molex®
one company › a world of innovation

SST-PBMS-PCI

User Manual

Document Edition: 1.1

Document #: 715-0057

Document Edition: 1.1

Date: January 9, 2012

This document applies to the SST-PBMS-PCI interface cards.

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

Table of Contents

1 Introduction

1.1 Purpose of this Document

This document is a user's guide for the SST-PBMS-PCI interface card. This card makes it possible for an application running on a host computer to emulate and monitor DP slaves on a Profibus DP network.

1.2 Card Overview

The card can:

- emulate up to 125 DP slaves
- monitor up to 125 DP slaves

The card supports the standard Profibus baud rates of 9.6K, 19.2K, 93.75K, 187.5K, 500K, 1.5M, 3M, 6M, 12M, 31.25K and 45.45K baud.

The card has 512 Kbytes of RAM that is shared with the host in 16K pages. The host determines which 16 Kbyte page of this shared RAM is mapped into the host memory by writing to a register on the card. This block of memory contains all the tables and buffers that are used to pass information between the interface card and the application software running in the host computer.

In addition, the card has 512 Kbytes of sectored flash memory, for storage of the software module that the card processor runs. The host computer uses the utility provided to store a software module into flash memory and run the module from flash. Once a software module has been stored in flash, it does not need to be reloaded each time you run the card.

1.3 Reference Documents

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

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

The Profibus specification can be downloaded from the Siemens bulletin board at (423) 461 2751.

1.4 Specifications

Part Number	SST-PBMS-PCI
Function	Interface card for Profibus DP networks
Description	• Intel 80186 processor • 512 Kbytes of onboard shared memory, accessible from the host computer in 16K pages 512 Kbytes of sectored flash memory
Current Consumed	maximum 700 mA at 5V
Environmental	operating temperature 0-50 degrees Celsius
Card connectors	• standard Profibus DB-9 connector • Phoenix Combicon connector, part number MSTB 2.5/5-ST-5.08

2 CARD INSTALLATION

2.1 Preparing the Computer

The SST-PBMS-PCI requires resources in the host computer, including:

- 16 I/O port addresses (highest 8 ports are reserved for future use)
- 64 Kbytes of memory (first 16K is used, the rest of this region is reserved for future use)

2.2 Handling Precautions

The SST-PBMS-PCI interface card contains components that are sensitive to electrostatic discharge. Do not remove the card from its protective bag without using the following precautions:

- Before handling the card, ground yourself by touching a grounded object, such as the case of your computer.
- Never touch the backplane connectors or pins. Handle the card by its mounting bracket.
- Always store the card in its protective bag.

2.3 Card Switches

2.3.1 Flash Write Enable Switch

Switch SW4 (Flash Write Enable) controls whether the host can write to flash memory. To enable writing to flash, place the switch in the “ON” position. To disable writing to flash, place the switch in the “OFF” position. The default is to enable writing to flash memory.

This feature can be used to protect the flash from being upgraded by the user.

2.3.2 Switches SW2 and SW5

These switches are used at the factory to initialize the hardware on the card DO NOT MOVE THESE SWITCHES FROM THE NORM POSITION.

2.4 Installing the Card

To install the card in your computer:

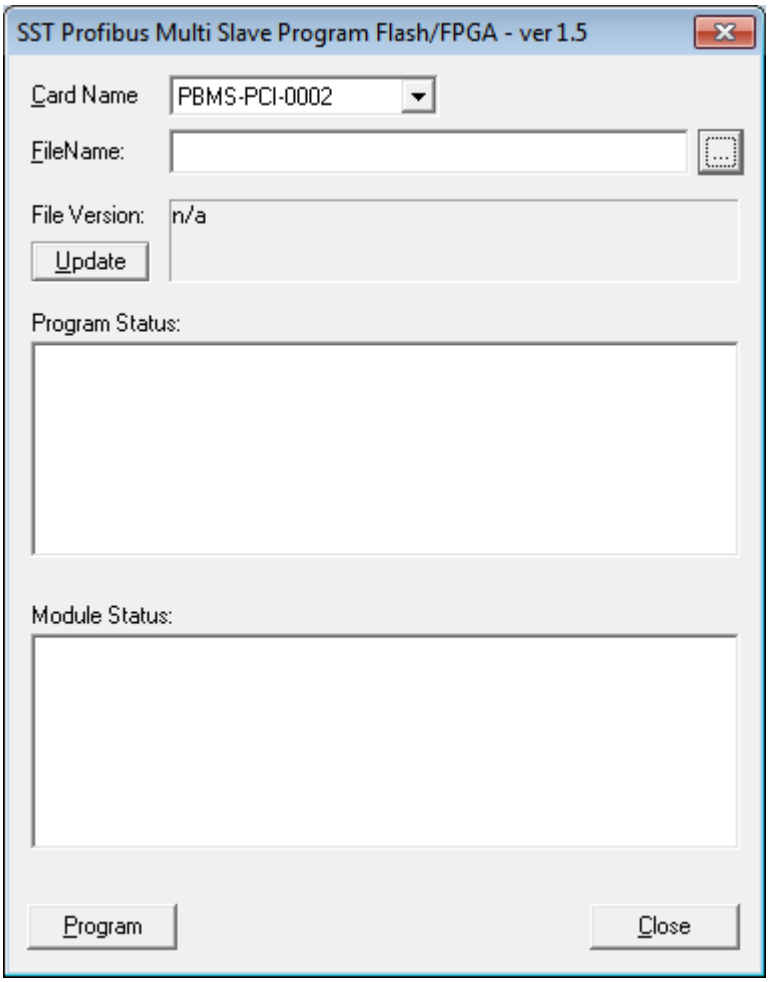
1. Ensure that all power to your computer is off.
2. Adequately ground yourself, as cautioned in Section 2.2, Handling Precautions.
3. Unplug the power cord and any network cables.
4. Remove the computer cover. Consult your computer user's guide for information on installing add-in boards.
5. Take the card out of its shipping container and anti-static bag, being careful not to touch any of the connectors or pins.
6. Firmly press the card into a compatible PCI-compliant connector.
7. Secure the card using the screws provided. Re-connect any items unplugged in Step 3.
8. Replace the computer cover and power up the machine.

2.5 Loading Software Modules to the Card

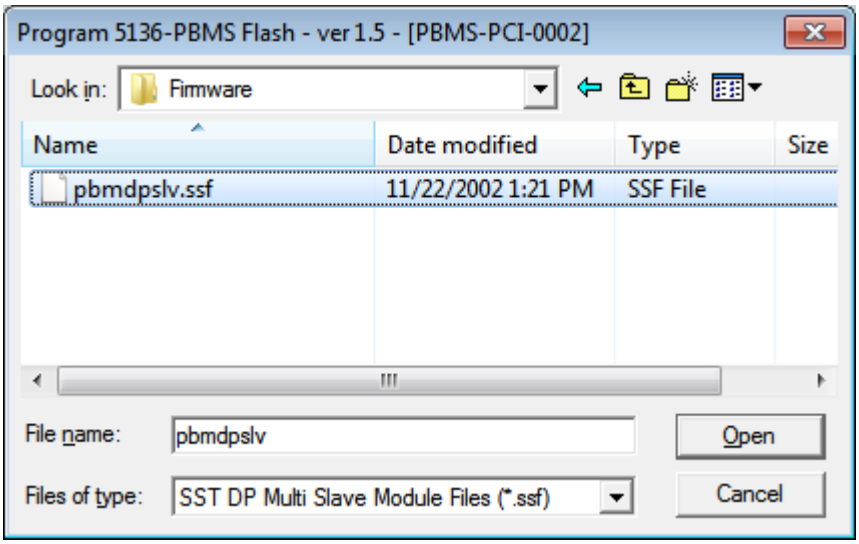
The host computer downloads the appropriate interface software to the flash memory on the card, then runs the card from flash. Normally you need to load flash memory only when you first receive the card and whenever you receive an update from Molex. Once a module has been stored in flash, it remains there indefinitely.

There are two parts of the card software that need to be maintained in flash, the card executable module, pbmdpslv.ssf, and the FPGA configuration file, mslvpga.ssp.

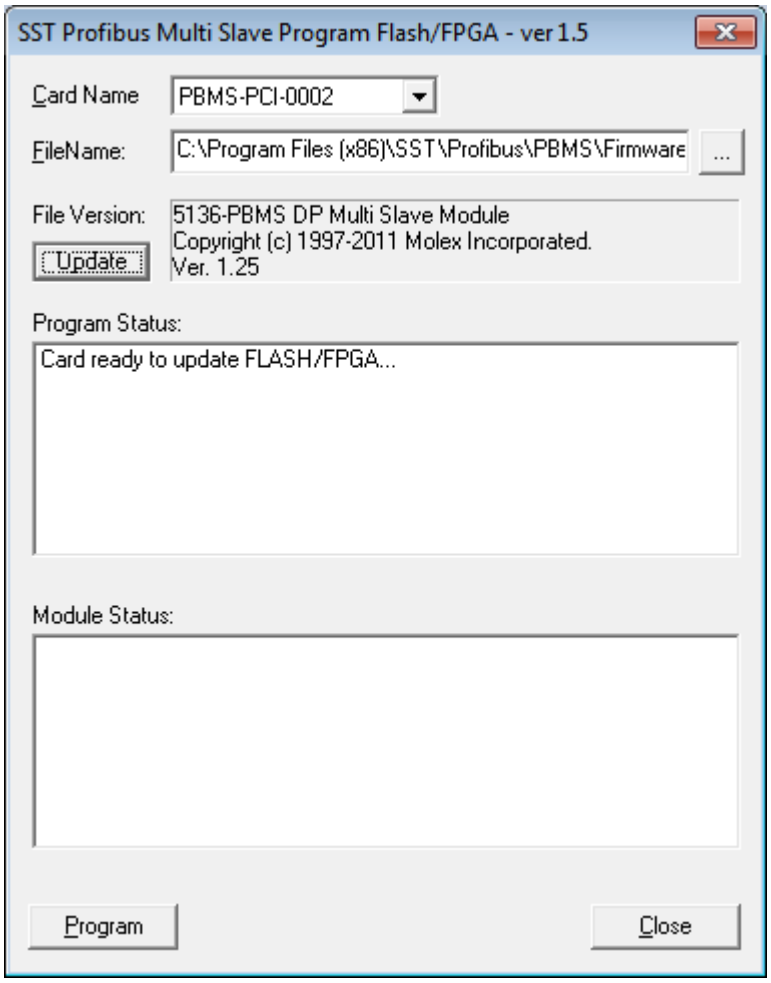
To upgrade firmware or FPGA on the SST-PBMS-PCI card use the PBMS Program Flash - FPGA sample program. (Refer to chapter 3 “Installing the Card Software” to install the Flash/FPGA upgrade software). Once installed, the update program is located under: Start Menu/Programs/SST Profibus/PBMS Program Flash – FPGA.



Click on the “…” button to browse for the firmware file PBMDPSLV.SSF.

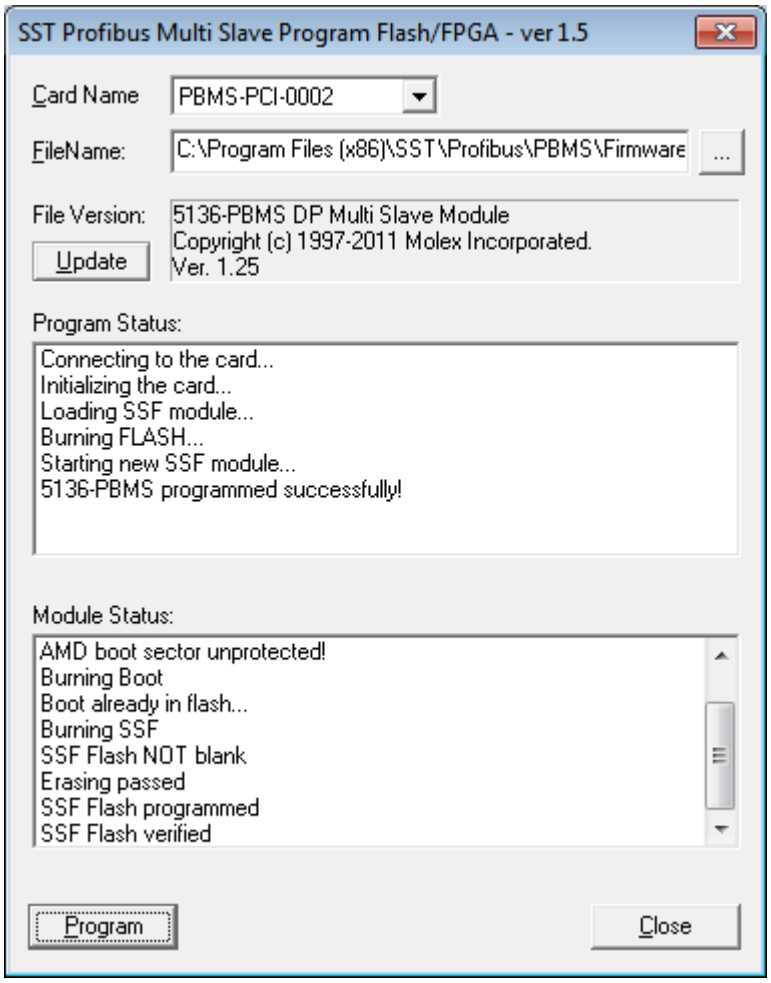


Select this file and click Open Button.

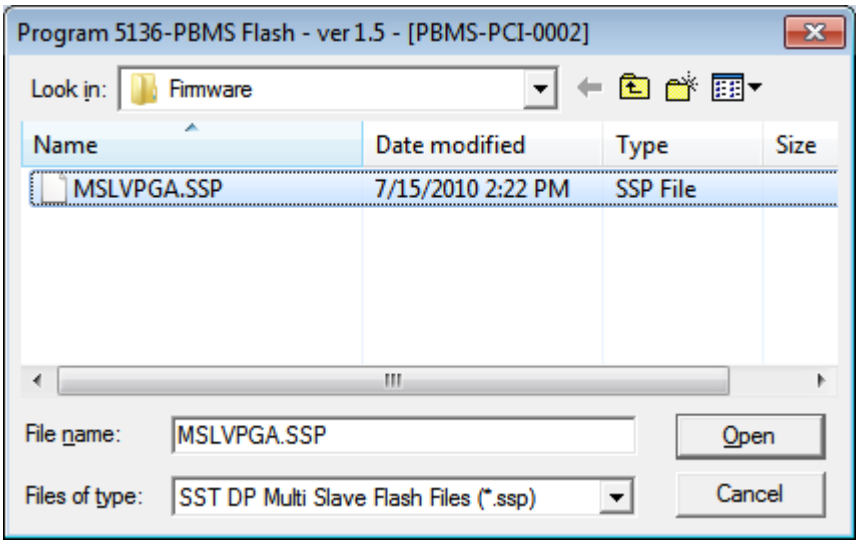


Click on Update button to verify the firmware version, before programming flash with this new firmware. Click on Program button to complete the burn process.

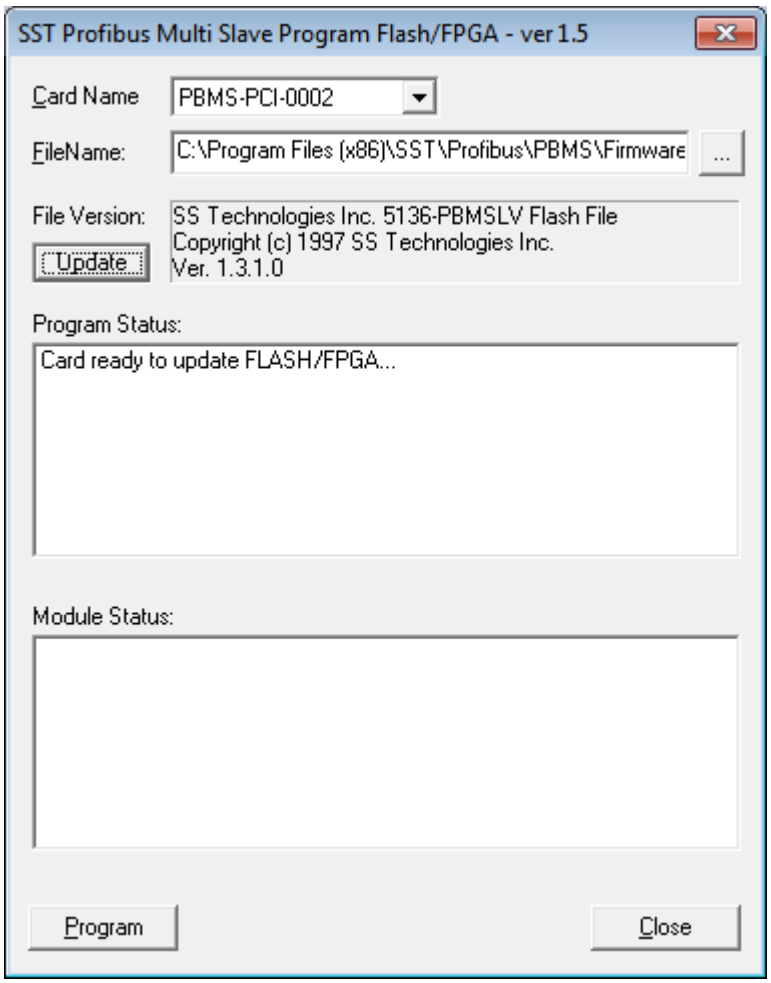
The following message is displayed if the flash was successfully written.



To upgrade the FPGA file, change “files of type” to *.SSP and select MSLVPGA.SSP and click on Open Button.

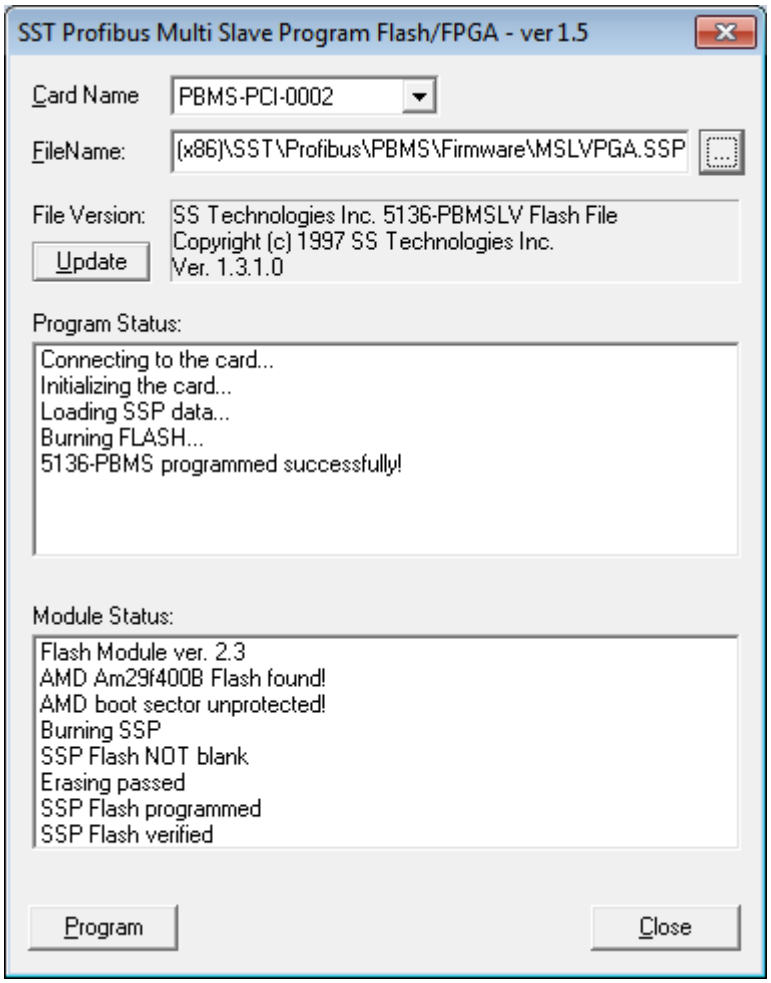


To upgrade the FPGA file, change “files of type” to *.SSP and select MSLVPGA.SSP and click on Open Button.

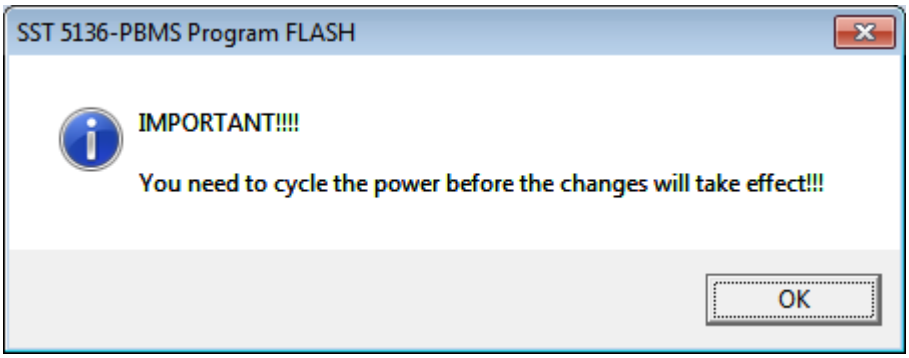


Click on Update button to verify the FPGA version, before programming flash with this new file. Click on Program button to complete the burn process.

The following message is displayed if the flash was successfully written.



After successfully programming the flash, the following dialog will be displayed.



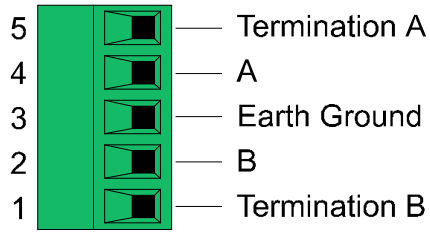
A complete Power cycle is required for the card to use the latest FPGA file.

2.6 Connecting to a Network

The card contains a standard Profibus DB9 connector that can be connected to a Profibus bus terminal.

Pin	Meaning
1	chassis ground
2	reserved
3	data +
4	Tx enable
5	Isolated ground
6	voltage plus
7	reserved
8	data -
9	reserved

The card also contains a 5-pin Phoenix Combicon connector to allow a direct connection to a Profibus network. The connector pins are:



To connect the internal terminators to the line, connect jumpers between 1 and 2, and between 4 and 5.

The DB9 and Phoenix connectors are internally connected. Connecting the terminators using the Phoenix connector also connects them to the DB9 connector.

The two physical ends of the network should be terminated. There should be two and only two terminators on a network.

The recommended cable is Belden 3079A. Examples include:

Siemens 6XV1 830-0AH10 Two Core Shielded

Siemens 6XV1 830-0BH10 w/PE Sheath

Siemens 6XV1 830-3AH10 for underground burial

Siemens 6XV1 830-3BH10 trailing cable

Bosch Comnet DP #913 548 Flexible PROFIBUS Cable

Bosch Comnet DP #917 201 Trailing PROFIBUS Cable

Bosch Comnet DP #917 202 Massive PROFIBUS Cable

Allen-Bradley blue hose, which has an impedance of 78 ohms, is not recommended.

2.7 The Card LEDs

There are two (2) LEDs on the card. They are visible through the mounting bracket at the back of the card.

The upper LED (closer to the network connectors) is the system status (SYS) LED. The lower LED is the communication status (COMM) LED.

The communication status LED is green whenever the card is transmitting. The only exception is when the card is acknowledging an FDL status request. If the card encounters any type of reception error, the LED goes red for at least one second.

The system status LED shows the current state of operations on the SST-PBMS-PCI. If the software module on the card has not been run, the LED is on and red. After you run the module but before the card is online, the LED is off. If the card is running and online, the LED is red if one or more slaves (monitored or virtual) is in error. The LED is green if all non-ignored slaves are OK. If the LED is yellow, some of the slaves are being scanned in stop mode.

When you configure each slave, you can tell the card to ignore the status of that slave on the LED. Refer to section 5.2.3 for information on how to do this.

The card also uses the LEDs to signal internal errors. If an internal error occurs, the card flashes the system status LED once red, then flashes an 8-bit error code sequentially on the communication status LED, from low bit to high bit. Red indicates the bit is zero, green indicates the bit is 1. Then the cycle repeats. Record the sequence for technical support.

2.8 Troubleshooting Installation

This section describes what to do if:

- The card cannot communicate on a network.

It also provides more detailed information on some common sources of problems.

2.8.1 Card loads correctly but does not go active on the network

- Check cabling for correct wiring to the card
- Check for shorted wires, leads on terminating resistors shorted to cables, strands from the shield shorting to the other wires.
- Check the baud rate
- Check network termination. Only the two nodes at the physical ends of the network should have terminating resistors.

2.9 Card Registers

The information in the following sections is provided here for reference and may be needed if you are writing your own application for the card.

2.9.1 Register Offsets

Five read/write registers are used in the host I/O map. The base address for these registers is the port address assigned to the card by the PCI host. The following table lists the registers.

Register Offset	Register Name
0	Card Command Register (CCM)
2	Card Configuration Register (CCR)
3	Memory Configuration Register (MCR)
4	Reserved Reads 0x00. The host should not write to this register
7	Card ID Register (IDR)

For example, if the base port address assigned by the PCI host 0x250, the CCM is at address 250, the CCR is at address 252, the MCR is at address 253, the ICR is at address 254, and the IDR is at address 257.

2.9.2 The Card Command Register (CCM)

The Card Command register contains bits that:

- enable and disable card memory

7	6	5	4	3	2	1	0
0	0	0	0	MEM_ ENA	MEM _DIS	0	0
R	R	R	R	R/W	R/W	R	R

Bits 2 and 3 enable and disable the card memory.

Bit 3	Bit 2	
1	0	enable memory
0	1	disable memory
0	0	no effect
1	1	no effect

Bits 0, 1, 4, 5, 6, and 7 are not used. They always read as 0. You should always write 0 to these bits.

At power-up, the CCM contains 4.

2.9.3 The Card Configuration Register (CCR)

The Card Configuration register contains bits that:

- control whether the card processor runs
- control whether the card processor runs from flash memory or shared memory
- select which of the four memory configuration registers is visible from the host computer

7	6	5	4	3	2	1	0
uPROC_RUN	0	0	SH_BOOT	0	0	MCR_Sel1	MCR_Sel0
R/W	R	R	R/W	R	R	R/W	R/W

Bits 0 and 1 determine which Memory Configuration register you see (refer to section 2.9.4 below).

Bit 1	Bit 0	Register	Use
0	0	MCR0	Page register
0	1	MCR1	-
1	0	MCR2	-
1	1	MCR3	Address compare

Set bit 4 to 1 to run the processor from shared RAM. If bit 4 is 0, the processor runs from flash.

Set bit 7 to 1 to run the processor on the card. If bit 7 is 0, the processor is reset.

Bits 2, 3, 5, and 6 always read as 0. You should always write 0 to these bits.

At power-up, the CCR contains 0.

2.9.4 The Memory Configuration Register (MCR)

The Memory Configuration register is actually four registers. You use bits in the Card Configuration register to select which of the four you currently see. The SST-PBMS-PCI card uses only MCR0 and MCR3. MCR0 selects the current page of card memory mapped into the host memory. MCR3 is used by the card's hardware for internal decode purposes.

Normally 0xD0 is written to MCR3 for internal decode purposes when the module is first run from flash. Then MCR0 is selected so that the host can switch card pages as required to access data for the various slaves.

MCR0

MCR0 is the page select register. It is also referred to as the Memory Page register (MPR).

7	6	5	4	3	2	1	0
DBL_ BNK	0	0	0	Page_ Sel_3	Page_ Sel_2	Page_ Sel_1	Page_ Sel_0
R/W	R	R	R	R/W	R/W	R/W	R/W

Bits 0, 1, 2, and 3 are used to select which 16 Kbyte page of card memory maps into the host computer memory.

Bits 4, 5, and 6 are not used and read as 0. You should always write 0 to these bits.

Bit 7 is used to select which 256 byte bank of card memory is used, but only while the card processor is reset. Set this bit to 1 to select the upper 256 byte bank; set it to 0 to select the lower 256 byte bank. When the card processor is running, the software on the card manages the bank switching and changing this bit from the host computer has no effect. Normally the only time you use this bit is when you are running the software module currently in flash.

MCR0 reads 0 at powerup.

MCR3

This register is referred to as the Memory Address Compare register or MCMP. After power-up, the Host must write 0xD0 to this register. Once this value is written, it does not need to be re-initialized unless power is cycled. This value is used by the card's hardware for internal decode purposes.

7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0
R	R/W	R/W	R/W	R/W	R/W	R	R

MCR3 reads 80 hex at power-up.

2.9.5 The Card ID Register (IDR)

The Card ID register can be used to identify the card. It contains the value ID_5136_PB_MSLV (2)

At power-up, the IDR contains 2.

2.10 Using Flash Memory

The SST-PBMS-PCI contains 512 Kbytes of sectored flash memory that is used to store the software module used by the card. You run the software module from flash.

To program a software module into flash, refer to section 2.5.

You cannot store slave configuration data in flash. You must configure the slaves from the host computer before you put the card online.

2.11 Running a Module from Flash

An application follows these steps to run a software module currently in flash.

Note: The SST Profibus Configuration Tool will perform the following steps automatically if it is used on the host system. These instructions are for users who wish to use the card in a non Windows based or, embedded systems.

1. Disable card memory.

2. Make sure the card processor is reset. Give the card processor time to ensure that it is reset.
3. Select the uppermost page of card memory. Select the higher 256 byte bank and select page 15 of card memory.
4. Write the value PROC_BUSY (0xa5) to the highest memory location on that page.
5. Run the card processor.
6. Wait up to 2 seconds for the card processor to change the value you wrote. If the value becomes CARD_OK (0x5a), everything has run correctly and there is a null-terminated copyright and module version string at offset 0x3f00 on the page. If the value becomes CARD_ERROR (0xee), the processor encountered a problem and there is a null-terminated error message vserion string at offset 0x3f00 on the page. If the value is unchanged (0xa5) or is any other value, the processor did not run correctly.
7. If the processor did not run successfully, reset the processor and disable card memory.
8. If the processor ran successfully, copy the copyright and vserion string from the card, then clear the uppermost location (set the value to RUN_MODULE = 0x00) to tell the card processor to run the module in flash.

In embedded applications, the card can be placed in a PC to load the PBMDPSLV.SSF module into flash memory on the card. Then the card can be placed in the embedded system and the embedded processor can run the module from flash, configure the slaves, and go online. If an update is required, the card can be removed from the embedded system, updated in a PC, then put it back in the embedded system.

3 Installing the Card Software

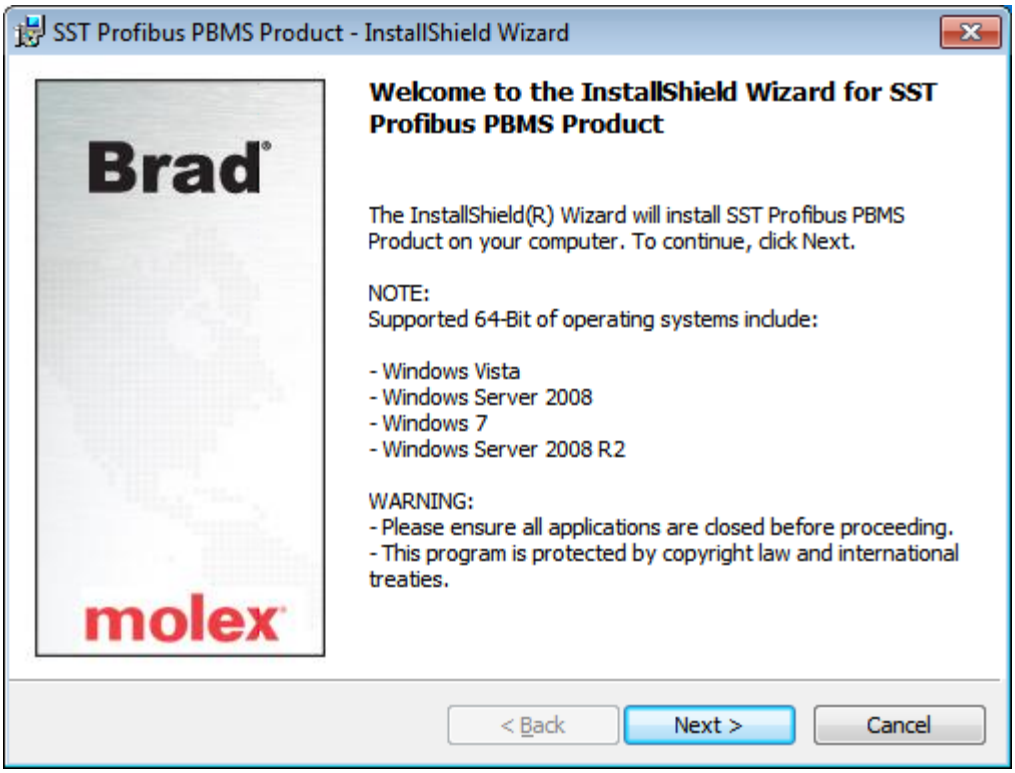
3.1 Introduction

This chapter describes how to install the SST Profibus configuration application, drivers, and tools.

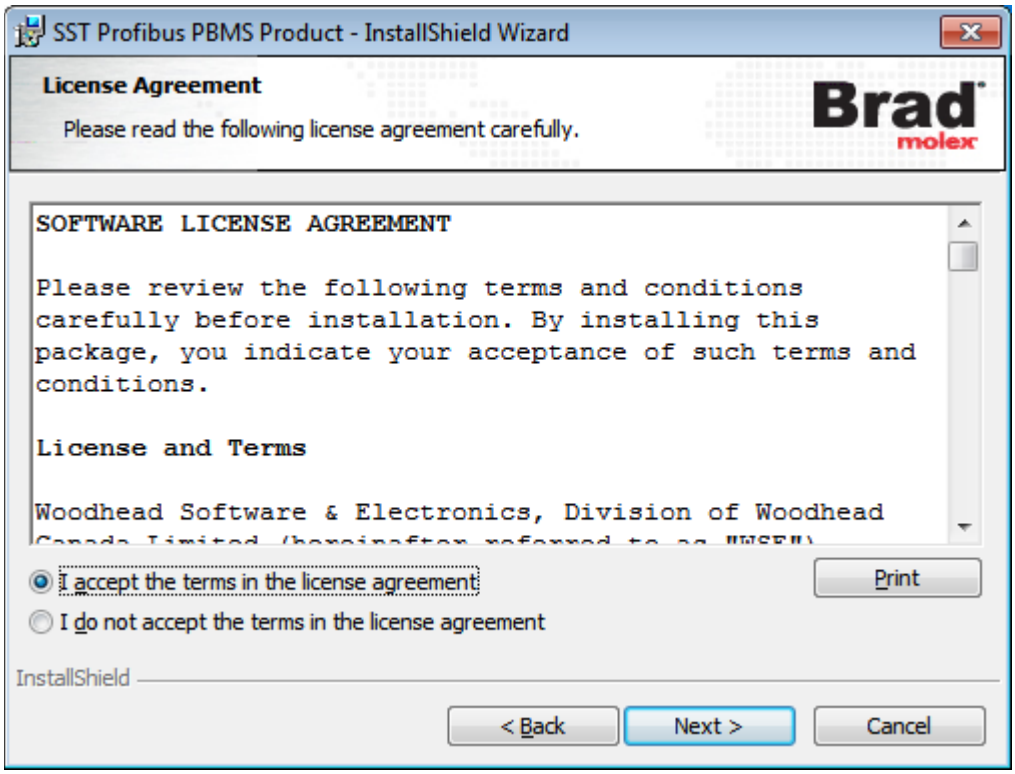
3.2 Installation

Follow the steps to install the card software:

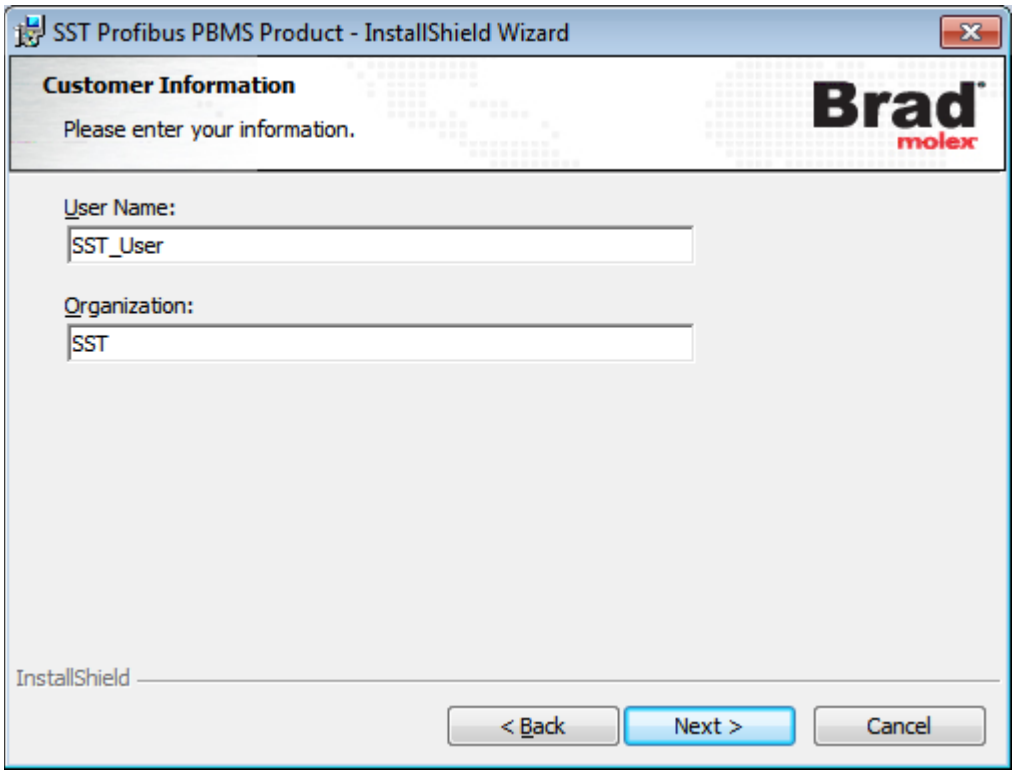
Insert the CD into the CD ROM. The following screen should be displayed.



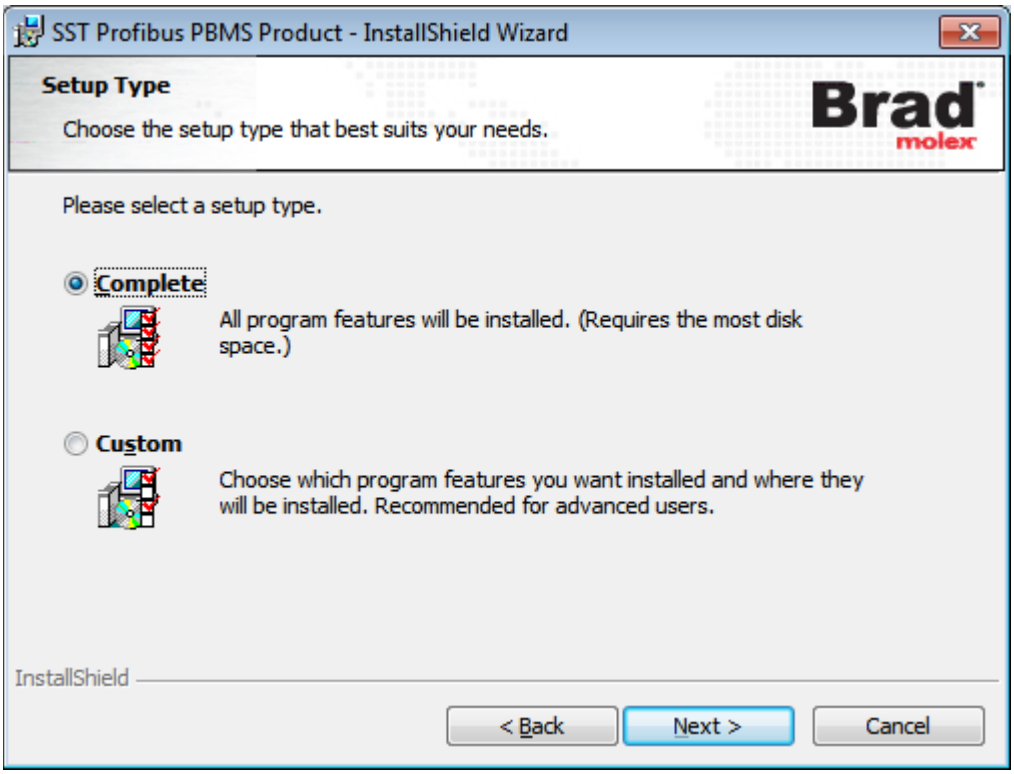
Click the “Next” button. The following dialog is displayed.



Read the “SOFTWARE LICENSE AGREEMENT”, and accept the terms in the license agreement, click “Next” to proceed with the installation.

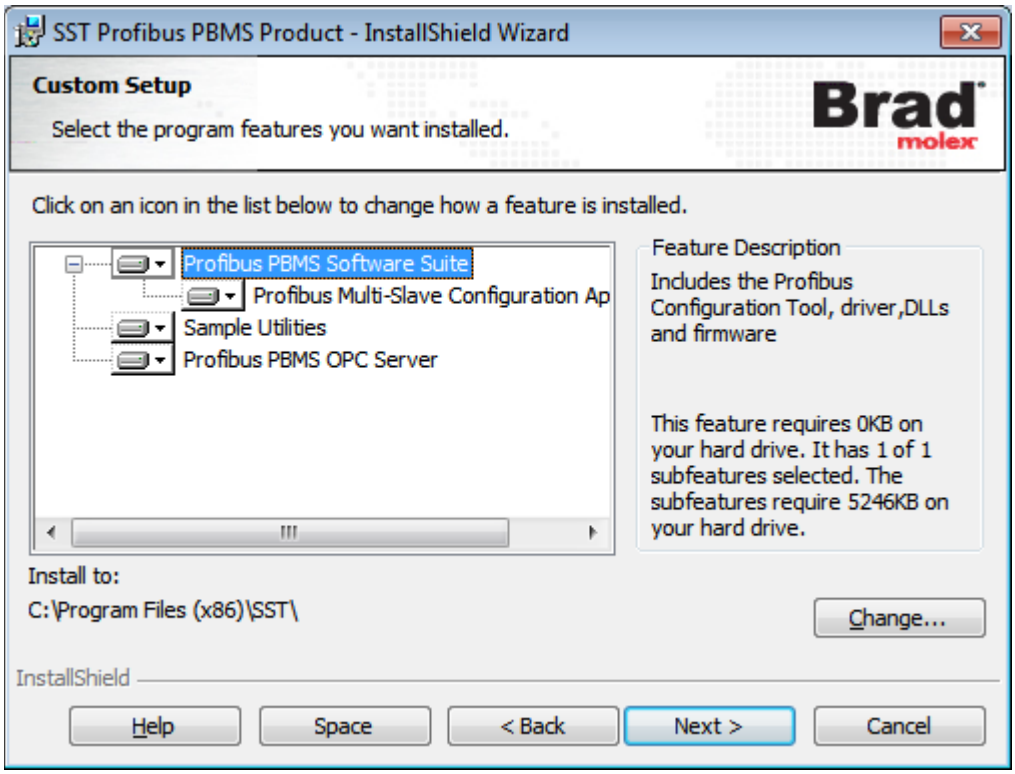


Enter the User Name and the Organization, click the “Next” button.

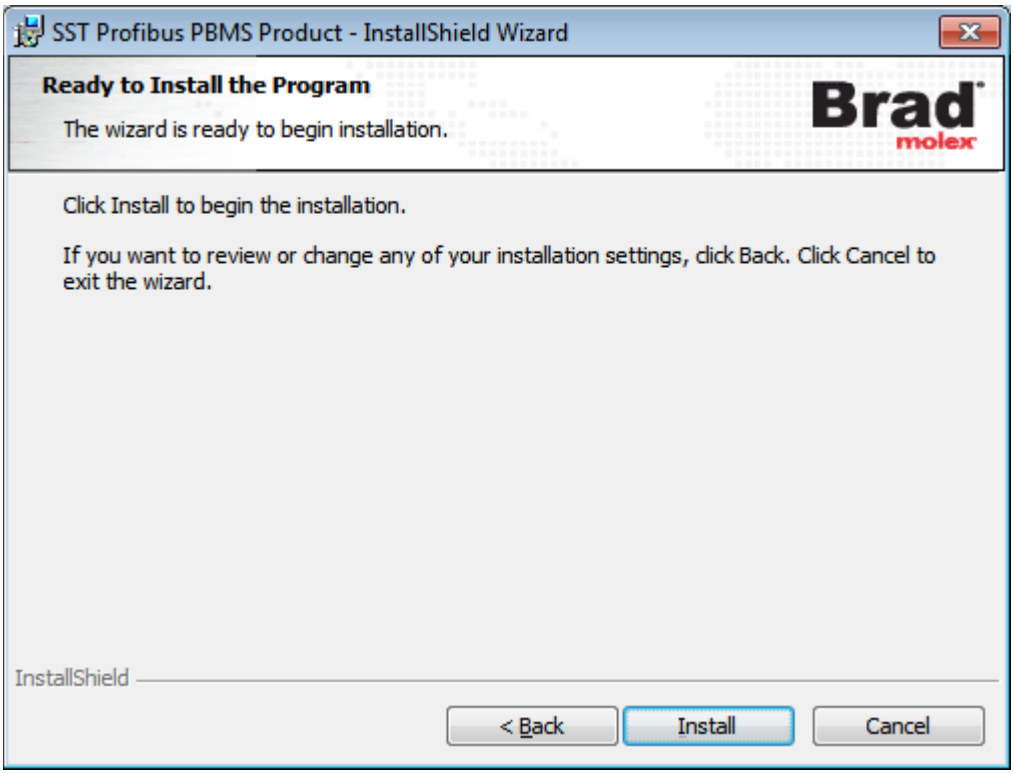


Choose Complete or Custom and click “Next”.

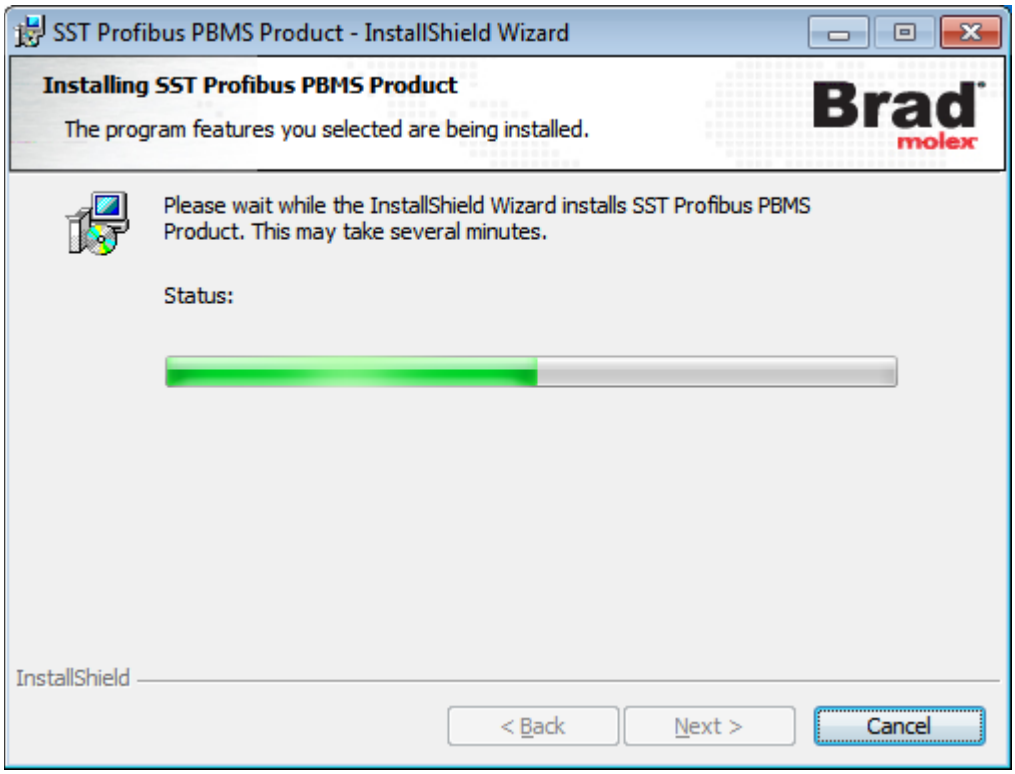
If Custom is selected, the following install page is displayed:



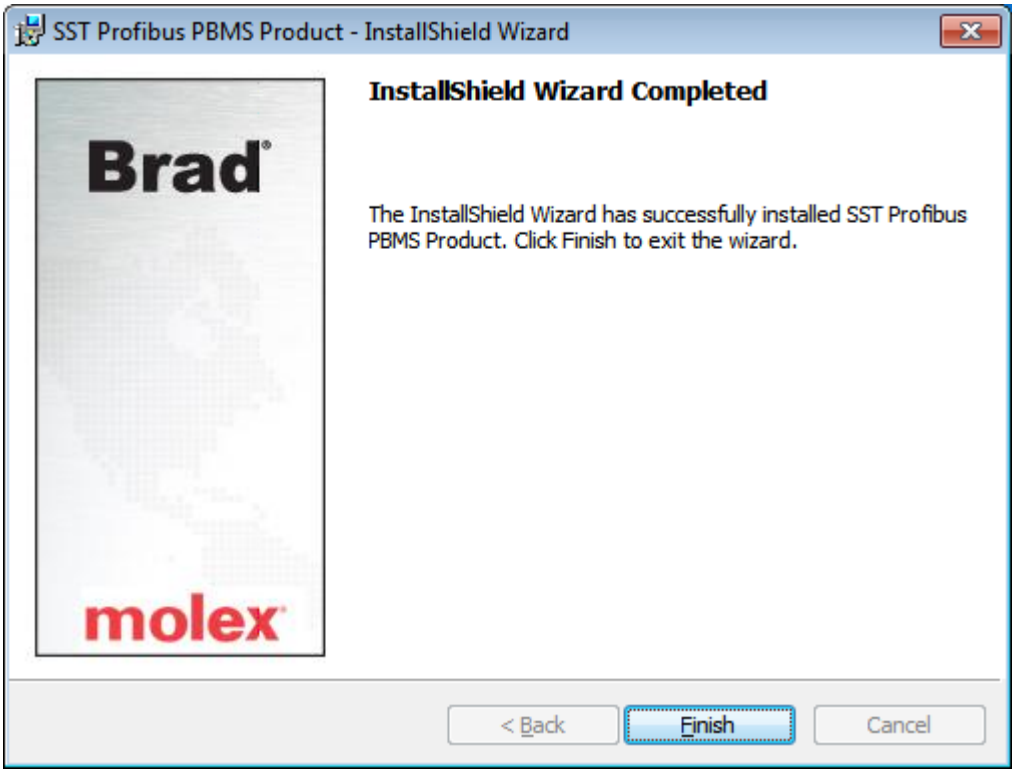
Select the components and change desired install directory you wish to install. click the “Next” button.



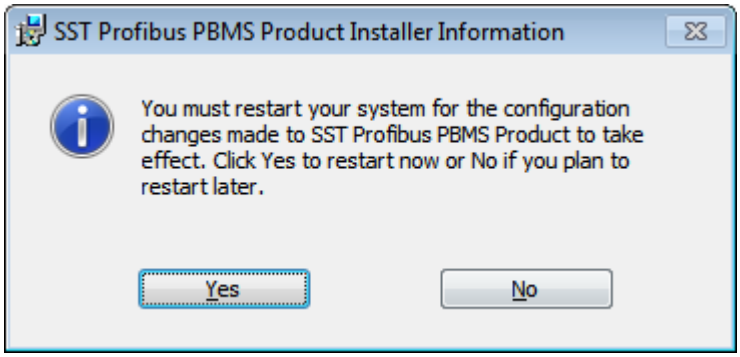
Click the “Install” button to proceed with the installation.



Setup will now perform the installation.



Click “Finish” to finish the installation.



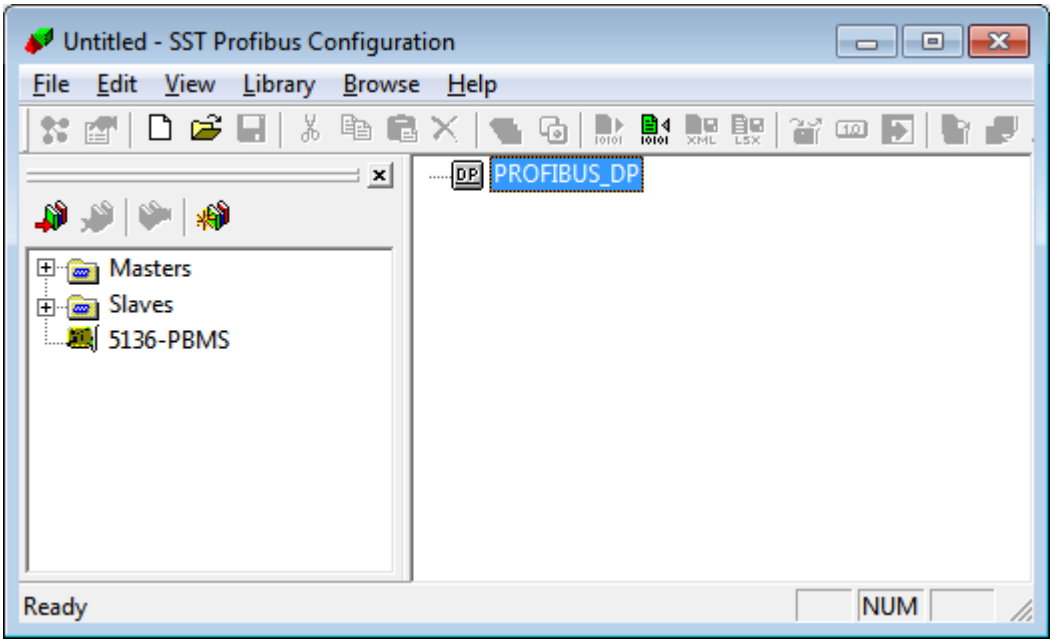
You must now reboot the computer before using the SST-PBMS-PCI software.

4 Configuring the SST-PBMS-PCI with the SST Profibus Configuration Tool

4.1 Configuring the SST-PBMS-PCI

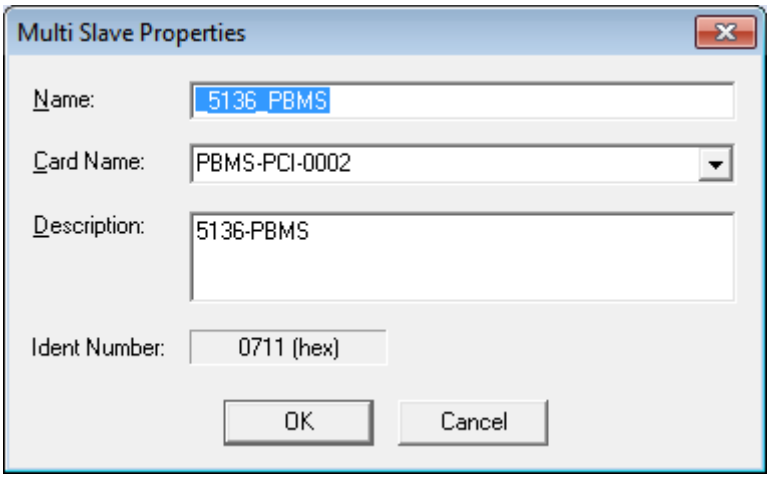
To configure the SST-PBMS-PCI in the SST-Profibus Configuration Tool, take the following steps:

1. Select 5136-PBMS from the Device Library.
2. Drag this device over to the network configuration tree window under **PROFIBUS DP**

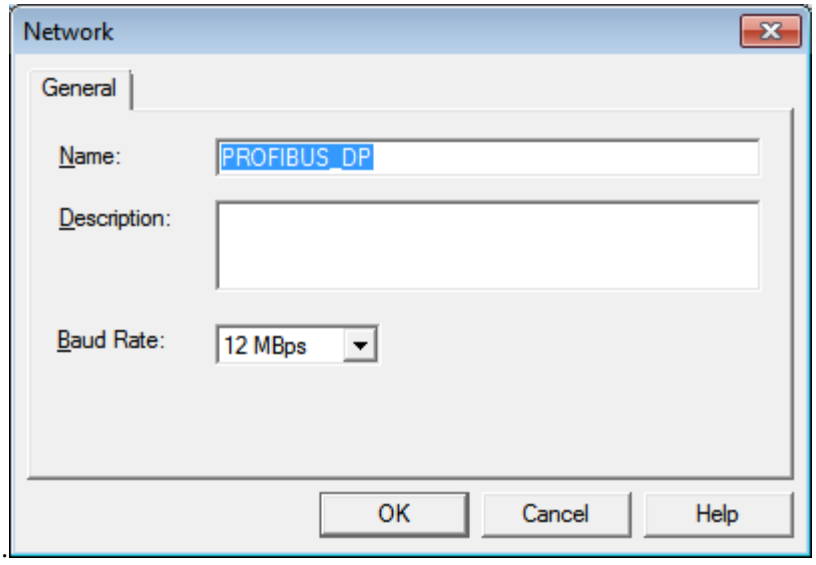


3. A window will appear where you're asked to select the card you want to use. If you have only one (1) of these cards in your system, accepting the default is okay since the configuration tool detects all cards automatically and displays them in Card Name List. You have the option of changing the Name field as you see fit.

   Click OK.

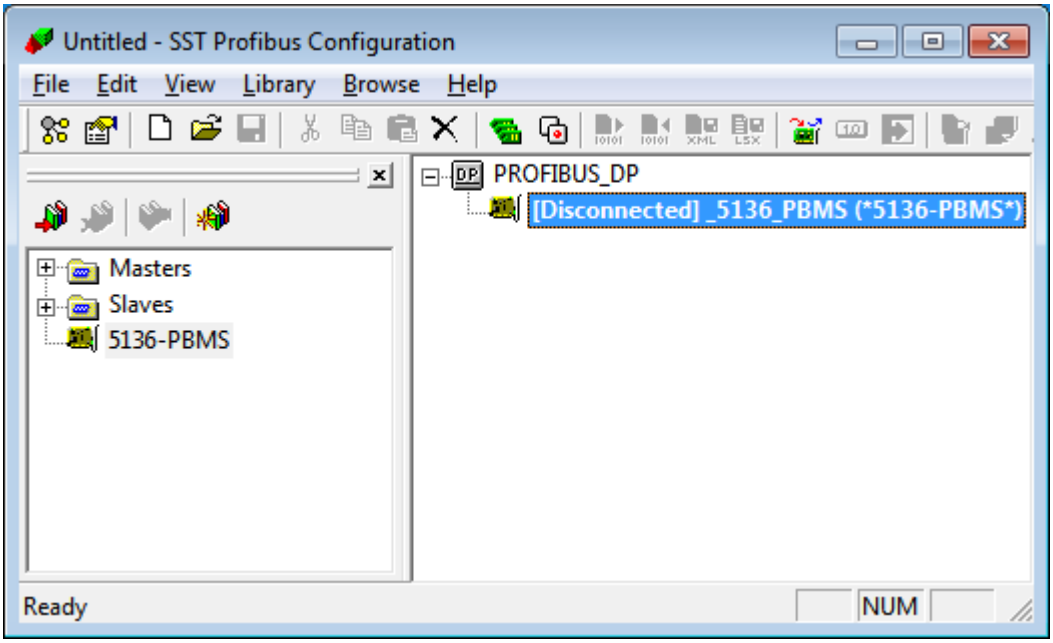


4. Configure the baud rate for the network by double clicking on **PROFIBUS_DP** in the network configuration window. The following screen will appear:

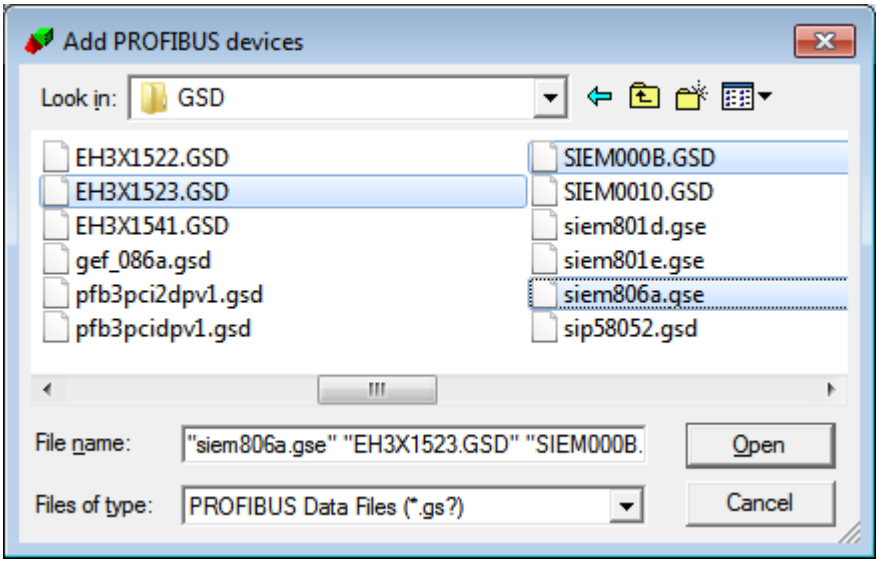


5. Click OK.

You will now need to configure your slave devices on the Profibus Multislave card. There are two (2) types of slaves you can configure: Virtual and Monitor. Virtual slaves will exist on this card and will be scanned by a DP Master. Monitored slaves are real slave devices that already exist on a Profibus network and configuring monitored slaves enables you to monitor these real slaves without disturbing them on the network. Before you can configure slaves you need the GSD files of all the slaves so that you can add them to the Device library of the Profibus configuration tool. Here's how:

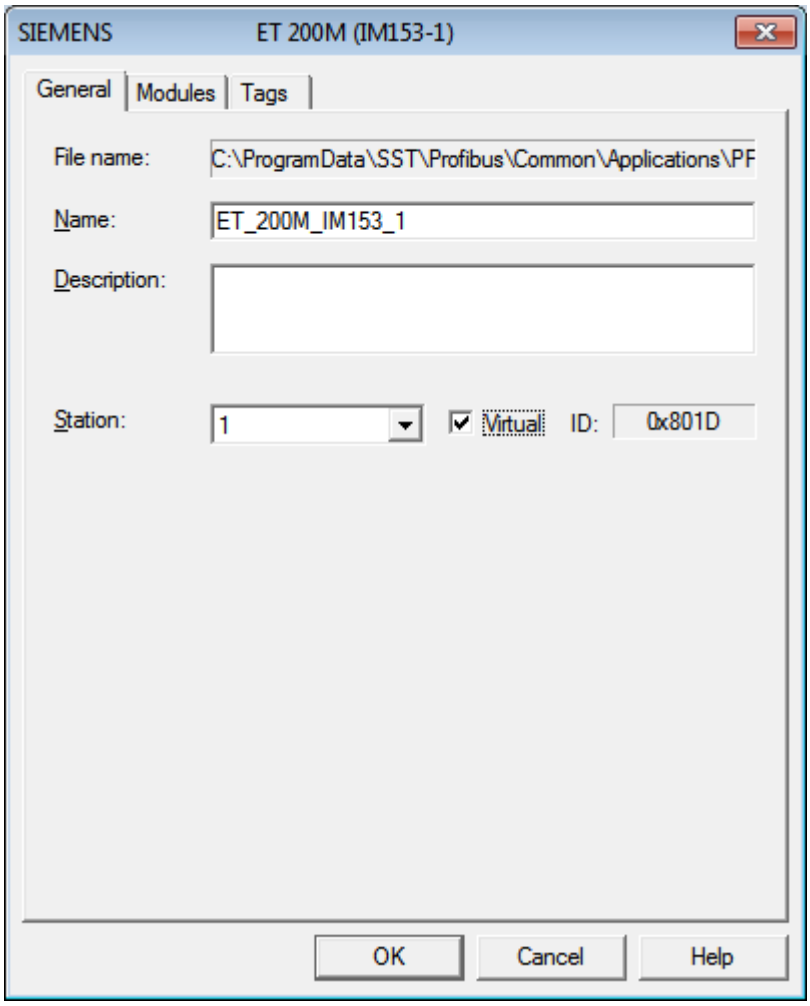


1. Click on this icon  to add GSD files to the device library and select all the GSD files you want to apply and click on Open. The following screen appears:

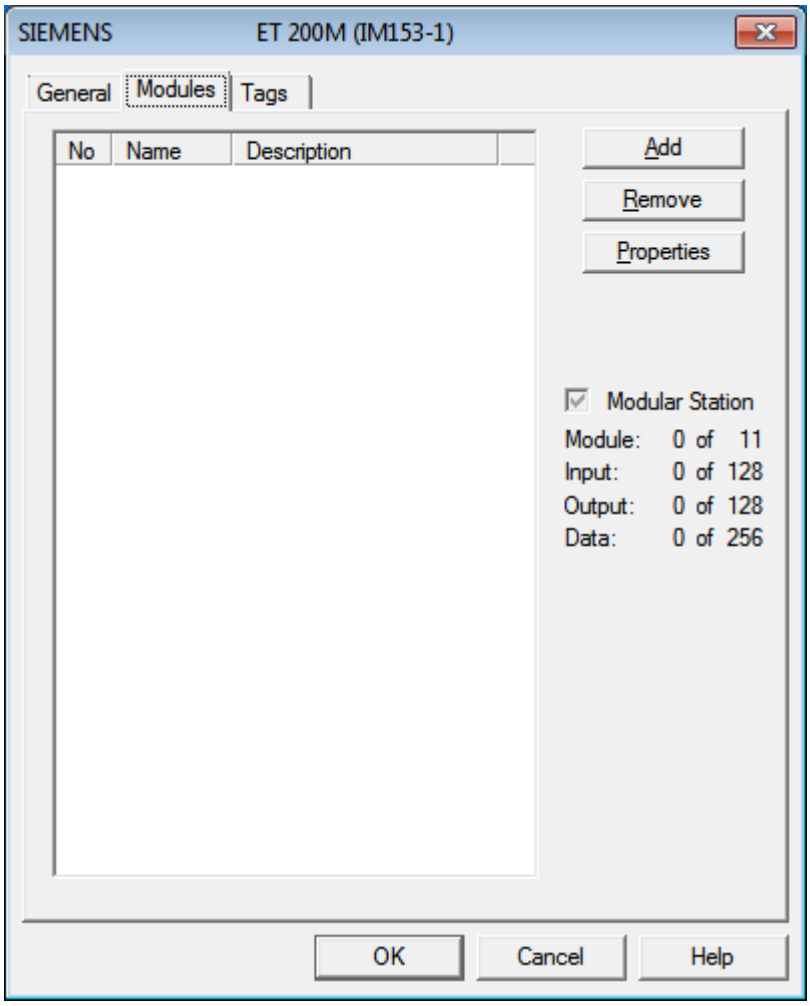


2. Your devices will now appear in the Device Library, enabling you to configure these devices as Monitor or Virtual. To configure a Virtual or Monitor slave, do the following:

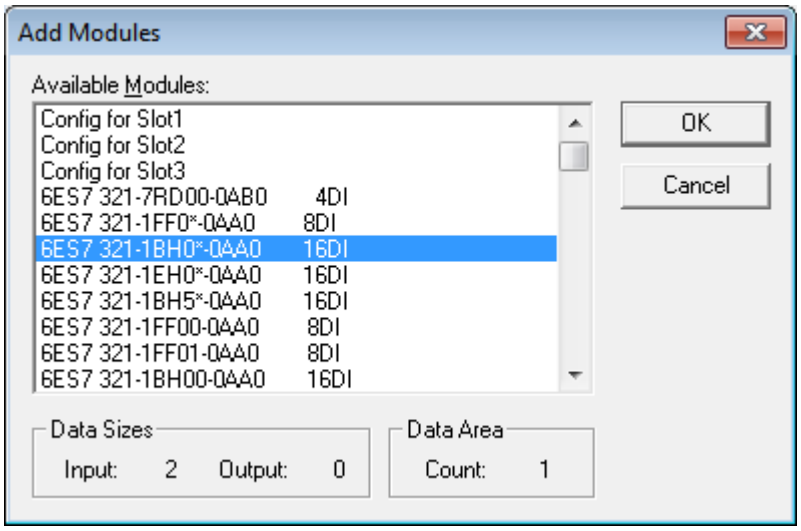
   - Select the Slave, in this case it's the ET 200M (IM153-1)
   - Drag this slave over to the network configuration tree. The slave properties box will appear.
   - Configure the station address and check virtual option for a virtual slave and leave it unchecked for a monitored slave.
   - Select Modules tab.

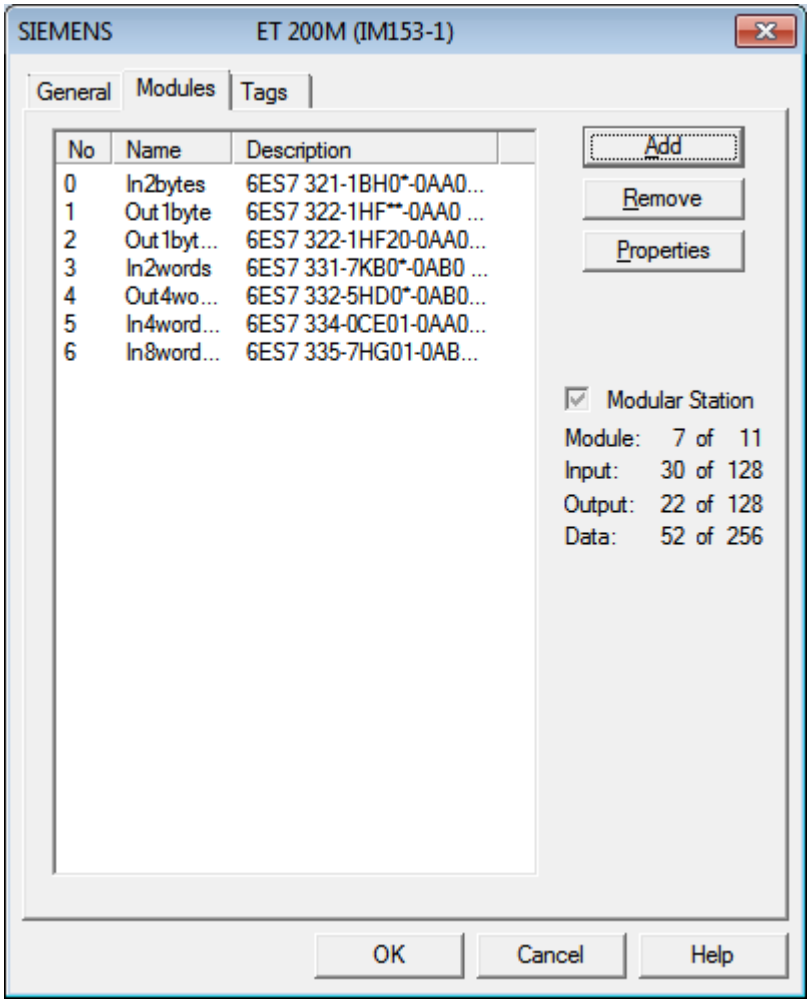


3. Configure the modules for the slave. Click on the Add button.

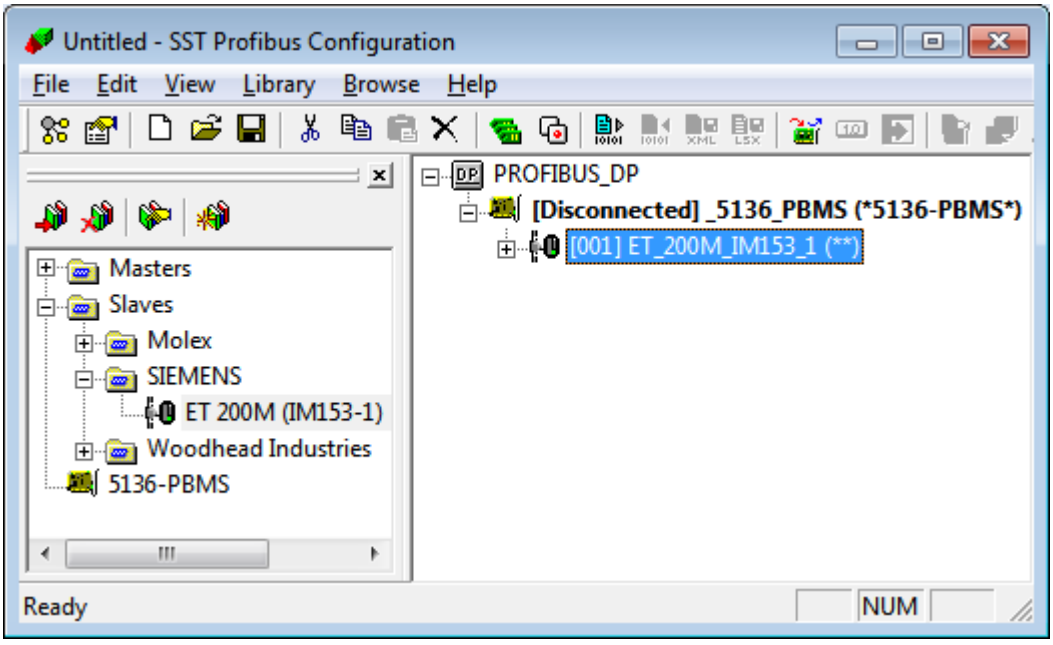


4. Select the module you want to add and click OK. Continue adding the modules as necessary.



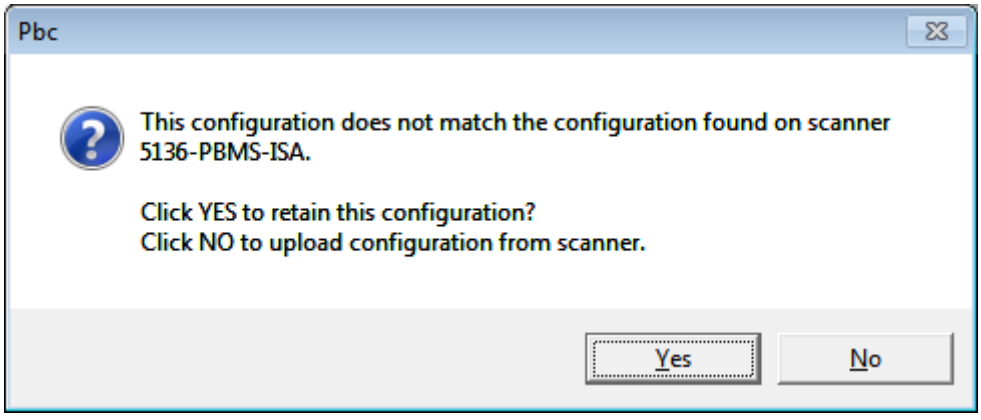


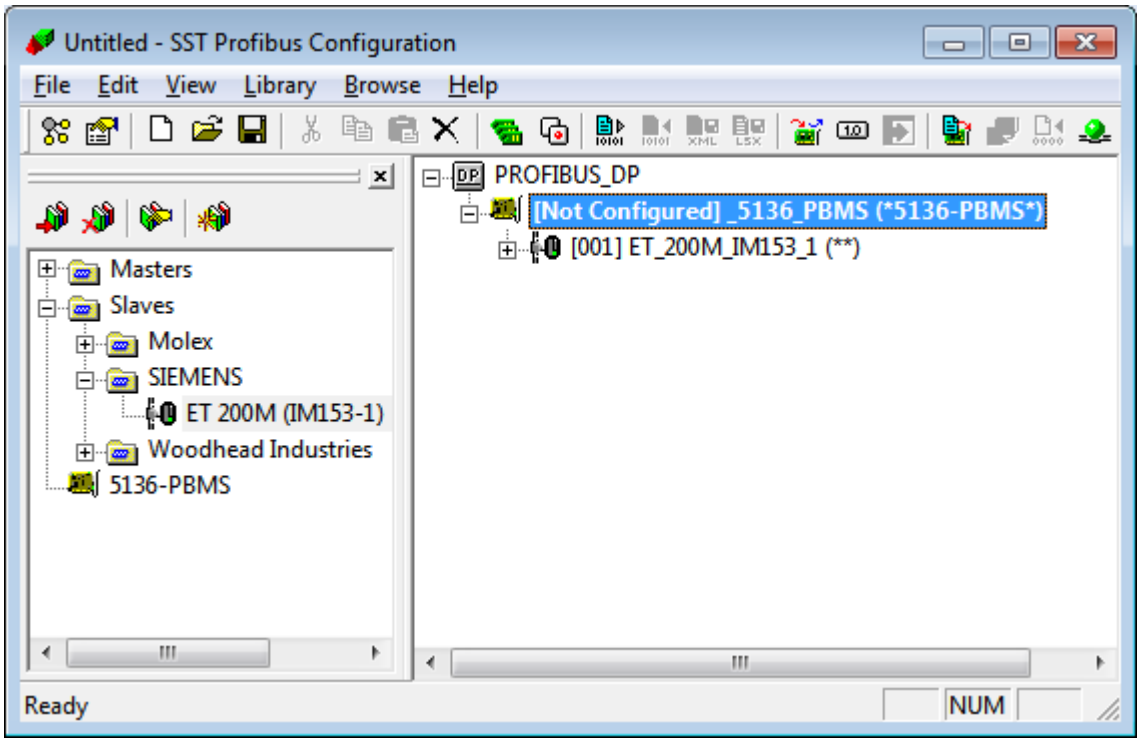
5. Click Ok.



6. Click on Connect/Disconnect icon or select Connect from the drop-down menu when right clicking on [Disconnected] _5136_PBMS (*5136-PBMS*) in the network configuration tree to connect to the card.

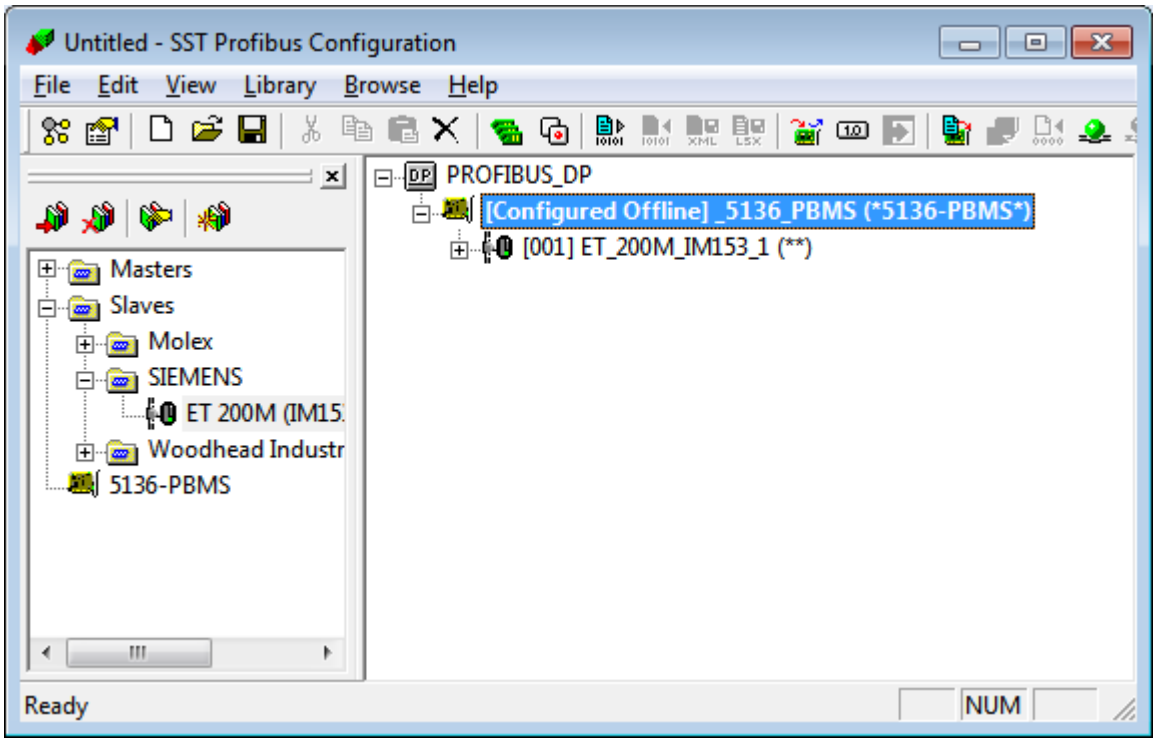
   You might experience a message indicating a configuration mismatch between the previous configuration loaded on card and the current one that you are working on. Click on Yes to retain your current configuration.



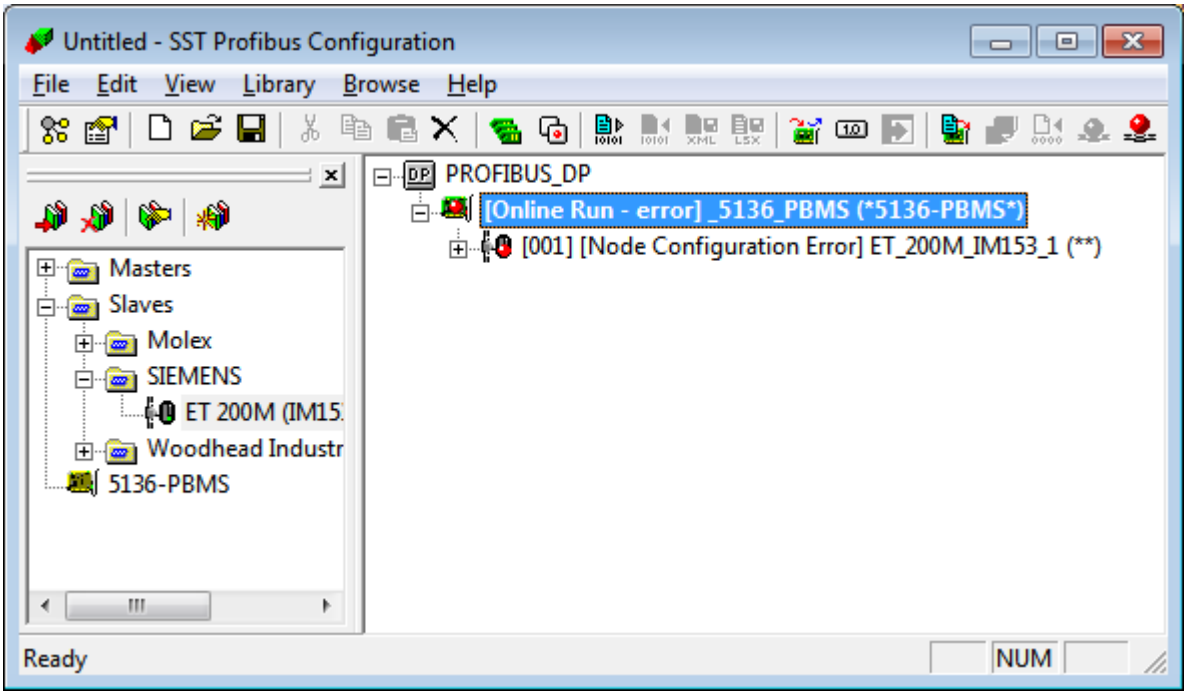


7. Click on *the Load configuration*  icon or select *Load Configuration* from the drop-down menu by right clicking on 

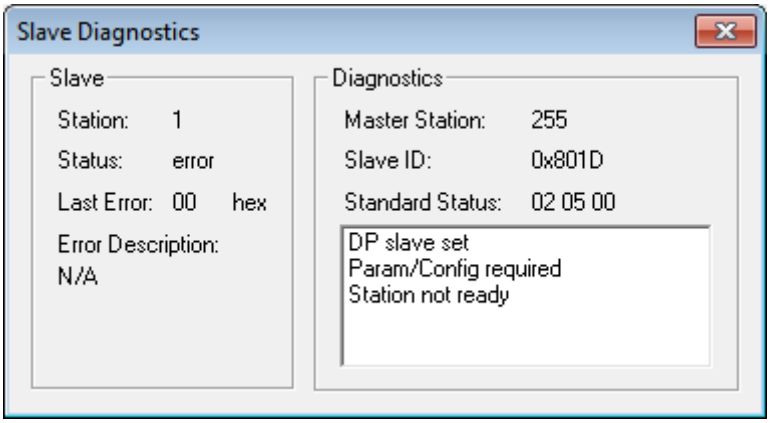
in the network configuration tree to load the current configuration to the card.



8. Click on the *Online* icon or select *Online* from the drop-down menu by right clicking on [Configured Offline] _5136_PBMS (*5136-PBMS*) in the network configuration tree to put the card online.



9. You will see the status of the slave appear as a red light to indicate there are errors present with this device. This status will go green when slave is communicating successfully. When you receive an error for a slave you can right click on slave and select Diagnostics from the drop-down menu. The following Slave Diagnostics window appears. This status information will help guide you to trouble shooting the slave device.



4.2 Diagnosing Slave Errors in the SST-PROFIBUS Configuration Tool

4.2.1 Introduction

This section provides detailed information about the slave status messages displayed in the slave diagnostic window. Refer to section 4.1 for details on how to display slave diagnostic information.

4.2.2 Station Non Existent

Check the Profibus slave interface to ensure it is powered on and connected to the network. The slave might not support the configured BAUD rate. This may occur when using an incorrect GSD file for the slave.

4.2.3 Configuration Data Fault

Check that I/O modules are configured in the order they appear in the slave device and that the I/O modules match the exact description of the modules you selected (i.e. module serial number).

4.2.4 Station Not Ready

If there are still outstanding errors, this is always present. Also, the master might be trying to scan this slave too fast. This could be caused by using the wrong GSD file for this slave.

4.2.5 Extended Diagnostic Data

This extended diagnostic data is slave specific. Compare values with descriptions that may be given for your slave by viewing properties of the slave and selecting the "Diagnostics" tab in the SST Profibus configuration tool.

4.2.6 Function Not Supported

Slave does not support a command coming from the master (i.e. SYNC or FREEZE). Check that the proper GSD file is being used.

4.2.7 Invalid Slave Response

Usually occurs when the slave is returning more diagnostic information than the master can handle. Check the GSD file.

4.2.8 Parameter Fault

Usually means an incorrect Ident-Number or Invalid parameter sent to the slave. Check that the parameters are set correctly. View slave properties and select the "Ext. Prms" tab. Also, view the module parameters by selecting the "Modules" tab under slave properties and view the module's properties and select the "Ext. Prms" tab.

4.2.9 Master Lock

The DP slave has been parameterized by another master. Take the controlling master offline.

4.2.10 Param/Config Required

This remains present until the parameterization has been completed properly. An incorrect GSD file may have been used, or, the slave and module parameters may not have been set to the correct values. These can be found under the slave properties, module tab, view properties of module, select Ext. Prms tab, and by selecting Ext. Prms tab under slave properties.

4.2.11 Ext Diags Overflow

More diagnostic data is being returned from the slave than what is specified in the GSD file. Check that the correct GSD file is being used.

5 Using the PBMDPSLV Module

This section is a guide for writing applications/drivers to interface to SST PBMDPSLV module (pbmdpslv.ssf) for the SST-PBMS-PCI interface card. This information is also required for firmware developers using this card in an embedded application. Detailed knowledge of this section is not required if the user is using the SST Profibus Configuration tool to configure and monitor the SST-PBMS-PCI card.

Using the PBMDPSLV software module, the SST-PBMS-PCI can:

- emulate up to 125 DP slaves
- monitor up to 125 DP slaves

For each configured slave, you can display:

- input and output data for the slave
- slave diagnostic data
- slave parameterization data
- slave configuration check data

For each emulated (virtual) slave, you can:

- change the input data
- change and update the diagnostic data

In addition, the PBMDPSLV module:

- supports all standard baud rates
- maintains a table showing the status of all the configured slaves
- maintains diagnostic counters

This section starts with some general information, then describes the module in detail, with information about the interface between your application and the module and the organization of the tables on the card.

5.1 PBMDPSLV Software Overview

5.1.1 Memory Overview

The host uses a memory mapped interface to the program running on the card. The user interface for the PBMDPSLV module is completely open. You don't need to link to any special libraries, making the card independent of the compiler and operating system being used.

The shared memory on the card is paged, with one 16 Kbyte page mapped into host memory at a time. Each page is divided into 2 Kbyte blocks, eight blocks per page. The lowest block on page 0 contains the host user interface structure. This structure is defined in the following table. Host applications use this structure to configure and control the overall operation of the card.

The remainder of page 0 memory and the other pages of card memory contain configuration information, I/O data and status for each of the DP slaves.

The following table briefly describes the layout of the host user interface structure.

Table 5.1

Offset	Size	Description
200	BYTE	Card Command register
201	BYTE	Card Status register
202	BYTE	Communication Status
203	BYTE	Network Baud Rate
204	WORD	Network options
20A	WORD	Card ID
20C	WORD	Module ID
20E	WORD	Module version
210-215		Event queue registers (not implemented)
216	BYTE	Trigger queue head pointer
217	BYTE	Trgger queue tail pointer
2C0-2D3		Diagnostic counters
300-37F	BYTE	Station monitor list
380-3FF	BYTE	Station transmit length table
400-47F	BYTE	Station status table
480-4FF	WORD	trigger queue
500-5FF	WORD	Event queue (not used)
600-6FF	WORD	Initial slave watchdog times

The remaining memory is used for slave configuration information and I/O data. The data block for each slave occupies 2 Kbytes (2048 bytes). The following table shows the memory areas used for each slave number.

Table 5.2

	Offset							
Page	0000	800	1000	1800	2000	2800	3000	3800
0		1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39
5	40	41	42	43	44	45	46	47
6	48	49	50	51	52	53	54	55
7	56	57	58	59	60	61	62	63
8	64	65	66	67	68	69	70	71
9	72	73	74	75	76	77	78	79
10	80	81	82	83	84	85	86	87
11	88	89	90	91	92	93	94	95
12	96	97	98	99	100	101	102	103
13	104	105	106	107	108	109	110	111
14	112	113	114	115	116	117	118	119
15	120	121	122	123	124	125		

Each slave block contains the following elements:

Table 5.3

Offset	Name	Description
0	outlen	Length of Output data (from master) + 3
4-f7	OutData	Output data
100	inpLen	Length of input data (to master) + 3
103	inpFC	Input frame control
104-1f7	inpData	Input data
300	dgnLen	Length of diagnostic status response + 5
306-3f9		Diagnostic status response data
400	prnLen	Length of parameterization data + 5
406-4f9		Parameterization data
600	cfgLen	Length of Configuration check data + 5
606-6f9	cfgChkData	Configuration check data

5.2 Configuring and Bringing the Card Online

The following is a description of the host system steps involved in bringing the card online:

Note: This section assumes that the card is already installed in the host system and the firmware module (PBMDPSLV) has been stored in the flash (section 2).

1. Confirm the card is present (section 5.2.1)
2. Configure the network (section 5.2.2)
3. Configure the slaves (section 5.2.3)
4. Put the card online (section 5.2.4)

5.2.1 Card and Module Identification

When a host application starts, it can use the following registers to confirm that the card is present, that the correct module has been loaded on the card, and to identify the module version.

Card ID

Applications can use the card ID, pbmCardId, to verify that the card is present. It contains a value of 0xaad1 or 43729 decimal.

Module ID

Applications can use the DP module ID, pbmModId , to verify that the correct module is loaded on the card. For the PBMDPSLV module, this register contains 0xbb02 or 47874 decimal.

Module Version

The version number of the PBMDPSLV module is stored in pbmModVer, with the major version number in the low byte and the minor version number in the high byte. For example, if the value you read is 0x0102, the version number is 1.02.

5.2.2 Configuring the Network

Before you put the card online, you must set the network baud rate and set any other required network options.

Baud Rate

The host uses the Network Baud Rate register in the host user interface structure (section 5.1.1) to set the network baud rate, according to the values in the following table:

Baud rate	PBMCTL.H	Value
9600	BAUD_9k6	0
19200	BAUD_19k2	1
93.75K	BAUD_93k75	2
187.5K	BAUD_187k5	3
500K	BAUD_500k	4
1.5M	BAUD_1m5	6
3M	BAUD_3m	7
6M	BAUD_6m	8
12M	BAUD_12m	9
31.25K	BAUD_31k25	10
45.45K	BAUD_45k45	11

The default baud rate is 9600 baud.

Network Options

The host sets options related to how the card operates on the network by setting bits in the Network Options register in the host user interface structure (section 5.1.1).

At present there are no network options.

5.2.3 Configuring the Slaves

Before you put the card online, you must configure the slaves you are emulating or monitoring. Emulated slaves are referred to as virtual slaves.

To configure a slave, you enter data into the tables and structures on the card. For a given slave, some of the data is written to tables in the host interface area (table 5.1) and some of it is written to the control block for the slave (table 5.3).

For each possible slave, there is a slave control block. Each slave control block occupies 2048 bytes. The contents of the slave control block are given in table 5.3. Use table 5.2 in section 5.1.1 to locate the page number and offset for a particular slave.

The host application must perform the following steps to configure a slave:

1. Enable the station by setting bit 0 in the element of the Station monitor list (table 5.1) table that corresponds to the station number. For example, station 2 corresponds to Station monitor list base address +2, or 0x302.
2. If the station is to be emulated by the card set bit 1 in the element of the Station monitor list (table 5.1) table that corresponds to the slave.
3. If you want this station to be ignored in the Communication Status register (table 5.1) and on the SYS LED, set bit 15 in the element of the Station monitor list (table 5.1) table that corresponds to the slave.

4. If the slave is virtual, set the slave Ident in the slave control block. The value you enter must match the Ident configured in the master. Write the high byte of the Ident to dgnIdentHi and the low byte to dgnIdentLo. DgnIdentHi and dgnIdentLo are located in the Diagnostic status response data area of the slave control block Refer to table 5.3 and Appendix A for more details. For real slaves, the Ident will be 0 until the card catches a configuration packet from the network and enters the Ident for the slave.
5. If the slave is virtual, set the length of the transmit data (to the master) in the element of the Station transmit length table (located in host interface structure table 5.1) that corresponds to the slave. This table contains one byte per station. The third byte offset into the table (pbmTxLen[3]) corresponding to station 3, etc. The range of transmit length values is 0 to 244. The default is 0. Refer to to pbmTxLen in Appendix A for more details
6. Set up any extended diagnostic data in the slave block. Write the diagnostic data to the dgnExtData table (located in the Diagnostic status response data area of the slave control bock) in the slave block for the slave and the length of the extended diagnostic data + 1 to dgnExtDataLen (located in the Diagnostic status response data area of the slave control bock). Refer to section 5.3.7 for information about slave diagnostics and extended diagnostics. Refer to Appendix A for the definition of dgnExtData and dgnExtDataLen.
7. For real slaves, set the initial slave watchdog time in the pbmWdTime table (located in the Initial slave watchdog times area of the host user interface structure). Allowed values are 1 to 65535 which correspond to increments of 10 ms. The default is 1000 or 10 seconds. The value will be replaced by the value from the master if the card sees a parameterization packet from the master to that slave. For virtual slaves, the card gets the watchdog time from the master and you don't need to enter a value. Refer to Appendix A for the definition of pbmWdTime.

The sample program PBMCNF.C shows how to configure slaves. It can be used with the sample data file manyslvs.cnf.

5.2.4 Putting the Card Online

The host uses the command register (located in the host user interface structure), to put the card online or take it offline.

The host issues the following commands to the PBMDPSLV module by writing the value shown to the command register.

Command	Value	Effect
CMD_GO_ON	0x01	card should now go online
CMD_GO_OFF	0x02	card should now go offline, or abort config
CMD_ERR_ACK	0xff	acknowledge offline error (online errors cannot be acked)

If the card is online and you issue any command other than the command to go offline, the card sets the command register to CMD_ERROR, sets the status register to STS_BAD_CMD, and goes offline. You must issue the CMD_ERR_ACK command.

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

State name	Value	Description
CMD_OFF	0xe0	card is offline ready to take commands
CMD_ON	0xe1	card is online ready to take offline command
CMD_ERROR	0xef	card is in error, status contains error code

When the card successfully executes the command to go offline, it sets the command register to CMD_OFF. When the card successfully executes the command to go online, it sets the command register to CMD_ON. If the command fails, the card sets the command register to CMD_ERROR.

When you issue an online or offline command, wait up to 2 seconds for the card to execute the command.

If there is an error in executing a command, the host should check the status register (see section 5.3) to determine what the error was. The host should clear the error by issuing the following command.

CMD_ERR_ACK	0xff	acknowledge offline error (online errors cannot be acknowledged)

This returns the card to the startup state, just after you ran the card processor. You must then reconfigure the slaves and put the card back online.

5.3 Accessing the I/O and Diagnostic Data

This section describes how to access the network I/O and the various diagnostic information.

5.3.1 Status Register

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

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

If the card encounters a problem executing a command, it writes the value CMD_ERROR (0xef) in the command register and returns a value in the status register that indicates the nature of the error.

If the host issues an invalid command, the card sets the status register to STS_BAD_CMD (0x01).

5.3.1.1 Fatal Errors

If the status register contains STS_CFG_INTERNAL_ERROR (0x80), there has been an internal error on the card. The card software must be rerun.

5.3.2 What happens when the master brings a slave online...

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

1. The master sends a diagnostic read (SRD).
2. The slave responds with its diagnostic data (DL, data low).
3. If everything in the diagnostic data is OK, the master then sends parameterization data to the slave (SRD).
4. The slave responds with a DL or ACK.
5. The master sends a configuration check data packet, see Profibus specification.
6. The slave replies with a DL or ACK.
7. The master sends a diagnostic read (SRD).
8. The slave replies with its diagnostic data (DL). If there are any problems with anything up to this point, the slave says so now in its reply. Otherwise, cyclic data transfer begins between the master and the slave and the slave is considered to be online.

5.3.3 Accessing I/O Data

The I/O data for each slave is found in the slave control block for that slave.

5.3.4 Locating the Slave Block

To select the correct page for the slave block:

1. Set the lower two bits in the CCR (see section 2.9.3) to 0 to select the memory page register. Usually you leave these two bits as 0 after you run the software module so that the memory page register is always selected.
2. The page number is:

   0 for the host interface

   slave# / 8 for any slave (integer divide)
3. Within the page, if the blocks are numbered from 0 to 7, the block that correspond to the slave is the slave number mod 8. For example, if the slave number is 22, 22 mod 8 = 6, and the data for slave 22 starts at 2048 * 6 = 12288 or 0x3000.

The table in section 5.1.1 shows the page and offset for all possible slaves.

5.3.5 Output Data (from the Master)

Read the value in outLen (table 5.3) and subtract 3 to determine the length of the output data. For example, if outLen contains 247, the output data length is 244.

The output data is located in outData (table 5.3).

5.3.6 Input Data (to the Master)

Read the value in inpLen (table 5.3) and subtract 3 to determine the length of the input data. For example, if inpLen contains 27, the input data length is 24.

The input data is in inpData (table 5.3).

5.3.7 Slave Diagnostic Data

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

The diagnostic data consists of:

- three (3) station status bytes
- the station ID of the master that parameterized the slave
- the slave Ident number
- the length of the extended diagnostics
- up to 25 bytes of vendor defined extended diagnostic data

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

The card stores the station status bytes in dgnSts1, dgnSts2 and dgnSts3 (located in the Diagnostic status response data area of the slave control bock) . The card software maintains the station status bytes. Refer to the Profibus specification for information on the meaning of bits within these status bytes (from which the following was obtained).

```
Octet 1: Station_status_1
         MSB                  LSB
         +--+--+--+--+--+--+--+--+
 Bit-No. !7 !6 !5 !4 !3 !2 !1 !0 !
         +--+--+--+--+--+--+--+--+
```

The individual bits have the following meaning:

Bit 7: Diag.Master_Lock

The DP-Slave has been parameterized from another master. This bit is set by the DP-Master (class 1), if the address in octet 4 is different from FFH and different from the own address. The DP-Slave sets this bit to zero.

Bit 6: Diag.Prm_Fault

This bit is set by the DP-Slave if the last parameter frame was faulty, e. g. wrong length, wrong Ident_Number, invalid parameters.

Bit 5: Diag.Invalid_Slave_Response

This bit is set by the DP-Master as soon as receiving a not plausible response from an addressed DP-Slave. The DP-Slave sets this bit to zero.

Bit 4: Diag.Not_Supported

This bit is set by the DP-Slave as soon as a function was requested which is not supported from this DP-Slave.

Bit 3: Diag.Ext_Diag

This bit is set by the DP-Slave. It indicates that a diagnostic entry exists in the slave specific diagnostic area (Ext_Diag_Data).

Bit 2: Diag.Cfg_Fault

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

Bit 1: Diag.Station_Not_Ready

This bit is set by the DP-Slave if the DP-Slave is not yet ready for data transfer.

Bit 0: Diag.Station_Non_Existent

This bit is set by the DP-Master if the respective DP-Slave can not be reached over the line. If this bit is set the diagnostic bits contain the state of the last diagnostic message or the initial value. The DP-Slave sets this bit to zero.

Octet 2: Station_status_2

```
         MSB                  LSB

         +--+--+--+--+--+--+--+--+

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

         +--+--+--+--+--+--+--+--+
```

The individual bits have the following meaning:

Bit 7: Diag.Deactivated

This bit is set by the DP-Master as soon as the DP-Slave has been marked inactive within the DP-Slave parameter set and has been removed from cyclic processing. The DP-Slave sets this bit always to zero.

Bit 6: reserved

Bit 5: Diag.Sync_Mode

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

Bit 4: Diag.Freeze_Mode

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

Bit 3: Diag.WD_On (Watchdog on)

This bit is set by the DP-Slave. If this bit is set to 1 the watchdog control at the DP-Slave has been activated.

Bit 2: This bit is set to 1 by the DP-Slave.

Bit 1: Diag.Stat_Diag (static diagnostics)

If the DP-Slave sets this bit the DP-Master shall fetch diagnostic data as long as this bit is reset again. For example, the DP-Slave sets this bit if it is not able to provide valid user data. This bit is set by the DP-Slave.

Bit 0: Diag.Prm_Req

If the DP-Slave sets this bit the respective DP-Slave shall be reparameterized and reconfigured. The bit remains set until parameterization is finished. This bit is set by the DP-Slave.

If bit 1 and bit 0 are set, bit 0 has the higher priority.

```
Octet 3: Station_status_3
         MSB                  LSB
         +--+--+--+--+--+--+--+--+
 Bit-No. !7 !6 !5 !4 !3 !2 !1 !0 !
         +--+--+--+--+--+--+--+--+
```

```
The individual bits have the following meaning:

Bit 7: Diag.Ext_Diag_Overflow

If this bit is set there exists more diagnostic information than specified in Ext_Diag_Data. For
example, the DP-Slave sets this bit if there are more channel diagnostics than the DP-Slave can
enter in its send buffer; or the DP-Master sets this bit if the DP-Slave sends more diagnostic
information than the DP-Master can enter in its diagnostic buffer.

Bit 0-6: reserved
```

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

The slave Ident number returned by the slave is stored in dgnIdentHi (high byte) and dgnIdentLo (low byte). For virtual slave, you must fill these two bytes in before you put the card online. For real slaves, the card fills in the Ident values when it sees a configuration message on the network.

The length of the extended diagnostics is stored in dgnExtDataLen. If there are no extended diagnostics, the length is 0. If there are extended diagnostics, the length includes the length byte itself. For example, if there are 5 bytes of extended diagnostics, the dgnExtDataLength is 6.

The extended diagnostic data is stored in dgnExtData.

5.3.8 Updating Diagnostics Online

If you want to update diagnostics for a virtual slave while the card is online, you post a trigger in the trigger queue of the host user interface structure. (See “pbmTrgQueue” in Appendix A)

The host accesses the trigger queue by means of the trigger queue head pointer, pbmTrgHead. To insert a request, the host puts the appropriate request value in the trigger queue at the head position (stored in the “Trigger queue head pointer” in the host user interface structure), then increments the head pointer.

The value the host writes is TRG_REQ_DIAG_UPDATE (0x1000) OR’d with the slave station number. That is, the high byte is 0x10 and the low byte is the slave station number.

The trigger queue contains 64 entries. When the host increments the head pointer, it must ensure that the head pointer wraps around to 0. It can do this by ANDing the head pointer with 0x3f before it writes the value.

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

The slave requests that the master update its diagnostics by sending a high priority data update (DH) instead of the usual low priority update (DL). You can use inpFC to determine if a diagnostic update is pending. If the value is FC_DH (0x0a) then a diagnostic update is pending. Normally the value is FC_DL (0x08).

5.3.9 Slave Parameterization Data (from master)

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

- master status byte
- slave watchdog factors (2 bytes)
- slave response delay time
- slave ident (2 bytes)
- group ID

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

5.3.9.1 Master Status Byte

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

5.3.9.2 Slave Watchdog Factors

The master sends two parameters, prmWdogFact1 and prmWdogFact2 to set the master watchdog time for the slave. The watchdog time in milliseconds is calculated as:

watchdogtime = 10 * prmWdogFact1 * prmWdogFact2

where the two factors can range from 1 to 255.

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

For virtual slaves, the card uses these watchdog factors to calculate the watchdog time and enters the value in the pbmWdTime table element for this slave.

5.3.9.3 Slave Response Delay

The master sends the minimum station response delay in prmRdyTme. This is the minimum time, in Tbits, that the slave waits before it sends a reply to the master.

A Tbit is the inverse of the baud rate.

If the value sent by the master in the parameter data is 0, the slave uses the value configured in its network parameters (which the card sets to a default value).

5.3.9.4 Slave Ident

Each Profibus DP slave module has a unique Ident assigned by the Profibus Trade Organization. The master sends the values configured in the master for this slave using prmIdentHi and prmIdentLo. These are the high byte and the low byte, respectively, of the slave Ident number. If the value configured in the master does not match the Ident number of the slave, the slave will not communicate with the master.

5.3.9.5 Group ID

The group ID, prmGrpId, is used in Profibus for functions such as sync and freeze. These functions are not supported by the card software and this byte should always be 0 for virtual slaves.

5.3.9.6 Extended Parameter Data

The master can send up to 25 bytes of extended parameter (up to 237 bytes with extensions) data to a slave. What this parameter data is used for depends on what kind of device the slave is.

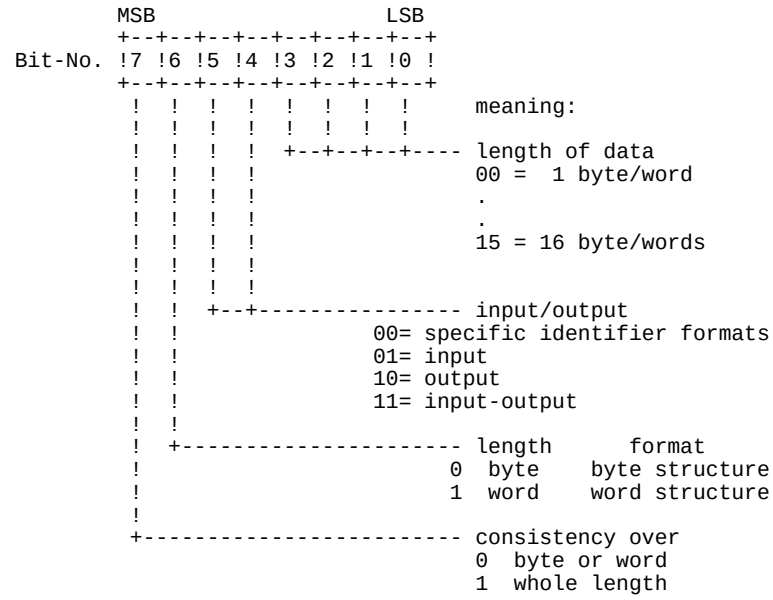
The card stores the total length of the parameterization data + 5 in prmLen. For example, if the master sends just the first 7 bytes, prmLen contains 12. If the master sends 22 bytes of extended parameter data in addition to the first 7 bytes, prmLen contains 34.

The extended parameterization data is stored in prmExtData.

5.3.10 Configuration Check Data

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

The configuration check data consists of a number of identifier bytes. The identifier byte has the following format (from the Profibus specification):



For more information about specific identifier formats or other details, refer to the Profibus specification.

The card writes the length of the configuration check data for the slave + 5 in cfgLen and the configuration check data in cfgChkData. For example, if cfgLen contains 9, there are 4 bytes of configuration check data in cfgChkData.

5.3.10.1 Configuration Check Example

In simple cases:

- the contents of the lower 4 bits + 1 equals the length of the data in bytes or words, i.e., if the length is 7, the lower 4 bits should be set to 6 or 0110
- bits 4 and 5 indicate the data type (01=inputs, 10=outputs, 11=input/output)
- bit 6 indicates whether the length is in bytes or words (0=bytes, 1=words)

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

For an input module with a length of 4 bytes, the configuration check byte would therefore be 0010 0011 = 0x23 = 35 decimal.

5.3.11 Monitoring Status

Communication Status

The card indicates overall status with pbmCommStatus. It sets bit 1, COMM_STS_ALL_VIRT_GOOD, if all virtual slaves are being updated. It sets bit 2, COMM_STS_ALL_GOOD, if all slaves are being updated.

If bit 15, COMM_STS_RUN, is set, the network is being updated in run mode. This bit is valid only if all slaves have good status, and only if there is only one master on the network.

In run mode the master updates the outputs and the slaves send inputs. In stop mode, the master sends all outputs as 0.

Slave Status

To check the status of the slaves, use the station status table, pbmStnSts. The card sets bit 7, STN_STS_OK, if the slave has been updated within the watchdog period.

5.3.12 Diagnostic Counters

The card maintains a variety of diagnostic counters to indicate general statistics on messages sent and received, etc.

To clear the counters, set pbmInitCtrs to 1. The card then clears the counters to 0 and sets pbmInitCtrs to 0 to indicate to the host that the counters have been cleared.

The card increments pbmRxFrames every time it receives a good packet.

The card increments pbmCfgFrames every time it receives a configuration packet from the master. This includes diagnostic status, parameterization, and configuration check packets. The counter also increments when you update diagnostics online.

The card increments pbmBadRxFrames whenever there's a parity error, framing error, wrong delimiters, bad length, or bad checksum.

The card increments pbmSlvTmeOuts any time it times out a slave (virtual or monitored).

The card increments pbmFrameErrors whenever it receives a packet with wrong delimiters or incorrect length.

The card increments pbmChkErrors whenever it receives a packet with a checksum error.

The card increments pbmParErrors whenever it receives a packet with a parity error.

The card increments pbmDupSlvErrors whenever it detects a duplicate slave on the network.

The card uses pbmErrArg as an argument for internal diagnostic.

6 Appendix

6.1 Appendix A

```
*-------------------------------------------------------
   SST-PBMS-PCI PBMDPSLV Shared Memory Interface Definitions

   Copyright (c) 1997-2012 Molex Incorporated

---------------------------------------------------
 Revision History
---------------------------------------------------

 REV       DATE                        Description
---------------------------------------------------
1.1       04/26/2004
-------------------------------------------------------*/

#ifndef _H__PBMCTL
#define _H__PBMCTL

#ifndef _H__PROFICTL
#ifndef _186_CODE
#define CCM  0  // Card Command Register
#define CCR  2  // Card Configuration Register
#define MCR  3  // Memory Configuration Register
#define ICR  4  // Interrupt Configuration Register
#define IDR  7  // Card ID Register

/*
#define CCM  port+0  // Card Command Register
#define CCR  port+2  // Card Configuration Register
#define MCR  port+3  // Memory Configuration Register
#define ICR  port+4  // Interrupt Configuration Register
#define IDR  port+7  // Card ID Register
*/

// CCM Equates
#define INT_PEND       0x02    // an interrupt is pending from the PBM card
#define MEM_ENA        0x08    // enable  card memory in host address space
#define MEM_DIS        0x04    // disable card memory in host address space

// CCR Equates
#define FLASH_BOOT     0x00
#define SH_BOOT        0x10    // boot from shared ram
#define uPROC_RESET    0x00    // clr to reset 188
#define uPROC_RUN      0x80    // set to release 188 reset
```

```
#define MPR            0x00    // Memory Page register select value
#define MCMP           0x03    // Memory address compare register select value

//MPR equates
#define DBL_BNK        0x80    // select double buffering second bank

// IDR Equates
#define ID_5136_PB_MSLV 0x02


#define PBM_MEM_PAGE_SIZE   16384   /* size of 5136-PBM shared memory page            */
#define PBM_MEM_NUM_PAGES   16      /* number of 5136-PBM shared memory pages (256k)  */

#endif // _186_CODE
#endif // _H__PROFICTL

/* data type definitions */
typedef unsigned char   uchr;   /*  8    bits */
typedef unsigned short  ushrt;  /* 16    bits */
typedef unsigned int    uint;   /* 16/32 bits */
typedef unsigned long   ulng;   /* 32    bits */


#ifndef _186_CODE
/* structure of 5136-PBMSLV user interface (page 0 offset 0) */
typedef struct tagPBM_USR
{
        uchr   Res1[0x200-0x000];  /* 000-1ff Reserved, DO NOT USE                    */

   volatile uchr   pbmCommand;      /* 200 Card command register                     */
   volatile uchr   pbmStatus;       /* 201 Card status register                      */
   volatile uchr   pbmCommSts;      /* 202 Communication status                      */

/* Profibus Basic Parameters */
   volatile uchr   pbmBaud;         /* 203 network baud rate                         */
   volatile ushrt  pbmOptions;      /* 204 network options (not currently used)      */

        uchr   Res2[4];          /* 206-209 Reserved, DO NOT USE                    */

/* 5136-PBM Card and Module Identification registers */
   volatile ushrt  pbmCardId;       /* 20a 5136-PBMSLV Card ID (0xaad1)              */
   volatile ushrt  pbmModId;        /* 20c PBMDPSLV Module ID     (0xbb02)           */
   volatile ushrt  pbmModVer;       /* 20e PBMDPSLV Module Version (ex. 0x0102 = 1.02) */

/* 5136-PBM --> Host Event and Interrupt control registers */
   volatile ushrt  pbmEvtEna;       /* 210 Event enable mask                         */
   volatile ushrt  pbmIntEna;       /* 212 Event interrupt enable mask               */
   volatile uchr   pbmEvtHead;      /* 214 Event queue head pointer (changed by card) */
   volatile uchr   pbmEvtTail;      /* 215 Event queue tail pointer (changed by host) */

/* 5136-PBM --> Host Trigger Queue control */
   volatile uchr   pbmTrgHead;      /* 216 pbm function trigger event head pointer (changed by host)*/
   volatile uchr   pbmTrgTail;      /* 217 pbm function trigger event tail pointer (changed by card) */
```

```
        uchr   Res3[0x2c0-0x218];  /* 240-218 Reserved, DO NOT USE                    */

/* Diagnostic Counters and control */
   volatile uchr   pbmInitCtrs;       /* 2c0 if non-zero PBM will init counters then set back to 0       */

        uchr   Res4;             /* 2c1 Reserved, DO NOT USE                          */

   volatile ushrt  pbmRxFrames;       /* 2c2 Total received frames (to virtual or monitored stations)   */
   volatile ushrt  pbmCfgFrames;      /* 2c4 Total received configuration frames                        */

   volatile ushrt  pbmBadRxFrames;    /* 2c6 Frames received with errors                                */
   volatile ushrt  pbmSlvTmeOuts;     /* 2c8 Slave watchdog timeouts                                    */

   volatile ushrt  pbmFrameErrors;    /* 2ca Length, Start delimiter or end delimiter errors            */
   volatile ushrt  pbmChkErrors;      /* 2cc Checksum errors during receive                             */
   volatile ushrt  pbmParErrors;      /* 2ce Parity errors during receive                               */
   volatile ushrt  pbmDupSlvErrors;   /* 2d0 duplicate slave errors during receive                      */

   volatile ushrt  pbmErrArg;         /* 2d2 argument for internal errors                               */

        uchr   Res5[0x300-0x2d4];  /* 2d4-2ff Reserved, DO NOT USE                                   */

   volatile uchr   pbmStnEna[128];    /* 300-37f Station monitor list                                   */
   volatile uchr   pbmTxLen[128];     /* 380-3ff Station Tx Len Table                                   */

   volatile uchr   pbmStnSts[128];    /* 400-47f Station status table                                   */
   volatile ushrt  pbmTrgQueue[64];   /* 480-4ff trigger queue                                          */

   volatile ushrt  pbmEventQueue[128]; /* 500-5ff event queue                                           */

   volatile ushrt  pbmWdTime[128];    /* 600-6ff Initial slave watchdog times                           */
} PBM_USR;
#endif

/* definitions related to pbmCommand */
#define CMD_OFF          0xe0      /* PBM  - card offline ready to take commands                       */
#define CMD_ON           0xe1      /* PBM  - card is online ready to take offline command              */
#define CMD_ERROR        0xef      /* PBM  - card is in error, status contains error code              */

#define CMD_GO_ON        0x01      /* HOST - card should now go on line                                */
#define CMD_GO_OFF       0x02      /* HOST - card should now go off line, or abort config              */

#define CMD_ERR_ACK      0xff /*HOST-acknowledge offline error(online errors cannot be acked)
*/


/* definitions related to pbmStatus */
#define STS_NO_ERROR              0x00
#define STS_BAD_CMD               0x01
#define STS_BAD_BAUD              0x02
#define STS_BAD_TRIGGER           0x08
```

```
/* the following are fatal errors. the card must be re-run */
#define STS_CFG_INTERNAL_ERROR          0x80

/* definitions related to pbmCommSts */
#define COMM_STS_ALL_VIRT_GOOD          0x02    /* all virtual slaves are being updated OK    */
#define COMM_STS_ALL_GOOD               0x04    /* all slaves are being updated OK            */
#define COMM_STS_RUN                    0x80    /* the DP network in RUN mode                 */

/* definitions related to pbmCardId */
#define CRD_5136_PBMSLV     0xaad1 /*apps can use this to verify that 5136-PBM card is present */

/* definitions related to pbmModId */
#define MOD_DPSLV           0xbb02  /*apps use this to verify that PBMDPSLV module is present*/

/* definitions related to pbmBaud */
#define BAUD_9k6        0
#define BAUD_19k2       1
#define BAUD_93k75      2
#define BAUD_187k5      3
#define BAUD_500k       4
#define BAUD_750k       5
#define BAUD_1m5        6
#define BAUD_3m         7
#define BAUD_6m         8
#define BAUD_12m        9
#define BAUD_31k25          10
#define BAUD_45k45    11

/* definitions related to pbmStnEna */
#define STN_ENA_MON           0x01      /* monitor this station                                 */
#define STN_ENA_VIRT          0x02      /* emulate this slave                                   */
#define STN_ENA_ERR_EVNT      0x40      /* generate event when this station errors              */
#define STN_ENA_IGNORE_STS    0x80      /* ignore status of this slave (ALL_OK, and LED)        */

/* definitions related to pbmStnSts */
#define STN_STS_OK            0x80      /* this station is being updated correctly             */

/* definitions related to pbmTrigQueue */
#define TRG_REQ_DIAG_UPDATE     0x1000



#ifndef _186_CODE
/* structure of slave entries */
typedef struct tagPBM_SLV
{
   volatile uchr   outLen;        /* 000     Length (length of data plus 3 - dst, src and frame control)  */
         uchr   Res1[3];       /* 001-003 reserved, do not use                                 */
   volatile uchr   outData[244];   /* 004-0f7 Output Data from master                              */
         uchr   Res2[8];       /* 0f8-0ff reserved, do not use                                 */

   volatile uchr   inpLen;        /* 100     Length (length of data plus 3 - dst, src and frame control   */
         uchr   Res3[2];       /* 101-102 reserved, do not use                                 */
```

```
    volatile uchr   inpFC;          /* 103     Input frame control                                  */
    volatile uchr   inpData[244];   /* 104-1f7 Input data to master                                 */
          uchr   Res4[8];        /* 1f8-1ff reserved, do not use                                 */

          uchr   Res5[0x100];    /* 200-2ff reserved, do not use                                 */

    volatile uchr   dgnLen;         /* 300     Length of diagnostic status response + 5             */
          uchr   Res6[5];        /* 301-305 reserved, do not use                                 */
    volatile uchr   dgnSts1;        /* 306     See profibus spec                                    */
    volatile uchr   dgnSts2;        /* 307     See profibus spec                                    */
    volatile uchr   dgnSts3;        /* 308     See profibus spec                                    */
    volatile uchr   dgnMstrAddr;    /* 309     See profibus spec                                    */
    volatile uchr   dgnIdentHi;     /* 30a     See profibus spec                                    */
    volatile uchr   dgnIdentLo;     /* 30b     See profibus spec                                    */
    volatile uchr   dgnExtDataLen;  /* 30c     Length of extended diagnostic data                   */
    volatile uchr   dgnExtData[237];/* 30d-3f9 Extended diagnostic data                             */
          uchr   Res7[6];        /* 3fa-3ff reserved, do not use                                 */

    volatile uchr   prmLen;         /* 400     Length of parameter information from master + 5      */
          uchr   Res8[5];        /* 401-405 reserved, do not use                                 */
    volatile uchr   prmSts;         /* 406     See profibus spec                                    */
    volatile uchr   prmWDogFact1;   /* 407     See profibus spec                                    */
    volatile uchr   prmWDogFact2;   /* 408     See profibus spec                                    */
    volatile uchr   prmRdyTme;      /* 409     See profibus spec                                    */
    volatile uchr   prmIdentHi;     /* 40a     See profibus spec                                    */
    volatile uchr   prmIdentLo;     /* 40b     See profibus spec                                    */
    volatile uchr   prmGrpID;       /* 40c     See profibus spec                                    */
    volatile uchr   prmExtData[237];/* 40d-4f9 Extended parameter information from master           */
          uchr   Res9[6];        /* 4fa-4ff reserved, do not use                                 */

          uchr   Res10[0x100];   /* 500-5ff reserved, do not use                                 */

    volatile uchr   cfgLen;         /* 600     Length of configuration check information from master + 5 */
          uchr   Res11[5];       /* 601-605 reserved, do not use                                 */
    volatile uchr   cfgChkData[244];/* 606-3f9 Configuration check information from master          */
          uchr   Res12[6];       /* 6fa-6ff reserved, do not use                                 */

          uchr   Res13[0x100];   /* 700-7ff reserved, do not use                                 */
} PBM_SLV;
#endif


/* definitions related to inpFC */
#define FC_DL       0x08
#define FC_DH       0x0a
#define FC_NAK_RS   0x02

#endif // _H__PBMCTL
```

6.2 Appendix B

6.2.1 Warranty

For warranty information pertaining to the card, refer to

http://www.molex.com/images/woodhead/woodhead_limited_warranty.pdf

6.2.2 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 and hardware configuration (other cards installed)
- Operating system type and version
- Details of the problem you are experiencing: application module type and version, target network, and circumstances that may have caused the problem

6.2.3 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. Technical notes for many supported third-party applications are also available on the Molex Web site.

North America

Canada:
Tel: +1-519-725-5136
Fax: +1-519-725-1515
Email: ic.support.na@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.

7 CE Notice

7.1 CISPR22 Compliance

This device meets or exceeds the requirements of the following standard:

- CISPR22:1997/EN 55022:1998 - Class A - Information technology equipment - Radio disturbance characteristics - Limits and methods of measurement

Warning

This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Caution

This equipment is neither designed for, nor intended for operation in installations where it is subject to hazardous voltages and hazardous currents.

Marking of this equipment with the symbol CE indicates compliance with European Council Directive 89/336/EEC - The EMC Directive as amended by 92/31/EEC and 93/68/EEC.

Note

To maintain compliance with the limits and requirements of the EMC Directive it is required to use quality interfacing cables and connectors when connecting to this device.