Where Automation Connects.

PTQ-PDPMV1

Quantum Platform

PROFIBUS DP Master Network Interface Module for Quantum

August 12, 2014

USER MANUAL

Your Feedback Please

We always want you to feel that you made the right decision to use our products. If you have suggestions, comments, compliments or complaints about our products, documentation, or support, please write or call us.

How to Contact Us

ProSoft Technology
5201 Truxtun Ave., 3rd Floor
Bakersfield, CA 93309
+1 (661) 716-5100
+1 (661) 716-5101 (Fax)
www.prosoft-technology.com
support@prosoft-technology.com

PTQ-PDPMV1 User Manual

August 12, 2014

ProSoft Technology® Product Documentation

In an effort to conserve paper, ProSoft Technology no longer includes printed manuals with our product shipments. User Manuals, Datasheets, Sample Ladder Files, and Configuration Files are provided on the enclosed CD-ROM in Adobe® Acrobat Reader file format (.PDFs). These product documentation files may also be freely downloaded from our web site: www.prosoft-technology.com

Information for ProTalk® Product Users

The statement "power, input and output (I/O) wiring must be in accordance with Class I, Division 2 wiring methods Article 501-10(b) of the National Electrical Code, NFPA 70 for installations in the U.S., or as specified in section 18-1J2 of the Canadian Electrical Code for installations within Canada and in accordance with the authority having jurisdiction".

The following or equivalent warnings shall be included:

A Warning - Explosion Hazard - Substitution of components may Impair Suitability for Class I, Division 2;
B Warning - Explosion Hazard - When in Hazardous Locations, Turn off Power before replacing Wiring Modules, and
C Warning - Explosion Hazard - Do not Disconnect Equipment unless Power has been switched Off or the Area is known to be Nonhazardous.
D Caution: The Cell used in this Device may Present a Fire or Chemical Burn Hazard if Mistreated. Do not Disassemble, Heat above 100°C (212°F) or Incinerate.

WARNING - EXPLOSION HAZARD - DO NOT DISCONNECT EQUIPMENT UNLESS POWER HAS BEEN SWITCHED OFF OR THE AREA IS KNOWN TO BE NON-HAZARDOUS.

AVERTISSEMENT - RISQUE D'EXPLOSION - AVANT DE DÉCONNECTER L'ÉQUIPEMENT, COUPER LE COURANT OU S'ASSURER QUE L'EMPLACEMENT EST DÉSIGNÉ NON DANGEREUX.

Warnings

North America Warnings

A Warning - Explosion Hazard - Substitution of components may impair suitability for Class I, Division 2.
B Warning - Explosion Hazard - When in hazardous locations, turn off power before replacing or rewiring modules. Warning - Explosion Hazard - Do not disconnect equipment unless power has been switched off or the area is known to be non-hazardous.
C Suitable for use in Class I, Division 2 Groups A, B, C and D Hazardous Locations or Non-Hazardous Locations.

ATEX Warnings and Conditions of Safe Usage

Power, Input, and Output (I/O) wiring must be in accordance with the authority having jurisdiction.

A Warning - Explosion Hazard - When in hazardous locations, turn off power before replacing or wiring modules.
B Warning - Explosion Hazard - Do not disconnect equipment unless power has been switched off or the area is known to be non-hazardous.
C These products are intended to be mounted in an IP54 enclosure. The devices shall provide external means to prevent the rated voltage being exceeded by transient disturbances of more than 40%. This device must be used only with ATEX certified backplanes.
D DO NOT OPEN WHEN ENERGIZED.

Warnings

Electrical Ratings

- Backplane Current Load: 1100 mA maximum @ 5 Vdc ± 5%
- Operating Temperature: 0°C to 60°C (32°F to 140°F)
- Storage Temperature: -40°C to 85°C (-40°F to 185°F)
- Shock: 30 g operational; 50 g non-operational; Vibration: 5 g from 10 Hz to 150 Hz
- Relative Humidity: 5% to 95% (with no condensation)
- All phase conductor sizes must be at least 1.3 mm(squared) and all earth ground conductors must be at least 4mm(squared).

Label Markings

<cULus>
E183151
Class I Div 2
Groups A,B,C,D T6
-30°C <= Ta <= 60°C
<Ex>
II 3 G
EEx nL IIc T6
-20°C <= Ta <= 60°C
Shock & Vibration tested to EN 60068 Standard

Agency Approvals and Certifications

CE
cULus
Shock & Vibration
CB Safety
GOST-R
RoHS
ATEX

Important Notice:

CAUTION: THE CELL USED IN THIS DEVICE MAY PRESENT A FIRE OR CHEMICAL BURN HAZARD IF MISTREATED. DO NOT DISASSEMBLE, HEAT ABOVE 100°C (212°F) OR INCINERATE.

Maximum battery load = 200 µA.

Maximum battery charge voltage = 3.4 Vdc.

Maximum battery charge current = 500 µA.

Maximum battery discharge current = 30 µA.

Contents

Guide to the PTQ-PDPMV1 User Manual

Function		Section to Read	Details
Introduction (Must Do)	→	Start Here (page 13)	This section introduces the customer to the module. Included are: package contents, system requirements, hardware installation, and basic configuration.
Diagnostic and Troubleshooting	→	Diagnostics and Troubleshooting (page 223)	This section describes Diagnostic and Troubleshooting procedures.
Reference Product Specifications	→	Reference (page 243) Product Specifications (page 244)	These sections contain general references associated with this product and its Specifications..
Support, Service, and Warranty Index	→	Support, Service and Warranty (page 293) Index	This section contains Support, Service and Warranty information. Index of chapters.

1 Start Here

In This Chapter

1.1 Hardware and Software Requirements

1.1.1 Quantum Hardware

This guide assumes that you are familiar with the installation and setup of the Quantum hardware. The following should be installed, configured, and powered up before you proceed:

- Quantum processor
- Quantum rack
- Quantum power supply
- Quantum Modbus Plus Network Option Module (NOM Module) (optional)
- Quantum to PC programming hardware
- NOM Ethernet or serial connection to PC

1.1.2 PC and PC Software

ProSoft Technology recommends the following minimum hardware to use the module:

- Windows PC with 80486 based processor (Pentium preferred) with at least one COM, USB, or Ethernet port
- 1 megabyte of system memory
- Unity™ Pro PLC programming software, version 3.0 or later
  or
  Concept™ PLC programming software, version 2.6 or later
  or
  Other Quantum Programming Software

Note: ProTalk module configuration files are compatible with common Quantum programming applications, including Unity Pro and Concept. For all other programming applications, please contact technical support.

1.2 Deployment Checklist

This is a list of the steps you must complete to install your ProTalk module. We recommend that you read this section completely before you begin the installation.

During this procedure, you will install the module in the rack with the processor, set up a PROFIBUS Master, connect one or more PROFIBUS slave devices, and then configure the processor with information about the PROFIBUS network. The example programs you will be configuring are designed to demonstrate that the module and the processor are correctly configured and communicating with each other over the backplane. After this initial installation, you may need to perform additional steps to configure the application for your specific needs.

You must complete these steps in the following order, otherwise the installation may not be successful.

1 Install the *ProSoft Configuration Builder* software on your PC

> **Important:** Earlier versions of ProSoft Configuration Builder do not support the Hot Standby (HSBY) feature on the PTQ-PDPMV1 module. To make full use of the HSBY feature, please download the latest version of ProSoft Configuration Builder and review the readme files from the ProSoft Technology website at www.prosoft-technology.com/pcb.

2 Install the ProTalk module in the rack
3 Configure the module
4 Configure the PROFIBUS Master and slaves
5 Export the processor files
6 Configure the processor
7 Verify communication between the processor and the module

1.3 Installing ProSoft Configuration Builder Software

You must install the *ProSoft Configuration Builder (PCB)* software to configure the module. You can always get the newest version of *ProSoft Configuration Builder* from the ProSoft Technology website.

To install ProSoft Configuration Builder from the ProSoft Technology website

1 Open your web browser and navigate to *http://www.prosoft-technology.com/pcb*
2 Click the **DOWNLOAD HERE** link to download the latest version of *ProSoft Configuration Builder*.
3 Choose **SAVE** or **SAVE FILE** when prompted.
4 Save the file to your *Windows Desktop*, so that you can find it easily when you have finished downloading.
5 When the download is complete, locate and open the file, and then follow the instructions on your screen to install the program.

If you do not have access to the Internet, you can install *ProSoft Configuration Builder* from the *ProSoft Solutions Product CD-ROM*, included in the package with your module.

To install ProSoft Configuration Builder from the Product CD-ROM

1 Insert the *ProSoft Solutions Product CD-ROM* into the CD-ROM drive of your PC. Wait for the startup screen to appear.
2 On the startup screen, click **PRODUCT DOCUMENTATION**. This action opens a *Windows Explorer* file tree window.
3 Click to open the **UTILITIES** folder. This folder contains all of the applications and files you will need to set up and configure your module.
4 Double-click the **SETUP CONFIGURATION TOOL** folder, double-click the **PCB_*.EXE** file and follow the instructions on your screen to install the software on your PC. The information represented by the "*****" character in the file name is the *PCB* version number and, therefore, subject to change as new versions of *PCB* are released.

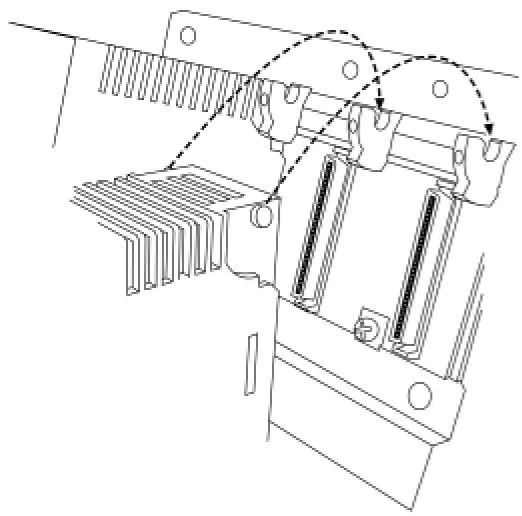
Note: Many of the configuration and maintenance procedures use files and other utilities on the CD-ROM. You may wish to copy the files from the *Utilities* folder on the CD-ROM to a convenient location on your hard drive.

1.4 Installing the Module

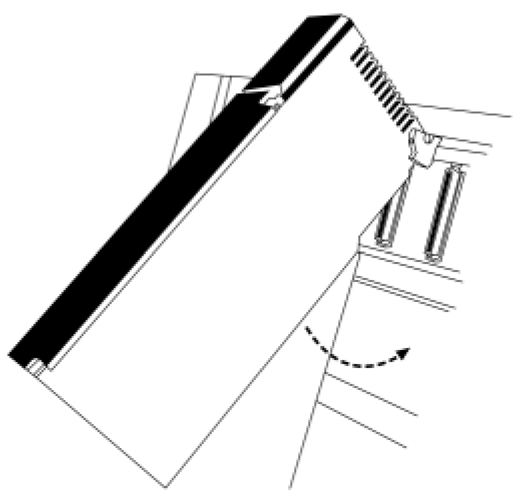
1.4.1 Installing the ProTalk Module in the Quantum Rack

1 Place the module in the Quantum rack. The ProTalk module must be placed in the same rack as the processor.

2 Tilt the module at a 45° angle and align the pegs at the top of the module with the slots on the backplane.



3 Push the module into place until it seats firmly in the backplane.



CAUTION: The PTQ module is hot-swappable, meaning that you can install and remove it while the rack is powered up. You should not assume that this is the case for all types of modules unless the user manual for the product explicitly states that the module is hot-swappable. Failure to observe this precaution could result in damage to the module and any equipment connected to it.

HSBY Note: For HSBY setup, repeat the above procedures for the Primary and Standby modules.

1.4.2 Connecting to the ProTalk Configuration/Debug Port

> **Note:** The module has a serial port as well as an Ethernet port. The first time you connect to the module to configure it, you can connect to the module's serial port using the supplied null-modem cable, because the module's default Ethernet settings may not match your network.
> **HSBY Note:** For HSBY units the Ethernet connection must be applied. This connection is used as a backup to ping status messages over the PROFIBUS network. It is also used for DPV1 remote (passive) Master buffer update during switchover.

PC to Ethernet Port Connection

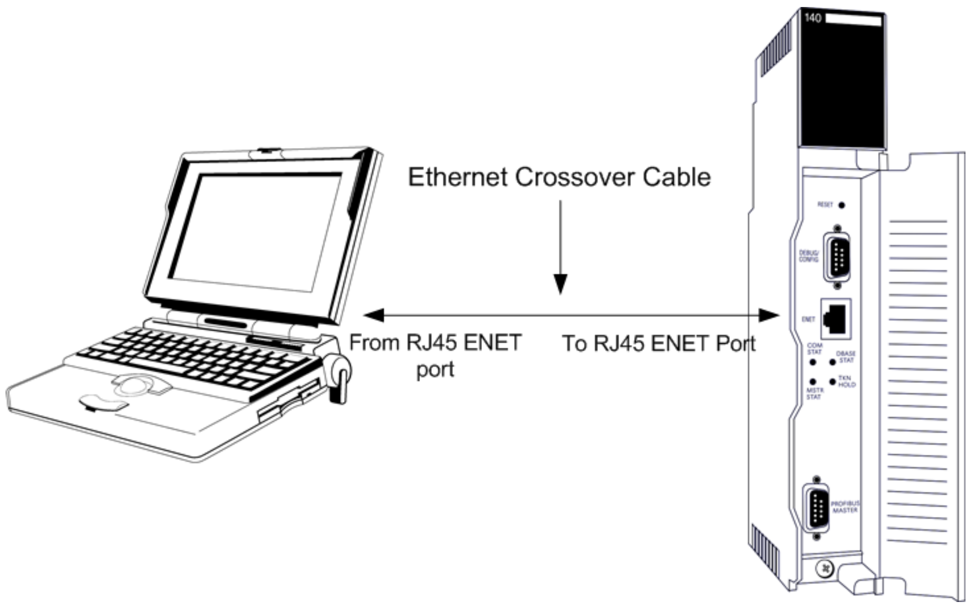
> **Important:** The PTQ-PDPMV1 module is equipped to use an Ethernet connection using the following defaults:

```
My_ip:    192.168.0.100
Netmask:  255.255.255.0
Gateway:  192.168.0.1
```

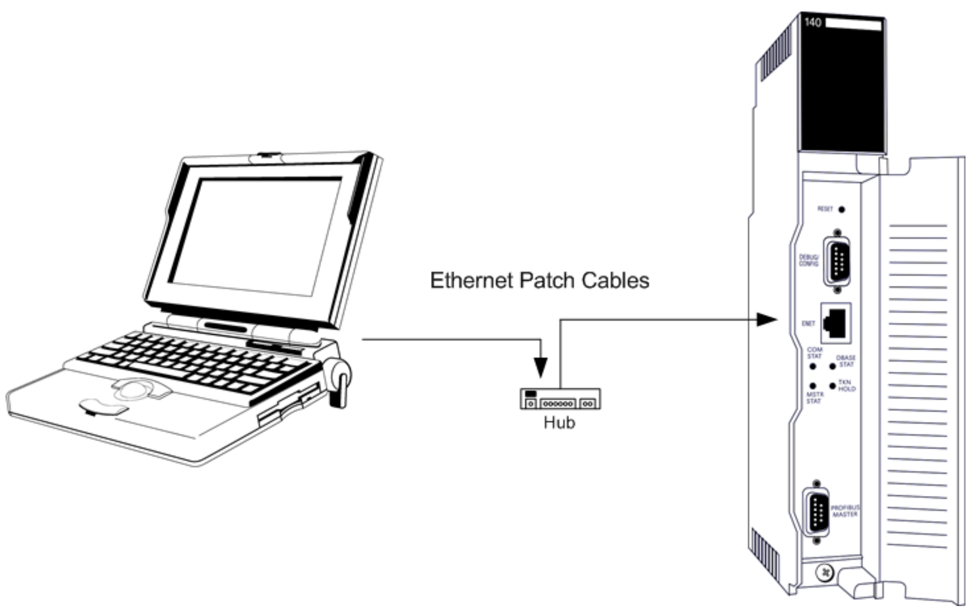
> **HSBY Note:** For HSBY units the remote (passive) Master module Ethernet connection is always Primary IP plus 1. For example, Primary IP = 192.168.0.100, Standby module IP = 192.168.0.101. This setting is not configurable: the module's firmware automatically sets the IP address of the remote (passive) Master.

If you cannot use these defaults for your connection, you must change them using ProSoft Configuration Builder and then download the new values to the PTQ-PDPMV1 module, either through a serial cable, or by using a Compact Flash (CF) writer. If you need to change the Ethernet addresses, use ProSoft Configuration Builder to change the values in the WATTCP file.

If the default values are valid on your network, and you are using an Ethernet connection, please connect your computer to the PTQ-PDPMV1 module using either of the methods described below:



Computer to Ethernet Port Connection via Hub



1.4.3 PTQ-PDPMV1 Configuration / Debug Port Note

After the Ethernet settings are correctly configured, only the Ethernet port should be used for configuration changes, diagnostics, and PROFIBUS monitoring.

2 Configuring the Module

In This Chapter

2.1 Configuring the Module with ProSoft Configuration Builder

In this step of the setup process, you will use ProSoft Configuration Builder to configure the parameters that affect the interface between the PTQ module and the processor (Quantum or Unity). These parameters indicate:

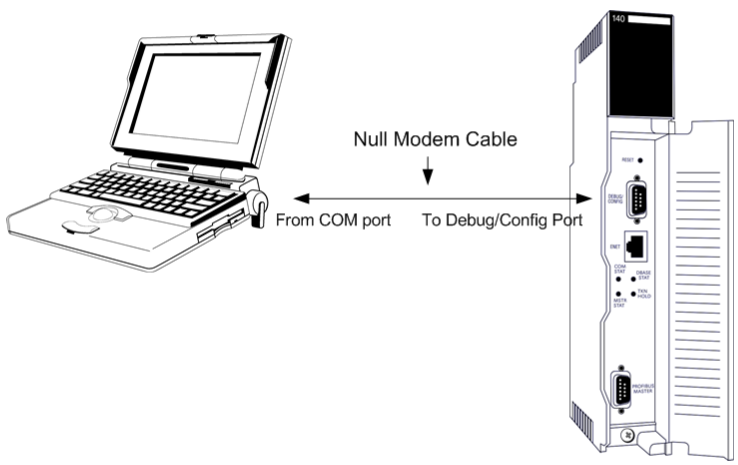
- The physical position of the module in the rack.

> **HSBY Note:** For HSBY units, the local (active) and passive modules must be placed in the same rack location in both racks.

- The starting memory address in the processor's State RAM for the module's input and output data images. For the purpose of this example, we use a starting address of 1000 for the input image and 3000 for the output image.

To begin, verify that the processor is correctly positioned in the rack, and is powered up. Connect your PC to the PTQ-PDPMV1 module using the supplied Null Modem serial cable, as shown in the following illustration.

> **Note:** The serial port should only be used for initial configuration of the Ethernet port through ProSoft Configuration Builder.

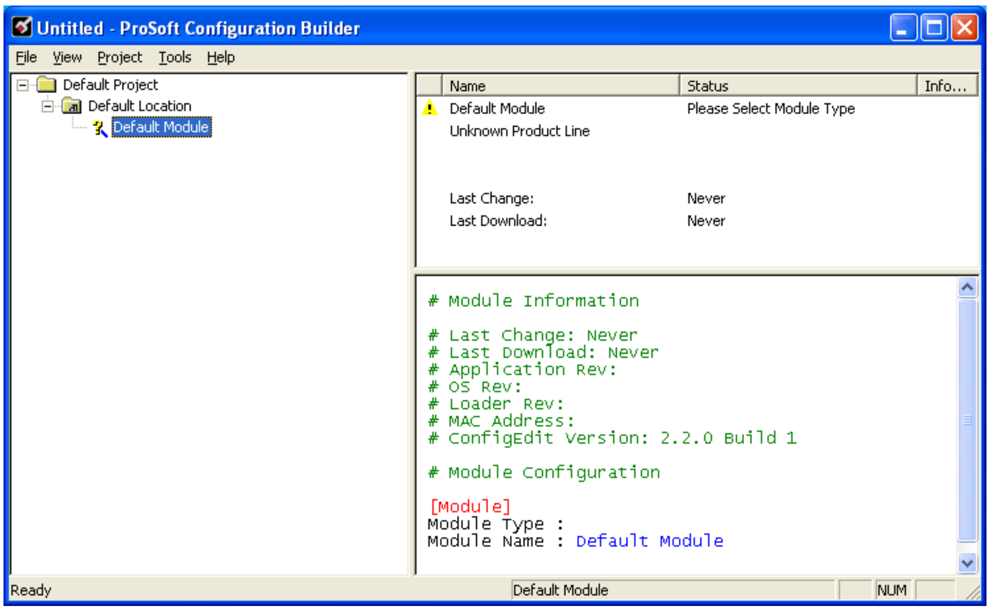


After the Ethernet settings are correctly configured, only the Ethernet port should be used for configuration changes, diagnostics, and PROFIBUS monitoring.

2.1.1 Setting Up the Project

To begin, start *ProSoft Configuration Builder*. If you have used other Windows configuration tools before, you will find the screen layout familiar. *ProSoft Configuration Builder's* window consists of a tree view on the left, and an information pane and configuration pane on the right side of the window.

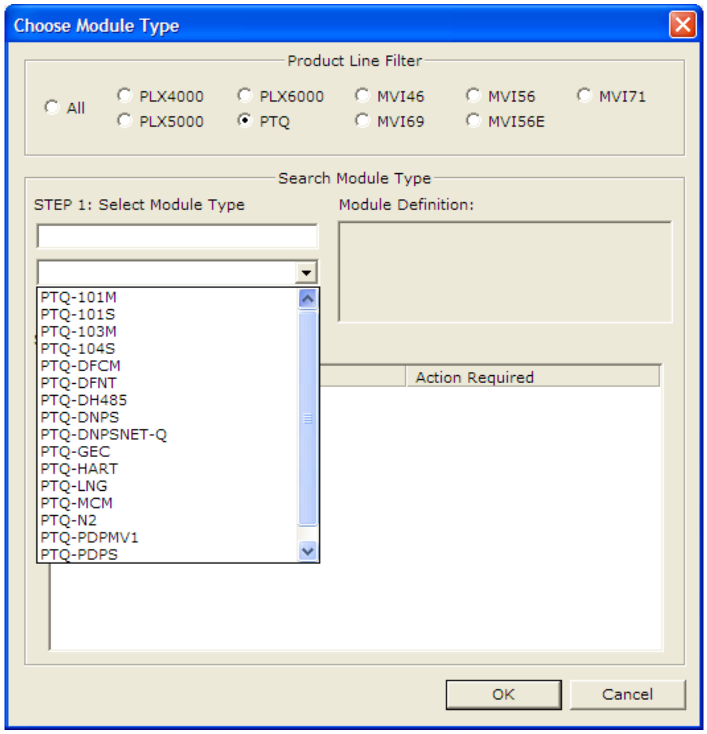
When you first start ProSoft Configuration Builder, the tree view consists of folders for *Default Project* and *Default Location*, with a *Default Module* in the *Default Location* folder. The illustration below shows the *ProSoft Configuration Builder* window with a new project.



Your first task is to add the PTQ-PDPMV1 module to the project.

1 Use the mouse to select **DEFAULT MODULE** in the tree view, and then click the right mouse button to open a shortcut menu.

2 On the shortcut menu, select **CHOOSE MODULE TYPE**. This action opens the *Choose Module Type* dialog box.



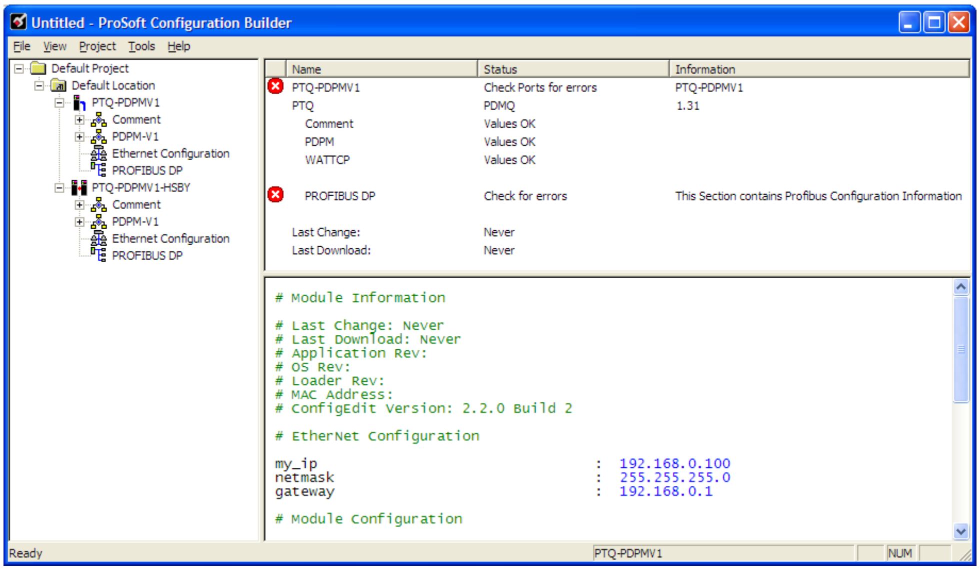
HSBY Note: For Hot Standby support, select the Enable "Hot Standby" checkbox.

3 In the *Product Line Filter* area of the dialog box, select **PTQ**. In the *Select Module Type* dropdown list, select **PTQ-PDPMV1**, and then click **OK** to save your settings and return to the *ProSoft Configuration Builder* window.

The next task is to set the module parameters.

2.1.2 Setting Module Parameters

Notice that the contents of the information pane and the configuration pane changed when you added the PTQ-PDPMV1 module to the project. The red "X" icon indicates that the module's configuration is incomplete.

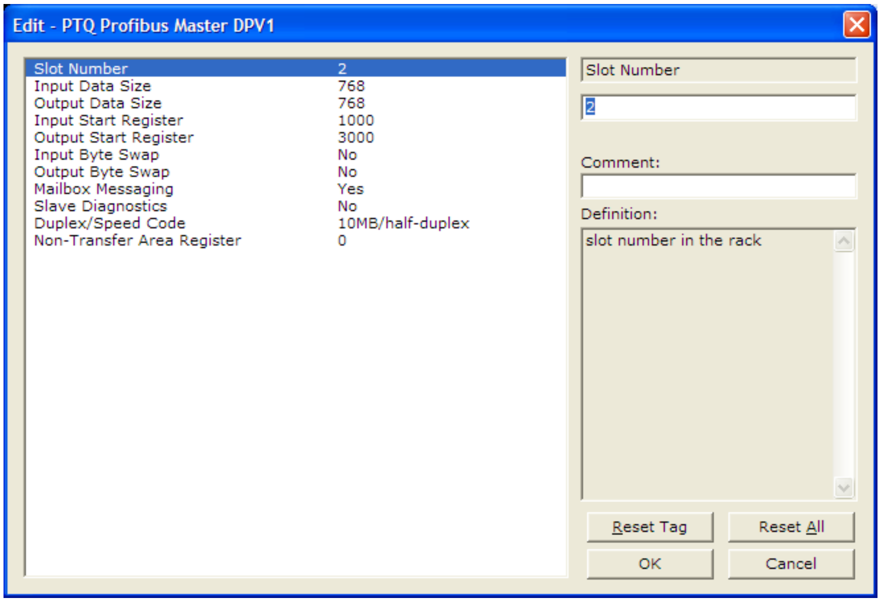


HSBY Note: For Hot Standby modules, a double module icon will be displayed.

In the following steps, you will provide the missing information to begin configuring the module.

1. Click the plus sign **[+]** next to the module to expand the module tree, and then expand the **PDPM-V1** tree.
2. Double-click the **PTQ PROFIBUS MASTER DPV1** object. This action opens the *Edit* dialog box.

3 In the *Edit* dialog box, change the values for the selections in this section of the configuration to match the values in the following illustration. To change a value, select the parameter to modify in the left pane, and then type the new value in the edit field in the right pane. If you are not sure what to enter here, use the default values.



Note: The values you enter for the purpose of this example configuration are used by the sample program that you will download to the processor later in this section. You may need to change these values as you implement your production system. Use the following chapters for your Quantum or Unity configuration software, or the online help system, for detailed information on each of the parameters associated with the module.

Slot Number

The *Slot Number* is the physical location of the module in the rack. The example here assumes a basic configuration with a power supply occupying the first slot, the processor occupying the next two slots, and the PTQ-PDPMV1 module occupying the fourth slot. In this case the module would be in slot 4.

> **Note:** If the module is not placed in the slot number specified, the module will not operate, and the CFG ERR light will illuminate. You must specify the actual slot number for the module in the module configuration file.

```
MODULE CONFIGURATION:

PTQ-PDPMV1_MBANDDIAG

Slot Number           :   4      (Found In Slot Number 4)
Profibus Input Size   : 768     (words)
Profibus Output Size  : 768     (words)
Input Start Reg       : 1000      Total Input Size  : 1369    (words)
Output Start Reg      : 3000      Total Output Size : 918     (words)
```

Input Data Size

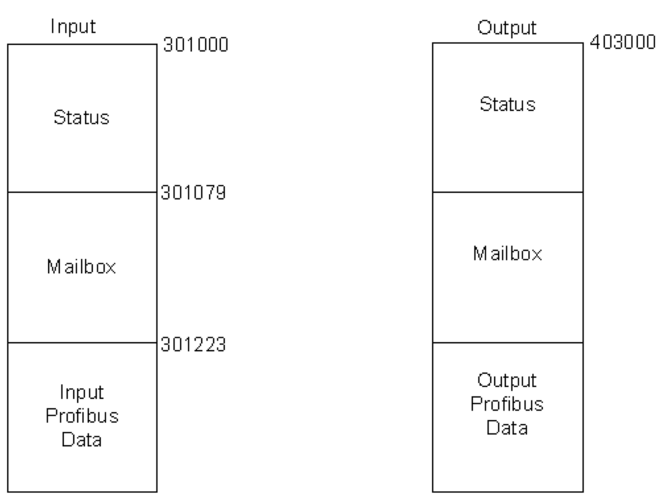
Number of PROFIBUS input point words. Leave this setting at its default value of 768 words.

Output Data Size

Number of PROFIBUS output point words. Leave this setting at its default value of 768 words.

Start Registers

Layout of I/O blocks



The *Input Start Register* address refers to the 3x (%IW) location in the processor's State RAM and the *Output Start Register* refers to the 4x (%MW) location of State RAM. You can view State RAM information in Unity XL Pro.

A common mistake is to assume that because the *Input Start Register* parameter starts at address 301000, then the PROFIBUS data associated with the slaves will also start at the same register. As the diagram above shows, the Input PROFIBUS Data would start at address 301223 for this example.

Important: The *Input* and *Output Start Register* parameters define the start registers for the input and output blocks that are transferred between the processor and the module. The PROFIBUS I/O associated to the slaves is part of these blocks. Refer to PTQ Input and Output Data Blocks (page 251) for a description. Each block contains status, PROFIBUS data, and Mailbox/Slave diagnostics, if chosen.

Input Start Register

The *Input Start Register* address refers to the 3x (%IW) location in the processor's State RAM. You can view State RAM information in Unity XL Pro.

Output Start Register

The *Output Start Register* refers to the 4x (%MW) location of State RAM. You can view State RAM information in Unity XL Pro.

Input Byte Swap

Swap bytes in input image (**YES** or **NO**). The default value is **NO**.

This is a user-configured flag to indicate if input data is swapped before being placed in the input image for the controller. If the parameter is set to **0**, no swapping occurs. If it is not **0**, then bytes are swapped.

For more information on byte swapping, please refer to Status Data in the Input Data Block (page 256).

Output Byte Swap

Swap bytes in output image (**YES** or **NO**). The default value is **NO**.

This is a user-configured flag to indicate if output data is swapped after being received from the controller. If the parameter is set to **0**, no swapping occurs. If it is not **0**, then bytes are swapped.

Mailbox Messaging

Use mailbox messaging over the backplane (**Y** or **N** with **Y**=default).

For this example, leave the setting at its default. For more information on the effect of this setting, please refer to Mailbox Messaging (page 151).

Slave Diagnostics

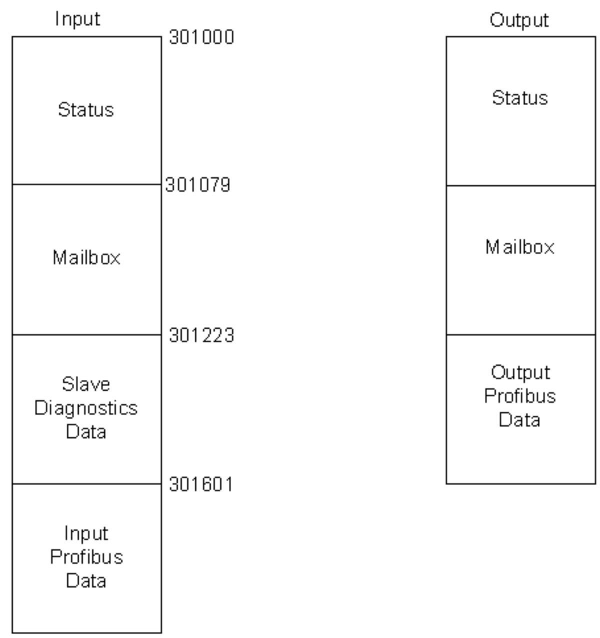
Get slave diagnostic data (**Y**/**N** with **N**=default).

For this example, leave the setting at its default.

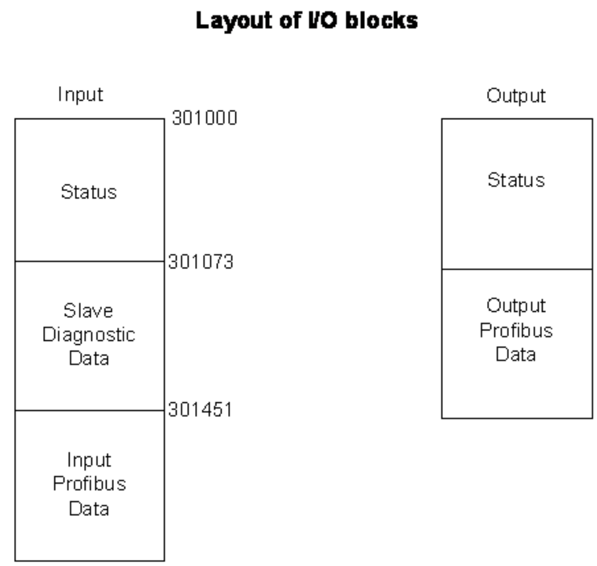
If you change the default value of this setting and the previous one (*Mailbox Messaging*) from their default values, the layout of the I/O blocks changes.

The following diagram shows the layout of the I/O blocks when *Mailbox Messaging* is set to **YES** (the default value), and *Get Slave Diagnostic Data* is set to **YES**.

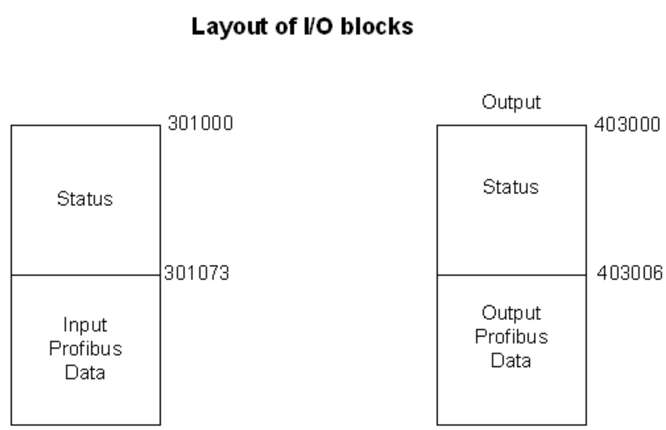
Layout of I/O blocks



The following diagram shows the layout of the I/O blocks if *Mailbox Messaging* is set to **NO**, and *Slave Diagnostics* to **YES**.



The following diagram shows the layout of the I/O blocks if *Mailbox Messaging* is set to **NO**, and *Slave Diagnostics* to **NO**.



In ProSoft Configuration Builder, the *Show Concept Map* and *Show Unity Map* commands show the layout of the entire input and output backplane blocks.

Refer to Input and Output Data Block Format (page 251) for detailed information on the contents of these blocks, and a discussion of how various configuration options change the layout of these blocks.

Duplex/Speed Code

0=Auto-negotiate

1=10 MB / half-duplex

2=10 MB / full-duplex

3=100 MB / half-duplex

4=100 MB / full-duplex

This parameter allows you to set the connection speed manually between 10 Mbps full / half-duplex and 100 Mbps full / half-duplex or to auto-negotiate the baud rate with a hub or switch. The default value is 10 MB / half-duplex.

Non-Transfer Area Register

> **Note:** This configuration option is only available for Hot Standby operation.

If this parameter is set to **0**, the PROFIBUS configuration CRC will be derived from the Output Area.

If this parameter is set to a value **GREATER THAN 0**, the PROFIBUS configuration CRC will be derived from the specified Non-Transfer Area register.

You can specify a register outside the Output Area to derive the CRC value for the PROFIBUS configuration. This parameter allows you to modify the PROFIBUS DP network configuration without stopping the system.

This data area uses 4 words, and must be located in the 4x memory area. The module will attempt to read this data asynchronously from the non-transfer data area. When new values are received, they are placed in the normal area used by the program. Because this operation is asynchronous to the scan, it may take 2 or more scans for the data to update.

Completing the Example Configuration

When you have finished updating the values, click **OK** to save your settings and return to the *ProSoft Configuration Builder* window.

At this time, you may wish to rename the *Default Project* and *Default Location* folders in the tree view. To rename an object:

1. Select the object, and then click the right mouse button to open a shortcut menu. From the shortcut menu, choose **RENAME**.
2. Type the name to assign to the object.
3. Click away from the object to save the new name.

The next task is to update the module's Ethernet settings. This allows you to connect from your computer to the module using an Ethernet cable rather than a serial cable.

2.1.3 Updating the Ethernet Settings

Use this procedure to configure the Ethernet settings for your module. You must assign an IP address, subnet mask and gateway address. After you complete this step, you can connect to the module with an Ethernet cable.

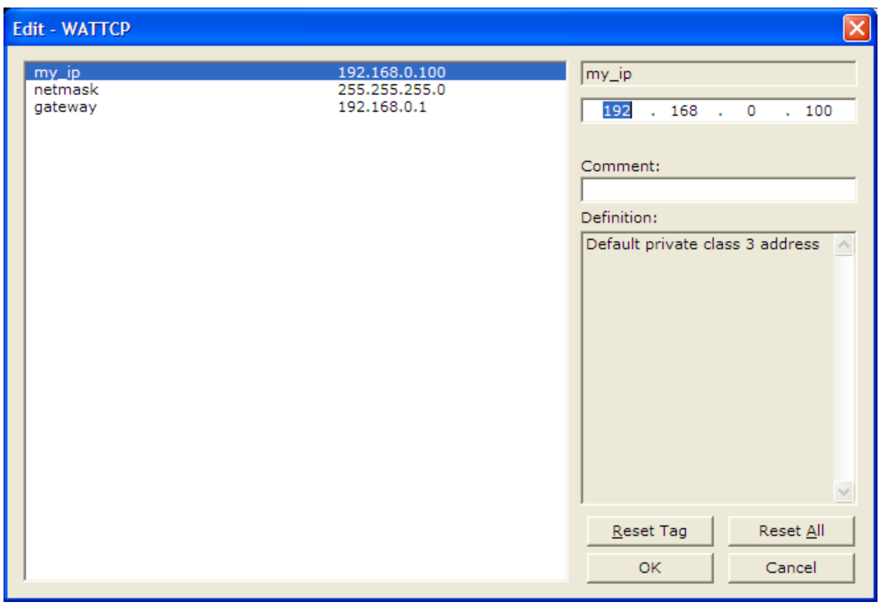
1 Determine the network settings for your module, with the help of your network administrator if necessary. You will need the following information:

- IP address (fixed IP required) _____ . _____ . _____ . _____
- Subnet mask _____ . _____ . _____ . _____
- Gateway address _____ . _____ . _____ . _____

HSBY Note: Hot Standby Primary IP is entered. The Standby IP address will always be the Primary IP address plus 1.

2 Click **[+]** to expand the tree for the PTQ-PDPMV1 module.

3 Double-click the **ETHERNET CONFIGURATION** object. This action opens the *Edit* dialog box.



4 Edit the values for my_ip, netmask (subnet mask) and gateway (default gateway).

5 When you are finished editing, click **OK** to save your changes and return to the *ProSoft Configuration Builder* window.

2.2 Downloading the Ethernet Configuration to the Module

In order for your changes to take effect, you must download (copy) the updated Ethernet configuration from your computer to the module.

1. Connect the serial cable between the module and the computer.
2. Select the **ETHERNET CONFIGURATION** icon, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **DOWNLOAD**. This action sends the new IP settings to the module, allowing Ethernet communication between the computer and the module.

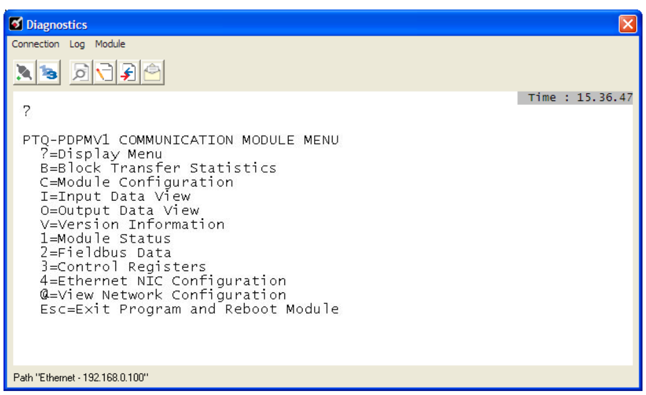
> **HSBY Note:** This serial download procedure must be performed for both Master HSBY modules.
> **Note:** The processor (Quantum or Unity) must be in STOP mode before you download the file to the module. Use the processor's softkeys on the display keypad, or the processor's configuration program to stop the processor.

The final step is to verify that ProSoft Configuration Builder can communicate with the module using an Ethernet connection.

1. Plug in an Ethernet cable between the module and an Ethernet hub or router.

> **HSBY Note:** You must leave the Ethernet cable connected to both Hot Standby modules at all times. The configuration download will not proceed unless both modules are connected.

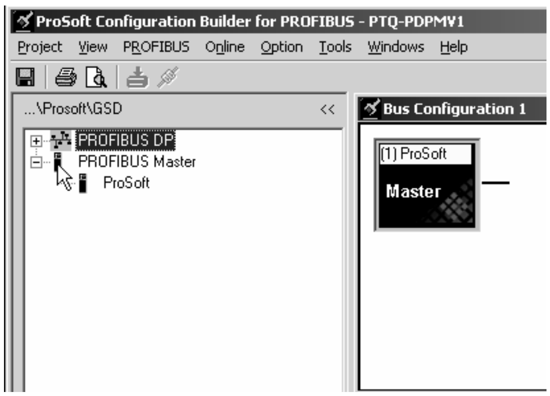
2. In the tree view in *ProSoft Configuration Builder*, click once to select the PTQ-PDPMV1 module.
3. Open the **PROJECT MENU,** and then choose **MODULE,** and then choose **DIAGNOSTICS.** This action opens the *Diagnostics* window.
4. Choose Ethernet as the connection type, and then enter the IP address. Press the **[?]** key on your keyboard. If the module is communicating successfully, you will see a menu like this:



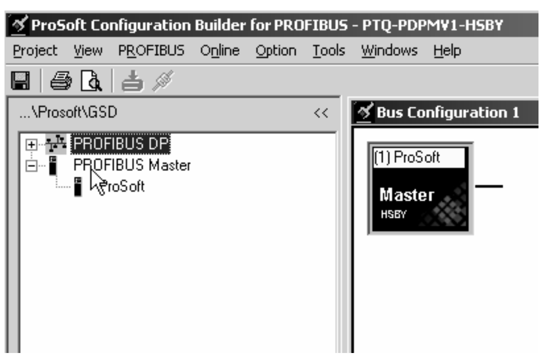
2.3 Configuring the PROFIBUS Master

In this task, you will configure the PROFIBUS Master, and then add PROFIBUS slaves to the network. When this step is complete, you will download the configuration information to the PTQ module. You will also export the I/O maps for the processor.

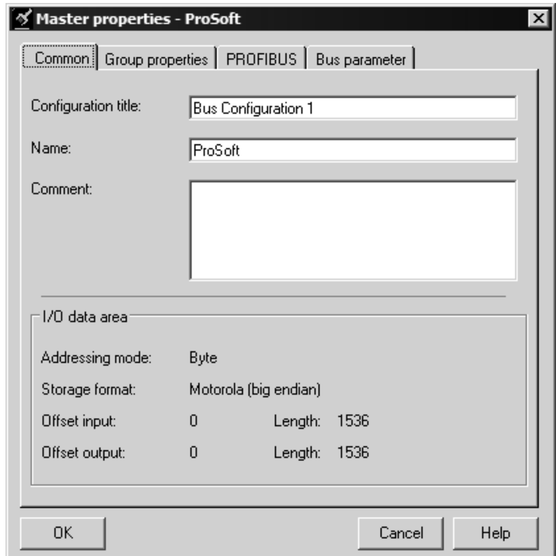
1. In *ProSoft Configuration Builder* tree view, click **[+]** to expand the PTQ-PDPMV1 tree, and then double-click the *PROFIBUS DP* icon. This action opens the *PROFIBUS Master Setup* dialog box.
2. Click the **CONFIGURE PROFIBUS** button. This action opens the *ProSoft Configuration Builder for PROFIBUS* application.
3. Click **[+]** to expand the PROFIBUS Master tree.
4. Drag the *ProTalk* icon into the *Bus Configuration* window. This is automatically done by the software for new applications.



For HSBY Units



5 Double-click the **PROFIBUS MASTER** icon in the *Bus Configuration* window. This action opens the *Master Properties* dialog box.



6 On the **COMMON** tab, name your PROFIBUS drop. The name should match the module name from step 4 in this procedure.

Note: The *PROFIBUS* tab contains the address setting and advanced configuration settings for the Master. The default settings on this tab work best in most applications.
HSBY Note: The correct profile setting for HSBY Master is *DP*; however, the Hot Standby check box will be checked. The minimum baud for Hot Standby module to switch over within 300 ms with an average processor scan time of 100ms, is 1500Kbits/second.

7 Click **OK** to save your changes and return to the *Bus Configuration* window.

2.3.1 Installing the GSD Files

The GSD configuration files contain information on PROFIBUS slaves that you can configure as part of your PROFIBUS network. In order for this configuration information to be available in ProSoft Configuration Builder, you must install the GSD files.

Tip: GSD configuration files for popular Schneider Electric and ProSoft Technology modules are included with the installation. If you have other GSD files for your PROFIBUS slaves, copy them into C:\Documents and Settings\All Users\Application Data\ProSoft\GSD (Windows XP / 2000) or C:\My Documents\ (Windows 98), and ProSoft Configuration Builder will load them automatically.

To install GSD files manually

1 In ProSoft Configuration Builder tree view, click **[+]** to expand the *PTQ-PDPMV1* tree, and then double-click the **PROFIBUS DP** icon. This action opens the *PROFIBUS Master Setup* dialog box.
2 Click the **CONFIGURE PROFIBUS** button. This action opens the *ProSoft Configuration Builder for PROFIBUS* application.
3 Click **[+]** to expand the *PROFIBUS DP* tree.
4 Click the right mouse button to open a shortcut menu.
5 On the shortcut menu, choose **INSTALL NEW GS* FILE**. This action opens a dialog box that allows you to browse for the location of the GSD configuration files to install.
6 Choose the file to install, and then click **OPEN**. If the file already exists in the *configuration file path* (see Tip above), you will be prompted to overwrite the file.
7 You will be prompted to associate the GSD configuration file with a bitmap image of the slave device. Use the *File Open* dialog box to browse for the location of the image file to use.

Note: This procedure does not automatically copy GSD configuration files from their original location to the GSD file path. In order to load GSD files automatically the next time you start ProSoft Configuration Builder, copy the files to the configuration file path in the Tip above.

2.3.2 Configuring the PROFIBUS Slaves

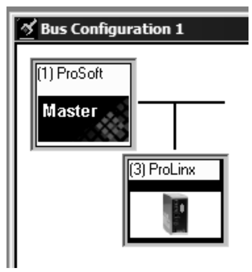
There are two essential steps to configuring a slave:

1 Add the slave in ProSoft Configuration Builder (PCB) as a device connected to the PROFIBUS Master, specifying the slave address and any necessary input and output configuration. Download the PROFIBUS Master configuration to the PTQ-PDPMV1 module.
2 Configure the slave (using PCB or the configuration tool supplied by the manufacturer, for some PROFIBUS slaves). Verify that the slave address configured in the slave module matches the slave address configured in PCB. Download the PROFIBUS Slave configuration to the slave module.

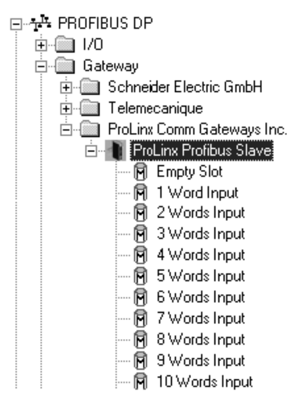
Scanning for Slaves Manually

In this part of the procedure, you will add and configure the PROFIBUS slaves. In the following steps, you will add and configure a ProLinx PROFIBUS slave module. The configuration information (.GSD file) for this module is provided on the PTQ-PDPMV1 Solutions CD-ROM.

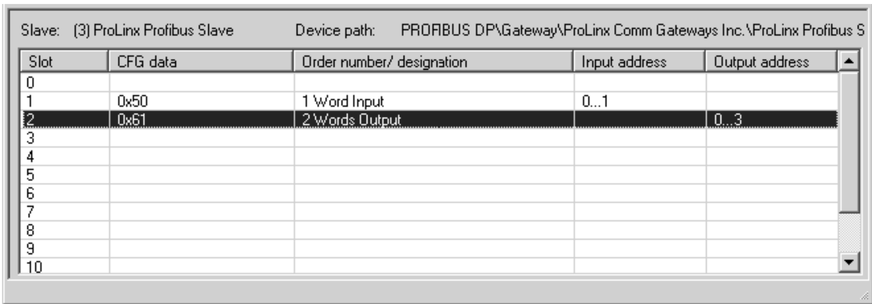
1. In *ProSoft Configuration Builder for PROFIBUS*, click the plus sign **[+]** to expand the *PROFIBUS DP* tree.
2. Navigate to the folder containing the type of slave device to add, and then click the plus sign **[+]** to expand the folder.
3. Drag the **SLAVE** icon into the *Bus Configuration* window. The slave device appears in the *Bus Configuration* window as a network location to the Master.



4. In the tree view, click the plus sign **[+]** to expand the slave device you added. This action opens a list of device configuration values. The following illustration shows the device configuration values for a ProLinx PROFIBUS slave. The values for other devices may be different, so you should review the specifications for the product you are installing in order to determine the correct values to use.

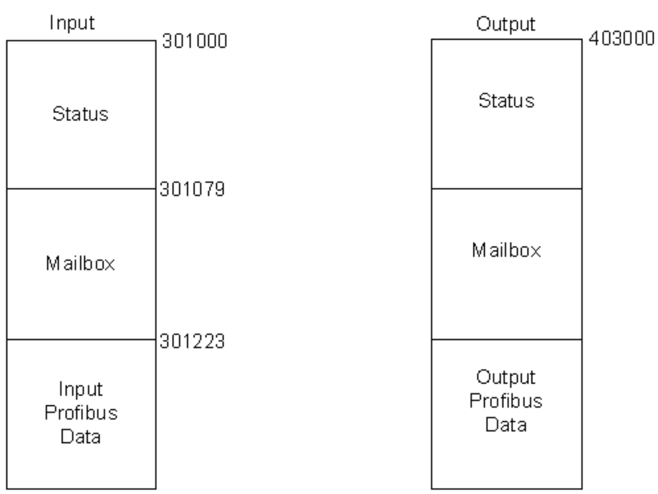


5 Drag the *input* and *output* parameters to the slot location grid below the *Bus Configuration* window. This view displays the configuration data, order number, and starting input and output addresses.



Important: The starting input and output addresses that you select here are actually byte offsets within the PROFIBUS Data area inside each Input and Output backplane block.
For example, for the sample configuration for the input block, where the Input Start Register Parameter = 1000:

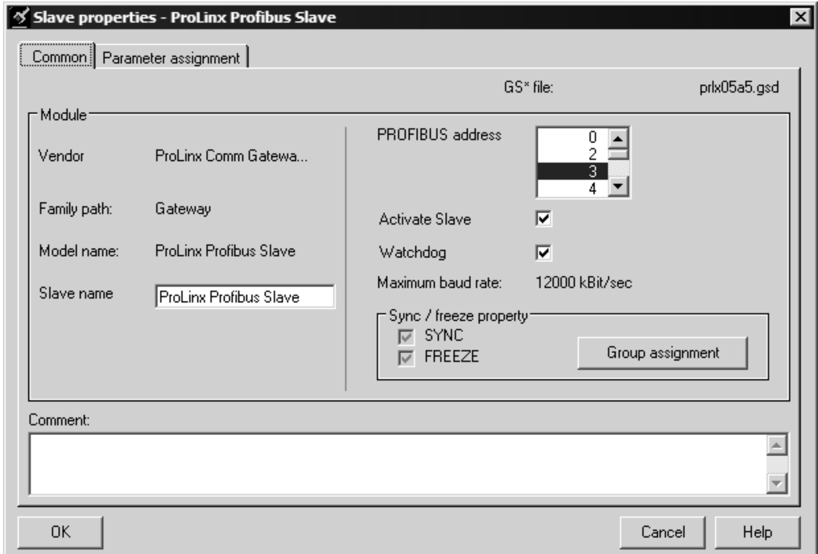
Layout of I/O blocks



The following table shows the actual Quantum address:

Input Address configured in PCB (Bytes)	Actual Quantum Input Register Address (Words)
0...1	301223
2...3	301224
4...5	301225
...	...

6 Double click the **SLAVE** icon to view the *Slave properties*.



In particular, note the following settings:

- Automatic PROFIBUS Address Assignment:

  ProSoft Configuration Builder automatically assigns a PROFIBUS address to each new slave. The address assignment begins at address 3, and is incremented by 1 for each new slave added to the network. You can change the address in the **COMMON** tab of the *Slave Properties* dialog box.

- Automatic Input/Output Address Assignment:

  For each new slave added to the PROFIBUS network, ProSoft Configuration Builder automatically converts the input/output byte addresses to word input/output addresses for the State RAM in the processor.

7 Repeat steps 2 through 6 for all slaves you intend to place on the network.

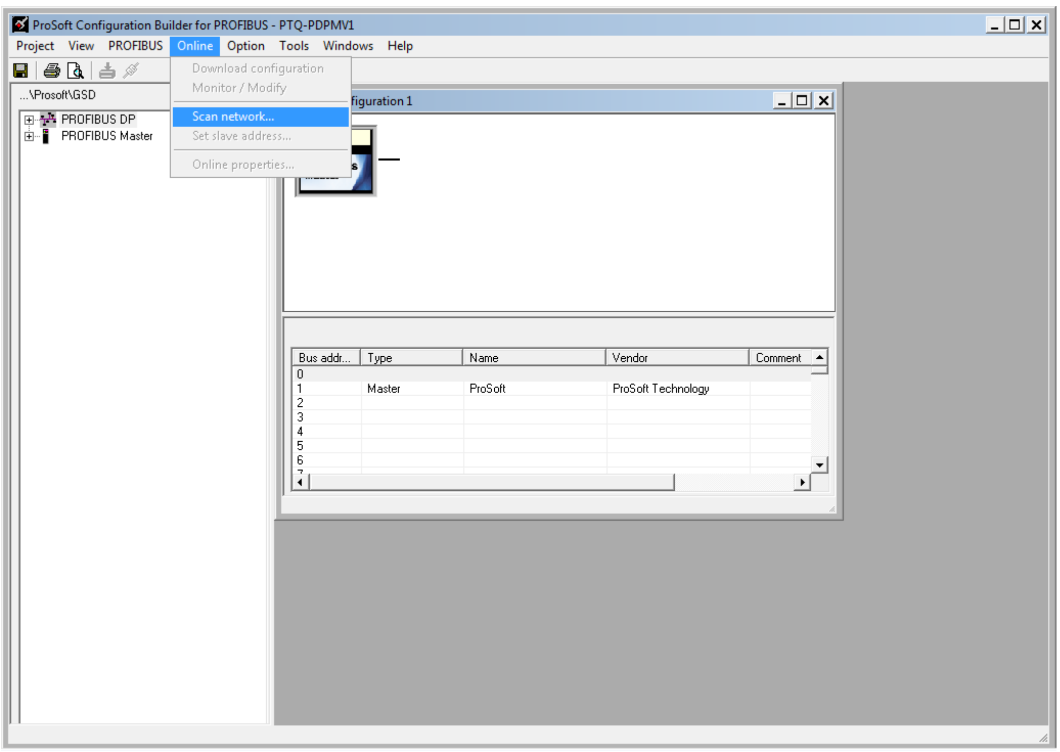
8 When you are finished adding slaves, open the **PROJECT** menu and choose **EXIT** to return to the *Master Setup* dialog box.

Using The Autoscan Feature

The concept of *Automatic network scanning* means that the user can instruct the *Bus Configuration* window to automatically gather information about slaves that are connected to the network. When the scan is completed the user can adopt the detected slaves to the *bus configuration* and download to the Master.

This is a quick way to get a network up and running. However, one should be aware that it is not guaranteed that any particular slave will enter data exchange since the user parameter data might not match. This is especially obvious if no associated GSD-file is found during the network scan, this means that no user parameter data would be sent to the slave.

NETWORK SCAN is selectable from the *Online* menu as well as from the drop-down menu for the **MASTER** icon.

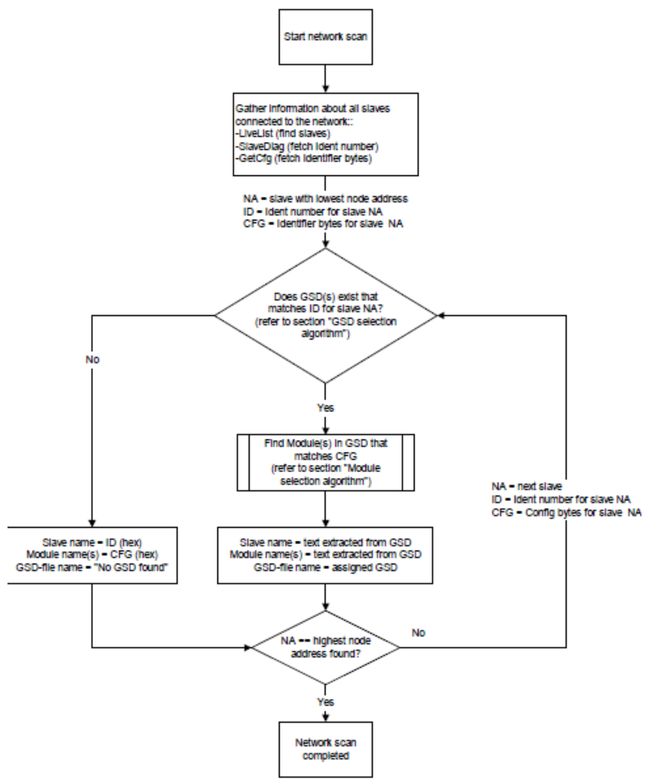


When the download is completed, the *PROFIBUS Master Configuration* window will initialize the Master to operate as a *Class 2 Master only*. In this mode it is possible to initialize the Master even if the database does not contain any slaves.

After successful initialization, the *PROFIBUS Master Configuration* window will issue the following mailboxes in order to gather information about the connected slaves:

1 1. Send FB_APPL_GET_LIVE_LIST in order to detect connected slaves,
2 2. Send FB_APPL_GET_SLAVE_DIAG (external request) to all devices identified as slaves according to the Live list.
3 3. Send FB_APPL_GET_SLAVE_CONFIG to all devices identified as slaves according to the Live list.

When the information is collected the *PROFIBUS Master Configuration* window will find a matching GSD-file and extract information from it. Refer to the flowchart below for this sequence:



GSD Selection Algorithm

If two or more matching GSD-files are found, the first one found should be selected. The other compatible files should be stored so that the user can select one of them instead. If the user selects another GSD-file, the *PROFIBUS Master Configuration* window will run through the *Module Selection Algorithm* (described below) again.

Module Selection Algorithm

The algorithm used to find modules in the GSD based on the Identifier byte(s) is as follows:

Select the module that matches the largest number of Identifier bytes. If the GSD contains two or more modules with the exact set of Identifier bytes, use the first module found.

Example:

If a slave responds with identifier bytes: 0x11, 0x21, 0x31 and that the associated GSD-file contains five modules: “A” = 0x11, “B” = 0x21, “C” = 0x31, “AB” = 0x11, 0x21 and “BC” = 0x21, 0x31. The *PROFIBUS Master Configuration* window will then select modules "AB" and "C".

Note: If no matching module is found in the GSD, The *PROFIBUS Master Configuration* window will display the identifier byte(s) instead.

Network scan window

The information extracted from the GSD-file(s) will be displayed in the *Network scan* window.

Select

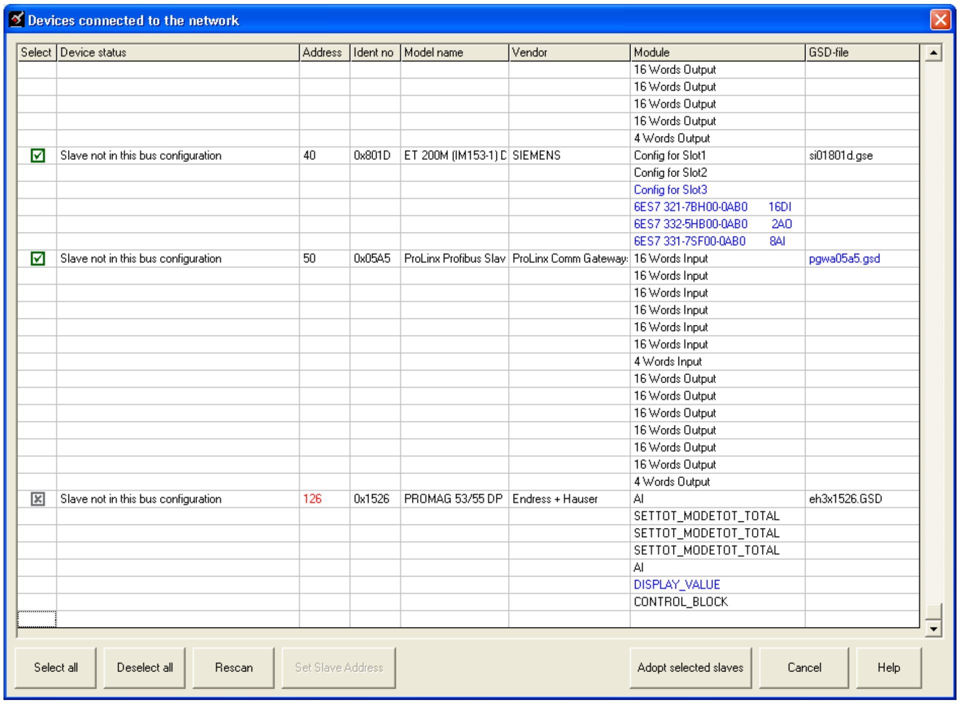
In this column all found slaves will be marked as selected by default, except for slaves with the special address 126 (refer to the next section that describes the Address column). Only selected slaves will be added to the *PROFIBUS Master Configuration* when the **ADOPT SELECTED SLAVES** button is clicked.

Address

In this column the node address of the slaves will be displayed. Found slaves should be listed in ascending order according to their node addresses.

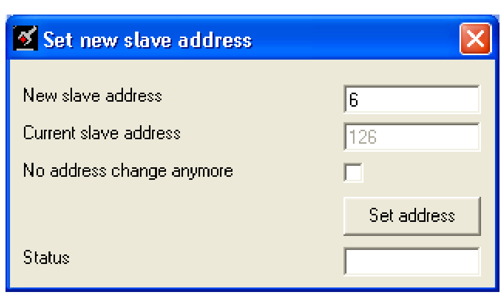
Special address 126 -Set Slave address:

If a slave with node address 126 is detected during the network scan, the *PROFIBUS Master Configuration* window will display the address in red color. It will not be possible for the user to adopt the slave to the configuration since it is not allowed to exchange data with devices having this address. The check box in the *Select* column will be grayed out.



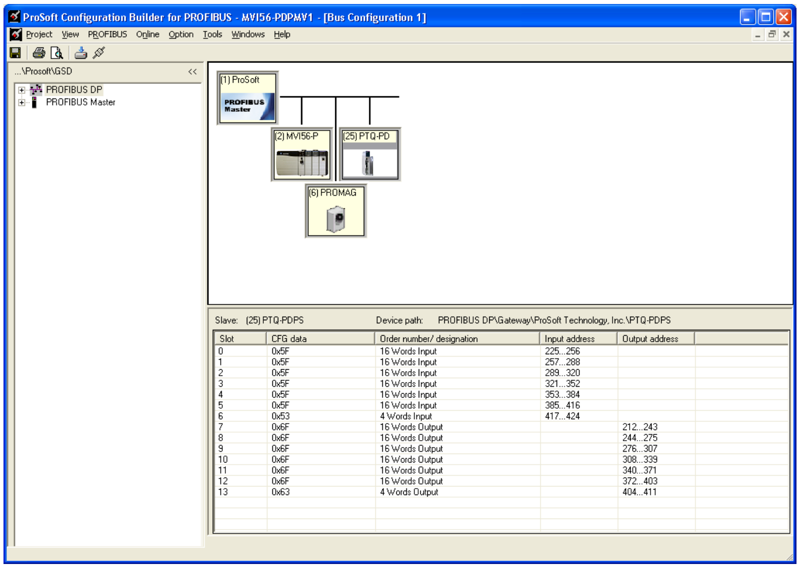
To be able to adopt a slave with address 126 the user must first assign a valid address by clicking the icon next to the node address. By doing so the *Set Slave Address* dialog box is started.

Note that the *Old slave address* is preset to a value of 126 that is not editable (grayed out).

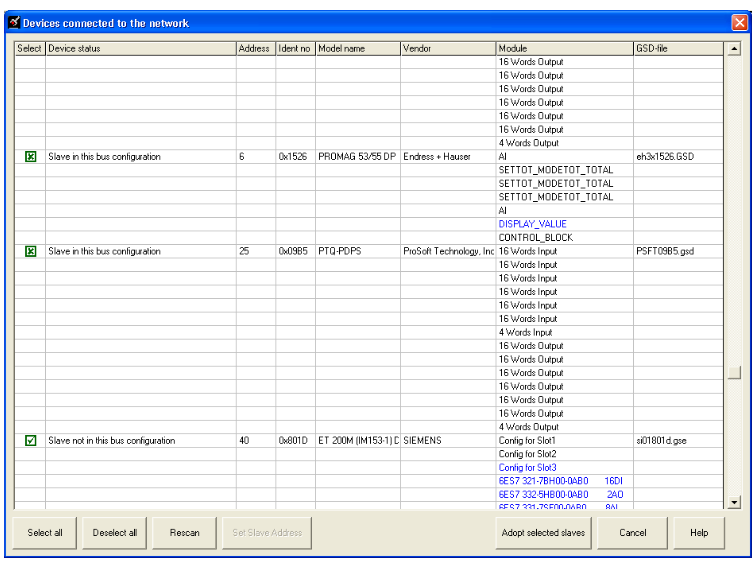


If the Slave is in the configuration already then it will not affect the addressing.

Example:



After scanning, the network finds these other slaves: 2, 6, 25, and 40
Slaves 2, 6, and 25 are found, but are marked as in the bus configuration (the mapping of the inputs and outputs will not be affected)
Slaves 40 is new and could be added and the input/output addressing will be appended to the end as shown on the last screen.



The *PROFIBUS Master Configuration* window will prevent the user from selecting a *New slave address* that is already occupied by another device; this includes detected Master stations as well. If the user selects an occupied address, a message similar to the one shown here will open.

When an address has been successfully assigned, the *PROFIBUS Master Configuration* window will update the *Network scan* window as shown here. The node address will be updated to the one that the user selected in the *Set Slave* dialog box. The check box in the *Select* column will be marked allowing the user to adopt the slave to the configuration.

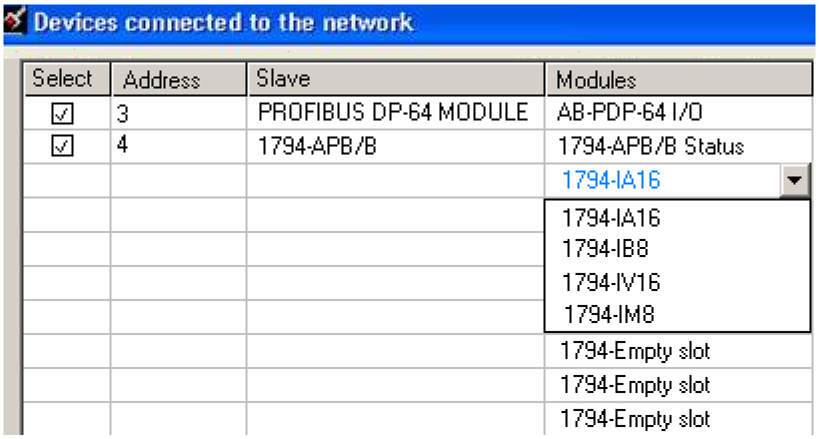
Slave

In this column the name of the slave as stated in the assigned GSD-file will be displayed. If no matching GSD-file is found the Ident number will be displayed in red color in the drop-down list.

Module

This column shows the name of the module(s) as stated in the assigned GSD-file, which matches the Identifier byte(s) derived from the *GetCfg* mailbox message. If no GSD-file or no matching module is found the Identifier byte(s) will be displayed in red color. If the configuration for a slave is constructed of several modules, the modules will be listed under each other.

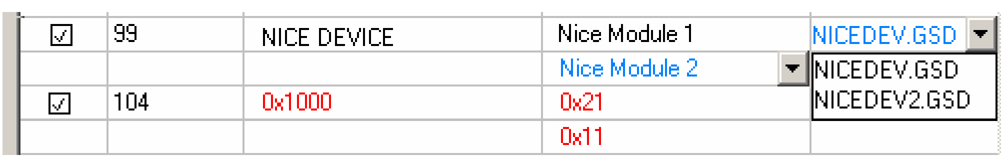
If there is more than one module in the GSD-file that matches the Identifer bytes, the first matching module will be displayed in blue color in a drop-down list. The drop-down list will contain all other matching modules so that the user can select the desired one.



> **Note**: Only modules that have the exact same Identifer bytes as the first matching module will be displayed in the drop-down list.

GSD-file

This column shows the name of the GSD-file that matches the Ident number derived from the *SlaveDiag* mailbox message. If there are more files with the same Ident number in the device catalog, the first matching GSD-file will be displayed in blue color in a drop-down list.



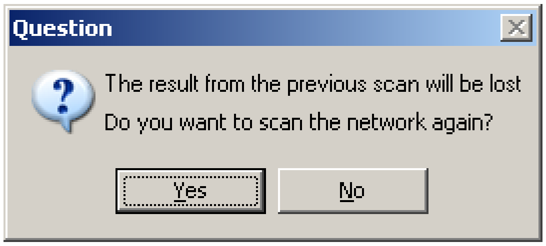
This could be the case if the device catalog contains two or more brand labeled devices, or GSD-files for two or more languages (for example NICEDEV.GSD and NICEDEV.GSE) exist.

> **Note**: If the user selects another GSD-file, The *PROFIBUS Master Configuration* window will update the modules for that slave accordingly.

If no GSD-file is found the user will be able to copy the expected GSD to the device catalog by clicking the icon next to the text *No GSD found*. This will start the Install *new GS*-file* dialog box. When the file is installed, the *PROFIBUS Master Configuration* window will verify that the installed file matches the slave and update the modules for the slave accordingly.

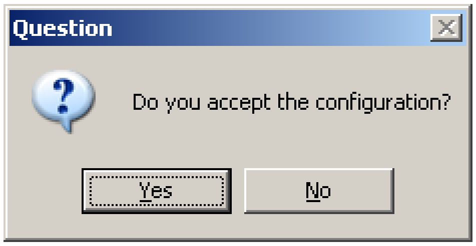
Rescan

Pressing the **YES** button will trigger a new network scan. Before proceeding with the scan a message similar to the one below will appear. If a new scan is accepted, detected slaves found during the previous scan will be lost.

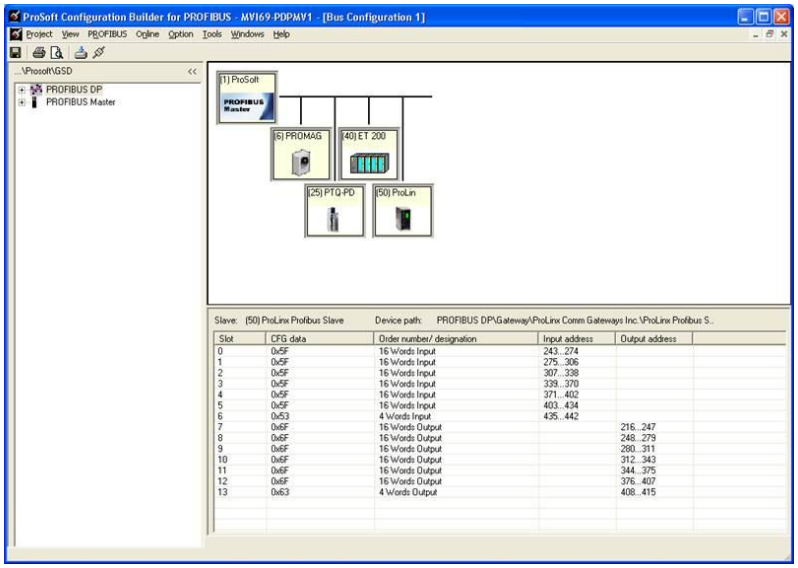


Adopt selected slaves

Pressing this button will cause all selected slaves to be adopted to the *PROFIBUS Master Configuration* window. Before carrying on with this action a message similar to the one below will appear.



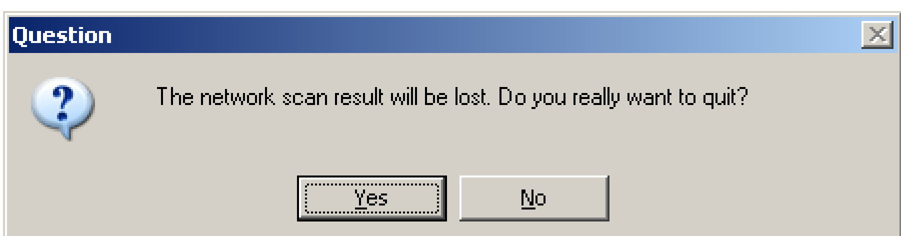
If accepted, the *network scan* window will close and the *PROFIBUS Master Configuration* window will be populated with the slaves that were found during the network scan.



Note: *Slave:* is equal to the Ident number and that the *Device path:* and *Order number/designation* fields are left empty.

Cancel and Help

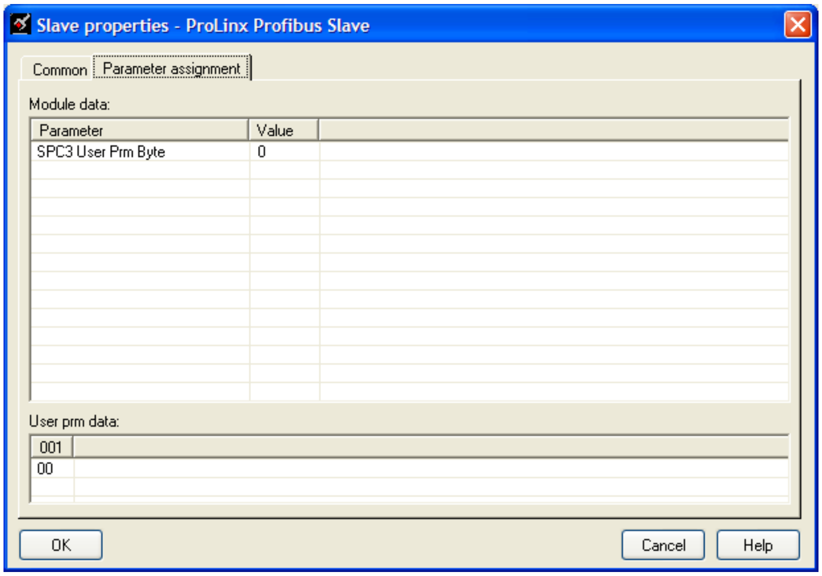
If the **CANCEL** button is pressed a message similar to the one below will appear.



If the **HELP** button is pressed the online help will start.

Set_Param (SAP61)

ProSoft PROFIBUS slave (PDPS) devices have a configurable parameter for SPC3 User Prm Byte. The following illustration shows the value of this parameter in *ProSoft Configuration Builder for PROFIBUS*, the configuration tool for ProSoft PROFIBUS Master devices.



Parameter Data Structure

SPC3 evaluates the first seven data bytes (without user prm data), or the first eight data bytes (with user prm data). The first seven bytes are specified according to the standard. The eighth byte is used for SPC3-specific communications. The additional bytes are available to the application.

Byte	Bit Position								Designation
	7	6	5	4	3	2	1	0	
0	Lock Req	Unio Req	Sync Req	Free Req	WD on	Res	Res	Res	Station status
1									WD_Fact_1
2									WD_Fact_2
3									MinTSDR
4									Ident_Number_High
5									Ident_Number_Low
6									Group_Ident
7									Spec_User_Prm_Byte
8 to 243									User_Prm_Data

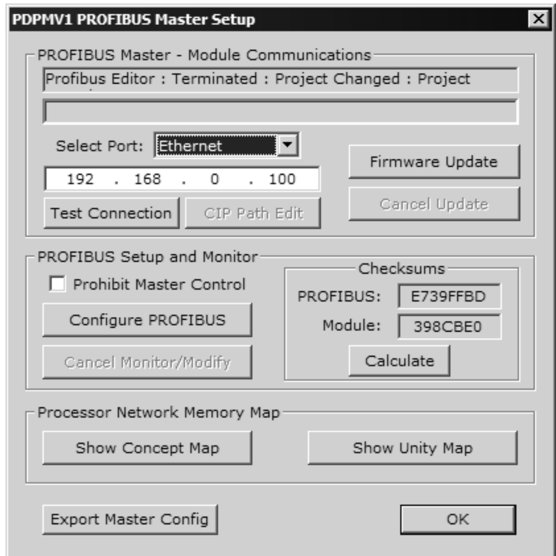
Byte 7	Spec_User_Prm_Byte		
Bit	**Name**	**Significance**	**Default State**
0	Dis_Startbit	The start bit monitoring in the receiver is switched off with this bit	Dis_Startbit = 1, Start bit monitoring is switched off.
1	Dis_Stopbit	Stop bit monitoring in the receiver is switched off with this bit	Dis_Stopbit = 0 Stop bit monitoring is not switched off.
2	WD_Base	This bit specifies the time base used to clock the watchdog. WD_Base = 0: time base 10 ms WD_Base = 1: time base 1 ms	WD_Base = 0 The time base is 10 ms.
3 to 4	Res	To be parameterized with 0	0
5	Publisher_Enable	DXB-publisher-functionality of the SPC3 is activated with this bit	Publisher_Enable = 0, DXB-request-telegrams are ignored; Publisher_Enable = 1, DXB-request-telegrams are processed
6 to 7	Res	To be parameterized with 0	0

2.3.3 Exporting the Processor Memory Map

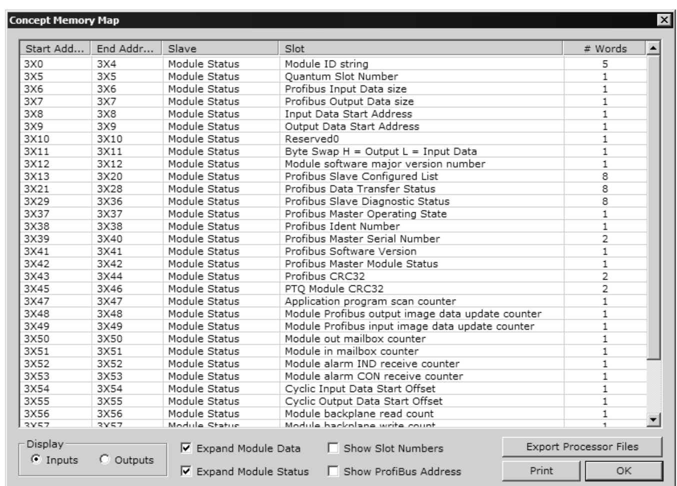
The import file (PTQ_PDPMV1.XSY for Unity, or PTQ_PDPMV1.DTY for Concept) that you create in this step uses the information in the Processor Memory Map to build the derived data tags for the slave devices on your PROFIBUS network. These tags allow the program running on the processor to access data within the module.

To export the processor memory map

1 In the *Master Setup* dialog box, click **SHOW CONCEPT MAP** (for processors configured with Concept software) or click **SHOW UNITY MAP** (for processors configured with ProSoft Configuration Builder software).



2 This action opens the *Memory Map* dialog box.



3 On the *Memory Map* dialog box, click **EXPORT PROCESSOR FILES**.

Note: For Unity Map, PCB will export the XSY file and XFM files in the same directory if mailbox parameter is chosen. The filenames will match the module name you chose in PCB.
For Concept Map, PCB will export .dty file, .txt file and .asc files if mailbox parameter is chosen.

4 Name the file and choose a location on your hard drive. The recommended location is your *My Documents* folder, and then click **SAVE.**

5 Click **PRINT** to print the input and output maps for reference.

When you import this memory map file into the processor configuration, it simplifies the task of establishing communications between the module and the processor. You will have to establish backplane communications using either Concept or Unity XL Pro software.

After you download the configuration to the PTQ-PDPMV1 module, save the .dty and .xsy files to a location on your hard drive (a folder below C:\PCBExportFiles), where you will import them into the processor during the processor configuration steps. These project files greatly reduce the amount of time it would otherwise take to perform the necessary configuration tasks.

If you are using Unity 7.X or later, you will use the *PTQ-PDPMV1.xfm* and *PTQ-PDPMV1.xsy* files.

If you are using Unity 2.X to 6.X, you will use the *PTQ-PDPMV1_V2.X-6.X.xfm* and *PTQ-PDPMV1_V2.X-6.X.xsy* files.

Refer to Configuring the Processor with Unity Pro (page 59) and Configuring the Processor with Concept (page 101) for detailed instructions on how to configure the processor.

Note: The recommended location for the files is the *My Documents* folder on your PC. The configuration tool for the processor will use this folder by default.

Calculating Checksums

The checksum (CRC) values are calculated from the PROFIBUS configuration data, and compare the contents of the configuration file in the module with the value reported by the processor. The checksum (CRC) value allows the processor to verify that the configuration file is valid, and has not changed since the last time the configuration file was imported to the processor. Any change to the contents of the configuration file in either location changes the unique numeric (CRC) value for the file.

If the checksum values do not match, the Master stops and indicates a configuration error, and the CFG light illuminates on the module.

1 On the *PTQ-PDPMV1 PROFIBUS Master Setup* dialog box, click the **CALCULATE CHECKSUMS** button.

2 Make a note of the checksum values so that you can enter them later if prompted.

3 To insert the checksum values in Unity Pro, refer to Updating Checksum Values: Unity Pro (page 67).

To insert the checksum values in Concept, refer to Configuration Validation & SETCRC Function Block (page 128).

2.3.4 Downloading the Project to the Module

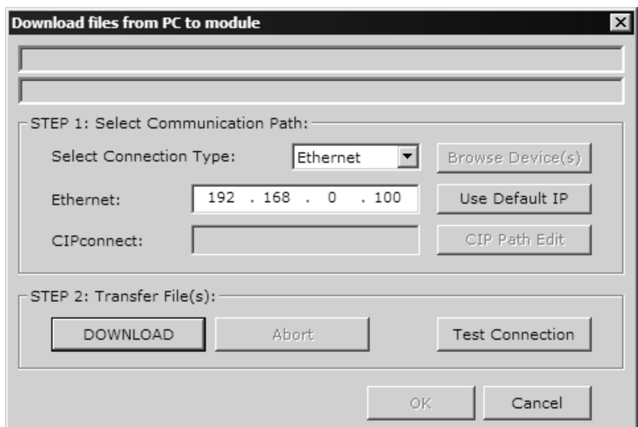
In order for the module to use the PROFIBUS network settings you configured, you must download (copy) the updated project file from your computer to the module.

Note: The processor (Quantum) must be in "Stop" mode before you download the file to the module. Use the processor's configuration tool or the softkeys on the processor to stop the processor.

To download the project file

1 In the tree view in *ProSoft Configuration Builder*, click once to select the *PTQ-PDPMV1 module*.

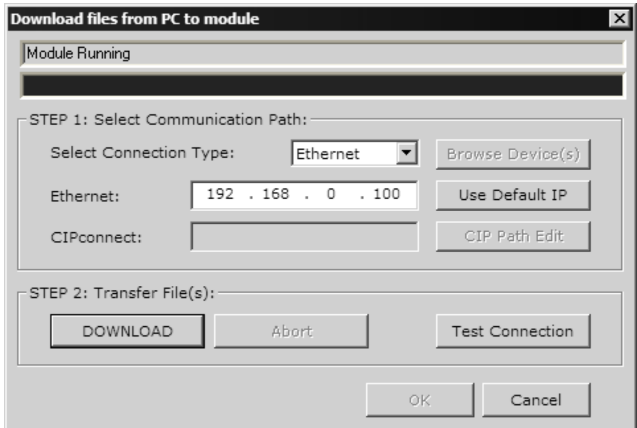
2 Open the **PROJECT** menu**,** and then choose **MODULE** > **DOWNLOAD**. This action opens the *Download Files* dialog box.



3 Choose **ETHERNET** from the dropdown list, and then click the **DOWNLOAD** button. When the download is complete, a dialog box will prompt you to place the processor back into RUN mode.

Note: If you have not yet downloaded the Ethernet Configuration (WATTCP.CFG) file, which contains the customized IP address settings for the module, you have the option on this dialog box to connect using the module's default IP address (192.168.0.100).
HSBY Note: For HSBY Ethernet downloading (Ethernet recommended), both HSBY modules must be connected to allow PCB to download to both modules. PCB will download to the first Master, and will then prompt you to download the project to the second module Master.

The module will perform a platform check to read and load its new settings. When the platform check is complete, the status bar in ProSoft Configuration Builder will be updated with the message *Module Running.*



2.3.5 Backing Up the Project

In this step, you will create a backup copy of your project and configuration files. The backup procedure saves your data for reuse on another machine, or allows you to restore your data in the event of a system failure.

To save your project and configuration files

1 In ProSoft Configuration Builder tree view, click **[+]** to expand the PTQ-PDPMV1 tree, and then double-click the **PROFIBUS DP** icon. This action opens the *PROFIBUS Master Setup* dialog box.

2 In the *PROFIBUS Master Setup* dialog box, click the **EXPORT MASTER CONFIG** button. This action saves the PROFIBUS network configuration for your module in an XML file. The recommended location for this file is your *My Documents* folder.

Tip: You can use the XML file created by ProSoft Configuration Builder in this step to simplify the task of configuring additional PROFIBUS network modules. Because its saves the entire network configuration, you can add modules quickly by modifying only the items that are unique for each device, typically the slot number and I/O addresses. To use this saved configuration, open Windows Explorer, navigate to the folder where you saved the Master Configuration XML file, and then drag the file onto the new PROFIBUS DP icon in the ProSoft Configuration Builder tree view.

3 **Unity Pro Users:** From the PROFIBUS Master Setup Screen, click the **SHOW UNITY MAP** button, then click the **EXPORT PROCESSOR FILES** button. This action exports the xfm file (created only if the *Mailbox* parameter is set to **YES**) and xsy file. The recommended location for these files is your *My Documents* folder.
If you are using Unity 7.X or later, you will use the *PTQ-PDPMV1.xfm* and *PTQ-PDPMV1.xsy* files.
If you are using Unity 2.X to 6.X, you will use the *PTQ-PDPMV1_V2.X-6.X.xfm* and *PTQ-PDPMV1_V2.X-6.X.xsy* files.

Concept Users: From the PROFIBUS Master Setup Screen, click the **SHOW CONCEPT MAP** button, and then click **EXPORT PROCESSOR FILES**. This action exports the DTY and related files. The recommended location for these files is your *My Documents* folder.

4 From the *PROFIBUS Master Setup* dialog box, click the **SHOW CONCEPT MAP** button. Then choose **EXPORT PROCESSOR FILES** to export the DTY, TXT and other related files if the *Mailbox* parameter is set to **YES**. The recommended location is your *My Documents* folder.
5 Click **OK** to close the *PROFIBUS Master Setup* dialog box.
6 In the ProSoft Configuration Builder, open the **FILE** menu, and then choose **SAVE AS**.
7 Name the project file, and click **SAVE**. The recommended location for this file is your *My Documents* folder.

Note: All PCB project files and module-related files are automatically saved to C:\PCBExportFiles.

A complete backup consists of the project and Master configuration files, plus the GSD configuration files. The default location for the GSD files is *C:\Documents and Settings\All Users\Application Data\ProSoft\GSD (Windows XP / 2000)* or *C:\My Documents*\. To move a project to a different PC, copy the .PPF, .XML, and .GSD files to the same directory structure on the new machine that they occupied on the old one.

The above method defines a manual approach in creating Quantum processor I/O and Function Block import files. The PCB will also automatically create these files when the PCB project is saved or closed (if the project is not saved then PCB will not export the files).

You can also generate these files manually from PCB. To create the files:

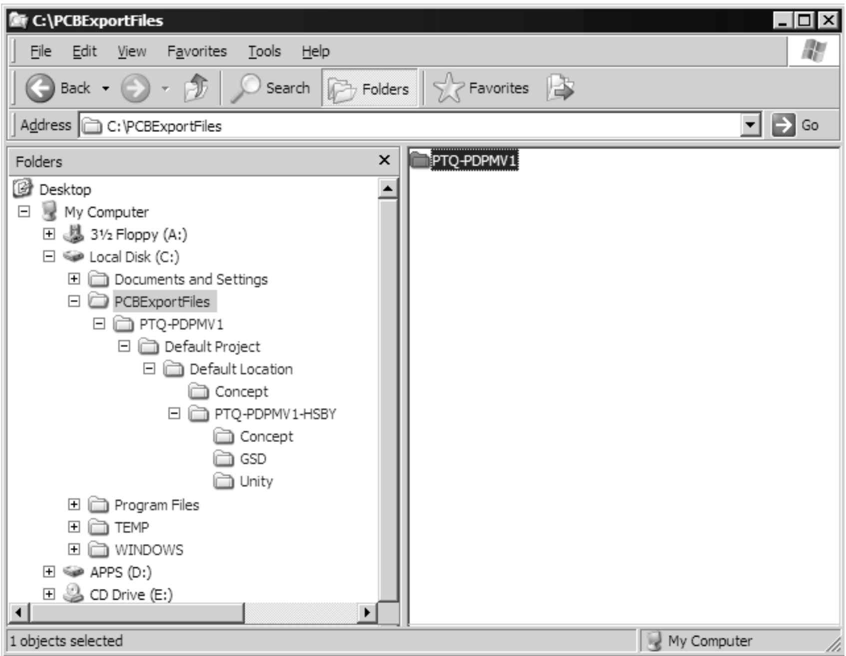
1 Open the **PROJECT** menu, and select **PROJECT** > **EXPORT FILES**.
2 If you are prompted to overwrite files, click **YES**.

2.3.6 File Locations

The folder structure implemented for each PCB project (ppf) is as follows:

{rootdrive}\PCBExportFiles\'ppf name'\'*Project Name'\'Location Name'\'Module Name'*

For example,



The following files will be created in each folder created by PCB

- *{rootdrive}\PCBExportFiles\'ppf name'*
  - Project ppf file (.ppf)
- *{rootdrive}\PCBExportFiles\'ppf name'\'Project Name'\'Location Name'*
  - (Concept folder created only for PTQ-PDPMV1 modules)
  - \Concept\.dty, .asc files
- *{rootdrive}\PCBExportFiles\'ppf name'\'Project Name'\'Location Name'\'Module Name'*
  - PROFIBUS xml file (modulename{ModuleName}.xml) PTQ cfg file (.cfg)
  - (Unity folder created only for PTQ-PDPMV1 modules)
  - \Unity\Unity xml files (.xsy, .xfm)
- (gsd folder created for all PDPMV1 modules)
- \gsd\GSD files used for module (.gsd)
- (Concept folder created only for PTQ-PDPMV1 modules)
- \Concept\txt files for variables

If you have followed the previous steps in order, your PTQ module is now configured with the settings for your PROFIBUS Master and slaves. The final task is to import this information into the processor. This task allows the processor to communicate with the PTQ module and its slave devices over the backplane. The following topics will describe the different procedures for Unity and Concept platforms.

IMPORTANT NOTE: The following steps are required in order to get the system up and running.

1 Download the configuration to the module from PCB
2 Export files (XFM and XSY) from PCB
3 Import the .XFM file that was exported in Step 2
4 Import the .XSY file that was exported in Step 2

3 Configuring the Processor with Unity Pro

In This Chapter

3.1 Importing the Functional Module

To simplify the task of programming the processor when communicating with the PTQ-PDPMV1 module, ProSoft Technology has created a Unity Pro Functional Module type (XFM).

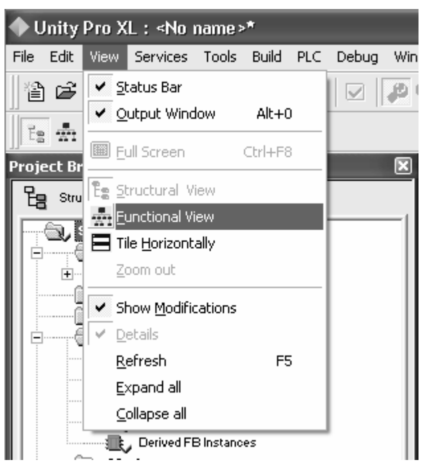
Warning: The Functional Module is intended for new installations of PTQ-PDPMV1. If you have an existing installation, the following procedure will overwrite your settings, and may cause loss of functionality. DO NOT overwrite a working application until you have thoroughly reviewed the following topics.

The Functional Module provides easy access to PROFIBUS slaves' cyclic data and the PTQ module's input/output status data. Specific mailbox commands are provided to perform DPV0/V1 acyclic functions such as *Get Live List*, *Get Slave Diagnostics*, and perform *Freeze* and *Sync* commands. The Functional Module exchange file name matches the module name you defined in PCB, with the extension .XFM. This file is created by PCB when you export the processor file from the Show Unity Map dialog box (page 54).

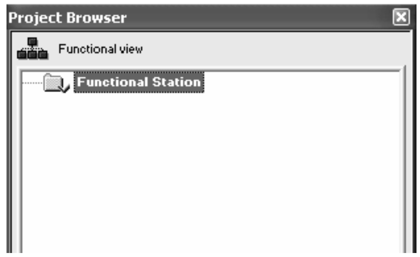
To import the Functional Module

Use the project you created in Unity Pro, and perform all of the following steps.

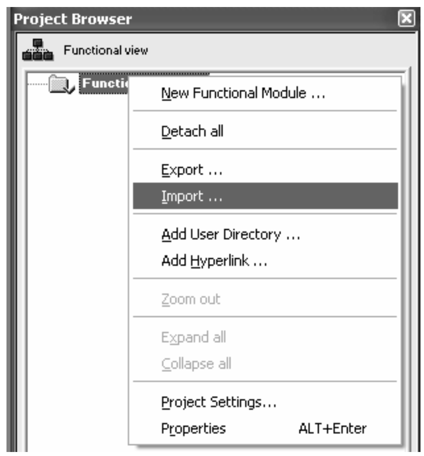
1 Open the **VIEW** menu, and then choose **FUNCTIONAL VIEW**.



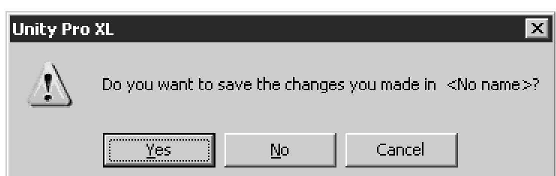
This action populates the *Project Browser* with a *Functional Station* icon, as shown in the following illustration.



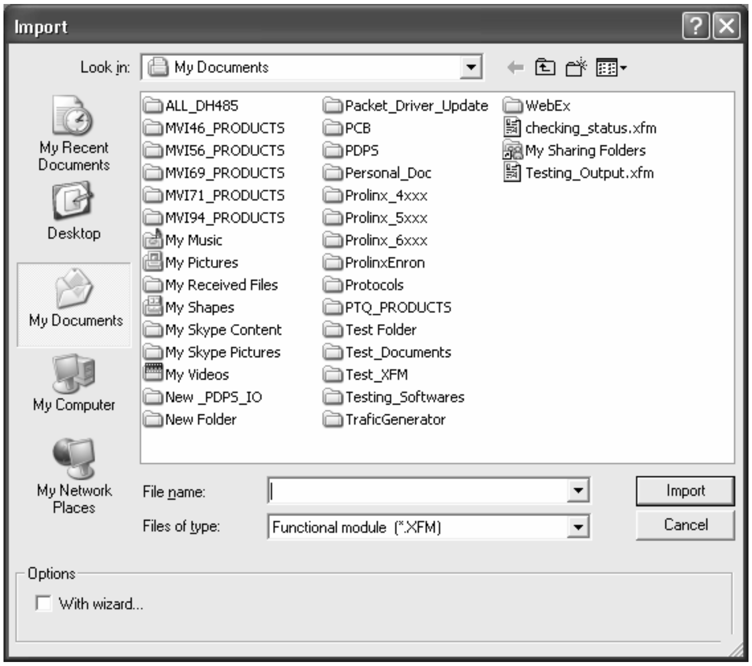
2 Select **FUNCTIONAL STATION**, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT.**



Click **NO** to dismiss the confirmation dialog box.



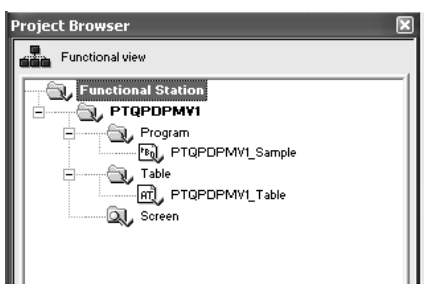
In the *Import* dialog box, choose **FUNCTIONAL MODULE** (*.XFM) in the Files of Type dropdown list, and then select the **XFM** file to import. The XFM file name matches the module name you defined in PCB and exported in step 3 of Back up the Project (page 54).



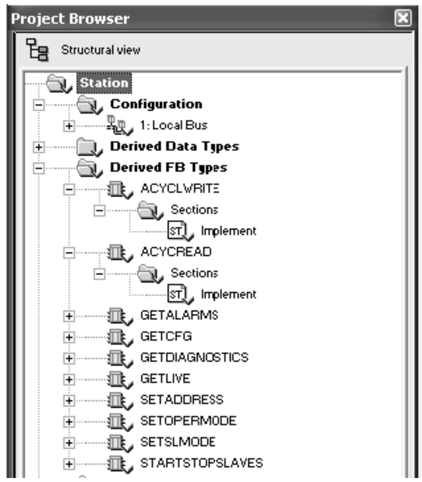
Click **IMPORT** to import the file.

Note: Use the XFM file created by PCB. The XFM file created by PCB is preferred, because it contains the I/O map representing your PROFIBUS network and contains the same variable names. This file will be created only if the *Mailbox* messaging parameter is set to **YES**.

Notice that the *Project Browser* is now populated with the *Functional Module*.



3 To view the *DFBs*, data types and variables associated with the *Functional Module*, open the **VIEW** menu and choose **STRUCTURAL VIEW**. Notice that all function blocks have been defined using the ST type language.



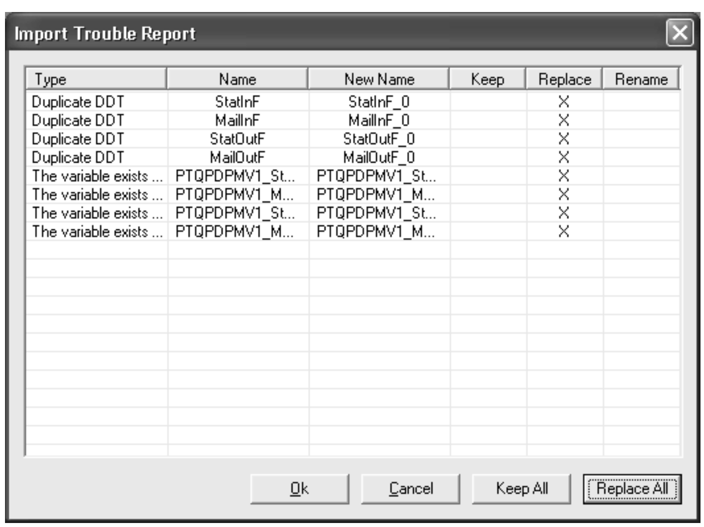
To import the variables

Import the PROFIBUS I/O table, found in the .xsy file which was created when the memory map was exported from ProSoft Configuration Builder (PCB) (see Backing Up the Project (page 54)). This file contains all the cyclic input and output variables configured by the PCB Master configuration software. It includes module status data, and may also include slave diagnostic data and mailbox data if these parameters were chosen.

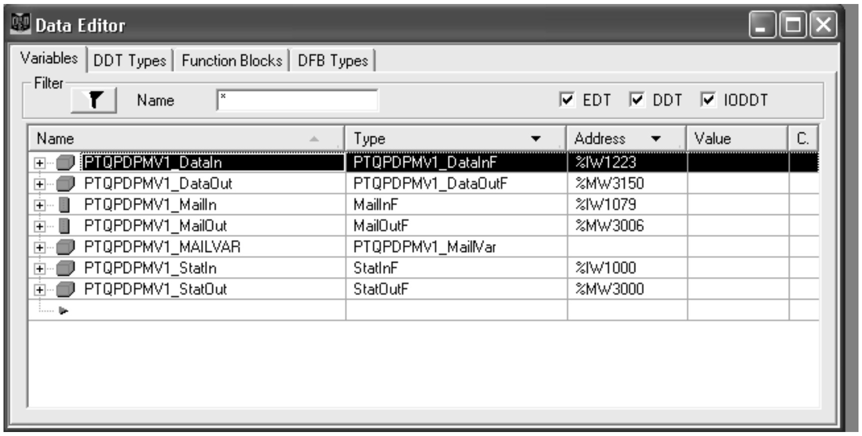
1 In the *Project Browser*, select **VARIABLES & FB INSTANCES**, and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT**.

2 In the **FILES OF TYPE** dropdown list, choose **DATA EXCHANGE FILE** (*.XSY). Select the **.XSY** file created in *Backing Up the Project*, and then click **IMPORT**.

3 In the *Import Trouble Report* window, click **REPLACE ALL**, then click **OK**.



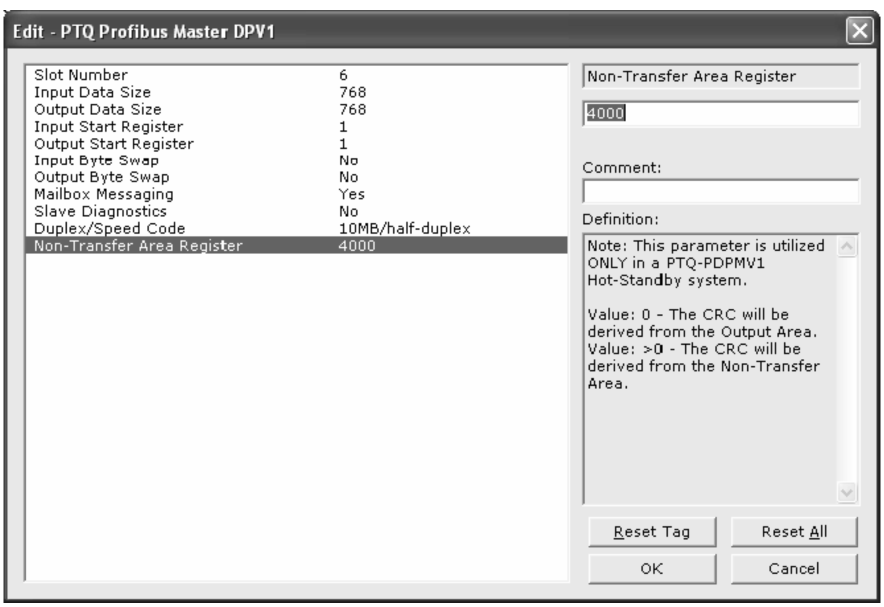
At this point, the input and output variables have been imported to the application.



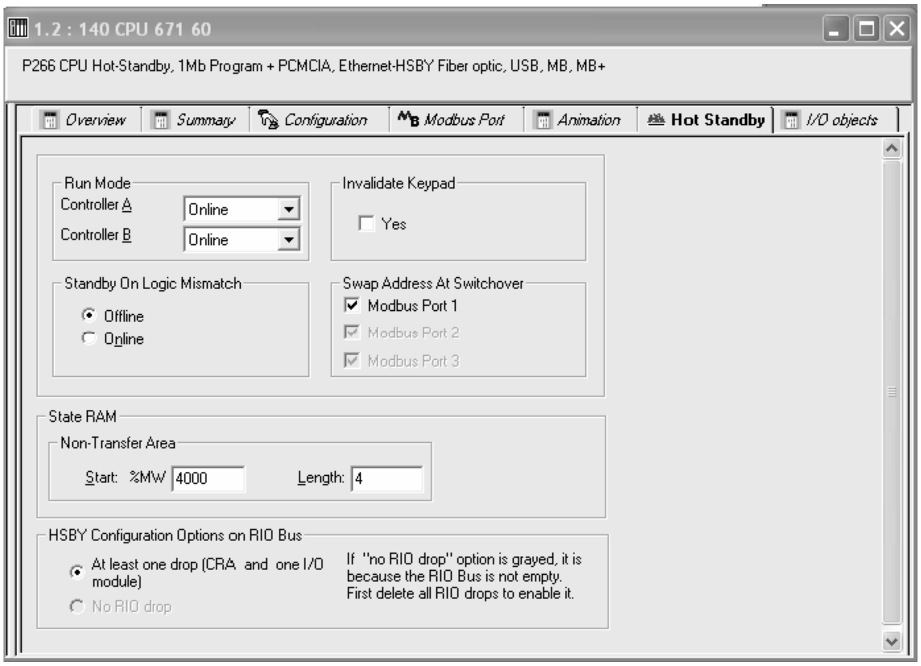
HSBY Note: If the *Non-Transfer* parameter value is used and it is greater than zero, then the XSY file will contain the correct CRC for the module.

The value you entered in ProSoft Configuration Builder for the *Non-Transfer* parameter should also be entered in the non-transfer area of the processor.

For example, if you entered *Non-Transfer Area Register* = 4000 in ProSoft Configuration Builder:



You must enter the same value in this location of the processor memory.

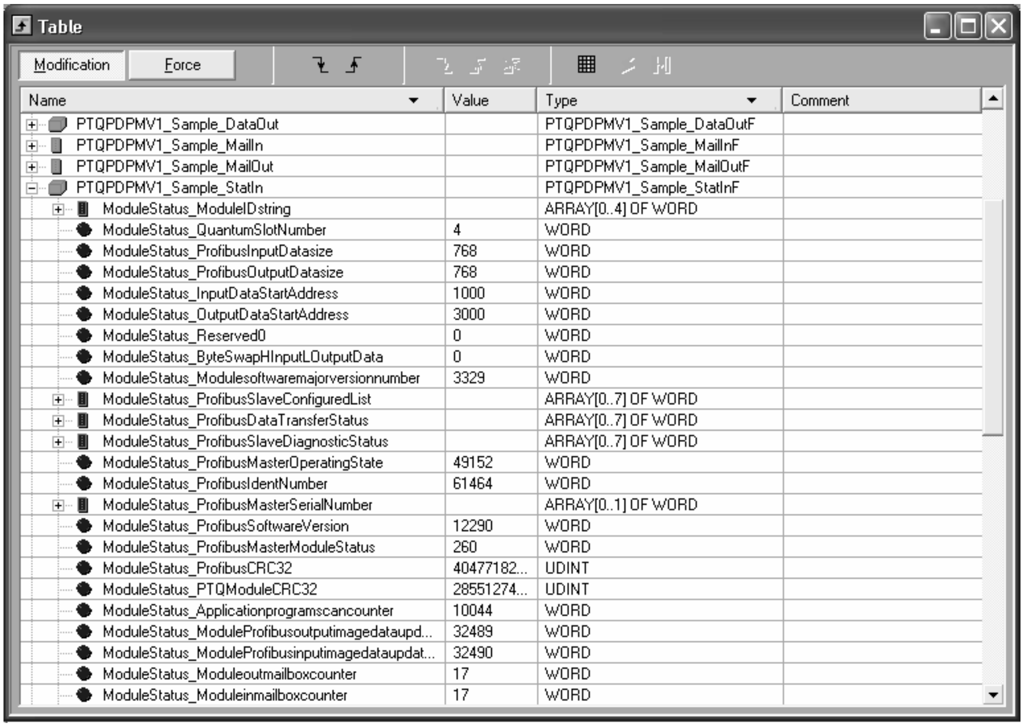


<u>To modify an animation table</u>

Note: An animation table is required to send and receive mailbox messages, monitor State Ram status and read/write IO data.

An animation table is provided with the XFM file, but certain data variables must be added to monitor the status or health of the module.

Double-click the animation table, and under the name column, select the **{MODULENAME}_STATIN** variables. Under the <inputs> folder, select the **MODIFICATION TAB**. You should see the module status counters update.



Table

Modification | Force

Name	Value	Type	Comment
PTQPDPMV1_Sample_DataOut		PTQPDPMV1_Sample_DataOutF	
PTQPDPMV1_Sample_MailIn		PTQPDPMV1_Sample_MailInF	
PTQPDPMV1_Sample_MailOut		PTQPDPMV1_Sample_MailOutF	
PTQPDPMV1_Sample_StatIn		PTQPDPMV1_Sample_StatInF	
ModuleStatus_ModuleIDstring		ARRAY[0..4] OF WORD	
ModuleStatus_QuantumSlotNumber	4	WORD	
ModuleStatus_ProfibusInputDatasize	768	WORD	
ModuleStatus_ProfibusOutputDatasize	768	WORD	
ModuleStatus_InputDataStartAddress	1000	WORD	
ModuleStatus_OutputDataStartAddress	3000	WORD	
ModuleStatus_Reserved0	0	WORD	
ModuleStatus_ByteSwapHInputLOutputData	0	WORD	
ModuleStatus_Modulesoftwaremajorversionnumber	3329	WORD	
ModuleStatus_ProfibusSlaveConfiguredList		ARRAY[0..7] OF WORD	
ModuleStatus_ProfibusDataTransferStatus		ARRAY[0..7] OF WORD	
ModuleStatus_ProfibusSlaveDiagnosticStatus		ARRAY[0..7] OF WORD	
ModuleStatus_ProfibusMasterOperatingState	49152	WORD	
ModuleStatus_ProfibusIdentNumber	61464	WORD	
ModuleStatus_ProfibusMasterSerialNumber		ARRAY[0..1] OF WORD	
ModuleStatus_ProfibusSoftwareVersion	12290	WORD	
ModuleStatus_ProfibusMasterModuleStatus	260	WORD	
ModuleStatus_ProfibusCRC32	40477182...	UDINT	
ModuleStatus_PTQModuleCRC32	28551274...	UDINT	
ModuleStatus_Applicationprogramscancounter	10044	WORD	
ModuleStatus_ModuleProfibusoutputimagedataupd...	32489	WORD	
ModuleStatus_ModuleProfibusinputimagedataupdat...	32490	WORD	
ModuleStatus_Moduleoutmailboxcounter	17	WORD	
ModuleStatus_Moduleinmailboxcounter	17	WORD	

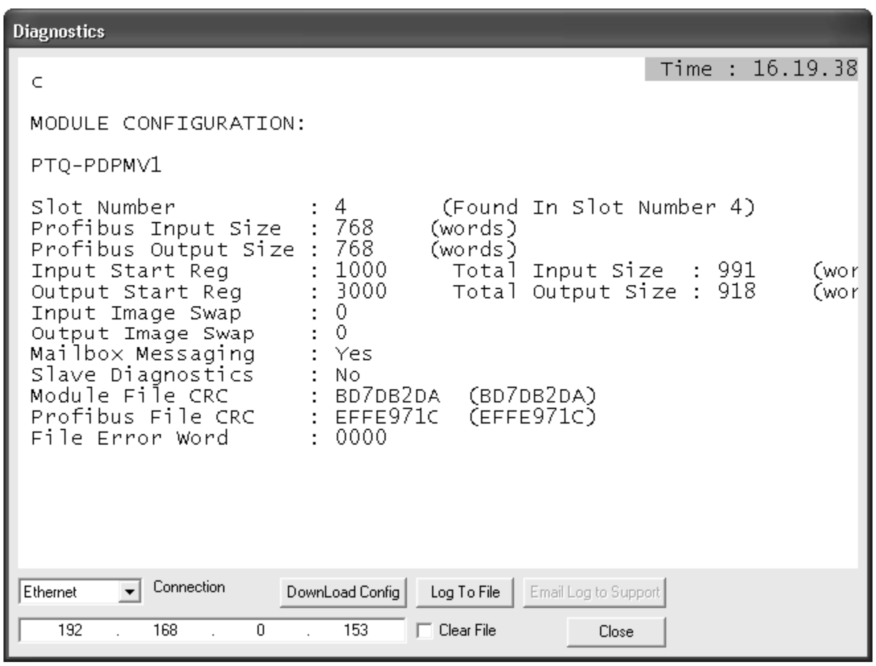
3.1.1 Updating Checksum Values

The PTQ-PDPMV1 module is almost ready, and the CRC values for the PROFIBUS configuration should match between the module and the processor.

<u>To confirm that both CRCs match</u>

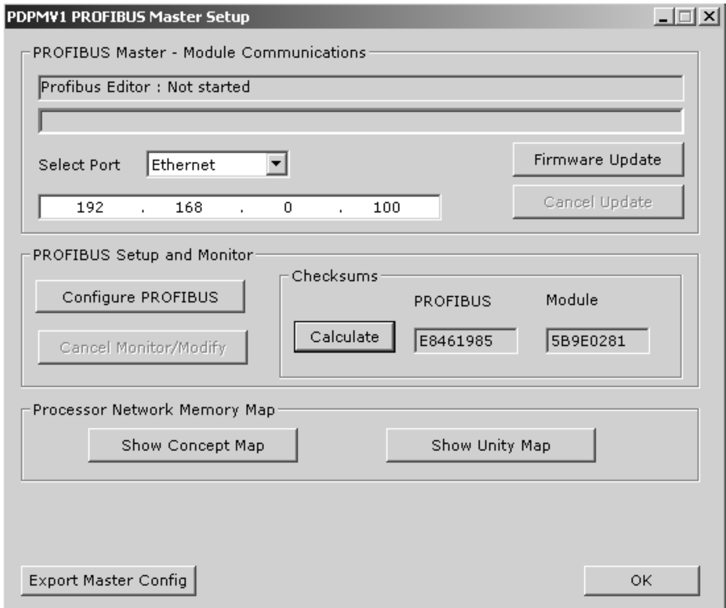
1. From PCB, select the **MODULE** icon, and then click the right mouse button to open a shortcut menu.
2. On the shortcut menu, choose **DIAGNOSTICS**. Wait for ProSoft Configuration Builder to go online with the module through the serial or Ethernet port.
3. When the module is online, press **[?]** to display the *Main* menu.
4. On the *Main* menu, press **[C]** to view the module configuration. The following illustration shows example CRC values for the *Module File* and the *PROFIBUS File.*

> **Note:** Because the CRC values are calculated for your unique configuration, the values on your screen will not be the same as the ones in the following illustration.



To calculate checksums

1 On the *PDPMV1 PROFIBUS Master Setup* dialog box, click the **CALCULATE CHECKSUMS** button.



Notice the NEW checksums for the module and PROFIBUS appear.

Note: The module checksum will change when parameters such as 3X or 4X starting address are changed. The PROFIBUS checksum will change if a network parameter is changed.
HSBY Note: For Hot Standby application, if you use the *Non-Transfer* parameter, the module expects the CRC to be taken from the offset you provided, which must match the offset you entered in the processor program.

The CRC value is provided in the XSY file exported by ProSoft Configuration Builder, as shown in the following illustration.

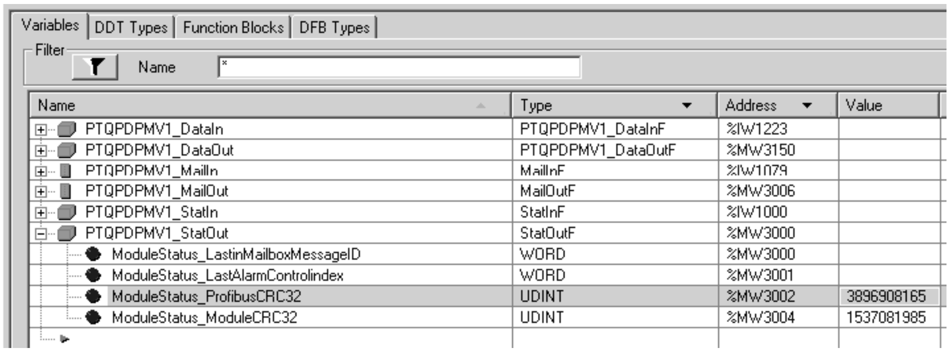
PTQPDPMV1HSBY_NONTRAN	ARRAY[0..1] OF UDINT	%MW4000	
PTQPDPMV1HSBY_NONTRAN[0]	UDINT	%MW4000	647746906
PTQPDPMV1HSBY_NONTRAN[1]	UDINT	%MW4002	1151774640

If the *Non-Transfer* parameter is set to **0** (not used), the CRC is provided in the *PTQPDPMV1HSBY_StatOut* area, as shown in the following illustration.

PTQPDPMV1HSBY_StatOut	StatOutF	%MW1	
ModuleStatus_LastinMailboxMessageID	WORD	%MW1	
ModuleStatus_LastAlarmControlindex	WORD	%MW2	
ModuleStatus_ProfibusCRC32	UDINT	%MW3	647746906
ModuleStatus_ModuleCRC32	UDINT	%MW5	1151774640

2 If the CRC values do not match, copy the PROFIBUS checksum by highlighting the text and right-clicking to **COPY**.

3 Go to the **UNITY PRO VARIABLES TAB** and select the *_StatOut* variable and expand the structure to expose the *ModuleStatus_ProfibusCRC32* element. Under the *Value* column area, paste the copied checksum.

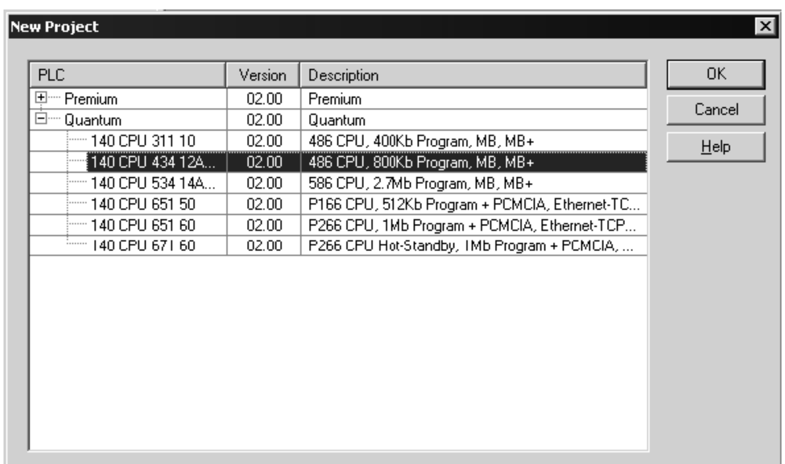


4 Repeat steps 2 and 3 above for the module checksum value. **PASTE** the value into *ModuleStatus_ModuleCRC32* variable.
5 Download the new values in the program to the processor.

3.1.2 Setting Up General Unity Pro Project Settings

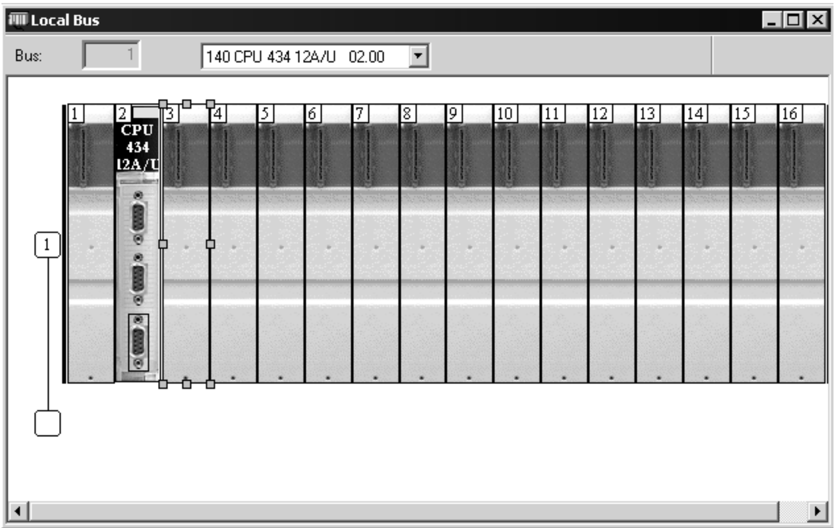
<u>To set up general Unity project settings</u>

1 Start *Unity Pro*. Open the **FILE** menu, and then select **NEW**. This action opens the *New Project* dialog box.

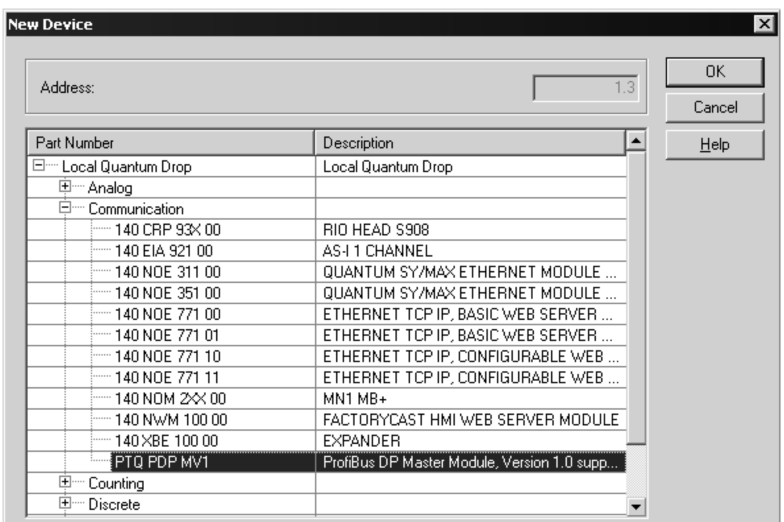


2 The *New Project* dialog box shows a list of processors that it can configure. Choose the processor you are configuring from the list, and then click **OK** to open the *Project Browser*.

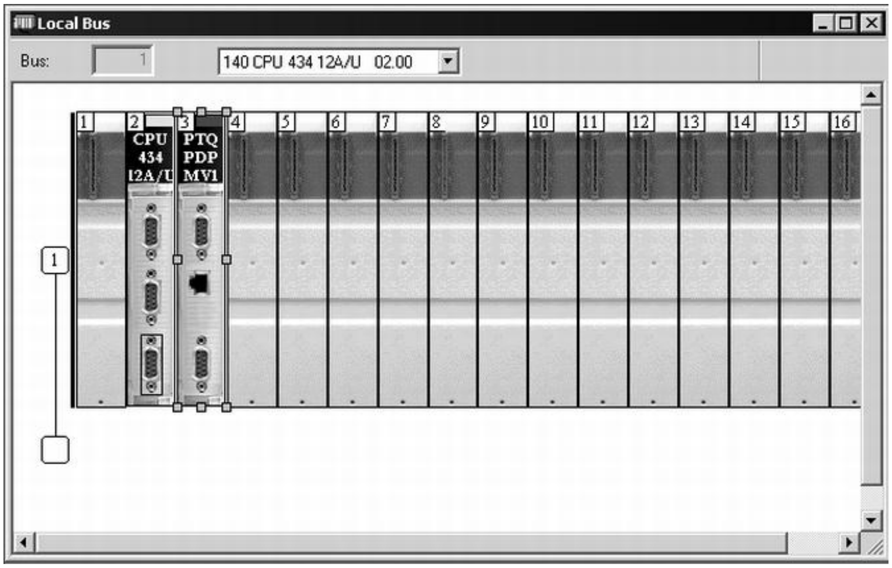
3 In the *Project Browser* tree view, double-click **LOCAL BUS** to open the *Local Bus* window. Notice that the image in the window shows the processor in the second position in the rack. (The first position is for the power supply, which you will add later. In the following steps, you will add an image of the PTQ module to the rack, in the same position where you physically installed the module.)



4 To add devices to the rack, double-click the *location* in the rack where the device is installed. This action opens the *New Device* dialog box.



5 Click the **[+]** sign next to *Communication* to open the list of communication devices. Select **PTQ PDPMV1** from the list, and then click **OK**. This action adds the module to the *Local Bus* image.



6 Repeat the previous two steps to add other devices, such as power supplies, to the rack.

7 When you have finished adding devices, open the **FILE** menu and choose **SAVE**. This action saves the project to the hard drive on your computer.

3.1.3 Configuring the Memory Size for the Processor

Part of the processor configuration process allocates memory to use in the processor to store input and output data from the module. For installations where the processor communicates with only one module, the default memory settings will work without further configuration. The following steps will help you determine the correct memory addresses to assign for more complex installations.

The processor memory maps that you configured in ProSoft Configuration Builder are exported from ProSoft Configuration Builder, and imported into the Unity Pro project. These values are calculated from the starting memory address in the processor's State RAM for the module's input and output data images. Refer to Configuring the Module (page 21) for more information on configuring memory addresses in ProSoft Configuration Builder.

Depending on the complexity of your installation, for example when you are deploying the PTQ-PDPMV1 module in an existing system, you should view the memory configuration for the processor in ProSoft Configuration Builder before you begin to configure memory addresses in Unity Pro.

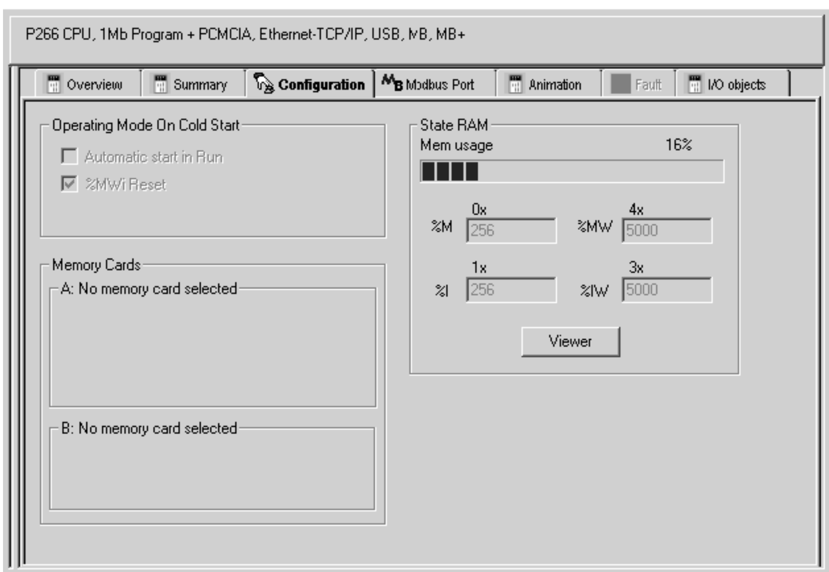
Some points to keep in mind are:

- As the programmer, you must be aware of the memory spaces that are available when deploying in an existing system, and assign values to the PTQ accordingly.

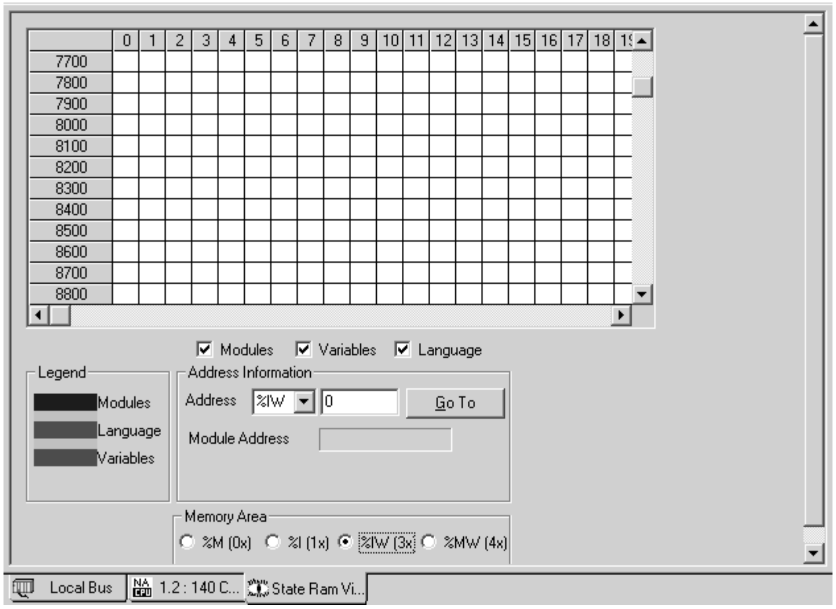
- Data registers must exceed starting registers. This is in the memory map page that you printed.
- You must assign the PTQ module to a block of processor memory that is not being used by any other device.
- You can use this simple formula to find a block of memory to use: If the module consumes 224 words of status data on input, and we know that it can take up to 768 words of I/O data, the total requirement is 992 words. The module will take a maximum of this %IW value. For convenience, round the number up to 1000 as the amount of memory to assign. A value of 5000 for %MW and %IW is a safe starting point.
- It is not possible to determine if the memory values are correct before building the project. If the build throws an error about memory addresses, go back to ProSoft Configuration Builder and change the input and output properties for the module, then re-import the memory map and try again.

To view memory usage in the processor

1 Start *Unity Pro.*
2 In the *Project Browser*, expand the *Configuration* tree, and then double-click the **LOCAL BUS** object.
3 In the **LOCAL BUS** window, double-click the processor. This action opens a tabbed window with information about the processor.
4 Click the **CONFIGURATION** tab. This tab describes the processor's memory configuration.



5 To view detailed information about the processor's memory configuration, click **VIEWER**. The viewer offers tools to view the types of data stored at specific addresses in the processor. Make note of memory areas that are already allocated, and select an area of contiguous memory that be allocated to the PTQ module.



3.1.4 Building the Project

Whenever you update the PTQ module's configuration, the PROFIBUS network, or the processor, you must import the changed configuration from the module, and then build (compile) the project before downloading it to the processor.

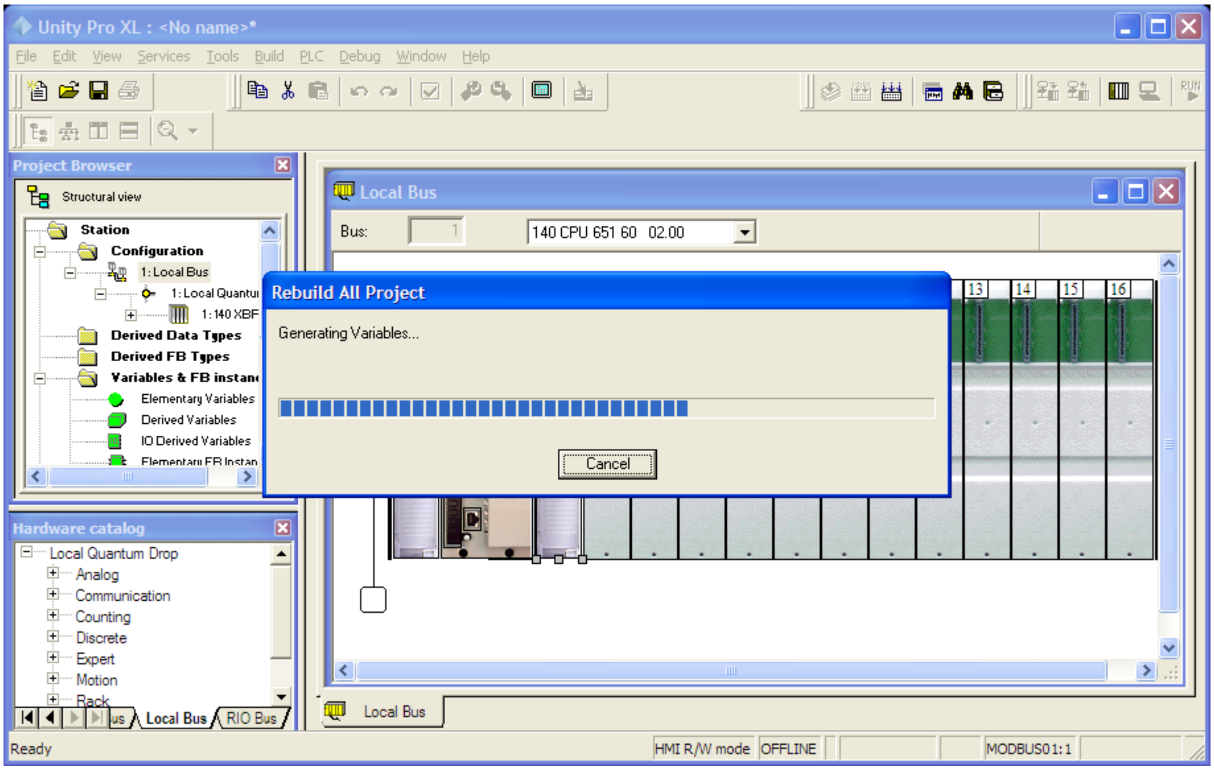
Note: The following steps show you how to build the project in Unity Pro. This is not intended to provide detailed information on using Unity Pro, or debugging your programs. Refer to the documentation for your processor and for Unity Pro for specialized information.

To build (compile) the project

1 Review the elements of the project in the *Project Browser*.

2 When you are satisfied that you are ready to download the project, open the **BUILD** menu, and then choose **REBUILD ALL PROJECT**. This action builds (compiles) the project into a form that the processor can use to execute the instructions in the project file. This task may take several minutes, depending on the complexity of the project and the resources available on your computer.

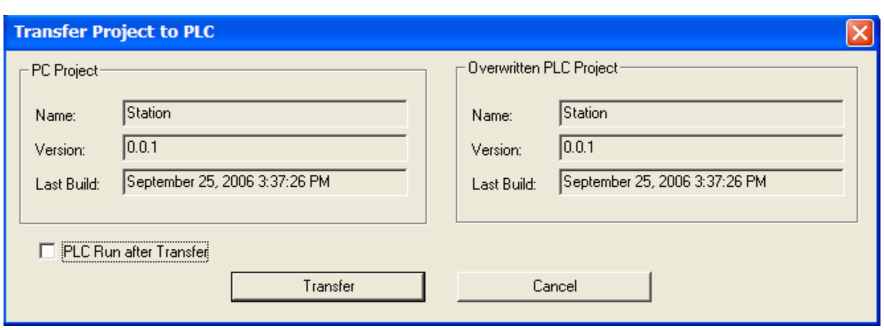
3 As the project is built, Unity Pro reports its process in a *Progress* dialog box, with details appearing in a pane at the bottom of the window. If you are using the sample project, the project should build without errors. The following illustration shows the build process under way.



3.1.5 Downloading the Project to the Quantum Processor

1 Open the **PLC** menu and then choose **CONNECT**. This action opens a connection between the Unity Pro software and the processor, using the address and media type settings you configured in the previous step.

2 On the **PLC** menu, choose **TRANSFER PROJECT TO PLC**. This action opens the **TRANSFER PROJECT TO PLC** dialog box. If you would like the PLC to go to RUN mode immediately after the transfer is complete, select (check) the **PLC RUN AFTER TRANSFER** check box.



3 Click the **TRANSFER** button to download the project to the processor. As the project is transferred, Unity Pro reports its process in a **PROGRESS** dialog box, with details appearing in a pane at the bottom of the window.

When the transfer is complete, place the processor in RUN mode. The processor will start scanning your process logic application.

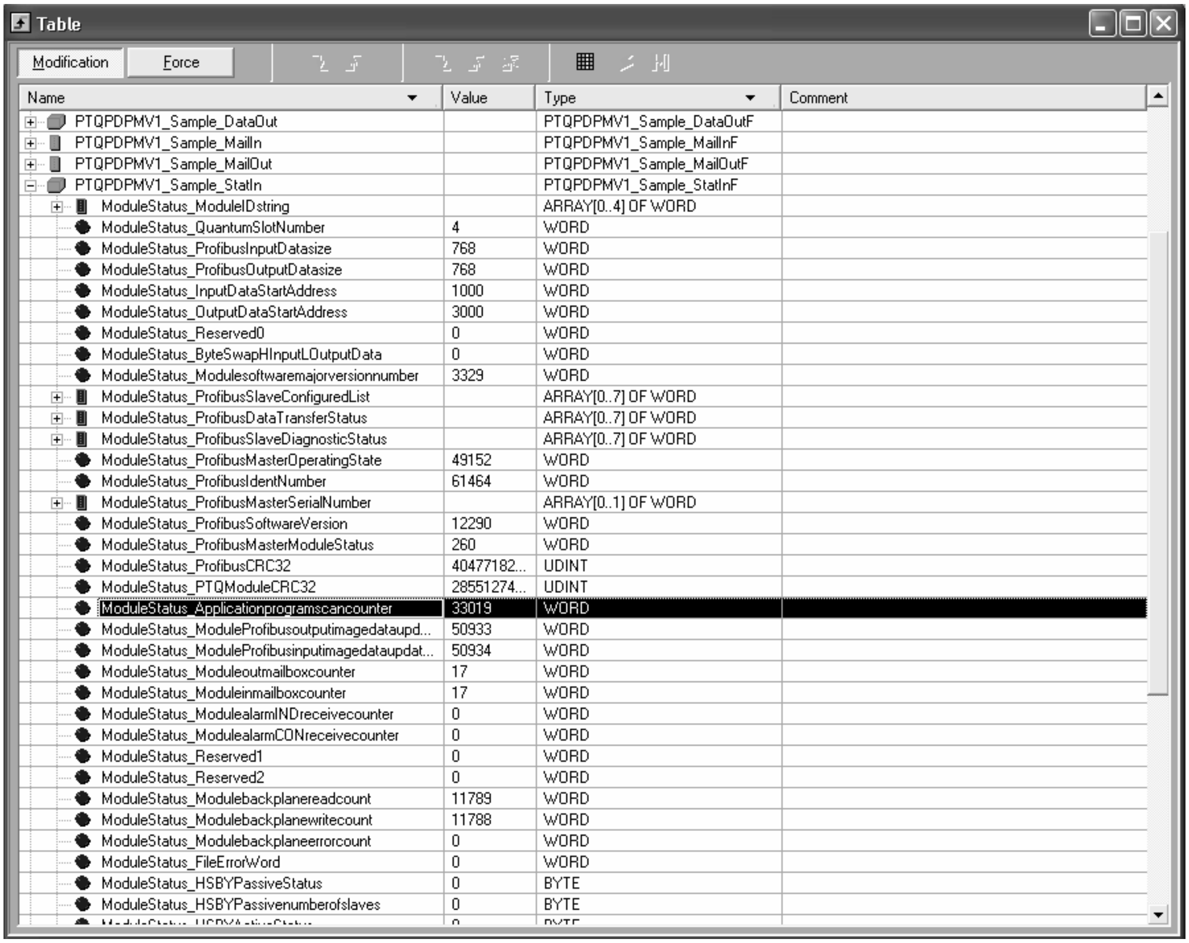
3.1.6 Verifying Communication between the Processor and the Module

In this step, you will verify that the processor and the PTQ module are communicating with each other over the backplane. The sample project includes an animation table called MailBox Commands. When the processor and the PTQ module are communicating, the values in this animation table are updated in real time.

To verify communication between the processor and the module

1 Place the processor in RUN mode, if you have not already done so.
2 In the Unity Pro project browser pane, click **[+]** to open the *Animation Tables* tree, and then double-click **MAIN TABLE**.
3 In the *Main:Table*, you will see all mailboxes, including *Get Live List, Get Diagnostics*, and so on.
4 You must include *{ModuleName}_StatIn, {ModuleName}_MailIn* and *{ModuleName}_DataIn,* using the same procedure for the Output *{ModuleName}_StatOut, {ModuleName}_MailOut and {ModuleName}_DataOut.*

5 Scroll within **{MODULENAME}_STATIN**. Notice that when the processor and the PTQ module are communicating successfully, the numbers in the Value column for items such as *ModuleStatus_Applicationprogramscancounter* are continuously updated.



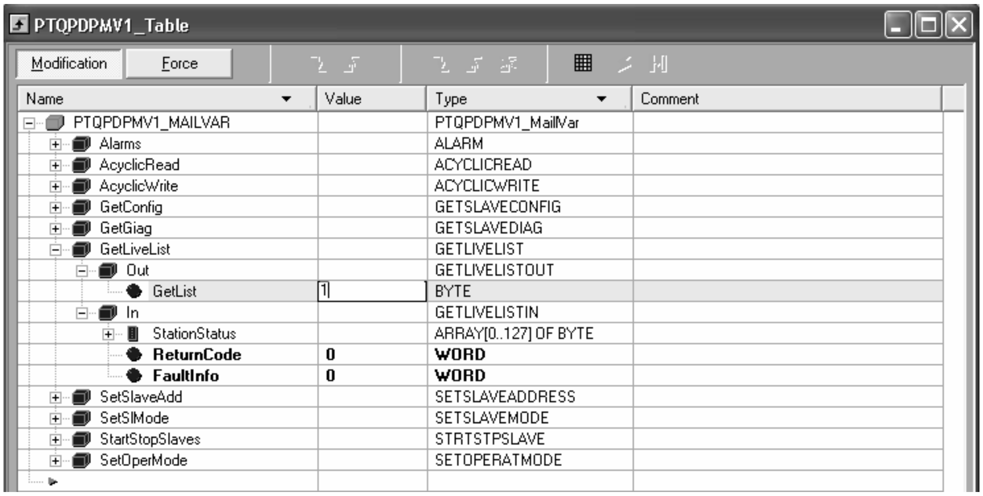
Name	Value	Type	Comment
PTQPDPMV1_Sample_DataOut		PTQPDPMV1_Sample_DataOutF	
PTQPDPMV1_Sample_MailIn		PTQPDPMV1_Sample_MailInF	
PTQPDPMV1_Sample_MailOut		PTQPDPMV1_Sample_MailOutF	
PTQPDPMV1_Sample_StatIn		PTQPDPMV1_Sample_StatInF	
ModuleStatus_ModuleIDstring		ARRAY[0..4] OF WORD	
ModuleStatus_QuantumSlotNumber	4	WORD	
ModuleStatus_ProfibusInputDatasize	768	WORD	
ModuleStatus_ProfibusOutputDatasize	768	WORD	
ModuleStatus_InputDataStartAddress	1000	WORD	
ModuleStatus_OutputDataStartAddress	3000	WORD	
ModuleStatus_Reserved0	0	WORD	
ModuleStatus_ByteSwapHInputLOutputData	0	WORD	
ModuleStatus_Modulesoftwaremajorversionnumber	3329	WORD	
ModuleStatus_ProfibusSlaveConfiguredList		ARRAY[0..7] OF WORD	
ModuleStatus_ProfibusDataTransferStatus		ARRAY[0..7] OF WORD	
ModuleStatus_ProfibusSlaveDiagnosticStatus		ARRAY[0..7] OF WORD	
ModuleStatus_ProfibusMasterOperatingState	49152	WORD	
ModuleStatus_ProfibusIdentNumber	61464	WORD	
ModuleStatus_ProfibusMasterSerialNumber		ARRAY[0..1] OF WORD	
ModuleStatus_ProfibusSoftwareVersion	12290	WORD	
ModuleStatus_ProfibusMasterModuleStatus	260	WORD	
ModuleStatus_ProfibusCRC32	40477182...	UDINT	
ModuleStatus_PTQModuleCRC32	28551274...	UDINT	
ModuleStatus_Applicationprogramscancounter	33019	WORD	
ModuleStatus_ModuleProfibusoutputimagedataupd...	50933	WORD	
ModuleStatus_ModuleProfibusinputimagedataupdat...	50934	WORD	
ModuleStatus_Moduleoutmailboxcounter	17	WORD	
ModuleStatus_Moduleinmailboxcounter	17	WORD	
ModuleStatus_ModulealarmINDreceivecounter	0	WORD	
ModuleStatus_ModulealarmCONreceivecounter	0	WORD	
ModuleStatus_Reserved1	0	WORD	
ModuleStatus_Reserved2	0	WORD	
ModuleStatus_Modulebackplanereadcount	11789	WORD	
ModuleStatus_Modulebackplanewritecount	11788	WORD	
ModuleStatus_Modulebackplaneerrorcount	0	WORD	
ModuleStatus_FileErrorWord	0	WORD	
ModuleStatus_HSBYPassiveStatus	0	BYTE	
ModuleStatus_HSBYPassivenumberofslaves	0	BYTE	

To test the Unity interface

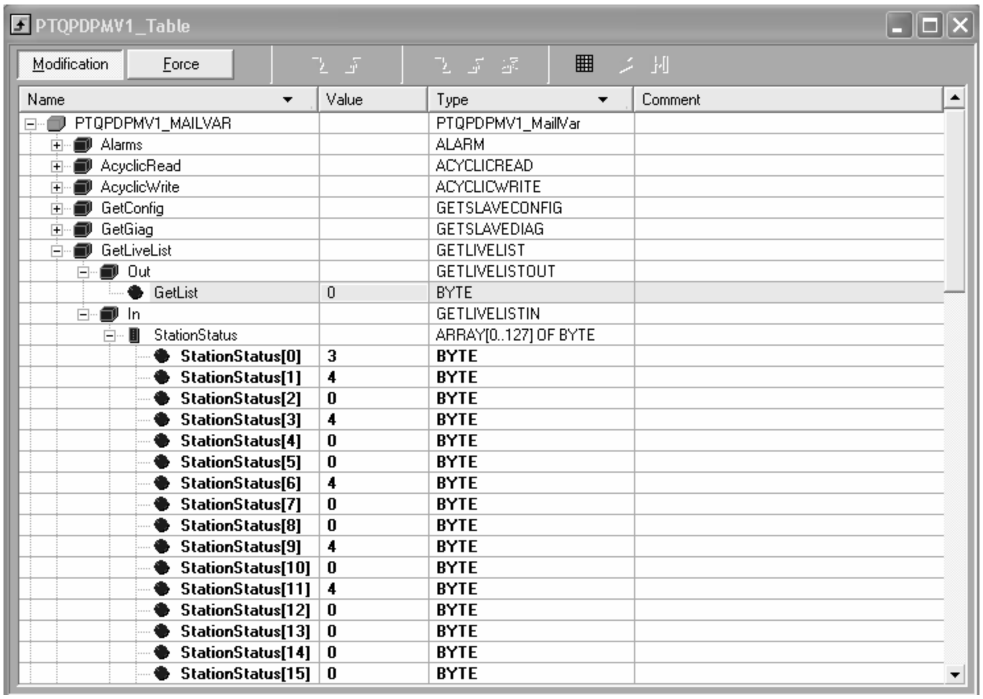
The following steps show how to use the mailbox message *GetLiveList*.

> **Note:** Make sure the Unity program is connected and the processor is running.

1 From the table (public folder), select **PTQPDPMV1_MAILVAR.GETLIVELIST.OUT.GETLIST** and set it to **1**.



2 The *GetLiveList* response will be automatically copied into the *GetLiveList.In.StationStatus* array. The following illustration shows an example where slave address 3 is connected to the Master (address 1). The *GetList* bit is automatically cleared. Refer to Mailbox Messaging (page 151) for specific help on the mailbox commands and response values.



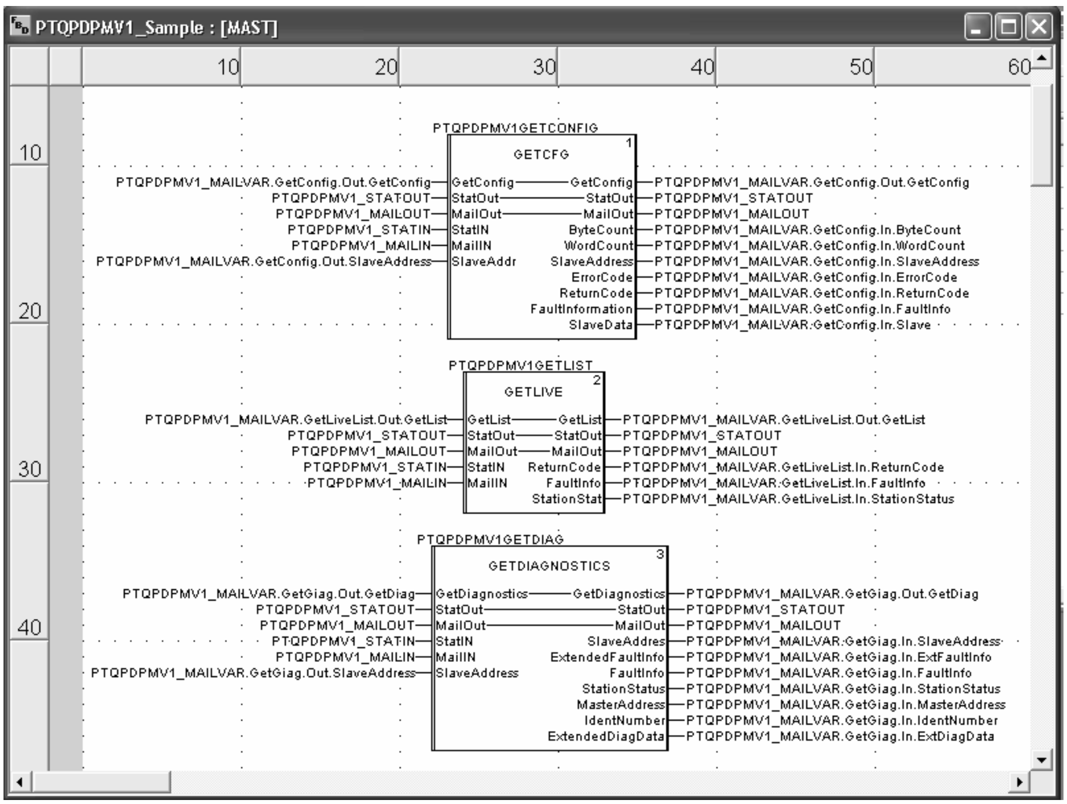
Name	Value	Type	Comment
PTQPDPMV1_MAILVAR		PTQPDPMV1_MailVar	
Alarms		ALARM	
AcyclicRead		ACYCLICREAD	
AcyclicWrite		ACYCLICWRITE	
GetConfig		GETSLAVECONFIG	
GetGiag		GETSLAVEDIAG	
GetLiveList		GETLIVELIST	
Out		GETLIVELISTOUT	
GetList	0	BYTE	
In		GETLIVELISTIN	
StationStatus		ARRAY[0..127] OF BYTE	
StationStatus[0]	3	BYTE	
StationStatus[1]	4	BYTE	
StationStatus[2]	0	BYTE	
StationStatus[3]	4	BYTE	
StationStatus[4]	0	BYTE	
StationStatus[5]	0	BYTE	
StationStatus[6]	4	BYTE	
StationStatus[7]	0	BYTE	
StationStatus[8]	0	BYTE	
StationStatus[9]	4	BYTE	
StationStatus[10]	0	BYTE	
StationStatus[11]	4	BYTE	
StationStatus[12]	0	BYTE	
StationStatus[13]	0	BYTE	
StationStatus[14]	0	BYTE	
StationStatus[15]	0	BYTE	

3.2 Function Blocks Operation Overview

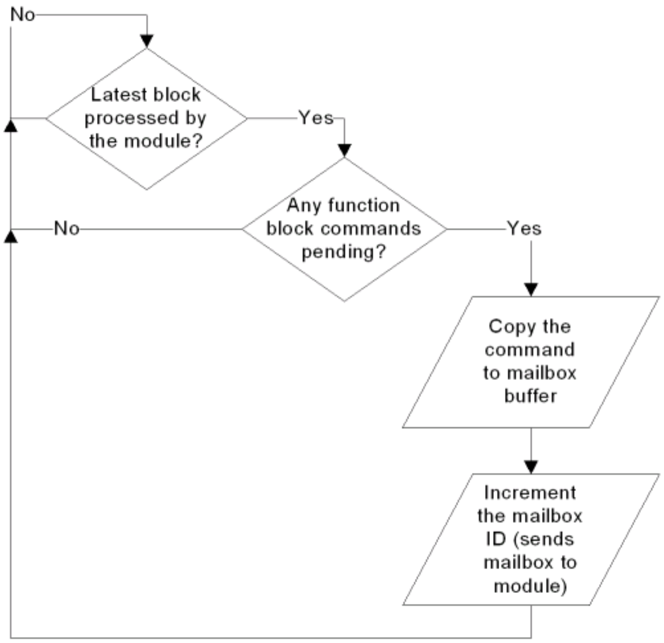
Function blocks define software components or modules that perform a specific function. Each function block has its own, pre-defined set of inputs and outputs.

The function blocks provided with the PTQ-PDPMV1 module contain the logic to handle PROFIBUS acyclic mailbox messages and alarms. They transfer data between the main output/input mailbox arrays and the corresponding slave devices.

The PTQ-PDPMV1 module is ready to receive a mailbox message from the processor when all function blocks have been called in the main program, which is provided in the sample.



```
MailOut[0] = StatIn.MailBoxData_LastOutMailboxMessageID
```



Each mailbox data structure is implemented through variables that are divided into Out and In data structures, where:

Out = values copied from the processor to the module

In = values copied from the module to the processor

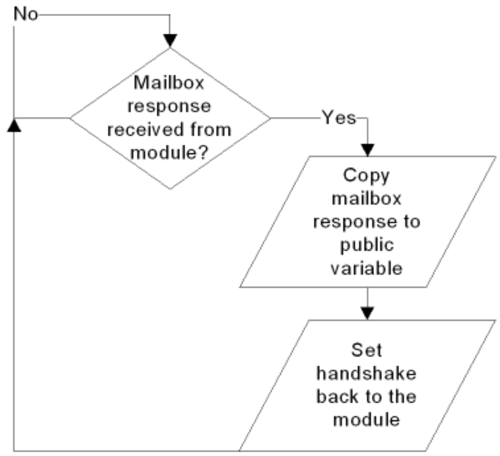
Each Out data structure contains a Cmd bit. After the Cmd bit is toggled, the logic will increment the mailbox ID (output) to send the mailbox request to the module.

The following illustration shows the interface for the SetOperatingMode mailbox:

SETOPERATINGMODE		SETOPERATINGMODE
Out		SETOPERATINGMODEOUT
SelectOperate	**0**	**BYTE**
SelectStop	**0**	**BYTE**
SelectClear	**0**	**BYTE**
ConfRequired	**0**	**BYTE**
In		SETOPERATINGMODEIN
Mode	**0**	**BYTE**
ConfRequired	**0**	**BYTE**
FaultInformation	**0**	**WORD**
StartSlaves		STARTSLAVES

The following condition indicates that the module has a mailbox response to be sent to the processor. Therefore, the function block implementation will handle the block by copying the response data to the appropriate mailbox data structure.

```
StatOut.ModuleStatus_LastinMailBoxMessageID <>
StatIn.MailBoxData_CurrentInMailboxControlIndex
```

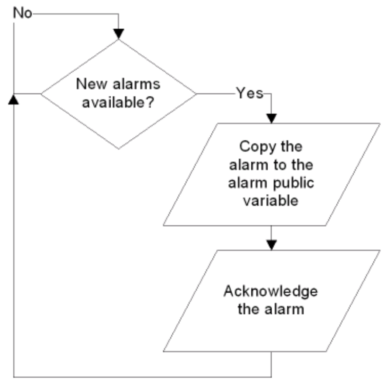


The module will increment *StatIn.MailBoxData_CurrentAlarmControlIndex* when the module contains alarms to be sent to the processor. The function block implementation will then copy the alarm to the appropriate data structure. The function block implementation uses the following expression to verify if any alarms are available:

```
StatOut.ModuleStatus_LastAlarmControlindex <>
StatIn.MailBoxData_CurrentAlarmControlIndex
```

After the alarm is copied, the logic then updates the alarm index for handshaking purposes:

```
StatOut.ModuleStatus_LastAlarmControlindex :=
StatIn.MailBoxData_CurrentAlarmControlIndex
```



Please refer to Mailbox Messaging (page 151) for further information about each mailbox parameter.

The following section provides examples of data structure groupings.

3.3 Derived Function Blocks Overview

The Unity Pro programming language for Schneider Electric Automation Quantum processors supports user-defined function blocks (DFB). The user function block types (Derived Function Blocks) are developed by the user using one or more languages (according to the number of sections). These languages are:

- Ladder language
- Structured Text language
- Instruction List language
- Functional block language FBD

A DFB type can have one or more instances where each instance is referenced by a name (symbol), and possesses DFB data types.

Derived Function blocks defined by Unity Pro software are entities containing:

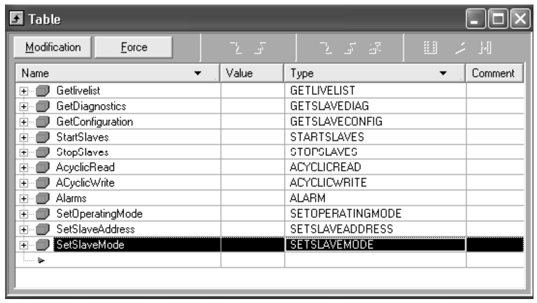
- Input and output variables acting as an interface with the application
- A processing algorithm that operates input variables and completes the output variables
- Private and public internal variables operated by the processing algorithm

3.3.1 Using the Derived Function Blocks

To simplify programming procedures, ProSoft Technology has included a Unity Pro XFM Functional Module used for communication with the PTQ-PDPMV1 module. The Functional Module provides easy access to the Master's cyclic and acyclic data. Specific mailbox acyclic commands are also provided to perform functions such as *Get Live List* and *Get Slave Diagnostics*, and to perform *Freeze* and *Sync* commands, and others.

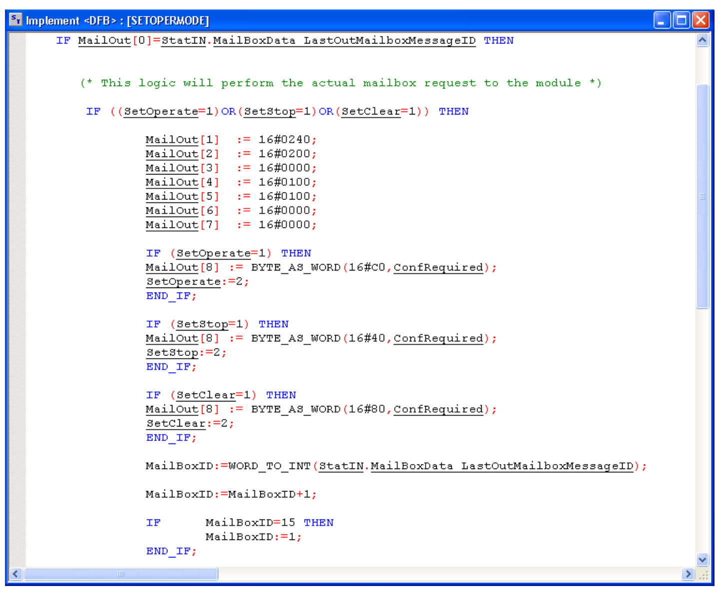
Note: It is not intended to include in-depth programming information in this reference manual. You should, therefore, be familiar with IEC Function Block programming and Unity Pro programming language.

The PTQ_PDPMV1_Sample Functional Module supports input and output variables used for PTQ status, acyclic mailbox and slave cyclic I/O data. All input information is located in the <Inputs>: *StatIn, MailIn* and *DataIn* area (data delivered to the Unity processor) and all output information is located in the <input/output>: *StatOut*, *MailOut*, and *DataOut* (data sent to the PTQ module). You can access the supported mailboxes in the provided table.



Every mailbox has its own function block that has a unique "Impl" ST derived FB type file.

The following illustration shows part of the function block implementation (structured text code) that performs the mailbox request after the command register is triggered by the processor application. For example, the *SetOperatingMode* command is executed when the *Out (SetOperate,SetStop,SetClear).Cmd* bit is true.



This bit is accessed and controlled in a tag in the provided table.

Note: Refer to Special Function Mailbox Messaging Commands (page 153) for more information about Mailbox Commands.

Mailbox data values are pre-defined for the specific mailbox command.

SETOPERATINGMODE		SETOPERATINGMODE
Out		SETOPERATINGMODEOUT
SelectOperate	**0**	**BYTE**
SelectStop	**0**	**BYTE**
SelectClear	**0**	**BYTE**
ConfRequired	**0**	**BYTE**
In		SETOPERATINGMODEIN
Mode	**0**	**BYTE**
ConfRequired	**0**	**BYTE**
FaultInformation	**0**	**WORD**

The first statement `MailOut[1] := 16#0240;` represents the Message Information of the command (4002h) Set Operating Mode (see specific mailbox command).

Note: The information is byte swapped for PTQ Master module (Motorola big-endian format).

The remaining values [2] to [7] set the Command, Data Size, Frame Count, Frame Number, Offset High and Low byte header information. Again, these values are pre-defined and controlled by the FB.

Most mailbox commands have response information. Refer to Mailbox Messaging (page 151) for more information. The response information will be written to the In area of the *SetOperatingMode* mailbox area. This information can be read after the mailbox is received and confirmed by the ID information contained in the *CurrentMailboxControlIndex* value.

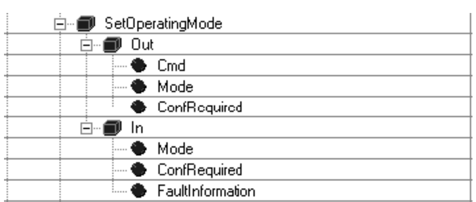
```
IF StatOut.ModuleStatus_LastinMailboxMessageID<>
   StatIN.MailBoxData_CurrentInMailboxControlIndex THEN
```

When this statement is true and the *Set Operating Command* was executed the following code will be executed:

```
IF MailIn[2] = 16#0200 THEN

    WORD_AS_BYTE (MailIn[8],Mode ,ConfRequired);
    FaultInformation:= MailIn[15];
```

The appropriate return value(s) for *Set Operating Mode* can now be read or accessed in the *In.Mode*, *ConfRequired* and *FaultInformation* values.



Each mailbox command can be executed and responded to using similar procedures as outlined above.

3.4 Using Mailbox Function Blocks

Function blocks define software components or modules that perform a specific function. Each function block has its own, pre-defined set of inputs and outputs.

The function blocks provided with the PTQ-PDPMV1 module contain the logic to handle acyclic mailbox messages and alarms. They transfer data between the main output/input mailbox arrays and the corresponding slave devices.

3.4.1 Overview

The mailbox function blocks build mailbox requests to the module and read the mailbox response from the module. These mailbox function blocks are optional, meaning that the project will update PROFIBUS data and status information even if no function blocks are used.

3.4.2 Configuration

The mailbox function block contains inputs, outputs and input/output pins that must be associated to specific variables.

The mailbox function blocks (except *GetAlarm,* which will be covered later) require the usage of the following pins (common for all mailbox function blocks):

Pin Name	Pin Type	Description
StatIn	Input	Input Status pin. Must be associated to the imported variable *PTQPDPMV1_STATIN*. It contains the status transferred from the module allowing the mailbox function block to receive the acknowledgment that the mailbox request was processed by the module. It is used also to check if a new mailbox response is available. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
MailIn	Input	Input Mailbox pin. Must be associated to the imported variable *PTQPDPMV1_MAILIN*. It contains the mailbox response message that is handled by the function block according to its mailbox ID. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
StatOut	Input/Output	Output Status pin. Must be associated to the imported variable *PTQPDPMV1_STATOUT*. It is used to check if a new mailbox response is available. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
MailOut	Input/Output	Output Mailbox pin. Must be associated to the imported variable *PTQPDPMV1_MAILOUT*. This variable stores the mailbox output variable that is updated from the function block when a new mailbox request is performed to the module. It consists on an array of words. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
"Trigger"	Input/Output	Move a value of 1 to this register to initiate the mailbox request. A request can only be initiated if its current value is 0 and all triggers for the other mailbox function blocks also have a value of 0. The actual name for this trigger register will be specific for each mailbox function block.

3.4.3 Trigger Bytes

Mailbox requests are initiated by "triggers" (bytes) that are defined as input/output pins. A mailbox request is initiated after the application moves a value of 1 to the appropriate trigger byte.

Only one mailbox function may be active at any given time. Therefore, in order for a mailbox request to be carried out, the value of all mailbox triggers in your application must be equal to 0 at the time the request is made. If you are using more than one mailbox function block, add program code to guarantee that this condition is satisfied.

The processor will only allow a new mailbox request to be sent out after it has received confirmation that the previous mailbox request was acknowledged by the module. The processor determines this condition by checking the status of all trigger bytes (0=OK). This procedure also prevents more than one mailbox request from being sent out during a single PLC scan.

The following table shows the trigger bytes used for each mailbox function block:

Function Block Name	Description	Trigger
ACYCREAD	Acyclic Read Mailbox	Acyclicread
ACYCLWRITE	Acyclic Write Mailbox	Acyclicwrite
GETALARMS	Alarm Mailbox	-
GETCFG	Get Configuration Mailbox	GetConfig
GETDIAGNOSTICS	Get Diagnostics Mailbox	GetDiagnostics
GETLIVE	Get Live List Mailbox	GetList
SETADDRESS	Set Slave Address Mailbox	SetAddress
SETOPERMODE	Set Operating Mode Mailbox	SetOperate, SetStop, SetClear
SETSLMODE	Set Slave Mode Mailbox	SetSlaveMode
STARTSTOPSLAVE	Set Start and Stop Slaves Dynamically	StartSlaves, StopSlaves
COLDBOOT	Remote Coldboot from PLC	ColdBoot

The *Get Alarm* function block does not require a trigger because this mailbox is initiated from the module. Refer to Alarm Indication (page 170) for more information.

The trigger byte is a variable that can assume different states as follows:

Value	Description
0	OK to send new mailbox request. The last mailbox request was already acknowledged by the module.
1	Mailbox request to be performed. The Quantum program should make sure that the required conditions are satisfied (as previously discussed) before moving a value of 1 to the trigger register. The function block will then build the mailbox request by copying all mailbox input parameters to the mailbox output variable that is transferred to the module through the backplane. Then the function block will automatically change the trigger's value to 2.
2	Processor has performed the mailbox request and is waiting for the acknowledgment from the module. The acknowledgment informs that the module has received the request (the actual mailbox response is actually sent later). After the acknowledgment is received, the function block will reset the trigger's value back to 0.

3.4.4 Specific Input Pins

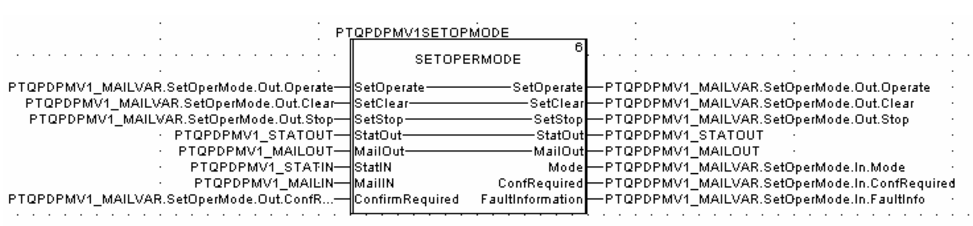
Each function block has input pins specifically for each mailbox. For example, in order to send a *Get Diagnostics* mailbox, the application must set the PROFIBUS slave address input pin. The processor program must configure the input pins before performing the mailbox request by moving a value of 1 to the mailbox trigger.

3.4.5 Specific Output Pins

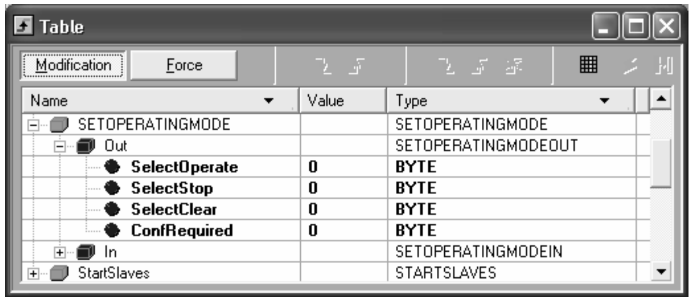
Each function block contains output pins that are updated after the mailbox response is received by the processor. For example, the *Get Diagnostics* function block has an *ExtendedDiagData* output pin that stores the diagnostic information received from the slave.

Example

If the *Set Operating Mode* mailbox function block is used as follows:



You can add *SETOPERMODE* to the table with three trigger variables for the *Set Operating Mode* mailbox. Start by moving a value of 1 to *SelectStop* in order to set the module's mode to STOP.

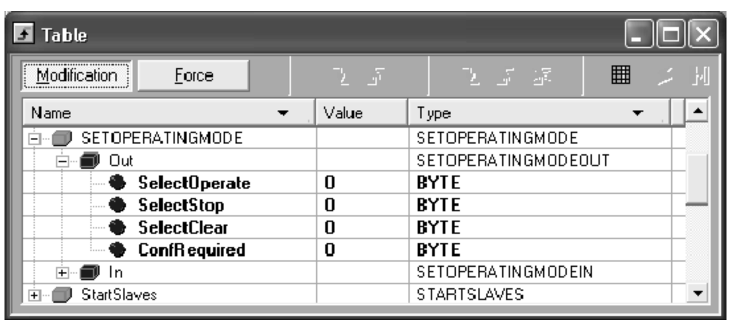


At this point, you should notice the following LED display, indicating that the module's mode was changed to STOP:

LED	Status
MSTR STAT	RED
COM STAT	OFF
DBASE STAT	GREEN
TK HOLD	GREEN

You will also notice that the function block automatically clears the trigger byte after it receives the acknowledgment from the module.

Move a value of 1 to the *SelectOperate* trigger byte.



At this point, you should notice the following LED display, indicating that the module's mode was changed to OPERATE:

LED	Status
MSTR STAT	GREEN
COM STAT	GREEN or OFF
DBASE STAT	GREEN
TK HOLD	GREEN

The *COM STAT* LED will be either *GREEN* if the Master is communicating with all slaves, blinking if it is communicating with some of the slaves or *OFF* if it is not communicating with any slaves.

You will also notice that the function block automatically clears the trigger byte after it receives the acknowledgment from the module.

3.5 Mailbox Overview

This section provides a brief description on how to use each mailbox function block. Refer to Mailbox Messaging (page 151) for detailed information about each mailbox parameter.

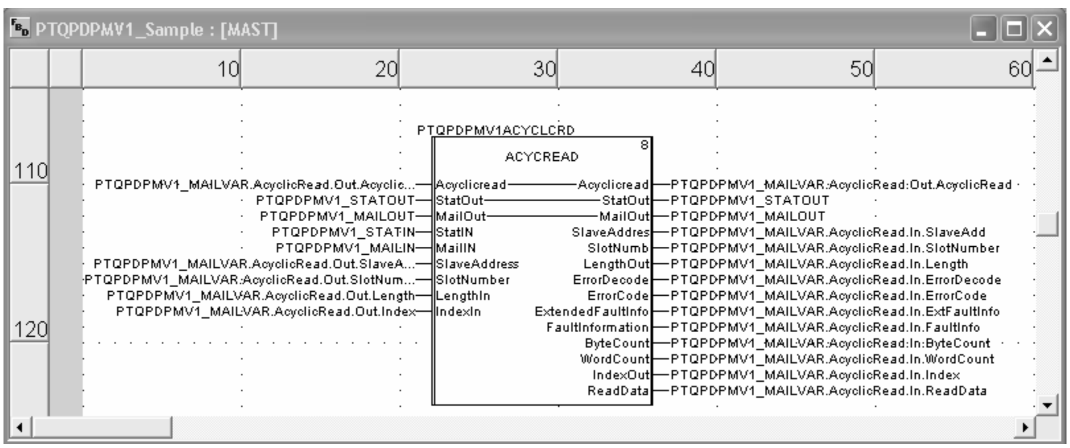
3.5.1 Acyclic Read Mailbox

Function Block: *ACYCREAD*

Trigger Byte: *Acyclicread*

Description: The *ACYCREAD* mailbox is used to perform an *Acyclic Read* request to a PROFIBUS slave device. The input pins *SlaveAddress* (PROFIBUS slave address), *SlotNumber* (slot number), *IndexIn* (index number), and *LengthIn* (length - number of bytes associated to acyclic read operation) must be configured before triggering the mailbox request. The acyclic read response data is copied to the *ReadData* output pin. The status information is available in the output pins (*ErrorCode, ErrorDecode, ExtendedFaultInfo, and FaultInformation*).

The following illustration shows a sample instance of the *Acyclic Read* mailbox function block:



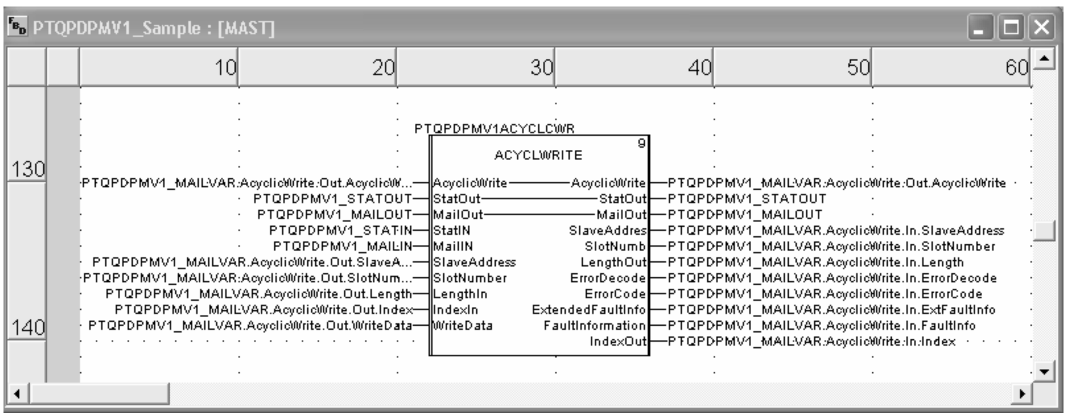
3.5.2 Acyclic Write Mailbox

Function Block: *ACYCLWRITE*

Trigger Byte: *Acyclicwrite*

Description: The *ACYCLWRITE* function block is used to perform an *Acyclic Write* request to a PROFIBUS slave device. The input pins *SlaveAddress* (PROFIBUS slave address), *SlotNumber* (slot number), I*ndexIn* (index number) and *LengthIn* (length - number of bytes associated to acyclic read operation) must be configured before triggering the mailbox request. The actual data to be written to the PROFIBUS slave should be associated to the *WriteData* input pin. The status information is available at the output pins (*ErrorCode, ErrorDecode, ExtendedFaultInfo and FaultInformation*).

The following illustration shows a sample instance of the *Acyclic Write* mailbox function.



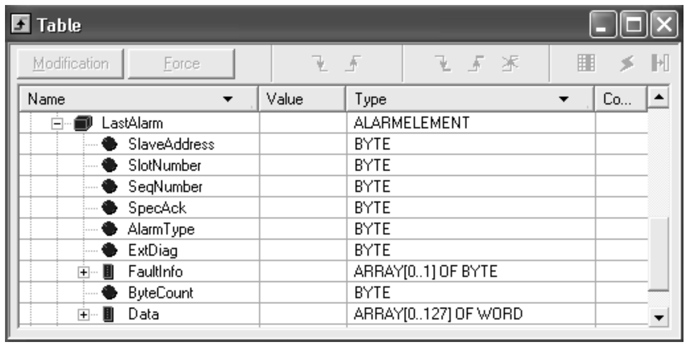
3.5.3 Alarm Mailbox

Function Block: *GETALARMS*

Trigger Byte: The *GETALARMS* function block does not require a trigger because this mailbox is initiated from the module.

Description: The *GETALARMS* mailbox is used to read the alarm mailbox messages sent by the module. The module will automatically generate the alarm mailboxes after it receives the alarm message from the PROFIBUS slave.

The last alarm received is copied at the *LastAlarm* output pin. This is a data structure that contains all alarm information:



Name	Value	Type	Co...
LastAlarm		ALARMELEMENT	
SlaveAddress		BYTE	
SlotNumber		BYTE	
SeqNumber		BYTE	
SpecAck		BYTE	
AlarmType		BYTE	
ExtDiag		BYTE	
FaultInfo		ARRAY[0..1] OF BYTE	
ByteCount		BYTE	
Data		ARRAY[0..127] OF WORD	

This function block also keeps track of the last 100 alarms through the *HistAlarm* output pin.

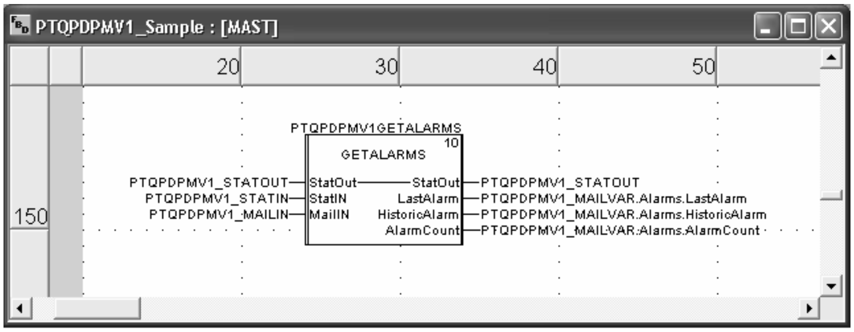
For example, if the module receives 100 alarms (first alarm - Sequence Number = 1, second alarm - Sequence Number = 2, and so on), after alarm #100 is received, the processor application could refer to these alarms stored at the following output pins:

```
Last Alarm - Alarm #100
HistAlarm[1] - Alarm # 99
HistAlarm[2] - Alarm # 98
HistAlarm[3] - Alarm # 97
HistAlarm[4] - Alarm # 96
HistAlarm[5] - Alarm # 95
HistAlarm[6] - Alarm # 94
HistAlarm[7] - Alarm # 93
HistAlarm[8] - Alarm # 92
HistAlarm[9] - Alarm # 91
HistAlarm[10] - Alarm #90
```

If the *HistoricAlarm* buffer is full and it receives a new alarm, then the oldest alarm in the queue will be deleted to reserve space for the new alarm.

The *AlarmCount* output pin is incremented every time the alarm mailbox is received. This register will roll over at 30000. The processor application can keep track of this register to determine when the processor has received a new alarm mailbox message from the module.

The following illustration shows a sample instance of the *GetAlarms* mailbox function block:

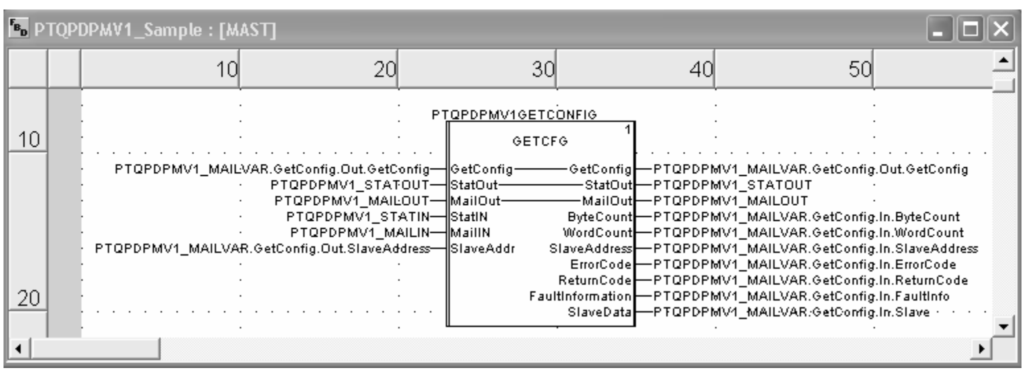


3.5.4 GetConfiguration Mailbox

Function Block: *GETCFG*

Trigger Byte: *GetConfig*

Description: The *GETCFG* function block can be used to read the configuration of any PROFIBUS slave connected to the PTQ-PDPMV1 module. The *SlaveAddr* input pin must be configured with the PROFIBUS slave address of the PROFIBUS device. The configuration data is stored at the *SlaveData* output pin. The byte count of the slave configuration is stored at *ByteCount* output pin. The *ErrorCode, ReturnCode, and FaultInformation* output pins can be used for status verification.

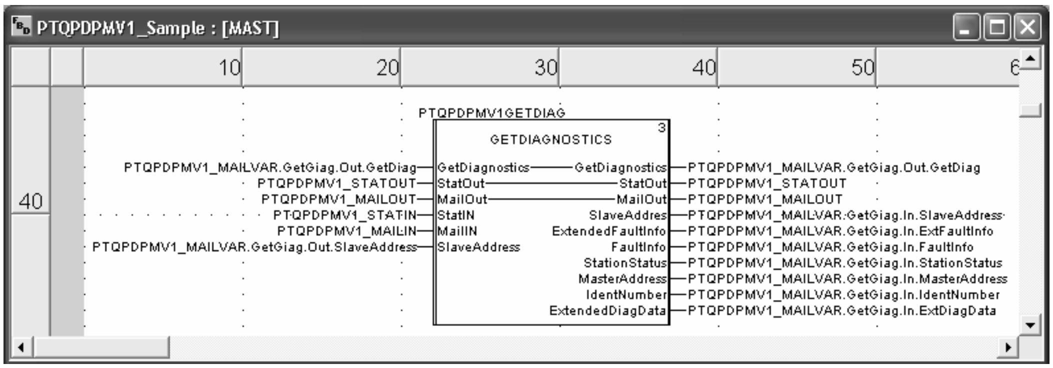


3.5.5 GetDiagnostics Mailbox

Function Block: *GETDIAGNOSTICS*

Trigger Byte: *GetDiagnostics*

Description: The *GETDIAGNOSTICS* function block can be used to read the diagnostics from any PROFIBUS slave connected to the PTQ-PDPMV1 module. The slave address must be set at the *SlaveAddress* input pin. The diagnostics data is copied at the *ExtendedDiagData* output pin. The number of bytes of the diagnostics message is stored at the *ByteCount* output pin (status+identification+extended diagnostics). The *ExtendedFaultInfo* and *FaultInformation* output pins can be used for status information. The Master address is stored at the *MasterAddress* output pin.



3.5.6 GetLiveList Mailbox

Function Block: *GETLIVE*

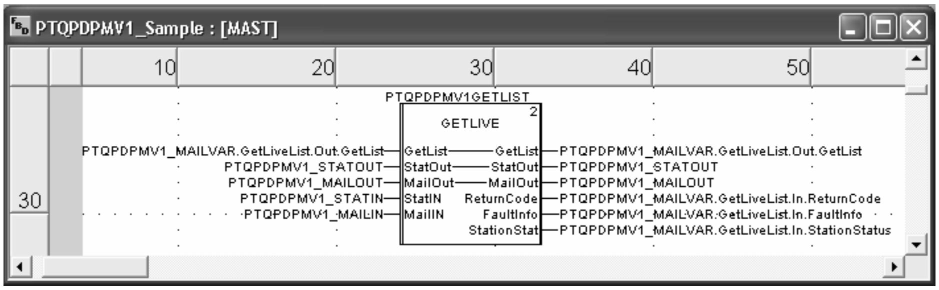
Trigger Byte: *GetList*

Description: The *GETLIVE* function block can be used to read the live list from the module containing the status of each device at the PROFIBUS network. The live list is stored at the *StationStatus* output pin. The live list data is an array of bytes stored as follows:

```
StationStatus [0] - status of device configured with PROFIBUS address 0
StationStatus [1] - status of device configured with PROFIBUS address 1
StationStatus [2] - status of device configured with PROFIBUS address 2
StationStatus [3] - status of device configured with PROFIBUS address 3
StationStatus [4] - status of device configured with PROFIBUS address 4

Etc…
```

Refer to Mailbox Messaging Error Codes (page 180) for further information about the valid status codes. The *ReturnCode* and *FaultInformation* output pins can be used for mailbox status information.

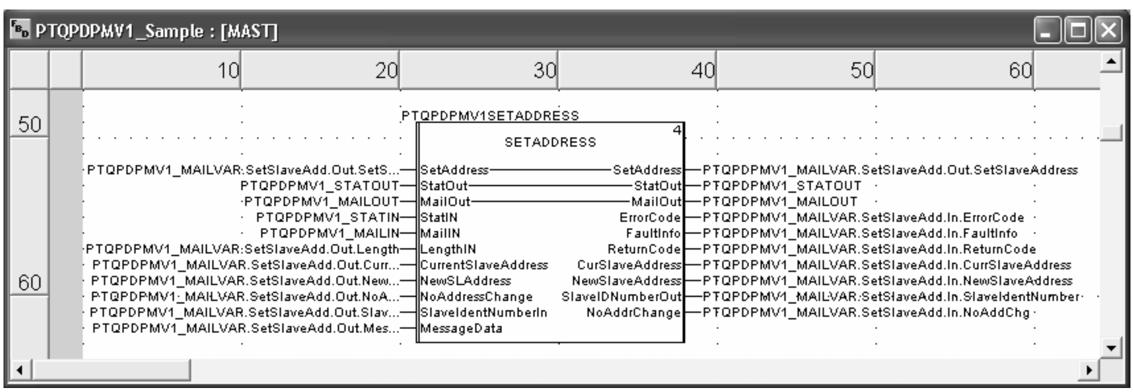


3.5.7 SetSlaveAddress Mailbox

Function Block: *SETADDRESS*

Trigger Byte: *SetAddress*

Description: The *SETADDRESS* mailbox can be used to change the slave address. Only specific PROFIBUS devices support this feature. The application must set the *CurrentSlaveAddress* (current address) and *NewSLAddress* (new address) input pins. It is also possible to deliver user data through the *MessageData* input pin (the number of bytes must be set through the *LengthIn* input pin). The *SlaveIdentNumberIn* input pin must be set with the *Ident* number for the slave. The *FaultInformation* output pin can be check for mailbox status information.



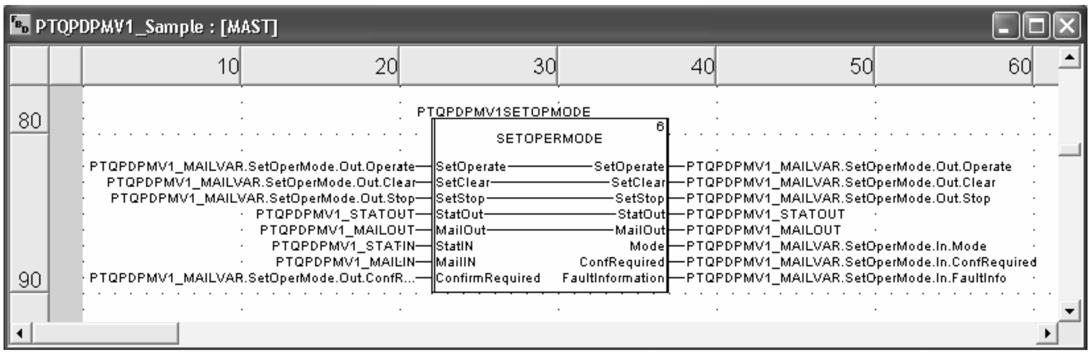
3.5.8 SetOperatingMode Mailbox

Function Block: *SETOPERMODE*

Trigger Byte: *SetOperate, SetStop, SetClear*

Description: The following trigger values can be used to change the current operating mode of the module:

SETOPERATINGMODE.Out.SelectOperate = Set Operate
SETOPERATINGMODE.Out.SelectStop = Set Stop
SETOPERATINGMODE.Out.SelectClear = Set Clear

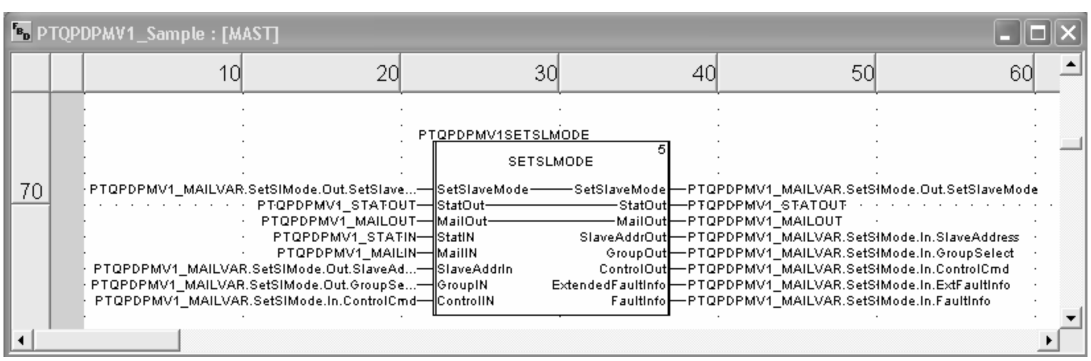


3.5.9 SetSlaveMode Mailbox

Function Block: *SETSLMODE*

Trigger Byte: *SetSlaveMode*

Description: The *SETSLMODE* function block can be used to request the module to sync, unsync, freeze, or unfreeze. The slave address must be selected through the *SlaveAddrIn* input pin. If the operation is directed to a group of slaves, then the group number must be set through the *GroupIn* input pin parameter. The actual code that will select the operation type must be configured through the *ControlIn* input pin. Please check the slave's user manual for valid control codes.



Important Note: The next mailbox is only for Anybus firmware version 3.50 and later. Earlier released versions do not support this feature.

To determine your Anybus firmware version, use ProSoft Configuration Builder to connect to the module and open the *Diagnostics* window. On the *Main* menu, press **[3]** to view the Control Registers. Note the firmware version number displayed on this screen.

```
Control Registers:
Boot Loader Ver.  = 2.05
Serial Number     = FFFFFFFF
Firmware version  = 3.80
Module Status     = 0401    (0x0401=Running with slaves)
```

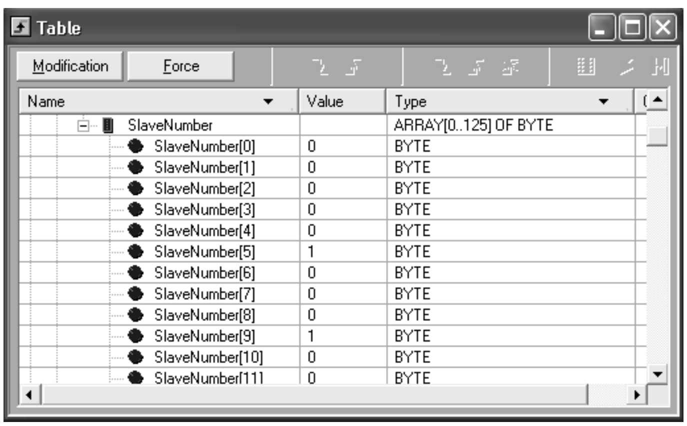
3.5.10 StartStopSlaves Mailbox

Function Block: *STARTSTOPSLAVES*

Trigger Byte: *StartSlaves, StopSlaves*

Description: The *STARTSTOPSLAVES* function block can be used to request the module to start or stop certain slaves dynamically. The slave address must be selected through the *SlaveNumber* input pin. This is an array of 126 slaves. Change the value for a specific slave from 0 to 1 to start or stop communication with the Master.

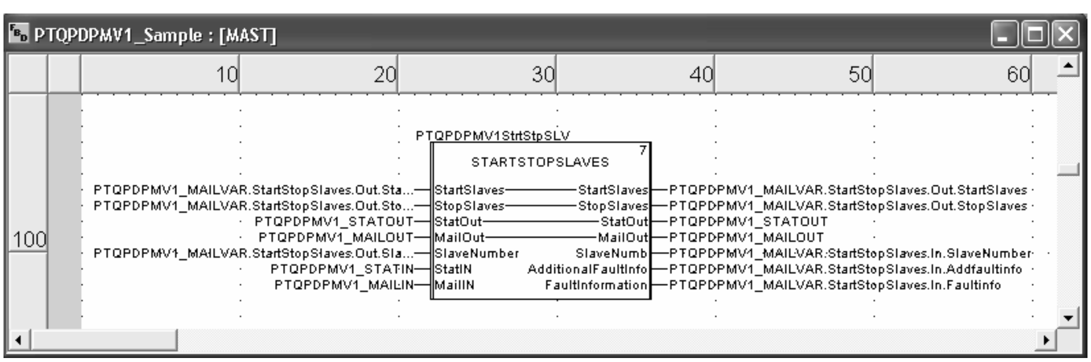
The following illustration shows that when you execute the mailbox command, Slave #5 and Slave #9 will start communicating with the Master.



Table

Name	Value	Type
SlaveNumber		ARRAY[0..125] OF BYTE
SlaveNumber[0]	0	BYTE
SlaveNumber[1]	0	BYTE
SlaveNumber[2]	0	BYTE
SlaveNumber[3]	0	BYTE
SlaveNumber[4]	0	BYTE
SlaveNumber[5]	1	BYTE
SlaveNumber[6]	0	BYTE
SlaveNumber[7]	0	BYTE
SlaveNumber[8]	0	BYTE
SlaveNumber[9]	1	BYTE
SlaveNumber[10]	0	BYTE
SlaveNumber[11]	0	BYTE

You can confirm the execution of the mailbox by verifying that the *SlaveNumb* output pin exactly matches the *SlaveNumber* input pin.



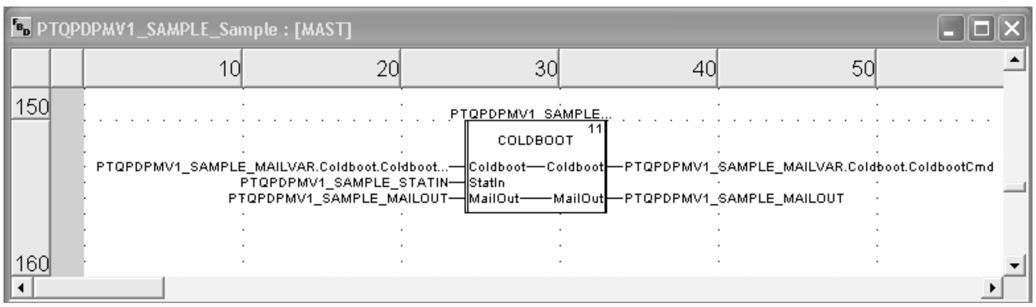
3.5.11 Coldboot Mailbox

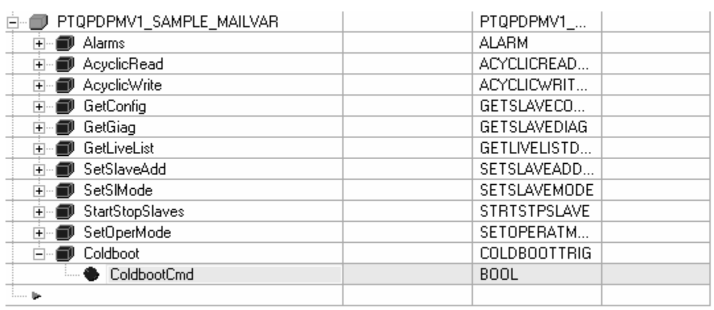
Important: The *Coldboot* mailbox is only supported on PTQ-PDPMV1 modules running firmware version 1.19 or newer. Earlier versions of the firmware do not support this feature. If you require this functionality, please contact ProSoft Technical Services for information on how to upgrade your module.

Function Block: *COLDBOOT*

Trigger Byte: *ColdBoot*

Description: The *COLDBOOT* function block allows you to remotely reboot the module. To trigger a reboot of the module, change the value of the *Coldboot* bit from **0** (zero) to **1** (one). The bit is reset back to 0 when the function is executed.





HSBY Note: This function block will reset both the local (active) Master and the remote (passive) Master.

Using a Control Word to Reboot the Module

If you need to cold boot the module from the processor without using the *Coldboot* mailbox, use the *ModuleStatus_SetOperatingMode* control word variable.

To reboot the module

1 Enter the hexadecimal value **16#9999** in the *ModuleStatus_SetOperatingMode* register, as shown in the following illustration.

PTQPDPMV1_NoMB_StatOut		StatOutF
ModuleStatus_SetOperatingMode	16#0000	WORD
ModuleStatus_Reserved	0	WORD
ModuleStatus_ProfibusCRC32	1160380008	UDINT
ModuleStatus_ModuleCRC32	848886032	UDINT

2 Add the following lines to the program file:

Coldbootfunction : [MAST]

```
IF PTQPDPMV1_NoMB_StatIn.ModuleStatus_ControlData=16#0099 THEN
    PTQPDPMV1_NoMB_StatOut.ModuleStatus_SetOperatingMode:=16#0000;
END_IF;
```

This logic will reset the value in *ModuleStatus_SetOperatingMode* to 16#0000.

Note: It is normal for the remote (passive) Master in Hot Standby applications to reboot twice during this procedure.

4 Configuring the Processor with Concept 2.6

In This Chapter

Important: The following steps are for Concept version 2.6 or newer. Earlier versions of Concept are not supported.

HSBY Note: Concept software does not support 140CPU67160 processor and therefore does not support the PTQ-PDPMV1 HSBY functions.

Important Note: Concept software does not report whether the PTQ module is present in the rack, and therefore is not able to report the health status of the module when the module is online with the Quantum processor. Please consider this when monitoring the status of the PTQ module.

4.1 Overview

This section will guide you through the steps required to set up your Concept Project with the PTQ-PDPMV1 module. There are a total of 6 steps required as follows:

Step 1: Export the Files from PCB (page 106)

This step shows how to export the required files from PCB (containing function blocks, variables and data type definitions) that will be used during this procedure.

Step 2: Convert the Function Blocks (page 109)

The .ASC function blocks (exported at step 1) must be converted before used in the Concept project. This step shows how to convert the function blocks from .ASC to .DFB format.

Step 3: Set Up the Concept Project (page 113)

This step shows how to set up the Concept Project and configure the required amount of processor memory for your application.

Step 4: Import the Variables (page 116)

This step shows how to import the variables into your Concept project by using the .txt file (obtained at step 1). The PCB configuration will determine the addressing of the variables.

Step 5: Create the Function Block Instances (page 119)

This step shows how to create an instance of the function blocks that were converted at step 2. It also shows that some function block pins must be linked with the variables that were imported at step 4.

Step 6: Download the Project to the Quantum Processor (page 126)

Once you download your project to the Quantum processor, the procedure is completed.

After you followed these steps, you can refer to the following topics for more information on how to perform basic tasks:

Using the Concept Project (page 127)

This section shows how to access PROFIBUS data and status information through the imported variables. It also shows how to perform a mailbox request from the processor.

Mailbox Overview (page 136)

This section provides a general overview of the mailbox function blocks that are supplied as a sample application.

4.2 Before You Begin

1 Verify that your computer has the following software tools installed:
- ProSoft Configuration Builder (version 2.0.0 Build 15 or later)
- Concept Programming Unit (version 2.6 or later)

2 Create a folder *C:\project\DFB*, where:

C:\project - will store the main Concept project (.PRJ)
C:\project\DFB - will store the data type definition file (.DTY) and the function blocks that will be used by the Concept project.

Warning: The Function Block is intended for new installations of PTQ-PDPMV1. If you have an existing installation, the following procedure will overwrite your settings, and may cause loss of functionality. DO NOT overwrite a working application until you have thoroughly reviewed the following topics.

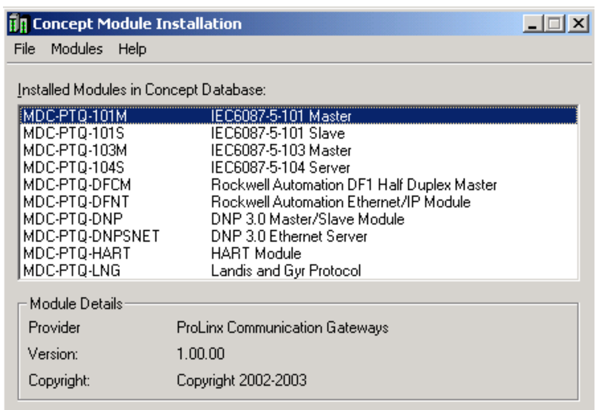
4.3 Information for Concept Version 2.6 Users

This guide uses Concept PLC Programming Software version 2.6 to configure the Quantum PLC. The ProTalk installation CD includes MDC module configuration files that help document the PTQ installation. Although not required, these files should be installed before you proceed to the next section.

4.3.1 Installing MDC Configuration Files

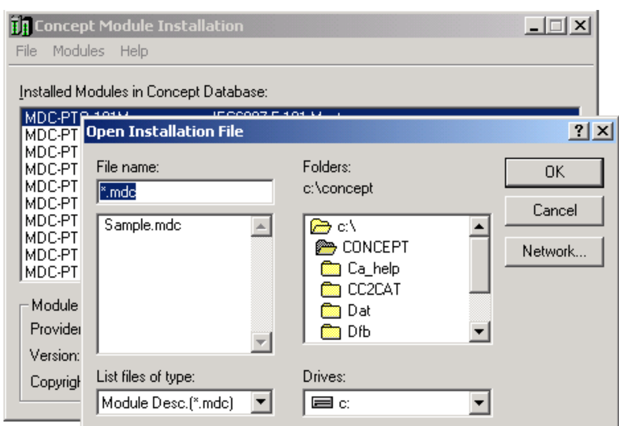
1 From a computer with Concept 2.6 installed, choose **START** > **PROGRAMS** > **CONCEPT** > **MODCONNECT TOOL**.

This action opens the *Concept Module Installation* dialog box.



2 Choose **FILE** > **OPEN INSTALLATION FILE.**

This action opens the *Open Installation File* dialog box.

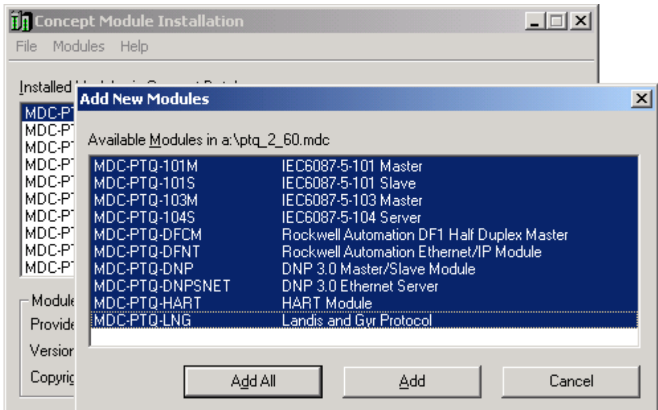


3 If you are using a Quantum processor, you will need the MDC files. In the *Open Installation File* dialog box, navigate to the *MDC Files* directory on the ProTalk CD.

4 Choose the MDC file and help file for your version of Concept:

- Concept 2.6 users: select PTQ_2_60.mdc and PTQMDC.hlp
- Concept 2.5 users: select PTQ_2_50.mdc and PTQMDC.hlp.

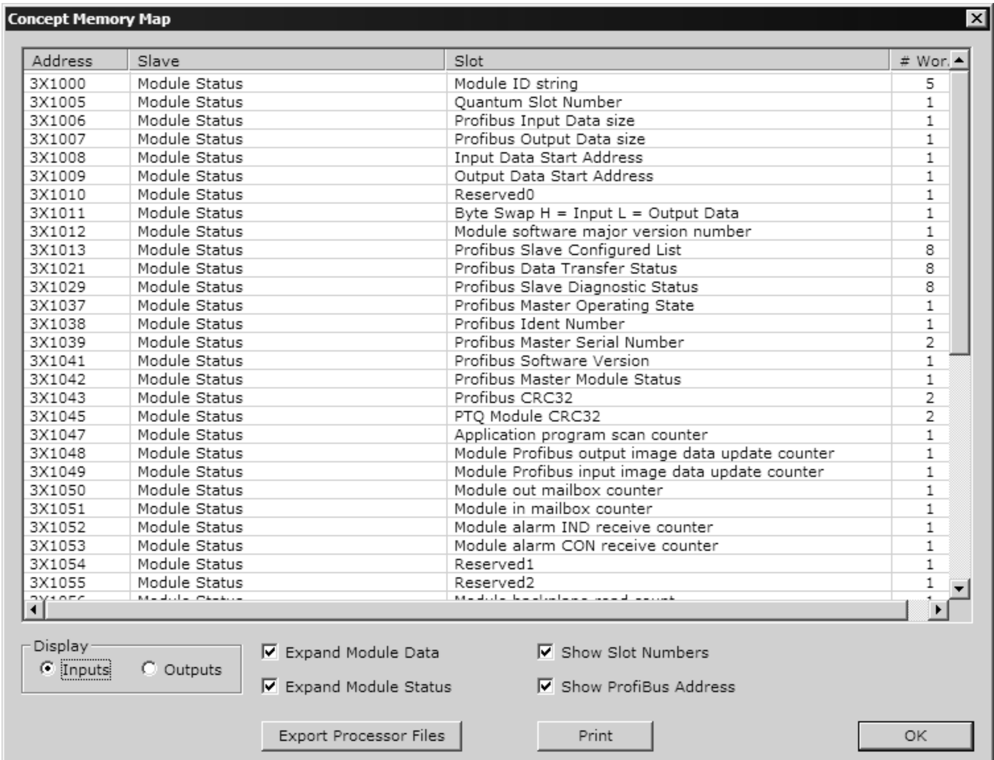
Select the files that go with the Concept version you are using, and then click **OK**. This action opens the *Add New Modules* dialog box.



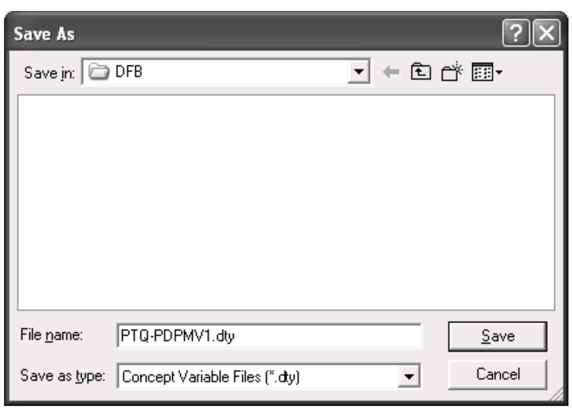
5 Click the **ADD ALL** button. A series of message boxes may appear during this process. Click **YES** or **OK** for each message that appears.
6 When the process is complete, open the **FILE** menu and choose **EXIT** to save your changes.

4.4 Step 1: Exporting the Files from PCB

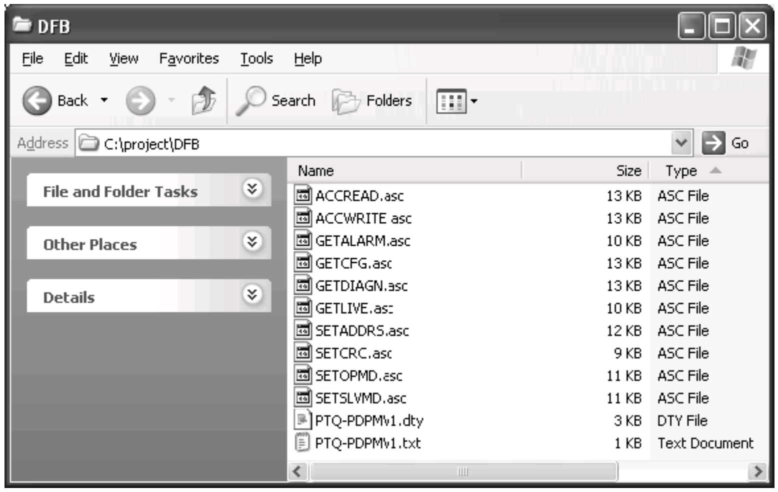
1 In ProSoft Configuration Builder, right-click the **PROFIBUS DP FOLDER**, and then click **CONFIGURE**.
2 Click **SHOW CONCEPT MAP**.
3 Click **EXPORT PROCESSOR FILES**.



4 Browse to the folder *C:\project\DFB* and click **OK**.



5 All the files required for your Concept application will be now located at *C:\project\DFB*.



The following section provides a general overview of the files that were exported.

4.4.1 -.ASC files

Each function block is available in ASCII format. These files can be converted through the Concept Converter tool in order to be used in the Concept project. Refer to Backing Up the Project (page 54) for file locations.

File Name	Description	Required/Optional[1]
ACCREAD.ASC	Acyclic Read Mailbox	Optional
ACCWRITE.ASC	Acyclic Write Mailbox	Optional
GETALARM.ASC	Alarm Mailbox	Optional
GETCFG.ASC	Get Configuration Mailbox	Optional
GETDIAGN.ASC	Get Diagnostics Mailbox	Optional
GETLIVE.ASC	Get Live List Mailbox	Optional
SETADDRS.ASC	Set Slave Address Mailbox	Optional
SETCRC.ASC	Set CRC (not mailbox - used to sync CRCs)	Required
SETOPMD.ASC	Set Operating Mode Mailbox	Optional
SETSLVMD.ASC	Set Slave Mode Mailbox	Optional
SETSLVS.ASC	Start/Stop Slaves Dynamically	Optional
ColdBT.ASC	Remote Coldboot from PLC	Optional

[1]Optional means that you should import this function block only if your application requires that specific mailbox. The status and PROFIBUS data will be available even if no optional mailbox function blocks are imported into your project. The SETCRC.ASC function block is required to synchronize the input and output CRCs (page 128) while configuring the module.

4.4.2 -.DTY file

This file contains the data type definitions that will be required for the Concept project. The default DTY file name will be the same as the PTQ module name in the PCB configuration.

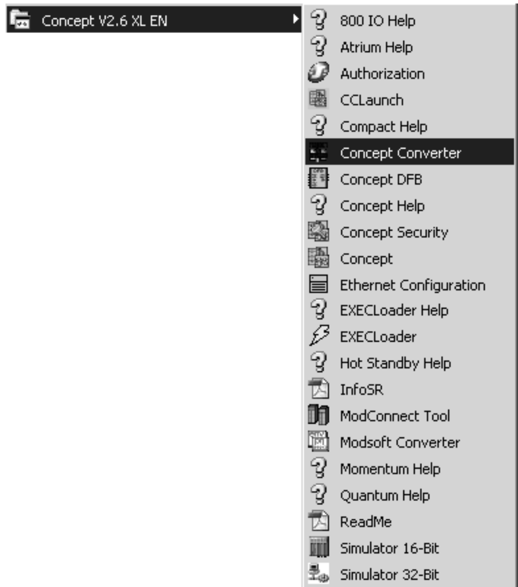
Note: If your application requires multiple PTQ-PDPMV1 modules for the same project, you must merge the .DTY files. Refer to Using Multiple PTQ-PDPMV1 Modules with Concept (page 291) for instructions.

4.4.3 -.TXT file

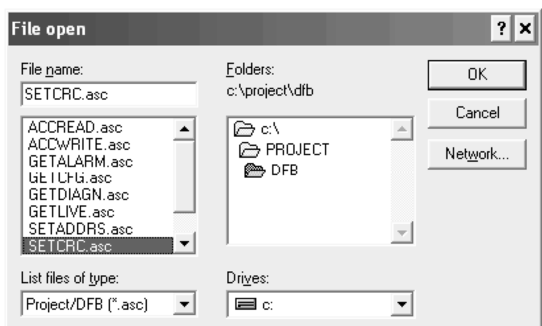
This file contains the variables that will be imported to Concept. The default TXT file name will be the same as the PTQ module name in the PCB configuration.

4.5 Step 2: Converting the Function Blocks

1 Run the *Concept Converter* tool as follows:

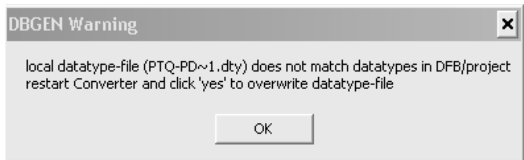


2 Click **FILE-IMPORT.** Browse the *SETCRC.ASC* file at the *C:\project\DFB* folder.



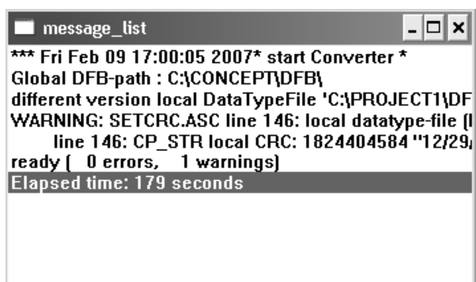
After you click **OK**, the following warning message will be displayed. Click **No**.

3 If you use the same filename as the ones PCB generated (PTQ-PDPMV1_sample) you will see the following message:

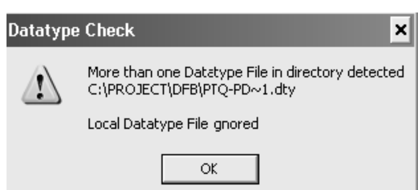


Click **OK** to dismiss the warning message.

Next, you will see the following message box:



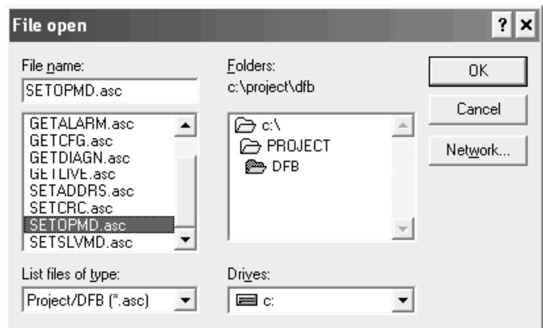
This message does not indicate an error condition, because the ASC files were built with a different DTY file, which was overwritten with the new DTY file exported by PCB. The new DTY file has all I/O information for cyclic data and (optional) slave diagnostics.
If you use a different filename than the one generated by PCB (for example Processline1_Master) you will see the following message when you attempt to import the new ASC file.



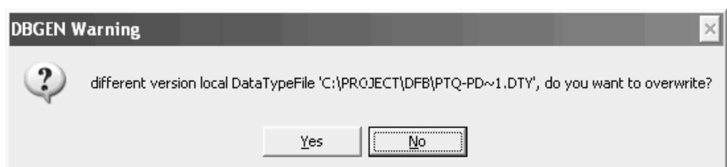
Click **OK** to dismiss the message. After importing all .ASC files, delete the old DTY file (PTQ-PD~1.dty). The old DTY file does not contain any Cyclic I/O data.

4 List all optional mailbox functions that will be required for your application. Repeat steps 2 and 3 for each required mailbox. This setup procedure will consider (as an example) that only the *Set Operating Mode* mailbox is required. So the following steps (5 and 6) will repeat the same procedure for the *Set Operating Mode* mailbox function block:

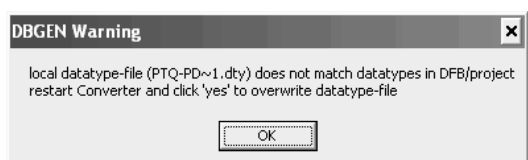
5 Click **FILE-IMPORT**. Browse the *SETOPMD.ASC* file in the *C:\project\DFB* folder.



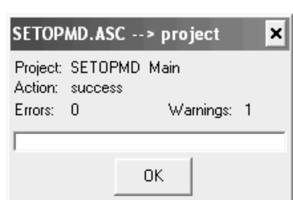
After you click **OK** the following warning message will be displayed. Click **NO**.



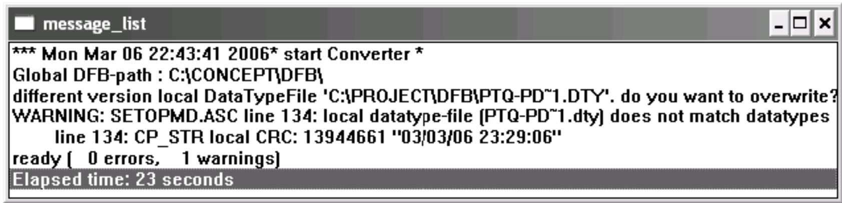
Then the following warning message is displayed. Click **OK**.



6 After the import procedure is concluded the following window is displayed. Click **OK**.



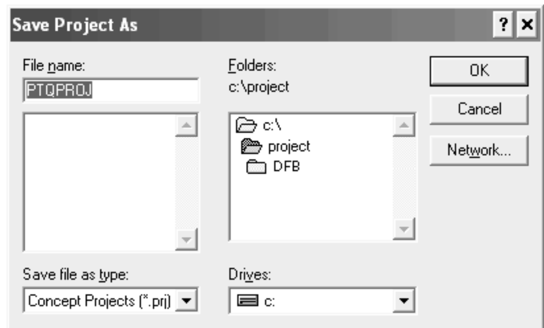
The following warning is displayed (this is expected).



7 At *Concept Converter* click **FILE-EXIT**.

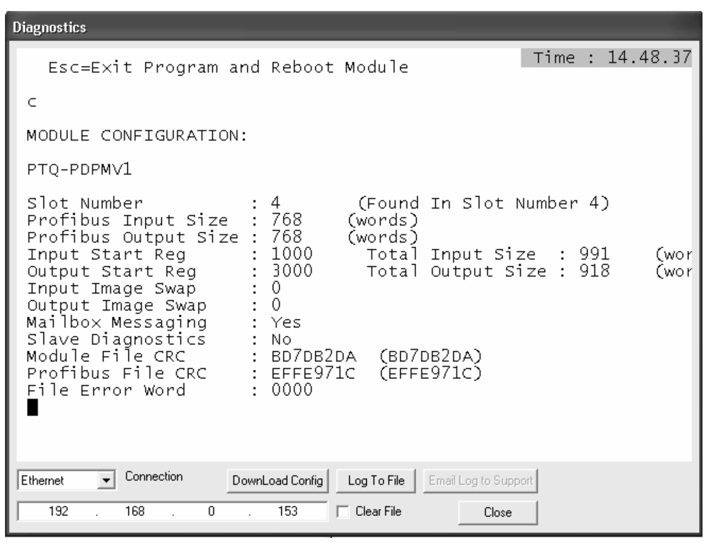
4.6 Step 3: Setting up the Concept Project

1. Start Concept Version 2.6.
2. Open the **FILE** menu, and then choose **NEW PROJECT**.
3. Open the **FILE** menu again, and then click **SAVE PROJECT AS**.
4. Navigate to *C:\project,* and enter PTQPROJ as the file name. Click **OK** to save the file.



5. Configure the general settings for your application. Select the correct Quantum processor type (PLC Selection) and other modules that will be located in the Quantum rack.
6. In PLC Configuration, double-click **PLC MEMORY PARTITION**. Verify that the number of input registers and output registers are sufficient for your application.

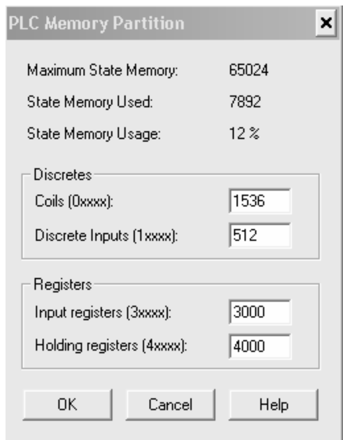
Note: You can view the number of input and output words required for your PTQ-PDPMV1 application in the *Diagnostics* window in ProSoft Configuration Builder. From the module's configuration/debug menu, press **[B]** to open the *Block Transfer Statistics* menu.



Using the example in the illustration, note the following values:

	Total Size	Start Address	Last Address
PROFIBUS Input	991	301000	301990
PROFIBUS Output	918	403000	403917

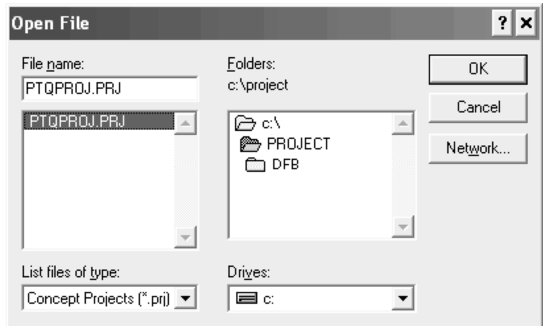
For this example, select 3000 input registers and 4000 holding registers as shown in the following illustration.



Note: Use these values for reference only. The illustration above indicates that you can only use 2000 registers, because the start register is at 1000 and the count is 3000. If your input requires more than 2000 registers, refer to the following paragraph.
Important: You must configure the number of registers required for your application correctly, otherwise the backplane driver will not transfer any data between the processor and the module. Please note that the *Input Data Size* and *Output Data Size* parameters configured in PCB will configure only the number of registers required for PROFIBUS data. However, the module will require more registers for status and mailbox transfer. For this reason, you must verify the total number of registers through the *Diagnostics* window.

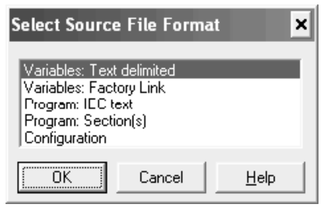
7 In Concept, open the **FILE** menu, and then choose **CLOSE PROJECT**.

8 Open the **FILE** menu again, and then choose **OPEN-PROJECT** to reopen the file that you have just saved. This step allows Concept to recognize the data type definitions and function blocks that are located in *C:\project\DFB*.

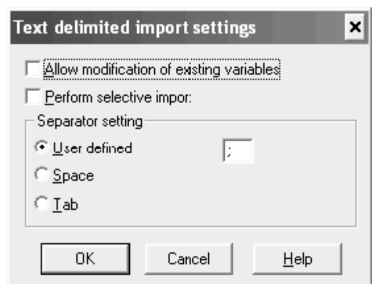


4.7 Step 4: Importing the Variables

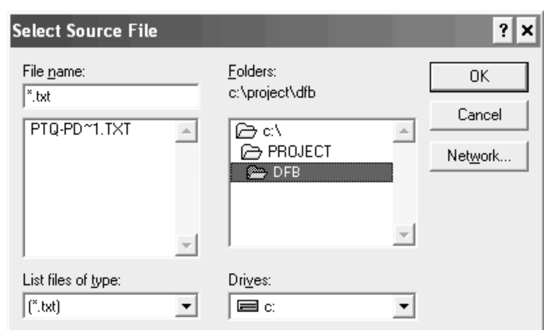
1 In Concept, open the **FILE** menu, and then choose **IMPORT**.
Select **VARIABLES: TEXT DELIMITED** and click **OK**.



2 Select **USER DEFINED**, with **;** as the delimiter, and leave all other options unselected. Click **OK**.



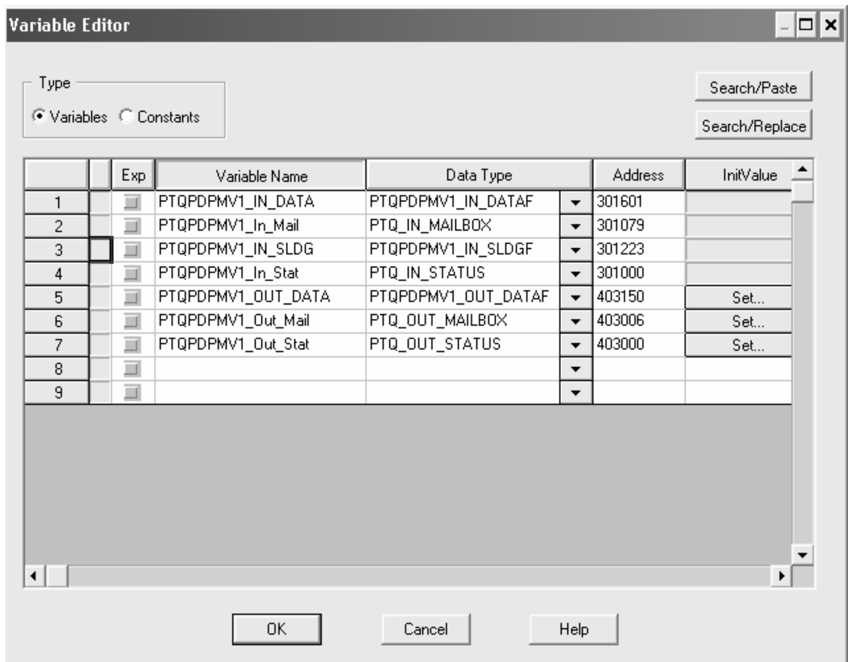
3 Navigate to the .TXT file you exported in Step 1, located in *C:\project\DFB* and then click **OK** to import the variables.



4 When the import procedure is completed, click **OK** to dismiss the *Import Status* message box.

To view the variables that were created during this procedure, open the **PROJECT** menu, and then choose **VARIABLE DECLARATIONS**. The following illustration shows a variable for *Slave Diagnostics (PTQPDPMV1_SLDG*), which is an optional selection. Refer to PTQ Input and Output Data Blocks (page 251) for detailed information on the structure of these blocks and how they are affected by various configuration options.

Note: The memory addresses will match the settings configured through ProSoft Configuration Builder.



The following variables are available for your application.

Variable	Transferred From	Transferred To	Description
PTQPDPMV1_In_Stat	PTQ-PDPMV1	Quantum	Status Data
PTQPDPMV1_In_Mail	PTQ-PDPMV1	Quantum	Mailbox Buffer
PTQPDPMV1_IN_DATA	PTQ-PDPMV1	Quantum	Input PROFIBUS Data
PTQPDPMV1_Out_Stat	Quantum	PTQ-PDPMV1	Status Data
PTQPDPMV1_Out_Mail	Quantum	PTQ-PDPMV1	Mailbox Buffer
PTQPDPMV1_OUT_DATA	Quantum	PTQ-PDPMV1	Output PROFIBUS Data
PDQPDPMV1_IN_SLDG	PTQ-PDPMV1	Quantum	Slave Diagnostic Data

Status: The status data can be used to monitor the status of the module and the PROFIBUS network (input). The function blocks also use the status data for handshaking purposes during the mailbox handling (input and output).

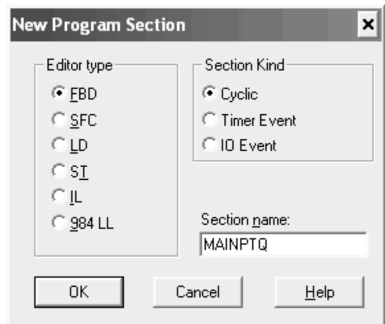
Mailbox Buffer: These variables store the mailbox requests and responses between the processor and the module. These mailbox variables (input and output) must be linked to the *MailIn* and *MailOut* function block pins (as covered later in this document).

PROFIBUS Data: These variables store the input and output PROFIBUS data associated to the PROFIBUS slaves.

Slave Diagnostic Data: Diagnostic for every configured slave on the PROFIBUS network will be passed from the PTQ-PDPMV1 to the Quantum processor.

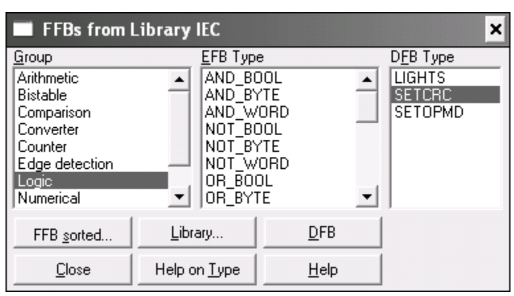
4.8 Step 5: Creating the Function Block Instances

1 In Concept click **PROJECT-PROJECT BROWSER**.
2 In *Project Browser* right-click **PROJECT: PTQPROJ** and click **NEW PROGRAM SECTION.**
3 Configure the *New Program Section* as follows (select the *Editor Type* as **FBD**).

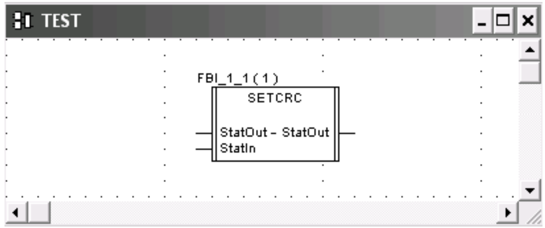


4 Double-click the **FBD** section you have just created.

5 Click **OBJECTS-FFB SELECTION**. Click the **DFB** button and select the **SETCRC** function block. Click the **CLOSE** button to confirm.

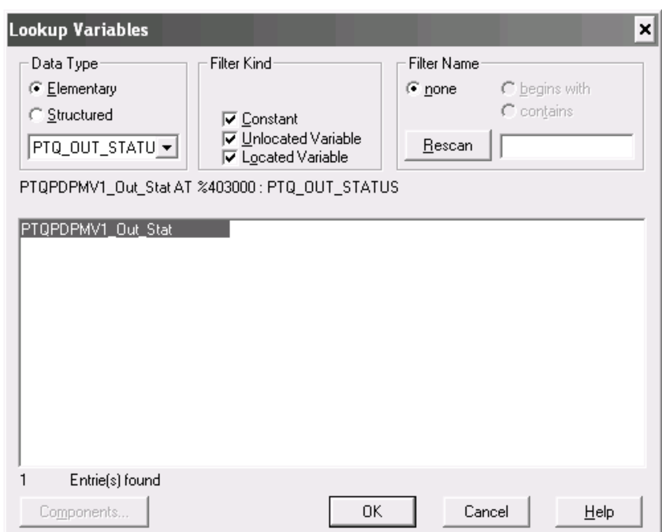


Insert the *SETCRC* function block in the *MAINPTQ* section.

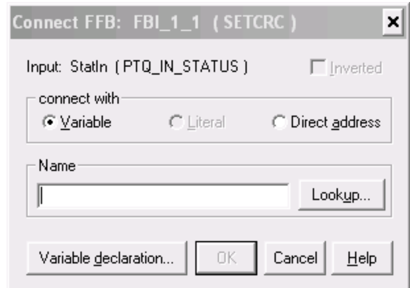


Select the imported variables to be associated with the input and output pins. The input status and output status variables must be the same ones that were previously imported by PCB.
Double-click the **STATOUT** pin and click **LOOKUP**.

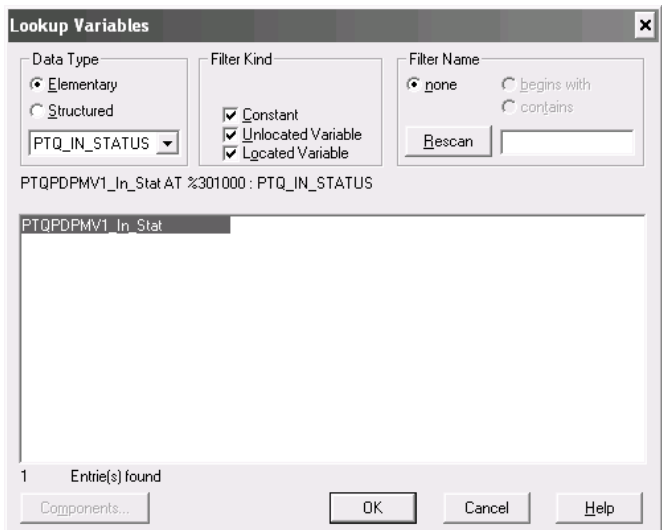
Select the existing variable that was previously imported to the project:



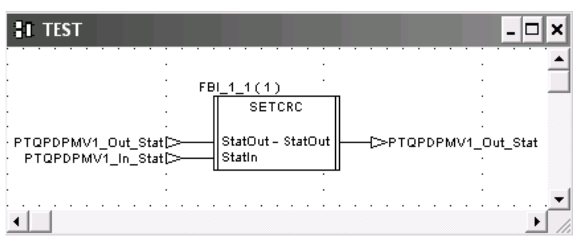
Double-click the **STATIN** pin and select **LOOKUP**.



Select the variable that was previously imported to the project.

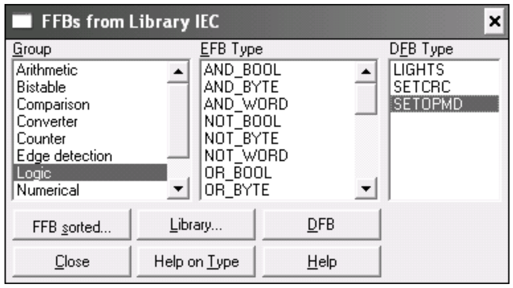


The function block is now ready:

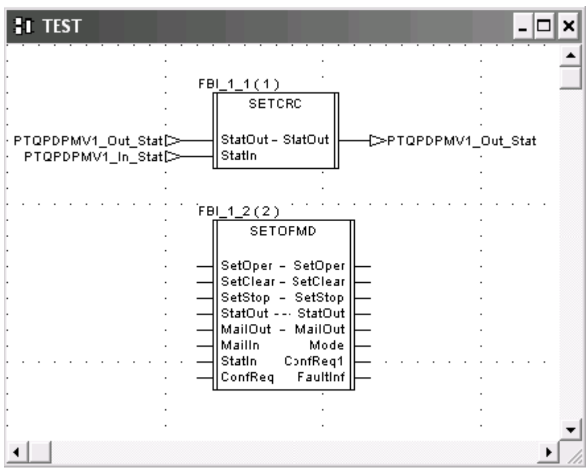


6 Now you should repeat steps 5 through 7 for every mailbox function block to be used by the application. This setup procedure considers only the *SETOPMD* (*Set Operating Mode* function block) for the next 3 steps.

7 Select **OBJECTS-FFB SELECTION**. Click the **DFB** button and select the *SETOPMD* function block. Click at the **CLOSE** button to confirm.



8 Insert the *SETOPMD* function block in the *MAINPTQ* section.

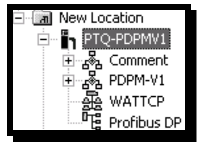


Associate the variables required for the input and output pins for this function block.

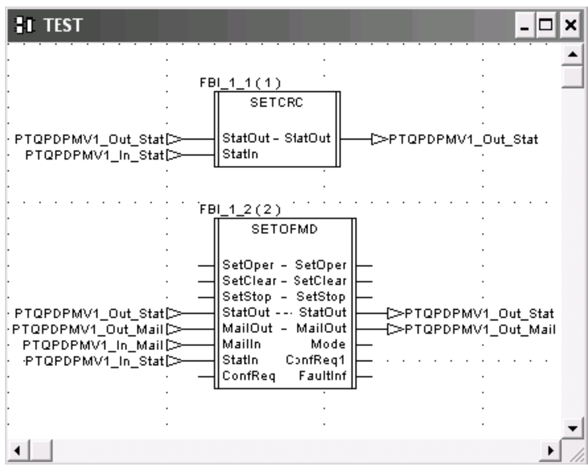
IMPORTANT: The following pins must be associated to the variables previously imported. The user cannot associate these pins to any other variables. This important step must be followed for all function blocks.

PIN	Type	Default Variable Name[1]	Data Type
StatOut	Input/output	PTQPDPMV1_Out_Stat	PTQ_OUT_STATUS
MailOut	Input/output	PTQPDPMV1_Out_Mail	PTQ_OUT_MAILBOX
StatIn	Input	PTQPDPMV1_In_Stat	PTQ_IN_STATUS
MailIn	Input	PTQPDPMV1_In_Mail	PTQ_IN_MAILBOX

The actual variable name will depend on the module name configured in PCB. The default module name is PTQPDPMV1. However, the data types used by these variables will always have a fixed name as shown in the table above.
Default file name (PTQ-PDPMV1):

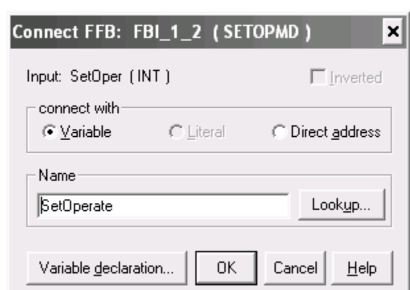


9 It is suggested to initially associate these variables to the correct pins before creating any variables for the other pins (as follows):

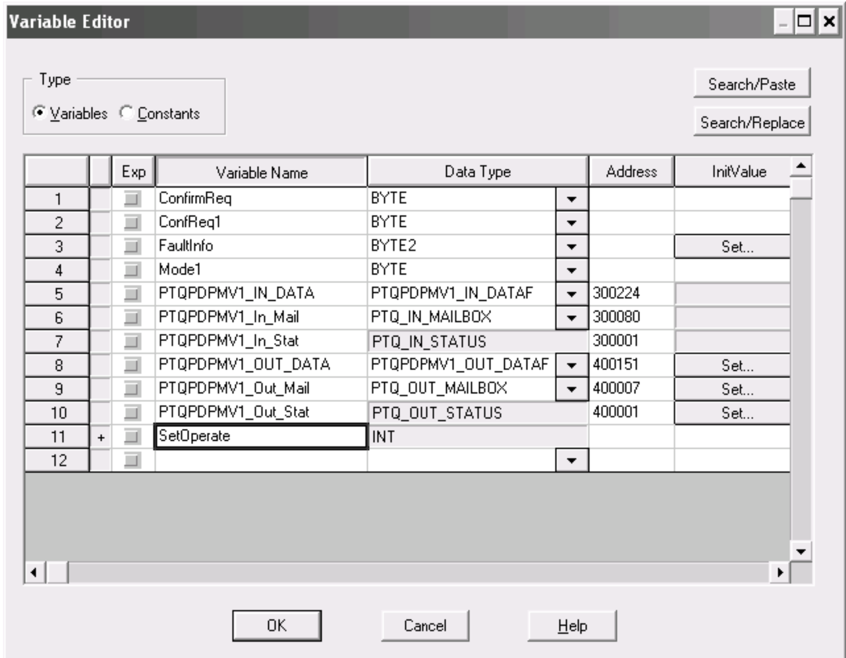


Now you can create other variables and associate these to the rest of the pins. For example, to associate a variable to the *SetOper* input/output pin follow the steps below.

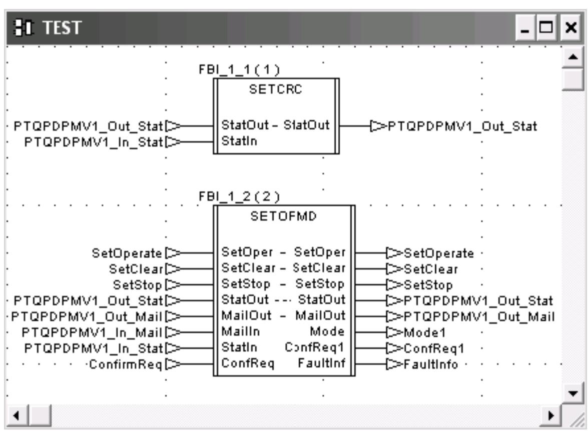
a) Double-click the **SETOPER** pin.
b) Choose a variable name (this example uses *SetOperate*) and click **OK**.



c) Click **OK** to confirm the variable creation.

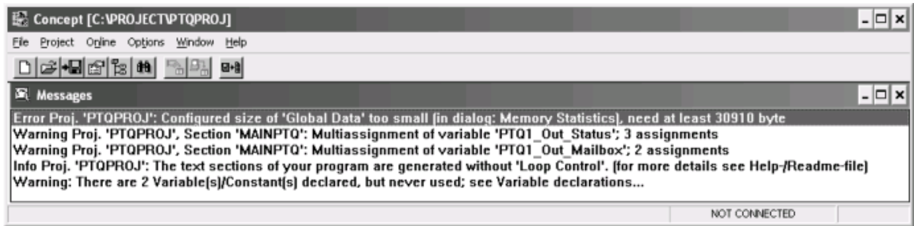


10 Repeat the same procedure for all other pins until the function block configuration is completed.



11 Save the Concept Project (**FILE-SAVE PROJECT**)

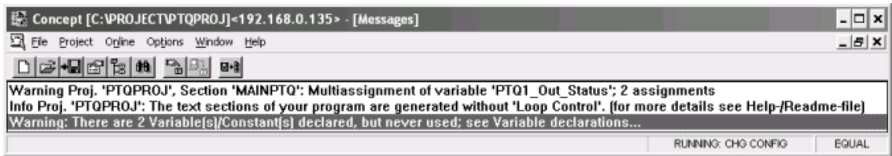
Note: While the project is being analyzed (depending on the number of mailbox function blocks used), the following error message might be generated:



The error message means that you must increase the size of the global output data for your project. You can select Project-PLC Configuration-PLC Selection to increase the size of the global output data.

4.9 Step 6: Downloading the Concept Project

Download the project to the Quantum processor (Online Connect and Online-Download). Once the download operation is concluded, there will be a few warning messages generated in Concept. The warnings, indicating that some input/output variables are being used by more than one function block, can be safely disregarded.



4.10 Using the Concept Project

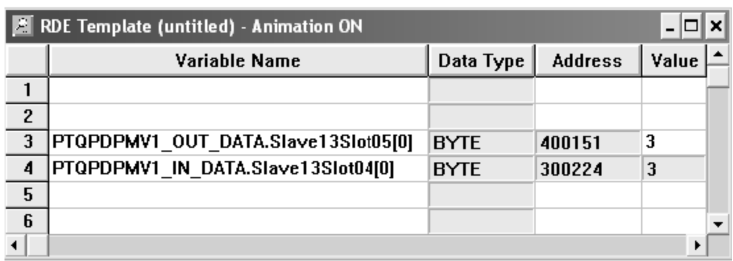
4.10.1 Accessing PROFIBUS Data

After the module's CRC values are synchronized (through the SETCRC function block) then no other function blocks are required for the PROFIBUS input/output data exchange. You can refer to the variables that were imported to the Concept project and use either the *PTQPDPMV1_OUT_DATAF* or *PTQPDPMV1_IN_DATAF* data types. These variables contain a structure of sub variables that will store the data associated to all slaves configured at ProSoft Configuration Builder.

The following illustration shows an example in which two variables are used to store the PROFIBUS input and output data.

PTQPDPMV1_OUT_DATA.Slave13Slot05[0]: Stores output byte 0 of slot 5 from the slave (PROFIBUS address 13)

PTQPDPMV1_IN_DATA.Slave13Slot04[0]: Stores input byte 0 of slot 4 from the slave (PROFIBUS address 13)



	Variable Name	Data Type	Address	Value
1				
2				
3	PTQPDPMV1_OUT_DATA.Slave13Slot05[0]	BYTE	400151	3
4	PTQPDPMV1_IN_DATA.Slave13Slot04[0]	BYTE	300224	3
5				
6				

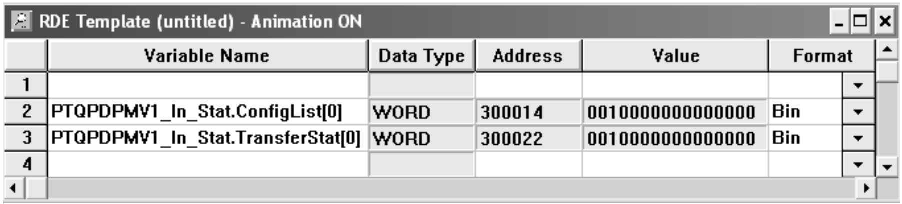
Note1: To verify that the CRC values are synchronized, look at the CFG ERR LED (OFF: CRCs are synchronized, ON: CRCs are not synchronized).
Note2: The PTQPDPMV1_IN_DATA variable is the default variable name. The actual name for your application might vary depending on the module name that you selected in PCB.

4.10.2 Accessing Status Data

The module constantly updates the status data to the processor. The status data provides general information about the module, PROFIBUS slaves and backplane status. It is automatic (no function blocks are required). Refer to Status Data in the Input Data Block (page 256) for more information.

Refer to the imported variables that use the PTQPDPMV1_IN_STATUS data type for the status data.

The following illustration shows an example of two registers that indicate whether the first 16 devices are currently configured (PTQPDPMV1_In_Stat.ConfigList[0]), and if these devices are in data exchange mode (PTQPDPMV1_In_Stat.TransferStat[0]). The following illustration shows an example in which slave 13 is configured and in data exchange mode.


RDE Template (untitled) - Animation ON

	Variable Name	Data Type	Address	Value	Format
1					
2	PTQPDPMV1_In_Stat.ConfigList[0]	WORD	300014	0010000000000000	Bin
3	PTQPDPMV1_In_Stat.TransferStat[0]	WORD	300022	0010000000000000	Bin
4					

Note: The actual variable name will depend on the project name you have selected in PCB (PTQPDPMV1 is default).

4.10.3 Configuration Validation & SETCRC Function Block

The configuration validation functionality prevents the module from causing unexpected results after it receives a new configuration (for example, if it receives a configuration that should have been downloaded to a different module). The PTQ-PDPMV1 module constantly transfers two CRC values to the processor (as part of the input status data) which are calculated based on its current configuration. The processor must copy back the same CRC values to the module (as part of the output status data). If the CRC values are not synchronized, the module will be switched to STOP mode and the CFG ERR LED will be illuminated. If the CFG ERR LED is OFF it means that the CRC values are synchronized. While the module is in STOP mode, there will be no data exchange with the configured PROFIBUS slaves.

The SETCRC function block is presented in this procedure for convenience purposes to get your PTQ-PDPMV1 module up and running. The SETCRC function block automatically updates the CRC through the following lines of structured text code:

StatOut.PROFIBUSCRC :=StatIn.PROFIBUSCRC;
StatOut.ModuleCRC :=StatIn.ModuleCRC;

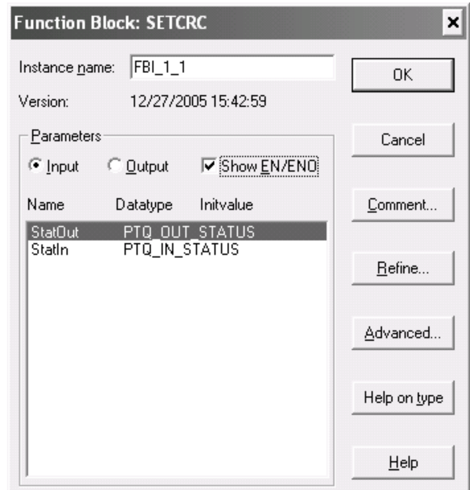
This procedure suggests the use of the SETCRC function block in order to avoid the module being set to STOP mode during successive configuration changes (these changes typically occur during the initial setup steps). However, please note that the SETCRC function block also prevents the module from supporting the configuration validation functionality, because the CRC values will always be synchronized even if an unexpected download occurs.

After the module is configured and the CRC values are synchronized, the SETCRC function block can be disabled (for applications that require configuration validation).

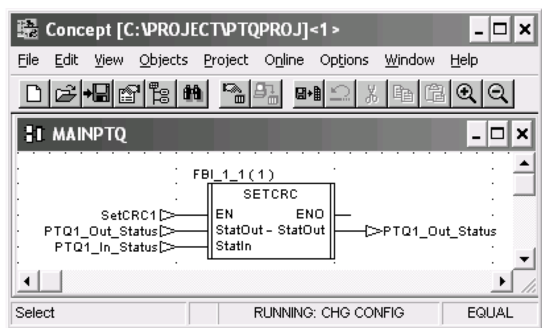
To disable the SETCRC function block, follow these steps:

1 Double-click the **SETCRC FUNCTION BLOCK** at the FBD section

2 Select the **SHOW EN/ENO** checkbox



Create a *BOOL* variable and associate it to the *EN* input pin of the *SETCRC* function block. If this variable has a value of 0 (OFF) the *SETCRC* function block will be disabled. Therefore, further changes to the module configuration would cause the module to be switched to STOP mode.



Note: If you use the EN bit to disable the function block, please remember that after a processor download, the output variables might be reset and may cause a CRC mismatch. After a processor download, you may need to re-enable the SETCRC function block once, to synchronize the CRCs again.

4.11 Using Mailbox Function Blocks

4.11.1 Overview

The mailbox function blocks build mailbox requests to the module and read the mailbox response from the module. These mailbox function blocks are optional, meaning that the project will update PROFIBUS data and status information even if no function blocks are used.

4.11.2 Configuration

The mailbox function block contains input, outputs and input/output pins that must be associated to specific variables.

The mailbox function blocks (except *Get Alarm*, which will be covered later) require the usage of the following pins (common for all mailbox function blocks):

Pin Name	Pin Type	Description
StatIn	Input	Input Status pin. Must be associated to the imported variable PTQPDPMV1_In_Stat. It contains the status transferred from the module allowing the mailbox function block to receive the acknowledgment that the mailbox request was processed by the module. It is used also to check if a new mailbox response is available. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
MailIn	Input	Input Mailbox pin. Must be associated to the imported variable PTQPDPMV1_In_Mail. It contains the mailbox response message that is handled by the function block according to its mailbox ID. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
StatOut	Input/Output	Output Status pin. Must be associated to the imported variable PTQPDPMV1_Out_Stat. It is used to check if a new mailbox response is available. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
MailOut	Input/Output	Output Mailbox pin. Must be associated to the imported variable PTQPDPMV1_Out_Mail. This variable stores the mailbox output variable that is updated from the function block when a new mailbox request is performed to the module. It consists of an array of words. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
"Trigger"	Input/Output	Move a value of 1 to this register to initiate the mailbox request. A request can only be initiated if its current value is 0 and all triggers for the other mailbox function blocks also have a value of 0. The actual name for this trigger register will be specific for each mailbox function block. For example, for the SetOperate mailbox (SETOPMD) the trigger register is *SetOper*.

[1]The actual variable name will depend on the module name configured by the user at PCB. These data type names are considering the default module name (PTQPDPMV1).

4.11.3 Trigger Register

The mailbox requests are initiated by the "trigger" register (INT) that is defined as an input/output pin. The mailbox request is initiated after the application moves a value of 1 to the trigger register.

The current value of all mailbox triggers for your application must be equal to 0 in order to perform a mailbox request. If you are using more than one mailbox function block, you must add program code to guarantee that this condition is satisfied. Therefore, a mailbox function block is only allowed to send a new mailbox request after the processor receives confirmation that the previous mailbox request was acknowledged by the module. This condition is determined by checking the status of all trigger registers (0=OK). It also prevents an attempt to send more than one mailbox request in a single PLC scan.

The following table shows the trigger registers used for each mailbox function block:

File Name	Description	Trigger
ACCREAD.ASC	Acyclic Read Mailbox	AcRead
ACCWRITE.ASC	Acyclic Write Mailbox	AcWrite
GETALARM.ASC	Alarm Mailbox	-
GETCFG.ASC	Get Configuration Mailbox	GetCnfg
GETDIAGN.ASC	Get Diagnostics Mailbox	GetDiag
GETLIVE.ASC	Get Live List Mailbox	GetList
SETADDRS.ASC	Set Slave Address Mailbox	SetAddr
SETOPMD.ASC	Set Operating Mode Mailbox	SetOper, SetStop, SetClear
SETSLVMD.ASC	Set Slave Mode Mailbox	SetMdSlv
SETSLVS.ASC	Start/Stop Slaves Dynamically	Start, Stop
ColdBT.ASC	Remote Coldboot from PLC	Cldboot

The *Get Alarm* function block does not require a trigger because this mailbox is initiated from the module (as covered later in this User Manual).

The trigger register is a variable that can assume different states as follows:

Value	Description
0	OK to send new mailbox request. The last mailbox request was already acknowledged by the module.
1	Mailbox request to be performed. The Quantum program should make sure that the required conditions are satisfied (as previously discussed) before moving a value of 1 to the trigger register. The function block will then build the mailbox request by copying all mailbox input parameters to the mailbox output variable that is transferred to the module through the backplane. Then the function block will automatically change the trigger's value to 2.
2	Processor has performed the mailbox request and is waiting for the acknowledgment from the module. The acknowledgment informs that the module has received the request (the actual mailbox response is actually sent later). After the acknowledgment is received, the function block will reset the trigger's value back to 0.

4.11.4 Specific Input Pins

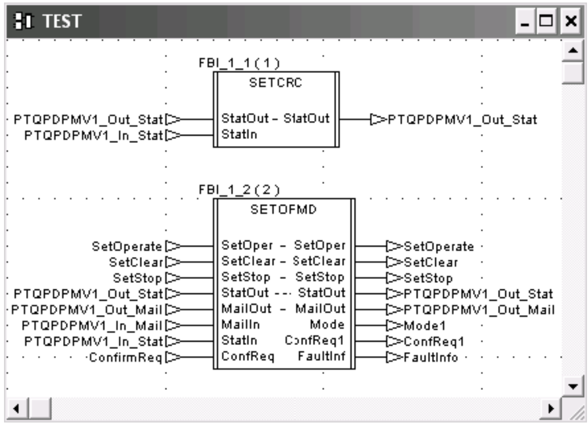
Each function block has input pins specifically for each mailbox. For example, in order to send a *Get Diagnostics* mailbox, the application must set the PROFIBUS slave address input pin. The processor program must configure the input pins before performing the mailbox request, by moving a value of 1 to the mailbox trigger. For a description of each function block input pin, you can double-click the function block instance and select **ADVANCED** for the comment about each input pin. Also refer to the module documentation for detail information about each mailbox parameter.

4.11.5 Specific Output Pins

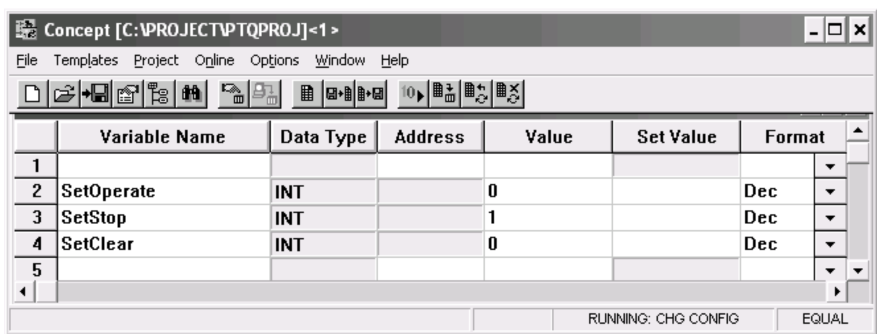
Each function block contains output pins that are updated after the mailbox response is received by the processor. For example, the *Get Diagnostics* function block has an *ExtDiag* output pin that stores the diagnostic information received from the slave.

Example

If the *Set Operating Mode* mailbox function block is used as follows:



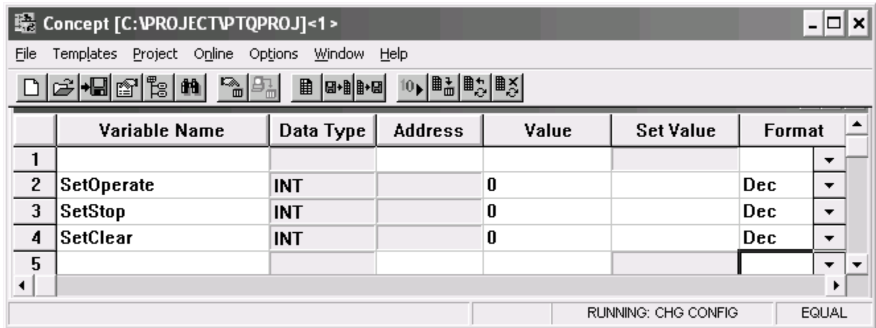
You can create a Reference Data Editor table with three trigger variables for the *Set Operating Mode* mailbox. Start by moving a value of 1 to *SetStop* in order to set the module's mode to STOP.



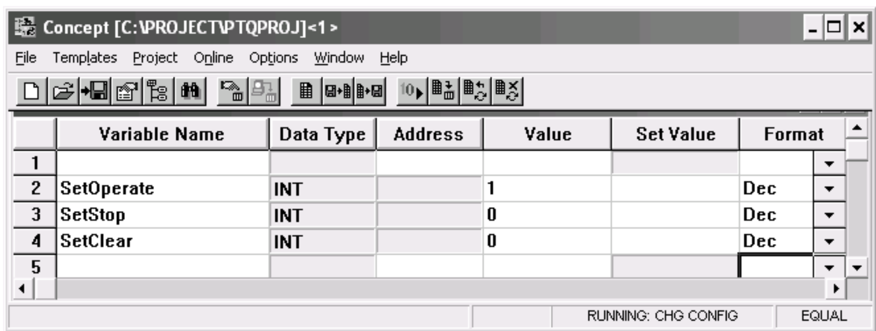
At this point, you should notice the following LED display, indicating that the module's mode was changed to STOP:

LED	Status
MSTR STAT	RED
COM STAT	OFF
DBASE STAT	GREEN
TK HOLD	GREEN

You will also notice that the function block automatically clears the trigger register after it receives the acknowledgment from the module.



Move a value of 1 to the *SetOperate* trigger variable.

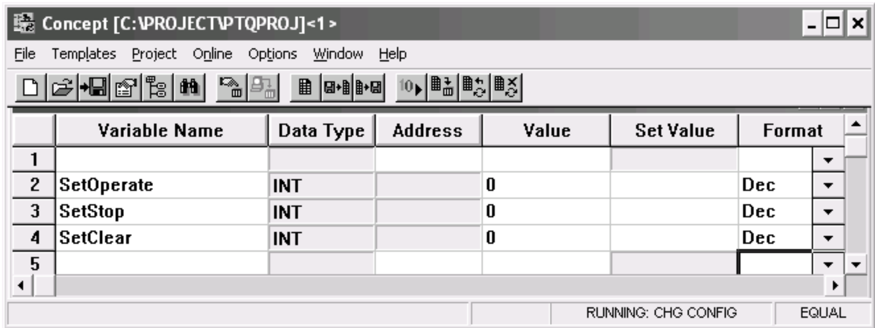


At this point, you should notice the following LED display, indicating that the module's mode was changed to OPERATE:

LED	**Status**
MSTR STAT	GREEN
COM STAT	GREEN or OFF[1]
DBASE STAT	GREEN
TK HOLD	GREEN

[1]The COM STAT LED will be either GREEN if the Master is communicating with all slaves, blinking if it is communicating with some of the slaves, or OFF if it is not communicating with any slaves.

You will also notice that the function block automatically clears the trigger register after it receives the acknowledgment from the module.



	Variable Name	Data Type	Address	Value	Set Value	Format
1						
2	SetOperate	INT		0		Dec
3	SetStop	INT		0		Dec
4	SetClear	INT		0		Dec
5						

4.12 Mailbox Overview

This section provides a brief description on how to use each mailbox function block. Refer to Mailbox Messaging (page 151) for detailed information about each mailbox parameter:

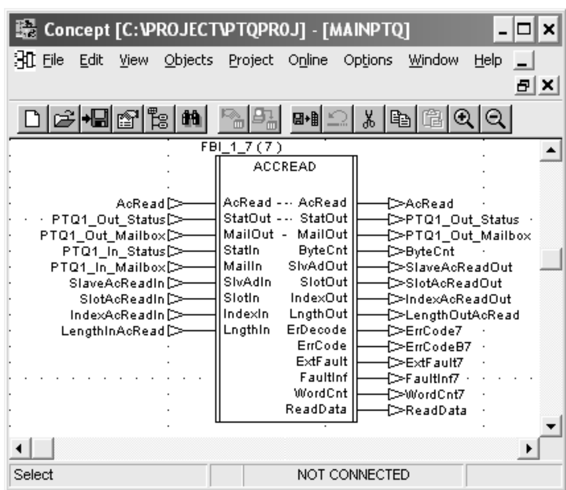
4.12.1 Acyclic Read Mailbox

Function Block: ACCREAD

Trigger Register: AcRead

Description: The ACCREAD mailbox is used to perform an Acyclic Read request to a PROFIBUS slave device. The input pins SlvAdIn (PROFIBUS slave address), SlotIn (slot number), IndexIn (index number), and LngthIn (length - number of bytes associated to acyclic read operation) must be configured before triggering the mailbox request. The acyclic read response data is copied to the ReadData output pin. The status information is available in the output pins (ErrCode, ErrDecode, ExtFault, and FaultInf).

The following illustration shows a sample instance of the *Acyclic Read* mailbox function block:



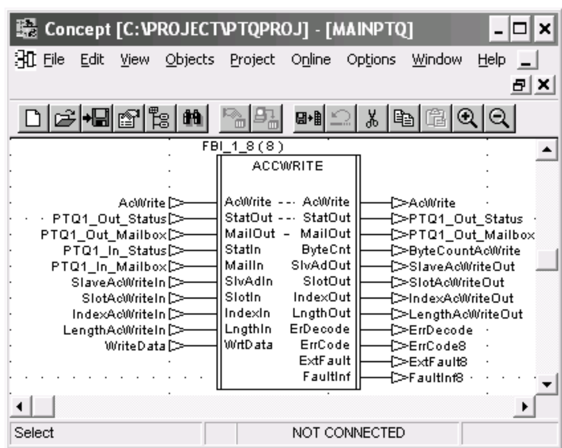
4.12.2 Acyclic Write Mailbox

Function Block: ACCWRITE

Trigger Register: AcWrite

Description: The ACCWRITE function block is used to perform an Acyclic Write request to a PROFIBUS slave device. The input pins SlvAdIn (PROFIBUS slave address), SlotIn (slot number), IndexIn (index number), and LngthIn (length - number of bytes associated to acyclic read operation) must be configured before triggering the mailbox request. The actual data to be written to the PROFIBUS slave should be associated to the WrtData input pin. The status information is available at the output pins (ErrCode, ErrDecode, ExtFault, and FaultInf).

The following illustration shows a sample instance of the *Acyclic Write* mailbox function block:



4.12.3 Alarm Mailbox

Function Block: *GETALARM*

Trigger Register: The *GETALARM* function block does not require a trigger because this mailbox is initiated from the module.

Description: The *GETALARM* mailbox is used to read the alarm mailbox messages sent by the module. The module will automatically generate the alarm mailboxes after it receives the alarm message from the PROFIBUS slave. Therefore, no triggers are required for this mailbox.

The last alarm received is copied at the *LastAlarm* output pin. This is a data structure that contains all alarm information:

```
ALARMTYPE definition:
   SlaveAddress: BYTE;
   SlotNumber: BYTE;
   SeqNumber: BYTE;
   SpecAck: BYTE;
   AlarmType: BYTE;
   ExtDiag: BYTE;
   FaultInfo: ARRAY[0..1] OF BYTE;
   ByteCount: BYTE;
   Data: ARRAY[0..127] OF WORD;
```

This function block also keeps track of the last 20 alarms through the *HistAlarm* output pin.

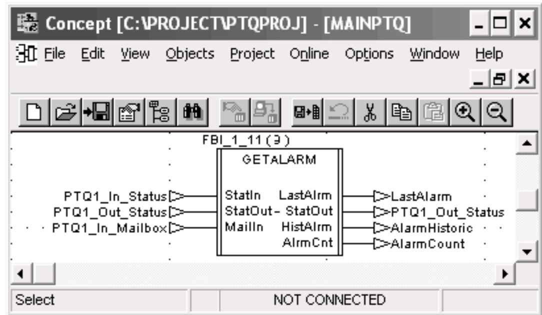
For example, if the module receives 10 alarms (first alarm - Sequence Number = 1, second alarm - Sequence Number = 2, and so on). After the alarm #10 is received, the processor application could refer to these alarms stored at the following output pins:

```
Last Alarm - Alarm #10
HistAlarm[1] - Alarm # 9
HistAlarm[2] - Alarm # 8
HistAlarm[3] - Alarm # 7
HistAlarm[4] - Alarm # 6
HistAlarm[5] - Alarm # 5
HistAlarm[6] - Alarm # 4
HistAlarm[7] - Alarm # 3
HistAlarm[8] - Alarm # 3
HistAlarm[9] - Alarm # 2
HistAlarm[10] - Alarm # 1
```

If the *HistAlarm* buffer is full and it receives a new alarm then the oldest alarm in the queue will be deleted to reserve space for the new alarm.

The *AlarmCnt* output pin in incremented every time the alarm mailbox is received. This register will roll over at 30000. The processor application can keep track of this register to determine when the processor has received a new alarm mailbox message from the module.

The following illustration shows a sample instance of the *GetAlarm* mailbox function block:

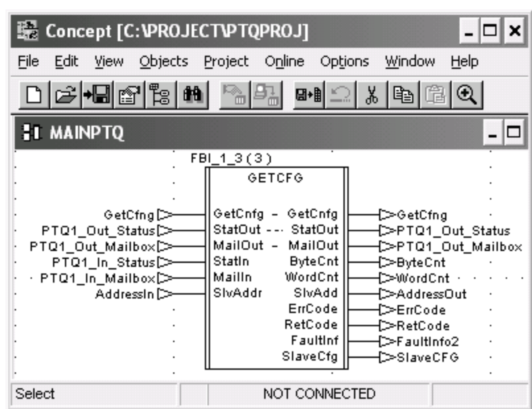


4.12.4 GetConfiguration Mailbox

Function Block: *GETCFG*

Trigger Register: *GetCnfg*

Description: The *GETCFG* function block can be used to read the configuration of any PROFIBUS slave connected to the PTQ-PDPMV1 module. The *SlvAddr* input pin must be configured with the PROFIBUS slave address of the PROFIBUS device. The configuration data is stored at the *SlaveCfg* output pin. The byte count of the slave configuration is stored at *ByteCnt* output pin. The *ErrCode, RetCode*, and *FaultInf* output pins can be used for status verification.

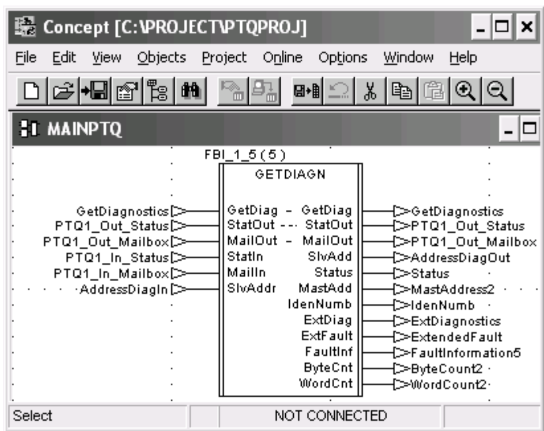


4.12.5 GetDiagnostics Mailbox

Function Block: *GETDIAGN*

Trigger Register: *GetDiag*

Description: The *GETDIAG* function block can be used to read the diagnostics from any PROFIBUS slave connected to the PTQ-PDPMV1 module. The slave address must be set at the *SlvAddr* input pin. The diagnostics data is copied at the *ExtDiag* output pin. The number of bytes of the diagnostics message is stored at the *ByteCnt* output pin (status+identification+extended diagnostics). The *ExtFault* and *FaultInf* output pins can be used for status information. The Master address is stored at the *MastAdd* output pin.



4.12.6 GetLiveList Mailbox

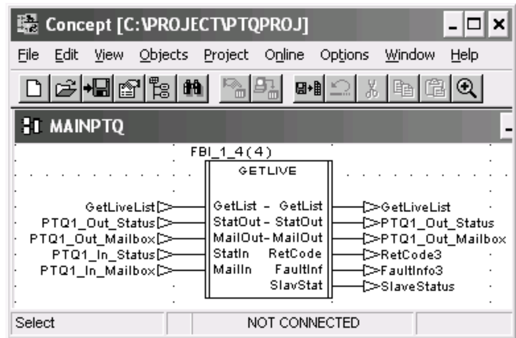
Function Block: *GETLIVE*

Trigger Register: *GetList*

Description: The *GETLIVE* function block can be used to read the live list from the module containing the status of each device at the PROFIBUS network. The live list is stored at the *SlavStat* output pin. The live list data is an array of bytes stored as follows:

```
SlaveStat[0] - status of device configured with PROFIBUS address 0
SlaveStat[1] - status of device configured with PROFIBUS address 1
SlaveStat[2] - status of device configured with PROFIBUS address 2
SlaveStat[3] - status of device configured with PROFIBUS address 3
SlaveStat[4] - status of device configured with PROFIBUS address 4
Etc…
```

Refer to the User Manual for more information about the valid status codes. The *RetCode* and *FaulInf* output pins can be used for mailbox status information.

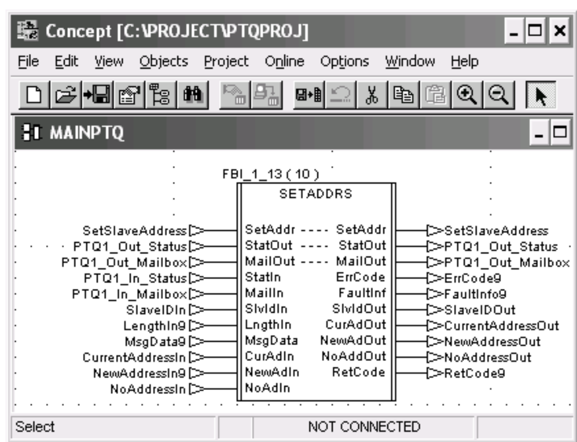


4.12.7 SetSlaveAddress Mailbox

Function Block: *SETADDRS*

Trigger Register: *SetAddr*

Description: The *SETADDRS* mailbox can be used to change the slave address. Only specific PROFIBUS devices support this feature. The application must set the *CurAdIn* (current address) and *NewAdIn* (new address) input pins. It is also possible to deliver user data through the *MsgData* input pin (the number of bytes must be set through the *LgnthIn* input pin). The *SlvIdIn* input pin must be set with the *Ident* number for the slave. The *FaultInf* output pin can be check for mailbox status information.



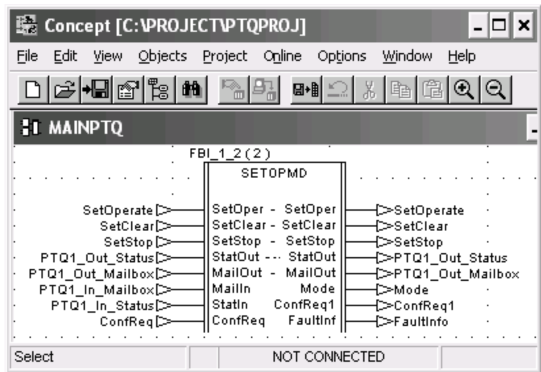
4.12.8 SetOperatingMode Mailbox

Function Block: *SETOPMD*

Trigger Register: *SetOper, SetStop, SetClear*

Description: The following trigger values can be used to change the current operating mode of the module:

```
SetOper = Set Operate
SetStop = Set Stop
SetClear = Set Clear
```

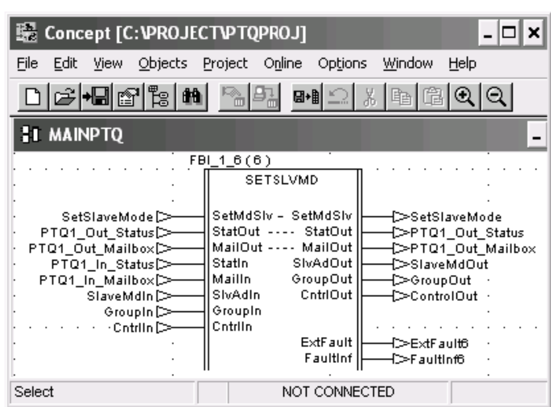


4.12.9 SetSlaveMode Mailbox

Function Block: *SETSLVMD*

Trigger Register: *SetMdSlv*

Description: The *SETSLVMD* function block can be used to request the module to sync, unsync, freeze, or unfreeze. The slave address must be selected through the *SlvAddIn* input pin. If the operation is directed to a group of slaves then the group number must be set through the *GroupIn* input pin parameter. The actual code that will select the operation type must be configured through the *CntrlIn* input pin. Refer to the User Manual for the valid control codes.

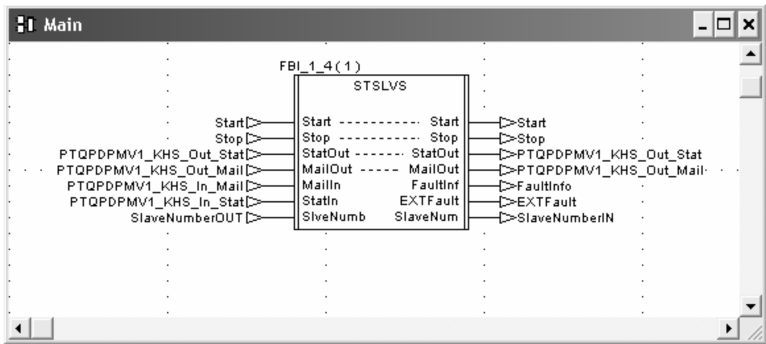


4.12.10 Start/Stop Slaves Mailbox

Function Block: *STSLVS*

Trigger Byte: *Start / Stop*

Description: The *STSLVS* function block can be used to request the module to start/stop certain slaves dynamically. The slave address must be selected through the *SlveNumb* input pin. This is an array of 126 slaves. Change the value for a specific slave from 0 to 1 to stop communication with the Master.



The following illustration shows that when you execute the mailbox command, Slave #4 and Slave #13 will start/stop communicating with the Master.

RDE Template (NEWTEST.RDE) - Animation OFF

	Variable Name	Data Type	Address	Value	Set Value	Format
1	Start	BYTE		0	0	Dec
2	Stop	BYTE		0	0	Dec
3						
4	SlaveNumberOUT[4]	BYTE		1	0	Dec
5	SlaveNumberOUT[13]	BYTE		1	0	Dec
6						
7						

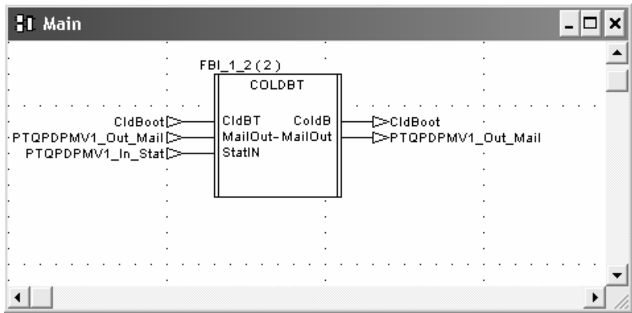
4.12.11 Coldboot Mailbox

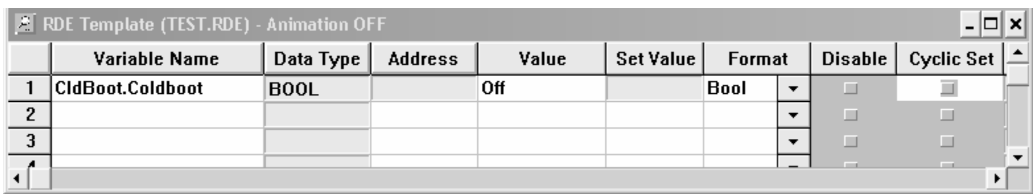
Important: The *Coldboot* mailbox is only supported on PTQ-PDPMV1 modules running firmware version 1.19 or newer. Earlier versions of the firmware do not support this feature. If you require this functionality, please contact ProSoft Technical Services for information on how to upgrade your module.

Function Block: *COLDBOOT*

Trigger Byte: *ColdBoot*

Description: The *COLDBOOT* function block allows you to remotely reboot the module. To trigger a reboot of the module, change the value of the *Cldboot* bit from OFF (zero) to ON (one). The bit is reset back to OFF when the function is executed.





HSBY Note: This function block will reset both the local (active) Master and the remote (passive) Master.

Using a Control Word to Reboot the Module

If you need to cold boot the module from the processor without using the *Coldboot* mailbox, use control word zero of the output image.

If mailbox messaging is **enabled** in the configuration:

- Use the *PTQPDPMV1_OUT_Stat.LastinMailID[0]* control word variable to cold boot the module.
- Entering the cold boot command, as shown below, causes word zero (*Last in Mailbox Message ID*) of the output image to be used for control, instead of for routine mailbox handshaking.

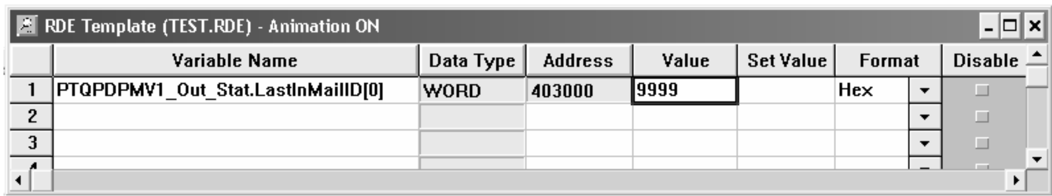
If mailbox messaging is **disabled** in the configuration:

- Word zero in the output image is used for control of the operating mode and is named *Set Operating Mode*.
- Use the *PTQPDPMV1_OUT_Stat.Setoperatingmode[0]* control word variable to cold boot the module.

Note: If mailbox messaging was originally enabled when the DTY file was exported from PCB and imported to the processor, the control word variable *PTQPDPMV1_OUT_Stat.LastinMailID[0]* will still be present, even if mailbox messaging has since been disabled. In this case, use *PTQPDPMV1_OUT_Stat.LastinMailID[0]* to cold boot the module. Entering the cold boot command causes word zero of the output image to be used for control and not for mailbox handshaking, whether or not mailbox messaging is enabled in the configuration.

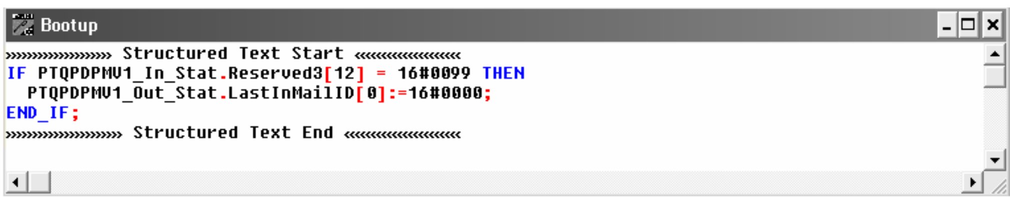
To reboot the module

1 Enter the hexadecimal value **16#9999** in the *PTQPDPMV1_Out_Stat.LastInMailID[0]* or the *PTQPDPMV1_OUT_Stat.Setoperatingmode[0]* register. The following illustration shows the *PTQPDPMV1_OUT.Stat.LastInMailID[0]* control word variable being used.


RDE Template (TEST.RDE) - Animation ON

	Variable Name	Data Type	Address	Value	Set Value	Format	Disable
1	PTQPDPMV1_Out_Stat.LastInMailID[0]	WORD	403000	9999		Hex	
2							
3							

2 Add the following Structured Text lines to the program file. These allow the processor to clear the cold boot command after the reboot, and return to the normal data transfer cycle.


Bootup

```
»»»»»»»»»»»»»» Structured Text Start ««««««««««««««
IF PTQPDPMV1_In_Stat.Reserved3[12] = 16#0099 THEN
  PTQPDPMV1_Out_Stat.LastInMailID[0]:=16#0000;
END_IF;
»»»»»»»»»»»»»»» Structured Text End «««««««««««««««
```

In order to reset the logic, the program must be written in a way to look for word 72 in the input status. If word 72 is equal to 16#0099, the logic will reset the value in *PTQPDPMV1_Out_Stat.LastInMailID[0]* to 16#0000.

Note: It is normal for the remote (passive) Master in Hot Standby applications to reboot twice during this procedure.

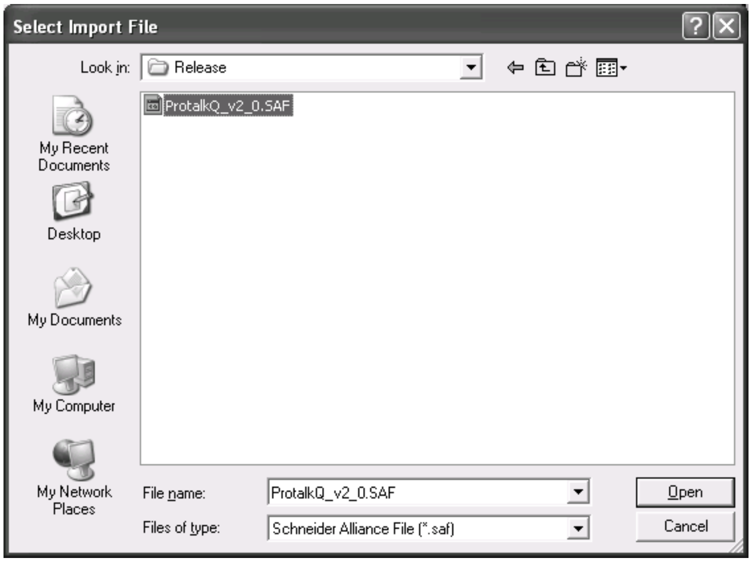
5 Configuring the Processor with ProWORX 32

HSBY Note: ProWorx32 software does not support the 140CPU67160 processor and therefore does not support the PTQ-PDPMV1 HSBY functions.

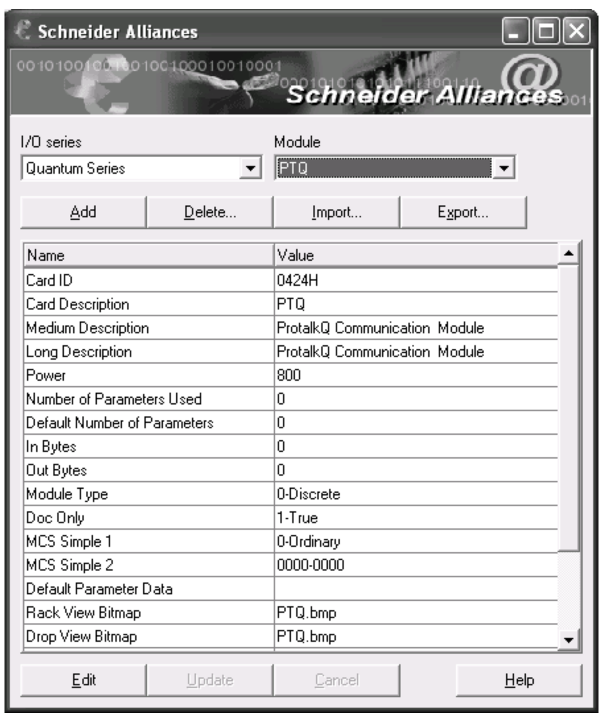
1 Run the **SCHNEIDER_ALLIANCES.EXE** application that is installed with the ProWORX 32 software.

2 Click **IMPORT**.

3 Select the **.SAF** file that is located on the CD-ROM shipped with the PTQ module.



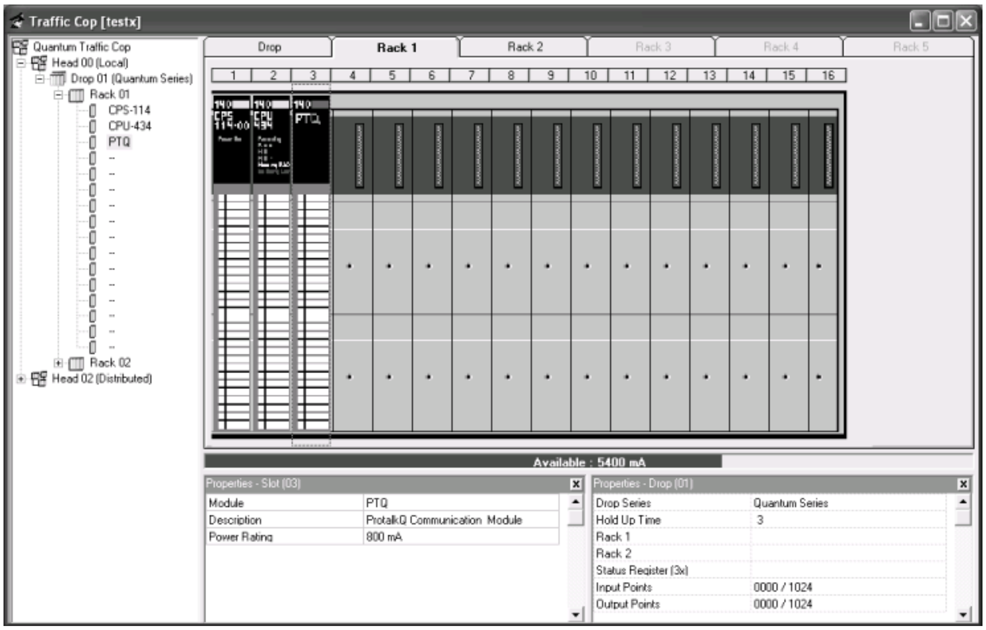
4 Select **OPEN** to import the PTQ module profiles (select I/O series as **QUANTUM**):



The following modules will be available after the .SAF file is imported:

Card Description	Use with the Following Module(s)
PTQPDPMV1	PTQ-PDPMV1 module
PTQ	All PTQ modules except PTQ-PDPMV1

5 Close the *Schneider Alliances* application and run the **PROWORX** software.
6 In the *Traffic Cop* section, select either the **PTQ** or **PTQPDPMV1** cards to be inserted in the selected slot.



6 Mailbox Messaging

In This Chapter

The PTQ-PDPMV1 PROFIBUS DP Master uses a process called Mailbox Messaging to exchange parameter data between the processor, Master, and slave devices. This process provides a way to encapsulate and prioritize commands and data sent between the PROFIBUS Master and slaves.

The PROFIBUS DP-V1 protocol specifies two types of data transmission messages (telegrams): Cyclic Data Telegrams and Acyclic Data Telegrams. Cyclic data communication is the exchange of normal slave input and output (I/O) data and is handled automatically by the Master in a defined, recurring, deterministic sequence based on the configuration you create in ProSoft Configuration Builder (PCB).

Acyclic communication extends data communication beyond normal I/O data to allow moving field device parameterization and calibration data over the bus during runtime and to allow for extended diagnostics and alarm messages. Acyclic data telegrams are transmitted in the gaps between cyclic data telegrams and, therefore, have a lower priority and get less bandwidth than cyclic data.

Mailbox Messaging commands are incorporated into the sample ladder logic. Mailbox messages and responses to mailbox messages are stored in mailbox data types.

The following chapter discusses these features in more detail.

6.1 Mailbox Message Queuing

The PTQ-PDPMV1 module operates asynchronously on the Mailbox Messages and as such provides for the queuing of the messages as they are received. The queue sizes used in the module are as follows:

Queue Type	Queue Size Max	Description
Output message from processor	126	Number of messages that the PTQ module will queue by type of message. Note that status of the queues can be monitored via the Queue Message Count values.
Input messages for processor	126	
Alarm messages from slaves for processor	100	

6.1.1 Queue Timeouts

The PTQ-PDPMV1 module will only allow a message to stay in a queue for up to 10 seconds. If the PROFIBUS Master (for output messages) or the processor (for input and alarm messages) has not successfully received a message within 10 seconds, the module will clear the message out of the queue.

6.2 Special Function Mailbox Messaging Commands

The PTQ-PDPMV1 module supports some extended PROFIBUS functions, using a mailbox data exchange mechanism.

The module supports the following special functions through this mailbox messaging scheme:

Initiated from Processor

Message	Description
Set Operation Mode	Controls the operating state of the PROFIBUS Master
Set Slave Mode	Sends special control command to one or several slaves (Sync/Freeze)
Get Slave Diag	Gets diagnostic information from a slave
Get Slave Config	Gets slave configuration
Set Slave Address	Sets node address of a slave (if supported by the slave)
Get Live List	Gets information from all nodes on the network
MSAC1 Read	DPV1 acyclic read (Class 1)
MSAC1 Write	DPV1 acyclic write (Class 1)

DPV1 Alarm Handling: Generated by Slave Devices

Message	Description
Alarm Indications	Spontaneous alarm indication from DPV1 slave. Structure of data is slave-dependent
Alarm Confirmation	This message is sent **by the PTQ module** automatically as a confirmation to the alarm indications.

The processor logic required to implement these messaging mailbox exchanges will be made simpler after the function blocks are provided.

Sending a mailbox message to the PTQ-PDPMV1 module is a relatively simple process, however, it is important to follow a certain sequence.

Remembering the PROFIBUS Output Data Memory Map:

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
40101	%MW101	0	Last in Mailbox Message ID
40102	%MW102	1	Last Alarm Control index
40103 40104	%MW103 %MW104	2	PROFIBUS CRC32: Computed for PROFIBUS Config
40105 40106	%MW105 %MW106	4	Module CRC32: computed for module data When the module first starts up or recognizes an initialization of the processor, it will compare the values of the two CRCs in the input and output images. If either one of the CRCs do not match, the module will be placed in STOP mode. If each set matches, the module will be placed in OPERATE mode.
40107 to 40150	%MW107 to %MW150	6 to 149	Outgoing Mailbox Data: Mailbox Message command being sent to the PTQ module
40151 to N	%MW151 to N	150 to N	PROFIBUS Output Data: Data going to the PROFIBUS network N is a function of the user-selected size of the PROFIBUS Output data block. Maximum size is 1536 bytes

The important section relevant to the Mailbox Messaging discussion is the Outgoing Mailbox Data section (Word Offsets 2 to 145). Within this section of data, the following structure exists:

Mailbox Message Structure: To PTQ module

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Type	Description
40107	%MW107	6	Message ID	Processor logic or user set. The Message ID field is used by the PTQ module to detect a new message in the PROFIBUS Output data image. When the value is detected as non-zero, the message is processed immediately.
40108	%MW108	7	Message Info	See individual commands for data values to be entered in each of these register locations
40109	%MW109	8	Command	
40110	%MW110	9	Data Size	
40111	%MW111	10	Frame Count	
40112	%MW112	11	Frame Number	
40113	%MW113	12	Offset high	
40114	%MW114	13	Offset Low	
40115	%MW115	14	Extended Word 1	
40116	%MW116	15	Extended Word 2	
40117	%MW117	16	Extended Word 3	
40118	%MW118	17	Extended Word 4	
40119	%MW119	18	Extended Word 5	
40120	%MW120	19	Extended Word 6	
40121	%MW121	20	Extended Word 7	
40122	%MW122	21	Extended Word 8	
-		-	See individual commands	
40150	%MW150	149		

Please keep the following key points in mind:

- If no message is to be sent in the mailbox, the Message ID value in the Output Image must be set to zero (0).
- The PTQ module will set the Last Out Mailbox Message ID value in the Input Image to zero (0).
- The Message ID field is used by the PTQ module to detect new outgoing messages. When the value is detected as non-zero, the PTQ processes the message immediately.
- The Message ID should be changed for each new outgoing mailbox message. A simple zero-to-one toggle scheme can be used, or an incrementing register value can be used (1 to 15).
- When a new message is to be sent:
  - **a** Copy or set up the message contents (keep Message ID value at zero) to the Output Data locations.
  - **b** Set the Message ID value to PTQ module.
- After the PTQ module processes the Outgoing Mailbox command, the PTQ will set the *Last_Out_Mailbox_Message_ID* in the Output Data image to match the outgoing Message ID in the Output image. This event on the processor side can be used by the processor logic to clear the outgoing Message ID if desired.

6.2.1 Mailbox Message: Set Slave Mode

In addition to station-related user data transfer, which is executed automatically, the Master can send control commands to a single slave, a group of slaves, or all slaves simultaneously. These control commands are transmitted as multicast commands. This permits use of sync and freeze modes for event-controlled synchronization of the slaves.

The slaves begin sync mode when they receive a sync command from their assigned Master. The outputs of all addressed slaves are then frozen in their current state. During subsequent user data transmissions, the output data are stored at the slaves, but the output states remain unchanged. The stored output data are not sent to the outputs until the next sync command is received. Sync mode is concluded with the unsync command.

Similarly, a freeze control command causes the addressed slaves to assume freeze mode. In this operating mode, the states of the inputs are frozen until the Master sends the next freeze command. Freeze mode is concluded with the unfreeze command.

Note 1: It is only possible to send control commands when operating mode is either CLEAR or OPERATE.
Note 2: Not all slaves support this feature. Refer to the documentation for the actual slave for more information.

Parameter	Description
Command Initiator	Application
Command Name	SET SLAVE MODE
Command Number	0300h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Set Slave Mode

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	0300h		0300h		Set Slave Mode
Data size	0000h		0000h		
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Group Select	Slave Address	Group Select	Slave Address	
Extended word 2		Control Command		Control Command	
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Extended Fault Information		

Message Information

Refer to Message Information (page 180).

Slave Address

Range 1 to 125; 127

If the request applies for only one slave, that slave address must be entered in the range 1 to 125. If a slave group is to be addressed, the slave address should be 127 (multicast address). The value entered will be byte-swapped. Example: a slave address of 0014 would be entered as 0E00h.

Group Select

Range 01h to FFh (bit coded)

This parameter decides which group should be addressed. Refer to the following example:

b7	b6	b5	b4	b3	b2	b1	b0
Group 8	Group 7	Group 6	Group 5	Group 4	Group 3	Group 2	Group 1

Example: To address Group 1, 2, and 4, the Group Select value should be D0h. If an individual slave should be addressed, the correct group selection must also be made, as the slave will ignore the message if it does not belong to the requested group(s).

What group(s) a slave belongs to is determined during network configuration, and is downloaded during initialization to each slave via the PROFIBUS telegram *Set_Prm*.

Control Command

This parameter specifies the command to send:

Bit	Explanation
0 (LSB)	Reserved, set to zero
1	Reserved, set to zero
2	Unfreeze input data
3	Freeze input data
4	Unsynchronize output data
5	Synchronize output data
6	Reserved, set to zero
7 (MSB)	Reserved, set to zero

Combinations of the bits (Unsync/Sync and Unfreeze/Freeze

Bits 0 or 6	Bits 1 or 7	Explanation
0	0	No function
0	1	Function will be activated
1	0	Function will be inactive
1	1	Function will be inactive

Fault Information and Extended Fault Information

"Fault Information" Contents		"Extended Fault Information" Contents	
0100h	Address out of range	-	
0200h	Group number 0 not permitted	-	
0A00h	Failed to send Global Control request	0A00h	Incorrect operation mode (CLEAR/OPERATE only)
		0150h	Invalid Freeze Group (Group is not initiated to be Freeze Group)
		0250h	Invalid Sync Group (Group is not initiated to be Sync Group)
		0350h	Incorrect Control Command
		0450h	No Sync -/ or Freeze groups enabled in Master configuration
FE00h	Command not possible in Class 2 only mode	-	
FF00h	Module not initialized	-	

6.2.2 Mailbox Message: Get Slave Diagnostics

This command reads diagnostic data from a specified slave.

Note: The response data size depends on the actual slave implementation. Range 6 to 244.

Parameter	Description
Command Initiator	Application
Command Name	GET SLAVE DIAGNOSTICS
Command Number	0400h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Get Slave Diagnostics

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	0400h		0400h		Get Slave Diagnostics
Data size	0000h		(Size of data)		
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Type of request	Slave Address	Type of request	Slave Address	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error code 2	Error code 1	
Extended word 6	-		Error code 4	Error code 3	
Extended word 7	-		Return Code		
Extended word 8	-		Fault Information		
			Station Status 2	Station Status 1	Response data word 1
			Station Status 4	Station Status 3	Response data word 2
			Ident Number		Response data word 3
			Extended Diagnostic Data		Response data word 4 ... ... Response data word n

Message Information

Refer to Message Information (page 180).

Slave Address

Range 1 to 125; specifies the slave to read diagnostics from.

Type of request

00h: Internal slave diagnostic request. The diagnostic information stored in the Master is returned. Can only be requested for slaves configured by the Master.

Note: Not allowed when operating in "Class 2-Only" mode.

01h: External slave diagnostic request. A diagnostic request is sent on the network to the specified slave. Can be requested for all slaves on the network.

Error code [1 ...4]

If "Return Code" equals 8030h ("Negative indication from lower layer"), status values according to the DP-specification may be available in "Error Code 1". Error Codes 2 to 4 are reserved.

Refer to Mailbox Messaging Error Codes (page 180).

Return Code

Refer to Mailbox Messaging Error Codes (page 180)

Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here.

0100h: Address out of range.

0200h: Incorrect "Type of request"

0A00h: Failed to read diagnostic data from slave. Refer to Return Codes (page 181) for additional fault information.

0B00h: Remote station failure. Refer to Return Codes (page 181) for additional fault information.

FE00h: Command not possible; module operates as a Class 2 Master only.

FF00h: Module offline (not initialized or no valid database).

Station Status [1 ... 3]

Refer to EN50170 Vol. 2 for more information.

Master Address

Address of the Master that parameterized the slave.

Ident Number

Unique ID assigned by the PROFIBUS User Organization.

Extended Diagnostic Data

Slave user-specific data. Refer to the documentation for the actual slave for more information.

6.2.3 Mailbox Message: Get Slave Configuration

This command reads the actual configuration (identifier bytes) of a specified slave.

Note: The response data size depends on the actual slave implementation. Range 6 to 244.

Parameter	Description
Command Initiator	Application
Command Name	GET SLAVE CONFIGURATION
Command Number	0500h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Get Slave Configuration

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	0500h		0500h		Get Slave Configuration
Data size	0000h		(Size of data)		Number of identifier bytes (n)
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1		Slave Address		Slave Address	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error Code 2	Error Code 1	
Extended word 6	-		Error Code 4	Error Code 3	
Extended word 7	-		Return Code		
Extended word 8	-		Fault Information		
			Identifier byte 1		Response data byte 1
			Identifier byte 2		Response data byte 2
			Identifier byte 3		Response data byte 3
			...		...
			Identifier byte n		Response data byte n

Message Information

Refer to Message Information (page 180).

Slave Address

Range 1 to 125; specifies the slave to read the configuration from.

Error Code [1 … 4]

If "Return Code" equals 3080h ("Negative indication from lower layer"), status values according to the DP-specification may be available in "Error Code 1", Error Codes 2 through 3 are reserved. Refer to Mailbox Messaging Error Codes (page 180).

Return Code

Refer to Mailbox Messaging Error Codes (page 180).

Fault Information

If "Invalid other" is returned in the Message Information word in the header of the response, information about the fault can be found here. Refer to Message Information (page 180).

0100h: Address out of range.

0A00h: Failed to execute request. Refer to Return Codes (page 181) for additional information.

0B00h: Remote station failure. Refer to Return Codes (page 181) for additional information.

FF00h: Module not initialized.

Identifier Bytes [1 … n]

Refer to EN50170 Vol. 2 for information on the structure of these bytes. In addition, refer to the documentation provided with the slave device for more information.

6.2.4 Mailbox Message: Set Slave Address

This command makes it possible to set the node address of a specified slave, if the slave supports this feature.

Note: The message data size depends on the actual slave implementation. Range 0 to 240 bytes.

Parameter	Description
Command Initiator	Application
Command Name	SET SLAVE ADDRESS
Command Number	0600h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Set Slave Address

	Command			Response		
Message ID	(ID)			(ID)		
Message information	0240h			0200h		
Command	0600h			0600h		Set Slave Address
Data size	(Size of data)			(Size of data)		No. of Slave Data bytes (n)
Frame count	0100h			0100h		
Frame number	0100h			0100h		
Offset high	0000h			0000h		
Offset low	0000h			0000h		
Extended word 1	New Slave Address	Current Slave Addr.		New Slave Address	Current Slave Addr.	
Extended word 2	Slave Ident Number			Slave Ident Number		
Extended word 3		No_add_ Chg	-		No_add_ Chg	
Extended word 4	-			-		
Extended word 5	-			Error Code 2	Error Code 1	
Extended word 6	-			Error Code 4	Error Code 3	
Extended word 7	-			Return Code		
Extended word 8				Fault Information		
Message Data byte 1	Slave Data 1			Slave Data 1		(slave data will appear byte swapped)
Message Data byte 2	Slave Data 2			Slave Data 2		
Message Data byte 3	Slave Data 3			Slave Data 3		
...	...			...		
Message Data byte "n"	Slave Data n			Slave Data n		

Message Information

Refer to Message Information (page 180).

Current Slave Address

Range 1 to 125; specifies the current address of the slave.

New Slave Address

Range 1 to 125; specifies the new address of the slave.

Slave Ident Number

Ident number for the slave, which address should be altered.

No_add_Chg

This parameter specifies whether it is allowed to change the slave address again at a later stage. If this is not allowed, it is only possible to change the address with this function after initial reset. After the initial reset, the slave takes the default address of 126.

00h: Change of address is still possible at a later stage.

01h-FFh: Change of address is only possible after the initial address (that is, the default address) = 126.

Error Code [1 …4]

If "Return Code" equals 3080h ("Negative indication from lower layer"), status values according to the DP-specification in available in "Error Code 1". Error Codes 2 and 3 are reserved. Refer to Return Codes (page 181).

Return Code

Refer to Return Codes (page 181).

Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here:

0100h: Current slave address out of range.

0200h: New slave address out of range.

0A00h: Failed to execute request.

0B00h: Remote station failure.

FF00h: Module not initialized.

Refer to Mailbox Messaging Error Codes (page 180).

Slave Data

With this parameter, it is possible to deliver user-specific data. The data is stored in the slave if possible (that is, EEPROM, FLASH, and so on).

6.2.5 Mailbox Message: Get Live List

This command returns 127 bytes of information about the nodes on the network. Every byte stands for one bus subscriber, and the position of the byte in the response data assigns the address (0 to 126). The content assigns the Station Type.

This command can be sent in all operation modes (that is, STOP, CLEAR, and OPERATE); however, the module must be initialized properly.

Parameter	Description
Command Initiator	Application
Command Name	GET LIVE LIST
Command Number	1800h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Get Live List

	Command	Response	
Message ID	(ID)	(ID)	
Message information	0240h	0200h	
Command	1800h	1800h	Get Live List
Data size	0000h	7F00h	127 Bytes of Data
Frame count	0100h	0100h	
Frame number	0100h	0100h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	Return Code	
Extended word 8	-	Fault Information	
Message Data byte 1		Station Type 0	Response Data Byte 1
Message Data byte 2		Station Type 1	Response Data Byte 1
Message Data byte 3		Station Type 2	Response Data Byte 1
...		...	Response Data Byte 1
Message Data byte "n"		Station Type 126	Response Data Byte 1

Message Information

Refer to Message Information (page 180).

Station Type [0 … 126]

00h: Slave Station

01h: Master Station not yet ready for Token ring (station only physically at the bus)

02h: Master Station ready to enter Token ring (there is not yet any Token transmission)

03h: Master Station in Token Ring (Token transmission through the station)

04h: Station does not exist

Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here. Refer to Message Information (page 180).

0AH00: Failed to build Live List.

FF00h: Module offline (not initialized or no valid database)

6.2.6 Mailbox Message: Acyclic Data Read: Class 1

This command initiates a DPV1 Class 1 acyclic read request. Refer to EN50170 (DPV1) for more information.

Parameter	Description
Command Initiator	Application
Command Name	MSAC1 READ
Command Number	2000h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Acyclic Read

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	2000h		2000h		Acyclic Read
Data size	0000h		(Size of data)		Number of data bytes (n)
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slot	Slave Addr.	Slot	Slave Addr.	
Extended word 2	Length	Index	Length	Index	
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error Decode		
Extended word 6	-		Error Code 2	Error Code 1	
Extended word 7	-		Extended Fault information		
Extended word 8	-		Fault Information		
			Data 1		Response Data byte 1
			Data 2		Response Data byte 1
			Data 3		Response Data byte 1
			...		...
			Data n		Response Data byte 1

Message Information

Refer to Message Information (page 180).

Slave Address

Station address of the slave responder.

Slot Number and Slot Index

Used in the slave to address the desired data block.

Length

This parameter specifies the number of bytes of the data block that have to be read. If the server data block length is less than requested, the length of the response will be the actual length of the data block. If the server data block is greater or equal, the response will contain the same amount of data.

The slave may answer with an error response if data access is not allowed.

Data [1 … n]

Returned data.

Fault Information and Extended Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here.

Fault Information		Extended Fault Information Contents
0100h	Address out of range	-
0A00h	Failed to execute request	Refer to Return Codes (page 181).
0B00h	Remote station failure	
1000h	Remote station DPV1 failure	Function_Number
1100h	Length out of range (>240 bytes)	-
1200h	Slave does not support DPV1	-
1300h	Slave not active or not present in configuration	-
FE00h	Command not possible in "Class 2-Only" mode	-
FF00h	Module offline (not initialized or no valid database)	-

Error Decode, Error Code 1 and Error Code 2

If "Fault Information" contains error code 1000h, more information according to the DPV1 specification can be found here.

6.2.7 Mailbox Message: Acyclic Data Write: Class 1

This command initiates a DPV1 Class 1 acyclic write request. Refer to EN50170 (DPV1) for more information.

Parameter	Description
Command Initiator	Application
Command Name	MSAC1 WRITE
Command Number	2100h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Acyclic Write

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	2100h		2100h		Acyclic Write
Data size	(Size of data)		(Size of data)		Number of data bytes (n)
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slot	Slave Addr.	Slot	Slave Addr.	
Extended word 2	Length	Index	Length	Index	
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error Decode-		
Extended word 6	-		Error Code 2	Error Code 1	
Extended word 7	-		Extended Fault information		
Extended word 8	-		Fault Information		
Message Data byte 1	Data 1		Data 1		
Message Data byte 2	Data 2		Data 2		
Message Data byte 3	Data 3		Data 3		
...	...		...		
Message Data byte n	Data n		Data n		

Message Information

Refer to Message Information (page 180).

Slave Address

Station address of the slave responder.

Slot Number and Slot Index

Used in the slave to address the desired data block.

Length

This parameter specifies the number of bytes that have to be written. If the destination data block size is less than requested, the response will contain an error message. If the data block length is greater than or equal to the requested length, the response contains the number of bytes that have been written. The slave may answer with an error response if data access is not allowed.

Data [1 … n]

Data that should be written.

Fault Information and Extended Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here:

Fault Information		Extended Fault Information Contents
0100h	Address out of range	-
0A00h	Failed to execute request	Refer to Return Codes (page 181).
0B00h	Remote station failure	
1000h	Remote station DPV1 failure	Function_Number
1100h	Length out of range (>240 bytes)	-
1200h	Slave does not support DPV1	-
1300h	Slave not active or not present in configuration	-
FE00h	Command not possible in "Class 2-Only" mode	-
FF00h	Module offline (not initialized or no valid database)	-

Error Decode, Error Code 1, and Error Code 2

If "Fault Information" contains error code 1000h, more information according to the DPV1 specification can be found here.

6.2.8 Mailbox Message: Alarm Indication

This mailbox message indicates to the application that a DPV1 slave has transferred an alarm message to the Master. The message is sent spontaneously by the module. That is, the module itself initiates the mailbox communications.

Detailed information about the cause of the alarm is presented in extended words 1 to 3 and in the message data field (see below).

Command and Response Layout: Alarm Indication

	Command		Response	
Message ID	(ID)		(ID)	
Message information	0240h		0200h	
Command	2200h		2200h	Alarm Indication
Data size	(request length)		0000h	
Frame count	0100h		0100h	
Frame number	0100h		0100h	
Offset high	0000h		0000h	
Offset low	0000h		0000h	
Extended word 1	Slot Number	Slave Address	-	
Extended word 2	Alarm Spec Ack	Seq Number	-	
Extended word 3	Ext Diag	Alarm Type	-	
Extended word 4	-		-	
Extended word 5	-		-	
Extended word 6	-		-	
Extended word 7	-		-	
Extended word 8	Fault Information		-	
Message Data byte 1	Data 1			
Message Data byte 2	Data 2			
Message Data byte 3	Data 3			
…	…			
Message Data byte n	Data n			

Slave Address

Station address of the slave the issued the alarm.

Slot Number

Used by the slave to indicate the source of the alarm. Range 0 to 254.

Seq Number

Unique identification number of the alarm. Range 0 to 31.

Alarm Spec Ack

Provides additional information about the alarm, such as an error appears or disappears. Also indicates whether the slave needs additional knowledge from the Master. For example, writing to a certain memory area with an Acyclic Write request.

Alarm Type

Identifies the alarm type such as Process Alarm, Plug Alarm, and so on. Range 1 to 6, 32 to 126.

Extended Diagnostic Flag

FFh: Slave sends an alarm message with "Extended Diag flag" set

00h: Slave sends an alarm message with "Extended Diag flag" cleared

Data [1 … n]

Additional manufacturer specific alarm information (Alarm - PDU)

Fault Information

If the Message Information word in the header of the message indicates "Invalid Other", addition information is available in this register.

3E00h: Module has received an invalid alarm indication data structure from a DPV1 slave ("Slave Address" contains the node address of the slave that issued the erroneous indication).

Refer to the PNO document "Extensions to EN50170 (DPV)" for more information on how to interpret these parameters.

6.2.9 Mailbox Message: Set Operating Mode

This command allows setting the operating mode of the module (that is, STOP, CLEAR, or OPERATE).

Parameter	Description
Command Initiator	Application
Command Name	SET OPERATING MODE
Command Number	0200h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Set Operating Mode

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	0200h		0200h		Set Operation Mode
Data size	0000h		0000h		
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Conf Req	Req Mode	Conf. Req	Act. mode.	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Appl. Specific Error Code		
Extended word 8	-		Fault Information		

Mode

40h: STOP

80h: CLEAR

C0h: OPERATE

Conf. Req.

00h: Confirmation is not required

01h: Confirmation required. All confirmations are automatically sent by the Master; the user is not required to send a confirmation message.

Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here. Refer to Return Codes (page 181) for more information.

0100h: Invalid operating mode

FF00h: Module not initialized

6.2.10 Mailbox Message: Start Slave

This mailbox message starts a selection of slaves that was previously removed from the processing cycle by means of the mailbox message FB_APPL_STOP_SLAVE.

The message is allowed in all operation modes (STOP, CLEAR and OPERATE).

Note: The message will be accepted even if one or several slaves are not part of the configuration and can therefore obviously not be started. The application can, however, find out about this situation by evaluating the "Fault information" and "Message data words" of the response.

Command and Response Layout: Start Slave

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	0B00h		0B00h		Start Slave
Data size	7E00h		7E00h		
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1					
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Additional Fault Information		
Extended word 8	-		Fault Information		
Message data word 1	Slave 1	Slave 0	Slave 1	Slave 0	
Message data word 2	Slave 3	Slave 2	Slave 3	Slave 2	
Message data word 3 to 62	...	...	...	...	
Message data word 63	Slave 125	Slave 124	Slave 125	Slave 124	

Command:

- Message data word 1-63

  Byte-array stating which slave/slaves to start. Array index is equal to slave address.
  0: Do not affect slave
  1: Start slave
  2-255: Reserved

Response:

- Message information (in response header)

  "Invalid Data Size" is returned if Data size in the command header does not equal 126.
  If "Invalid Other" is returned, further information is to be found in Extended word 8.

- Additional Fault information (Extended word 7)

  If Extended word 8 equals 0x000A -"Failed to execute request" additional info can be found here

- Fault information (Extended word 8)

  0x0001: Invalid setting in Message data word 1-63 of the command.
  0x0002: At least one slave reports a warning. Refer to Message data word 1-63.
  0x000A: Failed to execute request. Additional fault information is to be found in Extended word 7.
  0x00FE: Command not possible, module operates as Class 2 Master only.
  0x00FF: Module not initialized (this command is only possible after END_INIT).

- Message data word 1-63

  Byte-array stating the status of the slaves. Array index is equal to slave address.
  0: Slave unaffected
  1: Slave started
  2: Warning - Slave could not be started because it is not part of the configuration

6.2.11 Mailbox Message: Stop Slave

This mailbox message stops a selection of slaves from the processing cycle.

This message is allowed in all operation modes (STOP, CLEAR and OPERATE).

Note: The message will be accepted even if one or several slaves are not part of the configuration and can therefore obviously not be stopped. The application can, however, find out about this situation by evaluating the "Fault information" and "Message data words" of the response.

Command and Response Layout: Stop Slave

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	0C00h		0C00h		Stop Slave
Data size	7E00h		7E00h		
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	-		-		
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Additional Fault Information		
Extended word 8	-		Fault Information		
Message data word 1	Slave 1	Slave 0	Slave 1	Slave 0	
Message data word 2	Slave 3	Slave 2	Slave 3	Slave 2	
Message data word 3 to 62	...	...	...	...	
Message data word 63	Slave 125	Slave 124	Slave 125	Slave 124	

Command:

- Message data word 1-63

  Byte-array stating which slave/slaves to stop. Array index is equal to slave address.
  0: Do not affect slave
  1: Stop slave
  2-255: Reserved

Response:

- Message information (in response header)

  "Invalid Data Size" is returned if Data size in the command header does not equal 126.
  If "Invalid Other" is returned, further information is to be found in Extended word 8.

- Additional Fault information (Extended word 7)

  If Extended word 8 equals 0x000A -"Failed to execute request" additional info can be found here.

- Fault information (Extended word 8)

  0x0001: Invalid setting in Message data word 1-63 of the command.
  0x0002: At least one slave reports a warning. Refer to Message data word 1-63.
  0x000A: Failed to execute request. Additional fault information is to be found in Extended word 7.
  0x00FE: Command not possible, module operates as Class 2 Master only.
  0x00FF: Module not initialized (this command is only possible after END_INIT).

- Message data word 1-63

  Byte-array stating the status of the slaves. Array index is equal to slave address.
  0: Slave unaffected
  1: Slave stopped
  2: Warning - Slave could not be stopped because it is not part of the configuration
  3: Warning - Slave already stopped

6.3 Receiving Mailbox Message Responses from PTQ Module

After a mailbox message has been sent, a response from the command, usually containing the requested data or the status of the command, is returned from the PTQ module to the processor. The response is returned from the PTQ-PDPMV1 via the PROFIBUS Input data block.

Note: This is for the original layout with the default values; it changes if Slave diagnostics are chosen in PCB.

Remembering the PROFIBUS Input Data Memory Map:

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
41101	%MW1101	0	Configuration, Status and Control data
-	-	-	
41163	%MW1163	73	Number of Messages in the In Mailbox Queue
41164	%MW1164	74	Number of Messages in the Out Mailbox Queue
41165	%MW1165	75	Number of Messages in the Alarm Queue
41157	%MW1157	76	Last Out Mailbox Message ID processed from Output Image
41158	%MW1158	77	Current In Mailbox Control Index
41159	%MW1159	78	Current Alarm Control Index
41180	%MW1180	79	Incoming Mailbox Message data
-	-	-	
41223	%MW1223	222	
41224	%MW1224	223	PROFIBUS Input Data
-	-	-	
41101+N+1	%MW1101+N+1	N	

The important section relevant to the Mailbox Messaging discussion is the Incoming Mailbox Data section (Word Offsets 79 to 222). Within this section of data, the following structure exists:

Mailbox Message Structure: From PTQ module

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Type	Description
41180	%MW1180	79	Message ID	Message ID value will match value used to generate the outgoing mailbox message
41181	%MW1181	80	Message Info	See individual commands for data values to be entered in each of these register locations
41182	%MW1182	81	Command	
41183	%MW1183	82	Data Size	
44184	%MW1184	83	Frame Count	
41185	%MW1185	84	Frame Number	
41186	%MW1186	85	Offset high	
41187	%MW1187	86	Offset Low	
41188	%MW1188	87	Extended Word 1	
41189	%MW1189	88	Extended Word 2	
41190	%MW1190	89	Extended Word 3	
41191	%MW1191	90	Extended Word 4	
41192	%MW1192	91	Extended Word 5	
41193	%MW1193	92	Extended Word 6	
41194	%MW1194	93	Extended Word 7	
41195	%MW1195	94	Extended Word 8	
-	-	-	See individual commands	
41223	%MW1223	222		

Keep the following points in mind:

- If the In_Mailbox_Control_Index values are equal in the Input and Output Data blocks, the PTQ module will place the next message present in the mailbox queue into the Input Data image and increment the In_Mailbox_Control_Index in the Input Data image.
- After the processor processes a new In Mailbox Message, it should set the In_Mailbox_Control_Index (in the Output Image) to match the value received in the Input Image. This tells the PTQ module to transfer the next In Mailbox Message (if there is one) to the processor.

6.4 Mailbox Messaging Error Codes

6.4.1 Acyclic Message Status Word

This register contains bit and code information about the mailbox message. The register is divided into five areas according to the following illustration:

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Message Type								ERR	C/R	(reserved)		Error Code			

Bit / Field	Description	Contents
ERR	This bit indicates if the received command contained any errors.	0: Message OK 1: Error (See also "Error Code" below)
C/R	This bit specifies whether the message is a command or a response.	0: Response Message 1: Command Message
Error Code	If the ERR bit is set, this field contains additional information about the error.	0h: Invalid Message ID 1h: Invalid Message Type 2h: Invalid Command 3h: Invalid Data Size 4h: Message header malformed (offset 008h) 5h: Message header malformed (offset 00Ah) 6h: Message header malformed (offset 00Ch to 00Dh) 8h: Invalid Response 9h: Flash Config Error Fh: **Invalid Other** (All other values are reserved)
Message Type	This field specifies the type of the message.	1h: Application Message 2h: PROFIBUS Specific Message 3h: Memory Message 5h: Reset Message (All other values are reserved)

6.4.2 Return Codes

Possible error codes in Message Data word "Return Code" *(The Return Codes can be byte swapped)*

Return Code	Name	Meaning
8010h	DPMC_ERR_V1C_CLOSED	Internal DPMC instance no longer exists
8011h	DPMC_ERR_V1C_STOPPED	Internal DPMC instance has already been stopped
8012h	DPMC_ERR_V1C_STARTED	Internal DPMC instance has already been started
8013h	DPMC_ERR_V1C_STATE_UNKNOWN	Internal DPMC instance has entered an undefined state
8021h	DPMC_ERR_V1C_REQ_ACTIVE	A request is already active
8022h	DPMC_ERR_V1C_NOT_ALLOWED	Internal DPMC module not initialized correctly
8023h	DPMC_ERR_V1C_INVALID_PAR	Invalid parameter in user request
8024h	DPMC_ERR_V1C_MEM_ALLOC	Internal memory allocation error
8025h	DPMC_ERR_V1C_L2_REQ	Unknown opcode in the confirmation
8026h	DPMC_ERR_V1C_TIMEOUT	Active request terminated with timeout
8028h	DPMC_ERR_V1C_INVALID_LEN	Invalid length in user request
8030h	DPMC_ERR_V1C_REQ_NEG[1]	Negative indication from lower layer
8031h	DPMC_ERR_V1C_REQ_RE	Message frame format error in response
8042h	DPMC_ERR_V1C_REQ_WITHDRAW	Request was recalled
8043h	DPMC_ERR_V1C_REQ_NOT_FOUND	Associated request block not found
80C1h	DPMC_ERR_V1C_MM_FE	Format error in request frame
80C2h	DPMC_ERR_V1C_MM_NI	Function not implemented
80C3h	DPMC_ERR_V1C_MM_AD	Access denied
80C4h	DPMC_ERR_V1C_MM_EA	Area too large
80C5h	DPMC_ERR_V1C_MM_LE	Data block length too large
80C6h	DPMC_ERR_V1C_MM_RE	Format error in response frame
80C7h	DPMC_ERR_V1C_MM_IP	Invalid parameter
80C8h	DPMC_ERR_V1C_MM_SC	Sequence conflict
80C9h	DPMC_ERR_V1C_MM_SE	Sequence error
80CAh	DPMC_ERR_V1C_MM_NE	Area non-existent
80CBh	DPMC_ERR_V1C_MM_DI	Data incomplete or incorrect
80CCh	DPMC_ERR_V1C_MM_NC	Master parameter set not compatible

Refer to Error Codes (page 182).

6.4.3 Error Codes

If the return code indicates DPMC_ERR_V1C_REQ_NEG, the status values according to the DP-standard may be available in Error Code 1. Refer to the PROFIBUS DP specification for information on how to interpret these status values.

Error Code	Name	Meaning
01h	L2_STATUS_UE	
02h	L2_STATUS_RR	
03h	L2_STATUS_RS	
0Ch	L2_STATUS_RDL	Refer to PROFIBUS DP specification
0Dh	L2_STATUS_RDH	
0Fh	L2_STATUS_NA	

6.4.4 DP-V1 Error Codes

Possible error codes in Message Data word "Return Code".

Return Code	Name	Meaning
0003h	DPMC_ERR_M_MEM_ALLOC	Internal memory allocation error
0004h	DPMC_ERR_M_L2_REQ	Unknown opcode in the configuration
0005h	DPMC_ERR_M_INVALID_PAR	Invalid parameter in user request
0007h	DPMC_ERR_M_NOT_IN_DATA	Slave is not in DataExchange (thus no DP-V1 request can exist)
0012h	DPMC_ERR_M_REQ_ACTIVE	A request is already active
0018h	DPMC_ERR_M_NOT_ALLOWED	Internal DPMC module not initialized correctly
0021h	DPMC_ERR_M_CLOSED	Internal DPMC instance no longer exists
0022h	DPMC_ERR_M_STOPPED	Internal DPMC instance has already been stopped
0023h	DPMC_ERR_M_STARTED	Internal DPMC instance has already been started
0024h	DPMC_ERR_M_STATE_UNKNOWN	Internal DPMC instance has entered an undefined state
002Fh	DPMC_ERR_M_SLAVE_NOT_FOUND	Slave does not respond
0031h	DPMC_ERR_M_TIMEOUT	Active request terminated with timeout
0034h	DPMC_ERR_M_INVALID_LEN	Invalid length in user request
0035h	DPMC_ERR_M_REQ_NEG	Negative indication from lower layer
0036h	DPMC_ERR_M_REQ_RE	Message frame format error in response
0037h	DPMC_ERR_M_REQ_WITHDRAW	Request was recalled
0038h	DPMC_ERR_M_REQ_NOT_FOUND	Associated request block not found
0040h	DPMC_ERR_M_MM_FE	Format error in request frame
0041h	DPMC_ERR_M_MM_NI	Function not implemented
0042h	DPMC_ERR_M_MM_AD	Access denied
0043h	DPMC_ERR_M_MM_EA	Area too large
0044h	DPMC_ERR_M_MM_LE	Data block length too large
0045h	DPMC_ERR_M_MM_RE	Format error in response frame
0046h	DPMC_ERR_M_MM_IP	Invalid parameter
0047h	DPMC_ERR_M_MM_SC	Sequence conflict
0048h	DPMC_ERR_M_MM_SE	Sequence error
0049h	DPMC_ERR_M_MM_NE	Area non-existent
004Ah	DPMC_ERR_M_MM_DI	Data incomplete or incorrect
004Bh	DPMC_ERR_M_MM_NC	Master parameter set not compatible
004Ch	DPMC_ERR_M_S7_XA	
004Dh	DPMC_ERR_M_S7_XR	PROFIBUS error for DP-V1 (NRS-PDU received)
004Eh	DPMC_ERR_M_S7_XW	

7 Hot Standby Support

In This Chapter

7.1 Hot Standby Overview

This section describes the PTQ-PDPMV1 PROFIBUS DP Master module specifications and startup support for Modicon Quantum Hot Standby system.

Use a Modicon Quantum Hot Standby with Unity system and PROFIBUS when downtime cannot be tolerated. Hot standby systems deliver high availability through redundancy. A hot standby system consists of two identical configurations.

- Modicon Quantum 140 CPU 671 60
- Modicon Quantum Power Supply Module
- Modicon Quantum RIO Head
- ProSoft Technology PTQ-PDPMV1 module hardware version 1.13 or higher
- Modicon Optional Modules (NOE, NOM)

One of the 140 CPU 67160s acts as the Primary controller and the other acts as the Standby controller. The Primary controller runs the application program and operates the remote I/O.

Note: The Modicon Quantum RIO Head is required even if the Remote I/O will not be used.

7.1.1 Identical Configurations

Two backplanes are configured with identical hardware and software. One of the programmable logic controllers (PLCs) functions as the Primary controller and the other as a Standby controller, and either controller can be put in the Primary state, but the other must be in the Standby state or offline.

7.1.2 Primary and Standby Controllers

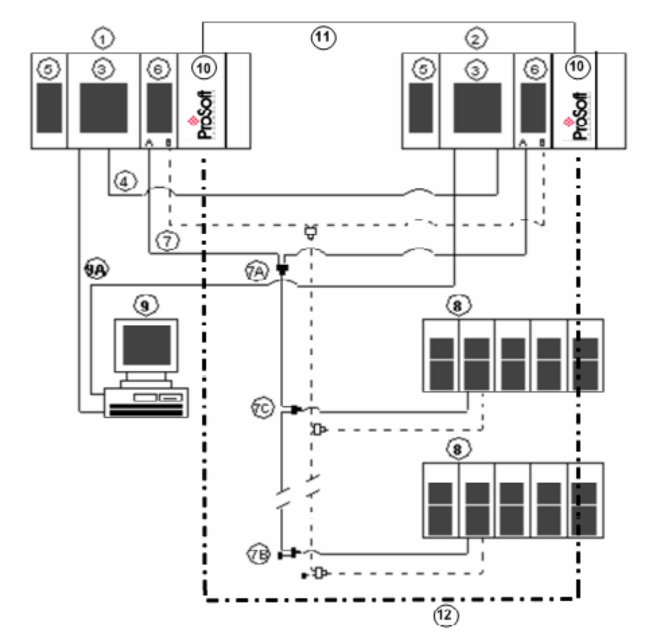
The Primary controller executes the application program, controls the remote I/O, and updates the Standby controller after every scan (program cycle). If the Primary controller fails, the Standby controller takes control within one scan. To determine if the Primary controller failed note controller's status displayed in the HE CPU LCD screen and the RIO Head's status displayed by the RIO Head's LEDs.

The Standby controller does not execute the full application program but only the first section, and the Standby controller does not control the remote I/O but checks out the availability of the Modicon Quantum Hot Standby with Unity equipment.

Note: For additional information on Quantum Hot Standby support, refer to the *Unity Pro Hot Standby User Guide*.

7.1.3 System Components

The following illustration shows the components required for a Modicon Quantum Hot Standby with Unity system.



1 Primary PLC
2 Standby PLC
3 Modicon Quantum Hot Standby with Unity controller with integrated coprocessor
4 Fiber Optic Cable to connect to both controllers
5 Modicon Quantum power supply module: Install power supply in first slot for better rack layout.
6 Modicon Quantum RIO head
7 Coaxial cable with splitters (7A) (MA-0186-100), trunk terminators (7B) (52-0422-000), and tap (7C) (MA-0185-100) for connecting the RIO heads (6) with the RIO drops (8). The dashed connections represent a redundant connection in the RIO network, which is not required for the Modicon Quantum Hot Standby with Unity system.
8 Modicon Quantum RIO drop
9 Unity Pro computer connected to both controllers via Modbus or Modbus Plus (9A)
10 PTQ-PDPMV1 HSBY modules
11 PTQ-PDPMV1 Ethernet redundancy communication cable
12 PROFIBUS network. Each PTQ-PDPMV1 placed at each network end.

Note: The 140 CRP 932 00 RIO Head unit is required for Hot Standby System to work.

7.1.4 Modicon Quantum Hot Standby with Unity and IEC Logic

Overview

A Modicon Quantum Hot Standby with Unity system requires two backplanes configured with identical hardware, software, and firmware. One of the controllers (PLC) functions as the Primary controller and the other as a Standby controller.

- The Primary updates the Standby after every scan.
- The Primary and Standby communicate constantly monitoring the health of the system.
- If the Primary fails, the Standby takes control within one scan.

7.1.5 Understanding System Scan Time in Modicon Quantum Hot Standby with Unity Systems

Effect on System Scan Time

The scan time of any Modicon Quantum Hot Standby with Unity system depends on the amount of data transferred. Because data must be transferred from Primary to Standby, any Modicon Quantum Hot Standby with Unity system always has a higher scan time than a comparable stand-alone system.

Performance Considerations

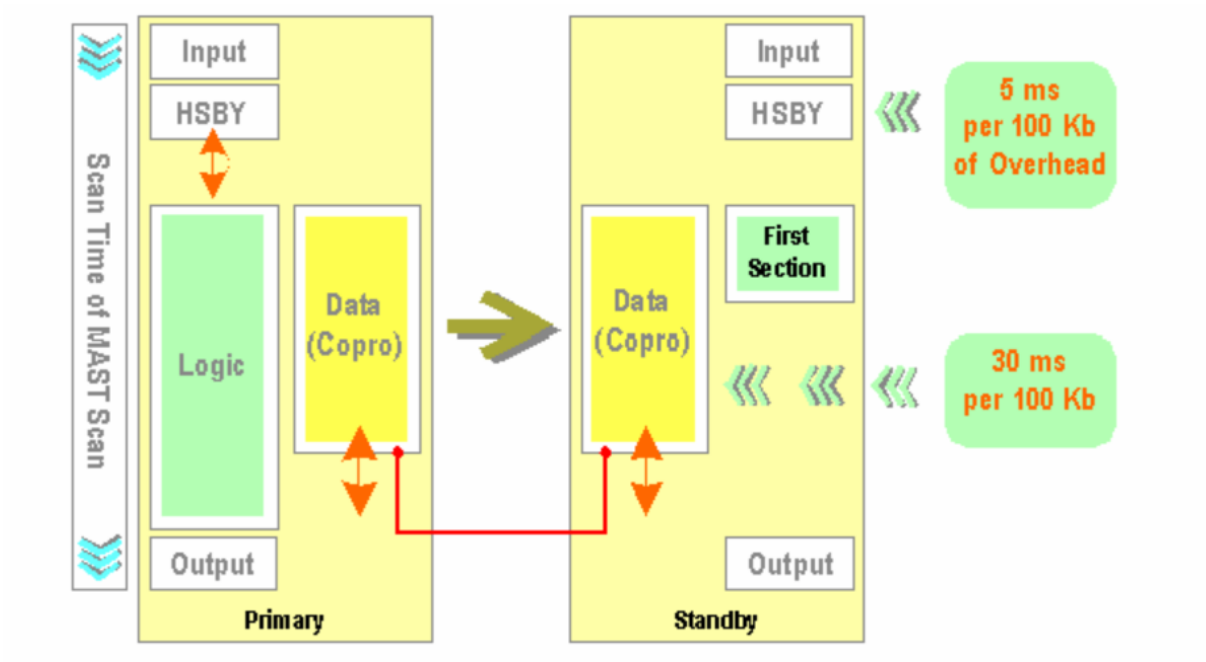
A Modicon Quantum Hot Standby with Unity system increases the length of a MAST scan, creating system overhead.

Note: System overhead is the time required to copy the application data to the communication link layer.

The network scan (communication between Primary and Standby "copros")

1. Exchanges data between both controllers
2. Runs in parallel with the application program.

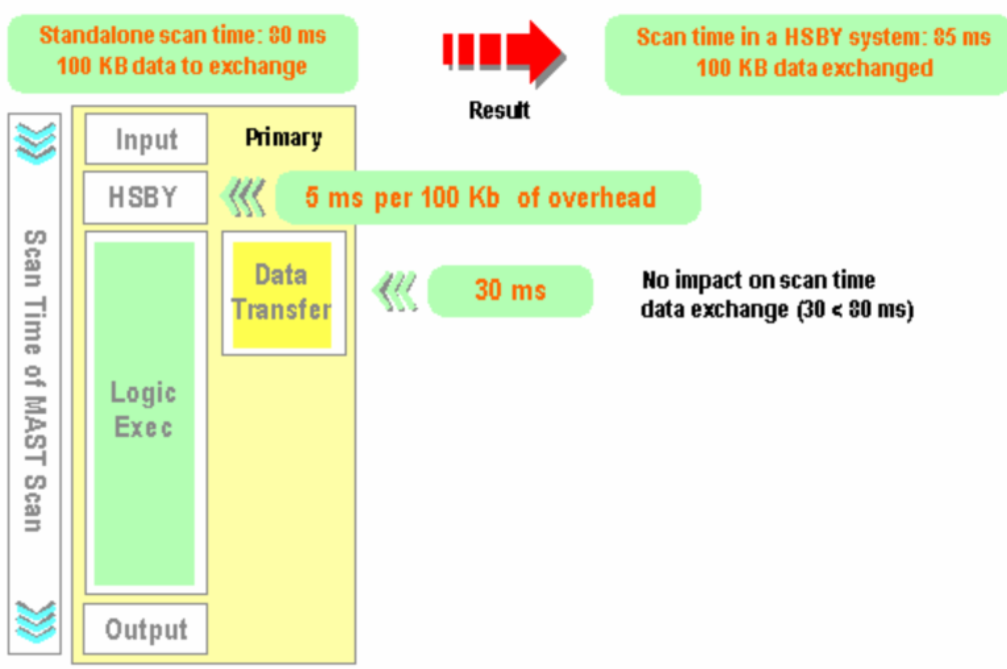
A Hot Standby system



Most of the time, the MAST scan hides the network scan. However, when some application programs are processed, additional system overhead may occur.

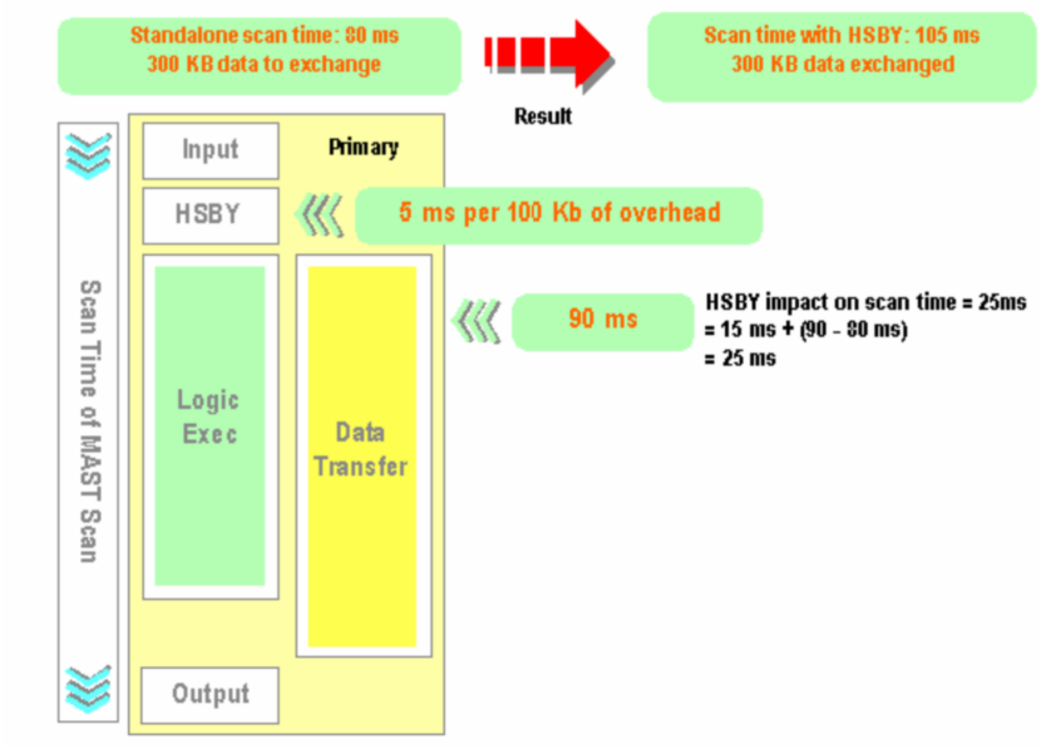
Example #1

- Stand-alone application scan time: 80 ms
- Data (state RAM + unallocated variables): 100 Kb



Example #2

- Stand-alone application scan time: 80 ms
- Data (state RAM + unlocated variables): 300 Kb



Note: In addition to the above times for system overhead, the PTQ-PDPMV1 module may acquire from 100 ms to 300 ms of switch-over time. All configured data is to be updated as fast as the combined asynchronous events occur based on the processor scan time, backplane transfer time, PTQ data handling time and PROFIBUS Master bus cycle time. PROFIBUS bus cycle time is based on slave reaction time, sync time, baud rate and other bus delay times for a given number of slaves on the network.

7.2 Setting Up the Modicon Quantum Hot Standby with Unity System

7.2.1 Overview

Schneider Electric is a leader in offering fault-tolerant, redundant systems, and Hot Standby. Setting up a Modicon Quantum Hot Standby with Unity system involves a number of processes, summarized in the following paragraphs here, and explained in detail elsewhere.

7.2.2 Mapping the Backplane Extensions

A Modicon Quantum Hot Standby with Unity requires two backplanes with at least four slots. You must map the two backplanes in an identical manner:

- same Modicon Quantum Hot Standby with Unity HE CPU with integrated coprocessor (Copro)
- same firmware
    - same revision level
    - same Modicon Quantum power supply module
    - same Modicon Quantum RIO Head

And, if other modules are used, for example local I/Os, NOMs, NOEs, those modules must be identical.

For additional information on Modicon Quantum Hot Standby Startup support refer to the Unity Pro User Guide.

7.2.3 PTQ-PDPMV1 Hot Standby Considerations

Limitations

The solution allows for up to six PTQ modules per rack (for both Primary and Standby).

It will not be possible to install a PTQ in a RIO drop.

HSBY Operating Modes

Generally, the user will have full control over the switchover via the command register through application program control. This is accomplished by the user application through the SW60 command register.

7.2.4 Hot Standby States

State Description

There are three normal running states of operation in a general Hot Standby system.

- **PRIMARY:** The PLC is set as the Primary CPU and controls the Input/Output process as if it is stand-alone.
- **STANDBY:** This PLC is set as the Standby CPU and is ready to take over as Primary at all times, but the Primary CPU controls process and network. Outputs are not applied.
- **OFFLINE:** The PLC is set to offline mode and the CPU cannot act like a Primary or Standby CPU.

It may be in STOP or disconnected mode. Here the PLC behaves as a normal non-HSBY CPU.

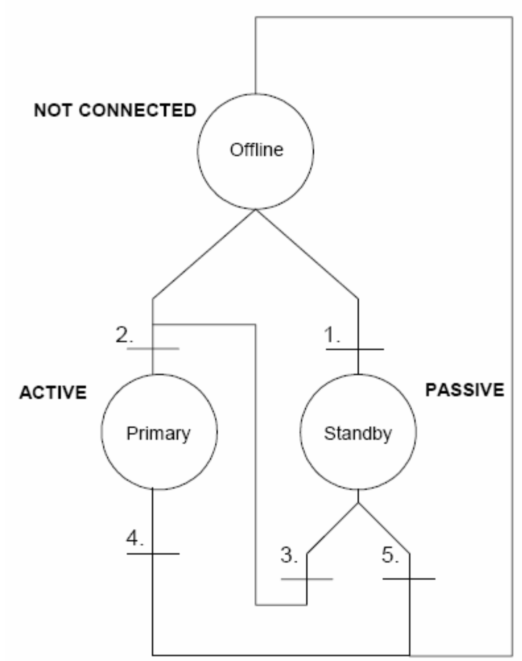


Figure 1 Hot standby states

The equivalent states for the Master module are "Not connected", "Active", and "Passive" (bold text in the illustration above).

- **NOT CONNECTED**: The Master module would be disconnected from the PROFIBUS network.
- **ACTIVE:** The Master module would act as a class 1 PROFIBUS DPV1 Master, managing I/O data, acyclic data, alarms, diagnostic and parameter data with its assigned slaves as if it were stand-alone.
- **PASSIVE:** The Master module would monitor the status of the local (active) Master, and if it detects any problems, it would inform the application about the situation. Note that the remote (passive) Master would not switch to local (active) unless the application tells it to do so.

7.2.5 Transition Description

Offline to Standby (1)

- The remote (passive) Master would attain its node address by subtracting one (1) from the address derived from the database. For example, if the Master address in the database equals 2, the remote (passive) Master would use address 1. If the Master address in the database equals 0, it would use address 125.
- After a switchover, it is important that the previously local (active) Master does not become passive before the counterpart has switched to active. If this statement is not adhered to, a dangerous situation with two Masters having the same address (two remote (passive) Masters in this case) would arise. The result of such a situation would be very unpredictable since there is really no way of detecting it.

Offline to Primary (2)

See Standby to Primary considerations.

Standby to Primary (3)

It is important that the Master becomes active before the watchdog of the slaves expires. To allow for this switchover time, the watchdog value would have to be extended in the bus parameter settings.

When the remote (passive) Master switches to active, it would change its node address to the primary address. To achieve this, the ASPC2 must be reset and reinitialized with the new bus parameter TS. Note that it is just the ASPC2 that is reinitialized, not the entire Master module.

- The state of the "PA-bit" in register "HSBY Local status" would change to 0, indicating that the Master module now operates as local (active) Master.
- When the remote (passive) Master switches to active, it will not re-parameterize the slaves that report being in the "DATA" state.
- After a switchover, the application will be informed when there is valid data available in the input output image area.
- Primary obtains the Master node address.

Primary to Offline (4)

The local (active) Master leaves the bus as fast as possible since the risk of having two Masters with the same address after a switchover must be avoided (two local (active) Masters in this case). The time it takes for leaving the bus will not exceed the switchover time.

Standby to Offline (5)

The time it takes to switch to offline is not critical since the transition would not influence the operation of the counter part, which will carry on working as a stand-alone Master.

> **Note:** For additional information and restrictions with Quantum processor behavior, refer to the Unity Pro HSBY User Guide.

7.2.6 HSBY State vs. Master Operation Mode

The matrix below indicates how the Master module would behave on a PROFIBUS network for all possible combinations of Master operation mode (OFFLINE, STOP, CLEAR, OPERATE) and "HSBY state" (NOT CONNECTED, PASSIVE, ACTIVE).

HSBY State vs. Operation Mode

	NOT CONNECTED	PASSIVE	ACTIVE
OFFLINE	No network traffic.	No network traffic.	
STOP		Ping requests are issued. Slave communication takes place	Ping requests are responded to. No slave communication takes place
CLEAR		The remote (passive) Master cannot attain any of these operation modes. The application can, however, instruct the Master to attain one of these modes after a switchover.	Ping requests are responded to. Slave communication takes place; only input data is read.
OPERATE			Ping requests are responded to. Slave communication takes place; both input- and output data is exchanged.

7.2.7 Ping Message

The remote (passive) Master would cyclically send ping messages to the local (active) Master, which in turn would respond to the message.

If the local (active) Master stops receiving ping requests (ping.req in figures below), it would assume that something is wrong with the counterpart or the field bus link.

If the remote (passive) Master does not get any response (ping.res in figures below) to its ping requests it would assume that something is wrong with the counterpart or the field bus link. In both mentioned cases, the erroneous situation would be signaled to the application by clearing (0) the COM-bit in the "HSBY Local Status" register.

The time between ping requests (TP) is bus cycle-dependent. One request is sent every time the remote (passive) Master is in possession of the token. If there is no response from the counterpart the ping message would be resent x times, where x is equal to bus parameter *max_retry_limit*, before the COM-bit is cleared. The time the remote (passive) Master waits for a response until it re-sends the message (TSL) is defined by bus parameter SlotTime.

If the local (active) Master fails when it is in possession of the token, the remote (passive) Master would sense this and reclaim the token after the timeout time TTO. In other words, the remote (passive) Master will not wait for the entire TTR (Target Rotation time) to expire before it can send a ping request and detect the faulty local (active) Master.

TTO is calculated according to the following formula (according to the FDL-layer specification):

- TTO = 6*TSL + 2*TS*TSL, where TS is the physical address of the remote (passive) Master.

The formula implies that the physical address of the remote (passive) Master should be kept as low as possible in order to achieve an optimal time-out time.

This means that the time it takes for the remote (passive) Master to detect an local (active) Master failure (TFA) is based on two factors: the time it takes to reclaim the token + the time it takes to send a ping message with retries:

- TFA = TTO + (*max_retry_limit* + 1)*TSL

The local (active) Master would poll for incoming ping requests every TA ms, and if no request has been received since the last poll, the COM-bit is cleared. The poll sequence is asynchronous to the ping sequence, so in the worst case it might take 2*TA (that is, the last poll took place just before the remote (passive) Master failed) before the local (active) Master detects a failure.

TA is calculated according to the following formula:

- TA = Max[30, TTR+SM], where SM is a safety margin (10% of TTR).

The formula implies that the minimum time between polls is 30 ms (highest timer resolution of the RTOS is 5 ms), while the maximum time is proportional to the Target Rotation time (TTR).

The time it takes for the local (active) Master to detect a remote (passive) Master failure is not so crucial since no switchover would take place. The local (active) Master would just carry on operating as a stand-alone Master. Another argument to keep TA at a reasonable value is that we would like to keep the CPU load as low as possible since the most important thing for the local (active) Master is to keep up the pace with its assigned slaves.

The following table gives an example of the discussed timing values for some baud rates.

Ping Timing Values

The values are based on a PCB PROFIBUS Master Configuration Software configuration consisting of 96 slaves, where each slave has 16 bytes input data and 16 bytes output data (that is, max DPRAM size). The bus profile in the bus parameters set-up is set to "Single Master". The local (active) Master has a physical address equal to one, which means that the passive one will use address zero. TTR in the example is doubled compared to the value that the Master configuration software calculates.

Calculated Times for Detecting a Missing Counterpart

Baud Rate	TA[ms]	TP[ms]	TFA[ms]
12 Mbps $\rightarrow T_{TR}$ = 50 ms, T_{SL} = 1000 Tbits	55	$\leq T_{TR}$	1
1.5 Mbps $\rightarrow T_{TR}$ = 186 ms, T_{SL} = 300 Tbits	205		2
500 kbps $\rightarrow T_{TR}$ = 500 ms, T_{SL} = 200 Tbits	550		4
93.75 kbps (PA) $\rightarrow T_{TR}$ = 2374 ms, T_{SL} = 100 Tbits	2611		9
45.45 kbps (PA) $\rightarrow T_{TR}$ = 14000 ms, T_{SL} = 640 Tbits	15400		113
9.6 kbps $\rightarrow T_{TR}$ = 23200 ms, T_{SL} = 100 Tbits	25520		83

Ping Sequence

This section shows the ping sequence, and how the COM-bit in register "HSBY Remote status" is affected under different scenarios.

The figures show the active and the remote (passive) Masters connected via the PROFIBUS network. For each Master, the DPRAM (that is, application interface) and the software objects that are responsible for the ping sequence are shown.

Start-up

Local (active) Master operational prior to passive

As soon as the local (active) Master is ready to communicate on the PROFIBUS network, it would start polling for ping messages from the passive one. As soon as it receives the first ping request, the COM-bit is altered from zero to one.

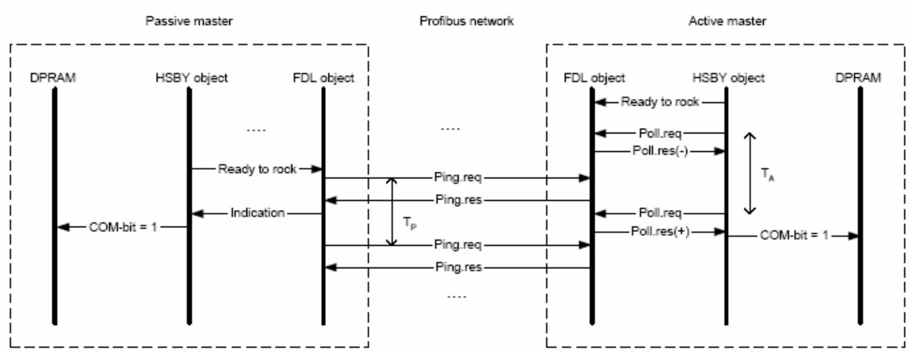


Figure 2 Active master is ready prior to Passive master

Remote (passive) Master operational prior to active

As soon as the remote (passive) Master is ready to communicate on the PROFIBUS network, it would start sending ping messages to the local (active) one. As soon as the local (active) Master responds, the COM-bit is altered from zero to one.

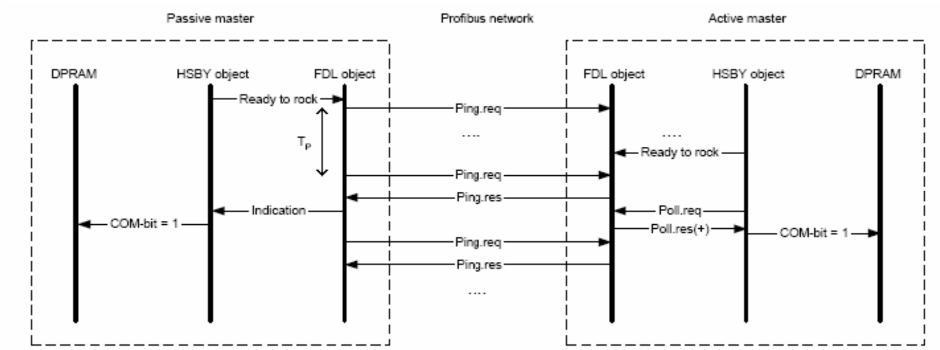


Figure 3 Passive master is ready prior to Active master

Remote (passive) Master failure

When the local (active) Master has not received any ping messages from the counterpart within TA, it would clear the COM-bit. This scenario would also apply when the field bus link between the two Masters is lost, due to a cable break, for example.

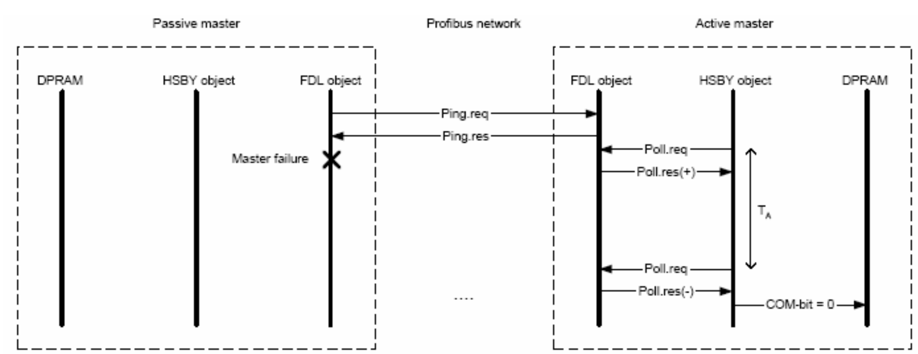


Figure 6 Passive master failure

Ping Message Structure

In addition to detecting a lost counterpart, the ping message is also used to communicate status information between the local (active) and remote (passive) Master.

The ping message would carry the following data (4 bytes).

Ping.req

Byte	Data	Description
0	HSBY local status	Status information of the remote (passive) Master
1	HSBY Nr of local slaves	Number of slaves accessible to the remote (passive) Master
2 to 3	16-bit CRC	CRC of the remote (passive) Master's database

Ping.res

Byte	Data	Description
0	HSBY local status	Status information of the local (active) Master
1	HSBY Nr of local slaves	Number of slaves accessible to the local (active) Master
2 to 3	16-bit CRC	CRC of the local (active) Master's database

7.2.8 PTQ Link Message

The PTQ modules require Ethernet UDP services for local (active) and passive module communications.

Two types of services are provided.

1 Service port 3001 - Used for status data and CRC data of the standby to primary unit (the CRC values are listed below from the PTQ-PDPMV1 Reference Guide).

Quantum Address	Unity Address	Word Offset	Name	Description
1043	%IW1043	43	PROFIBUS configuration checksum	CRC32 checksum for PROFIBUS Master configuration downloaded from configuration utility
1045	%IW1045	45	PTQ module configuration checksum	PTQ-PDPMV1 module configuration checksum for module configuration downloaded from configuration utility

2 Service port 3002 - Used for DPV1 messages. The transfer of these messages is necessary to insure the messages are delivered and received upon a switchover condition.

7.2.9 Crossed Status Information

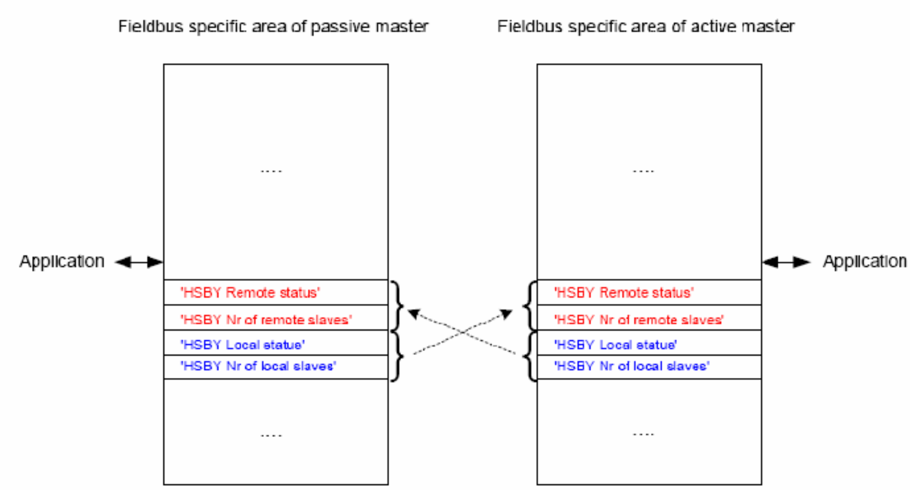
The ping message communicates status information between the active and remote (passive) Masters, and vice versa.

A part of this "crossed status information" (CSI) would be presented in the "HSBY Remote status"-/"HSBY Nr of remote slaves" registers located in the fieldbus specific area of the DPRAM.

These registers would be updated every time new status information is received from the counterpart. Polling for new status information takes place every TA ms.

Normally, the application would forward this information to a high-level system (for example, application of the primary PLC), which in turn would determine if a switchover should be carried out, or not.

The following illustration shows how CSI is communicated between the two Masters and how it is displayed in the fieldbus specific area of the DPRAM.



7.2.10 Conditions for Switchover

Each PTQ situated in the Primary or the Standby local rack must provide both CPUs with its own diagnostics in order to request and perform a switchover. Diagnostics must be crossed between Primary and Standby PTQs and the associated CPU. Both CPUs (Primary and Standby) must be informed anytime of the status of all PTQs. The PTQ module will post Local and Remote status information in the Input Status/Control Data Area block word offset 60 to 63.

Note: For the backplane driver, the 100 ms to 300 ms spec depends on scan time. The backplane driver cannot detect a switchover in 300 ms if the scan time is 500 ms. The backplane drivers can communicate with the PLC only at the end-of-scan.

In other words:

- The Primary must be informed of the status of its own PTQ(s)
- The Primary must inform the Standby of the status of its own PTQ(s)
- The Standby must be informed of the status of its own PTQ(s)
- The Standby must inform the Primary of the status of its own PTQ(s)

Conditions for user to consider switchover are as follows:

- PTQ Master module failure
- Bus not connected or all devices not responding
- PTQ Master module not configured
- Bus cable break. Status information provided to user to determine appropriate Master with most slaves will be available in the Input data block.

Based on this crossed information updated anytime and simultaneously, the user application code can perform a switchover using command register and status bits. A switchover can be performed only if the Primary PTQ fails and the Standby PTQ is able to take control. We cannot allow the system to switch over if the Standby PTQ status is not known with precision. In case of several PTQs per rack, the application code will have to diagnose the health status of all Standby PTQs before performing a switchover. These diagnostics have to be taken into consideration in order to avoid leaving a bad situation on one side (Primary) and getting a worse one on the other side.

7.3 PTQ-PDPMV1 Operation

While in Primary mode, the module will read the output area and write the input area. The module will constantly scan the HSBY control word to determine the HSBY state.

- Active LED will flash if the module is in Standby mode in a Hot Standby system. The Hot Standby Status Word of the Modicon Quantum processor is read during each end-of-scan.
- During switchover caused by a failure on the primary, all PROFIBUS I/Os will be held at their last values, until the moment the new PTQ takes control (no glitch on I/O devices).

7.3.1 PTQ-PDPMV1 HSBY Diagnostic Data

PTQ Input Data Block

The PTQ module reports the HSBY Local and Remote Status Registers via the Input Data Block.

Input Data Block *(HSBY words only)*

<table>
<tr><th>Quantum Address (Example)</th><th>Unity Address (Example)</th><th>Word Offset</th><th>Name</th><th>Description</th></tr>
<tr><td>1060</td><td>%IW1060</td><td>60</td><td>Low byte: HSBY Remote Status - from PROFIBUS interface
High byte: HSBY Remote number of slaves - from PROFIBUS interface</td><td>
<table>
<tr><th colspan="8">High Byte</th><th colspan="8">Low Byte</th></tr>
<tr><td colspan="4">16</td><td colspan="4">8</td><td colspan="4">7</td><td colspan="4">0</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>COM</td><td>-</td><td>-</td><td>OD</td><td>DB</td><td>CE</td><td>SO</td><td>PA</td></tr>
</table>
<table>
<tr><th>Bit</th><th>Explanation</th></tr>
<tr><td>PA</td><td>This bit would indicate the state of the local Master.
0 - Local (active) Master (controlled by the Primary PLC)
1 - Remote (passive) Master (controlled by the Standby PLC)</td></tr>
<tr><td>SO</td><td>This bit indicates if the local Master recognizes any of its assigned slaves as offline.
0 - At least one slave is offline
1 - All slaves OK</td></tr>
<tr><td>CE</td><td>This bit indicates if the local Master has recognized a critical error.
0 - No critical errors
1 - Critical error active
This bit is set when problems with the ping sequence is encountered.</td></tr>
<tr><td>DB</td><td>This bit indicates if the local Master has detected a database mismatch.
0 - Database OK
1 - Database mismatch</td></tr>
<tr><td>OD</td><td>This bit indicates when the data in the Output data area of the DPRAM is updated after a switchover.
0 - Output data is not updated
1 - Output data is updated (Once this bit is set, it remains set for the remaining session until the Anybus is either reset or HSBY state changes to "Not Connected")</td></tr>
<tr><td>-</td><td>Not used; set to zero</td></tr>
<tr><td>-</td><td>Not used; set to zero</td></tr>
<tr><td>COM</td><td>This bit indicates if the counterpart is present.
0 = Counterpart not present
1 = Counterpart is present</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Quantum Address (Example)</th><th>Unity Address (Example)</th><th>Word Offset</th><th>Name</th><th>Description</th></tr>
<tr><td>1061</td><td>%IW1061</td><td>61</td><td>Low byte: HSBY Local Status - from PROFIBUS interface
High byte: HSBY Local number of slaves - from PROFIBUS interface</td><td>
<table>
<tr><th colspan="8">High Byte</th><th colspan="8">Low Byte</th></tr>
<tr><th colspan="4">16</th><th colspan="4">8</th><th colspan="4">7</th><th colspan="4">0</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>HS</td><td>-</td><td>-</td><td>OD</td><td>DB</td><td>CE</td><td>SO</td><td>PA</td></tr>
<tr><th colspan="4">Bit</th><th colspan="12">Explanation</th></tr>
<tr><td colspan="4">HS</td><td colspan="12">This bit indicates if the Hot Standby functionality is enabled.
0 - HSBY disabled. Module operates as stand-alone Master or HSBY state equals "Not connected".
1 - HSBY enabled</td></tr>
</table>
</td></tr>
<tr><td>1062</td><td>%IW1062</td><td>62</td><td>HSBY Message length - from UDP HSBY Server</td><td>See explanation of bits OD, DB, CE, SO and PA in chart for address 1060.</td></tr>
<tr><td>1063</td><td>%IW1063</td><td>63</td><td>Low byte: HSBY Passive Status - from UDP HSBY Server
High byte: HSBY Passive number of slaves Message length - from UDP HSBY Server</td><td>Refer to word 60 for explanation. This is a backup word derived from Ethernet UDP messaging</td></tr>
<tr><td>1064 to 1165</td><td>%IW1064 to %IW1165</td><td>64 to 65</td><td>HSBY Passive PROFIBUS CRC32 - from UDP HSBY Server</td><td>CRC32 checksum for PROFIBUS Master configuration downloaded from configuration utility via UDP</td></tr>
<tr><td>1066 to 1167</td><td>%IW1066 to %IW1167</td><td>66 to 67</td><td>HSBY Passive User Cfg CRC32 - from UDP HSBY Server</td><td>PTQ-PDPMV1 module configuration checksum for module configuration downloaded from configuration utility via UDP</td></tr>
</table>

HSBY Input Status Data Word Details

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
1060	%IW1060	60	Remote HSBY Master Status data and number of slaves seen by this Master
1061	%IW1061	61	Local HSBY Master Status data and number of slaves seen by this Master

Word Offset 60 HSBY Local (Active) Master Status Data

HSBY Active # of Slaves (High byte)		HSBY Active Status (Low byte)	
15	8	7	0

Word Offset 61 HSBY Passive Master Status Data

HSBY Passive # of Slaves (High byte)		HSBY Passive Status (Low byte)	
15	8	7	0

Active and Passive HSBY Master Status Data Low Byte - Bits 0 to 7

Note. Bits 0 to 7 are not considered valid until the "HS-bit" equals one, that is, the Master module is initialized as a HSBY Master (passive or active). Once set, the "HS-bit" will keep this value for the remaining active session until the Master module is either reset or HSBY-state changes to "Not connected".

All bits would be set to zero at power-up, and when the HSBY-state equals "Not connected".

HSBY Local Status

Bit	Name	Explanation
0	PA	This bit indicates the state of the local Master. 0 = Local (active) Master: Master is controlled by the Primary PLC 1 = Remote (passive) Master: Master is controlled by the Standby PLC
1	SO	This bit indicates if the local Master recognizes any of its assigned slaves. 0 = At least one slave is offline 1 = All slaves OK
2	CE	This bit indicates if the local Master has recognized a critical error. 0 = No critical errors 1 = Critical error active This bit is set when problems with the ping sequence are encountered. Detailed information about the problem is dumped in the fieldbus-specific area at address hF90-FBC.
3	DB	This bit indicates if the local Master has detected a database mismatch. 0 = Database OK 1 = Database mismatch
4	OD	This bit indicates when the data in the Output area of the DPRAM is updated after a switchover. 0 = Output data is not updated 1 = Output data is updated (When this bit is set, it remains set for the remaining session until the Anybus is either reset or the HSBY state changes to "Not Connected")
5	-	Not used; set to zero.
6	-	Not used; set to zero.
7	HS	This bit indicates that the Hot Standby functionality is enabled. 0 = HSBY disabled. Module operates as stand-alone Master or HSBY-state equals "Not Connected". 1 = HSBY enabled.

Active and Passive HSBY Master Status Data High Byte - Bits 8 to 15

The "COM-bit" would be set to one when the counterpart is present on the network (that is, ping sequence is successfully running). The other bits (8 to 14) are considered valid only when the "COM-bit" equals one.

If the ping sequence is terminated the "COM-bit" and all other bits (8 to 14) are set to zero.

All bits would be set to zero at power-up, and when the HSBY-state equals "Not connected".

HSBY Nr of Active Slaves Byte

This byte will indicate the number of slaves accessible to the local Master. Based on this information, the high-level system could switch to the Master that recognizes the most slaves.

Note that this register only contains valid data when the "HS-bit" in register "HSBY Local status" is set.

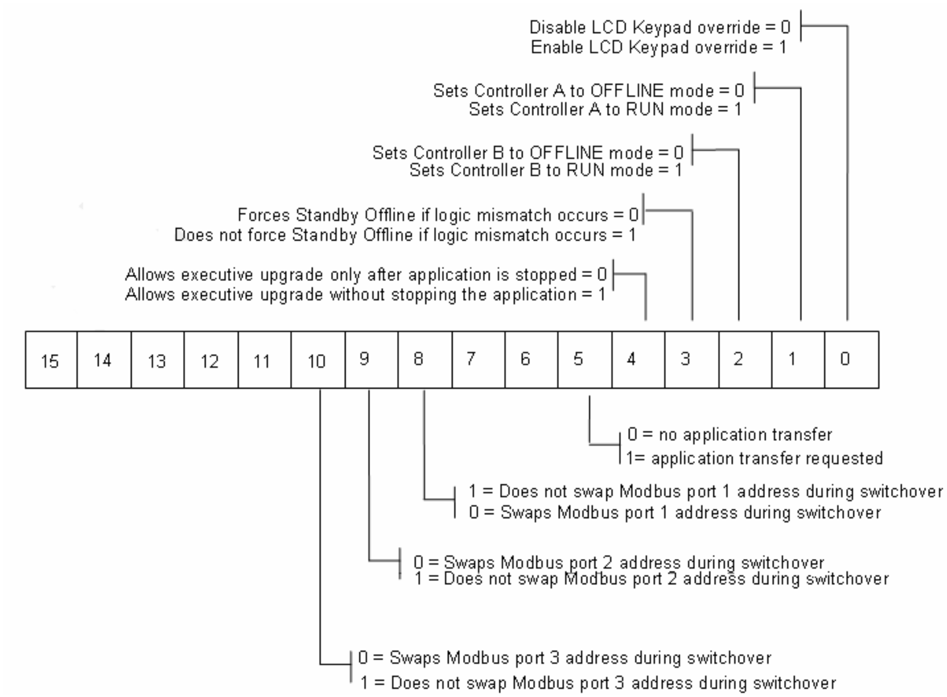
HSBY Nr of Passive Slaves Byte

This byte will indicate the number of slaves accessible to the counterpart (that is, a reflection of the counterpart's "HSBY Nr of local slaves" register). Based on this information, the high-level system could switch to the Master that recognizes the most slaves.

Note that this register only contains valid data when the "COM-bit" in register "HSBY Remote status" indicates that the counterpart is present. When the "COM-bit" is cleared, this register would be set to zero.

PLC HSBY Command Register

The following illustration identifies the operating options provided by the Command Register (%SW60). The Command Register defines the operation of the Hot Standby application. That means both the Primary and Standby. Therefore the Command Register is transferred to the Standby PLC each scan. As a result, any changes made to the Command Register on the Standby PLC will have no effect since the value transfer from the Primary side will overwrite it.



%SW60 Hot Standby Command Register

%SW60.0: This bit, if set to 1, allows the Command Register RUN status of the PLC to be set through the LCD Keypad.

Warning: If the keypad override is enabled while the Hot Standby system is running, the Primary PLC will immediately read bits 14 and 15 to determine its own state and the state of the Standby. If both bits are set to 0, a switchover will occur and the former Primary will go offline. The new Primary will continue to operate.

%SW60.1: Setting this bit = 1 will put PLC A in RUN mode. Setting the bit = 0 will put PLC A in OFFLINE mode. This bit takes effect only if bit 16 is set = 1.

%SW60.2: Setting this bit = 1 will put PLC B in RUN mode. Setting the bit = 0 will put PLC B in OFFLINE mode. This bit takes effect only if bit 16 is set = 1.

%SW60.3: Setting this bit = 0 will force the Standby PLC offline if a logic mismatch is detected. Logic mismatch is defined as either the MID, LID or CID being different on Primary and Standby sides. Setting this bit = 1 will allow the Standby PLC to continue to operate normally even if the MID is different on the Primary and Standby.

%SW60.4: Setting this bit = 1 allows the executive to upgrade on the Standby without having to stop the application. This means the Hot Standby system is allowed to operate with different versions of the OS running on the Primary and Standby. This option is provided to allow upgrades to be done without shutting down the process. Clearly, the Standby PLC must be stopped to do the executive upgrade, but it will be able to operate as a valid Standby when started again.

%SW60.5: Setting this bit = 1 commands the standby station to initiate an application transfer. That function is not required in UNITY V1.

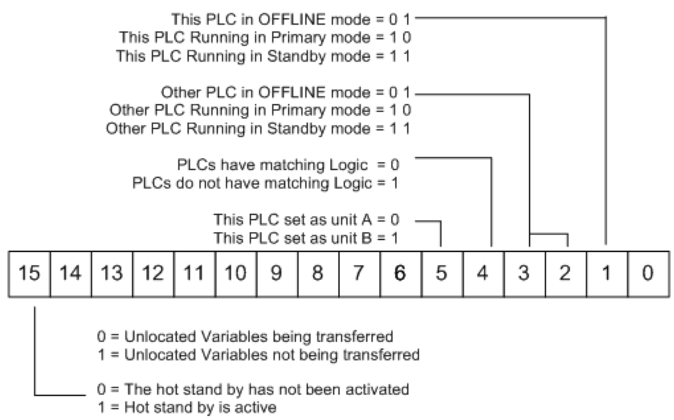
%SW60.8: If this bit is set = 1 the Modbus address on port 1 will be swapped when a switchover occurs. Swapping means to change address by ± 128 to keep the address in the range of 1 to 247. The purpose of this is to allow the P-unit of an HMI to always use the same address to connect to the Primary or Standby/Offline PLC.

PLC HSBY Status Register

The Status Register provides user information relative to the state of the two PLCs in the Hot Standby system. The Status Register is %SW61.

Both the Primary and Standby/Offline PLCs have their own copies of the Status register. The Status register is not transferred from Primary to Standby each scan. Each PLC must maintain its local Status Register based on the regular communication between the two PLCs.

The following illustration identifies the operating options provided by the Command Register.



%SW61 Hot Standby Status Register

The following gives additional detail on the various parts of the Status Register.

%SW61.0 to 3: These bits display the state of the local and remote Hot Standby PLCs.

%SW61.4: This bit is set = 1 whenever a logic mismatch is detected between the Primary and Standby PLCs. This means that either the MID, CID or LID is different on the two PLCs. Under this condition, if bit 13 of the Command Register is set = 0, bit 1 of the Status Register will be set = 1.

%SW61.5: This bit identifies the order reported by the copro at start time depends on the range of the MAC addresses.

- If the A/B designation is A, then bit 5 will be set = 0.
- If the A/B designation is B, then bit 5 will be set = 1.

%SW61.14: If set = 1 it indicates that a logic mismatch has been detected that disallows Unlocated Variables to be transferred from Primary to Standby. This feature was canceled for UNITY V1.1 because it was determined that a switchover with a partial application context posed too great a hazard.

%SW61.15: If set = 1 it indicates that the Copro device is set up correctly and working.

How the PTQ Module Detects a Switchover

The PTQ module and Master bus scanner will react within 100 ms to 300 ms when the PLC changes its state. The PTQ module reads the PLC status in word 102 (%SW61) of the configuration table with every scan.

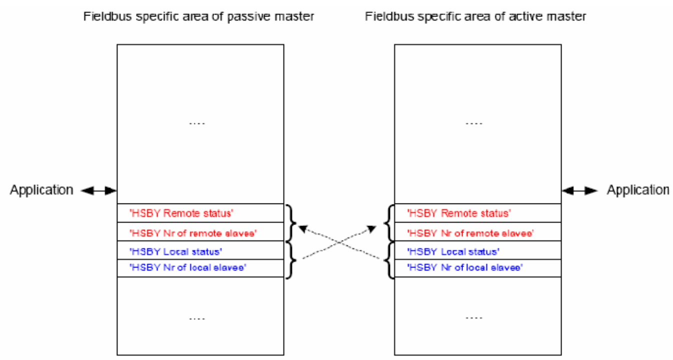
Crossed Status Information

The ping message communicates status information between the active and remote (passive) Masters, and vice versa.

A part of this "crossed status information" (CSI) is presented in the "HSBY Remote status HSBY Nr of remote slaves" registers. These registers are updated every time new status information is received from the counterpart. Polling for new status information takes place every TA ms.

The PTQ module application forwards this information to the Quantum processor, which, in turn, determines by user application whether or not a switchover should be carried out.

The following illustration shows how CSI is communicated between the two Masters and how it is displayed in the Field bus specific area of the DPRAM.



Slave Status

Both Masters inform the counterpart of its slave status as a part of the CSI.

The slave status information includes two parts, the SO-bit in register "HSBY local status", and register "HSBY Nr of local slaves". Based on this information, a high-level system could switch to the Master that recognize the most slaves.

- The "SO-bit" indicates if the Master recognizes any of its assigned slaves as offline.
- Register "HSBY Nr of local slaves" indicates how many slaves are online.

How this information is determined depends on if the Master module operates as active or remote (passive) Master:

- The local (active) Master would use the "state-report" information available from the Siemens stack. A slave is considered to be online when it participates in the cyclic Data Exchange sequence.

Note: This means that a slave that reports "Prm-fault" or "Config-fault" is considered to be offline even if it is physically accessible to the Master.

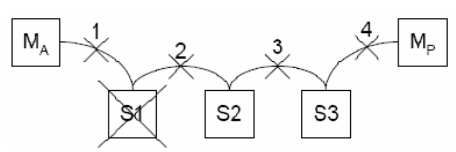
- The remote (passive) Master would ping all of its assigned slaves with FDL_Status telegrams once every time it holds the token. A slave is considered to be online when it responds to the telegram.

During a switchover, the slave status information would be reassembled. This means that the value of "HSBY Nr of local slaves" and "SO-bit" might dip for a short while until the "new" local (active) Master has detected its slaves.

Note: This feature affects the performance of the entire network, since the remote (passive) Master must query all slaves when it holds the token, thus increasing the token rotation time.

Example:

The following illustration shows a network with three slaves and two Masters. MA is the local (active) Master and MP the passive one.



The table lists the values of the slave status information under different scenarios.

Note: The only time the slave status information is forwarded to the counterpart is when a slave is disconnected or fails. In all other cases, the communication link is broken so the information would not reach the counterpart.

Slave Status Example

	M_A		M_P	
Type of Failure	**Local "SO-bit"**	**HSBY Nr of local slaves**	**Local "SO-bit"**	**HSBY Nr of local slaves**
No errors	1	3	1	3
Cable is cut at 1.	0	0	1	3
Cable is cut at 2.	0	1	0	2
Cable is cut at 3.	0	2	0	1
Cable is cut at 4	1	3	0	0
S1 is disconnected	0	2	0	2

Database Mismatch

Both Masters inform the counterpart of its database CRC value as a part of the Crossed Status Information.

Based on this information, both Masters compare their own CRC values with the one received from the counterpart to determine the state of the "DB-bit" in the "HSBY Local status" register.

It does not make sense to perform the CRC check in a cyclic manner. Instead, it would be enough to do it once when the counterpart has just been detected.

FDL Layer Access

Ping messages, slave status messages, and DPV1 status messages are communicated over the PROFIBUS network via the FDL-layer of the Siemens stack (a.k.a. AMPRO2).

A unique channel, reserved only for ping and slave status messages, is opened to the FDL-layer.

This would ensure that no "DPV1 class 2"- or "Live List" requests, which also use the FDL-layer, are interfering with the time-critical ping or slave status sequence.

Message type and priority

- Ping messages are sent as high-priority SRD-repeat telegrams
- Slave status messages are sent as low-priority FDL_Status-repeat telegrams
- DPV1 status messages are sent as low-priority SRD (single) telegrams

SAP number

- SAP 10 is used for ping messages
- SAP 11 is used for DPV1 status messages
- No SAP is defined for slave status messages (FDL_Status)

The following table lists FDL services that are needed for the HSBY functionality. The two right-hand columns indicate which services are used by the active and remote (passive) Masters.

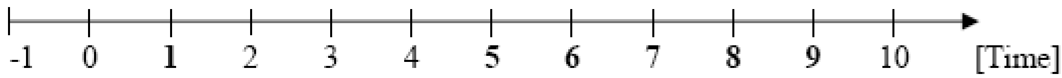
FDL Service	Description	Local (Active) Master	Remote (Passive) Master
Open-Channel	Establish a channel to the FDL-layer	X	X
SAP-Activate	Opens up a Service Access Point at the responder	X	X
IND-Resource-Provide	Makes a resource available for single SRD-requests		X
IND-Resource-Repeat-Provide	Makes a resource available for SRD-repeat-requests	X	
Reply-Update-Multiple	Updates the SRD-response data	X	
FLC-Repeat-Exchange	Reads out the latest SRD-request data	X	
SRD	Sends an SRD-request (used for DPV1 status messages)	X	
SRD-Repeat	Sends an SRD-request cyclically (used for Ping messages)		X
MAC-Repeat-Exchange	Exchanges SRD-request or SRD-response data		X
MAC-Reset	Resets ASPC2 during switchover		X
FDL_Status-repeat	Sends an FDL_Status request cyclically (Used for Slave status messages		X

SAP 10 is used for the DSAP of the responder (local (active) Master) as well as for the SSAP of the initiator (remote (passive) Master).

The SRD-request is sent as a "low priority" FDL-message.

7.3.2 Switchover Timeline

The overall switchover time for a Hot Standby system depends on several sequential events, which are outlined in the timeline below. The elapsed time between some events is not static but highly dependent on the baud rate; these events are indicated with **bold text**.



-1	Last DataExchange request before local (active) Master failure (that is, last re-triggering of "slave watchdog").
0	Local (active) Master failure.
1	Remote (passive) Master detects counterpart failure.
2	Remote (passive) Master indicates the situation by clearing the COM-bit in Fieldbus specific area.
3	Application of the remote (passive) Master detects that the local (active) Master has failed and forwards this information to a higher-level system for further processing.
4	Application of the remote (passive) Master initiates a switchover based on the decision from a higher-level system.
5	Remote (passive) Master performs the switchover (MAC reset with new TS)
6	Local (active) Master (former passive) issues an initial GlobalCtrl request (that is, first PROFIBUS telegram after reinitialization of MAC).
7	Local (active) Master issues a SlaveDiag request to all slaves (first re-triggering of "slave watchdog" after switchover).
8	Local (active) Master issues a DataExchange request to all slaves (second re-triggering of "slave watchdog").
9	Local (active) Master sets the OD-bit in the Fieldbus specific area indicating that there is fresh data in the output data area for the application to read.
10	Application of the local (active) Master detects that the OD-bit is set and accesses the output area.

7.3.3 Bus Parameters

Some bus parameters need to be altered to allow for Hot Standby functionality.

Slave Watchdog Time (WD_Factor1&2)

The watchdog time of the slaves would have to be increased to allow for the maximum time it takes for the HSBY system to perform a switchover.

Token Rotation Time (TTR)

To allow for Master-to-Master communication, TTR would have to be increased by a factor of 2 relative to a single-Master configuration.

Highest Station Address (HSA)

HSA defines the highest address that a Master can have to be included in the token ring. For example, a Master with address 35 will not be included in the token ring if the HSA is set to 34.

It is recommended to keep the HSA as low as possible since it affects the time it takes for a Master to enter the token ring.

Master Address (TS)

The address of the remote (passive) Master would be derived by subtracting one (-1) from bus parameter "TS" (This Station). For example, if TS in the database equals 2, the remote (passive) Master would use address 1. If TS in the database equals 0 it would use address 125.

During switchover, when the remote (passive) Master becomes active, the address would equal TS. Please note that this is handled by the Master module internally; the database downloaded to the two Masters must be identical.

It is preferable to keep the address of the remote (passive) Master as low as possible since it affects the time it takes to detect a faulty local (active) Master.

A problem with setting TS = 0 is that the remote (passive) Master would be assigned address 125, which in turn means that HSA must be set to its maximum value of 125 to allow the remote (passive) Master to enter the token ring. If not, the Master module will reject mailbox END_INIT (Fault info = 6 and Additional Fault info = 19).

This special case is handled by the PCB Master configuration, which will not accept TS = 0 if the "HSBY" bus parameter profile is selected.

7.3.4 HSBY Master GSD File

No changes to the original file PTQ_18F0.GSD would be necessary.

7.3.5 LED Indicators

- The Master status LED would flash red, indicating that the Master module is operating as a remote (passive) Master.
- When the Master module operates as a remote (passive) Master, the COM-status LED signals the same information as it does for a local (active) Master, or a combined C1/C2 Master.

7.3.6 Unsupported Functions

- When the Master module operates as a passive HSBY Master, the following mailboxes are not allowed and will be rejected with "Fault information code" 0x00FD.
  - *FB_APPL_SET_SLAVE_MODE*
  - *FB_APPL_GET_MASTER_DIAG*
  - *FB_APPL_GET_SLAVE_DIAG (internal request)*
  - *FB_APPL_MSAC1_PROFID_V3_PARAM*
- When the Master module operates as HSBY Master (passive or active), "Class 2 Master only" mode is not allowed. Refer to FB_INIT (special functions, bit 5).

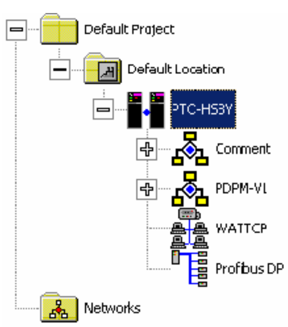
7.3.7 ProSoft Configuration Builder (PCB) HSBY Option Functionality

ProSoft Configuration Builder is required to set up, control, and identify matched Primary/Standby HSBY PTQ-PDPMV1 module sets.

Considerations should be identified for uploading and downloading of module configuration data as if the dual modules were a single module, whereas upon downloading the configuration files, files are sent to both PTQ modules automatically.

The PCB will not be required to be concerned with Primary or Standby initially. However, the PCB will be required to know which module is Primary in order to perform accurate online monitoring (Standby Master modules will not be communicating with the slave devices).

A new icon is created to indicate the PROFIBUS configuration is in Hot Standby mode. The icon is displayed as a double PTQ module as shown in the following illustration.



PCB Master Configuration Software

The configuration software should readily make available HSBY diagnostic and status information in the bus view configuration mode. A separate and single Master GSD file for a HSBY Master should be created and used.

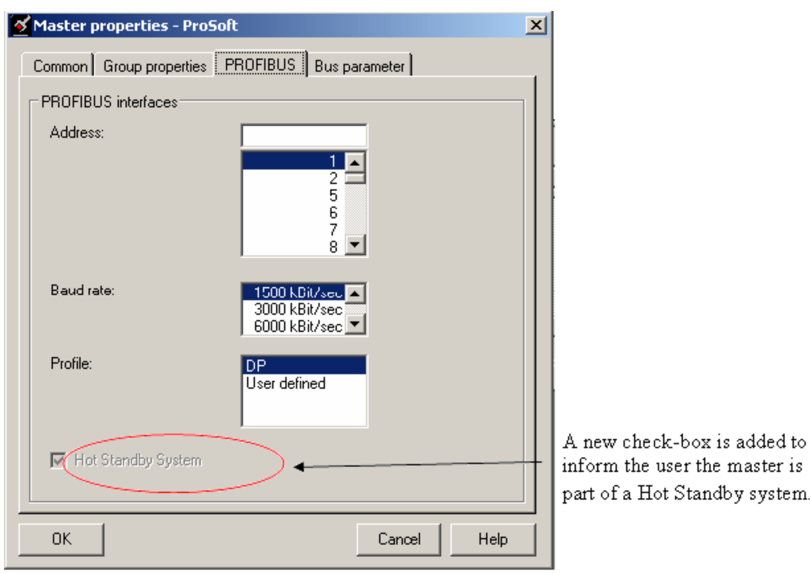
For simplicity, clarity, and synchronization purposes, the configuration software will allow for a single network configuration to be utilized for the HSBY PTQ Master modules. For example, the user will configure a single network and download the configuration; the software will download the configuration files to both units and indicate that the modules are synchronized and whether the download was successful for both modules.

Downloading the module's configuration files will be transparent to the user. Both modules are required to be connected to the Ethernet network, and upon download, PCB will automatically download configurations to both modules. If the download process is unsuccessful, the PCB will inform the user, and the modules may have different CRC database values determined by the PROFIBUS FDL ping message.

A Master in the configuration process is initiated as neutral (offline or not connected) until it reads status information from the PLC. Then it will operate in Active, Passive, or Offline mode.

Bus Parameters

The PROFIBUS Master bus parameters for a Hot Standby project are automatically adjusted for best performance by the software. A new checkbox in the *PROFIBUS* tab in the *Master properties* dialog box, called "Hot Standby System", has been added to identify the Master as a HSBY Master set. The same Bus Configuration will be downloaded to both Masters, ensuring that both the active and the remote (passive) Master get exactly the same configuration and checksum values.



The Watchdog Time, Delta Ttr and other parameters can only be edited by selecting the User-Defined profile.

When the "Hot Standby System" checkbox is marked, the PCB Master configuration tool follows the rules below:

Parameter Calculation

Ttr is to be doubled in order to support one "passive" and one "active" Hot Standby Master on the same network. Note that the Watchdog value also must be recalculated because of the doubled Ttr.

To allow a switchover the Watchdog value that is calculated from the Ttr must be increased. Two new parameters are introduced: HsbyWatchdogFactor and Host Delay Time. The watchdog is calculated according to the following equation:

Watchdog = (Calculated watchdog * HsbyWatchdogFactor) + Host Delay Time

The HsbyWatchdogFactor for different baud rates is defined in the following table:

Baud Rate	XML Keyword	Value
9.6 kBaud	HsbyWdFactor_9_6k	1
19.2 kBaud	HsbyWdFactor_19_2k	1
45.45 kBaud	HsbyWdFactor_45_45k	1
93.75 kBaud	HsbyWdFactor_93_75k	1
187.5 kBaud	HsbyWdFactor_187_5k	1
500 kBaud	HsbyWdFactor_500k	1
1.5 MBaud	HsbyWdFactor_1_5M	2
3 MBaud	HsbyWdFactor_3M	2
6 MBaud	HsbyWdFactor_6M	2
12 MBaud	HsbyWdFactor_12M	2

The Host Delay Time is set to 300 ms in the PCB Master configuration.

Invalid Master addresses:

1. The Master address must not equal 0.
2. The Master address must not equal any assigned slave address + 1.

Invalid Slave addresses:

1. A slave address must not equal the assigned Master address - 1.

7.4 PTQ-PDPMV1 Master Bus Properties for Use of a P&F DP/PA Segment Coupler

7.4.1 PROFIBUS DP Time Behavior

The segment coupler supports both cyclic and acyclic data exchange of the PROFIBUS DPV1. Cyclic communication services are performed based on a specific time grid, which is referred to as the bus cycle time.

A bus cycle time (or cycle time for short) is the worst-case-scenario time required to transfer input data from a PROFIBUS slave to the PROFIBUS Master, or output data from the PROFIBUS Master to the slave. All data that is to be updated cyclically is automatically updated in the common data area by the PROFIBUS PA Master.

The cycle time required depends on the volume of data that is transferred via the PROFIBUS PA channel. From the point of view of the PROFIBUS DP, the segment coupler represents a multi-slave. If the PROFIBUS DP Master sends a request to a slave address existing at the segment coupler, the gateway answers the request directly with the data that is stored in the common data range. Consequently, the PROFIBUS DP Master does not need to wait for the PROFIBUS PA slave to respond. Therefore, the cycle time of the entire system is:

```
tCycle = tCycle_PA-channel + tCycle_DP
```

The time tCycle_PA-channel can be estimated as follows:

```
tCycle_PA-channel = 10 ms + n*10.5 ms + 0.256 ms*(LE + LA)
```

where n = the number of PROFIBUS PA slaves

LE = total number of input bytes of all PROFIBUS PA slaves on the channel

LA = total number of all output bytes of all PROFIBUS PA slaves on the channel

The time tCycle_DP can be estimated as follows:

```
tcycle_DP = TBit * n * 500 + 11*TBit*(LE + LA)
```

where n = the number of PROFIBUS DP slaves

LE = total number of input bytes of all PROFIBUS slaves

LA = total number of output bytes of all PROFIBUS slaves

TBit = bit time = 1/transfer rate

For the time tCycle_DP a safety add-on of 10% should be included in the calculation in accordance with the PROFIBUS User Organization.

The equation above applies given the following conditions:

- The PROFIBUS DP is operated as a mono-Master system, i. e. there is only one Master on the PROFIBUS DP. If you want to use a multi-Master system, the token hold time and the corresponding pause times of the additional Masters must be added to the total.
- Only acyclic data exchange takes place. If the Master is also supposed to transfer acyclic telegrams, the time required for acyclic communication must be added into the total.

7.4.2 Commissioning of Communication with the SK1 Segment Coupler

Since the SK1 segment coupler works transparently, PROFIBUS PA stations are treated like PROFIBUS DP stations by the control system. This also applies to commissioning. To make it possible for the control system to exchange station data with a PROFIBUS, the GSD file of the station must be integrated into the configuration tool of the control system.

The SK1 segment couplers receive the PROFIBUS DP telegram, convert it simultaneously, and transmit it on the PROFIBUS PA side. The PROFIBUS PA slave responds immediately to this telegram. The response telegram is received by the segment coupler, is again converted simultaneously, and is transmitted on the PROFIBUS DP side as a slave response.

Note: To make it possible for data exchange between the PROFIBUS DP and PROFIBUS PA to work correctly, it is essential for the PROFIBUS DP transfer rate to be set to 93.75 kBd.

The time lapse between the Master call and the slave response is limited. Since the PROFIBUS PA is working at a lower transfer rate than the PROFIBUS DP, the standard settings of the bus parameters of the PROFIBUS DP Class 1 Master must be changed.

Note: If the bus parameters are not changed, no data exchange is possible between the PROFIBUS DP Master and the PROFIBUS PA slave.

PROFIBUS DP configuration tools do not always make it possible to set all bus parameters indicated in the following table. There are, however, dependencies that the configuration tool uses to calculate the dependant variable from the adjustable parameter value.

The following graph shows the standard settings of these parameter values for operation with the non-modular segment coupler:

Parameter	**Value**	**Description**
Baud rate [kBit/s]	93.75	PROFIBUS DP transfer rate
TSL[tbit DP]	4095	Slot-Time
Min TSDR [tbit DP]	22	Min. Station-Delay-Time
TID2 [tbit DP]	1000	Idle2-Time
Max TSDR	1000	Max. Station-Delay-Time
TID1 [tbit DP]	145	Idle1-Time
TSET [tbit DP]	55	Setup-Time
TQUI [tbit DP]	0	Quiet-Time
G	10	Gap-Factor
HSA	126	Highest-Station-Address
max_retry_limit	1	Repetitions in event of failure

The PROFIBUS DP transfer rate is fixed at 93.75 kbit/s for non-modular segment couplers. The bit time is thus tbit DP = 10.67 µs. The PROFIBUS PA transfer rate is fixed at 31.25 kBit/s; while the bit time is tbit PA = 32 µs.

The slot-time setting of the table above works if the total of input data bytes plus output data bytes < 253 bytes. If the data volume of a PROFIBUS PA slave exceeds this value, the slot-time should be set to 7192 [tbit DP]. For purposes of optimization, the ideal setting can be calculated as follows:

```
TSL > 13*(LS + LR) + 3*TSDR + 630
```

where LS is the number of data bytes in the Master_Request telegram and LR is the number of data bytes in the Slave_Response telegram.

Time TSDR refers to the actual time lapse that is counted between the Master_Request and the Slave_Response. This is typically 75 tbit DP.

Other possibilities for optimization in terms of cycle times are available through the Idle1-Time, Idle2-Time and the HSA setting.

The **Idle1-Time** (TID1) is an idle time to be observed by the Master between a response telegram and the prompt telegram following it. The TID1 parameter cannot be set directly in many tools. To optimize the Idle-Time in spite of this, the Setup-Time TSET parameter or, if it cannot be adjusted either, the Quiet-Time TQUI parameter must be adjusted. The Idle-Time is calculated as follows:

```
TID1 = 2*TSET + TQUI + 35*tbit DP
```

TID1 depends on the maximum response time (not to be confused with the Station-Delay-Time TSDR) of all PA bus stations. In the table above, values are indicated for TID1 and TSET corresponding to the current PROFIBUS guidelines. In some circumstances, older PROFIBUS devices that do not yet work with response times in accordance with "PROFIBUS DP Expansion for EN 50170 (DPV1)", may provoke telegram repetitions. If this behavior occurs, you can increase Idle1-Time as an emergency measure. This will, however, increase the bus cycle time.

The **Idle2-Time** (TID2) is the idle time between an SDN telegram (send data with no acknowledge) and the following call telegram. These SDN telegrams are used for global control services (SYNC, UNSYNC, FREEZE, UNFREEZE, and so on). This value should be set to 1000 tbit DP.

If TID2 cannot be set directly, you can use the parameter max TSDR. If max TSDR is greater than TID1, as is shown in the table, the value of max TSDR is automatically used for TID2.

Highest-Station-Address (HSA): A PROFIBUS Master queries the status of all stations cyclically up to the address value HSA (1 telegram per cycle). As soon as one station at an address lower than HSA does not respond, (for example because it is not connected) the relatively long Slot-Time TSL expires until the next call telegram is transferred.

If it can be ensured that a station is present at every address including the HSA, this Slot-Time can be avoided.

Information for Determining the Watchdog Time TWD

PROFIBUS devices are able to activate a watchdog mechanism that monitors each time interval of cyclic calls (data exchange) to make certain the PROFIBUS Master is still active. The time measurement takes place in the PROFIBUS slave.

If the watchdog is activated and the time TWD (Watch Dog Time) since the last cyclic call expires, the device leaves cyclic data exchange, goes into the original state (Wait_prm) and sets the outputs to the secure state.

The value of the time TWD and the activation of the watchdog are transferred in the parameterized telegram from the PROFIBUS Master to the PROFIBUS slave at startup (transition to the data exchange). In general, dimensioning of the time TWD is used-specific (not device-specific, not in the GSD). The value is bounded below by cycle times.

As a rule, the configuration tool is used to enter the time TWD. There are configuration tools in which the watchdog time is set 1x per PROFIBUS Master and others in which the watchdog time is set individually for each PROFIBUS PA station. This does not change with the value of the watchdog time.

For many tools, the time TWD is automatically calculated based on the cycle time of the Master with a corresponding baud rate.

At higher baud rates on the PROFIBUS DP side (for example 12MBd), cycle times on the PA side may be longer by a factor of 300. If parameters were set directly for a PROFIBUS PA device at a baud rate (DP) for a time calculated for a higher Master TWD, it would generally be less than the PA cycle and the device would not enter into the data exchange.

Behavior of Segment Coupler 2

To ensure reliable operation of the PROFIBUS the following bus parameters should be used:

- Transfer rate 45.45 kBd ... 12 MBd
- Watchdog time TWD = 5 s
- PROFIBUS DP Standard

Note: If there is a large number of PROFIBUS PA stations per channel of the SK2 segment coupler, the watchdog time TWD should be verified. The limit value is about 32 stations, but depends on the volume of data to be transferred.

Determining the Parameter TWD

The watchdog time for the value above is a number based on experience, in other words one that usually works. If it turns out that the watchdog time is too long (slaves are not switching into the secure state quickly enough) or too short (slaves are switching into the secure state without the Master ever having failed) this must be factored into the calculation. Depending on the configuration tool you are using, you can

- set parameters for only one watchdog time TWD for the entire PROFIBUS system. In this case, the greatest delay time must be used as the basis for determining TWD.
- set parameters for a watchdog time TWD for each individual slave.

The time TWD that is set (parameter) must be greater than the longest delay time TV_max that will occur. This is composed of a number of elements as follows:

```
TV_max = TCycle_DP + TCycle_PA_channel
```

where:

TCycle_PA_channel = the cycle time of the PROFIBUS PA channel

TCycle_DP = Cycle time of the PROFIBUS DP

Note: Pepperl+Fuchs recommends three times the PROFIBUS PA cycle time.

7.4.3 Details for calculating the TWD parameter

The PA cycle time TCycle_PA_channel depends on

1. The number n of stations on a channel
2. The effective data length LΣ (average of the total of input and output data of all devices [number of bytes (unit less)]):

Cycle time can be calculated in an approximate manner as

TCycle_PA_channel = n * (0.256 ms * LΣ + 12ms) + 40 ms

Note: For more information on calculating cycle time and other related data for the SK1 or SK2, refer to PEPPERL & FUCHS, Instructions Manual Segment Coupler SK1 and SK2.
The above information references PEPPERL & FUCHS, Instructions Manual Segment Coupler SK1 and SK2. For additional information about the SK1 or SK2, please contact PEPPERL & FUCHS.

8 Diagnostics and Troubleshooting

In This Chapter

The module provides information on diagnostics and troubleshooting in the following forms:

- LED status indicators on the front of the module provide general information on the module's status.
- Status data contained in the module can be viewed through the Configuration/Debug port, using the troubleshooting and diagnostic capabilities of *ProSoft Configuration Builder (PCB)*.
- Status data values can be transferred from the module to processor memory and can be monitored there manually or by customer-created logic.

8.1 Basic Troubleshooting Steps

1 Verify that the module is installed correctly, and is communicating with the processor.
2 Install the most current version of ProSoft Configuration Builder.
3 Note the color and behavior of the LED Status Indicators (lights) on the front panel. Refer to the tables in the following section for examples.

- The Active light should be ON whenever the module is communicating with the processor over the backplane. A processor must be positioned on the main rack and powered up. If this light does not come ON, either the module or the processor may be hung or offline. Power cycle the PTQ module and the PLC processor.
- The Configure light should be OFF. If the light is ON, it is still possible to communicate with the module, but the module and Master are prevented from going into RUN mode. There are two conditions that cause the Configuration light to be on:

  First: The configuration files are missing or corrupt. Configuration files are stored on the Compact Flash card inserted in the rear of the module. Remove the Compact Flash card and transfer the configuration files from your computer to the card using a card reader.

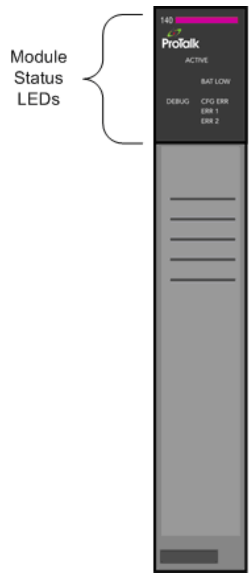
  Second: The checksum values for the PROFIBUS network configuration file have changed, but the processor did not acknowledge the configuration change by returning the correct checksum values to the module. Re-import the function block file using the procedure in the following section.

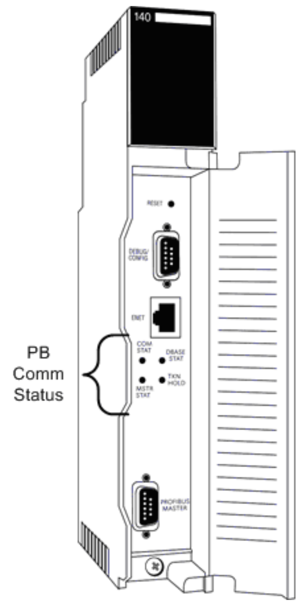
> **Special Note:** Transferring Configuration Data to Replacement Module. All module configuration data (including PTQ, PROFIBUS Network, and Ethernet) are stored on the Compact Flash in the PTQ module. Should a module failure occur, it is a simple matter of moving the Compact Flash from the old module to the replacement module in order to transfer the configuration data.

8.2 LED Indicators: Front of PTQ Module

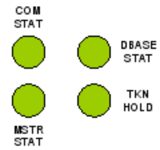
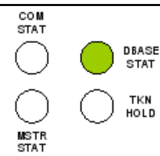
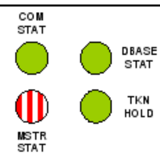
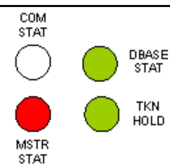
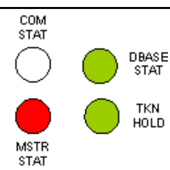
The LEDs indicate the module's operating status. The module has two sets of LED Indicators:

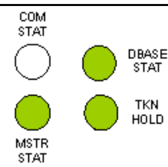
- PTQ Module Status LEDs on the front of the module near the top
- PROFIBUS Master Status LEDs behind the door on the front of the module.





The following table shows some of the possible status indicators:

	Processor Status	PTQ Module Status	PROFIBUS Master Status	PROFIBUS Master Status LED Description
Normal Operation	RUN (ON)	ACTIVE (ON)		**COM STAT (GREEN/Solid or Flash):** Master is communicating with slaves (Solid GREEN) or at least one (Blinking). **DBASE STAT (GREEN):** PROFIBUS has been configured. **TKN HOLD (GREEN):** PTQ is holding the PROFIBUS token. **MSTR STAT (GREEN):** Master is in OPERATE mode.
PTQ module not communicating with processor PTQ module rebooted for DEBUG only (HSBY)	RUN (ON) or STOP	ACTIVE (OFF)		**COM STAT (OFF):** Master is not communicating with slaves. **DBASE STAT (GREEN):** PROFIBUS has been configured. **TKN HOLD (OFF):** Master does not have the token and is inactive. **MSTR STAT (OFF):** Master is inactive.
HSBY: After the hot swap of the module, the PTQ module is correctly SET as the remote (passive) Master.	RUN (ON)	ACTIVE (ON)		**COM STAT (GREEN/Solid or Flash):** Master is communicating with slaves (Solid GREEN) or at least one (Blinking). **DBASE STAT (GREEN):** PROFIBUS has been configured. **TKN HOLD (GREEN):** PTQ is holding the PROFIBUS token. **MSTR STAT (RED Blinking):** Master is in remote (passive) Master mode.
PTQ PROFIBUS Master is stopped	RUN or STOP	ACTIVE		**COM STAT (OFF):** Master is not communicating with configured slaves. **DBASE STAT (GREEN):** PROFIBUS has been configured. **TKN HOLD (GREEN):** PTQ is holding the PROFIBUS token. **MSTR STAT (RED):** Master is in STOP mode.
CPU is stopped	STOP	ACTIVE		**COM STAT (OFF):** Master is not communicating with configured slaves. **DBASE STAT (GREEN):** PROFIBUS has been configured. **TKN HOLD (GREEN):** PTQ is holding the PROFIBUS token. **MSTR STAT (RED):** Master is in STOP mode.

	Processor Status	PTQ Module Status	PROFIBUS Master Status	PROFIBUS Master Status LED Description
CPU is running	RUN	ACTIVE		**COM STAT (OFF):** Master is operating, but there is no communication with slaves, or PROFIBUS cable is disconnected. **DBASE STAT (GREEN):** PROFIBUS has been configured. **TKN HOLD (GREEN):** PTQ is holding the PROFIBUS token. **MSTR STAT (GREEN):** Master is in OPERATE mode.

8.3 Module Status Indicators

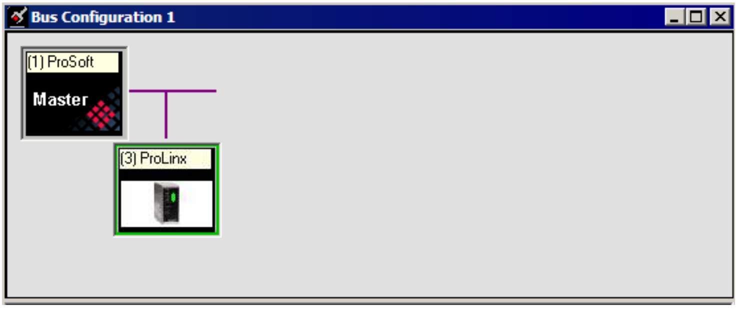
Indicator	Color	Status	Indication
DEBUG	Green	ON	Configuration/Debug Port is active
		OFF	Configuration/Debug Port is inactive
CFG ERR	Red	ON	Configuration Error: This LED is illuminated when the PROFIBUS and module CRC values do not match between input/output blocks. The module expects that the correct CRC values will be copied from the processor to the module, otherwise the module will be placed in STOP mode (MSTR STAT LED = RED) and the CFG ERR LED is illuminated to warn the user. Verify that the values match the values generated with the Calculate Checksums button in ProSoft Configuration Builder. A function block is provided for Concept to synchronize input CRCs with Output CRCs. This LED will also be illuminated if one of the module's configuration files is missing. This LED will illuminate if the module is placed in a slot other than the one for which it was configured. For HSBY, the LED will illuminate if the active and remote (passive) Masters' configuration files do not match. After downloading new configuration file to the PTQ module For more information on interpreting this error, refer to Word Offset 59 in Slave List Structure.
		Blinking	Major error occurred. Please contact ProSoft Technical Support. This error is typically caused by a hardware incompatibility after firmware upgrade.
		OFF	Normal operation (configuration OK)
ERR1 and ERR2		ON	The HSBY processors tried to perform a switch while a Master was in STOP mode. This is a critical error and the module can be soft booted from within PCB diagnostics or the module may be reseated.
Active	Green	ON	The LED is on when the module is able to communicate over the backplane.
		OFF	The LED is off when the module is unable to speak with the processor. The processor is either absent or not running.
		Flashing	This LED flashes on the remote (passive) Master
BAT Low	Red	OFF	The battery voltage is OK and running.
		ON	The battery voltage is low or the battery is not present. The battery LED will illuminate briefly upon the first installation of the module or if the unit has not had power for an extended period of time. This behavior is normal; however, should the LED come on in a working installation, please contact ProSoft Technology.
E-Link	Green	ON	The Ethernet port is connected to the TCP/IP network.
E-Data	Green	ON	Data is being transferred through the Ethernet port.

8.4 PROFIBUS Master Indicators

LED	State	Description
MSTR STAT	GREEN	OPERATE mode
	GREEN-Flashing	CLEAR mode
	RED	STOP mode
	OFF	Offfline
DBASE STAT	GREEN	Database OK
	GREEN-Flashing	Database download in progress
	RED	Invalid database
	OFF	No databases have been downloaded
COM STAT	GREEN	Data exchange with all configured slaves
	GREEN-Flashing	Data exchange with at least one of the configured slaves
	RED	Bus control error (possible bus short circuit or configuration error)
	OFF	No data exchange with any configured slave
TKN HLD	GREEN	The module has the token
	OFF	The module does not have the token
ALL LEDs	RED	Fatal error
	OFF	HSBY processor is stopped and Master is held in reset state (inactive) The module is not in the configured slot

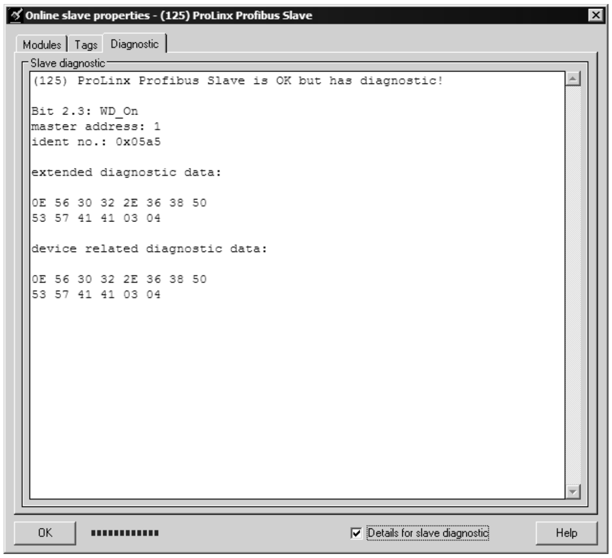
8.5 View the Online Status of the PROFIBUS Network

1 In *ProSoft Configuration Builder for PROFIBUS*, open the **ONLINE** menu, and then choose **MONITOR/MODIFY**. *ProSoft Configuration Builder* will establish communication with the PTQ-PDPMV1 module, and will indicate communication status.



- If the **SLAVE** icon in the *Bus Configuration* window has a green border, then the PTQ-PDPMV1 module is correctly communicating with the PROFIBUS slave.
- If the **SLAVE** icon in the *Bus Configuration* window has a red border, then the module is not communicating with the slave.
- If the **SLAVE** icon in the *Bus Configuration* window has a blue border, the slave is communicating with the Master, but is generating diagnostic data. To view diagnostic data for the slave, select the **SLAVE**, and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **ONLINE PROPERTIES**.

2 In the *Online Slave Properties* dialog box, click the **DIAGNOSTIC** tab, and select (check) **DETAILS** for slave diagnostic. Slave diagnostic information will appear in the *Diagnostic* window. Refer to the documentation for your PROFIBUS slave to determine the meaning of the diagnostic data.



8.6 Using ProSoft Configuration Builder (PCB) for Diagnostics

The *Configuration and Debug* menu for this module is arranged as a tree structure, with the *Main* menu at the top of the tree, and one or more sub-menus for each menu command. The first menu you see when you connect to the module is the *Main* menu.

Because this is a text-based menu system, you enter commands by typing the [command letter] from your computer keyboard in the *Diagnostic* window in *ProSoft Configuration Builder (PCB)*. The module does not respond to mouse movements or clicks. The command executes as soon as you press the **[COMMAND LETTER]** — you do not need to press **[ENTER].** When you type a **[COMMAND LETTER]**, a new screen will be displayed in your terminal application.

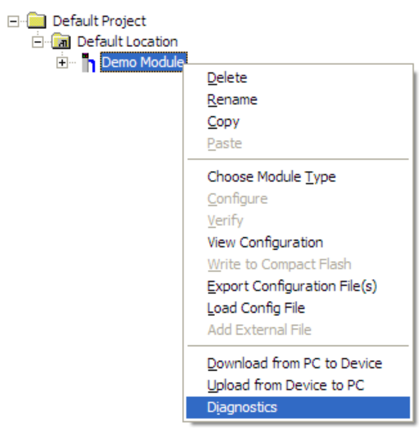
8.6.1 Using the Diagnostic Window in ProSoft Configuration Builder

Tip: You can have a ProSoft Configuration Builder *Diagnostics* window open for more than one module at a time.

To connect to the module's Configuration/Debug serial or Ethernet port

1 Start *PCB*, and then select the module to test. Click the right mouse button to open a shortcut menu.

2 On the shortcut menu, choose **DIAGNOSTICS**.



This action opens the *Diagnostics* dialog box.

3 Press **[?]** to display the *Main* menu.

```
PTQ-PDPMV1 COMMUNICATION MODULE MENU
  ?=Display Menu
  B=Block Transfer Statistics
  C=Module Configuration
  I=Input Data View
  O=Output Data View
  V=Version Information
  1=Module Status
  2=Fieldbus Data
  3=Control Registers
  4=Ethernet NIC Configuration
  @=View Network Configuration
  Esc=Exit Program and Reboot Module
```

<u>If there is no response from the module</u>

1 Verify that the cable between the module and your computer's serial or Ethernet port is connected properly. A regular serial cable will not work.

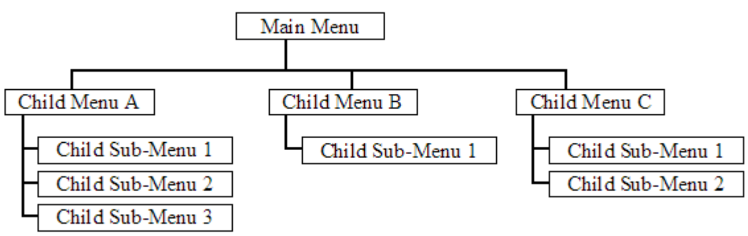
2 On computers with more than one serial port, verify that your communication program is connected to the same port that is connected to the module.

If you are still not able to establish a connection, you can contact ProSoft Technology for assistance.

8.6.2 Navigation

All of the submenus for this module contain commands to redisplay the menu or return to the previous menu. You can always return from a submenu to the next higher menu by pressing **[M]** on your keyboard.

The organization of the menu structure is represented in simplified form in the following illustration:



The remainder of this section shows the menus available for this module, and briefly discusses the commands available to you.

Keystrokes

The keyboard commands on these menus are usually not case sensitive. You can enter most commands in lowercase or uppercase letters.

The menus use a few special characters (**?**, **-**, **+**, @) that must be entered exactly as shown. Some of these characters will require you to use the **SHIFT**, **CTRL**, or **ALT** keys to enter them correctly. For example, on US English keyboards, enter the **?** command as **SHIFT** and ***/***.

Also, take care to distinguish the different uses for uppercase letter "eye" (**I**), lowercase letter "el" (**L**), and the number one (**1**). Likewise, uppercase letter "oh" (**O**) and the number zero (**0**) are not interchangeable. Although these characters look alike on the screen, they perform different actions on the module and may not be used interchangeably.

8.6.3 Main Menu

When you first connect to the module from your computer, your terminal screen will be blank. To activate the *Main* menu, press the **[M]** key on your computer's keyboard. If the module is connected properly, the following menu will appear.

```
PTQ-PDPMV1 COMMUNICATION MODULE MENU
  ?=Display Menu
  B=Block Transfer Statistics
  C=Module Configuration
  I=Input Data View
  O=Output Data View
  V=Version Information
  1=Module Status
  2=Fieldbus Data
  3=Control Registers
  4=Ethernet NIC Configuration
  @=View Network Configuration
  Esc=Exit Program and Reboot Module
```

Caution: Some of the commands available to you from this menu are designed for advanced debugging and system testing only, and can cause the module to stop communicating with the processor or with other devices, resulting in potential data loss or other communication failures. Use these commands only if you fully understand their potential effects, or if you are specifically directed to do so by ProSoft Technology Technical Support Engineers.
There may be some special command keys that are not listed on the menu but that may activate additional diagnostic or debugging features. If you need these functions, you will be advised how to use them by Technical Support. Please be careful when pressing keys so that you do not accidentally execute an unwanted command.

Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Viewing Block Transfer Statistics

Press **[B]** from the *Main* menu to view the *Block Transfer Statistics* screen.

Use this command to display the configuration and statistics of the backplane data transfer operations between the module and the processor. The information on this screen can help determine if there are communication problems between the processor and the module.

```
BACKPLANE STATISTICS:

DATA TRANSFER CONFIGURATION:
  I/O CONFIGIGURATION:
    0x Bit Count   : 256      0x Word Count : 16
    1x Bit Count   : 256      1x Word Count : 16
    3x Word Count  : 1024     4x Word Count : 1024
    Battery Status : GOOD
  BLOCK COUNTS:
    Read  : 12333   Write : 12333   Error   : 0

HOT-STANDBY:
  Value   = 800E (%SW61 in Hex) [1000 0000 0000 1110]
  Current = 7 (Active Master)
```

Tip: Repeat this command at one-second intervals to determine the number of blocks transferred each second. If the module is communicating over the backplane correctly, you will see these block numbers change each time you refresh the display.

HSBY Note: The Quantum 140CPU67160 processor control word %SW61 is displayed for HSBY processor status.

Viewing Module Configuration

Press **[C]** to view the *Module Configuration* screen.

Use this command to display the current configuration and statistics for the module.

Opening the Input Data View Menu

Press **[I]** to open the *Input Data View* menu. Use this command to view the contents of the input database. Refer to Input Data View Menu for information on the commands on this menu.

Opening the Output Data View Menu

Press **[O]** to open the *Output Data View* menu. Use this command to view the contents of the input database. Refer to Output Data View Menu for information on the commands on this menu.

Viewing Version Information

Press **[V]** to view Version information for the module.

Use this command to view the current version of the software for the module, as well as other important values. You may be asked to provide this information when calling for technical support on the product.

Values at the bottom of the display are important in determining module operation. The Program Scan Counter value is incremented each time a module's program cycle is complete.

> **Tip:** Repeat this command at one-second intervals to determine the frequency of program execution.

Viewing Fieldbus Data

Press **[2]** to view Fieldbus data. Use this command to view information related to the status of each slave in the PROFIBUS network, and to verify that each slave is configured *(SLAVE CFG LIST)*, exchanging data with the Master *(TRANSFER LIST)* and in diagnostic mode *(SLAVE DIAG LIST).*

You can also check the operation state of the module, where:

- 00 = Offline
- 40 = Stop
- 80 = Clear
- C0 = Operate

Viewing Module Status

Press **[1]** to view module status information. This screen also contains useful information for mailbox troubleshooting:

- Scan count
- Mailbox counters
- Alarm counters
- Hot Standby status
- Number of acyclic read and write operations performed by the module

You can also view the number of mailbox messages in the input and output queues, and the number of alarms in the alarm queue.

> **HSBY Note:** The following status reports new HSBY information.

```
IHSBY State        = 3       Remote PB Mstr = 0883  Local PB Mstr = 0392
IHSBY UDP Message  = len=12   Msg in Hex: 0883 FC3C078E DAD2D7E9
```

HSBY State:

0 = Not Connected

1 = Passive

2 = Active

3 = Stand-alone. This is not a HSBY state. Rather, it provides the state of the Master when HSBY is disabled.

Remote PB Mstr:

Same as offset word 60

Low byte: HSBY remote (passive) Master status

High byte: HSBY passive number of slaves

High Byte		Low Byte							
16	8	7				0			
		COM	--	--	OD	DB	CE	SO	PA

Bit	Explanation
PA	This bit indicates the state of the local Master. 0 - Local (active) Master (Master is controlled by the Primary PLC) 1 - Remote (passive) Master (Master is controlled by the Stand-by PLC)
SO	This bit indicates if the local Master recognizes any of its assigned slaves as offline. 0 - At least one slave is offline 1 - All slaves OK
CE	This bit indicates if the local Master has recognized a critical error. 0 - No critical errors 1 - Critical error active This bit is set when problems with the ping sequence are encountered.
DB	This bit indicates if the local Master has detected a database mismatch. 0 - Database OK 1 - Database mismatch
OD	This bit indicates when the data in the Output data area of the DPRAM is updated after a switchover. 0 - Output data is not updated 1 - Output data is updated (Once this bit is set, it remains set for the remaining session until the Anybus is either reset or HSBY state changes to "Not Connected")
-	Not used; set to zero
-	Not used; set to zero
COM	This bit indicates if the counterpart is present. 0 = Counterpart not present 1 = Counterpart is present

Local PB Mstr:

Same as offset word 61

Low byte: HSBY local (active) Master status

High byte: HSBY active number of slaves

High Byte		**Low Byte**							
16	8	7				0			
		HS	-	-	OD	DB	CE	SO	PA

Bit	**Explanation**
PA	Bits 0 to 6 refer to Remote Byte reference above.
SO	
CE	
DB	
OD	
-	
-	
HS	This bit indicates whether or not the Hot Standby functionality is enabled. 0 - HSBY disabled. Module operates as stand-alone Master or HSBY-state equals "Not connected". 1 - HSBY enabled

HSBY UDP Message:

Same as offset word 62. This is the UDP message length.

Msg in HEX:

Same as offset word 63 to 67. UDP message containing passive low byte, passive high byte, passive CRC32 checksum for PROFIBUS Master configuration, and CRC32 checksum for the module configuration.

Viewing Control Registers

Press **[3]** to view information about the PROFIBUS Master's Control Registers. Use this command to view general information about the module, such as the firmware version and its serial number. The module status contains two possible codes:

- 0400 = module is running but not communicating with slaves
- 0401 = module is running and communicating with slaves

If the module is in STOP mode, the status code will show as 0400.

Viewing Ethernet NIC Configuration

Press **[4]** to view the configuration for the Ethernet Network Interface Card (NIC) in the module.

```
Ethernet NIC Configuration
Link Yes
AutoNeg On
Speed 10M
Half-Duplex
```

```
Ethernet NIC Configuration
Link Yes
AutoNeg On
Speed 100M
Full-Duplex
```

Viewing the WATTCP.CFG File

Press **[@]** from the *Network* menu. Use this command to view the module's IP address settings.

Exiting the Program

Press **[ESC]** to restart the module and force all drivers to be loaded. The module will use the configuration stored in the module's flash memory to configure the module.

8.6.4 Input Data View Menu

Use this menu command to view the current contents of the selected database. Press **[?]** to view a list of commands available on this menu.

```
DATABASE VIEW MENU
 ?=Display Menu
 S=Show Again
 P=Previous Page
 N=Next Page
 D=Decimal Display
 H=Hexadecimal Display
 F=Float Display
 A=ASCII Display
 M=Main Menu
```

Viewing the Previous Page of Data

Press **[P]** to display the previous page of data.

Viewing the Next Page of Data

Press **[N]** to display the next page of data.

Viewing Data in Decimal Format

Press **[D]** from the *Database View* menu to display the data on the current page in decimal format.

Viewing Data in Hexadecimal Format

Press **[H]** from the *Database View* menu to display the data on the current page in hexadecimal format.

Viewing Data in Floating-Point Format

Press **[F]** from the *Database View* menu to display the data on the current page in floating-point format. The program assumes that the values are aligned on even register boundaries. If floating-point values are not aligned as such, they are not displayed properly.

Viewing Data in ASCII (Text) Format

Press **[A]** from the *Database View* menu to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Returning to the Main Menu

Press **[M]** to return to the *Main* menu.

8.6.5 Output Data View Menu

Use this menu command to view the current contents of the selected database. Press **[?]** to view a list of commands available on this menu.

```
OUTPUT DATA VIEW MENU
 ?=Display Menu
 S=Show Again
 P=Previous Page
 N=Next Page
 D=Decimal Display
 H=Hexadecimal Display
 F=Float Display
 A=ASCII Display
 M=Main Menu
```

Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Viewing the Previous Page of Data

Press **[P]** to display the previous page of data.

Viewing the Next Page of Data

Press **[N]** to display the next page of data.

Viewing Data in Decimal Format

Press **[D]** from the *Database View* menu to display the data on the current page in decimal format.

Viewing Data in Hexadecimal Format

Press **[H]** from the *Database View* menu to display the data on the current page in hexadecimal format.

Viewing Data in Floating-Point Format

Press **[F]** from the *Database View* menu to display the data on the current page in floating-point format. The program assumes that the values are aligned on even register boundaries. If floating-point values are not aligned as such, they are not displayed properly.

Viewing Data in ASCII (Text) Format

Press **[A]** from the *Database View* menu to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Returning to the Main Menu

Press **[M]** to return to the *Main* menu.

8.7 Standard PROFIBUS Slave Diagnostic Bytes

The diagnostic information consists of 6 bytes of standard diagnostic information plus any user-related diagnostic information. The standard information is shown in the tables below.

Byte	Description
0	Station status 1
1	Station status 2
2	Station status 3
3	Master address
4	Ident number high
5	Ident number low

8.7.1 Byte 0 - Station Status 1 Bits

Bit	Description
0	Station not existent
1	Station not ready
2	Configuration fault
3	Extended diagnostic data present
4	Not supported
5	Invalid slave response
6	Parameter fault
7	Master lock

8.7.2 Byte 1 - Station Status 2 Bits

Bit	Description
0	Parameter request
1	Static diagnostic
2	Slave device
3	Watchdog on
4	Freeze mode
5	Sync mode
6	Reserved
7	Slave deactivated

8.7.3 Byte 2 - Station Status 3 Bits

Bit	Description
0	Reserved
1	Reserved
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Extended diagnostic overflow

8.7.4 Byte 3 - Master Address

This byte shows the address of the assigned PROFIBUS Master after parameterization. If there is an error during the parameterization process, this byte will display the value FF (hexadecimal).

8.7.5 Byte 4 - Ident Number High

This byte shows the high byte of the specific Ident Number assigned to the module by the PROFIBUS User Organization.

8.7.6 Byte 5 - Ident Number Low

This byte shows the low byte of the specific Ident Number assigned to the module by the PROFIBUS User Organization.

9 Reference

In This Chapter

9.1 Product Specifications

The PTQ-PDPMV1 module is a powerful communication interface for Quantum platform processors. Developed under license from Schneider Electric, the module incorporates proprietary backplane technology that enables powerful data exchange with Quantum processors.

The PTQ-PDPMV1 PROFIBUS DP/DPV1 Master module supports complete Master specifications according to IEC 61158. Acyclic parameter data can be transferred with Class 1 or Class 2 DPV1 services, allowing processors to easily communicate with slave devices supporting the PROFIBUS DPV0/V1 protocol.

The module now includes new features and functionalities supporting the Quantum 140CPU67160 Hot Standby processor in the Unity Pro programming environment.

The new (HSBY) Hot Standby features will be included with the current product offering, part number PTQ-PDPMV1.

Most PTQ-PDPMV1 modules' firmware installed in the field can be flash upgraded, following the guidelines and restrictions below.

The module's operation will be identical to that of the stand-alone version with the following exceptions:

1. **Module Setup:** Once the module's firmware is updated, the module will automatically recognize the 140CPU67160 processor and activate the HSBY functions.
2. **PCB:** A new HSBY icon is displayed within PCB (ProSoft Configuration Builder) to identify the module as a HSBY unit.

Note: For specific HSBY instructions, pay attention to and follow the new HSBY icon for special instructions and guidance throughout this manual.

3. **Important Note on Field Firmware Flash Upgrades:** All modules having a serial number ≥ (greater than or equal to) 1451, having been shipped after 10/20/2005 can be field firmware upgraded. These modules have received a new hardware version 1.3 supporting the Hot Standby features. All other modules **must be** returned to ProSoft Technology for firmware upgrades.

Caution: Do not attempt to upgrade the firmware on modules with serial numbers 1450 or lower, otherwise the module may become inoperable. If the firmware upgrade fails, contact ProSoft technical support for assistance.

4 **Existing PROFIBUS networks:** PROFIBUS network baud rates greater than or equal to 500 kBaud are recommended to obtain a < 300 ms switchover time, based on an average processor scan time of 100 ms. For example, the switchover time for a network running 8 slaves utilizing 700 words input cyclic data and 700 words output cyclic data running at 500 kBaud with a processor scan time of 100 ms is around 218 ms. Please note Hot Standby units will increase the network token time as much as double because of two Masters communicating on the network.

5 **Module configuration network:** The module requires Ethernet connectivity to operate properly. The modules use UDP messaging between each other to back up data in status registers used by the processor logic files to determine switchover conditions in the event PROFIBUS FDL ping messages fail (cut-cable).

9.1.1 Hot Standby Support

The module provides support for 140CPU6716000 Hot Standby processor with Unity Pro programming software. Look for the HSBY (Hot Standby) icon for special notes relating to the support and configuration of the module.

HSBY Note: For detailed understanding of HSBY specification, refer to the Hot Standby Addendum.

9.1.2 General Specifications

- Single slot - Quantum backplane compatible
- The module is recognized as an Options module and has access to PLC memory for data transfer
- Configuration data is stored in non-volatile memory in the ProTalk® module
- Configuration software for Microsoft Windows XP, 2000 and NT is included with the module
- Up to six modules can be placed in a rack
- Local rack - The module must be placed in the same rack as the processor
- Compatible with all common Quantum programming packages, including Concept (version 2.6 or higher), Unity Pro (version 2.2 or higher), ProWORX (version 2.20 or later) (HSBY only available with Unity Pro environment)
- Quantum data types supported: 3x, 4x
- High speed data transfer across the backplane provides quick data update times
- Sample ladder file available

9.1.3 Hardware Specifications

Specification	Value
Backplane Current Load	1100 mA maximum @ 5 Vdc ± 5%
Operating Temperature	0°C to 60°C (32°F to 140°F)
Storage Temperature	-40°C to 85°C (-40°F to 185°F)
Relative Humidity	5% to 95% (with no condensation)
Vibration	Sine vibration 4-100 Hz in each of the 3 orthogonal axes
Shock	30 g, 11 mSec. in each of the 3 orthogonal axes
Dimensions (HxWxD), Approx.	250 x 103.85 x 40.34 mm 9.84 x 4.09 x 1.59 in
LED Indicators	Module Status Backplane Transfer Status Serial Port Activity LED Serial Activity and Error LED Status Master Status Operations Network Drop Communication Master Token-Hold Master database configuration
Debug/Configuration Ports	
Configuration Serial Port (PRT1)	DB-9M PC Compatible RS-232 only No hardware handshaking
Configuration Ethernet Port	RJ45 Connector Link and Activity LED indicators
Application Port	
PROFIBUS Master Port	DB-9F Optically Isolated RS-485 Ready, Run, Error and Token LED Indicators

9.1.4 Functional Specifications

- Easy-to-use drag and drop Busview configuration interface via ProSoft Configuration Builder software (see PSW-PCB Datasheet)
- Monitoring and modification of process data and DPV1 acyclic data with online slave diagnostics
- Supports PROFIBUS PA slaves on the network through DP/PA coupler or link
- Supports up to 125 slave devices with repeaters
- Supports extended diagnostic data (DPV1)
- Supports all standardized baud rates, up to 12 Mbits/s
- Auto baud detection at all valid PROFIBUS DPV1 rates
- Supports PROFIdrive 3.1 compliant parameter read and write operations
- Supports Sync and Freeze commands
- Alarm indications and confirmations handling (DPV1)
- Supports Multicast and Broadcast telegrams (DPV1)
- CRC checksum determination of slave configuration consistency to processor
- FDT/DTM PROFIBUS Master transport communication DTM software included (Product Number PSW-CDTM-PDPM)

Hot Standby

- Hot Standby features support the SE 140 671 CPU
- Supports up to six PTQ-PDPMV1 Hot Standby modules per rack
- Diagnostic and status words are provided for Active Primary and Passive Secondary Master health status
- PROFIBUS switchover time will be nominal 100 ms not to exceed 300 milliseconds
- Cable break detection with segmented network slave quantity information
- PROFIBUS health messages are generated from secondary Master via FDL ping services
- No setup parameters required. Module automatically detects Hot Standby system

Physical

- PROFIBUS DPV1 RS-485 interface with a 9-pin D shell female connector and isolated Opto-Couplers
- Master Status LED Indicators for Operations, Network Drop Communication, Master Token-Hold

9.2 Functional Overview

9.2.1 About the PROFIBUS Protocol

PROFIBUS (Process Field Bus) is a widely-used, open-standards protocol created by a consortium of European factory automation suppliers in 1989.

PROFIBUS is a Master/slave protocol. The Master establishes a connection to one or more remote slaves. When the connection is established, the Master sends the PROFIBUS poll messages (called telegrams in PROFIBUS) to the slave or slaves. The PTQ-PDPMV1 module works as a Master only. It cannot be a slave to some other Master.

The PTQ-PDPMV1 module also acts as an input/output module between devices on a PROFIBUS network and the Schneider Electric Quantum processor. The module uses an internal database to pass data and mailbox requests and responses between the processor and the slave devices on the PROFIBUS network.

PROFIBUS specifications include a variety of network types. The network type supported by the PTQ-PDPMV1 module is PROFIBUS DP version 1.0, which is designed for remote I/O systems, motor control centers, and variable speed drives.

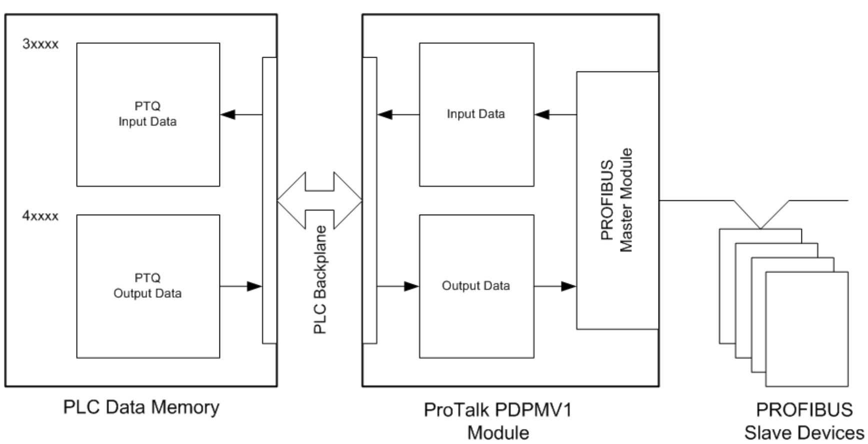
9.2.2 General Overview

The PTQ module communicates with the processor over the backplane using only the following two blocks of data:

- PTQ Input Data block
- PTQ Output Data block

This section of the manual describes the data structures and transfer mechanisms used to transfer data between the PTQ-PDPMV1 module and the Quantum processor.

The following illustration shows the Input/Output Data block flow between the Quantum processor and the PTQ-PDPMV1 module.



These two data blocks (Input Data and Output Data) consist of a data structure that provides for the movement of:

- Input Data image from PROFIBUS slave devices
- Output Data image for writing to PROFIBUS slave devices
- PTQ Module Configuration and Status (from PTQ to Quantum)
- PROFIBUS Messaging Mailbox commands (from Quantum to PTQ)
- PROFIBUS Messaging Mailbox responses (from PTQ to Quantum)

9.2.3 PROFIBUS DP Architecture

The network supports multiple Master systems with several slaves.

The following table shows the most important features of:

Standard	**EIN 501 70** **DIN 19245**
Transmission Equipment (Physical)	EIA RS-485 IEC 1158-2 (through link or coupler) Fiber Optic Cable (not available)
Transfer Procedure	Half-duplex
Bus Topology	Linear bus with active bus termination
Bus Cable Type	Shielded twisted pair conductors
Connector	9-pin D-Sub
Number of nodes on the bus	Max: 32 with no repeaters Max: 125 with 3 repeaters in 4 segments

Effective Range

Max Bus Cable Length Per Segment	**Baud Rates (for 12 Mbit/sec cable)**
1.2 km	9.6 kbps
1.2 km	19.2 kbps
1.2 km	93.75 kbps
1.0 km	187.5 kbps
0.5 km	500 kbps
0.2 km	1.5 Mbps
0.1 km	3 Mbps
0.1 km	8 Mbps
0.1 km	12 Mbps

Bus Access

Two different bus access procedures handle the various communication requirements for the topology:

- Token Passing
- Polling

Token Passing

Token passing ring is the basis for communication between the more complex, active stations. All stations have the same rights in that a token is passed from station to station in a logical ring. The token is passed to each station with a maximum, definable token cycle time. A station is given transmission rights for the duration of time that it has the token.

Master/Slave Polling

Master/slave polling guarantees a cyclic, real-time based data exchange between the station with transmission rights, the active station, and its subordinates, the passive stations. In this case, the Master is able to pass data to the slave and/or receive data. The services in layer 2 (field-bus data link in ISO-OSI reference model) organize this communication.

9.2.4 Master/Slave Communication Phases

The communication between the Master and the slaves is split up into the following phases:

- Parameterization and configuration phase
- Usable data transfer phase

Before a DP slave can be integrated into the usable data transfer phase, the parameterization and configuration phase runs a device identification test that verifies that the planned configuration matches the actual device configuration for each slave in the PROFIBUS network. The test verifies that:

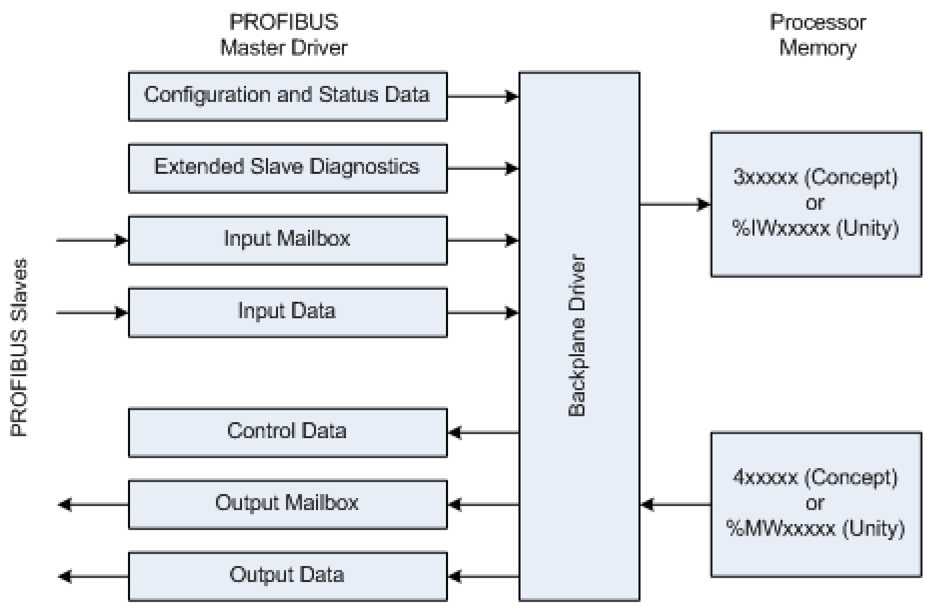
- the device is actually there
- it is the right type of device
- the address, which is set on the device, matches the station address on the bus
- the formats, telegram length information, and bus parameters are correct
- the number of configured inputs and outputs is correct

9.2.5 PTQ Input and Output Data Blocks

The PTQ-PDPMV1 Input Data block contains PROFIBUS input data received from slaves on the PROFIBUS network, as well as module and slave status data. It may also include extended slave diagnostics and acyclic message (mailbox) response data, if these are enabled. The module writes this Input Data block to Input Register addresses in the Quantum processor's state RAM (3xxxxx for Concept or %IWxxxxxx for Unity).

The PTQ-PDPMV1 module reads an Output Data block from Holding Register addresses in processor state RAM (4xxxxx for Concept or %MWxxxxxx for Unity). The Output Data block contains PROFIBUS output data to be sent to slaves on the PROFIBUS network, as well as control data. It may also include outgoing acyclic messages (mailbox commands), if Mailbox Messaging is enabled.

The sizes and starting register addresses for the Input and Output Data blocks are determined by the configuration specified in ProSoft Configuration Builder during module setup.



Normal Operation

The PTQ-PDPMV1 module's application code initiates the data transfers at the end of every Quantum PLC ladder scan. As such, the PTQ-PDPMV1 module is able to actively read and write the PROFIBUS Cyclic Input/Output data blocks in the appropriate locations.

Input and Output Data Block Format

With Mailbox Messaging and Without Slave Diagnostics

- Mailbox Messaging = Y
- Slave Diagnostics = N

Input Data Block from Module to Processor (991 Words Maximum)

Word Offset	Description
0 to 78	Configuration and status data
79 to 222	Incoming Mailbox Message data: 144-word Incoming Message block
223 to *n*	PROFIBUS Input Data: Data received from the PROFIBUS slave devices on the network Total size of the PROFIBUS Input Data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). *n* is a function of the user-selected size of the PROFIBUS Input Data block.

Output Data Block from Processor to Module (918 Words Maximum)

Word Offset	Description
0	Last In Mailbox Message ID
1	Last Alarm Control Index
2 to 3	PROFIBUS CRC32: computed for PROFIBUS configuration
4 to 5	Module CRC32: computed for module data When the module first starts up or recognizes an initialization of the processor, it will compare the values of the two CRCs in the input and output images. If either one of the CRCs does not match, the module will be placed in STOP mode. If each set matches, the module will be placed in OPERATE mode.
6 to 149	Outgoing Mailbox data: Mailbox Message command being sent to the PTQ module
150 to *n*	PROFIBUS Output Data: Data going to the PROFIBUS network Total size of the PROFIBUS Output Data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). *n* is a function of the user-selected size of the PROFIBUS Output Data block.

Without Mailbox Messaging and With Slave Diagnostics

- Mailbox Messaging = N
- Slave Diagnostics = Y

Input Data Block from Module to Processor (1219 Words Maximum)

Word Offset	Description
0 to 72	Configuration and status data
73 to 450	Incoming slave 6-byte diagnostics data for 126 slaves (378 words of data for slaves 0 to 125). Refer to Standard PROFIBUS Slave Diagnostic Bytes (page 241).
451 to *n*	PROFIBUS Input Data: Data received from the PROFIBUS slave devices on the network Total size of the PROFIBUS Input Data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). *n* is a function of the user-selected size of the PROFIBUS Input Data block.

Output Data Block from Processor to Module (774 Words Maximum)

Word Offset	Description
0	Set Operating Mode (New) Bit 15= Handshake (if equal to Input Word 72 Bit 15, then module has control of word and if not, then module has processed request) Bits 8-14= Reserved for future use Bits 0-7 contain the operation code: 0x40= Stop 0x80= Clear 0xC0= Operate
1	Reserved
2 to 3	PROFIBUS CRC32: computed for PROFIBUS configuration
4 to 5	Module CRC32: computed for module data When the module first starts up or recognizes an initialization of the processor, it will compare the values of the two CRCs in the input and output images. If either one of the CRCs does not match, the module will be placed in STOP mode. If each set matches, the module will be placed in OPERATE mode.
6 to *n*	PROFIBUS Output Data: Data going to the PROFIBUS network Total size of the PROFIBUS Output Data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). *n* is a function of the user-selected size of the PROFIBUS Output Data block.

Without Mailbox Messaging and Without Slave Diagnostics

- Mailbox Messaging = N
- Slave Diagnostics = N

Input Data Block from Module to Processor (841 Words Maximum)

Word Offset	Description
0 to 72	Configuration and status data
73 to *n*	PROFIBUS Input Data: Data received from the PROFIBUS slave devices on the network Total size of the PROFIBUS Input Data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). *n* is a function of the user-selected size of the PROFIBUS Input Data block.

Output Data Block from Processor to Module (774 Words Maximum)

Word Offset	Description
0	Set Operating Mode (New) Bit 15= Handshake (if equal to Input Word 72 Bit 15, then module has control of word and if not, then module has processed request) Bits 8-14= Reserved for future use Bits 0-7 contain the operation code: 0x40= Stop 0x80= Clear 0xC0= Operate
1	Reserved
2 to 3	PROFIBUS CRC32: computed for PROFIBUS configuration
4 to 5	Module CRC32: computed for module data When the module first starts up or recognizes an initialization of the processor, it will compare the values of the two CRCs in the input and output images. If either one of the CRCs does not match, the module will be placed in STOP mode. If each set matches, the module will be placed in OPERATE mode.
6 to *n*	PROFIBUS Output Data: Data going to the PROFIBUS network Total size of the PROFIBUS Output Data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). *n* is a function of the user-selected size of the PROFIBUS Output Data block.

With Mailbox Messaging and With Slave Diagnostics

- Mailbox Messaging = Y
- Slave Diagnostics = Y

Input Data Block from Module to Processor (1369 Words Maximum)

Word Offset	Description
0 to 78	Configuration and status data
79 to 222	Incoming Mailbox Message data: 144 word Incoming Message block
223 to 600	Incoming slave 6 byte diagnostics data for 126 slaves (378 words of data for slaves 0 to 125). Refer to Standard PROFIBUS Slave Diagnostic Bytes (page 241).
601 to *n*	PROFIBUS Input Data: Data received from the PROFIBUS slave devices on the network Total size of the PROFIBUS Input Data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). *n* is a function of the user-selected size of the PROFIBUS Input Data block.

Output Data Block from Processor to Module (918 Words Maximum)

Word Offset	Description
0	Last In Mailbox Message ID
1	Last Alarm Control Index
2 to 3	PROFIBUS CRC32: computed for PROFIBUS configuration
4 to 5	Module CRC32: computed for module data When the module first starts up or recognizes an initialization of the processor, it will compare the values of the two CRCs in the input and output images. If either one of the CRCs does not match, the module will be placed in STOP mode. If each set matches, the module will be placed in OPERATE mode.
6 to 149	Outgoing Mailbox Data: Mailbox Message command being sent to the PTQ module
150 to *n*	PROFIBUS Output Data: Data going to the PROFIBUS network Total size of the PROFIBUS Output Data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). *n* is a function of the user-selected size of the PROFIBUS Output Data block.

Status Data in Input Data Block

The PTQ-PDPMV1 module's Input Data block contains several types of data in addition to the PROFIBUS network Input data. Much of this data is useful for determining the operational status and the configuration of the module.

The types of data returned in the PROFIBUS Input Status area include:

1 Module configuration data values
2 PTQ software revision level
3 Key PROFIBUS configuration values
4 PROFIBUS Master hardware/software revision level
5 PTQ module statistics
6 Mailbox messaging control and status

The following data is received from the PTQ module during every PROFIBUS Input Data block update.

Note: If Mailbox Messaging is disabled, words 73 through 78 of this status data area are not used.

Word Offset	Name	Description
0 to 4	Module ID String	Unique module 10-byte pattern as text "PTQ-PDPMV1" for module verification
5	Quantum Slot Number	Slot number in the rack in which the PTQ-PDPMV1 module is located. Value is selected during user configuration
6	PROFIBUS Input Data Size	The number of words of PROFIBUS Input data to transfer from the PROFIBUS Master to the processor within the Input Data blocks. Value is selected during user configuration
7	PROFIBUS Output Data Size	The number of words to transfer from the processor in the PROFIBUS Output space during the Output data transfer. Value is selected during user configuration
8	Input Data Start Address	Starting 4xxxx Holding Register address for Input Data block in processor data memory. Value is selected during user configuration
9	Output Data Start Address	Starting 4xxxx Holding Register address for Output Data block in processor data memory. Value is selected during user configuration
10	Reserved	Reserved for future use
11	Input/Output Data Byte Swap	**High byte:** User-configured flag to indicate if output data is swapped after being received from the output image of the controller. If it is 0, no swapping occurs. If it is a nonzero value, then bytes are swapped. **Low byte:** User-configured flag to indicate if input data is swapped before being placed in the input image for the controller. If it is 0, no swapping occurs. If it is a nonzero value, then bytes are swapped.
12	Module Software Major/Minor Version Number	High byte: Module software major version number Low byte: Module software minor version number
13 to 20	PROFIBUS Slave Configured List	This is a 16-byte array with bit fields where one bit is assigned to each slave station address. The associate bit is set if the slave is present in the database. For the bit/slave relationship, refer to the Slave List Structure below.
21 to 28	PROFIBUS Data Transfer Status	This is a 16-byte array with bit fields where one bit is assigned to each slave station address. The associated bit is set if the slave has reached or retained the "Data Exchange" state at least once during the last three data cycles. For the bit/slave relationship, refer to the Slave List Structure below.

Word Offset	Name	Description
29 to 36	PROFIBUS Slave Diagnostic Status	This is a 16-byte array with bit fields where one bit is assigned to each slave station address. When a slave leaves the "Deactive" state for the first time, the associated bit is set. This bit is then cleared when the slave enters "Data Exchange" state. If a slave indicates "Extended Diagnostics" when it is in "Data Exchange" state, the associated bit is set. For the bit/slave relationship, refer to the Slave List Structure below.

Slave List Structure for Offset Words 13, 21 and 29

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Slave 7	Slave 6	Slave 5	Slave 4	Slave 3	Slave 2	Slave 1	Slave 0
Byte 1	Slave 15	Slave 14	Slave 13	Slave 12	Slave 11	Slave 10	Slave 9	Slave 8
Byte 2	Slave 23	Slave 22	Slave 21	Slave 20	Slave 19	Slave 18	Slave 17	Slave 16
Byte 3	Slave 31	Slave 30	Slave 29	Slave 28	Slave 27	Slave 26	Slave 25	Slave 24
Byte 4	Slave 39	Slave 38	Slave 37	Slave 36	Slave 35	Slave 34	Slave 33	Slave 32
...	...	...	...	...	...	...	...	...
Byte 15	Slave 127	Slave 126	Slave 125	Slave 124	Slave 123	Slave 122	Slave 121	Slave 120

Word Offset	Name	Description
37	PROFIBUS Master Operating State	PROFIBUS Master operating state 0x0000=Offline 0x4000=Stop 0x8000=Clear 0xC000=Operate
38	PROFIBUS Ident Number	PROFIBUS Master PNO Ident number. Bytes will be swapped
39 to 40	PROFIBUS Master Serial Number	Unique 32-bit serial number for the PROFIBUS Master
41	PROFIBUS Software Version	This is the software version number for the PROFIBUS Master software. Example for Version 1.40: High Byte - 0x40 Low Byte - 0x01

Word Offset	Name	Description
42	PROFIBUS Master Module Status	Represents the PROFIBUS Master module's operating status Bit 2=Application status 0 - Application stopped 1 - Application running Bit 8=Data exchange (FBRS) 0 - There is no data exchange with any of the assigned slaves 1 - There is data exchange with at least one of the assigned slaves Bit 9=Slave input frozen/cleared (FBFC) 0 - A slave's inputs in the IN area are cleared if a slave is not in Data Exchange 1 - A slave's inputs in the IN area are frozen if a slave is not in Data Exchange Bit 12=Reset (RDR) 0 - No action 1 - A reset is requested by the PROFIBUS Master module because a new database has been downloaded
43 to 44	PROFIBUS Configuration Checksum	CRC32 checksum for PROFIBUS Master configuration downloaded from configuration utility
45 to 46	PTQ Module Configuration Checksum	PTQ-PDPMV1 module configuration checksum for module configuration downloaded from configuration utility
47	Application Program Scan Counter	PTQ-PDPMV1 module program scan counter. Can be used to gauge application code scan time performance
48	Module PROFIBUS Output Image Data Update Counter	Counter representing the number of times the output data image is transferred to the module's internal Master
49	Module PROFIBUS Input Image Data Update Counter	Counter representing the number of times the input data image is transferred from the module's internal Master
50	Module Out Mailbox Counter	Incremented at every mailbox requested from the module
51	Module In Mailbox Counter	Incremented at every mailbox response sent to the Quantum
52	Module Alarm IND Receive Counter	Number of spontaneous alarm messages received from slave
53	Module Alarm CON Receive Counter	Number of confirmation messages received from slaves indicating that the slave received the confirmation message from the PTQ-PDPMV1 module
54	Cyclic Input Data Start Offset	Cyclic input data start offset
55	Cyclic Output Data Start Offset	Cyclic output data start offset
56	Module Backplane Read Count	Rollover counter of the number of PTQ-to-processor backplane read data transfers
57	Module Backplane Write Count	Rollover counter of the number of PTQ-to-processor backplane write data transfers
58	Module Backplane Error Count	Rollover counter of the number of PTQ-to-processor backplane data transfers that have failed

Word Offset	Name	Description
59	File Error Word	Bitmapped value that indicates which files are not present Bit 0: Problem with PDPMV1.CFG file Bit 1: Problem with WATTCP.CFG file Bit 2: Problem with PDPMV1.DDB file Bit 3: Problem with PDPMV1.ZIP file Bits 4 to 15 not used If this word has a value other than 0, the CFG ERR LED on the module will be illuminated.
60	HSBY Remote Status (Unity only)	**Low byte:** HSBY remote status - from PROFIBUS interface (0xEA0) Bit 0=PA 0 - Active Master (controlled by the Primary PLC) 1 - Passive Master (controlled by the Standby PLC) Bit 1=SO 0 - At least one slave is offline 1 - All slaves OK Bit 2=CE (This bit is set when problems with the ping sequence are encountered.) 0 - No critical errors recognized by local Master 1 - Active critical error recognized by Local Master Bit 3=DB 0 - Database OK 1 - Database mismatch Bit 4=OD (Indicates when the data in the Output Data area of the DPRAM is updated after a switchover.) 0 - Output data is not updated 1 - Output data is updated (Once this bit is set, it remains set for the remaining session until the Anybus is either reset of HSBY state changes to "Not Connected") Bits 5 and 6 not used; set to zero Bit 7=COM 0 - Counterpart is not present 1 - Counterpart is present **High byte:** HSBY remote number of slaves - from PROFIBUS interface (0xEA1)
61	HSBY Local Status (Unity only)	**Low byte:** HSBY local status - from PROFIBUS interface (0xEA4) Bits 1 through 6: See above under HSBY Remote Status Bit 7=HS 0 - HSBY disabled. Module operates as stand-alone Master or HSBY-state equals "Not connected." 1 - HSBY enabled. **High byte:** HSBY local number of slaves - from PROFIBUS interface (0xEA5)
62	HSBY Message Length (Unity only)	From UDP HSBY server
63	HSBY Passive Status (Unity only)	**Low byte:** HSBY passive status - from UDP HSBY server **High byte:** HSBY passive number of slaves - from UDP HSBY server This is a backup word derived from Ethernet UDP messaging. Refer to Word 60 for explanation.

Word Offset	Name	Description
64 to 65	HSBY Passive PROFIBUS CRC32 (Unity only)	CRC32 checksum for PROFIBUS Master configuration downloaded from configuration utility via UDP
66 to 67	HSBY Passive User Cfg CRC32 (Unity only)	PTQ-PDPMV1 module configuration checksum for module configuration downloaded from configuration utility via UDP
68 to 71	Reserved	Reserved for future use
72	Control Data	This word is used when Mailbox Messaging is disabled. Operating State Mode Return (New) Bit 15= Handshake bit Bit 14= Error bit (1=Error, 0=No error) Bits 12-13= Reserved for future use Bits 8-11= Error code if bit 14 set: (1=Queue full, 2=Memory allocation error, 3=Invalid operating mode command) Bits 0-7 contain the operation code requested: 0x40= Stop 0x80= Clear 0xC0= Operate
73	In Mailbox Queue Count	Number of message in the In Mailbox queue
74	Out Mailbox Queue Count	Number of message in the Out Mailbox queue
75	Alarm Queue Count	Number of message in the Alarm queue
76	Last Out Mailbox Message ID Processed from Output Image	The module confirms the receipt of a mailbox by copying its ID code (Message ID) to this register
77	Current In Mailbox Control Index	Incremented after the module has transferred a new mailbox response to the processor
78	Current Alarm Control Index	Incremented after the module has transferred a new alarm to the processor

9.3 PROFIBUS comDTM

DTM (Device Type Manager) is a standard way to provide all necessary data and functionality for a communication device, for example a PROFIBUS DP card. This technology is similar to the way Microsoft Windows supports printer drivers supplied by manufacturers and available to any Windows application, rather than requiring a custom printer driver for each specific application.

PROFIBUS comDTM, distributed by ProSoft Technology, is a DTM for PTQ and MVI series PDPMV1 modules and ProLinx PDPMV1 gateways. Configuration is available through Ethernet for the PTQ and ProLinx series PDPMV1, and through RS-232 serial for the MVI series PDPMV1 modules. Ethernet configuration is also available through the CIPConnect® feature for MVI56-PDPMV1 and MVI69-PDPMV1 modules (see note).

PROFIBUS comDTM allows configuration tools, instruments, and communication devices on a field network to recognize and use the module's capabilities.

Communication Channels Supported

	Serial	Ethernet	
Product	**Local RS-232 Port**	**Local Ethernet Port**	**CIPconnect**
MVI46-PDPMV1	Supported		
MVI56-PDPMV1	Supported		Supported (see note)
MVI69-PDPMV1	Supported		Supported (see note)
PTQ-PDPMV1		Supported	
ProLinx PDPMV1		Supported	

Note: MVI56-PDPMV1 requires firmware version 1.28.000 or later to support CIPconnect-enabled PC programs (1756-ENBT support). The 1756-ENBT card must be located in the same rack as the MVI56-PDPMV1. Bridging through multiple racks is not supported by the comDTM software.
MVI69-PDPMV1 requires firmware version 1.37.002 or later to support CIPconnect through the Ethernet port to L32E and L35E CompactLogix processors.
Note: This functionality requires comDTM version 1.0.1.5 with install version 1.01.0003. For information on how to check the comDTM version and install version, refer to Verifying the comDTM Version and comDTM Install Version (page 273).

9.3.1 ProSoft Technology Product Availability

Part Number	**Description**
PSW-cDTM-PDPM	PROFIBUS DPV1 Master comDTM software gateway

9.3.2 Introduction to PROFIBUS comDTM

Why Use PROFIBUS comDTM?

Customers from around the world have different machines, fieldbusses, and other end-user equipment. Each is equipped with the field bus requested by their end-user. Since there are so many variations, the automation solution in their standard machine ends up being different from case to case.

This means that service engineers need to have different configuration tools for every fieldbus. Or maybe even one for every device. You want one, but the reality is you have many. This is where PROFIBUS comDTM can help with configuring and communicating with different networks, products and suppliers.

What is FDT?

FDT (Field Device Tool) is the specification for software interfaces for the integration of field devices, drives, and controls in engineering and configuration tools. FDT is manufacturer-independent and allows for trouble-free parameterization and configuration of the user's specific processing system.

FDT technology standardizes the communication interface between field devices and systems. The key feature is its independence from the communication protocol and the software environment of either the device or the host system. FDT allows any device to be accessed from any host through any protocol.

The FDT container implements the FDT specification. It serves as an interface between FDT and a superior application. It uses the DTMs to gain access to the devices. FDT frame application is an engineering or configuration tool which has an FDT container.

FDT technology comprises three key components: the Frame Application, Device DTMs, and Communication DTMs.

- The DTM (Device Type Manager) is used for the configuration and maintenance of field devices, drives and so on. It is only functional with an FDT container.
- The FDT container implements the FDT specification. It serves as interface between FDT and a superior application. It uses the DTMs to gain access to devices.
- FDT frame application is an engineering or configuration tool that has an FDT container. The user interface of the DTMs is displayed here.

To better understand the functionality of these components, consider the analogy of the Internet - a standard web browser allows users to view countless web pages created by many content providers. The host system supplier typically creates the Frame Application, just as Microsoft supplies the Internet Explorer web browser. Just as a web browser opens a web page that contains code from the company that makes the web page, an FDT frame opens the Device DTM, which is the plug-in software from the device vendor.

Similar to a web browser, the Frame Application has menu bars, toolbars, and a navigation tree. Visually, the frame application surrounds the device vendor's DTM. Like opening a web page from a 'favorites' navigation tree, a user can navigate down a tree that lists the field device tags, click on one, and open the device vendor's DTM inside the frame. And, like web pages that let users interact with a reservation system or a shopping service, the Device DTMs let the user interact with the field device in a wide variety of ways. The Device DTM vendor can create a graphically rich user interface that does virtually anything possible in an advanced Windows PC-type interface. The third part of the technology, the Communication DTM, provides a standardized communication Application Process Interface (API) inside the PC, interfacing between the Device Vendor's DTM and the host system's specific driver that handles pass-through communications from the PC down to the fieldbus interface card.

The host system vendor supplies a Communication DTM (comDTM) for each supported fieldbus protocol. This ensures that the details of the PC, network, interface cards, and pass-through protocols of the host system, are transparent to the device vendor's DTM. This correlates back to the internet analogy where: the web page is transparent to the PC it's running in, the brand of the network interface card in the PC, or whether communication is DSL or broadband cable.

FDT technology complements and expands existing device description languages. It does not replace but rather builds upon existing DDs.

In particular, FDT expands the capabilities of DD for complex devices. Device Description languages have limitations in the graphical representation of the device at the user interface and allow only a limited integration of special features. FDT/DTM removes these limitations.

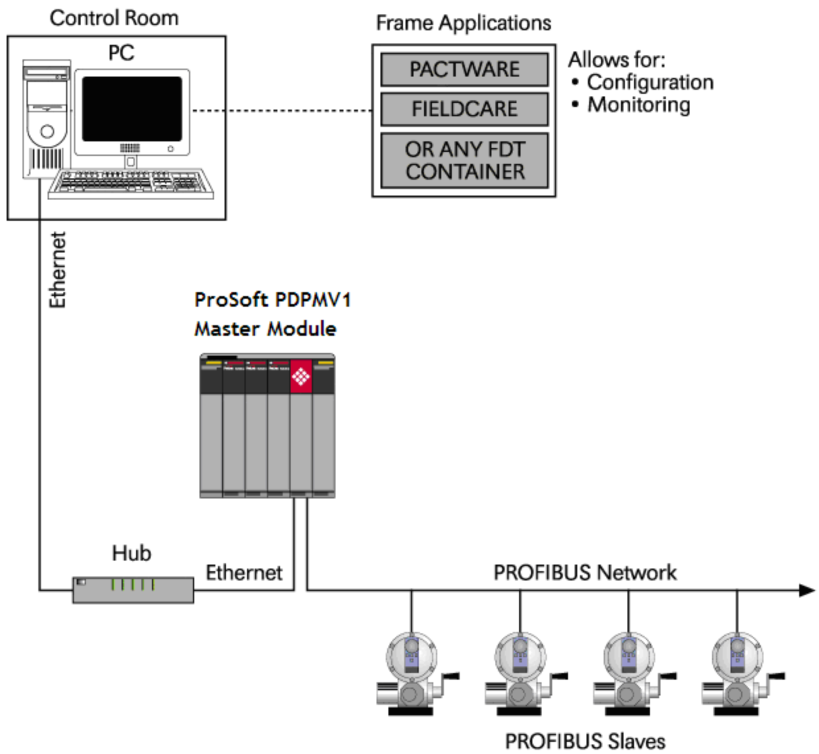
Typical frame applications are

- Pactware from The PACTware Consortium e.V (freeware)
- FieldCare from Endress & Hauser
- Field Control from ABB

What is DTM?

DTM (Device Type Manager) is a standard way to provide all necessary data and functionality for a communication device, for example a PROFIBUS DP card. This technology is similar to the way Microsoft Windows supports printer drivers supplied by manufacturers and available to any Windows application, rather than requiring a custom printer driver for each specific application.

PROFIBUS comDTM, distributed by ProSoft Technology, is a DTM for PTQ and MVI series PDPMV1 modules and ProLinx PDPMV1 gateways. It allows configuration tools, instruments and communication devices on a field network to recognize and use the module's capabilities.



9.3.3 System Requirements

Confirm that your system meets the following hardware and software requirements before you start the installation.

Hardware Requirements (Recommended)

- Pentium 4 processor rated for at least 2 GHz
- 450 MB hard drive space for DTM Libraries
- Video card capable of 1024 X 768 resolution at 64k colors
- Ethernet Network Interface Card (NIC)
- One of the following ProSoft Technology PROFIBUS DPV1 Master modules:
  - ProLinx PDPMV1 Ethernet only, serial port not supported
  - PTQ-PDPMV1 Ethernet only, serial port not supported
  - MVI series PDPMV1 RS232 serial
  - MVI56-PDPMV1 (with 1756-ENBT for Ethernet support)

Note for PTQ Users: The Ethernet connection implements UDP protocol, which dynamically allocates a random UDP port for every connection. This implementation limits the possibility of using most serial to Ethernet converters to access the PDPMV1 serial port through an Ethernet connection. Several Ethernet to serial converters require the configuration of a fixed UDP port, which is not available for the current implementation.

Software Requirements (Minimum)

- Windows NT 4.0 Service Pack 6A, Windows 2000 SP3 or Windows XP Professional SP2, or better
- Microsoft Internet Explorer Version 6.0, or better
- FDT 1.2.1 compliant FDT frame application. Compatible applications include:
  - PACTware
  - FieldCare
  - M&M fdtCONTAINER

Some FDT Containers require the following components:

- Microsoft Management Console
- Adobe Acrobat Reader 5.0, or better

9.3.4 Installation

Important: You must have Administrator rights on your computer to install this application.
Important: Please open and read the *Readme* file before starting the installation. The *Readme* file is located in Utilities > comDTM > Readme on the *ProSoft Solutions Product CD-ROM*.

To install comDTM

1 Insert the *ProSoft Solutions Product CD-ROM* in an available CD-ROM drive in your computer. Wait for the startup screen to appear.
2 On the startup screen, click **PRODUCT DOCUMENTATION**. This action opens a *Windows Explorer* file tree window.
3 Double-click to open the **UTILITIES** folder, then navigate to **COMDTM** > **COMDTM INSTALL**.
4 Double-click the **SETUP.EXE** file. This action starts the installation wizard.
5 Follow the instructions on the installation wizard to install the program.
6 Click **FINISH** to complete the installation. If you are prompted to restart your computer, save your work in any applications that are running, close the applications, and allow the computer to restart.

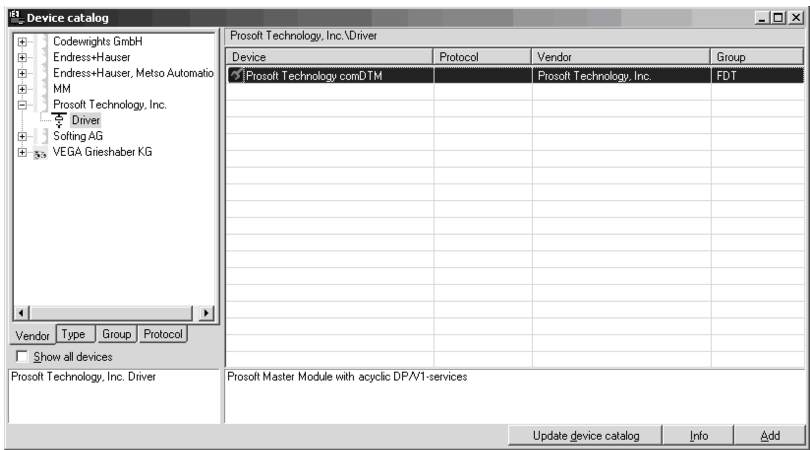
Note: During installation, you will be prompted to accept or change the location for the database folder. The default location for this folder is the Program Files directory on your local hard drive (normally Drive C:). If you intend to allow multiple workstations to access the same database folder, you should choose a network drive that other workstations can access.

9.3.5 Quick Start

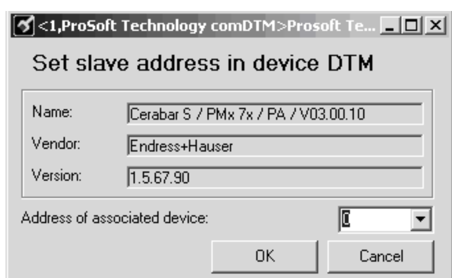
The following steps demonstrate how to start the FDT (Field Device Tool) program and configure the PROFIBUS comDTM.

Starting FDT

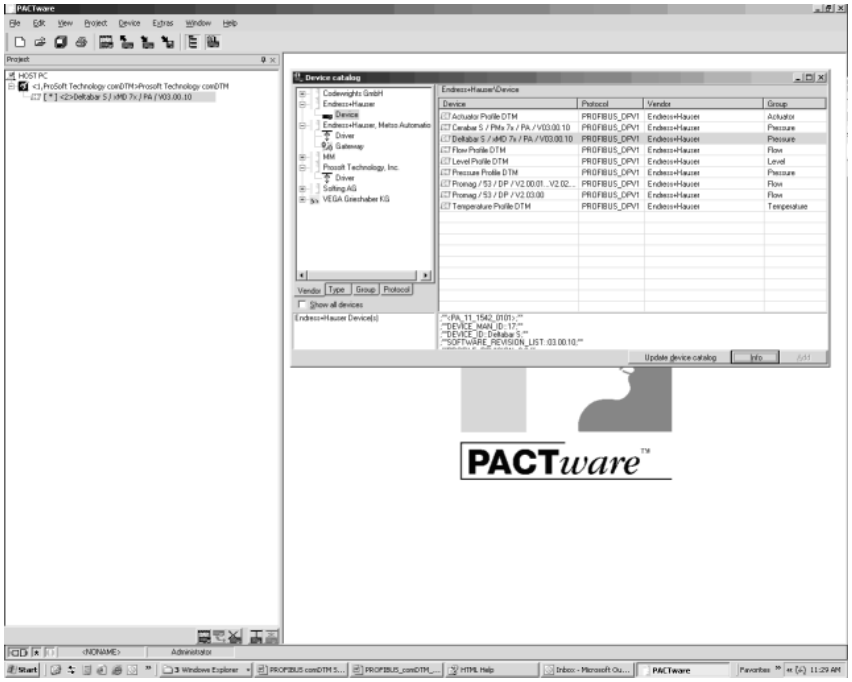
1 Start the FDT program and login as administrator. The following procedures use PACTware 3.0 software.
2 Click the **UPDATE DEVICE CATALOG** button. If PROFIBUS comDTM was installed successfully, it will appear in the *Device Catalog* window.



3 Select the **PROSOFT TECHNOLOGY COMDTM** entry in the device catalog, and then click **ADD**.
4 Repeat steps 1 through 3 to add any other manufacturer's device DTMs installed on your computer. Select the correct address for each device, and then click **OK** to proceed.



Note: You must use the same PROFIBUS device address that you used when setting up the device.

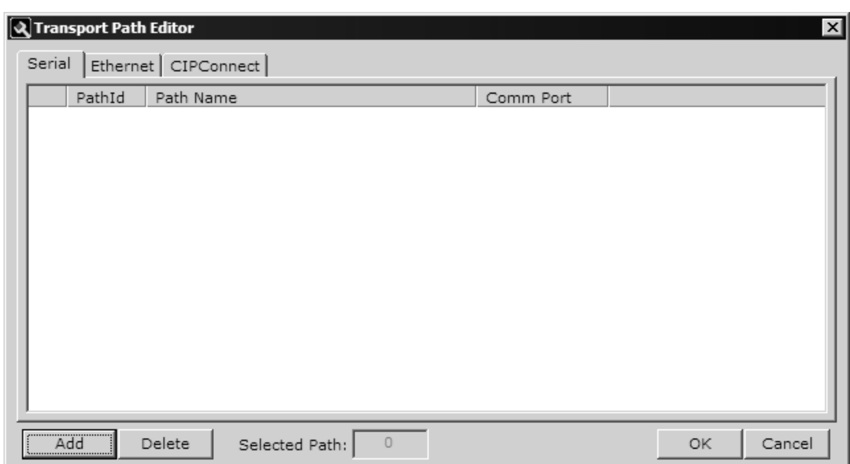


<u>Connecting the comDTM to the Master to Establish Communication</u>

> **Note:** The features described in this section require the current version of PROFIBUS comDTM. You can always download the newest version from www.prosoft-technology.com.

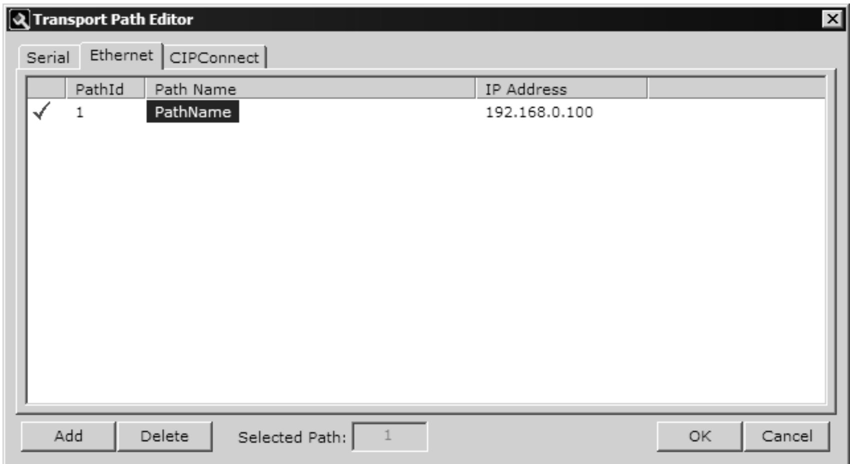
1 From the Windows **START** button, navigate to **PROGRAMS** > **PROSOFT TECHNOLOGY**, and then choose **PROSOFT TRANSPORT PATH EDITOR**.

2 If you have not created a communication path, click the **ADD** button. If you have already created a path, skip to step 5.

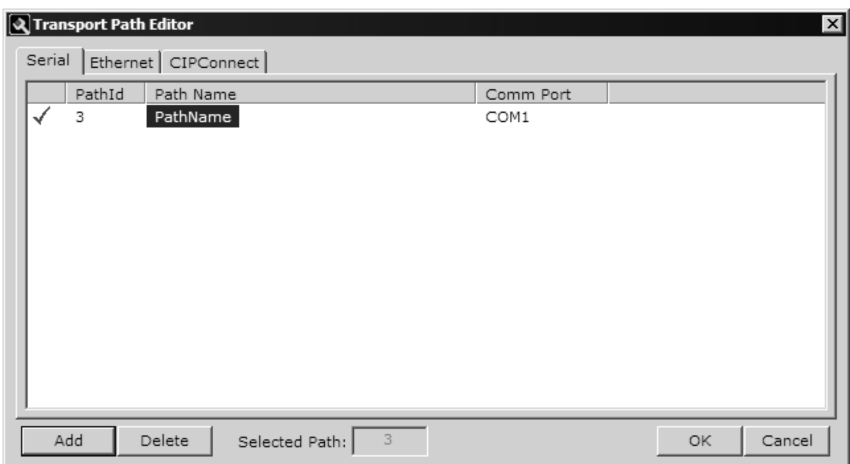


3 For PTQ-PDPMV1 and ProLinx-PDPMV1 users: Select the **ETHERNET** tab, and enter the IP Address of the module or gateway. Enter a descriptive path name.

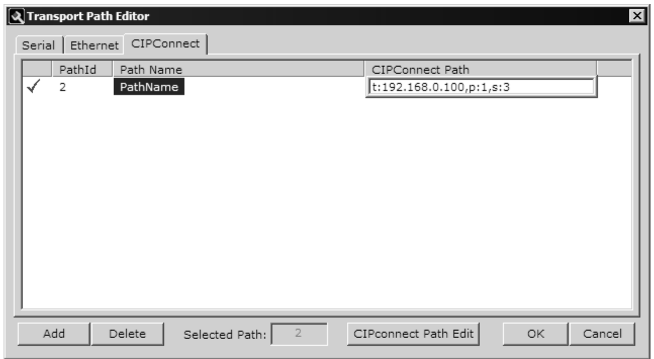
Note: Do not include the underscore (_) for the path name.



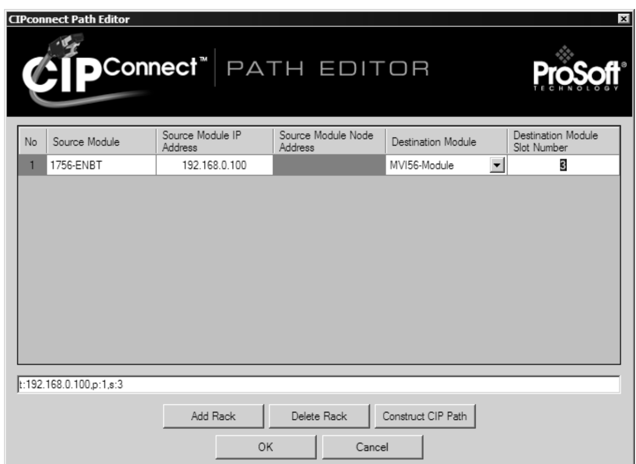
For MVI-PDPMV1 users: Select the **SERIAL** tab and enter the COM port number on your PC connected to the module. Enter a descriptive path name.



For MVI56-PDPMV1 and MVI69-PDPMV1 users with CIPconnect: Select the **CIPCONNECT** tab, and then click the **ADD** button.



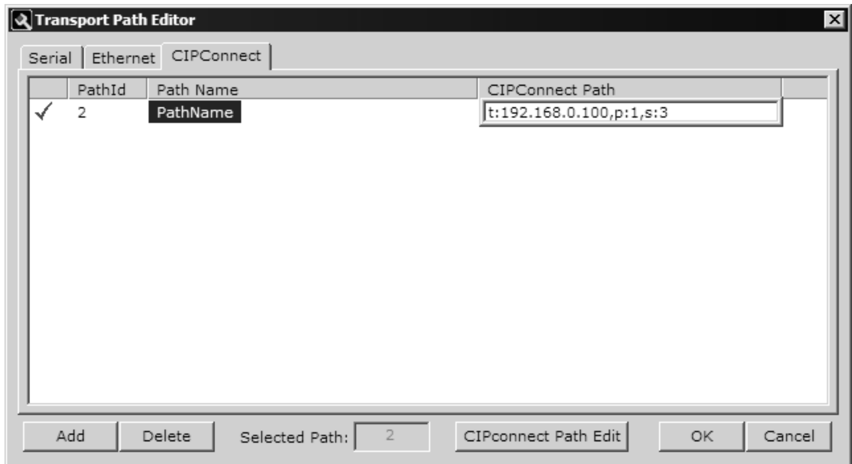
Click the **CIPCONNECT PATH EDIT** button to define the path for this application. The CIPconnect Path Editor allows the configuration of the path between your PC and the MVI56-PDPMV1 or MVI69-PDPMV1 module.
For the following example, the PC will be connected through Ethernet to a 1756-ENBT communication card (IP=192.168.0.100) and the MVI56-PDPMV1 card is located in slot 3 of the same rack.



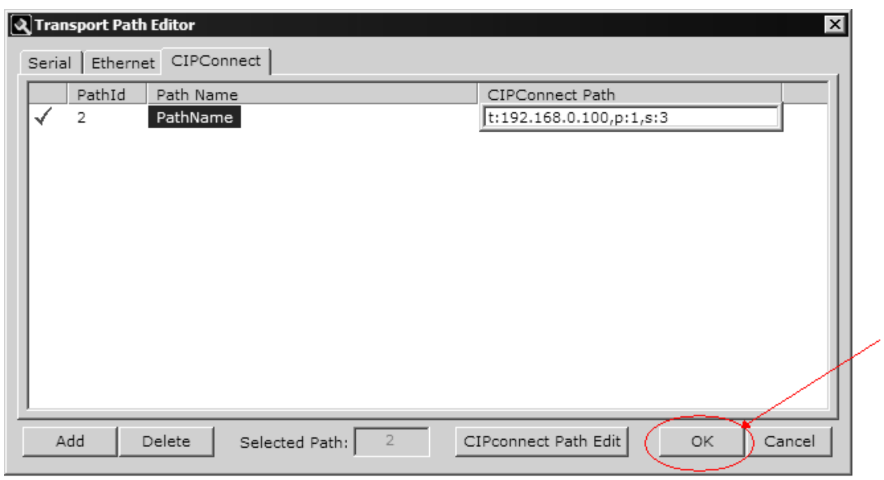
For more information, please refer to Using the CIPconnect Path Editor.

Note: CIPconnect is available for MVI56-PDPMV1 firmware version 1.28.000 (or later) and for MVI69-PDPMV1 firmware version 1.37.002 (or later). This functionality requires comDTM version 1.0.1.5 with install version 1.01.0003 (or later). For information on how to check the comDTM version and install version, refer to Verifying the comDTM Version and comDTM Install Version (page 273).

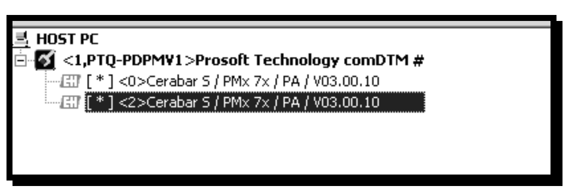
4 When you have configured the communication path, click the **OK** button to confirm. The communication path will be displayed at the top grid panel as shown in the following illustration.



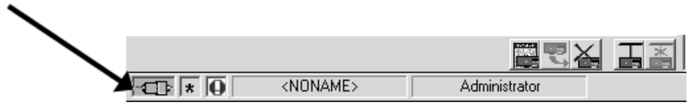
5 Select the path and click the **OK** button to exit the *Transport Path Editor* window.



6 Select the **COMDTM** icon and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **CONNECT**. If the connection is successful, the icon will be highlighted, as shown in the following illustration.



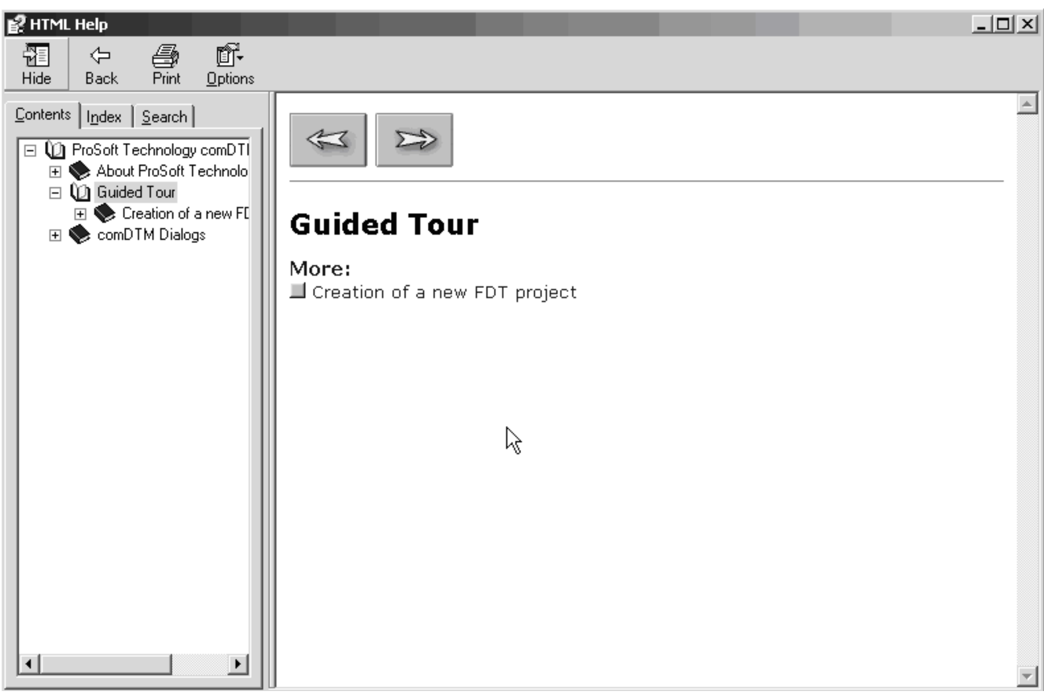
When the comDTM is connected with the Master, PACTware indicates the connection Master by displaying a green plug in the status bar.



This completes the installation and Quick Start Guide for the ProSoft Technology PROFIBUS comDTM. Refer to the online help and documentation additional information on each DTM component you have installed and configured.

The comDTM provides a *Guided Tour* section in the online help that explains the basic features and operation of the program. To open the online help, click the right mouse button on **PROSOFT TECHNOLOGY COMDTM**, and choose **ADDITIONAL FUNCTIONS** > **ONLINE HELP** from the shortcut menu.

Click the **GUIDED TOUR** icon. Use the navigation buttons on each help page to view the help topics.



Refer to the documentation and online help for your FDT frame program for specific FDT frame instructions.

9.3.6 Verifying the comDTM Version and comDTM Install Version

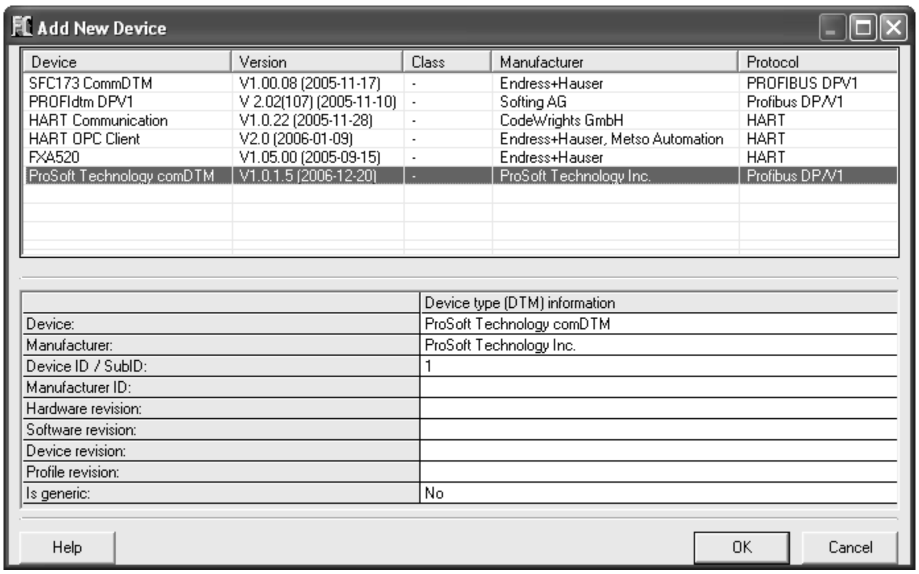
Introduction

There are two versions associated to the comDTM – the comDTM version and the comDTM install version. Starting with comDTM version 1.0.1.5, each upgrade will indicate the same comDTM version but a different comDTM install version.

This section describes how to check the comDTM version and comDTM install version.

Checking the comDTM Version

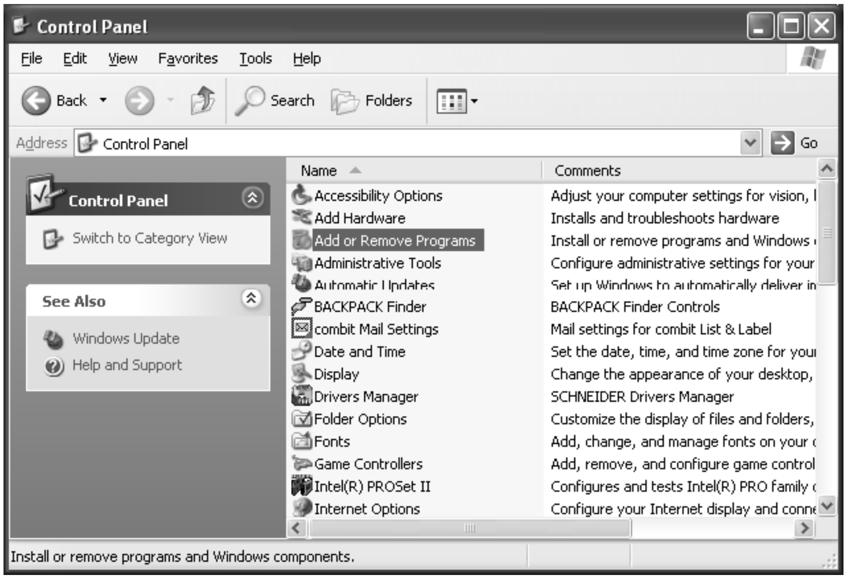
Refer to the *Version* column indicated when you add the comDTM to the DTM Container project.



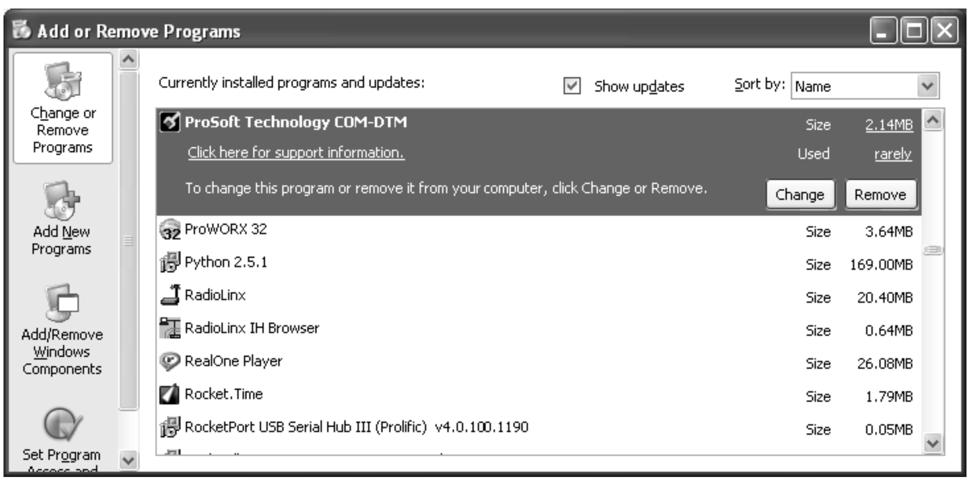
Checking the comDTM Install Version

1 Click the **START** menu and then choose **CONTROL PANEL**.

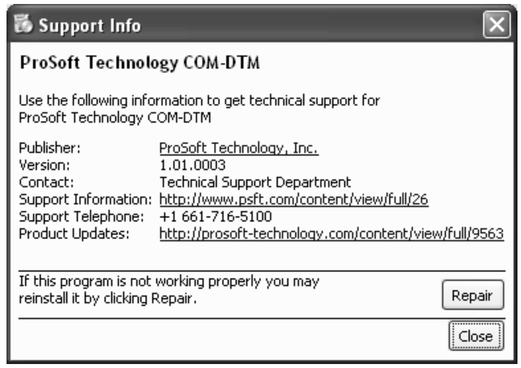
2 In the list of **CONTROL PANEL** applets, select **ADD OR REMOVE PROGRAMS**.



3 Select **PROSOFT TECHNOLOGY COM-DTM**, and then click on the link **CLICK HERE FOR SUPPORT INFORMATION**.

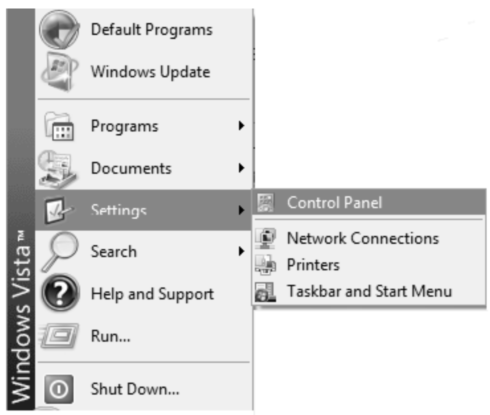


4 You will see the comDTM Install Version in the *Version* field, as shown in the following illustration (1.01.0003 for this example).



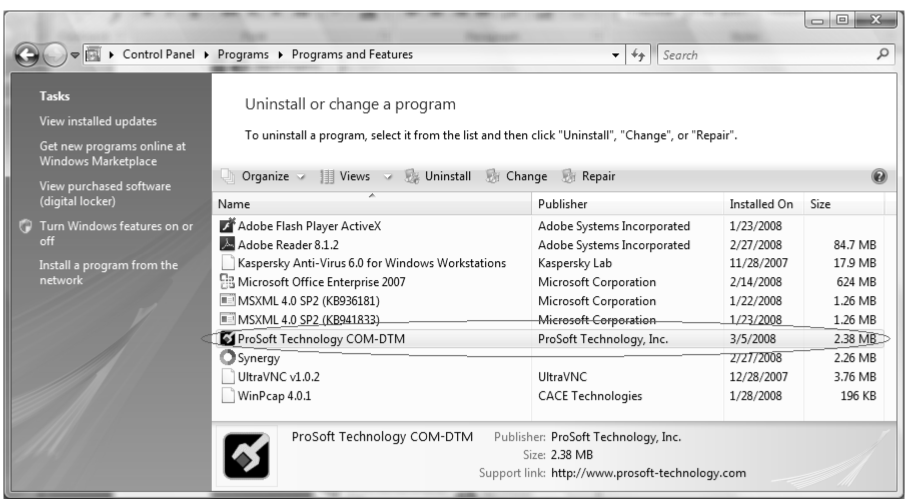
Checking the Install Version for Vista

1 Select **CONTROL PANEL**.

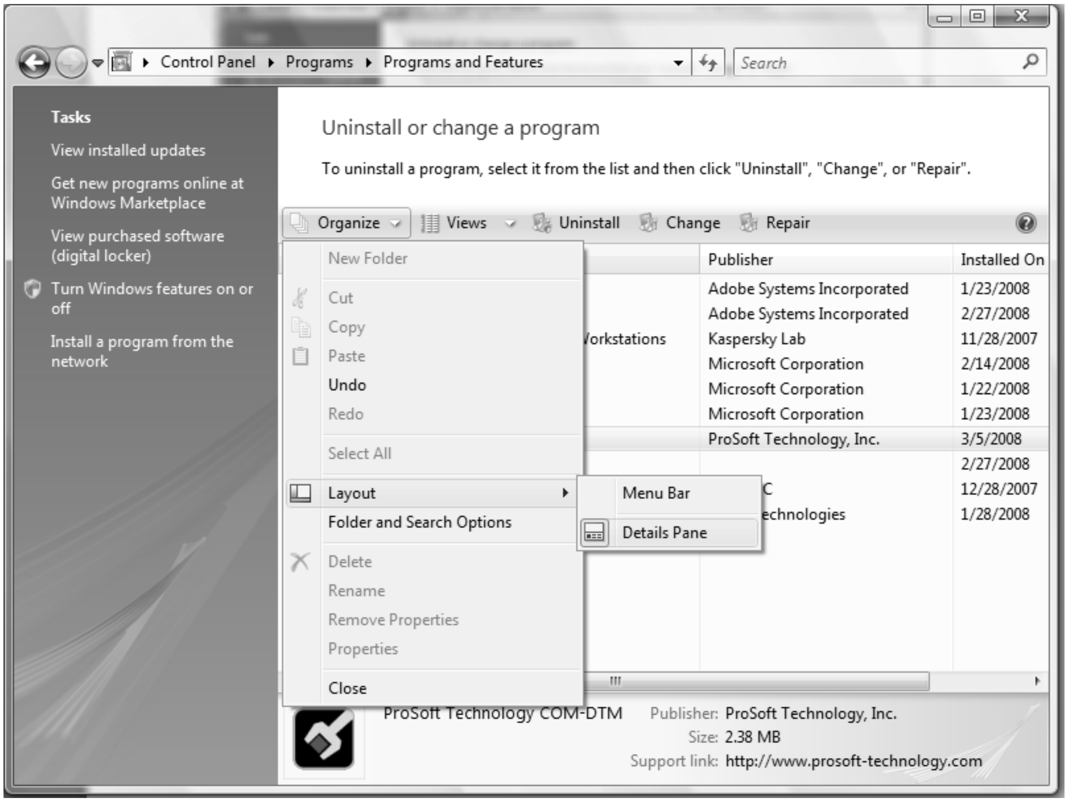


2 Select **UNINSTALL PROGRAMS**.

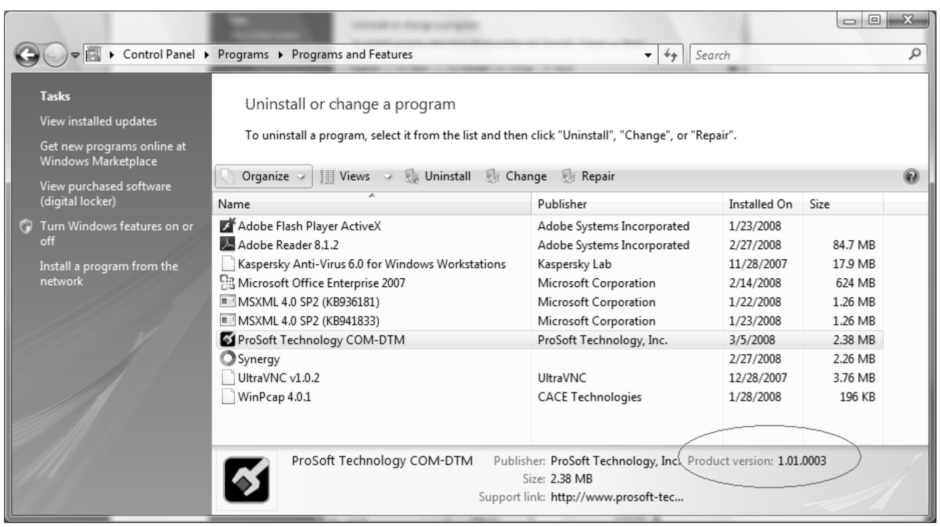
3 Select Prosoft Technology COM-DTM (click once)



4 Click the **ORGANIZE** tab and select **LAYOUT** > **DETAILS PANE**.



5 Check the Install Version at the bottom right portion of the window.



9.4 Cable Connections

The PTQ-PDPMV1 module has the following communication connections on the module:

- One Ethernet port (RJ45 connector)
- One RS-232 Configuration/Debug port (DB9 connector)

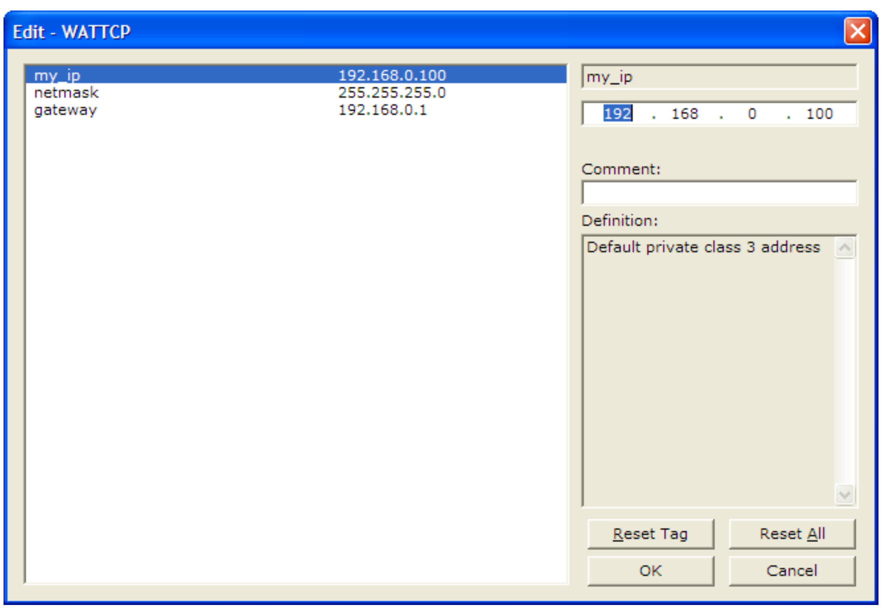
9.4.1 Ethernet Connection

The PTQ-PDPMV1 module has an RJ45 port located on the front of the module, labeled *Ethernet*, for use with the TCP/IP network. The module is connected to the Ethernet network using an Ethernet cable between the module's Ethernet port and an Ethernet switch or hub.

Note: Depending on hardware configuration, you may see more than one RJ45 port on the module. The Ethernet port is labeled *Ethernet*.
Warning: The PTQ-PDPMV1 module is NOT compatible with Power Over Ethernet (IEEE802.3af / IEEE802.3at) networks. Do NOT connect the module to Ethernet devices, hubs, switches or networks that supply AC or DC power over the Ethernet cable. Failure to observe this precaution may result in damage to hardware, or injury to personnel.
Important: The module requires a static (fixed) IP address that is not shared with any other device on the Ethernet network. Obtain a list of suitable IP addresses from your network administrator BEFORE configuring the Ethernet port on this module.

Ethernet Port Configuration - wattcp.cfg

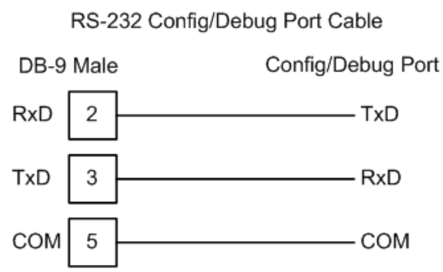
The wattcp.cfg file must be set up properly in order to use a TCP/IP network connection. You can view the current network configuration in *ProSoft Configuration Builder (PCB)*, as shown:



You may also view the network configuration using a PC serial port connection and an ASCII terminal program (like Windows HyperTerminal) by selecting **[@]** (Network Menu) and **[V]** (View) options when connected to the Debug port. For more information on serial port access, see the chapter on Diagnostics and Troubleshooting (page 223).

9.4.2 RS-232 Configuration/Debug Port

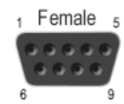
This port is physically an RJ45 connection. An RJ45 to DB-9 adapter cable is included with the module. This port permits a PC-based terminal emulation program to view configuration and status data in the module and to control the module. The cable pinout for communications on this port is shown in the following diagram.



9.5 PROFIBUS Master Port

The following diagram has been imported from the PROFIBUS Master documentation. Note that the signals to reference are the D-Sub signals in the table.

D-Sub (male)	Board to Board	Screw Terminal	Signal
Housing	1	5	Cable shield
1	4	-	-
2	7	-	-
3	6	4	B-Line
4	3	6	RTS
5	2	2	GND_BUS
6	8	1	+5V BUS (output)
7	9	-	-
8	5	3	A-Line
9	10	-	-

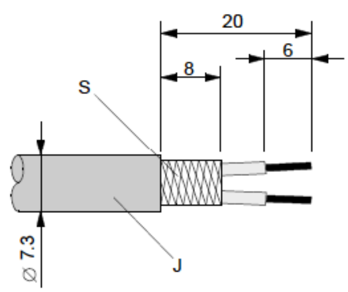


9.5.1 Constructing a Bus Cable for PROFIBUS DP

The bus cable for connecting PROFIBUS DP devices must be constructed by the user. A special PROFIBUS cable (twisted pair) is required here. This standard cable is available from various manufacturers and is a Belden part number 3079A.

<u>To construct the cable</u>

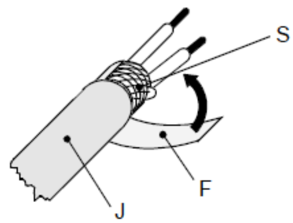
1 Cut the cable to the required length.
2 Prepare the cable ends as shown in the illustration (dimensions in mm):



J PVC jacket
S Braided shielding

3 Remove the PVC jacket J to the indicated length.

4 Wrap the provided copper shielding F around the shield braiding S:



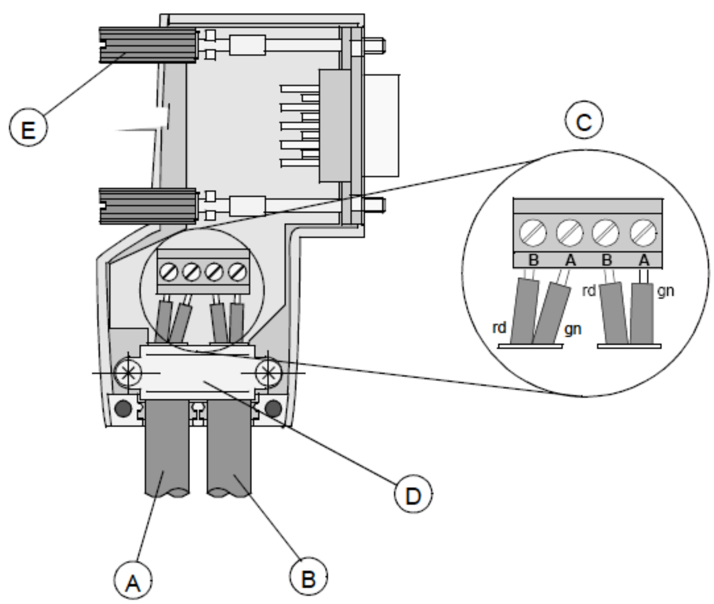
J PVC jacket
S Braided shielding
F Copper foil shielding
Additional foil can be obtained from 3M.

5 Plug the leads of the corresponding cable(s) into the terminals as shown:

- Green leads in terminal A
- Red lead in terminal B

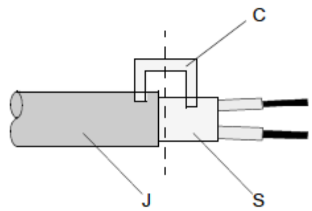
- **Note:** Do **not** tighten the corresponding screws yet.

Connection terminal assignment on the PROFIBUS DP:



A Incoming cable
B Outgoing cable
C Connection terminals (only once (B,A))
D Cable cleat for relieving tension
E Bus connector screws

6 Attach the cables with the provided cable cleat to create a robust shielded connection and to relieve any tension as shown:



J PVC jacket
S Braided shielding with foil shielding
C Cable cleat

- **Note**: Half of the cable jacket must lie under the cable cleat!

Pay attention to the cable cleat installation instructions.

7 Fasten the individual wires of the PROFIBUS cable to the terminals.
8 Close the connector housing.

- **Note:** The shielding of both cables is connected internally with the metal housing of the connector.

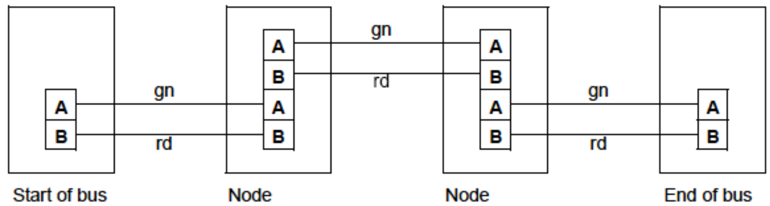
9 Complete the Central Shielding Measures (below) and grounding operations for the shielding before you connect the cable connector to the module.
10 Plug the PROFIBUS DP connector into the module and secure it with the screws.

Bus Begin and Bus End

The PROFIBUS connector with termination is required at the beginning and the end of the bus. These connectors emulate the line impedance.

It is recommended that at least one connector with diagnostics interface is used.

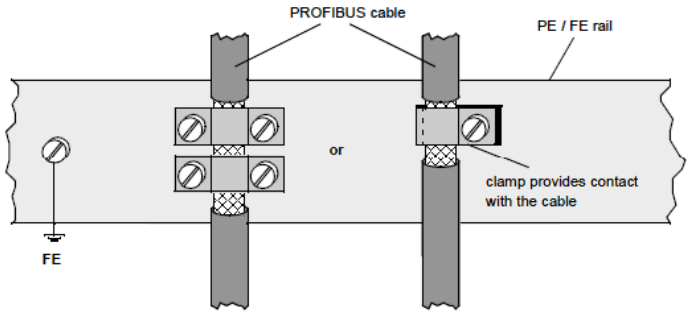
Wiring diagram for a PROFIBUS DP cable:



Grounding and Shielding for Systems with Equipotential Bonding

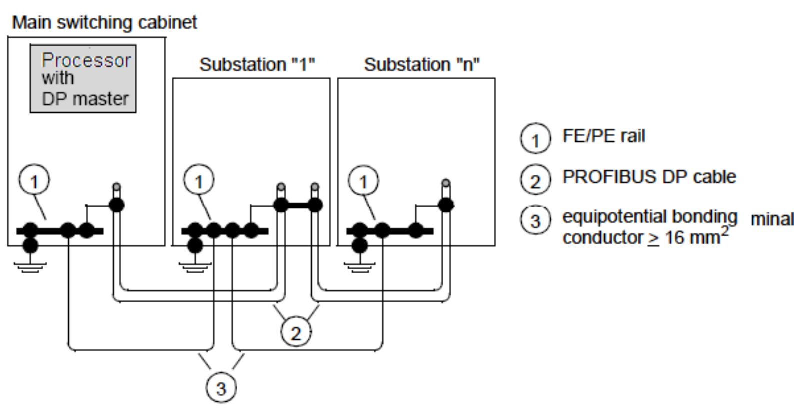
Each cable shield should be galvanically grounded with the earth using FE/PE grounding clamps immediately after the cable has been connected to the cabinet.

This example indicates the shielding connection from the PROFIBUS cable to the FE/PE rail.



Note: An equalization current can flow across a shield connected at both ends because of fluctuations in ground potential. To prevent this, it is imperative that there be potential equalization between all the attached installation components and devices.

This example indicates the system components and devices in a system with equipotential bonding.

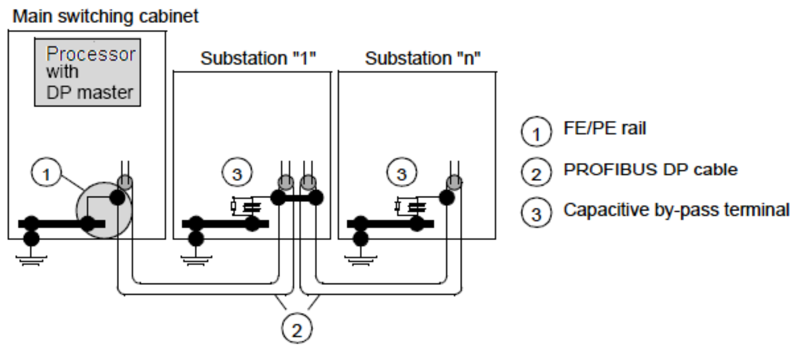


Grounding and Shielding for Systems without Equipotential Bonding

Note: Grounding and shielding is to be carried out the same as for systems **with** equipotential bonding.

If this is not possible because of system or construction specific reasons however, use distributed ground with a capacitive coupling of high frequency interference signals.

This representation shows distributed grounding with capacitive coupling.



9.6 Supported PROFIBUS Services

The following table lists all available services according to the PROFIBUS specification.

Service	PROFIBUS Version	Master Class 1		Master Class 2	
		Request	**Response**	**Request**	**Response**
DDLM_Data-Exchange	DPV0	Yes		No	
DDLM_Set_Prm	DPV0	Yes		No	
DDLM_Chk_cfg	DPV0	Yes		No	
DDLM Slave Diag	DPV0	Yes		No	
DDLM_Global_Control	DPV0	Yes		No	
DDLM_Get_Cfg	DPV0			Yes	
DDLM_Set_Slave_Add	DPV0			Yes	
DDLM_Read_Input	DPV0			No	
DDLM_Read_Output	DPV0			No	
DDLM_Get_Master_Diag	DPV0		Yes		
DDLM_Start_Seq	DPV0		No	No	
DDLM_Download	DPV0		No	No	
DDLM_Upload	DPV0		No	No	
DDLM_End_Seq	DPV0		No	No	
DDLM_Act_Param_Brct	DPV0		No	No	
DDLM_Act_Param	DPV0		No	No	
MSAC1_Read	DPV1	Yes			
MSAC1_Write	DPV1	Yes			
MSAL1_Alarm	DPV1		Yes		
MSAL1_Alarm_Ack	DPV1		Yes		
MSAC2_Initiate	DPV1			No	
MSAC2_Read	DPV1			No	
MSAC2_Write	DPV1			No	
MSAC2_DataTransport	DPV1			No	
MSAC2_Abort	DPV1			No	
Data_eXchange_Broadcast	DPV2	No			
Isochrone_mode (Takt sync)	DPV2	No			
Extended_Set_Prm (Subscriber)	DPV2	No			

9.7 Quantum to PTQ Communication Protocol

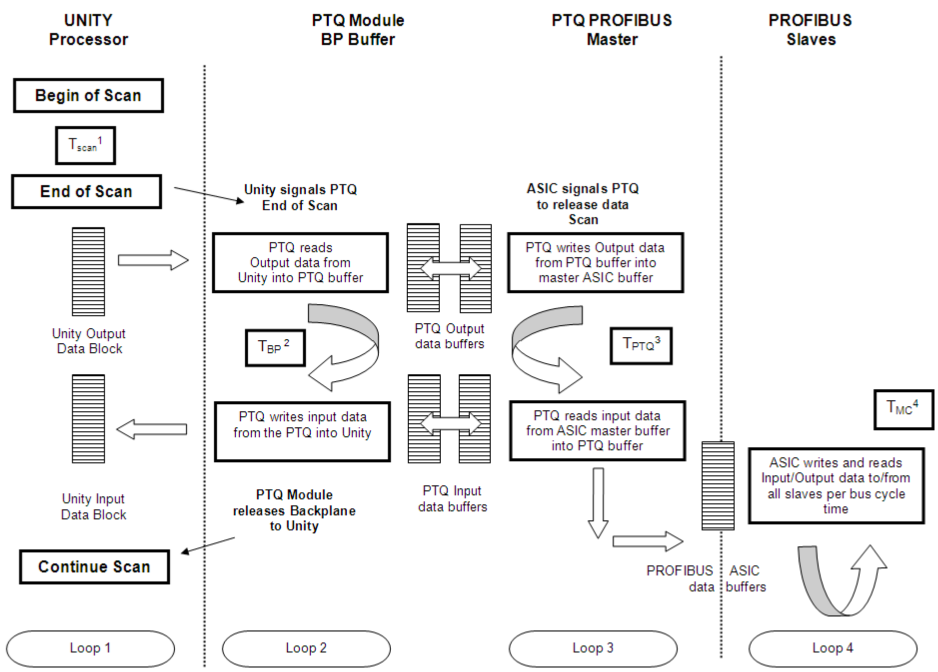
The vehicle utilized for transferring data between the PTQ module and the processor are two blocks of data:

- PTQ Input Data block
- PTQ Output Data block

Each of these data blocks (controlled by the PTQ) consists of a structure of data that provides for the movement of:

- Input Data image from PROFIBUS slave devices
- Output Data image for writing to PROFIBUS slave devices
- PTQ Module Configuration and Status (from PTQ to processor)
- PROFIBUS Messaging Mailbox commands (from processor to PTQ)
- PROFIBUS Messaging Mailbox responses (from PTQ to processor)

It is important to understand the process and flow of this data. The following illustration describes the mechanism in a block diagram overview. Several asynchronous data transfer loops occur simultaneously.



PLC Scan Loop 1

The PLC processor performs cyclic program, backplane, and network communication tasks for each PLC scan time. This time is referred to in this document as T_{SCAN} time. Refer to the Quantum processor manual for additional PLC cyclic task processing information.

Backplane Loop 2

When the processor reaches end of scan (EOS), the processor provides an interrupt to the PTQ. The PTQ locks out the processor and takes control of the backplane for a period of time to transfer all input and output data described above. After the data is transferred, the PTQ releases the backplane hook to the processor, and the processor continues with the next scan. The process repeats for every scan time. The time, T_{BP} backplane transfer time, is nominally 7.0 ms for a complete transfer of data. This is maximum time for all 1984 input bytes and 1838 output bytes (this includes cyclic, acyclic, mailbox command data, and input output status data). The user is able to control the number of input and output bytes within ProSoft Configuration Builder. The module supports 1536 bytes input data and 1536 bytes output cyclic data.

PTQ-PDPMV1 Main Loop 3

The PTQ transfers the data from the backplane buffer to the PROFIBUS Master buffer in preparation to condition the data for slave type data consistency. This time, T_{PTQ}, PTQ process time, is nominally 45 ms. During this loop, the PTQ module alternatively locks the input and output data areas and exchanges the data. It also transfers acyclic mailbox data.

PROFIBUS Loop 4

The PROFIBUS cycle time is based on many factors, including:

- synchronization time
- slave response time
- idle time
- bus baud rate

The cycle time "TMC" is calculated and added to processor scan time "T_{PTQ}" and "T_{BP}" time to arrive at the maximum response time of the PROFIBUS system.

9.8 Calculating System Response Time

Processor, PTQ module, and PROFIBUS system response times are essentially dependent on the following factors:

- T_{SCAN}^{1} processor scan time (Loop1)
- T_{BP}^{2} max. BP transfer time (Loop 2)
- T_{PTQ}^{3} max. PTQ process time (Loop 3)
- T_{MC}^{4} time of PROFIBUS message cycle time (Loop 4)

PTQ max. T_{SR} (System Response Time) = $T_{SCAN}^{1} + {}_{TBP}^{2} + T_{PTQ}^{3} + T_{MC}^{4}$

TSR = 7.0 ms + 45 ms + 2.84 ms + TSCAN1

where:

T_{BP}^{2} = ~7.0 ms
T_{PTQ}^{3} = max. ~45 ms
T_{MC}^{4} = 2.84 ms (12PDPMV1s)

9.8.1 How to Calculate PROFIBUS Time: TMC4

First, a review of a few terms before getting into the details of calculating bus cycle times:

Bit-Time: To help simplify timing calculations, it is convenient to normalize the time units with respect to the baud rate by using units of Bit-Time (Tbit). One bit-time is the time it takes to transmit one bit and is the reciprocal of the transmission rate (baud rate). For example:

- 1 Tbit (Bit Time) at 12 MB = 1/12000000 bps = 83 ns/bit
- 1 Tbit (Bit Time) at 1.5 MB = 1/1500000 bps = 667 ns/bit

Sync-Time (T_{SYN})

The synchronization time is the minimum time a station must remain in the idle state before it can accept another request. For PROFIBUS DP, an idle state of 33 Tbits (bit-time) must be present before every request telegram and this is called the sync-time.

Slave Reaction Time (T_{SDR})

The reaction time is the time it takes a slave to respond to a message. This time is often expressed as a minimum value (min T_{SDR}), or maximum value (max T_{SDR}). Min T_{SDR} is set within the parameterization telegram during startup. Max T_{SDR} varies with the transmission rate and is specified at the supported baud rates within the device GSD file. For PROFIBUS DP, this value may range from a minimum of 11 Tbits (min T_{SDR} default) to a maximum of 255 Tbits.

Initiator Delay Time (T_{SDI})

T_{SDI} refers to the station delay of the initiator of a request or token frame (the Master).

Initiator Idle Time (T_{ID}^1)

After receiving the last character of a telegram, the initiator must wait this amount of time before it sends the next telegram. The idle time (T_{ID}^1) is the time between transmission of the last bit of a frame (no acknowledge) and the transmission of the first bit of the next frame. It is at least the sync time (T_{SYN}), plus some safety margin (T_{SM}), but is also calculated as the maximum of these three values: T_{SYN} + T_{SM}, min $_{TSDR}$, or T_{SDI} (station delay of telegram initiator). The addition of safety margin (T_{SM}) is very important at high baud rates.

Minimum Slave Interval

The minimum slave interval is the minimum time that must expire between two slave polling cycles in which a slave can exchange data with the Master. To permit the slave station to be able to respond during every data cycle, it controls the bus cycle with this parameter. It is defined in the slave's GSD file via the parameter *Min_Slave_Interval*, which is specified as a 16-bit factor of 100 μs (*Min_Slave_Interval* = 1 is 100 μs). On some older equipment, the PROFIBUS link was implemented in software (as opposed to within the slave ASIC) and a typical value was about 2 ms. On newer equipment with modern ASICs, values down to 100 μs can be achieved.

9.8.2 Calculating System Reaction Time

A simplified calculation of system reaction time for a PROFIBUS DP system is derived from the following parameters:

- T_{SDR} (Station Reaction Time)
- The Transmission (Baud) Rate
- The Net Data Length specified
- Min_Slave_Interval (min time between two slave polling cycles)

Example:

One Master and *x* slaves are connected via PROFIBUS DP. We will use the maximum available 1536 bytes of output data and 1536 bytes of input data. These are to transfer to the maximum number slaves allowed, using 1536 input and output bytes at 12PDPMV1s. Each slave utilizes an SPC3 ASIC.

To calculate the relative bus cycle time for this network:

Let T_{MC} = Time of 1 telegram cycle (request telegram + T_{SDR} + slave response).
Let T_{BC} = Time of 1 bus cycle (the sum of all telegram cycles).

Given:

T_{SYN} = 33 TBits (Bus idle time or PROFIBUS Sync-Time)
T_{ID}^1 = 75 TBits (SPC3 bus idle time, at 1.5 MB $_{TID}^1$ = 36 TBit).
T_{SDR} = 30 TBits typical for baud rates ≥ 1.5 MB (SPC3 ASIC).
Min_Slave_Interval = 1 (100 μs, from slave GSD file).

Calculate:

1 Tbit (Bit Time) at 12 MB = 1/12000000 bps = 83 ns/bit

In data exchange mode, a telegram header consists of only 9 character bytes. If we include the bits of the character frame, there are 11 bits for every character byte (Start Bit + 8bits/char + Stop Bit + Parity). Because only 1 Master is present, we can ignore the token hold time of token telegrams. Thus, the basic time required by one telegram cycle (not including data) is obtained by adding the relevant bus times and the time to transmit the telegram header as follows:

T_{MC}^{4} (in TBits) = 2 * 9(header byte length) * 11 bits/byte + T_{SDR} + T_{SYN} + Tid[1]

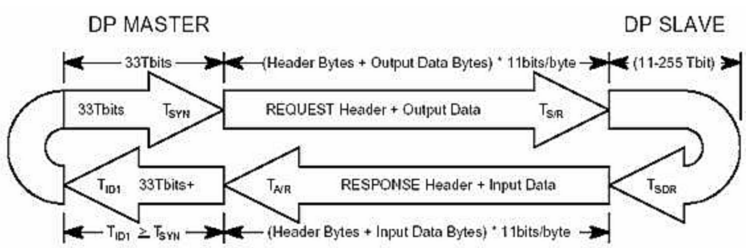
T_{MC}^{4} = 198 bits + 30 bits + 33 bits + 75 bits = 336 Tbits

T_{MC4} (μs) = 336 Tbits * 83 ns/Tbit = 28 μs

Thus, 28 μs is the basic time required by the telegram header including the bus times, without accounting for the data. For our example, we must include the data (1536 bytes Output + 1536 bytes Input - maximum bytes for a PTQ-PDPMV1). The time for a single telegram cycle with this data included is:

TMC4 = [336 Tbits] + amount of net data = 336 + [(1536 bytes Output + 1536 bytes Input)*(11 bits/byte)] = 34128 Tbits
TMC4 = 34128 * 83 ns/bit
TMC4 = 2.84 ms

To simplify this calculation, you can assume that a basic transfer time of 28 us plus approximately 1 μs per DU data byte (actually 0.83 μs/byte) is required to complete a telegram cycle. The following illustration gives an overview of the dominant bus times in a telegram cycle (assuming no interference or repetitions).



TELEGRAM CYCLE WITH RELEVANT BUS TIMES

Timing of 1 Message Cycle = $T_{MC} = ((T_{S/R} + T_{SDR} + T_{A/B})*T_{TD}) + T_{ID}$

Note that the slave has a *Min_Slave_Interval* of 100 μs and this dominates the bus timing for one telegram cycle. However, the *Min_Slave_Interval* is 100 μs between two polling cycles at the same station. If you have at least 3 stations present, then the actual transmission time at 12 MB will become the determining time factor for the bus cycle rather than the *Min_Slave_Interval*. Refer to the EN50170 standard for a more detailed calculation of transmission time.

Reference: INTRODUCTION TO PROFIBUS DP, ACROMAG INCORPORATED.

9.9 Using Multiple PTQ-PDPMV1 Modules with Concept

If your application requires more than one PTQ-PDPMV1 module for a single Quantum processor, you must manually merge the exported DTY files for each module into a single DTY file.

The only difference between the DTY files for each module is the PROFIBUS data types (input and output) that define the data associated to configured slaves. If you have already exported the processor files for the first modules (C:\project\DFB), repeat the following steps for each additional module to include in the Concept project.

1 Export the new processor files to a different folder (for example C:\temp).

2 Use a text editor such as Notepad.exe to open the exported DTY file. Select and copy the PROFIBUS "DATAF" data type definitions. These are the last two data types defined in the DTY file. The names of these data types will vary depending on the module name you entered, but they will always have "_IN_DATAF" and "OUT_DATAF" suffixes. For example, if the module name was configured as "PTQPDPMV1", select and copy the following data types:

```
TYPE PTQPDPMV1_IN_DATAF:
...
...
END_TYPE
TYPE PTQPDPMV1_OUT_DATAF:
...
...
END_TYPE
```

3 Open the DTY file in C:\project\DFB and paste the data types at the end of the file. Save and close the file.

With this procedure, you will obtain a final merged DTY file for all PTQ-PDPMV1 modules for your application.

9.10 Frequently Asked Questions

9.10.1 How do I configure the module?

The PTQ-PDPMV1 requires a simple text-based configuration file to make it operational.

9.10.2 Is a .MDC available for configuration of the module?

Yes. The CD-ROM that ships with the module should have a version for both Concept 2.5 and 2.6 in the PTQ-PDPMV1 directory.

9.10.3 Does the module work in a remote rack?

The module is designed to be located in the chassis with the PLC and will not operate in a remote chassis. If your application requires remote placement of the communication device you should investigate other members of the ProSoft Technology family such as the ProLinx gateway solutions.

9.10.4 Can I use the module in a hot backup system?

The PTQ-PDPMV1 module supports the 140CPU67160 Hot Standby processor. Refer to Hot Standby Support (page 185) for setup and configuration instructions.

10 Support, Service & Warranty

In This Chapter

Contacting Technical Support

ProSoft Technology, Inc. (ProSoft) is committed to providing the most efficient and effective support possible. Before calling, please gather the following information to assist in expediting this process:

1. Product Version Number
2. System architecture
3. Network details

If the issue is hardware related, we will also need information regarding:

1. Module configuration and associated ladder files, if any
2. Module operation and any unusual behavior
3. Configuration/Debug status information
4. LED patterns
5. Details about the serial, Ethernet or fieldbus devices interfaced to the module, if any.

Note: *For technical support calls within the United States, an after-hours answering system allows 24-hour/7-days-a-week pager access to one of our qualified Technical and/or Application Support Engineers. Detailed contact information for all our worldwide locations is available on the following page.*

Internet	Web Site: www.prosoft-technology.com/support E-mail address: support@prosoft-technology.com
Asia Pacific (location in Malaysia)	Tel: +603.7724.2080, E-mail: asiapc@prosoft-technology.com Languages spoken include: Chinese, English
Asia Pacific (location in China)	Tel: +86.21.5187.7337 x888, E-mail: asiapc@prosoft-technology.com Languages spoken include: Chinese, English
Europe (location in Toulouse, France)	Tel: +33 (0) 5.34.36.87.20, E-mail: support.EMEA@prosoft-technology.com Languages spoken include: French, English
Europe (location in Dubai, UAE)	Tel: +971-4-214-6911, E-mail: mea@prosoft-technology.com Languages spoken include: English, Hindi
North America (location in California)	Tel: +1.661.716.5100, E-mail: support@prosoft-technology.com Languages spoken include: English, Spanish
Latin America (Oficina Regional)	Tel: +1-281-2989109, E-Mail: latinam@prosoft-technology.com Languages spoken include: Spanish, English
Latin America (location in Puebla, Mexico)	Tel: +52-222-3-99-6565, E-mail: soporte@prosoft-technology.com Languages spoken include: Spanish
Brasil (location in Sao Paulo)	Tel: +55-11-5083-3776, E-mail: brasil@prosoft-technology.com Languages spoken include: Portuguese, English

10.1 Return Material Authorization (RMA) Policies and Conditions

The following Return Material Authorization (RMA) Policies and Conditions (collectively, "RMA Policies") apply to any returned product. These RMA Policies are subject to change by ProSoft Technology, Inc., without notice. For warranty information, see Limited Warranty (page 297). In the event of any inconsistency between the RMA Policies and the Warranty, the Warranty shall govern.

10.1.1 Returning Any Product

a) In order to return a Product for repair, exchange, or otherwise, the Customer must obtain a Return Material Authorization (RMA) number from ProSoft Technology and comply with ProSoft Technology shipping instructions.

b) In the event that the Customer experiences a problem with the Product for any reason, Customer should contact ProSoft Technical Support at one of the telephone numbers listed above (page 293). A Technical Support Engineer will request that you perform several tests in an attempt to isolate the problem. If after completing these tests, the Product is found to be the source of the problem, we will issue an RMA.

c) All returned Products must be shipped freight prepaid, in the original shipping container or equivalent, to the location specified by ProSoft Technology, and be accompanied by proof of purchase and receipt date. The RMA number is to be prominently marked on the outside of the shipping box. Customer agrees to insure the Product or assume the risk of loss or damage in transit. Products shipped to ProSoft Technology using a shipment method other than that specified by ProSoft Technology, or shipped without an RMA number will be returned to the Customer, freight collect. Contact ProSoft Technical Support for further information.

d) A 10% restocking fee applies to all warranty credit returns, whereby a Customer has an application change, ordered too many, does not need, etc. Returns for credit require that all accessory parts included in the original box (i.e.; antennas, cables) be returned. Failure to return these items will result in a deduction from the total credit due for each missing item.

10.1.2 Returning Units Under Warranty

A Technical Support Engineer must approve the return of Product under ProSoft Technology's Warranty:

a) A replacement module will be shipped and invoiced. A purchase order will be required.
b) Credit for a product under warranty will be issued upon receipt of authorized product by ProSoft Technology at designated location referenced on the Return Material Authorization
   i. If a defect is found and is determined to be customer generated, or if the defect is otherwise not covered by ProSoft Technology s warranty, there will be no credit given. Customer will be contacted and can request module be returned at their expense;
   ii. If defect is customer generated and is repairable, customer can authorize ProSoft Technology to repair the unit by providing a purchase order for 30% of the current list price plus freight charges, duties and taxes as applicable.

10.1.3 Returning Units Out of Warranty

a) Customer sends unit in for evaluation to location specified by ProSoft Technology, freight prepaid.
b) If no defect is found, Customer will be charged the equivalent of $100 USD, plus freight charges, duties and taxes as applicable. A new purchase order will be required.
c) If unit is repaired, charge to Customer will be 30% of current list price (USD) plus freight charges, duties and taxes as applicable. A new purchase order will be required or authorization to use the purchase order submitted for evaluation fee.

The following is a list of non-repairable units:

- 3150 - All
- 3750
- 3600 - All
- 3700
- 3170 - All
- 3250
- 1560 - Can be repaired, only if defect is the power supply
- 1550 - Can be repaired, only if defect is the power supply
- 3350
- 3300
- 1500 - All

10.2 LIMITED WARRANTY

This Limited Warranty ("Warranty") governs all sales of hardware, software, and other products (collectively, "Product") manufactured and/or offered for sale by ProSoft Technology, Incorporated (ProSoft), and all related services provided by ProSoft, including maintenance, repair, warranty exchange, and service programs (collectively, "Services"). By purchasing or using the Product or Services, the individual or entity purchasing or using the Product or Services ("Customer") agrees to all of the terms and provisions (collectively, the "Terms") of this Limited Warranty. All sales of software or other intellectual property are, in addition, subject to any license agreement accompanying such software or other intellectual property.

10.2.1 What Is Covered By This Warranty

a) *Warranty On New Products*: ProSoft warrants, to the original purchaser, that the Product that is the subject of the sale will (1) conform to and perform in accordance with published specifications prepared, approved and issued by ProSoft, and (2) will be free from defects in material or workmanship; provided these warranties only cover Product that is sold as new. This Warranty expires three (3) years from the date of shipment for Product purchased **on or after** January 1st, 2008, or one (1) year from the date of shipment for Product purchased **before** January 1st, 2008 (the "Warranty Period"). If the Customer discovers within the Warranty Period a failure of the Product to conform to specifications, or a defect in material or workmanship of the Product, the Customer must promptly notify ProSoft by fax, email or telephone. In no event may that notification be received by ProSoft later than 39 months from date of original shipment. Within a reasonable time after notification, ProSoft will correct any failure of the Product to conform to specifications or any defect in material or workmanship of the Product, with either new or remanufactured replacement parts. ProSoft reserves the right, and at its sole discretion, may replace unrepairable units with new or remanufactured equipment. All replacement units will be covered under warranty for the 3 year period commencing from the date of original equipment purchase, not the date of shipment of the replacement unit. Such repair, including both parts and labor, will be performed at ProSoft's expense. All warranty service will be performed at service centers designated by ProSoft.

b) *Warranty On Services*: Materials and labor performed by ProSoft to repair a verified malfunction or defect are warranteed in the terms specified above for new Product, provided said warranty will be for the period remaining on the original new equipment warranty or, if the original warranty is no longer in effect, for a period of 90 days from the date of repair.

10.2.2 What Is Not Covered By This Warranty

a) ProSoft makes no representation or warranty, expressed or implied, that the operation of software purchased from ProSoft will be uninterrupted or error free or that the functions contained in the software will meet or satisfy the purchaser's intended use or requirements; the Customer assumes complete responsibility for decisions made or actions taken based on information obtained using ProSoft software.

b) This Warranty does not cover the failure of the Product to perform specified functions, or any other non-conformance, defects, losses or damages caused by or attributable to any of the following: (i) shipping; (ii) improper installation or other failure of Customer to adhere to ProSoft's specifications or instructions; (iii) unauthorized repair or maintenance; (iv) attachments, equipment, options, parts, software, or user-created programming (including, but not limited to, programs developed with any IEC 61131-3, "C" or any variant of "C" programming languages) not furnished by ProSoft; (v) use of the Product for purposes other than those for which it was designed; (vi) any other abuse, misapplication, neglect or misuse by the Customer; (vii) accident, improper testing or causes external to the Product such as, but not limited to, exposure to extremes of temperature or humidity, power failure or power surges; or (viii) disasters such as fire, flood, earthquake, wind and lightning.

c) The information in this Agreement is subject to change without notice. ProSoft shall not be liable for technical or editorial errors or omissions made herein; nor for incidental or consequential damages resulting from the furnishing, performance or use of this material. The user guide included with your original product purchase from ProSoft contains information protected by copyright. No part of the guide may be duplicated or reproduced in any form without prior written consent from ProSoft.

10.2.3 Disclaimer Regarding High Risk Activities

Product manufactured or supplied by ProSoft is not fault tolerant and is not designed, manufactured or intended for use in hazardous environments requiring fail-safe performance including and without limitation: the operation of nuclear facilities, aircraft navigation of communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly or indirectly to death, personal injury or severe physical or environmental damage (collectively, "high risk activities"). ProSoft specifically disclaims any express or implied warranty of fitness for high risk activities.

10.2.4 Intellectual Property Indemnity

Buyer shall indemnify and hold harmless ProSoft and its employees from and against all liabilities, losses, claims, costs and expenses (including attorney's fees and expenses) related to any claim, investigation, litigation or proceeding (whether or not ProSoft is a party) which arises or is alleged to arise from Buyer's acts or omissions under these Terms or in any way with respect to the Products. Without limiting the foregoing, Buyer (at its own expense) shall indemnify and hold harmless ProSoft and defend or settle any action brought against such Companies to the extent based on a claim that any Product made to Buyer specifications infringed intellectual property rights of another party. ProSoft makes no warranty that the product is or will be delivered free of any person's claiming of patent, trademark, or similar infringement. The Buyer assumes all risks (including the risk of suit) that the product or any use of the product will infringe existing or subsequently issued patents, trademarks, or copyrights.

a) Any documentation included with Product purchased from ProSoft is protected by copyright and may not be duplicated or reproduced in any form without prior written consent from ProSoft.
b) ProSoft's technical specifications and documentation that are included with the Product are subject to editing and modification without notice.
c) Transfer of title shall not operate to convey to Customer any right to make, or have made, any Product supplied by ProSoft.
d) Customer is granted no right or license to use any software or other intellectual property in any manner or for any purpose not expressly permitted by any license agreement accompanying such software or other intellectual property.
e) Customer agrees that it shall not, and shall not authorize others to, copy software provided by ProSoft (except as expressly permitted in any license agreement accompanying such software); transfer software to a third party separately from the Product; modify, alter, translate, decode, decompile, disassemble, reverse-engineer or otherwise attempt to derive the source code of the software or create derivative works based on the software; export the software or underlying technology in contravention of applicable US and international export laws and regulations; or use the software other than as authorized in connection with use of Product.
f) **Additional Restrictions Relating To Software And Other Intellectual Property**

   In addition to compliance with the Terms of this Warranty, Customers purchasing software or other intellectual property shall comply with any license agreement accompanying such software or other intellectual property. Failure to do so may void this Warranty with respect to such software and/or other intellectual property.

10.2.5 Disclaimer of all Other Warranties

The Warranty set forth in What Is Covered By This Warranty (page 297) are in lieu of all other warranties, express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

10.2.6 Limitation of Remedies **

In no event will ProSoft or its Dealer be liable for any special, incidental or consequential damages based on breach of warranty, breach of contract, negligence, strict tort or any other legal theory. Damages that ProSoft or its Dealer will not be responsible for include, but are not limited to: Loss of profits; loss of savings or revenue; loss of use of the product or any associated equipment; loss of data; cost of capital; cost of any substitute equipment, facilities, or services; downtime; the claims of third parties including, customers of the Purchaser; and, injury to property.

** Some areas do not allow time limitations on an implied warranty, or allow the exclusion or limitation of incidental or consequential damages. In such areas, the above limitations may not apply. This Warranty gives you specific legal rights, and you may also have other rights which vary from place to place.

10.2.7 Time Limit for Bringing Suit

Any action for breach of warranty must be commenced within 39 months following shipment of the Product.

10.2.8 No Other Warranties

Unless modified in writing and signed by both parties, this Warranty is understood to be the complete and exclusive agreement between the parties, suspending all oral or written prior agreements and all other communications between the parties relating to the subject matter of this Warranty, including statements made by salesperson. No employee of ProSoft or any other party is authorized to make any warranty in addition to those made in this Warranty. The Customer is warned, therefore, to check this Warranty carefully to see that it correctly reflects those terms that are important to the Customer.

10.2.9 Allocation of Risks

This Warranty allocates the risk of product failure between ProSoft and the Customer. This allocation is recognized by both parties and is reflected in the price of the goods. The Customer acknowledges that it has read this Warranty, understands it, and is bound by its Terms.

10.2.10 Controlling Law and Severability

This Warranty shall be governed by and construed in accordance with the laws of the United States and the domestic laws of the State of California, without reference to its conflicts of law provisions. If for any reason a court of competent jurisdiction finds any provisions of this Warranty, or a portion thereof, to be unenforceable, that provision shall be enforced to the maximum extent permissible and the remainder of this Warranty shall remain in full force and effect. Any cause of action with respect to the Product or Services must be instituted in a court of competent jurisdiction in the State of California.

Glossary of Terms

A

Active Master

An active Master is controlled by the Primary PLC and exchanges I/O data, diagnostics and parameter data with its assigned slaves.

Application

If nothing else is stated, the term "application" refers to the application of the Master module.

ASPC2

Bus controller ASIC

C

Counterpart

The remote Master.

CSI

Crossed Status Information

F

FDL-layer

Lower layer of a PROFIBUS communication stack (Layer 2 of the OSI model).

H

Hot Standby (HSBY)

Refers to a redundant system with one Primary PLC and one Standby PLC where the Standby PLC is ready to take over if the Primary PLC fails.

P

Passive Master

A passive Master is controlled by the Standby PLC and is ready to take over the communication with the slaves if the active Master fails.

PCB (ProSoft Configuration Builder)

Software configuration tool for the Master module and PROFIBUS network.

R

RTOS

Real Time Operating System

S

SRD

FDL-service for Send and Receive Data in one request.

Switchover

A switchover occurs when the Standby PLC takes over control and becomes Primary.

T

TBD

Short for "To Be Defined"

Index

%

.

A

B

C

D

E

F

P

Q

R

S

T