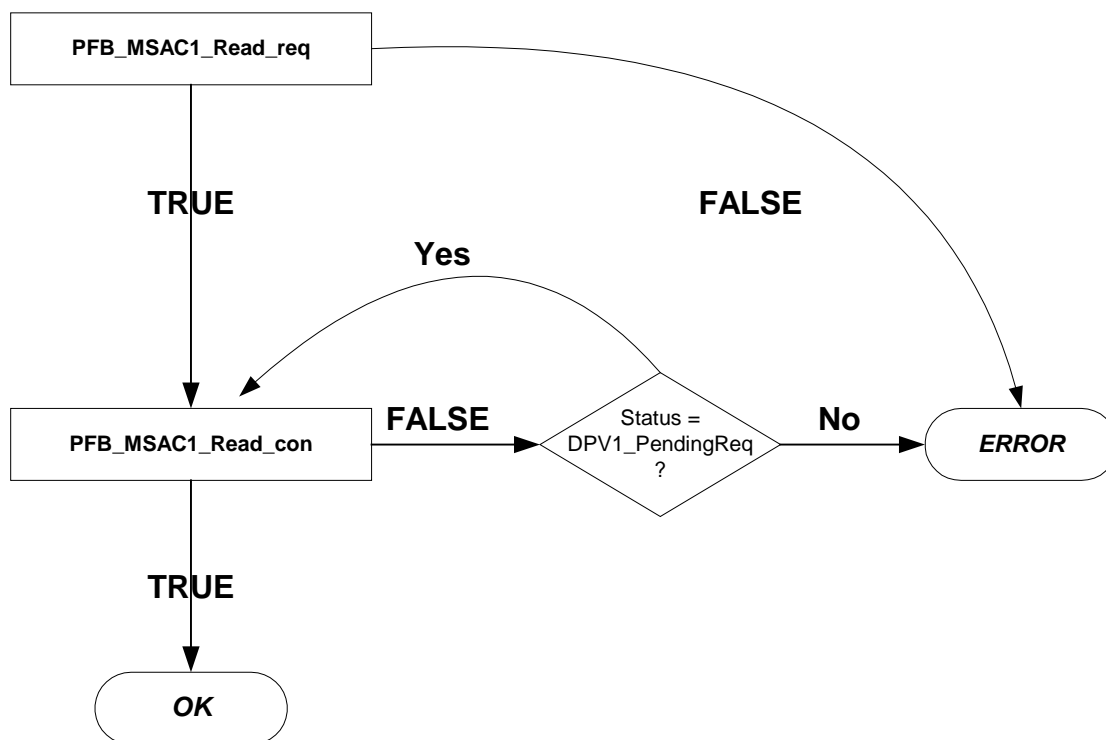
- 1 function to send the request
- 1 function to request the confirmation

	DPV1 Services	API
DPV1 Class 1	MSAC1_Read	MSAC1_Read_req MSAC1_Read_con
	MSAC1_Write	MSAC1_Write_req MSAC1_Write_con
DPV1 Class 2	MSAC2_Initiate	MSAC2_Initiate_req MSAC2_Initiate_con
	MSAC2_Read	MSAC2_Read_req MSAC2_Read_con
	MSAC2_Write	MSAC2_Write_req MSAC2_Write_con
	MSAC2_Idle	MSAC2_Idle_req MSAC2_Idle_con
	MSAC2_Abort	MSAC2_Abort_req

3.3.2.1 Example of Read Class 1 Function

The application uses *PFB_MSAC1_Read_req()* to send the request to the device, and *PFB_MSAC1_Read_con()* to receive the confirmation while the status is *DPV1_PendingReq*.

Figure 1: Example of Read Class 1 Function

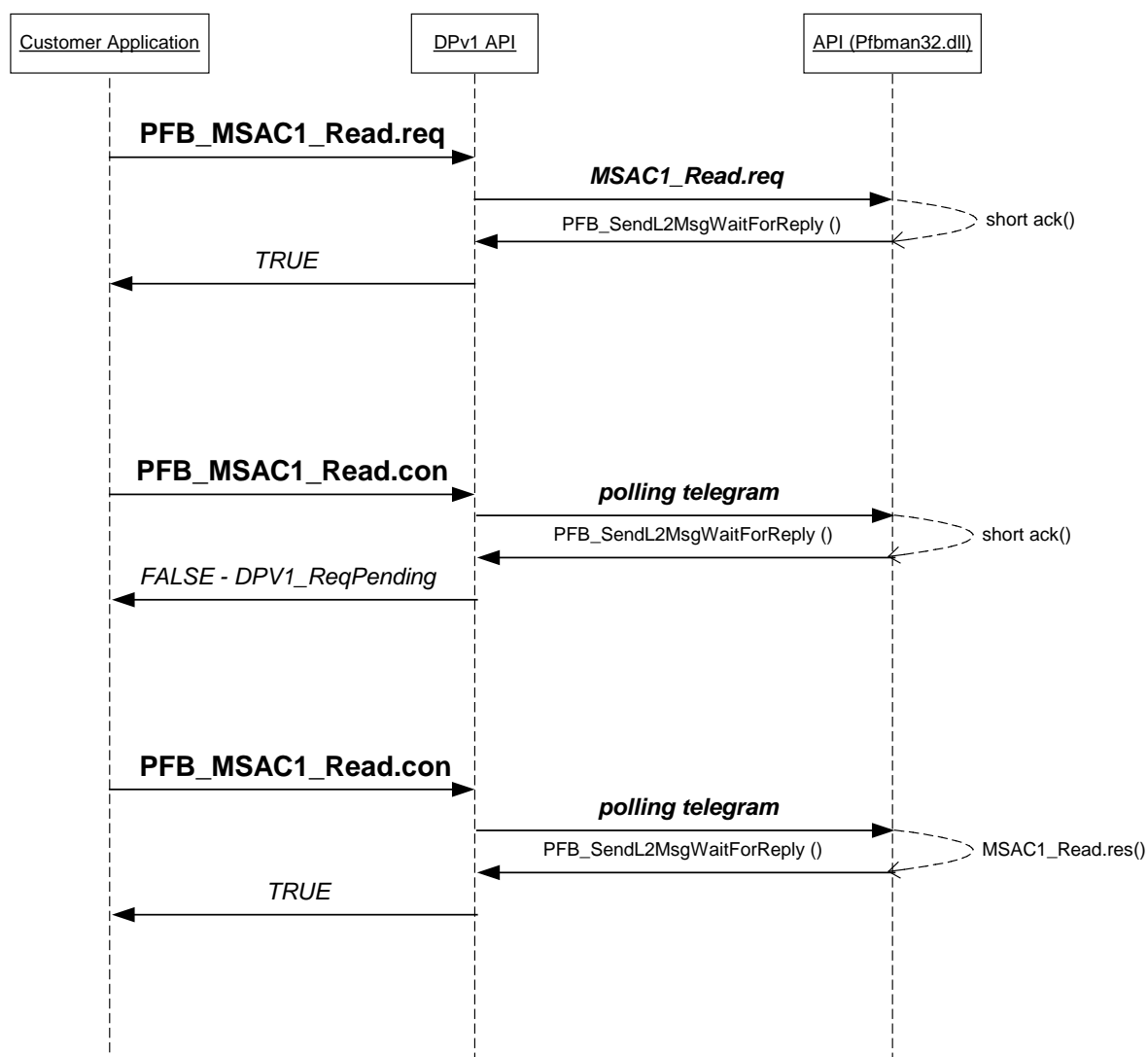


How it Works

All functions use *PFB_SendL2MsgWaitForReply()*

- The request function (*_req*) sends the DPV1 frame to the device
- The confirmation function (*_con*) sends a polling telegram to the device

Figure 2: Example of DPV1 Master Class 1 Read Services Typical Sequence



How to Use DPV1 Class 2:

- Initiate a connection with **PFB_MSAC2_Initiate_req** and **PFB_MSAC2_Initiate_con**
- Use DPV1 class 2 functions. If you don't use these functions during **Send_TimeOut** time, the slave will close the connection. The **MSAC2_Idle** could be used to keep this connection alive.
- Close the connection with **PFB_MSAC2_Abort_req** function

3.3.3 DPV1_Init

3.3.3.1 Description

Allocates resources needed by DPV1 functions. Use DPV1_Exit to free resources.

3.3.3.2 Prototype

```
BOOL PFB_DPV1_Init (  DWORD      CardHandle,
                      BYTE      Nb_Blocks,
                      BYTE *     Nb_Blocks_Allocated  ) ;
```

3.3.3.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Nb_Blocks	Number of FDL blocks to allocate (1 to 128)
Nb_Blocks_Allocated	Number of FDL blocks allocated (0 to 128)

3.3.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.3.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied
2000 0114	Invalid Pointer – Nb_Blocks_Allocated
2000 011A	Card access timeout
2000 04XX	Not Offline state - cannot allocate resources
2000 0604	Invalid number of blocks requested
2000 060A	No FDL block available
2000 060B	DPV1 already initialized

3.3.4 DPV1_Exit

3.3.4.1 Description

Releases resources allocated by the DPV1_Init function.

3.3.4.2 Prototype

```
BOOL PFB_DPV1_Exit ( DWORD CardHandle );
```

3.3.4.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard

3.3.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.4.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0102	Invalid CardHandle
2000 0103	Write access denied
2000 011A	Card access timeout
2000 060C	DPV1 already released or not initialized

3.3.5 MSAC1_Read (PFB_MSAC1_Read_req)

3.3.5.1 Description

Sends the Read Data request to the slave. The data to be read is addressed via the Slot and Index.

3.3.5.2 Prototype

```
BOOL PFB_MSAC1_Read_req ( DWORD      CardHandle,
                          BYTE        Rem_Add,
                          BYTE        Slot_Number,
                          BYTE        Index,
                          BYTE        Length,
                          DWORD        Timeout,
                          PFB_DPV1_STS *DPV1Sts );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.5.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Rem_Add	Slave Address (0 to 126)
Slot_Number	Slot number of the data to be read (0 to 254)
Index	Index of the data to be read (0 to 254)
Length	Number of bytes to be read (1 to 240)
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.5.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer – DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0502	FDL error - see PFB_DPVS1_STS structure definition
2000 0601	Invalid slave address
2000 0602	Invalid length
2000 0603	No more FDL blocks available
2000 060C	DPV1 not initialized

3.3.6 MSAC1_Read (PFB_MSAC1_Read_con)

3.3.6.1 Description

Sends the polling telegram to the slave to receive the Read Data response.

3.3.6.2 Prototype

```
BOOL PFB_MSAC1_Read_con ( DWORD      CardHandle,
                           BYTE       Rem_Add,
                           BYTE *     Length,
                           BYTE *     Data,
                           DWORD      Timeout,
                           PFB_DPVS1_STS * DPV1Sts );
```

For details, refer to [PFB_DPVS1_STS](#), in Section 3.3.19.1.

3.3.6.3 Arguments

Name	Description
CardHandle	The handle returned by PFB_OpenCard
Rem_Add	Slave Address (0 to 126)
Length	Number of bytes read (0 to 240)
Data	Data Buffer
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.6.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.6.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - Length, Data and/or DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0501	DPV1 error - see PFB_DPV1_STS structure definition
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0601	Invalid slave address
2000 0603	No more FDL blocks available
2000 0608	Slave is not ready to deliver the response to a request
2000 0609	Invalid slave response
2000 060C	DPV1 not initialized

3.3.7 MSAC1_Write (PFB_MSAC1_Write_req)

3.3.7.1 Description

Sends the Write Data request to the slave. The data to be written is addressed via the Slot and Index.

3.3.7.2 Prototype

```
BOOL PFB_MSAC1_Write_req (      DWORD      CardHandle,
                               BYTE         Rem_Add,
                               BYTE         Slot_Number,
                               BYTE         Index,
                               BYTE         Length,
                               BYTE *       Data,
                               DWORD        Timeout,
                               PFB_DPV1_STS * DPV1Sts );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.7.3 Arguments

Name	Description
CardHandle	The handle returned by PFB_OpenCard
Rem_Add	Slave Address (0 to 126)
Slot_Number	Slot number of the Data to be written (0 to 254)
Index	Index of the Data to be written (0 to 254)
Length	Number of bytes to be written (1 to 240)
Data	Data Buffer
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.7.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.7.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - Data and/or DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0502	FDL error - see PFB_DPVS1_STS structure definition
2000 0601	Invalid slave address
2000 0602	Invalid length
2000 0603	No more FDL blocks available
2000 060C	DPV1 not initialized

3.3.8 MSAC1_Write (PFB_MSAC1_Write_con)

3.3.8.1 Description

Sends the polling telegram to the slave to receive the Write Data response.

3.3.8.2 Prototype

```

BOOL PFB_MSAC1_Write_con (      DWORD      CardHandle,
                               BYTE         Rem_Add,
                               BYTE *      Length,
                               DWORD      Timeout,
                               PFB_DPVS1_STS * DPV1Sts );

```

For details, refer to [PFB_DPVS1_STS](#), in Section 3.3.19.1.

3.3.8.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
Rem_Add	Slave Address (0 to 126)
Length	Number of bytes written (0 to 240)
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPVS1_STS)).

3.3.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.8.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - Length and/or DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0501	DPV1 error - see PFB_DPV1_STS structure definition
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0608	Slave is not ready to deliver the response to a request
2000 0601	Invalid slave address
2000 0603	No more FDL blocks available
2000 0609	Invalid slave response
2000 060C	DPV1 not initialized

3.3.9 MSAC2_Initiate (PFB_MSAC2_Initiate_req)

3.3.9.1 Description

Sends the Initiate request to the slave. The C_Ref parameter is the reference of the connection and must be used with class 2 functions.

3.3.9.2 Prototype

```

BOOL PFB_MSAC2_Initiate_req (DWORD           CardHandle,
                             DWORD *         C_Ref,
                             BYTE            Rem_Addr,
                             WORD            Send_TimeOut,
                             WORD            Features_Supported,
                             WORD            Profile_Features_Supported,
                             WORD            Profile_Ident_Number,
                             PFB_DPV1_ADDR_PARAM Add_Addr_Param,
                             DWORD           Timeout,
                             PFB_DPV1_STS *  DPV1Sts );

```

For details, refer to [PFB_DPV1_ADDR_PARAM](#), in Section 3.3.18.1, and [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.9.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference returned
Rem_Addr	Slave Address (0 to 126)
Send_Timeout	Time expected in 10 ms
Features_Supported	Features supported by the master
Profile_Features_Supported	Service functionality supported by the master
Profile_Ident_Number	Desired profile ID number (0 = no profile)
Add_Addr_Param	Subnet parameters - see structure definition
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.9.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.9.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - C_Ref and/or DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0501	DPV1 error - see PFB_DPV1_STS structure definition
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0601	Invalid slave address
2000 0603	No more FDL blocks available
2000 0607	Invalid Add_Addr_Param parameters
2000 0609	Invalid slave response
2000 060C	DPV1 not initialized

3.3.10 MSAC2_Initiate (PFB_MSAC2_Initiate_con)

3.3.10.1 Description

Sends the polling telegram to the slave to receive the Initiate response. The connection must be polled to remain alive. The time is defined by Send_TimeOut parameter. More information on Class 2 Initialization can be obtained from “Profibus – DP Extensions to EN 50170 (DPV1)” by PNO.

3.3.10.2 Prototype

```

BOOL PFB_MSAC2_Initiate_con (DWORD   CardHandle,
                             DWORD    C_Ref,
                             WORD *   Send_TimeOut,
                             WORD *   Features_Supported,
                             WORD *   Profile_Features_Supported,
                             WORD *   Profile_Ident_Number,
                             DWORD    Timeout
                             PFB_DPV1_STS *   DPV1Sts );

```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.10.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference
Send_Timeout	Updated Time by the slave in 10 ms
Features_Supported	Features supported by the slave
Profile_Features_Supported	Service functionality supported by the slave
Profile_Ident_Number	Profile ID supported by the slave
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.10.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - Send_TimeOut, Features_Supported, Profile_Features_Supported, Profile_Ident_Number and/or DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0501	DPV1 error - see PFB_DPV1_STS structure definition
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0603	No more FDL blocks available
2000 0606	Class 2 connection not initialized
2000 0608	Slave is not ready to deliver the response to a request
2000 0609	Invalid slave response
2000 060C	DPV1 not initialized
2000 060D	Invalid C_Ref

3.3.11 MSAC2_Abort (PFB_MSAC2_Abort_req)

3.3.11.1 Description

Sends the Abort request to the slave. This aborts the initiated connection. There is no confirmation function.

3.3.11.2 Prototype

```
BOOL PFB_MSAC2_Abort_req (    DWORD    CardHandle,
                             DWORD    C_Ref,
                             BYTE     Subnet,
                             WORD     Instance_ReasonCode,
                             DWORD    Timeout,
                             PFB_DPV1_STS *    DPV1Sts    );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.11.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference
Subnet	Location of the abort request source
Instance/Reason_Code	Protocol instance and reason for the abort
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.11.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.11.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0603	No more FDL blocks available
2000 0606	Class 2 connection not initialized
2000 060C	DPV1 not initialized
2000 060D	Invalid C_Ref

3.3.12 MSAC2Read (PFB_MSAC2_Read_req)

3.3.12.1 Description

Sends the Read Data request to the slave. The data to be read is addressed via the Slot and Index.

3.3.12.2 Prototype

```
BOOL PFB_MSAC2_Read_req ( DWORD    CardHandle,
                          DWORD    C_Ref,
                          BYTE     Slot_Number,
                          BYTE     Index,
                          BYTE     Length,
                          DWORD     Timeout,
                          PFB_DPV1_STS * DPV1Sts );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.12.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference
Slot_Number	Slot number of the Data to be read (0 to 254)
Index	Index of the Data to be read (0 to 254)
Length	Number of bytes to be read (1 to 240)
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.12.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0602	Invalid length
2000 0603	No more FDL blocks available
2000 0606	Class 2 connection not initialized
2000 060C	DPV1 not initialized
2000 060D	Invalid C_Ref

3.3.13 MSAC2Read (PFB_MSAC2_Read_con)

3.3.13.1 Description

Sends the polling telegram to the slave to receive the Read Data response.

3.3.13.2 Prototype

```
BOOL PFB_MSAC2_Read_con ( DWORD    CardHandle,
                          DWORD    C_Ref,
                          BYTE *    Length,
                          BYTE *    Data,
                          DWORD    Timeout,
                          PFB_DPV1_STS * DPV1Sts );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.13.3 Arguments

Argument	Type
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference
Length	Number of bytes read (0 to 240)
Data	Data Buffer
Timeout	Time allowed for message reply in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.13.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.13.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - Length, Data and/or DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0501	DPV1 error - see PFB_DPV1_STS structure definition
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0603	No more FDL blocks available
2000 0608	Slave is not ready to deliver the response to a request
2000 0606	Class 2 connection not initialized
2000 0609	Invalid slave response
2000 060C	DPV1 not initialized
2000 060D	Invalid C_Ref

3.3.14 MSAC2_Write (PFB_MSAC2_Write_req)

3.3.14.1 Description

Sends the Write Data request to the slave. The data to be written is addressed via the Slot and Index.

3.3.14.2 Prototype

```
BOOL PFB_MSAC2_Write_req (    DWORD    CardHandle,
                             DWORD      C_Ref,
                             BYTE        Slot_Number,
                             BYTE        Index,
                             BYTE        Length,
                             BYTE *      Data,
                             DWORD      Timeout,
                             PFB_DPV1_STS * DPV1Sts );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.14.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference
Slot_Number	Slot number of the Data to be written (0 to 254)
Index	Index of the Data to be written (0 to 254)
Length	Number of bytes to be written (1 to 240)
Data	Data Buffer
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.14.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.14.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - Data and/or DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0602	Invalid length
2000 0603	No more FDL blocks available
2000 0606	Class 2 connection not initialized
2000 060C	DPV1 not initialized
2000 060D	Invalid C_Ref

3.3.15 MSAC2_Write (PFB_MSAC2_Write_con)

3.3.15.1 Description

Sends the polling telegram to the slave to receive the Write Data response.

3.3.15.2 Prototype

```
BOOL PFB_MSAC2_Write_con (    DWORD    CardHandle,  
                             DWORD     C_Ref,  
                             BYTE *    Length,  
                             DWORD     Timeout,  
                             PFB_DPV1_STS * DPV1Sts );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.15.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference
Length	Number of bytes written (0 to 240)
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.15.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.15.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - Length and/or DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0501	DPV1 error - see PFB_DPV1_STS structure definition
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0603	No more FDL blocks available
2000 0606	Class 2 connection not initialized
2000 0608	Slave is not ready to deliver the response to a request
2000 0609	Invalid slave response
2000 060C	DPV1 not initialized
2000 060D	Invalid C_Ref

3.3.16 MSAC2_Idle (PFB_MSAC2_Idle_req)

3.3.16.1 Description

Sends the Idle request to the slave. This function keeps the connection with the slave open.

3.3.16.2 Prototype

```
BOOL PFB_MSAC2_Idle_req ( DWORD    CardHandle,  
                          DWORD     C_Ref,  
                          DWORD     Timeout,  
                          PFB_DPV1_STS * DPV1Sts );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.16.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference
Timeout	Time allowed for message reply, in milliseconds
DPV1Sts	DPV1 status (refer to Section 3.3.19, Structure Definition (PFB_DPV1_STS)).

3.3.16.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.16.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0502	FDL error - see PFB_DPV1_STS structure definition
2000 0603	No more FDL blocks available
2000 0606	Class 2 connection not initialized
2000 060C	DPV1 not initialized
2000 060D	Invalid C_Ref

3.3.17 MSAC2_Idle (PFB_MSAC2_Idle_con)

3.3.17.1 Description

Sends the polling telegram to the slave to receive the Idle response.

3.3.17.2 Prototype

```
BOOL PFB_MSAC2_Idle_con ( DWORD    CardHandle,  
                          DWORD    C_Ref,  
                          DWORD    Timeout,  
                          PFB_DPV1_STS *    DPV1Sts  );
```

For details, refer to [PFB_DPV1_STS](#), in Section 3.3.19.1.

3.3.17.3 Arguments

Argument	Description
CardHandle	The handle returned by PFB_OpenCard
C_Ref	Connection reference
Timeout	Time allowed for message reply in milliseconds
DPV1Sts	DPV1 status – see structure definition

3.3.17.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve the error code.

3.3.17.5 Error Codes

Value (hex)	Description
2000 0003	Driver not loaded
2000 0100	General failure
2000 0102	Invalid CardHandle
2000 0114	Invalid pointer - DPV1Sts
2000 011A	Card access timeout
2000 0257	Message reply timeout
2000 025C	Online state expected
2000 0501	DPV1 error - see PFB_DPV1_STS structure definition
2000 0502	FDL error – PFB_DPV1_STS see structure definition
2000 0603	No more FDL blocks available
2000 0606	Class 2 connection not initialized
2000 0608	Slave is not ready to deliver the response to a request
2000 0609	Invalid slave response
2000 060C	DPV1 not initialized
2000 060D	Invalid C_Ref

3.3.18 Structure Definition (PFB_DPV1_ADDR_PARAM)

3.3.18.1 Declaration

```
/* DPV1 C2 sub lan addressing structure definition*/
```

```
typedef struct
{
    BYTE          STYPE;    /* 0 or 1 */
    BYTE          S_Addr_len;
    BYTE          DTYPE;    /* 0 or 1 */
    BYTE          D_Addr_len;
    PFB_DPV1_SUBADDRESS S_Addr;
    PFB_DPV1_SUBADDRESS D_Addr;
} PFB_DPV1_ADDR_PARAM;
```

```
typedef struct
{
    BYTE  API;                /* 0 to 255 */
    BYTE  SCL;                /* 0 to 255 */
    BYTE  Network_Address[6];
    BYTE  *Mac_Address;
} PFB_DPV1_SUBADDRESS;
```

- If STYPE = 1, the optional Network_Address and Mac_Address are present in S_Addr
- S_Addr_Len indicates the length of the S_Addr:
 - If STYPE = 0, the S_Addr_Len should be 2
 - If STYPE = 1, the S_Addr_Len should be 8 + the length of Mac_Address
 - If DTYPE = 1, the optional Network_Address and Mac_Address are present in D_Addr
- D_Addr_Len indicates the length of the D_Addr:
 - If DTYPE = 0, the D_Addr_Len should be 2
 - If DTYPE = 1, the D_Addr_Len should be 8 + the length of Mac_Address
- API identifies the applicom process instance

- SCL identifies the access level
- Network_Address identifies the network address, according to ISO/OSI-Network addresses
- MAC_Address identifies the MAC_Address

3.3.19 Structure Definition (PFB_DPV1_STS)

3.3.19.1 Declaration

```
typedef struct
{
    DWORD   Error_Code;           /* Error Code of the function */
    BYTE    ExtError[4];         /* Depending of the Error_Code */
    PFB_DPV1_STS
}
```

If Error_Code = 0x20000501 (DPV1 Error)

```
ExtError[0]=Error_Decode; /* 128=DPV1, 254=FMS, 255=HART */
ExtError[1]=Error_Code1; /* DPV1 Error Class & Error Code */
ExtError[2]=Error_Code2; /* DPV1 user-specific */
```

If Error_Code = 0x20000502 (FDL Error)

```
ExtError[0]=FDL_ErrorCode; /* see Lay2sRspStatus */
```


3.3.20 Meaning of FDL_ErrorCode

The following table describes the meaning of the FDL_ErrorCode.

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, etc
9Fh	No access
AFh	Double token detected, problems with wiring, termination, etc.
BFh	Response buffer too small.
8Fh	Noise at SM command, problems with wiring, termination, etc.

3.3.21 Meaning of DPV1 Error

If a DV1 error occurs, the extended error structure contains 3 bytes:

- The first indicates the type of error (its value is always 0x80: DPV1 error)
- The second consists of two parts:
 - The highest four bits represent the error class
 - The lower four bits represent the error code
- The third one contains a slave specific error code

The following table describes the signification for the error class and error code of the second error byte.

Error_Class	Meaning	Error_Code
0 to 9	Reserved	-
10=	Application	0 = read error 1 = write error 2 = module failure 3 to 7 : reserved 8 : version conflict 9 : features not supported 10 to 15 : User specific
11=	Access	0 = invalid index 1 = write length error 2 = invalid slot 3 = type conflict 4 = invalid area 5 = state conflict 6 = access denied 7 = invalid range 8 = invalid parameter 9 = invalid type 10...15 : user specific
12 =	Resource	0 = read constrain conflict 1 = write constrain conflict 2 = resource busy 3 = resource unavailable 4 to 7 : reserved 8 to 15: use specific
13 to 15	User specific	-

4

PFBMAN32 API Structure Definitions

Chapter Sections:

- This chapter contains an alphabetical list of the DLL data types

4.1 Profibus Network

4.1.1 Card Status Structure

4.1.1.1 Declaration

```
typedef struct
{
/* 5136-PFB Basic Control/Status registers */
    BYTE    pfbCommand;    /* Card command register - see below */
    BYTE    pfbStatus;     /* Card status register - see below */

/* 5136-PFB Card and Module Identification registers */
    WORD    pfbCardId;     /* 5136-PFB Card ID (0xaad0) */
    WORD    pfbModId;      /* PFBPROFI Module ID (0xbb01 or 0xbb03) */
    WORD    pfbModVer;     /* PFBPROFI Module Version (ex. 0x0102 = 1.02) */
} PFB_CARD_STS;

/* definitions related to pfbCommand */

CMD_OFF      0xE0        /* PFB - card offline ready to take commands */
CMD_ON       0xE1        /* PFB - card is online ready to take offline
                           command */

CMD_COM_CFG  0xE2        /* PFB - card is being configured from serial
                           port */

CMD_ERROR    0xEF        /* PFB - card is in error, status contains error
                           code */

CMD_GO_ON    0x01        /* HOST - card should now go on line */
CMD_GO_OFF   0x02        /* HOST - card should now go offline, or
                           abort configuration */

CMD_REINIT   0x03        /* HOST - card should reinitialize memory
                           and all parameters */

CMD_CLR_CFG_BUF  0x05    /* HOST - card should check network parameters,
                           and assign defaults */

CMD_CPY_MAS_CFG  0x06    /* HOST - card should copy a page of
                           configuration from pfbBinCfgPage */

CMD_AUTO_BAUD_DET  0x07  /* HOST - card should do an automatic baud
                           detect */

CMD_MAS_ASSIGN_ADDR  0x08 /* HOST - assign and fill in data address
                           (page/offset) for DP master */

CMD_AUTODP_CFG  0x09    /* HOST - configure dp master using online
                           auto-configuration */

CMD_CFG_ABF_SHRAM  0x0F  /* HOST - configure DP master from ABF binary
                           previously copied */
```

CMD_CFG_2BF_SHRAM	0x10	/* HOST - configure DP master from ET200 binary previously copied */
CMD_CFG_FROM_FLASH	0x11	/* HOST - configure card from configuration in flash */
CMD_CFG_FLASH_GO_ON	0x14	/* HOST - configure card from flash then on line */
CMD_PGM_TO_FLASH	0x21	/* HOST - burn card configuration into flash */
CMD_PGM_SID_FLASH	0x22	/* HOST - burn special id (SST only interface) */
CMD_GET_SID_FLASH	0x23	/* HOST - get special id (SST only interface) */
CMD_ERR_ACK	0xFF	/* HOST - acknowledge offline error (online errors cannot be acknowledged) */
/* definitions related to pfbStatus */		
STS_NO_ERROR	0x00	/* last command executed successfully */
STS_BAD_CMD	0x01	/* invalid, non supported command */
STS_BAD_BAUD	0x02	/* invalid, non supported BAUD rate */
STS_BAD_STN_ADR	0x03	/* invalid, non supported station address */
STS_BAD_HI_STN_ADR	0x04	/* invalid, non supported highest station address */
STS_BAD_TOK_ROT	0x05	/* token rotation timeout of range */
STS_BAD_SLOT_TME	0x06	/* slot timeout of range */
STS_BAD_IDLE_1	0x07	/* idle time 1 out of range */
STS_BAD_IDLE_2	0x08	/* idle time 2 out of range */
STS_BAD_RDY_TME	0x09	/* ready timeout of range */
STS_BAD QUI_TME	0x0A	/* quiet timeout of range */
STS_BAD_GAP_UPD	0x0B	/* gap update timeout of range */
STS_BAD_TOK_RETRY	0x0C	/* token retry limit out of range */
STS_BAD_MSG_RETRY	0x0D	/* message retry limit out of range */
STS_BAD_TOK_ERR_LIM	0x0E	/* token error limit out of range */
STS_BAD_RSP_ERR_LIM	0x0F	/* response error limit out of range */
STS_BAUD_DET_ERROR	0x10	/* unable to detect baud rate (no activity detected) */
STS_CFG_BAD_CHK_PATTERN	0x20	/* binary configuration file corrupted */
STS_CFG_BIN_TOO_SHORT	0x21	/* binary configuration file corrupted */
STS_CFG_BIN_TOO_LONG	0x22	/* binary configuration file corrupted */
STS_CFG_BAD_CHKSUM	0x23	/* binary configuration file corrupted */
STS_CFG_INVALID_CPU_HDR	0x24	/* binary configuration file corrupted */
STS_CFG_INVALID_SLV_REC_TYP	0x25	/* binary configuration file corrupted */
STS_CFG_RX_OVERFLOW	0x26	/* too much Master RX data has been configured (16k max.) */
STS_CFG_TX_OVERFLOW	0x27	/* too much Master TX data has been configured (16k max.) */
STS_CFG_DESIG_NAME_TOO_LONG	0x28	/* Nm= parameter, name too long 12 chars max.) */
STS_CFG_DESIG_BAD_ARG	0x29	/* unrecognized argument (Nm=, TX=, RX=, Ch) */

STS_CFG_INV_RX_OFS	0x2A	/* RX= parameter, invalid offset (0-3FF8h) */
STS_CFG_INV_TX_OFS	0x2B	/* TX= parameter, invalid offset (0-3FF8h) */
STS_CFG_DESIG_OFS_NOT_SPEC	0x2C	/* RX, TX Offsets have been specified for one, but not all slaves */
STS_CFG_RX_OVERLAP	0x2D	/* RX data of one block overlaps another */
STS_CFG_TX_OVERLAP	0x2E	/* TX data of one block overlaps another */
STS_CFG_INV_LEN	0x2F	/* invalid parameter or check data length */
STS_CFG_MAS_EXT_ALLOC_ERR	0x35	/* out of master extension memory */
STS_CFG_ADDR_OUT_OF_RANGE	0x37	/* out of master extension memory */
STS_CFG_COPY_TABLE_OVERUN	0x38	/* not enough room in copy table */
STS_CFG_AUTOCFG_DONE	0x39	/* The DP master auto-configuration command is done */
STS_ACFG_STATE_ERR	0x40	/* The DP master auto-configuration command failed */
STS_CFG_NO_CONFIG	0x30	/* no configuration present to program into flash */
STS_FLASH_BAD_ID	0x31	/* internal flash error */
STS_FLASH_ERASE_ERR	0x32	/* internal flash error */
STS_FLASH_PROG_ERR	0x33	/* internal flash error */
STS_FLASH_VRFY_ERR	0x34	/* internal flash error */
STS_LAY2M_INV_MAX_LEN	0x36	/* invalid maximum length for L2 message */
/* the following are fatal errors, the card must be reloaded with the firmware module */		
STS_CFG_INTERNAL_ERROR	0x80	
STS_OUT_OF_APBS	0x81	
STS_HOST_WD_BITE	0x82	/* host did not kick PFB watchdog within watchdog period */
STS_HEAP_ALLOC_FAIL	0x83	
STS_SH_HEAP_ALLOC_FAIL	0x84	
STS_NET_ERROR	0x90	/* net error, OPTION_STAY_OFF_ERR set, card is offline */

4.1.2 Diagnostic Counters Structure

4.1.2.1 Declaration

```
typedef struct
{
    /* General Statistics */

    BYTE   errLanOffline;           /* LAN encountered errors and went into offline
                                   state */

    WORD   diagConf;               /* Total confirmations (to requests from us)
                                   (MAS,LAY2,FMS) */

    WORD   errNotOk;               /* Total Not OK confirmations and/or indications
                                   (MAS,LAY2,FMS) */

    DWORD  diagTokHldTime;         /* Actual Instantaneous token hold time in
                                   tBit */

    DWORD  diagMinTokHldTime;      /* Minimum Actual token hold time in tBit */

    /* DP Master Block Statistics */

    WORD   diagMasterUpdate;       /* Master I/O update cycles completed */
    BYTE   errMasErr;              /* Master->DP slave communication errors */
    BYTE   errReConfig;           /* Master->DP went offline and had to be
                                   reconfigured */

    DWORD  diagMasScanTime;        /* Instantaneous master scan time in
                                   microseconds */

    DWORD  diagMasMaxScanTime;     /* Maximum master scan time in microseconds */

    /* DP Slave Statistics */

    WORD   diagSlaveUpdate;        /* Slave updates */
    BYTE   errSlvErr;              /* Slave configuration failures */
    BYTE   errSlvTout;            /* Slave watchdog timeouts */

    /* Layer 2 Message Statistics */

    WORD   diagLay2MsgOk;          /* Layer 2 messages sent OK */
    BYTE   errLay2MsgNotOk;        /* Layer 2 message errors */
    BYTE   errLay2MsgProcOvrn;     /* Layer 2 periodic processing overruns */

    /* Layer 2 SAP Statistics */

    WORD   diagLay2SapOk;          /* Layer 2 SAP requests processed OK */
    BYTE   errLay2SapNotOk;        /* Layer 2 SAP errors */
    BYTE   errLay2SapTout;        /* Layer 2 SAP update timeouts */
}
```

```
/* ASPC2 PROFIBUS controller Statistics */

BYTE  errInvReqLen;      /* invalid request length errors */
BYTE  errFifo;           /* FIFO overflow errors */
BYTE  errRxOverrun;      /* receive overrun errors */
BYTE  errDbtTok;         /* double token errors (bad wiring or hardware) */
BYTE  errRespErr;        /* response errors (bad wiring or hardware) */
BYTE  errSyniErr;        /* SYNI errors (bad wiring or hardware) */
BYTE  errNetTout;        /* network timeout errors */
BYTE  errHsa;            /* Station higher than HSA was heard */
BYTE  errStn;            /* Duplicate Station Detected */
BYTE  errPasTok;         /* Unable to Pass Token (bad wiring or hardware) */
BYTE  errLasBad;         /* active station list invalid (bad wiring or hardware) */

BYTE  errInternal;       /* internal error */
BYTE  errArg;            /* internal error argument */
BYTE  errEventOverrun;   /* a new event occurred before the last one was cleared */

} PFB_DIAG_CTRS;
```


4.2 Station List Structure

4.2.1.1 Declaration

```
typedef struct
{
  BYTE pfbAckLasChnge;           /* bit 0=1 to acknowledge station list change
                                  (card clears) */

  BYTE pfbUpdPasv;              /* bit 0=1 to update status of passive stations
                                  (card clears) */

  BYTE pfbActStnList[128];      /* Active Station list, 1 byte per station
                                  - see below */
} PFB_STN_LST;

/* definitions related to pfbActStnList */

LAS_PASSIVE      0x01          /* this station is passive */
LAS_ACTIVE       0x04          /* this station is active */
LAS_CHANGED      0x08          /* status of this station has changed */
```

4.2.2 Network Configuration Structure

4.2.2.1 Declaration

```
typedef struct
{
/* 5136-PFB --> Host Event and Interrupt control registers */

WORD pfbEvtEna;           /* Event enable mask */
WORD pfbIntEna;          /* Event interrupt enable mask */

/* ASPC2 PROFIBUS controller Basic Parameters */

BYTE pfbStnAddr;          /* PFB Local station address, (0-126) */
BYTE pfbHiStnAddr;        /* highest station address on network,
                           (0-126), 255 - disable */

BYTE pfbActive;           /* 1=active station, 0=passive station */
BYTE pfbBaud;             /* network baud rate - see below */
WORD pfbOptions;          /* network options - see below */

/* ASPC2 PROFIBUS controller Bus Parameters (PFB will put in defaults if left
unchanged) */

DWORD pfbTokRotTime;      /* Target token rotation time, 0 - set default value,
                           (256-16,777,215 tBit) */

WORD pfbSlotTime;         /* Slot time, (37-16,383 tBit) */
WORD pfbIdleTime1;        /* Idle time 1, (35-1,023 tBit) */
WORD pfbIdleTime2;        /* Idle time 2, (35-1,023 tBit) */
WORD pfbReadyTime;        /* Ready time, (11-1,023 tBit) */
BYTE pfbGapUpdFact;       /* Gap update factor, (0-255) */
BYTE pfbQuietTime;        /* Time quiet, (0-127 tBit) */

/* ASPC2 PROFIBUS controller Error Handling Parameters */

BYTE pfbTokRetryLimit;    /* Token retry limit, (0-15) */
BYTE pfbMsgRetryLimit;    /* Message retry limit, (0-15) */
BYTE pfbTokErrLimit;      /* Token error limit, (0-255) */
BYTE pfbRespErrLimit;     /* Response error limit, (0-15) */
char pfbLocIdUsrStr[112]; /* Text for the 4 fields to return in the ID
                           response */
} PFB_NET_CFG;

/* definitions related to pfbEvtEna */

EVT_ENA_LAS_CHANGE      0x0001 /* active station list change */
EVT_ENA_BUS_ERROR       0x0002 /* PROFIBUS error */
```

```
/* definitions related to pfbIntEna */

INT_ENA_LAS_CHANGE      0x0001  /* active station list change */
INT_ENA_BUS_ERROR       0x0002  /* PROFIBUS error */
INT_ENA_DP_MAS_EVENT    0x0004  /* DP master event has occurred */
INT_ENA_DP_SLV_EVENT    0x0008  /* DP slave event has occurred */
INT_ENA_LY2M_EVENT      0x0010  /* Layer 2 message event has occurred */
INT_ENA_LY2S_EVENT      0x0020  /* Layer 2 SAP event has occurred */

/* definitions related to pfbBaud */

BAUD_9k6                0
BAUD_19k2               1
BAUD_93k75              2
BAUD_187k5              3
BAUD_500k               4
BAUD_750k               5
BAUD_1m5                6
BAUD_3m                 7
BAUD_6m                 8
BAUD_12m                9

#define BAUD_31k25       10
#define BAUD_45k45       11

/* definitions related to pfbOptions */

OPTION_REPEATER          0x0001  /* 1=repeater on network, 0=no repeater on network */
OPTION_FMS               0x0002  /* 1=fms devices on network, 0=dp only on network */
OPTION_STAY_OFF_ERR      0x0004  /* stay offline if token error limit or msg.
                                error limit exceeded */
```

4.3 DP Master

4.3.1 Master Global Configuration Structure

4.3.1.1 Declaration

```
typedef struct
{
    BYTE   pfbMasCntrlCfg;           /* Global control for all master blocks -
                                     see below */

    BYTE   pfbMasCntrlPage;          /* Memory page which contains master block
                                     control table - READ ONLY */

    BYTE   pfbMasRxPage;             /* Memory page which contains master RX data
                                     (from slaves) - READ ONLY */

    BYTE   pfbMasTxPage;            /* Memory page which contains master TX data
                                     (to slaves) - READ ONLY */

    WORD   pfbMasMinIoCycTme;        /* Minimum I/O cycle time in 100us increments
                                     (0-6.5535s) */

    WORD   pfbMasMaxIoCycTme;        /* Maximum I/O cycle time in 10ms increments
                                     (0.01-655.35s) */
} PFB_MAS_GLB_CFG;

/* definitions related to pfbMasCntrlCfg */

PFB_MAS_CTRL_RUN_MODE    0x02      /* scan i/o in RUN mode
    */
PFB_MAS_CTRL_USR_OFS     0x04      /* user defined RX and TX data offsets (no PROFIBUS
    auto assign) */

PFB_MAS_CTRL_ENABLE      0x08      /* enable master mode on PFB card */
PFB_MAS_CTRL_DIS_LED     0x10      /* disable status led for DP master function */
PFB_MAS_CTRL_HOLD_INTR   0x20      /* hold DP master event interrupt(s) until end
    of scan */

PFB_MAS_CTRL_EVT_SCAN    0x40      /* generate event (interrupt) at end of scan */
DONE

PFB_MAS_CTRL_ADDR_       0x80      /* DP master data addresses have been assigned */
ASSIGNED
```

4.3.2 Master Status Structure

4.3.2.1 Declaration

```
typedef struct
{
    BYTE masStatus;           /* Status Register (Host only Reads) - see below */
    BYTE masError;            /* Error indication - see below */
    BYTE masEvent;            /* Event Flags - see below */
    BYTE masDiagEvent;        /* Diagnostic Event Flags - see below */
    BYTE masExtErrInfo;       /* Extended error */
} PFB_MAS_STS;

/* definitions related to masStatus */

MAS_STS_OK                0x80    /* current status of this master block is OK */

/* definitions related to masError */

MAS_ERR_CFG_FAILURE       0x01    /* failure while trying to configure slave */
MAS_ERR_SLV_ID_MISMATCH   0x02    /* slave real ID does not match slave's configured
ID */

MAS_ERR_DATA_UPD_FAILURE  0x03    /* frame delivery problem while updating slave
data */

MAS_ERR_CFG_DIAG_READ_FAILURE 0x04 /* frame delivery problem while reading slave
diagnostics */

MAS_ERR_CFG_DIAG_STS1_ERR 0x05    /* error in diagnostic status byte #1 during
configure */

MAS_ERR_CFG_DIAG_STS2_ERR 0x06    /* error in diagnostic status byte #2 during
configure */

MAS_ERR_UPD_DIAG_STS1_ERR 0x07    /* error in diagnostic status byte #1 during
diagnostic read */

MAS_ERR_UPD_DIAG_STS2_ERR 0x08    /* error in diagnostic status byte #2 during
diagnostic read */

MAS_ERR_CFG_STN_MISMATCH  0x09    /* station address from diagnostic read does not
match */

MAS_ERR_IO_CYC_TOUT       0x0a    /* timeout waiting for i/o update */
MAS_ERR_SLV_WD_OFF        0x0b    /* **warning slave watchdog is not enabled */
```

```
/* definitions related to masEvent */

MAS_EVT_UPD          0x01      /* slave has been updated */
MAS_EVT_RX_DATA_CHG  0x02      /* RX data has changed */

/* definitions related to masDiagEvent */

MAS_DEVT_DIAG_UPD    0x01      /* slave diagnostics have been updated */
```

4.4 Master Configuration Structure

4.4.1.1 Declaration

```
typedef struct
{
    BYTE masStn;                /* station address of slave, (0-126) */
    BYTE masCntCfg;             /* Control and configuration register - see below */
    WORD masRxDataOfs;          /* Data received from slave offset within memory
                                page */

    WORD masTxDataOfs;          /* Data to be sent to slave offset within memory
                                page */

    BYTE masRxDataLen;          /* Data received from slave length in bytes */
    BYTE masTxDataLen;          /* Data to be sent to slave length in bytes */
    BYTE masSiemType;           /* SIEMENS Device Type - see below */
    char masDesign[13];         /* slave designation (text) */
} PFB_MAS_CFG;

/* definitions related to masCntCfg */

MAS_CTL_IGNORE_STS    0x01    /* ignore status of this master block */
MAS_CTL_EVT_RX_CHG    0x02    /* generate event (interrupt) when RX data from slave
                                changes */

MAS_CTL_RX_BYTE_SWAP  0x04    /* receive byte swap */
MAS_CTL_EVT_UPDTE     0x08    /* generate event (interrupt) when slave update is
                                complete */

MAS_CTL_FAIL_SAFE     0x10    /* slave is a fail safe slave */
MAS_CTL_TX_BYTE_SWAP  0x40    /* reserved */
MAS_CTL_ENABLE        0x80    /* master enable block */

/* definitions related to masSiemType */

MAS_SIEM_TYP_OLD_DP   0x80    /* old type of SIEMENS module */
```

4.4.2 DP Configuration Check Structure

4.4.2.1 Declaration

```
typedef struct
{
    BYTE dpChkLen;           /* length of configuration check data */
                             /* Configuration check to slave length in bytes */
    BYTE dpChk[244];         /* Configuration Check values from master
                             (32 bytes) */

                             /* Configuration check values to slave (128 bytes) */
} PFB_DP_CHK_CFG;
```

4.5 DP Diagnostic Information Structure

4.5.1.1 Declaration

```
typedef struct
{
    BYTE dpSts1;             /* status byte 1 to master */
                             /* status byte 1 from slave */

    BYTE dpSts2;             /* status byte 2 to master */
                             /* status byte 2 from slave */

    BYTE dpSts3;             /* status byte 3 to master */
                             /* status byte 3 from slave */

    BYTE dpMasStn;           /* Station that configured Slave */
                             /* Station that configured Slave */

    BYTE dpID_hi;            /* ID value to master hi byte (default 0x67) */
                             /* ID hi byte sent back from slave */

    BYTE dpID_lo;            /* ID value to master low byte (default 0x15) */
                             /* ID low byte sent back from slave */

    BYTE dpDiagLen;          /* Length of diagnostics to master */
                             /* Diagnostic from slave length in bytes */

    BYTE dpDiag[244];        /* Diagnostics to master */
                             /* vendor defined diagnostic information from
                             slave */
} PFB_DP_DIAG_INFO;
```


4.5.2 DP Parameter Data Structure

4.5.2.1 Declaration

```
typedef struct
{
    BYTE dpParmLen;                /* length of parameters from master */
                                   /* Parameters to slave length in bytes */

    BYTE dpMasSts;                 /* status byte from master */
                                   /* status byte to slave */

    BYTE dpWdFact1;                /* Watchdog factor 1 from master */
                                   /* Watchdog factor 1 to slave */

    BYTE dpWdFact2;                /* Watchdog factor 2 from master */
                                   /* Watchdog factor 2 to slave */

    BYTE dpReadyTime;              /* Response delay time (tBit) from master */
                                   /* Response delay time (tBit) to slave */

    BYTE dpMasID_hi;               /* ID value from master (hi byte) */
                                   /* ID value to slave hi */

    BYTE dpMasID_lo;               /* ID value from master (low byte) */
                                   /* ID value to slave low */

    BYTE dpGrpId;                  /* Group ID value from master */
                                   /* Group ID value for slave (not supported,
                                   always 0) */

    BYTE dpParm[237];              /* parameters from master (237 bytes) */
                                   /* parameters to slave (237 bytes) */
} PFB_DP_PARM_DATA;
```

4.6 Layer 2 Message

4.6.1 Layer 2 Message Configuration Structure

4.6.1.1 Declaration

```
typedef struct
{
    BYTE  lay2mCntCfg;           /* Control and configuration register - see below */
    WORD  lay2mUpdTime;         /* update interval for periodic * 1ms (1-16380,
                                0=cyclic) */

    WORD  lay2mErrTime;         /* retry interval for periodic if error occurs * 1ms
                                (1-6380) default: 0 - no retry */

    WORD  lay2mRxDataOfs;       /* receive data (from response) offset within Layer 2
                                data memory page */

    WORD  lay2mTxDataOfs;       /* transmit data (request) offset within Layer 2 data
                                memory page */

    BYTE  lay2mRxDataPage;      /* receive data (from response) data memory page */
    BYTE  lay2mTxDataPage;      /* transmit data (request) data memory page */
} PFB_L2MSG_CFG;

/* definitions related to lay2mCntCfg */

LAY2M_CTL_PERIODIC      0x01    /* set periodic messages */
LAY2M_CTL_RETRY_
PERIODIC                0x02    /* if error set retry periodic */

LAY2M_CTL_HI_PRI        0x04    /* set high priority for message */
LAY2M_CTL_EVT_RX_CHG    0x08    /* generate event (interrupt) when RX data to this
                                message changes */

LAY2M_CTL_IGNORE_STS    0x10    /* ignore status of this message block (all_ok, LED,
                                and event) */

LAY2M_CTL_EVT_CONFIRM    0x20    /* generate event (interrupt) when this message is
                                confirmed */

LAY2M_CTL_RX_BYTE_SWAP  0x40    /* receive swap byte */
```

4.6.2 Layer 2 Message Status Structure

4.6.2.1 Declaration

```
typedef struct
{
    BYTE  lay2mState;           /* Current state of message block - see below */
    BYTE  lay2mStatus;         /* Status register (Host only Reads) - see below */
    BYTE  lay2mError;          /* Error register - see below */
    BYTE  lay2mEvent;          /* Events - see below */
    BYTE  lay2mRspStatus;      /* Response Status if not OK */
    BYTE  lay2mRxLen;          /* actual receive data length in bytes */
} PFB_L2MSG_STS;

/* definitions related to lay2mState */

LAY2M_STE_DISABLE      0x00
LAY2M_STE_IN_CONFIGURE 0x01
LAY2M_STE_ENABLE       0x02
LAY2M_STE_ACTIVE       0x03
LAY2M_STE_DONE         0xFF

/* definitions related to lay2mStatus */

LAY2M_STS_OK           0x80

/* definitions related to lay2mError */

LAY2M_ERR_NOT_OK       0x01
LAY2M_ERR_RX_LEN      0x02
LAY2M_ERR_TX_LEN      0x03

/* definitions related to lay2mEvent */

LAY2M_EVT_CONFIRM      0x01
LAY2M_EVT_RX_DATA_CHG  0x02
```

4.7 Layer 2 Message Transmit Structure

4.7.1.1 Declaration

```
typedef struct
{
  BYTE   lay2mDstStn;           /* destination station address (or with 0x80 to
                                enable DstSap) */

  BYTE   lay2mDstSap;           /* destination SAP (0-63, 255 - no SAP),
                                default 255 */

  BYTE   lay2mSrcSap;           /* source SAP check (0-63, 255 - no SAP),
                                default 255) */

  BYTE   lay2mFrmCntrl;         /* frame control byte - see below */
  BYTE   lay2mTxLen;            /* transmit data (request) length in bytes (0-244) */
} PFB_L2MSG_TX;

/* definitions related to lay2mFrmCntrl */

FC_SDA1      0x03      /* Send Data with ACK Low Priority */
FC_SDAh      0x05      /* Send Data with ACK High Priority */
FC_SDN1      0x04      /* Send Data No ACK Low Priority */
FC_SDNh      0x06      /* Send Data No ACK High Priority */
FC_SRD1      0x0C      /* Send and Request Data Low Priority */
FC_SRDh      0x0D      /* Send and Request Data High Priority */
FC_SmTime1   0x00      /* First SM time message */
FC_SmTime2   0x80      /* Second SM time message */
FC_SmSDN     0x02      /* SM Send Data No ACK */
FC_SmSRD     0x01      /* SM Send and Request Data */
FC_SmSRDSltDel 0x0A     /* SM Send and Request Data Slot Del */
FC_SmSRDSltKeep 0x0B    /* SM Send and Request Data Slot Keep */
FC_DdbSRD    0x07      /* DDB Send and Request Data */
FC_DiagSRD   0x08      /* Diagnosis Send and Request Data */
FC_ReqFDL    0x09      /* Request FDL status */
FC_ReqId     0x0E      /* Request ID */
FC_ReqLSAPSts 0x0F     /* Request LSAP Status */
```

4.8 Layer 2 SAP

4.8.1 Layer 2 SAP Configuration Structure

4.8.1.1 Declaration

```
typedef struct
{
    BYTE    lay2sStn;                /* if strict station, accept updates only from this
                                     station (0-126) */

    BYTE    lay2sSap;                /* if strict SAP, accept updates only from this SAP
                                     (0-63) */

    WORD    lay2sCntCfg;             /* Control and configuration register - see below */
    WORD    lay2sTimeOut;            /* timeout for SAP * 10ms (1-8190, 0=disable) */
    BYTE    lay2sFrmCntrl;           /* if strict FC, accept updates only specified Frame
                                     Control values - see below */

    WORD    lay2sRxDataOfs;          /* receive data (from request) offset within memory
                                     page */

    WORD    lay2sTxDataOfs;          /* transmit data (response) offset within memory
                                     page */

    BYTE    lay2sRxDataPage;         /* receive data (from request) data memory page */
    BYTE    lay2sTxDataPage;         /* transmit data (response) data memory page */
} PFB_L2SAP_CFG;

/* definitions related to lay2sCntCfg */

LAY2S_CTL_RX_BYTE_SWAP 0x0004      /* swap hi and low bytes of RX data to this SAP */
LAY2S_CTL_EVT_RX_CHG   0x0008      /* generate event (interrupt) when RX data to this
SAP changes */

LAY2S_CTL_IGNORE_STS   0x0010      /* ignore status of this SAP block (all_ok flag,
LED and event) */

LAY2S_CTL_EVT_UPDTE    0x0020      /* generate event (interrupt) when this SAP is
updated (Indication) */

/* definitions related to lay2sFrmCntrl */

LAY2S_FC_ALL           0x00        /* accept all requests */
LAY2S_FC_SDN_LO        0x01        /* accept only SDN low priority */
LAY2S_FC_SDN_HI        0x02        /* accept only SDN high priority */
LAY2S_FC_SDN_LO_HI     0x03        /* accept only SDN high or low priority */

LAY2S_FC_SDA_LO        0x05        /* accept only SDA low priority */
LAY2S_FC_SDA_HI        0x06        /* accept only SDA high priority */
```

```

LAY2S_FC_SDA_LO_HI      0x07      /* accept only SDA high or low priority */
LAY2S_FC_SRD_LO         0x09      /* accept only SRD low priority */
LAY2S_FC_SRD_HI         0x0a      /* accept only SRD high priority */
LAY2S_FC_SRD_LO_HI      0x0b      /* accept only SRD high or low priority */
LAY2S_FC_DDB_REQ        0x0c      /* DDB Request */
LAY2S_FC_DDB_REQ_LO     0x0d      /* DDB Request low priority */
LAY2S_FC_DDB_REQ_HI     0x0e      /* DDB Request high priority */
LAY2S_FC_DDB_REQ_LO_HI  0x0f      /* DDB Request low and high priority */

```

4.9 Layer 2 SAP Status Structure

4.9.1.1 Declaration

```

typedef struct
{
    BYTE  lay2sStn;          /* if strict station, accept updates only from this
                           station (0-126) */

    BYTE  lay2sSap;          /* if strict SAP, accept updates only from this SAP
                           (0-63) */

    BYTE  lay2sSrcSap;
    BYTE  lay2sStatus;      /* Status register (Host Reads Only) - see below */
    BYTE  lay2sError;       /* Error register - see below */
    BYTE  lay2sEvent;       /* Events - see below */

    BYTE  lay2sSrcStn;
    BYTE  lay2sRspStatus;   /* Response Status if NOT_OK */
    BYTE  lay2sRxLen;       /* actual receive data length in bytes */
} PFB_L2SAP_STS;

/* definitions related to lay2sStatus */

LAY2S_STS_OK             0x80

/* definitions related to lay2sError */

LAY2S_ERR_NOT_OK        0x01
LAY2S_ERR_TIME_OUT      0x02
LAY2S_ERR_RX_LEN        0x03
LAY2S_ERR_TX_LEN        0x04
LAY2S_ERR_BAD_PARAM     0x05

/* definitions related to lay2sEvent */

LAY2S_EVT_UPDATE        0x01
LAY2S_EVT_RX_DATA_CHG   0x02

```

4.10 DP Slave

4.10.1 Slave Configuration Structure

4.10.1.1 Declaration

```
typedef struct
{
WORD slvCntCfg;                /* Control/configuration for DP slave function - see
                                below */

BYTE slvRxDataLen;             /* Receive data length (data from master), (0-244) */
BYTE slvTxDataLen;             /* Transmit data length (data to master) , (0-244) */
} PFB_SLV_CFG;

/* definitions related to slvCntCfg */

SLV_CTL_DIAG_UPD               0x0001  /* request diagnostic read from master */
SLV_CTL_EVT_RX_CHG             0x0002  /* generate event (interrupt) when RX data to slave
changes */

SLV_CTL_FORCE_READY_          0x0004  force response time (pfbReadyTime) and ignore what
TIME                           the master sends */

SLV_CTL_IGN_SYNC_             0x0008  /* don't generate error if master request sync
FRZ_ERR                        and/or freeze */

SLV_CTL_RX_BYTE_SWAP          0x0010  /* swap upper and lower bytes of RX data */
SLV_CTL_EVT_UPDTE             0x0020  /* generate event (interrupt) when slave is
updated */

SLV_CTL_EVT_MODE_             0x0040  /* generate event (interrupt) when slave mode changes
CHANGE                         (master RUN/STOP) */

SLV_CTL_IGNORE_STS            0x0080  /* ignore status of slave (no event) */
SLV_CTL_DIS_LED               0x4000  /* disable status led for DP slave function */
SLV_CTL_ENABLE                0x8000  /* enable PFB to act as DP slave */
```

4.11 Slave Status Structure

4.11.1.1 Declaration

```
typedef struct
{
WORD slvCntCfg;                /* Control/configuration for DP slave function -
                                see below */

BYTE slvStatus;                /* Status for DP slave function (Host Read Only) -
                                see below */

BYTE slvError;                 /* Error indication for DP slave function -
                                see below */

BYTE slvEvent;                 /* Event Flags for DP slave function -
                                see below */

BYTE slvDiagEvent;             /* Diagnostic Event Flags for DP slave function -
                                see below */

BYTE slvReqRxDataLen;          /* Receive data length requested by master */
BYTE slvReqTxDataLen;          /* Transmit data length requested by master */
} PFB_SLV_STS;

/* definitions related to slvCntCfg */

SLV_CTL_DIAG_UPD      0x0001  /* request diagnostic read from master */
SLV_CTL_EVT_RX_CHG    0x0002  /* generate event (interrupt) when RX data to slave
                                changes */

SLV_CTL_FORCE_READY_  0x0004  /* force response time (pfbReadyTime) and ignore
TIME                  what the master ends */

SLV_CTL_IGN_SYNC_     0x0008  /* don't generate error if master request sync and/or
FRZ_ERR               freeze */

SLV_CTL_RX_BYTE_SWAP  0x0010  /* swap upper and lower bytes of RX data */

SLV_CTL_EVT_UPDTE     0x0020  /* generate event (interrupt) when slave is
                                updated */

SLV_CTL_EVT_MODE_     0x0040  /* generate event (interrupt) when slave mode changes
CHANGE                (master run/stop) */

SLV_CTL_IGNORE_STS    0x0080  /* ignore status of slave (no event) */

SLV_CTL_DIS_LED       0x4000  /* disable status led for DP slave function

SLV_CTL_ENABLE        0x8000  /* enable PFB to act as DP slave */
```



```
/* definitions related to slvStatus */

SLV_STS_RUN_MODE      0x40      /* we're being scanned in run mode */
SLV_STS_OK            0x80      /* current slave status is OK */

/* definitions related to slvError */

SLV_ERR_ID_MISM       0x01      /* ID from master does not match configured ID */
SLV_ERR_READY_TIME_   0x02      /* pfbReadyTime does not match what master sent */
MISM

SLV_ERR_UNSUP_REQ     0x03      /* PROFIBUS is requesting Freeze or Sync,
                                which is not supported */

SLV_ERR_RX_LEN_MISM   0x04      /* length of data from master to us incorrect */
SLV_ERR_TX_LEN_MISM   0x05      /* length of data from us to master incorrect */
SLV_ERR_WD_FACT_INV   0x06      /* slvWdFact1 or slvWdFact2 from master was 0 */
SLV_ERR_TIME_OUT      0x07      /* slave watchdog timeout (check response timeout) */
SLV_ERR_WARN_WD_DIS   0x08      /* slave timeout watch dog disabled from master */

/* definitions related to slvEvent */

SLV_EVT_UPD           0x01      /* master has updated us */
SLV_EVT_RX_DATA_CHG   0x02      /* RX data from master has changed */

/* definitions related to slvDiagEvent */

SLV_DEVT_DIAG_UPD     0x01      /* master has read diagnostic information */
```

4.12 Master Class II

4.12.1 Master Class II Slave Configuration Structure

4.12.1.1 Declaration

```
typedef struct
{
    BYTE    lay2mCntCfg;           /* Control and configuration register */
    WORD    lay2mUpdTime;         /* update interval for periodic * 1ms (1-16380,
                                0=cyclic) */

    WORD    lay2mErrTime;         /* retry interval for periodic if error occurs *
                                1ms */

    BYTE    lay2mDstStn;          /* destination station address */
    BYTE    lay2mDstSap;          /* destination SAP (0xff to disable) */
    BYTE    lay2mSrcSap;          /* source SAP (0xff to disable) */
    BYTE    lay2mFrmCntrl;        /* frame control byte */
} PFB_MC2_SLV_CFG;

/* definitions related to lay2mDstSap */

MC2_SLV_SETADDR      55
MC2_SLV_RD_IP        56
MC2_SLV_RD_OP        57
MC2_SLV_RD_CFG       59
MC2_SLV_RD_DIAG      60
```

4.13 Master Class II Slave Read Structure

4.13.1.1 Declaration

```
typedef struct
{
    BYTE    lay2mState;           /* Current state of message block */
    BYTE    lay2mStatus;         /* Status register (Host only Reads) */
    BYTE    lay2mError;          /* Error register */
    BYTE    lay2mEvent;          /* Events */
    BYTE    lay2mRspStatus;       /* Response Status if not OK */
    BYTE    lay2mRxLen;          /* Length of the response data */
} PFB_MC2_SLV_RD;
```

4.13.2 Master Class II Set Slave Address Structure

4.13.2.1 Declaration

```
typedef struct
{
    BYTE    lay2mState;           /* Current state of message block */
    BYTE    lay2mStatus;         /* Status register (Host only Reads) */
    BYTE    lay2mError;          /* Error register */
    BYTE    lay2mEvent;          /* Events */
    BYTE    lay2mRspStatus;       /* Response Status if not OK */
    BYTE    pfbNewSlvAddr;       /* The new slave address */
    WORD    pfbSlvProfiID;       /* The slaves profibus ID */
    BYTE    pfbSlvNoAddrChg;     /* Prevent future changes before next reset */
    BYTE    pfbRemDataLen;       /* Len of the optional data for the slave */
} PFB_MC2_SETSLV_ADDR;
```

4.14 Master Class II Master Configuration Structure

4.14.1.1 Declaration

```
typedef struct
{
    BYTE    mc2CntCfg;                /* The control and configuration options for
                                      Master Class II */

    WORD    mc2PollTime;              /* The update interval for retries * lms (1-16380) */
    WORD    mc2PollLimit;             /* The maximum number of retries */
    BYTE    mc2DstStn;               /* The destination station address */
} PFB_MC2_MASTER_CFG;

/* definitions related to mc2CntCfg */

MC2_CTL_UNCONF        0x01
MC2_CTL_HI_PRI        0x04
MC2_CTL_EVT_CONFIRM   0x20          /* generate event (intr) when this message is
                                     confirmed */

//Master Class 2 Services Definitions

SYSTEM_DIAGS          0x7E
MASTER_STATUS          0x7F
DATA_XFER_LIST        0x80
MASTER_C1_STOP        0x40
MASTER_C1_CLEAR       0x80
MASTER_C1_ON          0xc0
// Macros
TEST_BITLIST(mem, stn) mem[stn/8]&(0x01<<(stn-(((unsigned char)(stn/8))*8)))
```

4.15 Master Class II Read Master Data Structure

4.15.1.1 Declaration

```
typedef struct
{
    BYTE  mc2State;           /*The state of the current transaction */
    BYTE  mc2Status;         /* The status of the last transaction */
    BYTE  mc2Error;          /* The last error code */
    BYTE  mc2Event;          /* The last event code */
    BYTE  mc2RspSts;         /* The code returned by the Master Class I target */
    BYTE  mc2RxLen;          /* The length of the returned message */
} PFB_MC2_MASTER_RD;

/* definitions related to mc2State */

MC2_STE_DISABLE      0x00
MC2_STE_BUSY         0x01
MC2_STE_ENABLE       0x02
MC2_STE_ACTIVE       0x03
MC2_STE_DONE         0xFF

/* definitions related to mc2Status */

MC2_STS_OK           0x80

/* definitions related to mc2Error */

MC2_ERR_NOT_OK       0x01
MC2_ERR_RX_LEN       0x02
MC2_ERR_TX_LEN       0x03
MC2_ERR_POLL_TOUT    0x04

/* definitions related to mc2Event */

MC2_EVT_CONFIRM      0x01
```

4.16 Master Class II Read Block Structure

4.16.1.1 Declaration

```
typedef struct
{
    BYTE    mc2CntCfg           /* The control and configuration options for
                               Master Class II */

    BYTE    mc2State           /* The state of the current transaction */
    BYTE    mc2Status;         /* The status of the last transaction */
    BYTE    mc2Error;          /* The last error code */
    BYTE    mc2Event;          /* The last event code */
    BYTE    mc2RspSts;         /* The code returned by the Master Class I target */
    BYTE    mc2TxLen;          /* The length of the transaction request
                               message buffer */

    BYTE    mc2RxLen;          /* The length of the transaction response
                               message buffer */

    WORD    mc2PollTime;       /* The update interval for retries * lms (1-16380) */
    WORD    mc2PollLimit;      /* The maximum number of retries */
    BYTE    mc2DstStn;         /* The destination station address */
} PFB_MC2_MASTER_BLK;
```

4.16.2 Master Class II Master Write Structure

4.16.2.1 Declaration

```
typedef struct
{
    BYTE    mc2State;          /* The state of the current transaction */
    BYTE    mc2Status;         /* The status of the last transaction */
    BYTE    mc2Error;          /* The last error code */
    BYTE    mc2Event;          /* The last event code */
    BYTE    mc2RspSts;         /* The code returned by the Master Class I target */
    BYTE    mc2TxLen;          /* The length of the transaction request
                               message buffer */
} PFB_MC2_MASTER_WRT;
```



Warranty and Support

Appendix Contents:

- Warranty
- Technical Support

A.1 Warranty

For warranty information, refer to

http://www.molex.com/images/woodhead/woodhead_limited_warranty.pdf

A.2 Technical Support

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

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

A.2.1 Getting Help

Technical support is available during regular business hours by telephone, fax or email from www.Molex.com. Documentation and software updates are also available on the Web site.



Note

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

North America

Canada:

Tel: +1-519-725-5136

Fax: +1-519-725-1515

Email: WoodheadSupportNA@molex.com

Europe

France:

Tel: +33 2 32 96 04 22

Fax: +33 2 32 96 04 21

Email: ic.support.eu@molex.com

Germany:

Tel: +49 7252 9496 555

Fax: +49 7252 9496 99

Email: WoodheadIC.SupportDE@molex.com

Italy:

Tel: +39 010 5954 052

Fax: +39 010 5954 052

Email: WoodheadIC.SupportIT@molex.com

Other countries:

Tel: +33 2 32 96 04 23

Fax: +33 2 32 96 04 21

Email: WoodheadIC.SupportEU@molex.com

Asia-Pacific

Japan:

Tel: +81 46 265 2428

Fax: +81 46 265 2429

Email: WoodheadIC.SupportAP@molex.com

Singapore:

Tel: +65 6268 6868

Fax: +65 6264 6055

Email: WoodheadIC.SupportAP@molex.com

China:

Tel: +86 21 5835 9885

Fax: +86 21 5835 9980

Email: WoodheadIC.SupportAP@molex.com

For the most current contact details, please visit <http://www.molex.com>.