



PTQ-MCM

Quantum Platform

Modbus Master/Slave Communication
Module

May 4, 2018

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, Inc.

+1 (661) 716-5100

+1 (661) 716-5101 (Fax)

www.prosoft-technology.com

support@prosoft-technology.com

Copyright © 2018 ProSoft Technology, Inc. All rights reserved.

PTQ-MCM User Manual

May 4, 2018

ProSoft Technology®, is a registered copyright of ProSoft Technology, Inc. All other brand or product names are or may be trademarks of, and are used to identify products and services of, their respective owners.

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 at our website:
<http://www.prosoft-technology.com>

Content Disclaimer

This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither ProSoft Technology nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. Information in this document including illustrations, specifications and dimensions may contain technical inaccuracies or typographical errors. ProSoft Technology makes no warranty or representation as to its accuracy and assumes no liability for and reserves the right to correct such inaccuracies or errors at any time without notice. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of ProSoft Technology. All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components. When devices are used for applications with technical safety requirements, the relevant instructions must be followed. Failure to use ProSoft Technology software or approved software with our hardware products may result in injury, harm, or improper operating results. Failure to observe this information can result in injury or equipment damage.

© 2018 ProSoft Technology. All Rights Reserved.

Printed documentation is available for purchase. Contact ProSoft Technology for pricing and availability.

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.

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 to 150 Hz
- Relative Humidity: 5% to 95% (without condensation)
- All phase conductor sizes must be at least 1.3 mm(squared) and all earth ground conductors must be at least 4mm(squared).

Markings:

CE	EMC-EN61326-1:2006; EN61000-6-4:2007
CSA	C22.2 No. 213-1987
CSA CB Certified	IEC61010
ATEX	EN60079-0 Category 3, Zone 2 EN60079-15

**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 μ A.

Maximum battery charge voltage = 3.4 Vdc.

Maximum battery charge current = 500 μ A.

Maximum battery discharge current = 30 μ A.

Contents

Your Feedback Please.....	2
How to Contact Us	2
Content Disclaimer.....	2
Information for ProTalk® Product Users.....	3
Warnings	3
Important Notice:.....	4
1 Start Here	9
1.1 Hardware and Software Requirements	10
1.1.1 Package Contents	10
1.1.2 Quantum Hardware	10
1.1.3 PC and PC Software	11
1.2 Installing ProSoft Configuration Builder Software	12
2 Configuring the Processor with Unity Pro	13
2.1 Creating a New Project	14
2.2 Adding the PTQ Module to the Project.....	16
2.3 Building the Project	18
2.4 Connect Your PC to the Processor	19
2.4.1 Connecting to the Processor with TCPIP.....	21
2.5 Downloading the Project to the Processor	22
3 Configuring the Processor with Concept	23
3.1 Information for Concept Version 2.6 Users	24
3.1.1 Installing MDC Configuration Files	24
3.2 Creating a New Project	26
3.3 Adding the PTQ Module to the Project.....	29
3.4 Setting up Data Memory in Project	32
3.5 Downloading the Project to the Processor	35
3.6 Verifying Successful Download	37
4 Configuring the Processor with ProWORX	41
5 Setting Up the ProTalk Module	45
5.1 Installing the ProTalk Module in the Quantum Rack.....	46
5.1.1 Verifying Jumper Settings	46
5.1.2 Inserting the 1454-9F connector	46
5.1.3 Installing the ProTalk Module in the Quantum Rack.....	47
5.2 Connect the PC to the ProTalk Configuration/Debug Port.....	48
6 Configuring the Module	51
6.1 Using ProSoft Configuration Builder	52

6.1.1	Set Up the Project.....	52
6.1.2	Renaming PCB Objects.....	54
6.2	Edit the Configuration File	55
6.2.1	[Module]	55
6.2.2	[Backplane Configuration].....	56
6.2.3	[MCM Port X]	58
6.2.4	[Modbus Port 1 Commands].....	64
6.3	Downloading the Project to the Module Using a Serial COM port.....	69
6.4	Verification and Troubleshooting	70
7 Diagnostics and Troubleshooting		71
7.1	LED Status Indicators	72
7.2	Using ProSoft Configuration Builder (PCB) for Diagnostics	73
7.2.1	Using the Diagnostic Window in ProSoft Configuration Builder	73
7.2.2	Navigation	75
7.2.3	Main Menu	75
7.2.4	Modbus Database View Menu.....	77
7.2.5	Backplane Menu	79
7.2.6	Data Analyzer	80
7.2.7	Protocol Serial Menu	83
7.2.8	Master Command Error List Menu.....	84
7.3	Reading Status Data from the Module	86
7.3.1	Error Status Table.....	86
8 Reference		87
8.1	Product Specifications	88
8.1.1	General Specifications	88
8.1.2	Hardware Specifications	89
8.1.3	General Specifications - Modbus Master/Slave.....	90
8.1.4	Functional Specifications	90
8.2	Functional Overview	91
8.2.1	About the Modbus Protocol	91
8.2.2	Backplane Data Transfer	92
8.2.3	Special Functions	93
8.2.4	Event Command Block	93
8.2.5	Command Control Block.....	94
8.2.6	Warm Boot Block (9998) - PTQ 1 to 63.....	97
8.2.7	Cold Boot Block (9999) - PTQ 1 to 63.....	97
8.2.8	Pass-Thru Control Blocks	98
8.3	Cable Connections	101
8.3.1	RS-232 Configuration/Debug Port.....	101
8.3.2	RS-232 Application Port(s)	101
8.3.3	RS-485 Application Port(s)	104
8.3.4	RS-422.....	104
8.4	Status Data Definition	105
8.4.1	Status Data Block Structure.....	105
8.5	Configuration Data	107
8.5.1	Port 1 Setup.....	107
8.5.2	Port 2 Setup.....	108
8.5.3	Port 1 Commands.....	108
8.5.4	Port 2 Commands.....	109

8.6	Modbus Protocol Specification	110
8.6.1	Commands Supported by the Module.....	110
8.6.2	Read Coil Status (Function Code 01)	111
8.6.3	Read Input Status (Function Code 02).....	112
8.6.4	Read Holding Registers (Function Code 03)	113
8.6.5	Read Input Registers (Function Code 04).....	114
8.6.6	Force Single Coil (Function Code 05)	115
8.6.7	Preset Single Register (Function Code 06).....	116
8.6.8	Diagnostics (Function Code 08)	117
8.6.9	Force Multiple Coils (Function Code 15).....	118
8.6.10	Preset Multiple Registers (Function Code 16)	119
8.6.11	Modbus Exception Responses.....	120
8.7	Frequently Asked Questions	123
8.7.1	What kind of data transfer rates can I expect between the PLC and the module?.....	123
8.7.2	Does the module work in a remote rack?.....	123
8.7.3	Can I use the module in a hot backup system?	123
9	Support, Service & Warranty	125
9.1	Contacting Technical Support	125
9.2	Warranty Information	127
Index		129

1 Start Here

In This Chapter

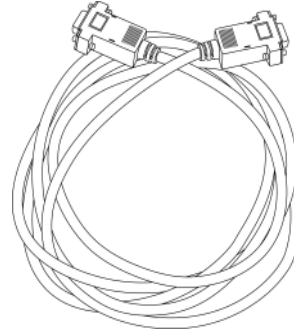
- ❖ Hardware and Software Requirements 10
- ❖ Installing ProSoft Configuration Builder Software 12

This guide is intended to guide you through the ProTalk module setup process, from removing the module from the box to exchanging data with the processor. In doing this, you will learn how to:

- Set up the processor environment for the PTQ module
- View how the PTQ module exchanges data with the processor
- Edit and download configuration files from your PC to the PTQ module
- Monitor the operation of the PTQ module

1.1 Hardware and Software Requirements

1.1.1 Package Contents



ProTalk Module

Null Modem Serial Cable



1454-9F DB-9 Female to 9 Pos Screw Terminal adapter (Serial protocol modules only)

Note: The DB-9 Female to 5 Pos Screw Terminal adapter is not required on Ethernet modules and is therefore not included in the carton with these types of modules.

1.1.2 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 proceeding:

- 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

1.1.3 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 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.

Installing ProSoft Configuration Builder from the ProSoft Technology website

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

2 Configuring the Processor with Unity Pro

In This Chapter

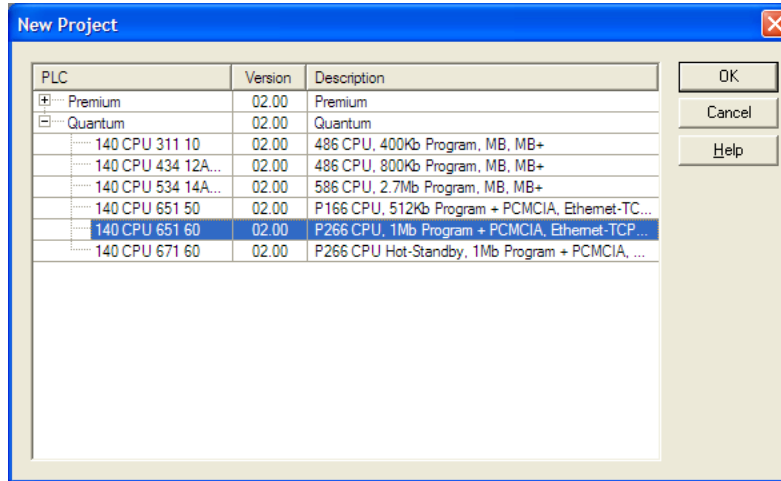
- ❖ Creating a New Project..... 14
- ❖ Adding the PTQ Module to the Project 16
- ❖ Building the Project 18
- ❖ Connect Your PC to the Processor 19
- ❖ Downloading the Project to the Processor..... 22

The following steps are designed to ensure that the processor (Quantum or Unity) is able to transfer data successfully with the PTQ module. As part of this procedure, you will use Unity Pro to create a project, add the PTQ module to the project, set up data memory for the project, and then download the project to the processor.

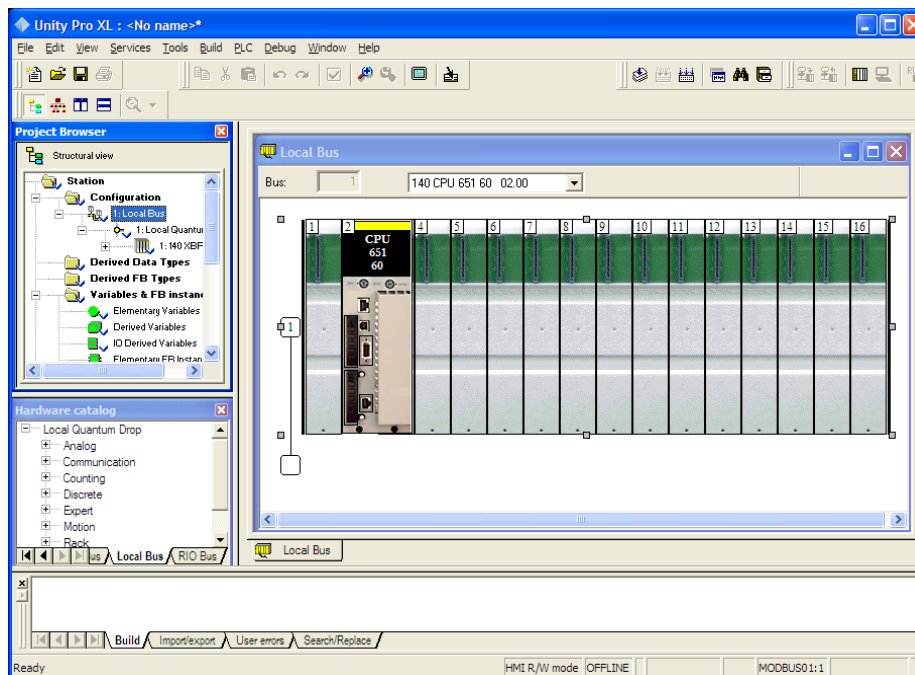
2.1 Creating a New Project

The first step is to open Unity Pro and create a new project.

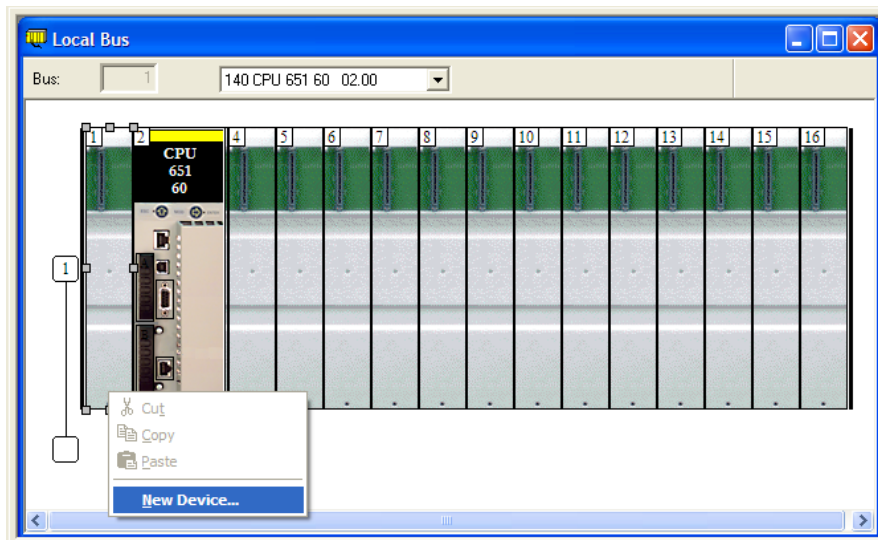
- 1 In the *New Project* dialog box, choose the CPU type. In the following illustration, the CPU is 140 CPU 651 60. Choose the processor type that matches your own hardware configuration, if it differs from the example. Click **OK** to continue.



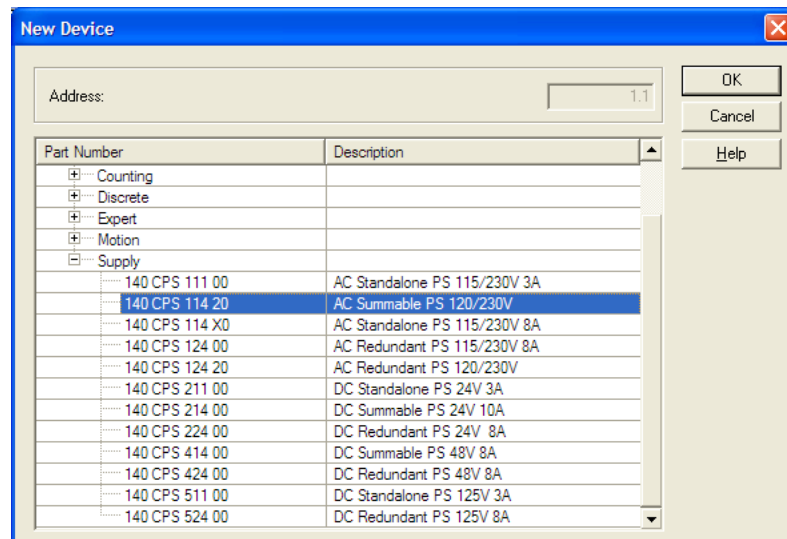
- 2 Next, add a power supply to the project. In the *Project Browser*, expand the *Configuration* folder, and then double-click the **1:LOCALBUS** icon. This action opens a graphical window showing the arrangement of devices in your Quantum rack.



- 3 Select the rack position for the power supply, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW DEVICE**.



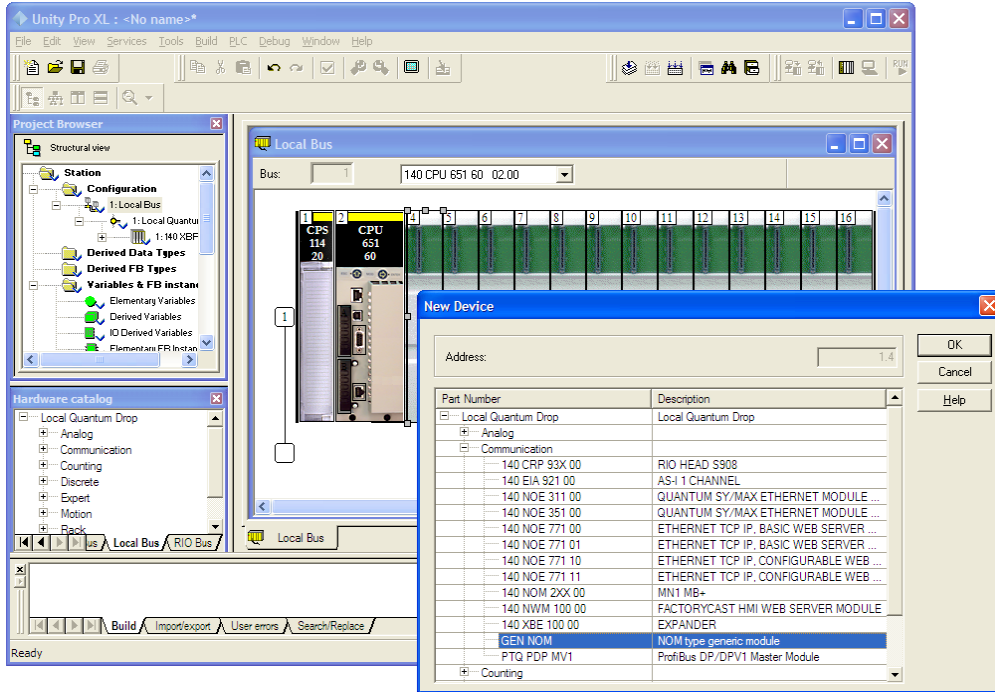
- 4 Expand the *Supply* folder, and then select your power supply from the list. Click **OK** to continue.



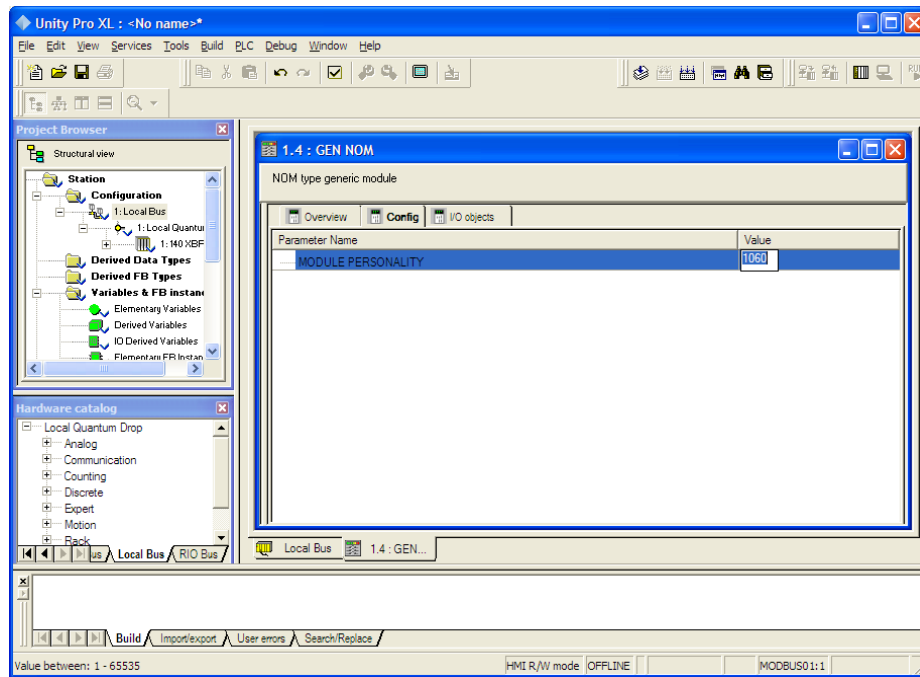
- 5 Repeat these steps to add any additional devices to your Quantum Rack.

2.2 Adding the PTQ Module to the Project

- 1 Expand the *Communication* tree, and select **GEN NOM**. This module type provides extended communication capabilities for the Quantum system, and allows communication between the PLC and the PTQ module without requiring additional programming.



- Next, enter the module personality value. The correct value for ProTalk modules is 1101 decimal (044D hex).



- Before you can save the project in Unity Pro, you must validate the modifications. Open the **EDIT** menu, and then choose **VALIDATE**. If no errors are reported, you can save the project.
- SAVE** the project.

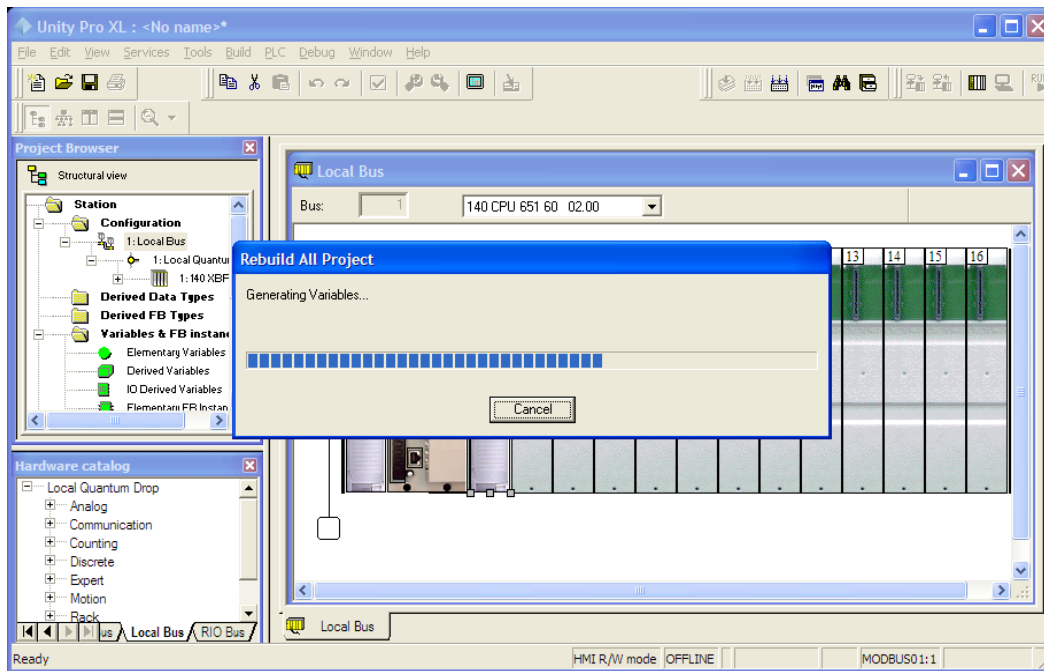
2.3 Building the Project

Whenever you update the configuration of your PTQ module 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 PC.
- 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. The following illustration shows the build process under way.



After the build process is completed successfully, the next step is to download the compiled project to the processor.

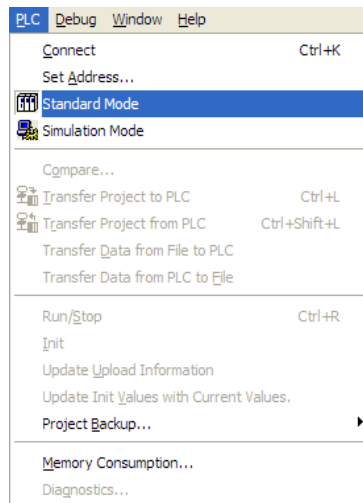
2.4 Connect Your PC to the Processor

The next step is to connect to the processor so that you can download the project file. The processor uses this project file to communicate over the backplane to modules identified in the project file.

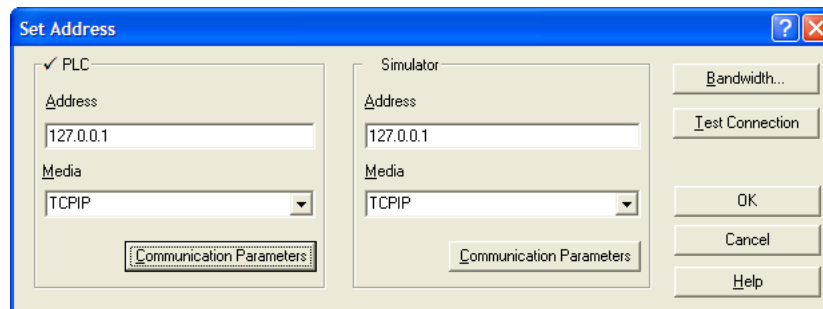
Note: If you have never connected from the PC to your processor before, you must verify that the necessary port drivers are installed and available to Unity Pro.

To verify address and driver settings in Unity Pro

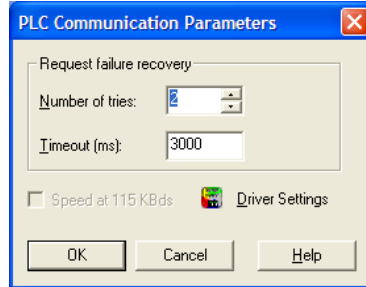
- 1 Open the **PLC** menu, and choose **STANDARD MODE**. This action turns off the PLC Simulator, and allows you to communicate directly with the Quantum or Unity hardware.



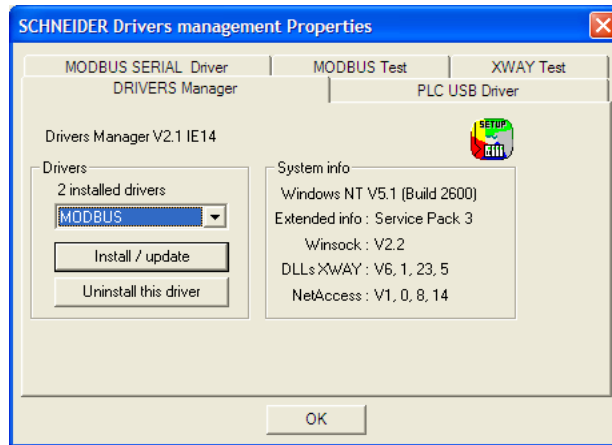
- 2 Open the **PLC** menu, and choose **SET ADDRESS...** This action opens the *Set Address* dialog box. Open the **MEDIA** dropdown list and choose the connection type to use (*TCPIP* or *USB*).



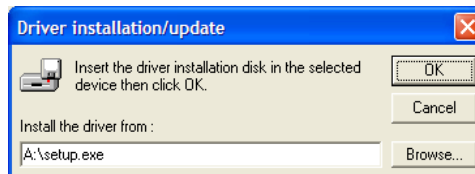
- If the **MEDIA** dropdown list does not contain the connection method you wish to use, click the **COMMUNICATION PARAMETERS** button in the PLC area of the dialog box. This action opens the *PLC Communication Parameters* dialog box.



- Click the **DRIVER SETTINGS** button to open the *SCHNEIDER Drivers management Properties* dialog box.



- Click the **INSTALL/UPDATE** button to specify the location of the Setup.exe file containing the drivers to use. You will need your Unity Pro installation disks for this step.



- Click the **BROWSE** button to locate the *Setup.exe* file to execute, and then execute the setup program. After the installation, restart your PC if you are prompted to do so. Refer to your Schneider Electric documentation for more information on installing drivers for Unity Pro.

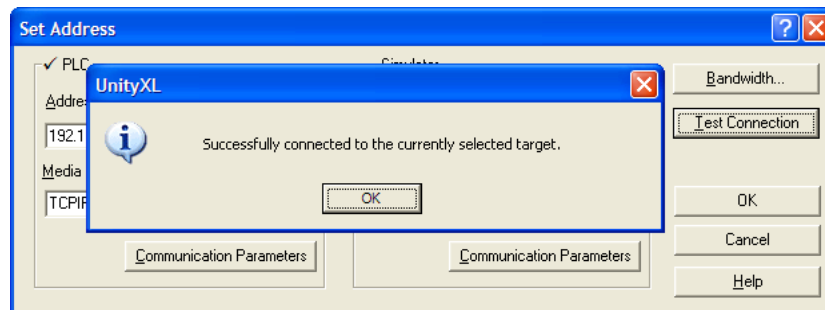
2.4.1 Connecting to the Processor with TCP/IP

The next step is to download (copy) the project file to the processor. The following steps demonstrate how to use an Ethernet cable connected from the Processor to your PC through an Ethernet hub or switch. Other connection methods may also be available, depending on the hardware configuration of your processor, and the communication drivers installed in Unity Pro.

- 1 If you have not already done so, connect your PC and the processor to an Ethernet hub.
- 2 Open the **PLC** menu, and then choose **SET ADDRESS**.

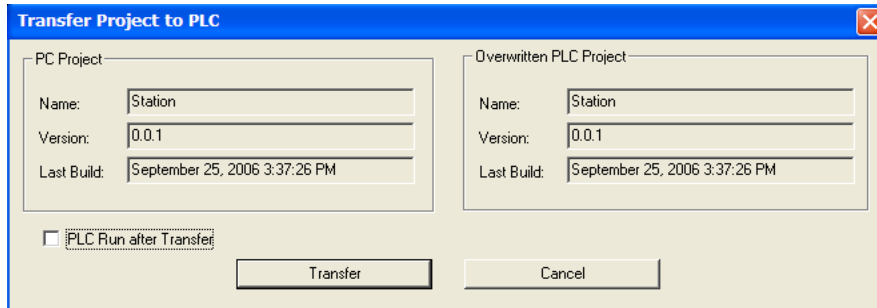
Important: Notice that the *Set Address* dialog box is divided into two areas. Enter the address and media type in the *PLC* area of the dialog box, not the *Simulator* area.

- 3 Enter the IP address in the address field. In the **MEDIA** dropdown list, choose **TCP/IP**.
- 4 Click the **TEST CONNECTION** button to verify that your settings are correct.



2.5 Downloading the Project to the 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.
- 4 When the transfer is complete, place the processor in *Run* mode.

3 Configuring the Processor with Concept

In This Chapter

- ❖ Information for Concept Version 2.6 Users..... 24
- ❖ Creating a New Project..... 26
- ❖ Adding the PTQ Module to the Project 29
- ❖ Setting up Data Memory in Project..... 32
- ❖ Downloading the Project to the Processor..... 35
- ❖ Verifying Successful Download 37

The following steps are designed to ensure that the processor is able to transfer data successfully with the PTQ module. As part of this procedure, you will use Concept configuration software from Schneider Electric to create a project, add the PTQ module to the project, set up data memory for the project, and then download the project to the processor.

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.

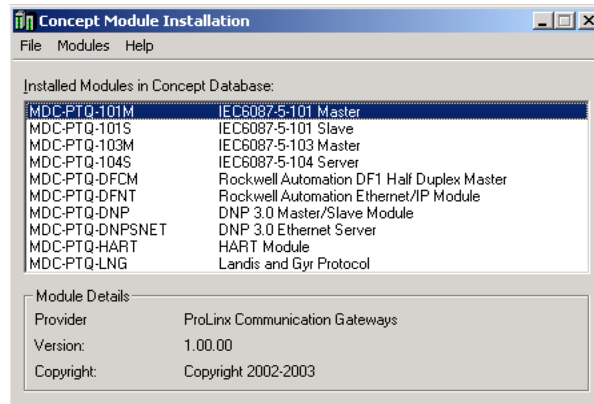
3.1 Information for Concept Version 2.6 Users

This guide uses Concept PLC Programming Software version 2.6 to configure the Quantum PLC. Although not required, these files should be installed before proceeding to the next section.

3.1.1 Installing MDC Configuration Files

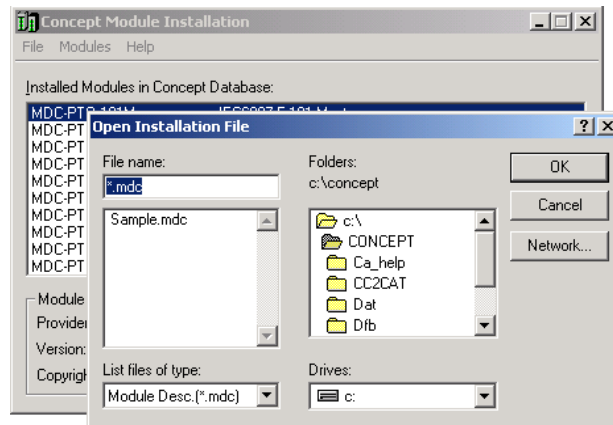
- 1 From a PC with Concept 2.6 installed, choose **START / PROGRAMS / CONCEPT / ModCONNECT TOOL**.

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



- 2 Choose **FILE / OPEN INSTALLATION FILE**.

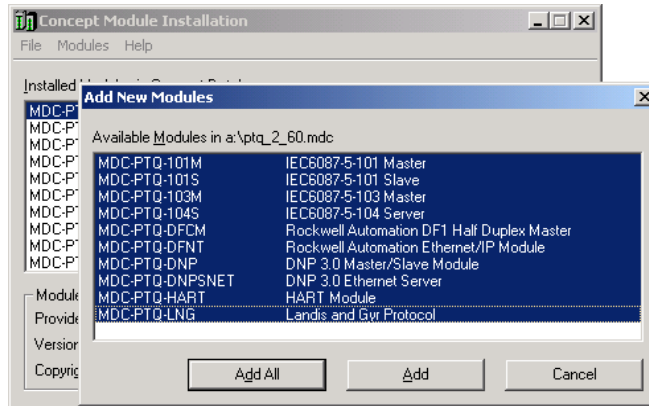
This action opens the *Open Installation File* dialog box:



- 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.

- 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.

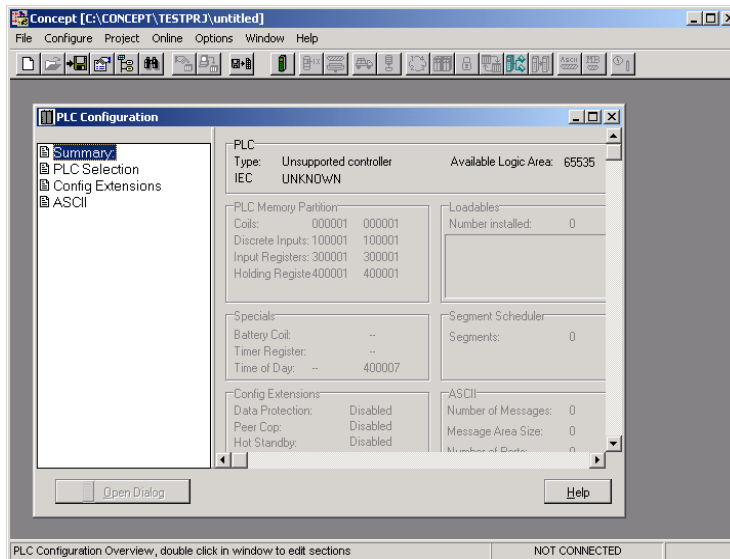


- 5 Click the **ADD ALL** button. A series of message boxes may appear during this process. Click **YES** or **OK** for each message that appears.
- 6 When the process is complete, open the **FILE** menu and choose **EXIT** to save your changes.

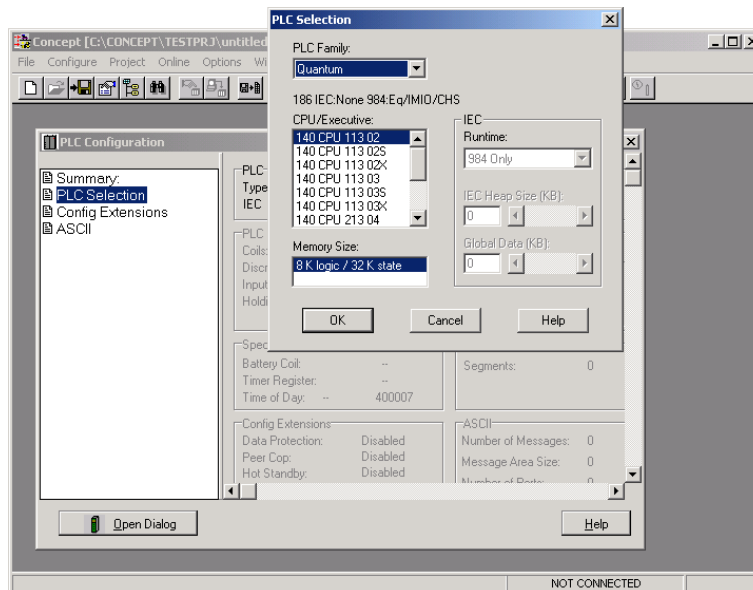
3.2 Creating a New Project

This phase of the setup procedure must be performed on a computer that has the Concept configuration software installed.

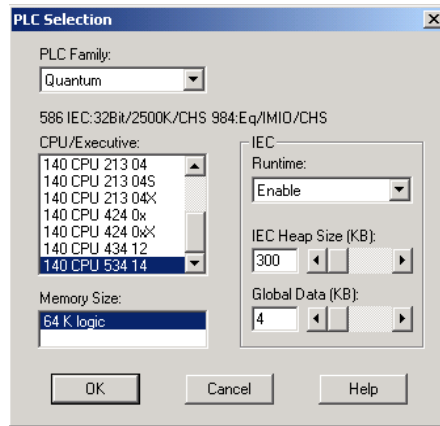
- 1 From your computer, choose **START / PROGRAMS / CONCEPT V2.6 XL.EN / CONCEPT**. This action opens the *Concept* window.
- 2 Open the File menu, and then choose **NEW PROJECT**. This action opens the *PLC Configuration* dialog box.



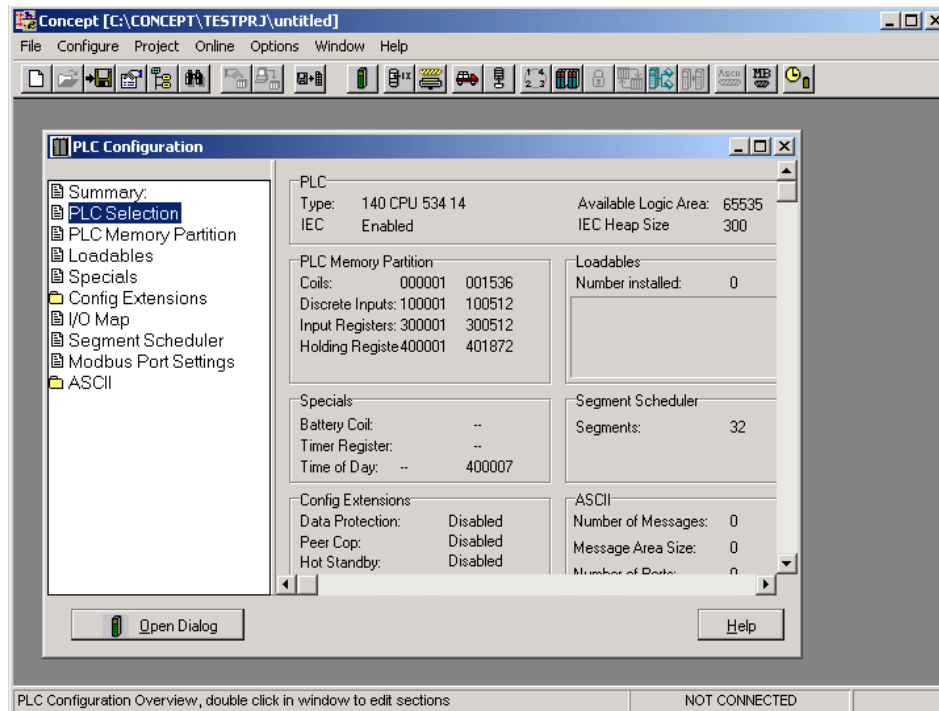
- 3 In the list of options on the left side of this dialog box, double-click the **PLC SELECTION** folder. This action opens the *PLC Selection* dialog box.



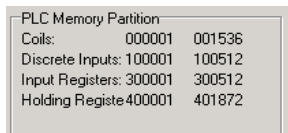
- 4 In the *CPU/Executive* pane, use the scroll bar to locate and select the **PLC** to configure.



- 5 Click **OK**. This action opens the *PLC Configuration* dialog box, populated with the correct values for the PLC you selected.



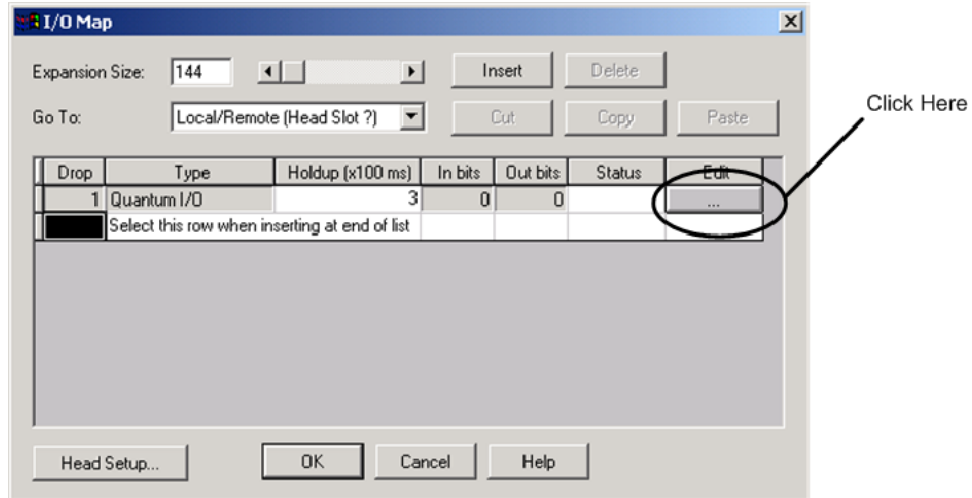
- 6 Make a note of the holding registers for the module. You will need this information when you modify your application. The Holding Registers are displayed in the *PLC Memory Partition* pane of the *PLC Configuration* dialog box.



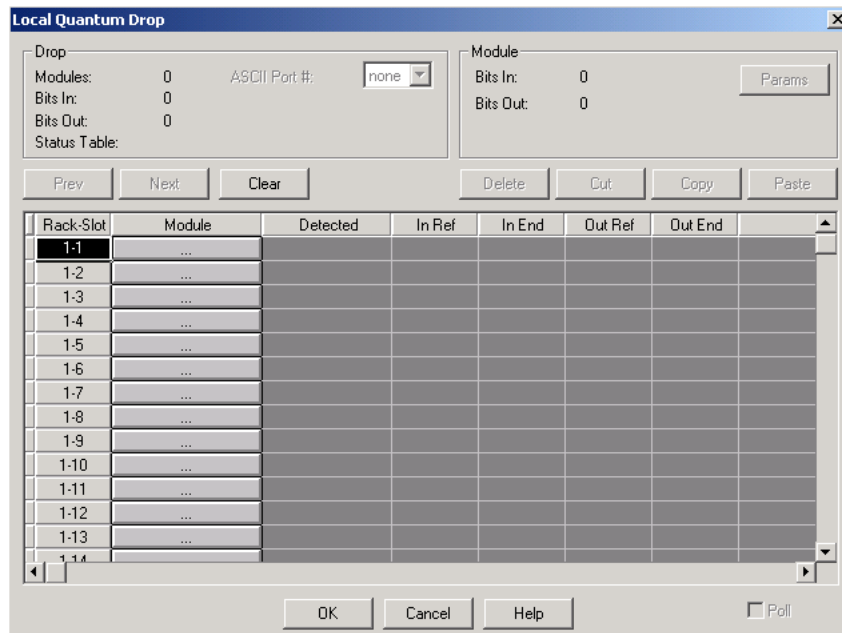
PLC Memory Partition	
Coils:	000001 001536
Discrete Inputs:	100001 100512
Input Registers:	300001 300512
Holding Registers:	400001 401872

3.3 Adding the PTQ Module to the Project

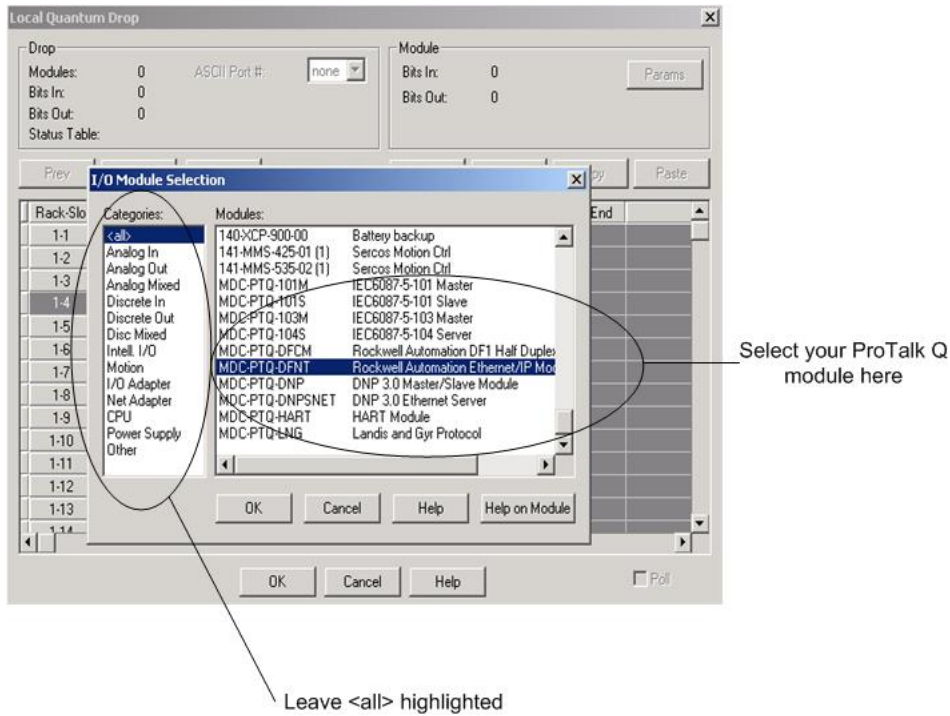
- 1 In the list of options on the left side of the *PLC Configuration* dialog box, double-click **I/O MAP**. This action opens the *I/O Map* dialog box.



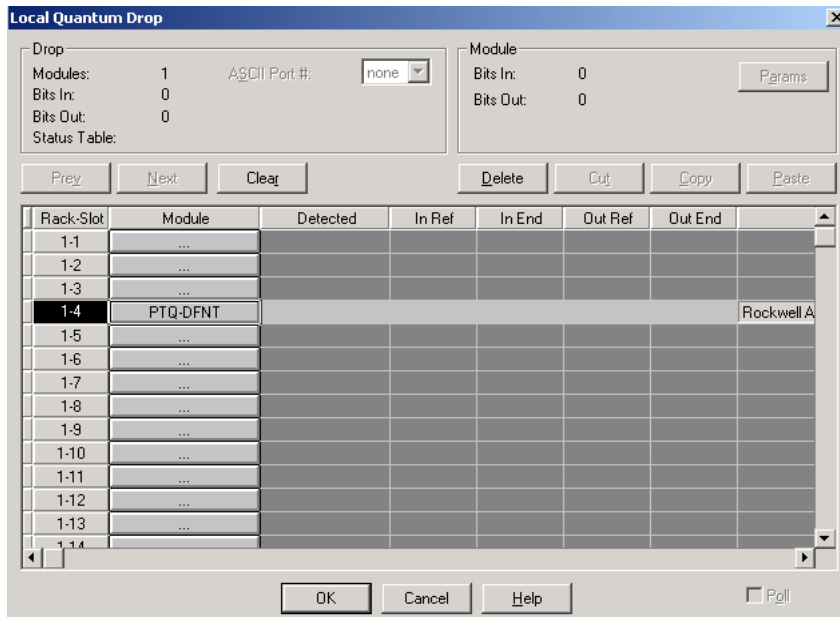
- 2 Click the **EDIT** button to open the *Local Quantum Drop* dialog box. This dialog box is where you identify rack and slot locations.



- Click the **MODULE** button next to the rack/slot position where the ProTalk module will be installed. This action opens the *I/O Module Selection* dialog box.

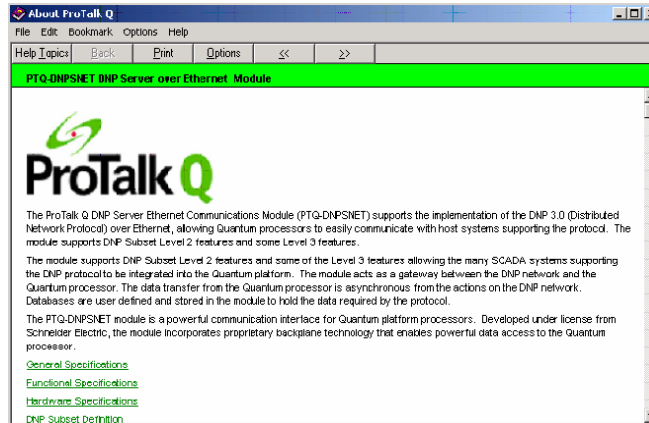


- In the *Modules* pane, use the scroll bar to locate and select the ProTalk module, and then click **OK**. This action copies the description of the ProTalk module next to the assigned rack and slot number of the *Local Quantum Drop* dialog box.



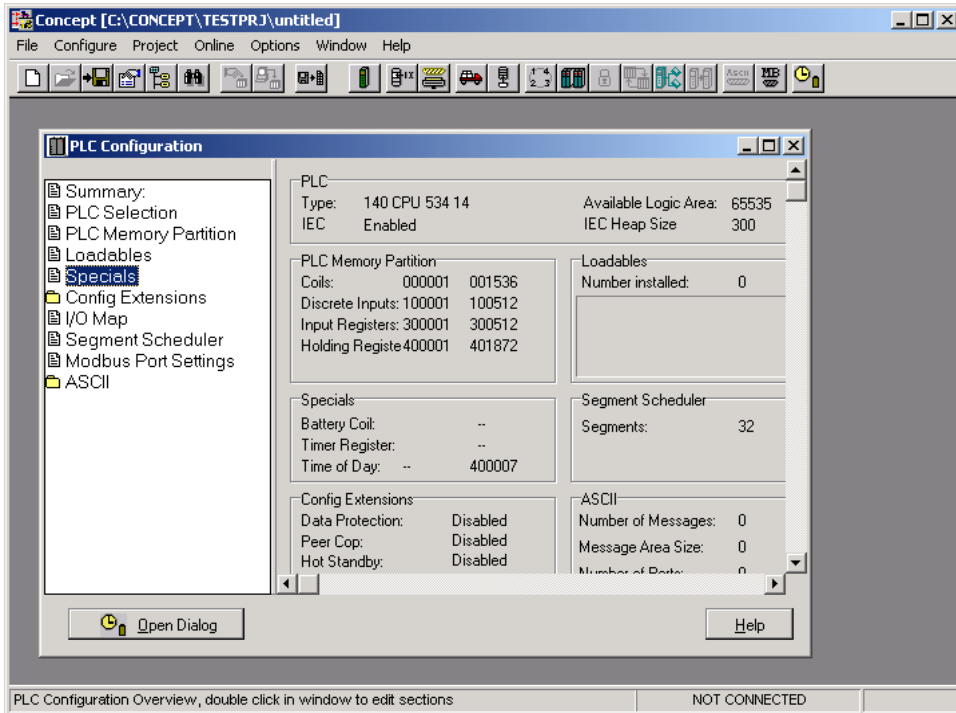
- Repeat steps 3 through 5 for each ProTalk module you plan to install. When you have finished installing your ProTalk modules, click **OK** to save your settings. Click **YES** to confirm your settings.

Tip: Select a module, and then click the Help on Module button for help pages.

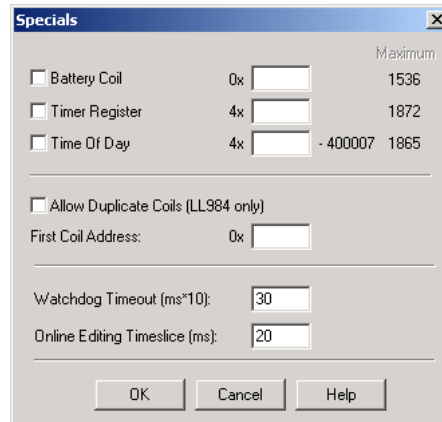


3.4 Setting up Data Memory in Project

- 1 In the list of options on the left side of the *PLC Configuration* dialog box, double-click **SPECIALS**.

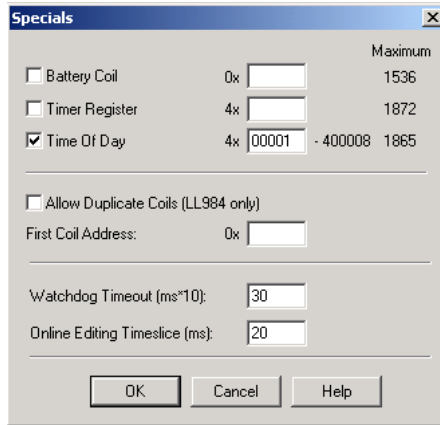


- 2 This action opens the *Specials* dialog box.



Selecting the Time of Day

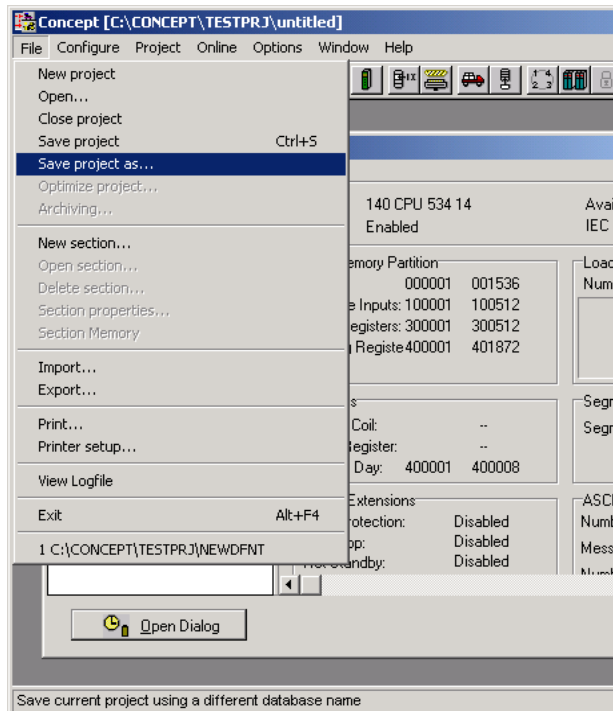
- 1 Select (check) the *Time of Day* box, and then enter the value 00001 as shown in the following illustration. This value sets the first time of day register to 400001.



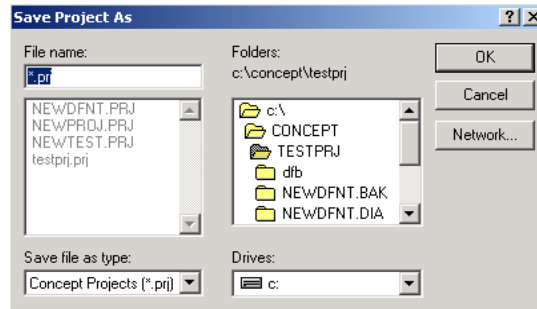
- 2 Click **OK** to save your settings and close the *Specials* dialog box.

Saving your project

- 1 In the *PLC Configuration* dialog box, choose **FILE / SAVE PROJECT AS.**



- 2 This action opens the *Save Project As* dialog box.

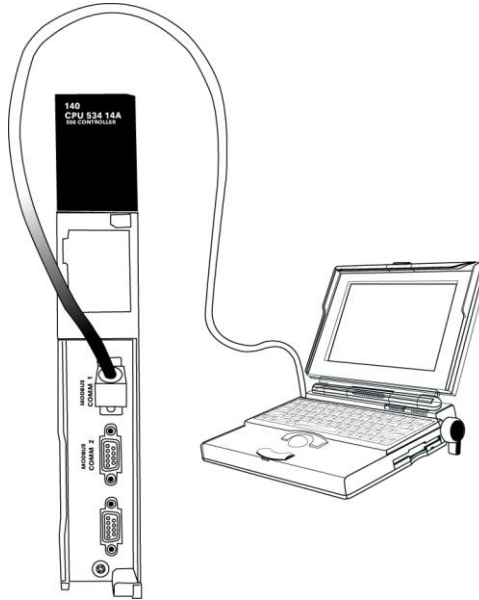


- 3 Name the project, and then click **OK** to save the project to a file.

3.5 Downloading the Project to the Processor

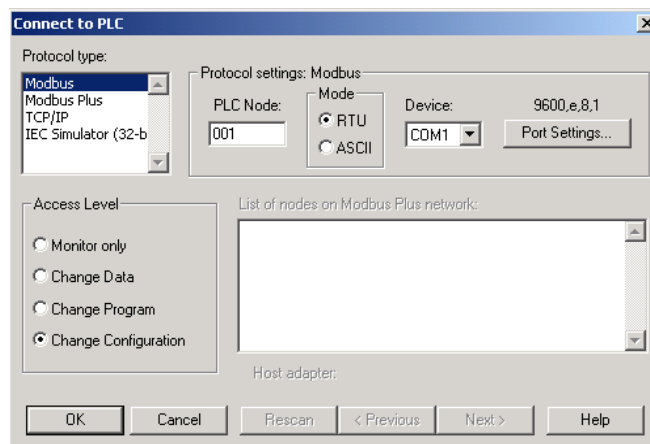
Next, download (copy) the project file to the Quantum Processor.

- 1 Use the null modem cable to connect your PC's serial port to the Quantum processor, as shown in the following illustration.



Note: You can use a Modbus Plus Network Option Module (NOM Module) module in place of the serial port if necessary.

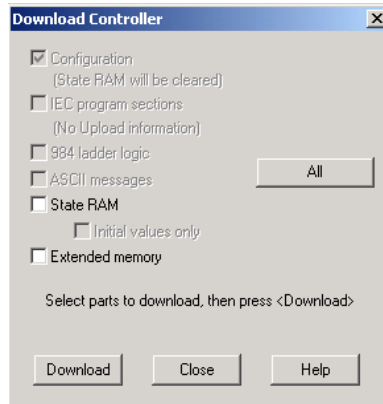
- 2 Open the **PLC** menu, and then choose **CONNECT**.
- 3 In the *PLC Configuration* dialog box, open the **ONLINE** menu, and then choose **CONNECT**. This action opens the *Connect to PLC* dialog box.



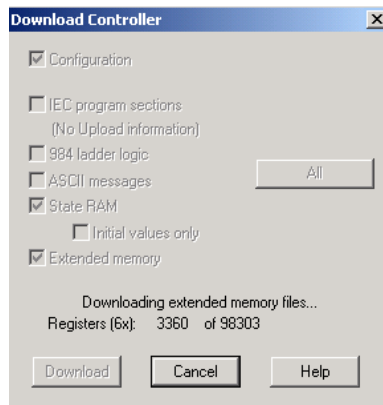
- 4 Leave the default settings as shown and click **OK**.

Note: Click **OK** to dismiss any message boxes that appear during the connection process.

- 5 In the *PLC Configuration* window, open the **ONLINE** menu, and then choose **DOWNLOAD**. This action opens the *Download Controller* dialog box.



- 6 Click **ALL**, and then click **DOWNLOAD**. If a message box appears indicating that the controller is running, click **YES** to shut down the controller. The *Download Controller* dialog box displays the status of the download as shown in the following illustration.

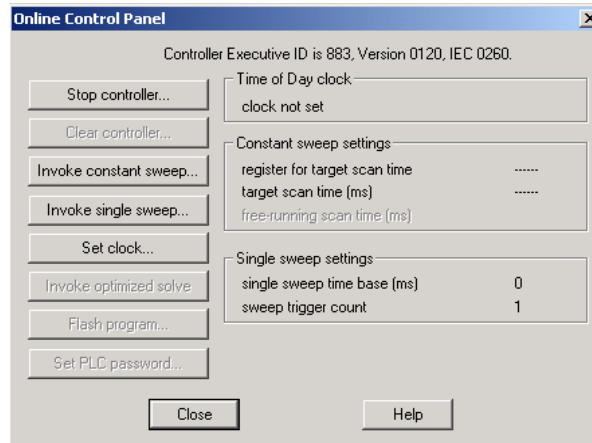


- 7 When the download is complete, you will be prompted to restart the controller. Click **YES** to restart the controller.

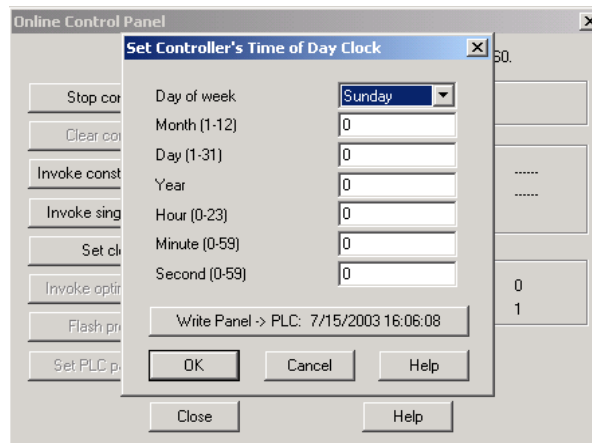
3.6 Verifying Successful Download

The final step is to verify that the configuration changes you made were received successfully by the module, and to make some adjustments to your settings.

- 1 In the *PLC Configuration* window, open the **ONLINE** menu, and then choose **ONLINE CONTROL PANEL**. This action opens the *Online Control Panel* dialog box.



- 2 Click the **SET CLOCK** button to open the *Set Controller's Time of Day Clock* dialog box.



- 3 Click the **WRITE PANEL** button. This action updates the date and time fields in this dialog box. Click **OK** to close this dialog box and return to the previous window.
- 4 Click **CLOSE** to close the *Online Control Panel* dialog box.
- 5 In the *PLC Configuration* window, open the **ONLINE** menu, and then choose **REFERENCE DATA EDITOR**. This action opens the *Reference Data Editor* dialog box. On this dialog box, you will add preset values to data registers that will later be monitored in the ProTalk module.

- Place the cursor over the first address field, as shown in the following illustration.

	Variable Name	Data Type	Address	Value	Set Value
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

- In the *PLC Configuration* window, open the **TEMPLATES** menu, and then choose **INSERT ADDRESSES**. This action opens the Insert addresses dialog box.
- On the *Insert Addresses* dialog box, enter the values shown in the following illustration, and then click **OK**.

Insert Addresses

First Reference To Insert: 400001

Last Reference To Insert: 400010

Number of References to Insert: 10

Display Format: Dec

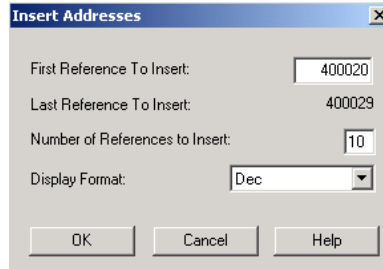
OK Cancel Help

- Notice that the template populates the address range, as shown in the following illustration. Place your cursor as shown in the first blank address field below the addresses you just entered.

Place cursor here

	Variable Name	Data Type	Address	Value	Set Value
2			400002		
3			400003		
4			400004		
5			400005		
6			400006		
7			400007		
8			400008		
9			400009		
10			400010		
11					
12					
13					

10 Repeat steps 6 through 9, using the values in the following illustration:



11 In the *PLC Configuration* window, open the **ONLINE** menu, and then choose **ANIMATE**. This action opens the *RDE Template* dialog box, with animated values in the *Value* field.

	Variable Name	Data Type	Address	Value	Set Value
3			400003	7	
4			400004	17	
5			400005	3	
6			400006	15	
7			400007	2	
8			400008	49	
9			400009	0	
10			400010	0	
11					
12			400020	24576	
13			400021	5	
14			400022	7	

- 12 Verify that values shown are cycling, starting from address 400065 and up.
- 13 In the *PLC Configuration* window, open the **TEMPLATES** menu, and then choose **SAVE TEMPLATE AS**. Name the template *ptqclock*, and then click **OK** to save the template.
- 14 In the *PLC Configuration* window, open the **ONLINE** menu, and then choose **DISCONNECT**. At the disconnect message, click **YES** to confirm your choice.

At this point, you have successfully

- Created and downloaded a Quantum project to the PLC
- Preset values in data registers that will later be monitored in the ProTalk module.

You are now ready to complete the installation and setup of the ProTalk module.

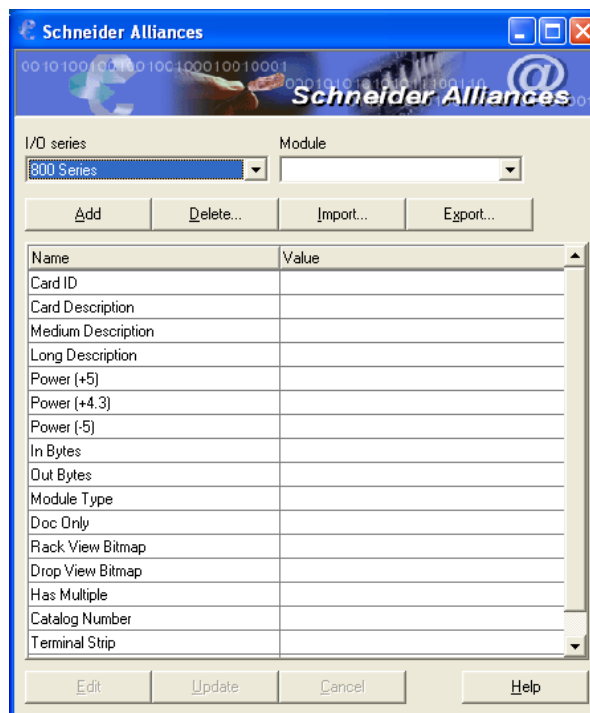
4 Configuring the Processor with ProWORX

Important Note: ProWORX 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.

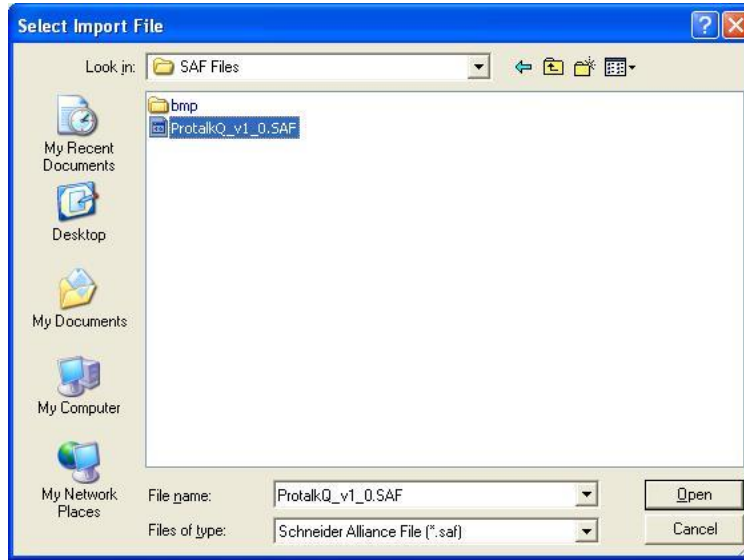
- 1 Run the **SCHNEIDER_ALLIANCES.EXE** application that is installed with the ProWORX 32 software:



- 2 Click on **IMPORT...**



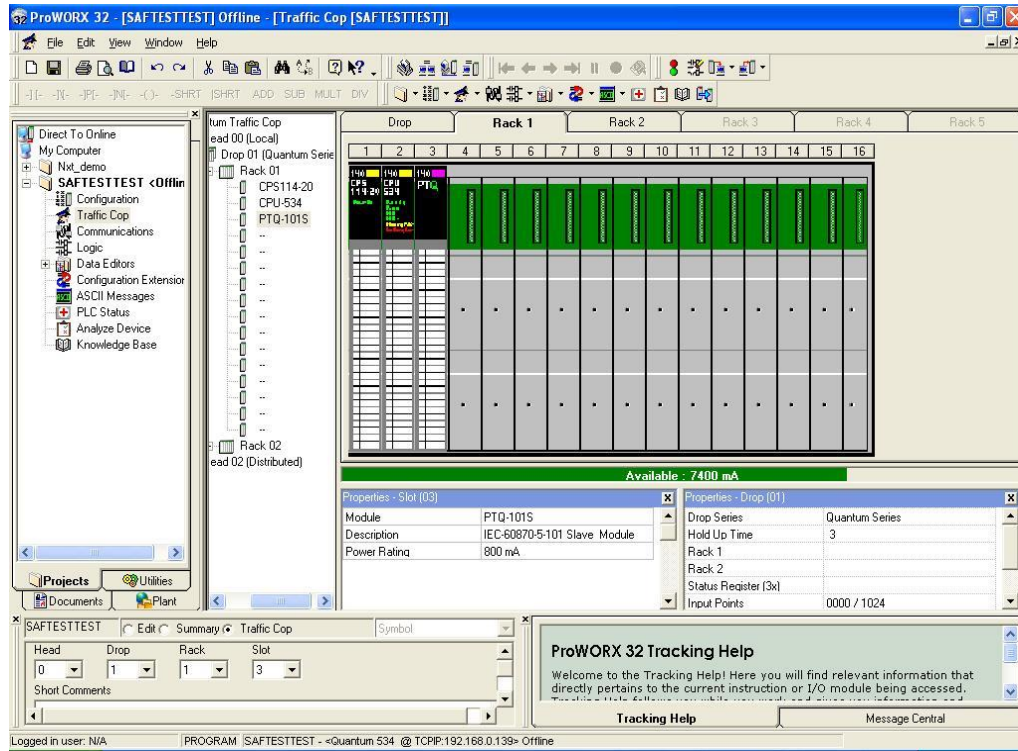
3 Select the .SAF File.



4 After you click on **OPEN** you should see the PTQ modules imported (select **I/O SERIES** as **QUANTUM**):



Now you can close the Schneider alliances application and run the ProWORX 32 software. At the *Traffic Cop* section, select the PTQ module to be inserted at the slot:



5 Setting Up the ProTalk Module

In This Chapter

- ❖ Installing the ProTalk Module in the Quantum Rack..... 46
- ❖ Connect the PC to the ProTalk Configuration/Debug Port..... 48

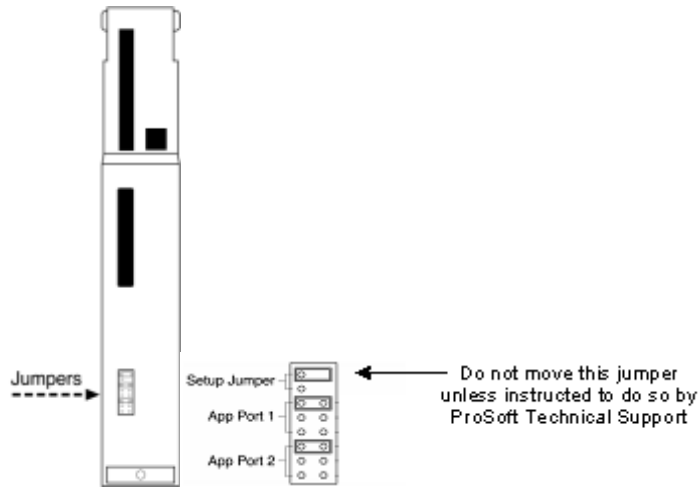
After you complete the following procedures, the ProTalk module will actively be transferring data bi-directionally with the processor.

5.1 Installing the ProTalk Module in the Quantum Rack

5.1.1 Verifying Jumper Settings

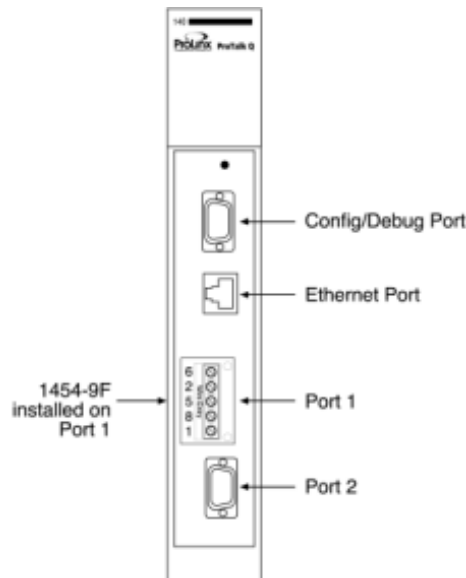
ProTalk modules are configured for RS-232 serial communications by default. To use RS-422 or RS-485, you must change the jumpers.

The jumpers are located on the back of the module as shown in the following illustration:



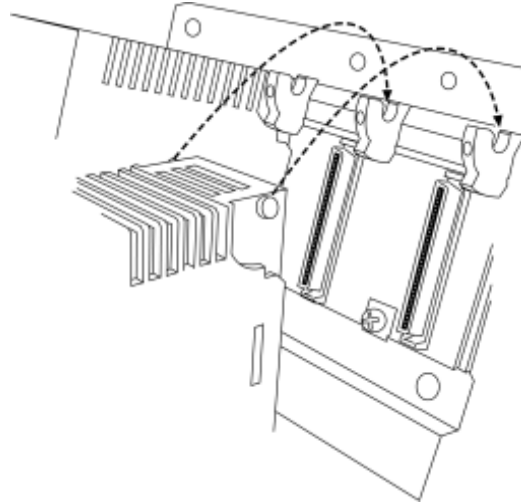
5.1.2 Inserting the 1454-9F connector

Insert the 1454-9F connector as shown. Wiring locations are shown in the table:

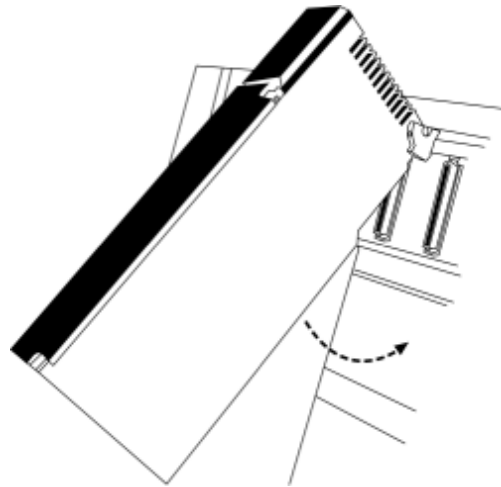


5.1.3 Installing the ProTalk Module in the Quantum Rack

- 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 slots on the backplane.



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

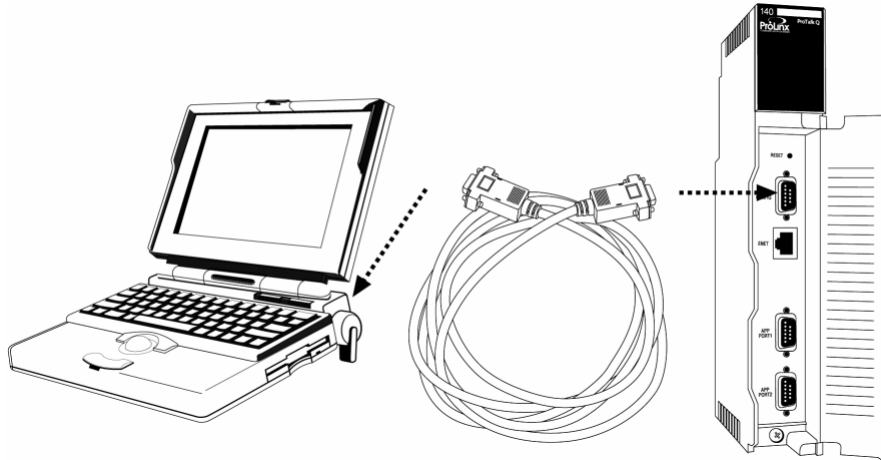


Caution: The PTQ module is hot-swappable, meaning that you can install and remove it while the rack is powered up. You should not assume that this is the case for all types of modules unless the user manual for the product explicitly states that the module is hot-swappable. Failure to observe this precaution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

5.2 Connect the PC to the ProTalk Configuration/Debug Port

Make sure you have exited the Quantum programming software before performing these steps. This action will avoid serial port conflict.

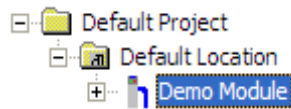
Using the supplied Null Modem cable, connect your PC to the Configuration/Debug port on the ProTalk module as shown



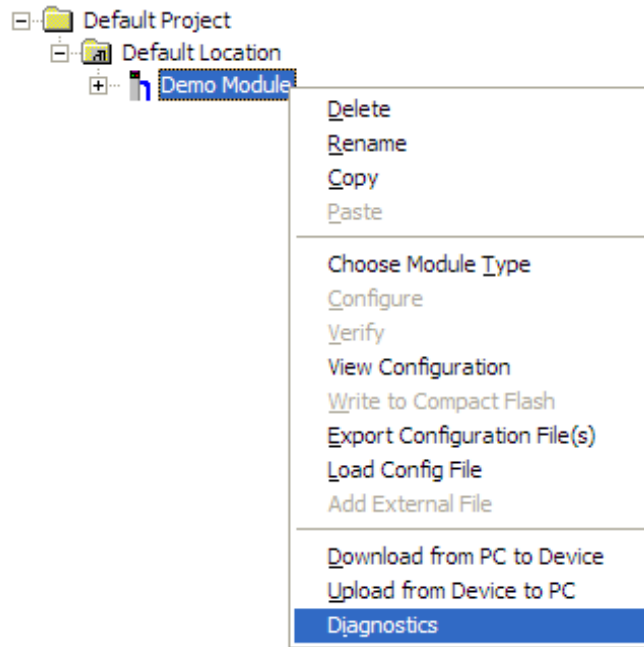
Tip: You can have a ProSoft Configuration Builder Diagnostics window open for more than one module at a time.

To connect to the module's Configuration/Debug serial port

- 1 Start *PCB*, and then select the module to test. Click the right mouse button to open a shortcut menu.

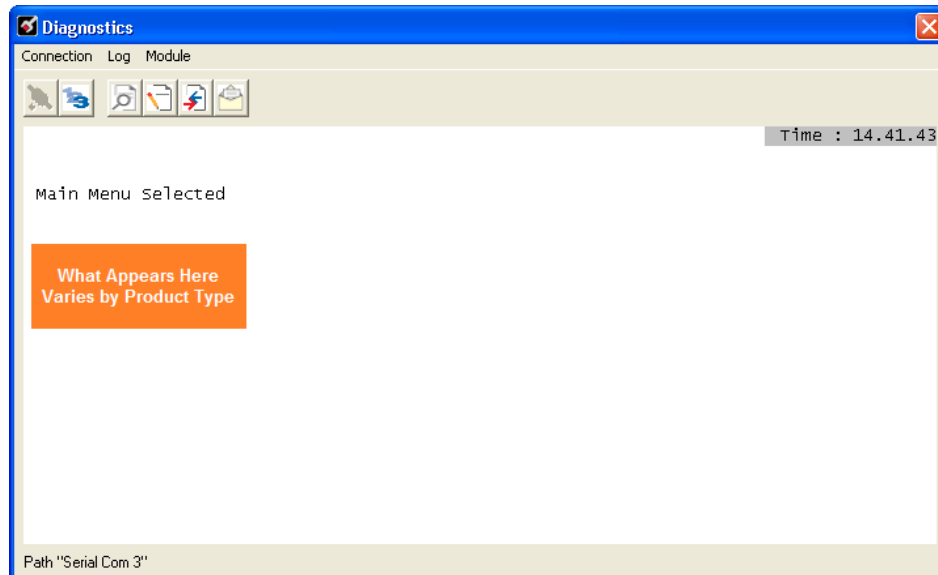


- 2 On the shortcut menu, choose **DIAGNOSTICS**.



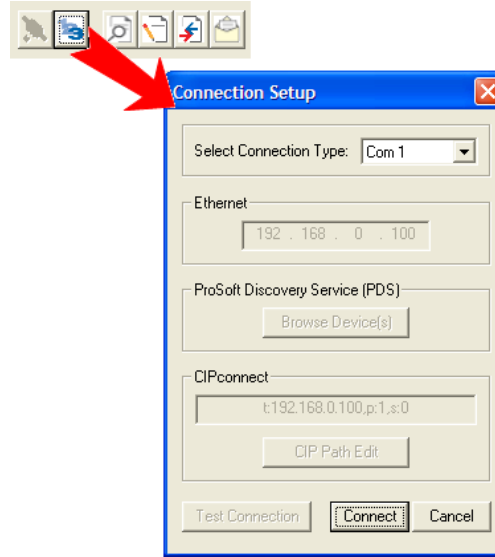
This action opens the *Diagnostics* dialog box.

- 3 Press [?] to open the *Main* menu.



If there is no response from the module, follow these steps:

- 1 Click to configure the connection. On the *Connection Setup* dialog box, select a valid com port or other connection type supported by the module.



- 2 Verify that the null modem cable is connected properly between your computer's serial port and the module. A regular serial cable will not work.
- 3 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.
- 4 If you are still not able to establish a connection, contact ProSoft Technology for assistance.

6 Configuring the Module

In This Chapter

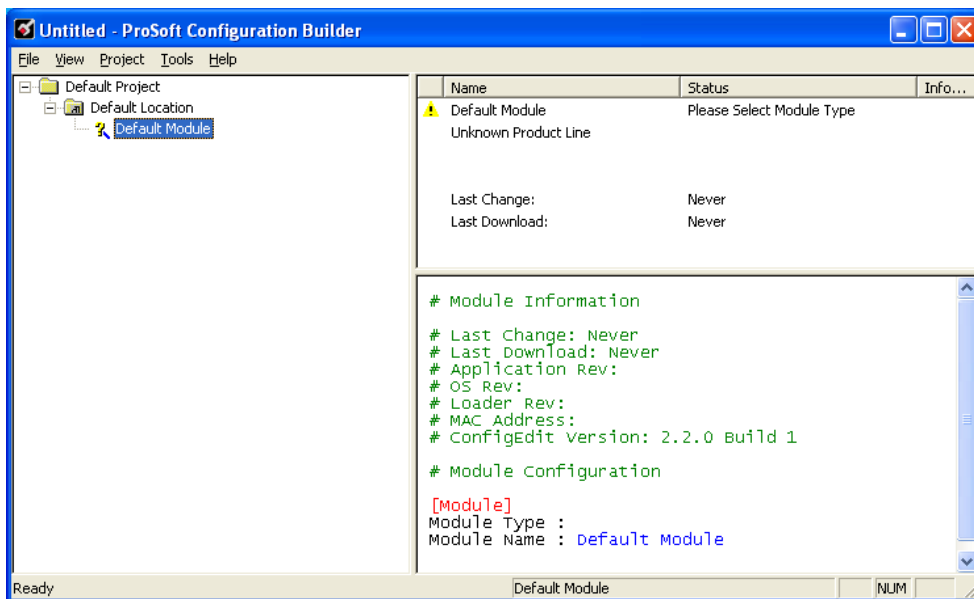
- ❖ Using ProSoft Configuration Builder 52
- ❖ Edit the Configuration File 55
- ❖ Downloading the Project to the Module Using a Serial COM port 69
- ❖ Verification and Troubleshooting 70

6.1 Using ProSoft Configuration Builder

ProSoft Configuration Builder (PCB) provides a quick and easy way to manage module configuration files customized to meet your application needs. *PCB* is not only a powerful solution for new configuration files, but also allows you to import information from previously installed (known working) configurations to new projects.

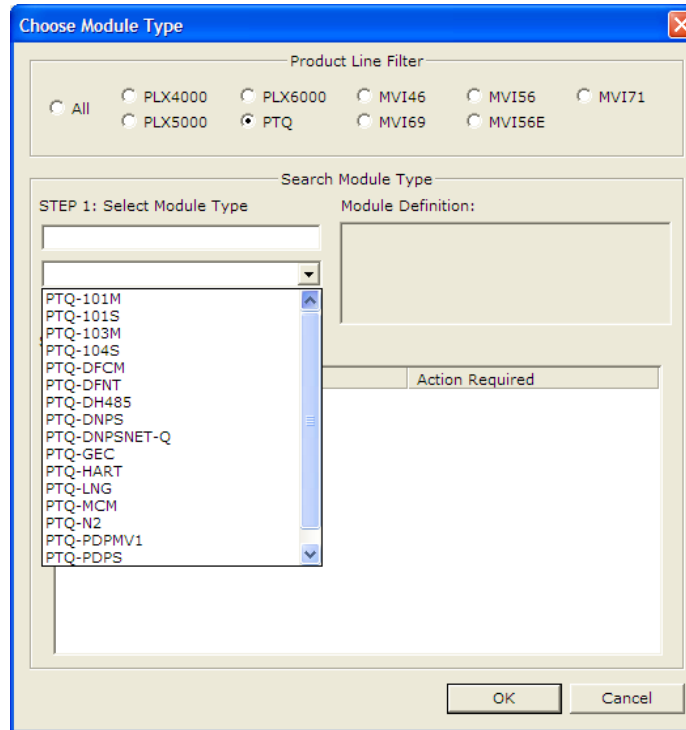
6.1.1 Set Up the Project

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, an information pane and a 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 following illustration shows the *ProSoft Configuration Builder* window with a new project.



Your first task is to add the PTQ-MCM 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, choose **CHOOSE MODULE TYPE**. This action opens the *Choose Module Type* dialog box.

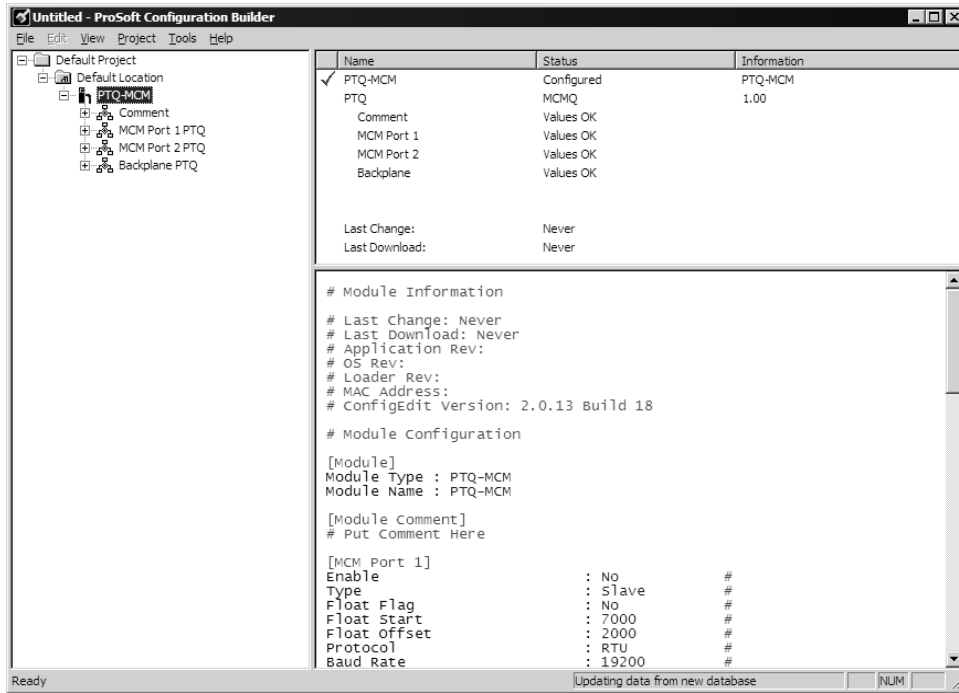


- 3 In the **PRODUCT LINE FILTER** area of the dialog box, select **PTQ**. In the **SELECT MODULE TYPE** dropdown list, select **PTQ-MCM**, and then click **OK** to save your settings and return to the *ProSoft Configuration Builder* window.

The next task is to set the module parameters.

6.1.2 Renaming PCB Objects



Notice that the contents of the information pane and the configuration pane changed when you added the module to the project.



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

- 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.

Configuring Module Parameters

- 1 Click on the **[+]** sign next to the module icon to expand module information.
- 2 Click on the **[+]** sign next to any  icon to view module information and configuration options.
- 3 Double-click any  icon to open an *Edit* dialog box.
- 4 To edit a parameter, select the parameter in the left pane and make your changes in the right pane.
- 5 Click **OK** to save your changes.

6.2 Edit the Configuration File

Note: It is important that you plan your configuration before modifying the configuration files. The remainder of this step provides the information to make the appropriate modifications to the configuration files.

Important: This module supports a maximum configuration file size of 128 kilobytes (131072 bytes). If the configuration file is larger than this size, the module will not accept the download. You can reduce the size of the configuration file by opening the file in a text editor and removing comment lines (lines preceded with the # character).

The PTQMCM.CFG file has the following main sections:

- [Module]
- [Backplane Configuration]
- [MCM Port 1]
- [MCM Port 1 Commands]
- [MCM Port 2]
- [MCM Port 2 Commands]

Important notes to consider when editing the sample configuration file:

- Comments within the file are preceded by the pound (#) sign. Any text on a line that occurs after the # character will be ignored.
- Do not use tabs or other non-printing characters instead of spaces to separate parameters (spacebar).
- Parameter names must begin in the first column of a line, and may not be preceded with a space (spacebar) or other non-printing character.

6.2.1 [Module]

```
# This section defines the configuration for the Module level
# data.
#
[Module]
Module Type           : PTQ-MCM
Module Name           : Test Example of PTQ-MCM Communication Module
```

Module Type Parameter

```
Module Type           : PTQ-MCM
```

The Module Type parameter is used to assign a name to the module that can be viewed using the configuration/debug port. Use this parameter to identify the module and the configuration file. You can enter a name from 0 to 80 characters.

Module Name

0 to 80 characters

This parameter assigns a name to the module that can be viewed using the configuration/debug port. Use this parameter to identify the module and the configuration file.

6.2.2 [Backplane Configuration]

```
[Backplane Configuration]
#These values are required to define the data area to transfer between the
module
#and the processor.
Read Register Start :      0 #Database start register to move to processor
Read Register Count :    100 #Number of words moved from module to processor
Write Register Start:    500 #Database start register where data placed from
                           #processor
Write Register Count:    100 #Number of words moved from processor to module

#Used to define the area in the Processor for the module to interface with
3x Register Start:      1 #3x start register where data moved from module to
                           #processor (1-n)
4x Register Start:      1 #4x start register where data moved from processor
                           #to module (1-n)

Initialize Output Data : No #Initialize the database file on startup
Failure Flag Count     : 0 #
Error/Status Block Pointer : 3000 #Number of register to store error and
                              #status
```

Read Register Start

0 to 3999

This parameter specifies the starting register address of a block of data registers to transfer from the module to the processor.

Read Register Count

0 to 4000

This parameter specifies the number of registers to transfer from the module to the processor.

Write Register Start

0 to 3999

This parameter specifies the starting register address of a module register block where data transferred from the processor will be stored.

Write Register Count

Range 0 to 4000

This parameter specifies the number of registers to transfer from the processor to the module. Valid entry for this parameter is 0 to 3999.

3x Register Start

1 to N

The 3x Register Start parameter defines the starting address in the processor's 3x (Quantum) or %iw (Unity) memory area to use for data being moved from the module. Take care to use a starting address that will accommodate the entire block from the module, but that will not overwrite data that is used for other purposes.

4x Register Start

1 to N

The 4x Register Start parameter defines the starting address in the processor's 4x (Quantum) or %iw (Unity) memory area to use for data being moved from the processor to the module. Take care to use a starting address that does not contain data in the processor's registers that is used for other purposes.

Initializing Output Data

YES or NO

This parameter determines if the output data for the module should be initialized with values from the processor. If the value is set to **No** (0), the output data will be initialized to 0. If the value is set to **YES** (1), the data will be initialized with data from the processor. Use of this option requires associated ladder logic to pass the data from the processor to the module.

Failure Flag Count

0 through 65535

This parameter specifies the number of successive transfer errors that must occur before halting communication on the application port(s). If the parameter is set to **0**, the application port(s) will continue to operate under all conditions. If the value is set larger than **0** (**1 to 65535**), communications will cease if the specified number of failures occur.

Error/Status Block Pointer

1 to 7000

The Error/Status Block Pointer parameter is used to specify the range of database registers to use for error and status data. The value should be no lower than 700, to avoid overwriting data, and no higher than 6970, to allow sufficient space for the error/status block.

6.2.3 [MCM Port X]

The information in this section applies to both Port 1 and Port 2.

```
# This section defines the port 1 configuration for the MCM device
#
[MCM Port 1]
Enable           :   Yes  #No=Port Disabled, Yes=Port Enabled
Type            :       5  #Type of port (0=Master, 1=Slave, 4=Direct PT,
                          5=Direct PT Swap)
Pass Thru Address :   300  #Address where pass-through block id is used
                          only
                          #Type equal to 2 or 3
Float Flag      :   No   #Float Flag (No=No Floating Point Data,
                          #Yes=Use Floating Point Data)
Float Start     :   4000 #The first register of floating-point data.
Float Offset    :   2000 #Internal DB offset to start of floating point
                          #data
Protocol        :       r  #ASCII or RTU
Baud Rate      :   192   #Baud rate for port (300, 600, 1200, 2400,
                          4800,
                          #9600, 19200, 38400, 57600, 115(for 115200))
Parity         :   None  #N=None, O=Odd, E=Even
Data Bits      :       8  #7 or 8 data bits for messages
Stop Bits      :       1  #1 or 2 stop bits for messages
RTS On        :       0  #Delay after RTS set before message sent (mSec)
RTS Off       :       0  #Delay after message before RTS dropped (mSec)
Minimum Response Delay : 0  #Number of mSec to delay before response
Use CTS Line   :   No   #Monitor CTS modem line (Y/N)
Slave Address  :       1  #0-255 Slave Node Address
Bit Input Offset : 0    #Internal DB offset to bit input data (Slave)
Word Input Offset : 0    #Internal DB offset to word input data (Slave)
Output Offset  : 0    #Internal DB offset to bit output data (Slave)
Hold Offset    : 0    #Internal DB offset to bit output data (Slave)
Command Count  :       3  #Command list count (Master)
Min Cmd Delay  :       1  #Minimum time between each command
                          #(Master 0-65535)
Cmd Err Pointer : 4500  #Internal DB location to place command error
                          #list (Master 0-4900)
Response Timeout : 0    #Response timeout for command (Master)
Retry Count    :       2  #Retry count for failed request (Master)
Error Delay Count : 0    #Command cycle count if error (Master)
```

Enable

This parameter defines if this Modbus port will be used. If the parameter is set to 0, the port is disabled. A value of 1 enables the port.

Type

This parameter specifies which device type the port will emulate. For specific information on types 2-5, refer to the Pass-Thru Functionality section of this manual.

Pass Thru Address

This parameter specifies the starting processor holding register address where the pass-thru block will be copied to. This parameter is only used if the type parameter is configured for Block-Pass-Thru operation (codes 2 and 3). The module adds 400001 to the actual value configured by the module. For example, a value of 200 would imply in a processor address of 400201

Float Flag

This flag specifies if the floating-point data access functionality is to be implemented. If the float flag is set to 1, Modbus functions 3, 6, and 16 will interpret floating-point values for registers as specified by the two following parameters.

Float Start

This parameter defines the first register of floating-point data. All requests with register values greater than or equal to this value will be considered floating-point data requests. This parameter is only used if the Float Flag is enabled.

Float Offset

This parameter defines the start register for floating-point data in the internal database. This parameter is only used if the Float Flag is enabled.

Protocol

RTU or ASCII

This parameter specifies the Modbus protocol to be used on the port. Valid protocols are: rtu = Modbus RTU and ascii = Modbus ASCII.

Baud Rate

This is the baud rate to be used on the port. Enter the baud rate as a value. For example, to select 19K baud, enter 19200. Valid entries are 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 576, and 115.

Parity

None, Odd, Even

Parity is a simple error checking algorithm used in serial communication. This parameter specifies the type of parity checking to use.

All devices communicating through this port must use the same parity setting.

Data Bits

7 or 8

This parameter sets the number of data bits for each word used by the protocol. All devices communicating through this port must use the same number of data bits.

Stop Bits

1 or 2

Stop bits signal the end of a character in the data stream. For most applications, use one stop bit. For slower devices that require more time to re-synchronize, use two stop bits.

All devices communicating through this port must use the same number of stop bits.

RTS On

0 to 65535 milliseconds

This parameter sets the number of milliseconds to delay after *Ready To Send* (RTS) is asserted before data will be transmitted.

RTS Off

0 to 65535 milliseconds

This parameter sets the number of milliseconds to delay after the last byte of data is sent before the RTS modem signal will be set low.

Minimum Response Delay

0 to 10000 milliseconds

This parameter sets the number of milliseconds to delay after the last byte of data is sent before the RTS modem signal will be set low. Valid values are in the range of 0 to 10000 milliseconds. When the port is configured for Modbus Slave operation, increase this parameter to allow the slave to send the message response to the port for the master.

Use CTS Line

YES or NO

This parameter specifies if the Clear To Send (CTS) modem control line is to be used or not. If the parameter is set to **No**, the CTS line will not be monitored. If the parameter is set to **YES**, the CTS line will be monitored and must be high before the module will send data. Normally, this parameter is required when half-duplex modems are used for communication (2-wire). This procedure is commonly referred to as *hardware handshaking*.

Slave Address

This parameter defines the virtual Modbus slave address for the internal database. All requests received by the port with this address are processed by the module. Verify that each device has a unique address on a network. Valid range for this parameter is 1 to 255 (247 on some networks). The address 0 is used for broadcast operations. When the slave port is configured without address 0 it will process all write commands received from the master but it will not send a message response.

Bit Input Offset

This parameter specifies the offset address in the internal Modbus database for network requests for Modbus Function 2 commands. For example, if the value is set to 150, an address request of 0 will return the value at register 150 in the database.

Word Input Offset

This parameter specifies the offset address in the internal Modbus database for network requests for Modbus Function 4 commands. For example, if the value is set to 150, an address request of 0 will return the value at register word 150 in the database.

Output Offset

This parameter specifies the offset address in the internal Modbus database for network requests for Modbus function 1, 5, or 15 commands. For example, if the value is set to 50, an address request of 0 will correspond to register word 50 (bit 800) in the database.

Hold Offset

0 to 4999

This parameter specifies the offset address in the internal Modbus database for network requests for Modbus function 3, 6, or 16 commands. For example, if a value of 50 is entered, a request for address 0 will correspond to the register 50 in the database.

Command Count

0 to 100

This parameter specifies the number of commands to be processed by the Modbus Master port.

Min Cmd Delay

0 to 65535

This parameter specifies the number of milliseconds to wait between issuing each command. This delay value is not applied to retries.

Cmd Err Pointer

-1 to 3995

This parameter sets the address in the internal Modbus database where the command error will be placed. If the value is set to -1, the data will not be transferred to the database. The valid range of values for this parameter is -1 to 4999. For example, if this parameter is configured for 1000, the command errors will be copied to the database as follows:

1000: error code for command 0

1001: error code for command 1

and so on.

An error code of 0 means that the command was successfully sent (no error). Refer to Error Status Table for the command error code listings.

Response Timeout

0 to 65535 milliseconds

This parameter sets the command response timeout period in 1 millisecond increments. This is the time that a port configured as a Master will wait for a response from the addressed slave before re-transmitting the command (Retries) or skipping to the next command in the Command List. The value to set depends on the communication network used and the expected response time (plus a little extra) of the slowest device on the network.

Retry Count

0 to 10

This parameter specifies the number of times a command will be retried if it fails.

Error Delay Counter

0 to 65535

This parameter specifies the number of polls to skip on the slave before trying to re-establish communications. After the slave fails to respond, the Master will skip commands to be sent to the slave the number of times entered in this parameter.

```
MODBUS MASTER/SLAVE COMMUNICATION MODULE (MVI56-MCMR) MENU
?=Display Menu
A=Data Analyzer
B=Block Transfer Statistics
C=Module Configuration
D=Modbus Database View
Master Command Errors : E=Port 1   F=Port 2
Master Command List   : I=Port 1   J=Port 2
Slave Status List     : O=Port 1   P=Port 2
R=Receive Configuration from Remote
S=Send Configuration to Remote
V=Version Information
W=Warm Boot Module
Communication Status : 1=Port 1   2=Port 2
Port Configuration   : 6=Port 1   7=Port 2

Esc=Exit Program

PORT 1 MODBUS STATUS:
Enabled : Y
Retries : 2      Cur Cmd : 0      State : 3
ComState: 0     Cur Err : 1     Last Err: 0

Number of Command Requests: 18
Number of Cmd Responses   : 0
Number of Command Errors  : 70
Number of Requests       : 71
Number of Responses       : 0
Number of Errors Received : 0
Number of Errors Sent     : 0

PORT 1 CONFIGURATION:
Enabled : Y      Port Type : (0) - MASTER
SLAVE SETUP:
Modbus Slave ID: 0      Pass-Through = DISABLED
Offsets:
  BitIn: 0      wordIn: 0      output: 0      Holding: 0
Floating-point Data:
  Flag: N      Start: 0      offset: 0
Route Count : 0
Function 99 Offset : 0
Use Packet Gap : N      Packet Gap Delay : 0
MASTER SETUP:
Command Count : 2      Cmd Delay: 0      Cmd offset : 0
Response Timeout: 1000 Retries : 3      Delay Count: 0
COMMUNICATION PARAMETERS:
Protocol: 0 (Modbus RTU)
Baud: 4800      Parity: NONE      Databits: 8      Stopbits: 1
RTS On: 0      RTS Off: 0      Use CTS Line: N
```

6.2.4 [Modbus Port 1 Commands]

Configure the following parameters for each command:

Parameter	Description										
Enable	<p>This parameter defines how the command will be sent. The following codes are valid:</p> <p>0: Disabled Disables the command. This command could still be activated from the processor using the Command Control special block (requires backplane command code 3)</p> <p>1: Continuous The command will be issued with a frequency determined by the poll interval parameter that defines minimum delays between consecutive commands.</p> <p>2: Conditional The conditional command is sent when the source data changes. It only applies to Modbus write commands (Modbus functions 5,6, 15 and 16)</p>										
Internal Address	<p>Configure the PTQ-MCM database address associated with the Modbus command.</p> <p>For a read command: this is the database address where the data read from the slave device will be copied to.</p> <p>For a write command: this is the database address where the data written to the slave device will be copied from.</p> <p>For bit commands (for example: 1, 2, 5 and 15) the module expects this value to represent the bit-address in the database. For example, a value of 100 would actually configure the Internal Address as BIT 100.</p> <p>For word commands (for example: 3, 4, 6 and 16) the module expects this value to represent the word-address in the database. For example, a value of 100 would actually configure the Internal Address as WORD 100 (1 word = 2 bytes).</p>										
Poll Interval	The minimum delay to be applied between consecutive polls for this command. Enter this number as milliseconds.										
Reg Count	Register count. Enter the number of bits or words for this command. The maximum value for word commands (3, 4, 6 and 16) is 125 words. The maximum value for word commands (1, 2, 5 and 15) is 800 bits.										
Swap Code	<p>This value configures if the data bytes and words should be swapped. The possible values are:</p> <table border="1"> <thead> <tr> <th>Swap Code</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>None - No Change is made in the byte ordering (1234 = 1234)</td> </tr> <tr> <td>1</td> <td>Words - The words are swapped (1234=3412)</td> </tr> <tr> <td>2</td> <td>Words & Bytes - The words are swapped then the bytes in each word are swapped (1234=4321)</td> </tr> <tr> <td>3</td> <td>Bytes - The bytes in each word are swapped (1234=2143)</td> </tr> </tbody> </table>	Swap Code	Description	0	None - No Change is made in the byte ordering (1234 = 1234)	1	Words - The words are swapped (1234=3412)	2	Words & Bytes - The words are swapped then the bytes in each word are swapped (1234=4321)	3	Bytes - The bytes in each word are swapped (1234=2143)
Swap Code	Description										
0	None - No Change is made in the byte ordering (1234 = 1234)										
1	Words - The words are swapped (1234=3412)										
2	Words & Bytes - The words are swapped then the bytes in each word are swapped (1234=4321)										
3	Bytes - The bytes in each word are swapped (1234=2143)										
Node Address	The slave address to which the command will be sent. This parameter can be assigned numbers from 2 to 255.										

Parameter	Description
Modbus Function	The following Modbus functions are supported: 1- Read Coil Status (bit) 2- Read Input Status (bit) 3- Read Holding Registers (word - 16 bits) 4- Read Input Registers (word - 16 bits) 5- Force (Write) Single Coil (bit) 6- Preset (Write) Single Register (word - 16 bits) 15- Force (Write) Multiple Coils (bit) 16- Preset (Write) Multiple Registers (word - 16 bits)
MB Address in Device	This parameter defines the starting address in the device being considered by the command. Values entered in this field are dependent on the node's database definition. Refer to the specific manufacturer's database definition for the device to determine the location of the data to be interfaced.

ENRON Floating Point Support

Many manufacturers have implemented special support in their drivers to support what is commonly called the Enron version of the MODBUS protocol. In this implementation, register addresses > 7000 are presumed to be floating point values. The significance to this is that the count field now becomes a 'number of values' field. In floating point format, each value represents two words.

Configuring the Floating Point Data Transfer

A common question when using the module as a Modbus Master is how floating point data is handled. This really depends on the slave device and how it addresses this application.

Just because your application is reading/writing floating point data, does not mean that you must configure the Float Flag, Float Start, and Float Offset parameters within the module.

These parameters are only used to support what is typically referred to as Enron or Daniel Modbus, where one register address must have 32 bits, or one floating point value. Below is an example:

Example #1

Modbus Address	Data Type	Parameter
47101	32 bit REAL	TEMP Pump #1
47102	32 bit REAL	Pressure Pump #1
47103	32 bit REAL	TEMP Pump #2
47104	32 bit REAL	Pressure Pump #2

With the module configured as a Master, you only need to enable these parameters to support a write to this type of addressing (Modbus FC 6 or 16).

If the slave device shows addressing as shown in Example #2, then you need not do anything with the Float Flag, Float Start parameters, as they use two Modbus addresses to represent one floating point value:

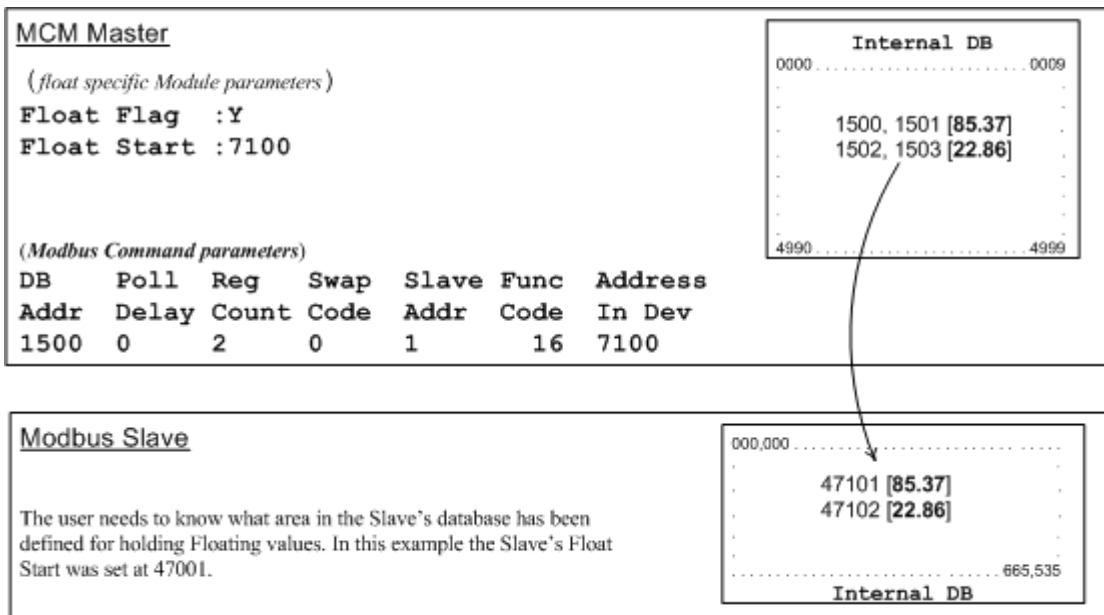
Example #2

Modbus Address	Data Type	Parameter
47101	32 bit REAL	TEMP Pump #1
47103	32 bit REAL	Pressure Pump #1
47105	32 bit REAL	TEMP Pump #2
47107	32 bit REAL	Pressure Pump #2

Because each 32 bit REAL value is represented by two Modbus Addresses (example 47101 and 47102 represent TEMP Pump #1), then you need not set the Float Flag, or Float Start for the module for Modbus FC 6 or 16 commands being written to the slave.

The next few pages show three specific examples:

Specific Example #1: Master is issuing Modbus command with FC 16 (with Float Flag: Yes) to transfer Float data to Slave.



(Float specific module parameters)

Float Flag: "Y" tells the Master to consider the data values that need to be sent to the Slave as floating point data where each data value is composed of 2 words (4 bytes or 32 bits).

Float Start: Tells the Master that if this address number is <= the address number in "Addr in Dev" parameter to double the byte count quantity to be included in the Command FC6 or FC16 to be issued to the Slave. Otherwise the Master will ignore the "Float Flag: Y" and treat data as composed of 1 word, 2 bytes.

(Modbus Command parameters)

DB Addr - Tells the Master where in its data memory is the beginning of data to obtain and write out to the Slave (slave) device.

Reg Count - Tells the Master how many data points to send to the Slave. Two counts will mean two floating points with Float Flag: Y and the "Addr in Dev" => the "Float Start" Parameter.

Swap Code - Tells the Master how to orient the Byte and Word structure of the data value. This is device dependent. Check Command Entry formats Section.

Func Code - Tells the Master to write the float values to the Slave. FC16.

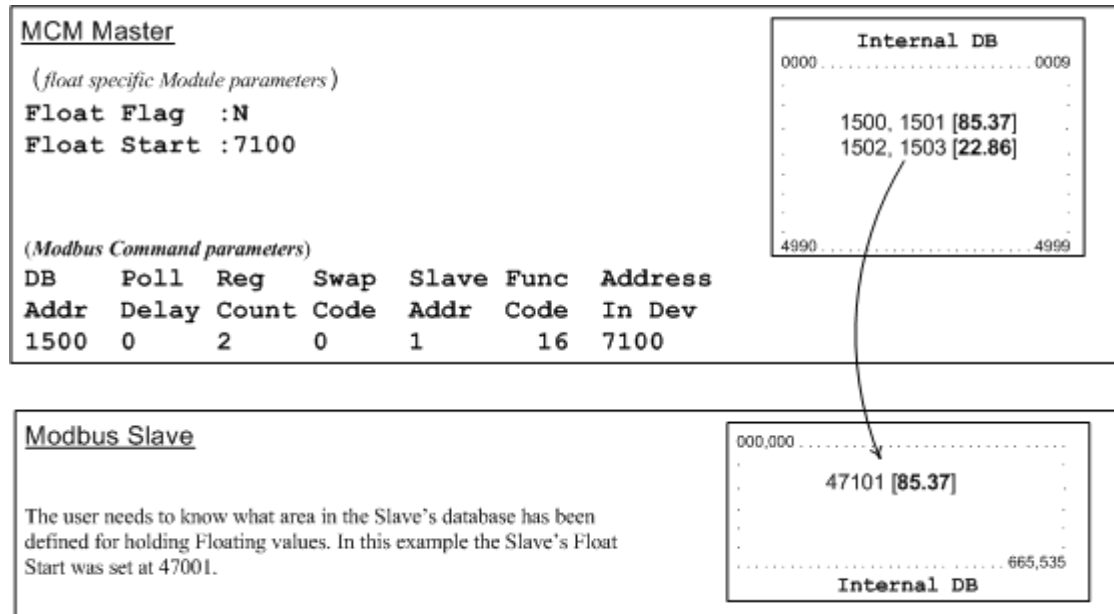
Addr in Dev - Tells the Master where in the Slave's database to locate the data.

In the above example, the Master's Modbus command to transmit inside the Modbus packet will be as follows.

	Slave address	Function Code	Address in Device	Reg count	Byte Count	Data
DEC	01	16	7100	2	8	85.37 22.86
HEX	01	10	1B BC	00 02	08	BD 71 42 AA E1 48 41 B6

In this example, the Master's Modbus packet contains the data byte and data word counts that have been doubled from the amount specified by Reg Count due to the Float flag set to Y. Some Slaves look for the byte count in the data packet to know the length of the data to read from the wire. Other slaves know at which byte the data begins and read from the wire the remaining bytes in the packet as the data the Master is sending.

Specific Example#2: Master is issuing Modbus command with FC 16 (with Float Flag: No) to transfer Float data.



Float Flag: "N" tells the Master to ignore the floating values and treat each register data as a data point composed of 1 word, 2 bytes or 16 bits.

Float Start: Ignored.

DB Addr - same as when Float Flag: Y.

Reg Count - Tells the Master how many data points to send to the Slave.

Swap Code - same as when Float Flag: Y.

Func Code - same as when Float Flag: Y.

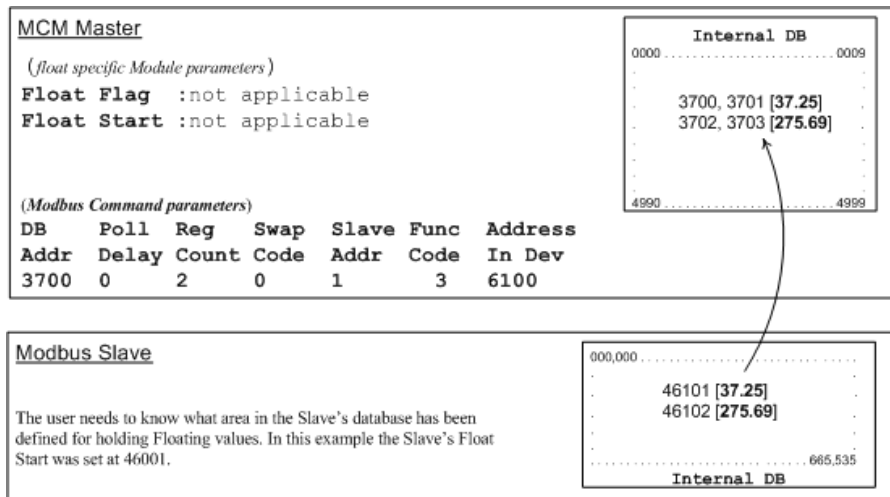
Addr in Dev - same as when Float Flag: Y as long as the Slave's Float Flag = Y.

In the above example, the Master's Modbus command to transmit inside the Modbus packet will be as follows.

	Slave address	Function Code	Address in Device	Reg Count	Byte Count	Data
DEC	01	16	7100	2	4	85.37
HEX	01	10	1B BC	00 02	04	BD 71 42 AA

In this example, the Master's Modbus packet contains the data byte and data word counts that have NOT been doubled from the amount specified by Reg Count due to the Float Flag set to N. The Slave looks for the byte count in the data packet to know the length of the data to read from the wire. Because of insufficient byte count, some slaves will read only half the data from the Master's transmission. Other slaves will read all 8 bytes in this example because they will know where in the packet the data starts and ignore the byte count parameter inside the Modbus packet.

Specific Example#3: Master is issuing Modbus command with FC 3 to transfer Float data from Slave.



Float Flag: Not applicable with Modbus Function Code 3.

Float Start: Not applicable with Modbus Function Code 3.

DB Addr - Tells the Master where in its data memory to store the data obtained from the Slave.

Reg Count - Tells the Master how many registers to request from the Slave.

Swap Code - same as above.

Func Code - Tells the Master to read the register values from the Slave. FC3.

Addr in Dev - Tells the Master where in the Slave's database to obtain the data.

In the above example, the Master's Modbus command to transmit inside the Modbus packet will be as follows.

	Slave address	Function Code	Address in Device	Reg count
DEC	01	3	6100	2
HEX	01	03	17 D4	00 02

In the above example the (Enron/Daniel supporting) Slave's Modbus command to transmit inside the Modbus packet will be as follows.

	Slave address	Function Code	Byte Count	Data
DEC	01	3	8	32.75 275.69
HEX	01	03	08	00 00 42 03 D8 52 43 89

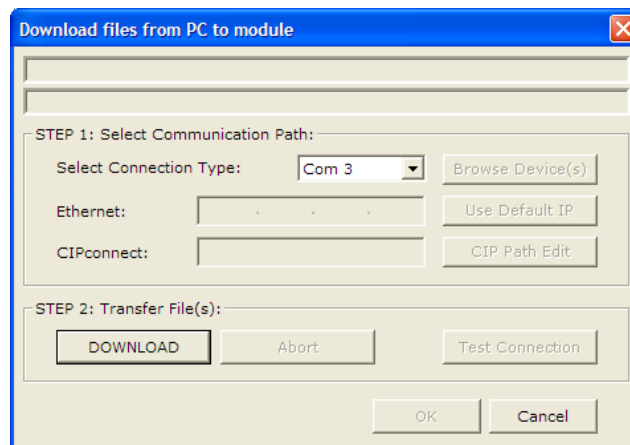
In the above example the (a NON-Enron/Daniel supporting) Slave's Modbus command that will be transmitted inside the Modbus packet will be as follows.

	Slave address	Function Code	Byte Count	Data
DEC	01	3	4	32.75
HEX	01	03	04	00 00 42 03

6.3 Downloading the Project to the Module Using a Serial COM port

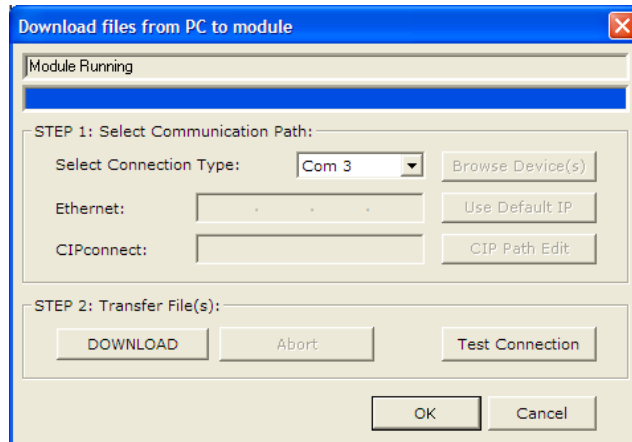
For the module to use the settings you configured, you must download (copy) the updated *Project* file from your PC to the module.

- 1 In the tree view in *ProSoft Configuration Builder*, click once to select the module.
- 2 Open the *Project* menu, and then choose **MODULE/DOWNLOAD**. The program will scan your PC for a valid com port (this may take a few seconds). When *PCB* has found a valid COM port, the *Download* dialog box will open.



- 3 Choose the COM port to use from the dropdown list, and then click the **DOWNLOAD** button.

The module will perform a platform check to read and load its new settings. When the platform check is complete, the status bar in the *Download* dialog box will display the message *Module Running*.



6.4 Verification and Troubleshooting

You can now verify that the module is configured properly by viewing parameters that you specified in the configuration file. This is done using the module's Main Menu. If you are not already at the Main menu, press **[SHIFT][/] OR [?]**.

Use the database menu to verify that data appears in registers that you've mapped. Refer to *Diagnostics and Troubleshooting* (page 71) for information on accessing information on the operation of the module.

7 Diagnostics and Troubleshooting

In This Chapter

- ❖ LED Status Indicators..... 72
- ❖ Using ProSoft Configuration Builder (PCB) for Diagnostics..... 73
- ❖ Reading Status Data from the Module 86

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.

7.1 LED Status Indicators

The LEDs will indicate the module’s operating status as follows:

LED	Color	Status	Indication
DEBUG	Green	On	Data is being transferred between the module and a remote terminal using the Configuration/Debug port.
		Off	No data is being transferred on the Configuration/Debug port.
PRT1	Green	On	Port is communicating.
		Off	Port is not communicating.
PRT2	Green	On	Port is communicating.
		Off	Port is not communicating.
ERR1 CFG Fail	Red	Off	The PTQ-MCM is working normally.
		On	The PTQ-MCM module program has recognized an application error. This LED will also be turned on if any command presents an error.
ERR2 P1 Fail	Red	On	Time-out. The post sent a Modbus command but did not receive a response from the slave.
		Off	Normal Operation
ERR3 P2 Fail	Red	On	Time-out. The post sent a Modbus command but did not receive a response from the slave.
		Off	Normal Operation
Active	Green	On	The LED is on when the module recognizes a processor and is able to communicate if the [Backplane Data Exchange] section specifies data transfer commands.
		Off	The LED is off when the module is unable to speak with the processor. The processor either absent or not running.
BAT Low	Red	Off	The battery voltage is OK and functioning.
		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 been un-powered 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	N/A	Not Used
		N/A	Not Used
E-Data	Green	N/A	Not Used
		N/A	Not Used

If your module is not operating, and the status LEDs are not illustrated in the table above, please call ProSoft Technology for technical assistance.

7.2 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 submenus for each menu command. The first menu you see when you connect to the module is the *Main* menu.

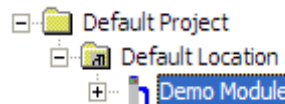
Because this is a text-based menu system, you enter commands by typing the [command letter] from your computer keyboard in the *Diagnostic* window in *ProSoft Configuration Builder (PCB)*. The module does not respond to mouse movements or clicks. The command executes as soon as you press the [COMMAND LETTER] — you do not need to press [ENTER]. When you type a [COMMAND LETTER], a new screen will be displayed in your terminal application.

7.2.1 Using the Diagnostic Window in ProSoft Configuration Builder

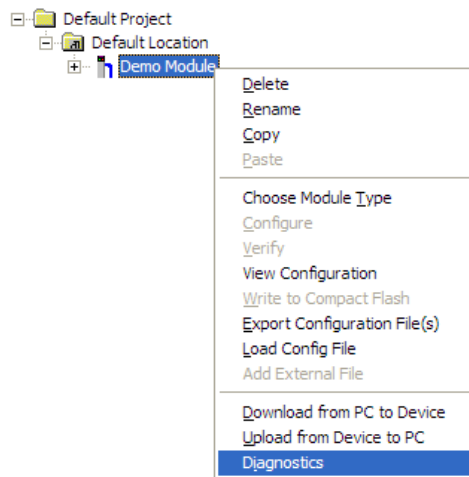
Tip: You can have a ProSoft Configuration Builder Diagnostics window open for more than one module at a time.

To connect to the module's Configuration/Debug serial port

- 1 Start *PCB*, and then select the module to test. Click the right mouse button to open a shortcut menu.

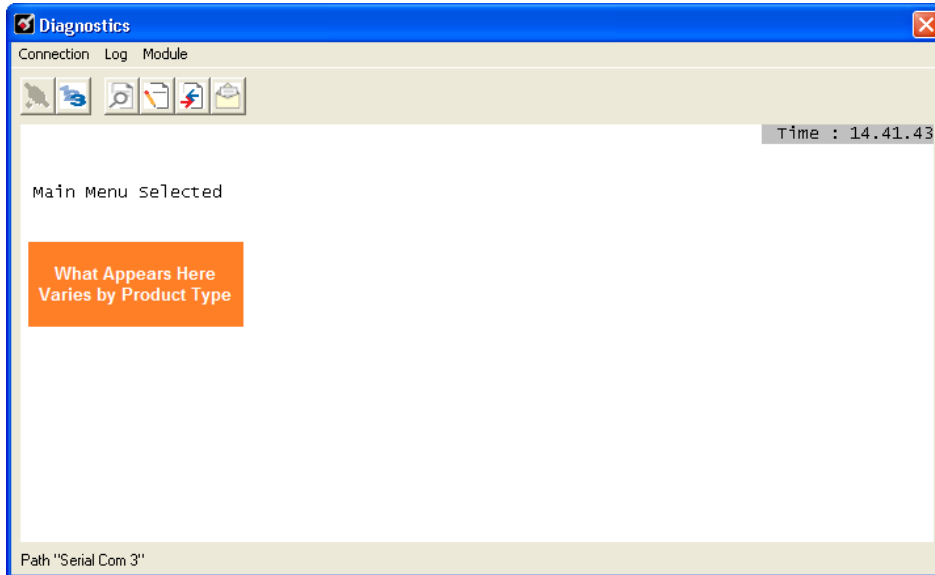


- 2 On the shortcut menu, choose **DIAGNOSTICS**.



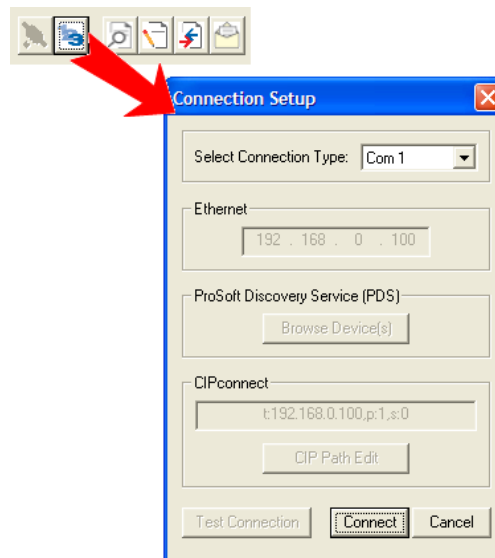
This action opens the *Diagnostics* dialog box.

- 3 Press [?] to open the *Main* menu.



If there is no response from the module, follow these steps:

- 1 Click to configure the connection. On the *Connection Setup* dialog box, select a valid com port or other connection type supported by the module.

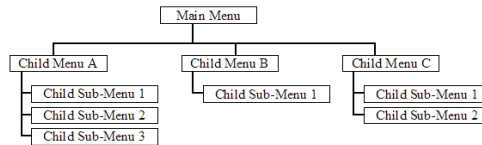


- 2 Verify that the null modem cable is connected properly between your computer's serial port and the module. A regular serial cable will not work.
 - 3 On computers with more than one serial port, verify that your communication program is connected to the same port that is connected to the module.
- If you are still not able to establish a connection, contact ProSoft Technology for assistance.

7.2.2 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.

7.2.3 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 **[?]** key on your computer's keyboard. If the module is connected properly, the following menu will appear.

```

PTQ-MCM MENU
?=Display Menu
V=Version Information
D=Database Menu
C=Clear diagnostic data
B=Backplane Menu
0=Protocol_Serial_MCM 1
1=Protocol_Serial_MCM 2
S=Transfer Configuration from Unit to PC
R=Transfer Configuration from PC to Unit
W=Warm Boot Module
Esc=Exit Program
  
```

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 failures. Only use these commands if you are specifically directed to do so by ProSoft Technology Technical Support staff. Some of these command keys are not listed on the menu, but are active nevertheless. Please be careful when pressing keys so that you do not accidentally execute an unwanted command.

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.

Opening the Database View Menu

Press **[D]** to open the *Database View* menu.

Use this menu command to view the current contents of the module's database. For more information about this submenu, see Database View Menu (page 77).

Clearing Diagnostic Data

Press **[C]** to clear diagnostic data from the module's memory.

Opening the Backplane Menu

Press **[B]** from the Main Menu to view the Backplane Data Exchange List. Use this command to display the configuration and statistics of the backplane data transfer operations.

Tip: Repeat this command at one-second intervals to determine the number of blocks transferred each second.

Opening the Protocol Serial Menu

Press **[0]** or **[1]** from the Main Menu to open the Protocol_Serial_ menu for Ports 1 and 2.

Use this command to view communication status and statistics for the selected port. This information can be useful for trouble-shooting communication problems.

Transferring the Configuration File from The Module to the PC

On the Diagnostics Menu this is referred to as *Send Module Configuration*.

Press **[S]** to send (upload) the configuration file from the module to your PC.

Press **[Y]** to confirm the file transfer, and then follow the instructions on the terminal screen to complete the file transfer process.

After the file has been successfully uploaded, you can open and edit the file to change the module's configuration.

Transferring the Configuration File from the PC to the Module

On the Diagnostics Menu this is referred to as *Receive Module Configuration*.

Press **[R]** to receive (download) the configuration file from your PC to the module and store the file on the module's Compact Flash Card (Personality Module) or Flash RAM.

Press **[Y]** to confirm the file transfer, and then follow the instructions on the terminal screen to complete the file transfer process.

After the file has been successfully downloaded, the module will restart the program and load the new configuration information. Review the new configuration using menu commands **[6]** and **[0]** to verify that the module is configured correctly.

Warm Booting the Module

Press **[W]** from the *Main* menu to warm boot (restart) the module.

This command will cause the program to exit and reload, refreshing configuration parameters that must be set on program initialization. Only use this command if you must force the module to reboot.

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.

7.2.4 Modbus Database View Menu

Press **[D]** to open the *Modbus Database View* menu. Use this command to view the module's internal database values. Press **[?]** to view a list of commands on this menu.

```
DATABASE VIEW MENU
?=Display Menu
0-4=Pages 0 to 4000
S=Show Again
-=Back 5 Pages
P=Previous Page
+=Skip 5 Pages
N=Next Page
D=Decimal Display
H=Hexadecimal Display
F=Float Display
A=ASCII Display
M=Main Menu
```

All data contained in the module's database is available for viewing using the commands. Refer to the Modbus Protocol Specification (page 110) for information on the structure of Modbus messages. Each option available on the menu is discussed in the following topics.

Viewing Register Pages

To view sets of register pages, use the keys described below:

Command	Description
[0]	Display registers 0 to 99
[1]	Display registers 1000 to 1099
[2]	Display registers 2000 to 2099

And so on. The total number of register pages available to view depends on your module's configuration.

Redisplaying the Current Page

Press **[S]** to display the current page of data.

Moving Back Through 5 Pages of Registers

Press **[-]** from the *Database View* menu to skip five pages back in the database to see the 100 registers of data starting 500 registers before the currently displayed page.

Viewing the Previous Page of Registers

Press **[P]** from the *Database View* menu to display the previous page of data.

Moving Forward Through 5 Pages of Registers

Press **[+]** from the *Database View* menu to skip five pages ahead in the database to see 100 registers of data 500 registers ahead of the currently displayed page.

Viewing the Next Page of Registers

Press **[N]** from the *Database View* menu 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

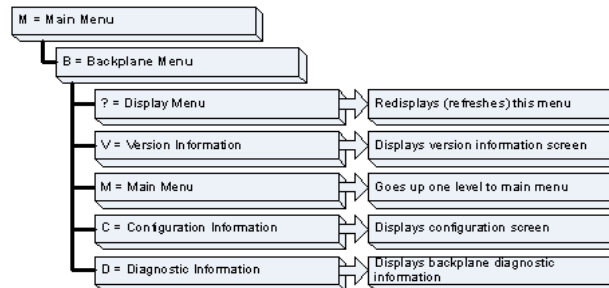
Press **[A]** from the *Database View* menu to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Returning to the Main Menu

Press **[M]** to return to the *Main* menu.

7.2.5 Backplane Menu

Press **[B]** from the Main Menu to view the Backplane Data Exchange List. Use this command to display the configuration and statistics of the backplane data transfer operations. Press **[?]** to view a list of commands available on this 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 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.

Returning to the Main Menu

Press **[M]** to return to the *Main* menu.

Viewing Configuration Information

Press **[C]** to view configuration information for the selected port, protocol, driver or device.

Viewing Backplane Diagnostic Information

Press **[D]** to view Backplane Diagnostic information.

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.

Tip: Repeat this command at one-second intervals to determine the number of blocks transferred each second

7.2.6 Data Analyzer

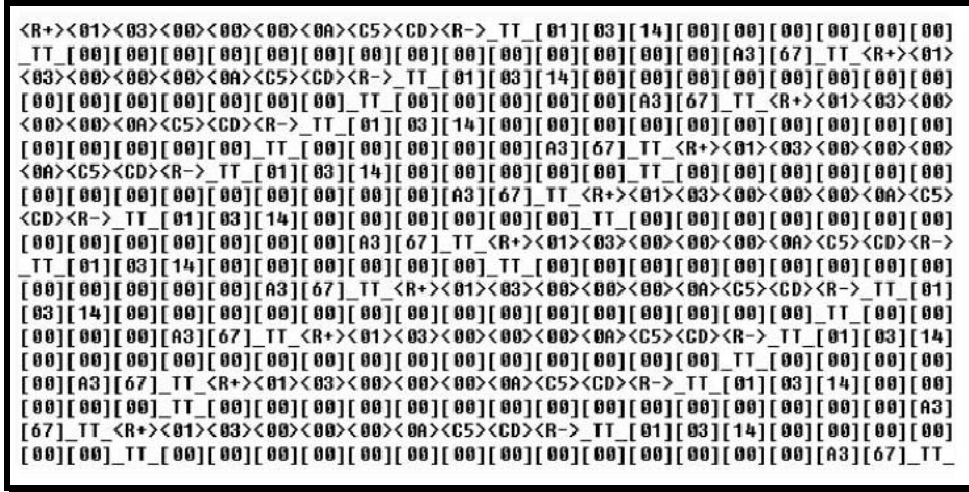
The data analyzer mode allows you to view all bytes of data transferred on each port. Both the transmitted and received data bytes are displayed. Use of this feature is limited without a thorough understanding of the protocol.

Note: The Port selection commands on the Data Analyzer menu differs very slightly in different modules, but the functionality is basically the same. Use the illustration above as a general guide only. Refer to the actual data analyzer menu on your module for the specific port commands to use.

Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please press **[S]** to stop the data analyzer, and then press **[M]** to return to the main menu. This action will allow the module to resume its normal high speed operating mode.

Starting the Data Analyzer

Press **[B]** to start the data analyzer. After the key is pressed, all data transmitted and received on the currently selected port will be displayed. The following illustration shows an example.



The Data Analyzer displays the following special characters:

Character	Definition
[]	Data enclosed in these characters represent data received on the port.
< >	Data enclosed in these characters represent data transmitted on the port.
<R+>	These characters are inserted when the RTS line is driven high on the port.
<R->	These characters are inserted when the RTS line is dropped low on the port.
<CS>	These characters are displayed when the CTS line is recognized high.
TT	These characters are displayed when the timing mark interval has been reached. This parameter is user defined.

Stopping the Data Analyzer

Press **[S]** to stop the data analyzer. Use this option to freeze the display so the data can be analyzed. To restart the analyzer, press **[B]**.

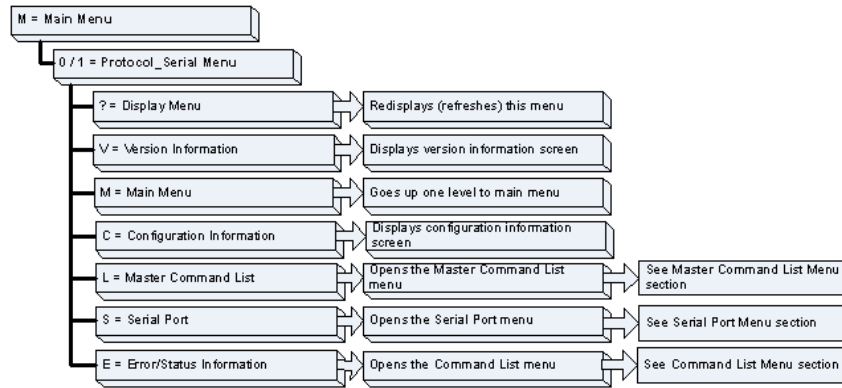
Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please press **[S]** to stop the data analyzer, and then press **[M]** to return to the main menu. This action will allow the module to resume its normal high speed operating mode.

Returning to the Main Menu

Press **[M]** to return to the *Main* menu.

7.2.7 Protocol Serial Menu

Press **[0]** or **[1]** to view protocol serial information for ports 1 and 2, respectively.
Use this command to view a variety of error and status screens for the port.
Press **[?]** to view a list of commands available on this 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.

Tip: Repeat this command at one-second intervals to determine the frequency of program execution.

Returning to the Main Menu

Press [M] to return to the *Main* menu.

Viewing Configuration Information

Press [C] to view configuration information for the selected port, protocol, driver or device.

Opening the Command List Menu

Press [L] to open the Command List menu. Use this command to view the configured command list for the module.

Opening the Serial Port Menu

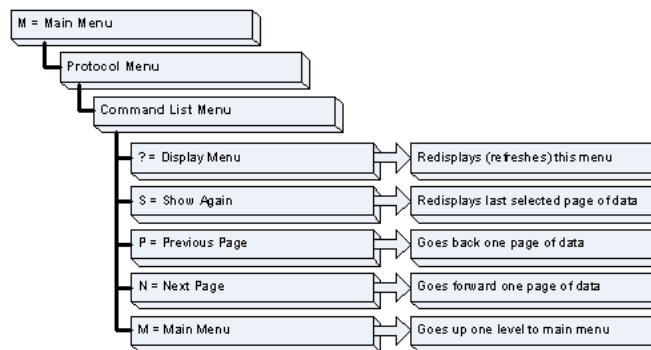
Press [S] to open the Serial Port menu. Use this command to view and change additional serial port driver settings.

Viewing Error and Status Data

Press [E] to display the error/status data for the module.

7.2.8 Master Command Error List Menu

Use this menu to view the command error list for the module. Press [?] to view a list of commands available on this menu.



Redisplaying the Current Page

Press **[S]** to display the current page of data.

Moving Back Through 5 Pages of Commands

Press **[-]** to display data for last 5 page commands.

Viewing the Previous Page of Commands

Press **[P]** to display the previous page of commands.

Moving Forward (Skipping) Through 5 Pages of Commands

Press **[+]** to display data for the next page of commands.

Viewing the Next Page of Commands

Press **[N]** to display the next page of commands.

Returning to the Main Menu

Press **[M]** to return to the *Main* menu.

7.3 Reading Status Data from the Module

The PTQ-MCM module provides the status data in each read block. This data can also be located in the module's database.

7.3.1 Error Status Table

The program maintains an error/status table that is transferred to the processor in each read block. You can use the error/status data to determine the "health" of the module. The data block structure is describe in the Status Data Definition section. The following tables describe the error codes generated by the module.

Standard Modbus Protocol Errors

Code	Description
1	Illegal Function
2	Illegal Data Address
3	Illegal Data Value
4	Failure in Associated Device
5	Acknowledge
6	Busy, Rejected Message

Module Communication Error Codes

Code	Description
-1	CTS modem control line not set before transmit
-2	Timeout while transmitting message
-11	Timeout waiting for response after request
253	Incorrect slave address in response
254	Incorrect function code in response
255	Invalid CRC/LRC value in response

Command List Entry Errors

Code	Description
-41	Invalid enable code
-42	Internal address > maximum address
-43	Invalid node address (< 0 or > 255)
-44	Count parameter set to 0
-45	Invalid function code
-46	Invalid swap code

8 Reference

In This Chapter

❖ Product Specifications	88
❖ Functional Overview	91
❖ Cable Connections	101
❖ Status Data Definition	105
❖ Configuration Data	107
❖ Modbus Protocol Specification	110
❖ Frequently Asked Questions	123

8.1 Product Specifications

The PTQ Modbus Master/Slave Communication Module allows Schneider Electric® Quantum® processors to interface easily with other Modbus protocol compatible devices.

The module acts as an input/output module between the Modbus network and the Quantum backplane. Compatible devices include not only Modicon® PLCs (almost all support the Modbus protocol) but also a wide range of process and control devices from a variety of manufacturers. Many SCADA packages also support the Modbus protocol.

8.1.1 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 four modules can be placed in a rack
- Local rack - The module must be placed in the same rack as 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), and ModSoft.
- Quantum data types supported: 3x, 4x
- High speed data transfer across backplane provides quick data update times
- Sample ladder file available

8.1.2 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% (without condensation)
Vibration	Sine vibration 4-100 Hz in each of the 3 orthogonal axes
Shock	30g, 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 Serial Activity and Error Status
Debug/Configuration Port (Debug)	
CFG Port (DEBUG)	DB-9M PC Compatible RS-232 only No hardware handshaking
Application Ports	
Application Serial Ports (PRT1, PRT2)	DB-9M PC Compatible RS-232/422/485 jumper selectable RS-422/485 screw termination included RS-232 handshaking configurable 500V Optical isolation from backplane

8.1.3 General Specifications - Modbus Master/Slave

Communication parameters	Baud Rate: 110 to 115K baud Stop Bits: 1 or 2 Data Size: 7 or 8 bits Parity: None, Even, Odd RTS Timing delays: 0 to 65535 milliseconds	
Modbus Modes	RTU mode (binary) with CRC-16 ASCII mode with LRC error checking	
Floating Point Data	Floating point data movement supported, including configurable support for Enron, Daniel®, and other implementations	
Modbus Function Codes Supported	1: Read Coil Status 2: Read Input Status 3: Read Holding Registers 4: Read Input Registers 5: Force (Write) Single Coil 6: Preset (Write) Single Holding Register 8: Diagnostics (Slave Only, Responds to Subfunction 00)	15: Force(Write) Multiple Coils 16: Preset (Write) Multiple Holding Registers 17: Report Slave ID (Slave Only) 22: Mask Write Holding Register (Slave Only) 23: Read/Write Holding Registers (Slave Only)

8.1.4 Functional Specifications

Modbus Master

A port configured as a virtual Modbus Master actively issues Modbus commands to other nodes on the Modbus network. The Master ports have an optimized polling characteristic that polls slaves with communication problems less frequently.

Command List	Up to 100 command per Master port, each fully configurable for function, slave address, register to/from addressing and word/bit count.
Polling of command list	Configurable polling of command list, including continuous and on change of data, and dynamically user or automatic enabled.
Status Data	Error codes available on an individual command basis. In addition, a slave status list is maintained per active Modbus Master port.

Modbus Slave

A port configured as a Modbus slave permits a remote Master to interact with all data contained in the module. This data can be derived from other Modbus slave devices on the network, through a Master port, or from the gateway.

Node address	1 to 247 (software selectable)
Status Data	Error codes, counters and port status available per configured slave port

8.2 Functional Overview

8.2.1 About the Modbus Protocol

Modbus is a widely-used protocol originally developed by Modicon in 1978. Since that time, the protocol has been adopted as a standard throughout the automation industry.

The original Modbus specification uses a serial connection to communicate commands and data between Master and Slave devices on a network. Later enhancements to the protocol allow communication over other types of networks. Modbus is a Master/Slave protocol. The Master establishes a connection to the remote Slave. When the connection is established, the Master sends the Modbus commands to the Slave. The PTQ-MCM module can work as a Master and as a Slave.

The PTQ-MCM module also works as an input/output module between itself and the Schneider Electric backplane and processor. The module uses an internal database to pass data and commands between the processor and Master and Slave devices on Modbus networks.

8.2.2 Backplane Data Transfer

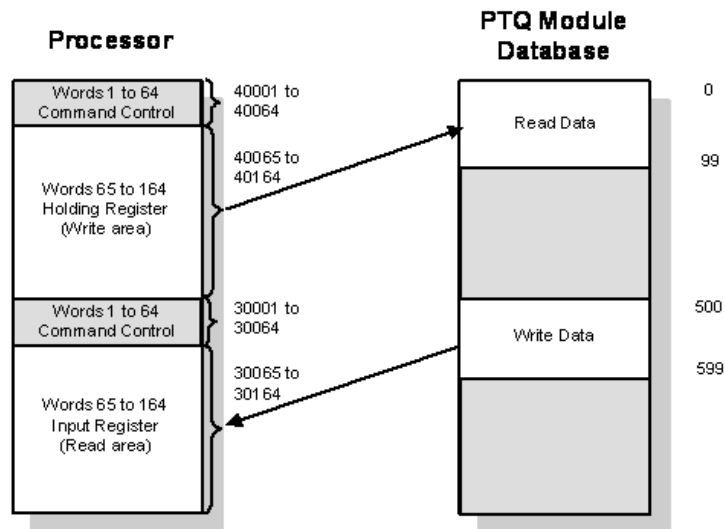
The current version of the PTQ-MCM backplane driver (version 2.10 or newer), uses a Large I/O model, which differs from previous versions of the backplane driver in that it transfers all of the data in the Read and Write databases between the module and the processor on every scan.

The [Backplane Configuration] section of the configuration file defines the starting registers for read and write operations, as well as the number of registers to use for each data area.

[Backplane Configuration]

```
Failure Flag Count      : 0      #
Error/Status Block Pointer : 3000 #Number of register to store error and
                               #status
```

The values in the example configuration file section above are illustrated in the following diagram.



The module transfers the entire read and write areas at the end of every processor scan. The module will hold the processor scan for a certain period of time, which allows the module to transfer the entire read and write areas. This means that the larger the read and write areas, the longer the processor scan time will be. Refer to Module Performance for more detailed information on determining scan times.

Note: The diagram above shows the memory addresses for a Quantum processor. If you are deploying the PTQ-MCM with a Unity processor, substitute %MW for read only data, and %IW for read/write data.

8.2.3 Special Functions

The first 64 words of each block are reserved for Special functions. Each special function block has a Block ID number (shown in parentheses below) that identifies the special function instruction. The PTQ-MCM module supports the following special function blocks:

- Event Command (1000 to 1247 Port 1; 2000 to 2247 Port 2)
- Command Control (5001 to 5006 Port 1; 5101 to 5106 Port 2)
- Warm Boot (9998)
- Cold Boot (9999)

The value in word 0 of this 64 word block is the block sequence number. This number identifies whether the contents of the block have changed. This is the actual trigger to send the function request to the module.

8.2.4 Event Command Block

Event command control blocks send Modbus commands directly from the ladder logic to one of the master ports. The format for these blocks is displayed below:

Event Command Block Request from Processor to the Module

Offset	Description	Length
0	1000 to 1247 or 2000 to 2247	1
1	Internal DB Address	1
2	Point Count	1
3	Swap Code	1
4	Function Code	1
5	Destination Code	1
6 to 63	Spare	1

The block number defines the Modbus port to be considered. Block 1000 commands are directed to Port 1, and block 2000 commands are directed to Port 2. The slave address is represented in the block number in the range of 0 to 255. The sum of these two values determines the block number. The parameters passed with the block are used to construct the command.

- The **Internal DB Address** parameter specifies the module's database location to associate with the command.
- The **Point Count** parameter defines the number of registers for the command.
- The **Swap Code** is used to change the word or byte order.
- The **Node Address** parameter defines the device on the Modbus network to consider.
- The **Function Code** parameter is one of those defined in the section [Modbus Port 1 Commands].

The parameter fields in the block should be completed as required by the selected function code. Each command has its own set of parameters. When the block is received, the module will process it and place the command in the command queue. The module will respond to each command block with a read block. The following table describes the format of this block.

Event Command Block Response from Module to Processor

Offset	Description	Length
0	0	1
1	Write Block ID	1
2	0=Fail, 1=Success	1
3 to n	Spare	

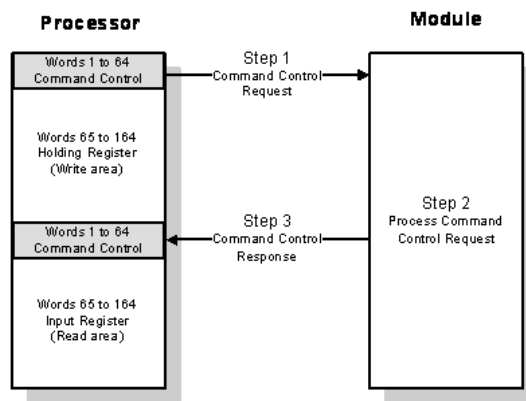
Word two of the block can be used by the processor logic to determine if the command was added to the command queue of the module. The command will only fail if the command queue for the port is full (100 commands for each queue) or the command requested is invalid.

8.2.5 Command Control Block

Processor logic must be built to handle the command control functionality. The logic would typically follow these steps:

- 1 Move the block request to output command control area.
- 2 Move a new value to the output block sequence number.
- 3 If the input block sequence number equals the output block sequence number + 1, copy the block response to appropriate variables in the module's memory.

Note: Command Control blocks are not copied to the module database. You must define variables in the module's main memory, and use processor logic to process the command control request.



The following table shows the contents of the command control area when a command control block such as 9998 (Warm Boot module) is issued.

Note: The diagram above shows the memory addresses for a Quantum processor. If you are deploying the PTQ-MCM with a Unity processor, substitute %MW for read only data, and %IW for read/write data.

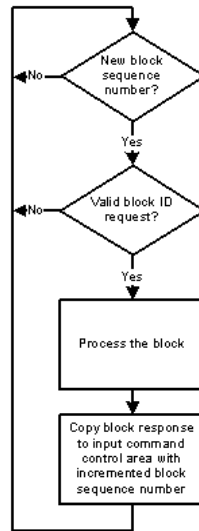
Note: The processor memory locations in the example tables below use the 3x register start and 4x register start values defined in Backplane Data Transfer (page 92). You can configure any valid 3x and 4x start address that is not used by other processes.

Command Control Word	Description
40001	Output sequence number
40002	Block ID
40003	Block request word 1
40004	Block request word 2
40005	Block request word 3
...	...
40064	Block request word 62

The following table shows the results of the PTQ-MCM response to the command control block.

Command Control Word	Description
30001	Input sequence number
30002	Block ID
30003	Block response word 1
30004	Block response word 2
30005	Block response word 3
...	...
30064	Block response word 62

The module recognizes that there is a new block request when it identifies that the block sequence number has changed. If the block ID is valid, the module will process the block and copy the response to the input command control area (3x for Quantum or %IW for Unity). The module will increment the block sequence number by one, as shown in the following illustration.



Command Control Block Request from Processor to the Module

Offset	Description	Length
0	5001 to 5006 or 5101 to 5106	1
1	Command index	1
2	Command index	1
3	Command index	1
4	Command index	1
5	Command index	1
6	Command index	1
7 to 63	Spare	

Blocks in the range of 5001 to 5006 are used for Port 1, and blocks in the range of 5101 to 5106 are used for Port 2. The last digit in the block code defines the number of commands to process in the block. For example, a block code of 5003 contains 3 command indexes for Port 1. The Command index parameters in the block have a range of 0 to 99 and correspond to the master command list entries. The module responds to a command control block with a block containing the number of commands added to the command queue for the port. The format of the block is displayed below:

Command Control Block Response from Module to Processor

Offset	Description	Length
0	5000 to 5006 or 5100 to 5106	1
1	Write Block ID	1
2	Number of commands added to command queue	1
3 to (n+1)	Spare	

8.2.6 Warm Boot Block (9998) - PTQ 1 to 63

This block is sent from the processor to the module when the module is required to perform a warm-boot (software reset) operation. The following table describes the format of the control block.

Offset	Description	Length
0	9998	1
1 to 63	Spare	63

8.2.7 Cold Boot Block (9999) - PTQ 1 to 63

This block is sent from the processor to the module when the module is required to perform the cold boot (hardware reset) operation. This block is sent to the module when a hardware problem is detected by the processor logic that requires a hardware reset. The following table describes the format of the control block.

Offset	Description	Length
0	9999	1
1 to 63	Spare	63

8.2.8 Pass-Thru Control Blocks

Overview

Caution: This feature allows a remote Modbus master to directly update the processor memory. The user should carefully read this section before considering this feature in order to avoid transferring data to unexpected memory areas in the Quantum processor.

The pass-thru operation consists of the slave port receiving a write command from the master and passing the data directly to the processor without updating the PTQ-MCM internal database. The module supports direct pass-thru with a swap byte option.

During block pass-thru operation the module generates a specific block to the processor when it receives a write command from the master. The module uses different block IDs depending on the type of write command received (Modbus functions 5,6,15 and 16). The user should implement processor logic to handle the pass-thru block once it is received. The block contains relevant information such as data count, destination address, and data

During direct pass-thru operation the module passes the data received from a write command to the processor. The module will use the destination address from the Modbus command to select the start memory address where the data will be copied to.

Pass-Thru Configuration

Each port can be configured for pass-thru functionality.

Type: 0 #Type of port (0=Master, 1=Slave, 4=Direct Pass-Thru, 5=Direct Pass-Thru Swap)

The following codes are supported for Pass-Thru:

Code	Description	Modbus Functions Supported
4	Direct Pass-Thru	6, 16
5	Direct Pass-Thru with Byte Swap	6, 16

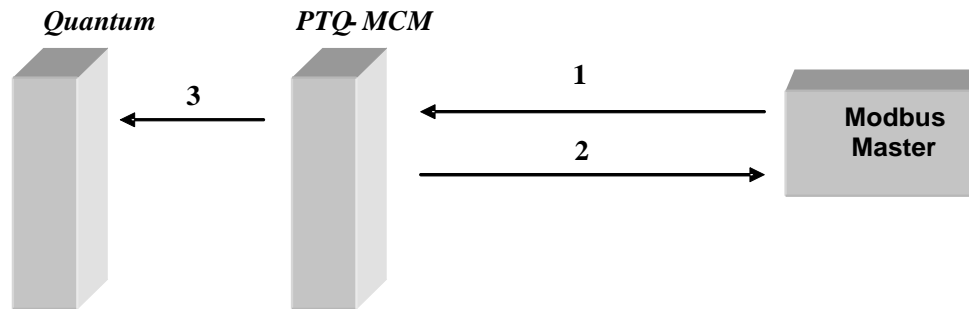
The maximum number of bits supported for Modbus function 15 is 800 bits. The maximum number of words supported for Modbus function 16 is 125 words.

Direct Pass-Thru

When a PTQ-MCM port is configured for pass-thru operation it will work as a Modbus slave transferring the data received directly to the processor memory (holding register memory). The configuration of backplane commands is not required to transfer data from the module to the processor during direct pass-thru operation. This feature does not require any processor program to receive the data from the module.

The module uses the destination address received from the Modbus command to determine the memory location in the processor where the data will be written to.

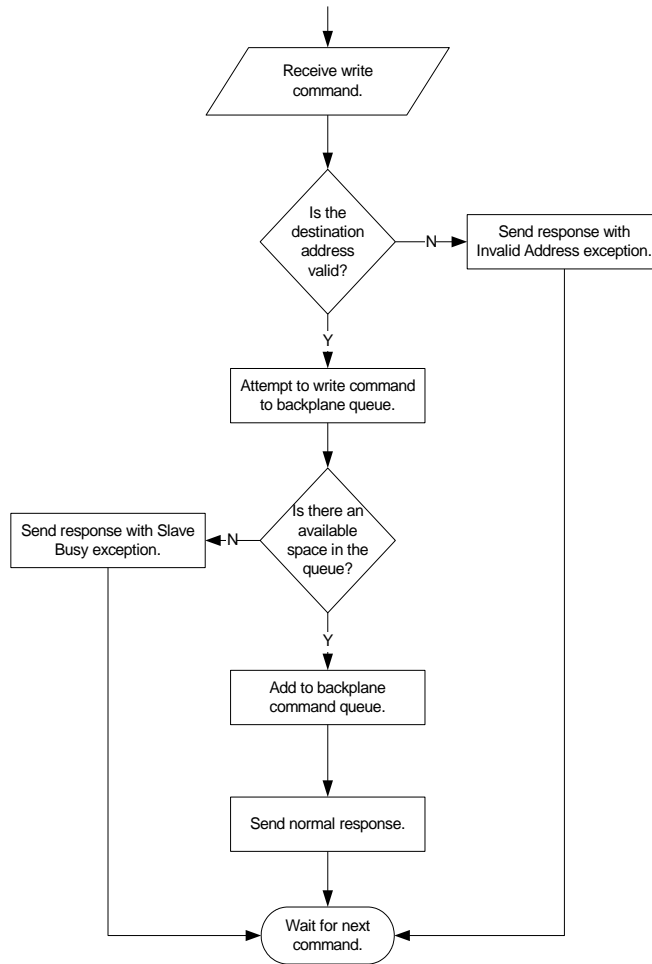
The following illustration shows the three basic steps performed during the pass-thru operation.



The module receives a Modbus write command. For example:

- 1 Modbus Function: Preset Multiple Registers (function 16)
- 2 Count: 10 words
- 3 Destination Address: 500
- 4 The module verifies that the destination address is a valid address in the processor. If not, it sends a response with an exception code (illegal data address). If the destination address is valid the module will insert the data into its internal backplane command queue to transfer to the Quantum. After the request is placed in the queue the module sends the response to the master device.
- 5 The time period for the data to transfer to the Quantum processor will basically depend on the processor's scan rate. Depending on the processor scan rate the module could receive a command while the queue is full. In this situation the module will send a "Slave Device Busy" exception response (exception code 6) to the Modbus master and the data will not be copied to the processor.
- 6 The module writes the data directly to the Quantum address defined by the Modbus command. For the example from item 1 the data would be written starting at Quantum address 40501.

The following flow chart show the implementation of the direct pass-thru block:



The following Modbus commands are accepted during direct pass-thru operation:

Function Code	Description Code	Maximum Count	Quantum Memory Area
6	Preset (Write) Single Register	1	4x (holding register)
16	Preset (Write) Multiple Registers	125	4x (holding register)

The destination address where the data will be copied at the processor is calculated as: the module will add 40001 to the destination address received.

Both ports can operate in pass-thru mode simultaneously. If the port is configured for pass-thru swap, the module will swap the data bytes (high order/low order) before transferring the data to the processor.

Since the module transfers the data directly to the processor the PTQ-MCM database is not updated during pass-thru operation.

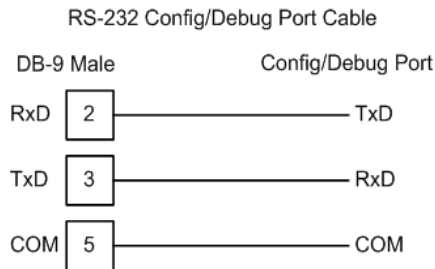
8.3 Cable Connections

The application ports on the PTQ-MCM module support RS-232, RS-422, and RS-485 interfaces. Please inspect the module to ensure that the jumpers are set correctly to correspond with the type of interface you are using.

Note: When using RS-232 with radio modem applications, some radios or modems require hardware handshaking (control and monitoring of modem signal lines). Enable this in the configuration of the module by setting the UseCTS parameter to 1.

8.3.1 RS-232 Configuration/Debug Port

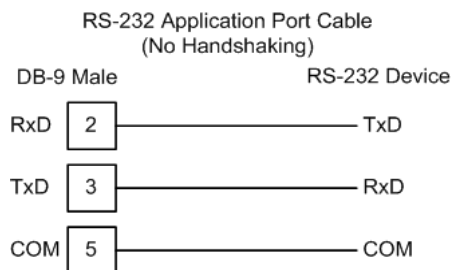
This port is physically a DB-9 connection. This port permits a PC based terminal emulation program to view configuration and status data in the module and to control the module. The cable for communications on this port is shown in the following diagram:



The Ethernet port on this module (if present) is inactive.

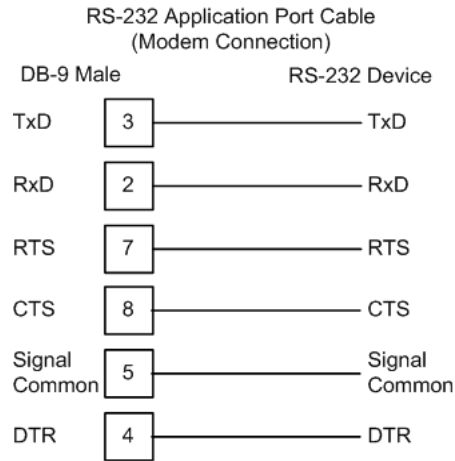
8.3.2 RS-232 Application Port(s)

When the RS-232 interface is selected, the use of hardware handshaking (control and monitoring of modem signal lines) is user definable. If no hardware handshaking will be used, here are the cable pinouts to connect to the port.



RS-232: Modem Connection (Hardware Handshaking Required)

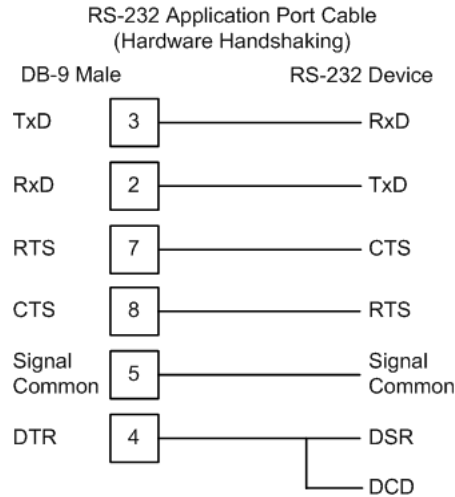
This type of connection is required between the module and a modem or other communication device.



The "Use CTS Line" parameter for the port configuration should be set to 'Y' for most modem applications.

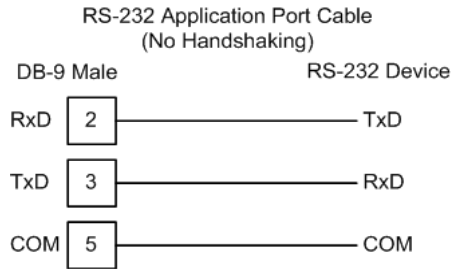
RS-232: Null Modem Connection (Hardware Handshaking)

This type of connection is used when the device connected to the module requires hardware handshaking (control and monitoring of modem signal lines).

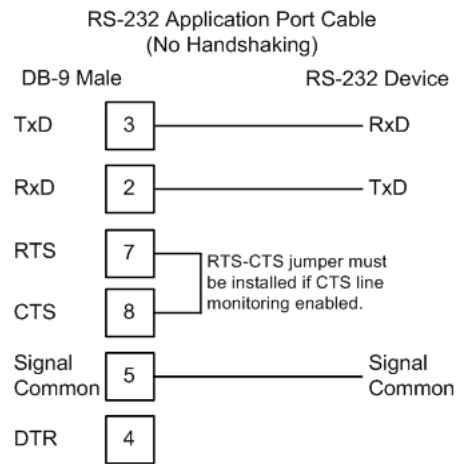


RS-232: Null Modem Connection (No Hardware Handshaking)

This type of connection can be used to connect the module to a computer or field device communication port.

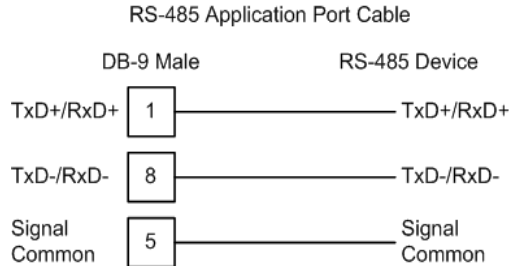


Note: For most null modem connections where hardware handshaking is not required, the *Use CTS Line* parameter should be set to **N** and no jumper will be required between Pins 7 (RTS) and 8 (CTS) on the connector. If the port is configured with the *Use CTS Line* set to **Y**, then a jumper is required between the RTS and the CTS lines on the port connection.



8.3.3 RS-485 Application Port(s)

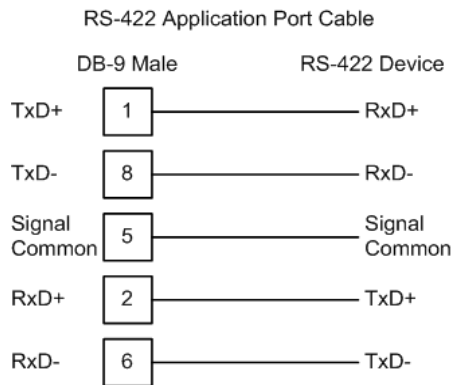
The RS-485 interface requires a single two or three wire cable. The Common connection is optional, depending on the RS-485 network devices used. The cable required for this interface is shown below:



Note: Terminating resistors are generally not required on the RS-485 network, unless you are experiencing communication problems that can be attributed to signal echoes or reflections. In these cases, installing a 120-ohm terminating resistor between pins 1 and 8 on the module connector end of the RS-485 line may improve communication quality.

8.3.4 RS-422

The RS-422 interface requires a single four or five wire cable. The Common connection is optional, depending on the RS-422 network devices used. The cable required for this interface is shown below:



RS-485 and RS-422 Tip

If communication in the RS-422 or RS-485 mode does not work at first, despite all attempts, try switching termination polarities. Some manufacturers interpret + and -, or A and B, polarities differently.

8.4 Status Data Definition

This section contains a description of the members present in the status block. This data is transferred from the module to the processor as part of each read block.

8.4.1 Status Data Block Structure

Offset	Content	Description	Length (Words)
0	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.	1
1	Product Code	These two registers contain the product code of "MCM"	2
3	Product Version	These two registers contain the product version for the current running software.	2
5	Operating System	These two registers contain the month and year values for the program operating system.	2
7	Run Number	These two registers contain the run number value for the currently running software.	2
9	Port 1 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.	1
10	Port 1 Command List Response	This field contains the number of slave response messages received on the port.	1
11	Port 1 Command List Errors	This field contains the number of command errors processed on the port. These errors could be due to a bad response or command.	1
12	Port 1 Requests	This field contains the total number of messages sent out of the port.	1
13	Port 1 Responses	This field contains the total number of messages received on the port.	1
14	Port 1 Errors Sent	This field contains the total number of message errors sent out of the port.	1
15	Port 1 Errors Received	This field contains the total number of message errors received on the port.	1
16	Port 2 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.	1
17	Port 2 Command List Response	This field contains the number of slave response messages received on the port.	1
18	Port 2 Command List Errors	This field contains the number of command errors processed on the port. These errors could be due to a bad response or command.	1
19	Port 2 Requests	This field contains the total number of messages sent out the port.	1
20	Port 2 Responses	This field contains the total number of messages received on the port.	1
21	Port 2 Errors Sent	This field contains the total number of message errors sent out of the port.	1
22	Port 2 Errors Received	This field contains the total number of message errors received on the port.	1

Offset	Content	Description	Length (Words)
23	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.	1
24	Write Block Count	This field contains the total number of write blocks transferred from the processor to the module.	1
25	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.	1
26	Command Event Block Count	This field contains the total number of command event blocks received from the processor.	1
27	Command Block Count	This field contains the total number of command blocks received from the processor.	1
28	Error Block Count	This field contains the total number of block errors recognized by the module.	1
29	Pass Thru Received	Pass thru request received on both ports.	1
30	Port 1 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.	1
31	Port 1 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.	1
32	Port 2 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.	1
33	Port 2 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.	1

8.5 Configuration Data

This section contains listings of the PTQ-MCM module's database that are related to the module's configuration. This data is available to any node on the network and is read from the Quantum processor when the module first initializes.

Refer to this section for information about the configuration file parameters.

8.5.1 Port 1 Setup

Database Address	Content
5010	Enable
5011	Pass-Thru
5012	Type
5013	Float Flag
5014	Float Start
5015	Float Offset
5016	Protocol
5017	Baud Rate
5018	Parity
5019	Data Bits
5020	Stop Bits
5021	RTS On
5022	RTS Off
5023	Minimum Response Time
5024	Use CTS Line
5025	Slave ID
5026	Bit in Offset
5027	Word in Offset
5028	Out in Offset
5029	Holding Reg Offset
5030	Command Count
5031	Minimum Command Delay
5032	Command Error Pointer
5033	Response Timeout
5034	Retry Count
5035	Error Delay Counter

8.5.2 Port 2 Setup

Database Address	Content
5040	Enable
5041	Passthru
5042	Type
5043	Float Flag
5044	Float Start
5045	Float Offset
5046	Protocol
5047	Baud Rate
5048	Parity
5049	Data Bits
5050	Stop Bits
5051	RTS On
5052	RTS Off
5053	Minimum Response Time
5054	Use CTS Line
5055	Slave ID
5056	Bit in Offset
5057	Word in Offset
5058	Out in Offset
5059	Holding Reg Offset
5060	Command Count
5061	Minimum Command Delay
5062	Command Error Pointer
5063	Response Timeout
5064	Retry Count
5065	Error Delay Counter

8.5.3 Port 1 Commands

Database Address	Content
5070	Command #1
5078	Command #2
-	-
5862	Command #100

Each of these commands uses eight registers (Enable, Internal Address, Poll Interval, Reg Count, Swap Code, Node Address, Modbus Func, MB Address in Device).

8.5.4 Port 2 Commands

Database Address	Content
5870	Command #1
5877	Command #2
-	-
6662	Command #100

Each of these commands uses eight registers (Enable, Internal Address, Poll Interval, Reg Count, Swap Code, Node Address, Modbus Func, MB Address in Device).

8.6 Modbus Protocol Specification

The following pages give additional reference information regarding the Modbus protocol commands supported by the PTQ-MCM.

8.6.1 Commands Supported by the Module

The format of each command in the list depends on the Modbus Function Code being executed.

The following table lists the functions supported by the module.

Function Code	Definition	Supported in Master	Supported in Slave
1	Read Coil Status	X	X
2	Read Input Status	X	X
3	Read Holding Registers	X	X
4	Read Input Registers	X	X
5	Set Single Coil	X	X
6	Single Register Write	X	X
8	Diagnostics		X
15	Multiple Coil Write	X	X
16	Multiple Register Write	X	X
17	Report Slave ID		X
22	Mask Write 4X		X
23	Read/Write		X

Each command list record has the same general format. The first part of the record contains the information relating to the communication module and the second part contains information required to interface to the Modbus slave device.

8.6.2 Read Coil Status (Function Code 01)

Query

This function allows the user to obtain the ON/OFF status of logic coils used to control discrete outputs from the addressed Slave only. Broadcast mode is not supported with this function code. In addition to the Slave address and function fields, the message requires that the information field contain the initial coil address to be read (Starting Address) and the number of locations that will be interrogated to obtain status data.

The addressing allows up to 2000 coils to be obtained at each request; however, the specific Slave device may have restrictions that lower the maximum quantity. The coils are numbered from zero; (coil number 1 = zero, coil number 2 = one, coil number 3 = two, and so on).

The following table is a sample read output status request to read coils 0020 to 0056 from Slave device number 11.

Adr	Func	Data Start Pt Hi	Data Start Pt Lo	Data # Of Pts Ho	Data # Of Pts Lo	Error Check Field
11	01	00	13	00	25	CRC

Response

An example response to Read Coil Status is as shown in Figure C2. The data is packed one bit for each coil. The response includes the Slave address, function code, quantity of data characters, the data characters, and error checking. Data will be packed with one bit for each coil (1 = ON, 0 = OFF). The low order bit of the first character contains the addressed coil, and the remainder follow. For coil quantities that are not even multiples of eight, the last characters will be filled in with zeros at high order end. The quantity of data characters is always specified as quantity of RTU characters, that is, the number is the same whether RTU or ASCII is used.

Because the Slave interface device is serviced at the end of a controller's scan, data will reflect coil status at the end of the scan. Some Slaves will limit the quantity of coils provided each scan; thus, for large coil quantities, multiple PC transactions must be made using coil status from sequential scans.

Adr	Func	Byte Count	Data Coil Status 20 to 27	Data Coil Status 28 to 35	Data Coil Status 36 to 43	Data Coil Status 44 to 51	Data Coil Status 52 to 56	Error Check Field
11	01	05	CD	6B	B2	OE	1B	CRC

The status of coils 20 to 27 is shown as CD(HEX) = 1100 1101 (Binary). Reading left to right, this shows that coils 27, 26, 23, 22, and 20 are all on. The other coil data bytes are decoded similarly. Due to the quantity of coil statuses requested, the last data field, which is shown 1B (HEX) = 0001 1011 (Binary), contains the status of only 5 coils (52 to 56) instead of 8 coils. The 3 left most bits are provided as zeros to fill the 8-bit format.

8.6.3 Read Input Status (Function Code 02)

Query

This function allows the user to obtain the ON/OFF status of discrete inputs in the addressed Slave PC Broadcast mode is not supported with this function code. In addition to the Slave address and function fields, the message requires that the information field contain the initial input address to be read (Starting Address) and the number of locations that will be interrogated to obtain status data.

The addressing allows up to 2000 inputs to be obtained at each request; however, the specific Slave device may have restrictions that lower the maximum quantity. The inputs are numbered form zero; (input 10001 = zero, input 10002 = one, input 10003 = two, and so on, for a 584).

The following table is a sample read input status request to read inputs 10197 to 10218 from Slave number 11.

Adr	Func	Data Start Pt Hi	Data Start Pt Lo	Data #of Pts Hi	Data #of Pts Lo	Error Check Field
11	02	00	C4	00	16	CRC

Response

An example response to Read Input Status is as shown in Figure C4. The data is packed one bit for each input. The response includes the Slave address, function code, quantity of data characters, the data characters, and error checking. Data will be packed with one bit for each input (1=ON, 0=OFF). The lower order bit of the first character contains the addressed input, and the remainder follow. For input quantities that are not even multiples of eight, the last characters will be filled in with zeros at high order end. The quantity of data characters is always specified as a quantity of RTU characters, that is, the number is the same whether RTU or ASCII is used.

Because the Slave interface device is serviced at the end of a controller's scan, data will reflect input status at the end of the scan. Some Slaves will limit the quantity of inputs provided each scan; thus, for large coil quantities, multiple PC transactions must be made using coil status for sequential scans.

Adr	Func	Byte Count	Data Discrete Input 10197 to 10204	Data Discrete Input 10205 to 10212	Data Discrete Input 10213 to 10218	Error Check Field
11	02	03	AC	DB	35	CRC

The status of inputs 10197 to 10204 is shown as AC (HEX) = 10101 1100 (binary). Reading left to right, this show that inputs 10204, 10202, and 10199 are all on. The other input data bytes are decoded similar.

Due to the quantity of input statuses requested, the last data field which is shown as 35 HEX = 0011 0101 (binary) contains the status of only 6 inputs (10213 to 10218) instead of 8 inputs. The two left-most bits are provided as zeros to fill the 8-bit format.

8.6.4 Read Holding Registers (Function Code 03)

Query

Read Holding Registers (03) allows the user to obtain the binary contents of holding registers 4xxxx in the addressed Slave. The registers can store the numerical values of associated timers and counters which can be driven to external devices. The addressing allows up to 125 registers to obtained at each request; however, the specific Slave device may have restriction that lower this maximum quantity. The registers are numbered form zero (40001 = zero, 40002 = one, and so on). The broadcast mode is not allowed.

The example below reads registers 40108 through 40110 from Slave 584 number 11.

Adr	Func	Data Start Reg Hi	Data Start Reg Lo	Data #of Regs Hi	Data #of Regs Lo	Error Check Field
11	03	00	6B	00	03	CRC

Response

The addressed Slave responds with its address and the function code, followed by the information field. The information field contains 1 byte describing the quantity of data bytes to be returned. The contents of the registers requested (DATA) are two bytes each, with the binary content right justified within each pair of characters. The first byte includes the high order bits and the second, the low order bits.

Because the Slave interface device is normally serviced at the end of the controller's scan, the data will reflect the register content at the end of the scan. Some Slaves will limit the quantity of register content provided each scan; thus for large register quantities, multiple transmissions will be made using register content from sequential scans.

In the example below, the registers 40108 to 40110 have the decimal contents 555, 0, and 100 respectively.

Adr	Func	ByteCnt	Hi Data	Lo Data	Hi Data	Lo Data	Hi Data	Lo Data	Error Check Field
11	03	06	02	2B	00	00	00	64	CRC

8.6.5 Read Input Registers (Function Code 04)

Query

Function code 04 obtains the contents of the controller's input registers at addresses 3xxxx. These locations receive their values from devices connected to the I/O structure and can only be referenced, not altered from within the controller. The addressing allows up to 125 registers to be obtained at each request; however, the specific Slave device may have restrictions that lower this maximum quantity. The registers are numbered for zero (30001 = zero, 30002 = one, and so on). Broadcast mode is not allowed.

The example below requests the contents of register 3009 in Slave number 11.

Adr	Func	Data Start Reg Hi	Data Start Reg Lo	Data #of Regs Hi	Data #of Regs Lo	Error Check Field
11	04	00	08	00	01	CRC

Response

The addressed Slave responds with its address and the function code followed by the information field. The information field contains 1 byte describing the quantity of data bytes to be returned. The contents of the registers requested (DATA) are 2 bytes each, with the binary content right justified within each pair of characters. The first byte includes the high order bits and the second, the low order bits.

Because the Slave interface is normally serviced at the end of the controller's scan, the data will reflect the register content at the end of the scan. Each PC will limit the quantity of register contents provided each scan; thus for large register quantities, multiple PC scans will be required, and the data provided will be from sequential scans.

In the example below the register 3009 contains the decimal value 0.

Adr	Func	Byte Count	Data Input Reg Hi	Data Input Reg Lo	Error Check Field
11	04	02	00	00	E9

8.6.6 Force Single Coil (Function Code 05)

Query

This message forces a single coil either ON or OFF. Any coil that exists within the controller can be forced to either state (ON or OFF). However, because the controller is actively scanning, unless the coil is disabled, the controller can also alter the state of the coil. Coils are numbered from zero (coil 0001 = zero, coil 0002 = one, and so on). The data value 65,280 (FF00 HEX) will set the coil ON and the value zero will turn it OFF; all other values are illegal and will not affect that coil.

The use of Slave address 00 (Broadcast Mode) will force all attached Slaves to modify the desired coil.

Note: Functions 5, 6, 15, and 16 are the only messages that will be recognized as valid for broadcast.

The example below is a request to Slave number 11 to turn ON coil 0173.

Adr	Func	Data Coil # Hi	Data Coil # Lo	Data On/off Ind	Data	Error Check Field
11	05	00	AC	FF	00	CRC

Response

The normal response to the Command Request is to re-transmit the message as received after the coil state has been altered.

Adr	Func	Data Coil # Hi	Data Coil # Lo	Data On/ Off	Data	Error Check Field
11	05	00	AC	FF	00	CRC

The forcing of a coil via MODBUS function 5 will be accomplished regardless of whether the addressed coil is disabled or not (*In ProSoft products, the coil is only affected if the necessary ladder logic is implemented*).

Note: The Modbus protocol does not include standard functions for testing or changing the DISABLE state of discrete inputs or outputs. Where applicable, this may be accomplished via device specific Program commands (*In ProSoft products, this is only accomplished through ladder logic programming*).

Coils that are reprogrammed in the controller logic program are not automatically cleared upon power up. Thus, if such a coil is set ON by function Code 5 and (even months later), an output is connected to that coil, the output will be "hot".

8.6.7 Preset Single Register (Function Code 06)

Query

Function (06) allows the user to modify the contents of a holding register. Any holding register that exists within the controller can have its contents changed by this message. However, because the controller is actively scanning, it also can alter the content of any holding register at any time. The values are provided in binary up to the maximum capacity of the controller unused high order bits must be set to zero. When used with Slave address zero (Broadcast mode) all Slave controllers will load the specified register with the contents specified.

Note Functions 5, 6, 15, and 16 are the only messages that will be recognized as valid for broadcast.

Adr	Func	Data Start Reg Hi	Data Start Reg Lo	Data #of Regs Hi	Data #of Regs Lo	Error Check Field
11	06	00	01	00	03	CRC

Response

The response to a preset single register request is to re-transmit the query message after the register has been altered.

Adr	Func	Data Reg Hi	Data Reg Lo	Data Input Reg Hi	Data Input Reg Lo	Error Check Field
11	06	00	01	00	03	CRC

8.6.8 Diagnostics (Function Code 08)

Modbus function code 08 provides a series of tests for checking the communication system between a Master device and a slave, or for checking various internal error conditions within a slave.

The function uses a two-byte sub-function code field in the query to define the type of test to be performed. The slave echoes both the function code and sub-function code in a normal response. Some of the diagnostics commands cause data to be returned from the remote device in the data field of a normal response.

In general, issuing a diagnostic function to a remote device does not affect the running of the user program in the remote device. Device memory bit and register data addresses are not accessed by the diagnostics. However, certain functions can optionally reset error counters in some remote devices.

A server device can, however, be forced into 'Listen Only Mode' in which it will monitor the messages on the communications system but not respond to them. This can affect the outcome of your application program if it depends upon any further exchange of data with the remote device. Generally, the mode is forced to remove a malfunctioning remote device from the communications system.

Sub-function Codes Supported

Only Sub-function 00 is supported by the PTQ-MCM module.

00 Return Query Data

The data passed in the request data field is to be returned (looped back) in the response. The entire response message should be identical to the request.

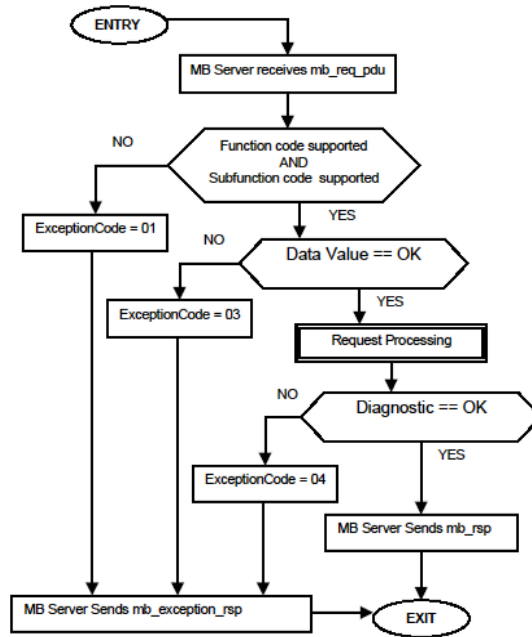
Sub-function	Data Field (Request)	Data Field (Response)
00 00	Any	Echo Request Data

Example and State Diagram

Here is an example of a request to remote device to Return Query Data. This uses a sub-function code of zero (00 00 hex in the two-byte field). The data to be returned is sent in the two-byte data field (A5 37 hex).

Request		Response	
Field Name	(Hex)	Field Name	(Hex)
Function	08	Function	08
Sub-function Hi	00	Sub-function Hi	00
Sub-function Lo	00	Sub-function Lo	00
Data Hi	A5	Data Hi	A5
Data Lo	37	Data Lo	27

The data fields in responses to other kinds of queries could contain error counts or other data requested by the sub-function code.



8.6.9 Force Multiple Coils (Function Code 15)

Query

This message forces each coil in a consecutive block of coils to a desired ON or OFF state. Any coil that exists within the controller can be forced to either state (ON or OFF). However, because the controller is actively scanning, unless the coils are disabled, the controller can also alter the state of the coil. Coils are numbered from zero (coil 00001 = zero, coil 00002 = one, and so on). The desired status of each coil is packed in the data field, one bit for each coil (1= ON, 0= OFF). The use of Slave address 0 (Broadcast Mode) will force all attached Slaves to modify the desired coils.

Note: Functions 5, 6, 15, and 16 are the only messages (other than Loopback Diagnostic Test) that will be recognized as valid for broadcast.

The following example forces 10 coils starting at address 20 (13 HEX). The two data fields, CD =1100 and 00 = 0000 000, indicate that coils 27, 26, 23, 22, and 20 are to be forced on.

Adr	Func	Hi Add	Lo Add	Quantity	Byte Cnt	Data Coil Status 20 to 27	Data Coil Status 28 to 29	Error Check Field
11	0F	00	13	00	0A	02	CD	00 CRC

Response

The normal response will be an echo of the Slave address, function code, starting address, and quantity of coils forced.

Adr	Func	Hi Addr	Lo Addr	Quantity	Error Check Field
11	0F	00	13	00	0A CRC

The writing of coils via Modbus function 15 will be accomplished regardless of whether the addressed coils are disabled or not.

Coils that are unprogrammed in the controller logic program are not automatically cleared upon power up. Thus, if such a coil is set ON by function code 15 and (even months later) an output is connected to that coil, the output will be hot.

8.6.10 Preset Multiple Registers (Function Code 16)

Query

Holding registers existing within the controller can have their contents changed by this message (a maximum of 60 registers). However, because the controller is actively scanning, it also can alter the content of any holding register at any time. The values are provided in binary up to the maximum capacity of the controller (16-bit for the 184/384 and 584); unused high order bits must be set to zero.

Note: Function codes 5, 6, 15, and 16 are the only messages that will be recognized as valid for broadcast.

Adr	Func	Hi Add	Lo Add	Quantity	Byte Cnt	Hi Data	Lo Data	Hi Data	Lo Data	Error Check Field
11	10	00	87	00	02 04	00	0A	01	02	CRC

Response

The normal response to a function 16 query is to echo the address, function code, starting address and number of registers to be loaded.

Adr	Func	Hi Addr	Lo Addr	Quantity		Error Check Field
11	10	00	87	00	02	56

8.6.11 Modbus Exception Responses

When a Modbus Master sends a request to a Slave device, it expects a normal response. One of four possible events can occur from the Master's query:

- If the server device receives the request without a communication error, and can handle the query normally, it returns a normal response.
- If the server does not receive the request due to a communication error, no response is returned. The Master program will eventually process a timeout condition for the request.
- If the server receives the request, but detects a communication error (parity, LRC, CRC, ...), no response is returned. The Master program will eventually process a timeout condition for the request.
- If the server receives the request without a communication error, but cannot handle it (for example, if the request is to read a non-existent output or register), the server will return an exception response informing the Master of the nature of the error.

The exception response message has two fields that differentiate it from a normal response:

Function Code Field: In a normal response, the server echoes the function code of the original request in the function code field of the response. All function codes have a most-significant bit (MSB) of 0 (their values are all below 80 hexadecimal). In an exception response, the server sets the MSB of the function code to 1. This makes the function code value in an exception response exactly 80 hexadecimal higher than the value would be for a normal response.

With the function code's MSB set, the Master's application program can recognize the exception response and can examine the data field for the exception code.

Data Field: In a normal response, the server may return data or statistics in the data field (any information that was requested in the request). In an exception response, the server returns an exception code in the data field. This defines the server condition that caused the exception.

The following table shows an example of a Master request and server exception response.

Request		Response	
Field Name	(Hex)	Field Name	(Hex)
Function	01	Function	81
Starting Address Hi	04	Exception Code	02
Starting Address Lo	A1		
Quantity of Outputs Hi	00		
Quantity of Outputs Lo	01		

In this example, the Master addresses a request to server device. The function code (01) is for a Read Output Status operation. It requests the status of the output at address 1245 (04A1 hex). Note that only that one output is to be read, as specified by the number of outputs field (0001).

If the output address is non-existent in the server device, the server will return the exception response with the exception code shown (02). This specifies an illegal data address for the Slave.

Modbus Exception Codes

Code	Name	Meaning
01	Illegal Function	The function code received in the query is not an allowable action for the Slave. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the Slave is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.
02	Illegal Data Address	The data address received in the query is not an allowable address for the Slave. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed; a request with offset 96 and length 5 will generate exception 02.
03	Illegal Data Value	A value contained in the query data field is not an allowable value for Slave. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program, because the Modbus protocol is unaware of the significance of any particular value of any particular register.
04	Slave Device Failure	An unrecoverable error occurred while the Slave was attempting to perform the requested action.
05	Acknowledge	Specialized use in conjunction with programming commands. The Slave has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the Master. The Master can next issue a poll program complete message to determine if processing is completed.
06	Slave Device Busy	Specialized use in conjunction with programming commands. The Slave is engaged in processing a long-duration program command. The Master should retransmit the message later when the Slave is free.
08	Memory Parity Error	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The Slave attempted to read record file, but detected a parity error in the memory. The Master can retry the request, but service may be required on the Slave device.
0a	Gateway Path Unavailable	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.
0b	Gateway Target Device Failed To Respond	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

8.7 Frequently Asked Questions

8.7.1 What kind of data transfer rates can I expect between the PLC and the module?

Data transfer rates between the PLC and the module depend on a number of variables, among them the number of words being transferred per command, the amount of other network traffic at the time data is being transferred, and overall processor scan times.

8.7.2 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 the other members of the ProLinx family such as the 4202-MNET-DFCM (if you require DF1 connectivity for example although many others are available). This module for example would allow you to communicate with DF1 devices and allow you to map the contents of its memory using Modbus TCP/IP.

8.7.3 Can I use the module in a hot backup system?

Support for Hot Backup is not currently implemented in the module. We are currently investigating the addition of this functionality but until this development can be finalized, it may be possible to use one of the 4000 series of ProSoft communication products. Please call our technical support technicians when considering this application.

9 Support, Service & Warranty

In This Chapter

- ❖ Contacting Technical Support 125
- ❖ Warranty Information 127

9.1 Contacting Technical Support

ProSoft Technology, Inc. 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, ProSoft's 24/7 after-hours phone support is available for urgent plant-down issues. Detailed contact information for all our worldwide locations is available on the following page.

Asia Pacific	Europe / Middle East / Africa
<p>Regional Office Phone: +603.7724.2080 asiapc@prosoft-technology.com Languages spoken: Bahasa, Chinese, English, Japanese, Korean REGIONAL TECH SUPPORT support.ap@prosoft-technology.com</p>	<p>Regional Office Phone: +33.(0)5.34.36.87.20 europe@prosoft-technology.com Languages spoken: French, English REGIONAL TECH SUPPORT support.emea@prosoft-technology.com</p>
<p>North Asia (China, Hong Kong) Phone: +86.21.5187.7337 china@prosoft-technology.com Languages spoken: Chinese, English REGIONAL TECH SUPPORT support.ap@prosoft-technology.com</p>	<p>Middle East & Africa Phone: +971.4.214.6911 mea@prosoft-technology.com Languages spoken: Hindi, English REGIONAL TECH SUPPORT support.emea@prosoft-technology.com</p>
<p>Southwest Asia (India, Pakistan) Phone: +91.98.1063.7873 india@prosoft-technology.com Languages spoken: English, Hindi, Urdu</p>	<p>North Western Europe (UK, IE, IS, DK, NO, SE) Phone: +44.(0)7415.864.902 nweurope@prosoft-technology.com Language spoken: English</p>
<p>Australasia (Australia, New Zealand) Phone: +603.7724.2080 pacific@prosoft-technology.com Language spoken: English</p>	<p>Central & Eastern Europe, Finland Phone: +48.22.250.2546 centraleurope@prosoft-technology.com Languages spoken: Polish, English, Russian & CIS Phone: +7.499.704.53.46 russia@prosoft-technology.com Languages spoken: Russian, English</p>
<p>Southeast Asia (Singapore, Indonesia, Philippines) Phone: +603.7724.2080 seasia@prosoft-technology.com Languages spoken: English, Bahasa, Tamil</p>	<p>Austria, Germany, Switzerland Phone: +33.(0)5.34.36.87.20 germany@prosoft-technology.com Language spoken: English, German</p>
<p>Northeast & Southeast Asia (Japan, Taiwan, Thailand, Vietnam, Malaysia) Phone: +603.7724.2080 neasia@prosoft-technology.com Languages spoken: English, Chinese, Japanese</p>	<p>BeNeLux, France, North Africa Phone: +33(0)5.34.36.87.27 france@prosoft-technology.com Languages spoken: French, English</p>
<p>Korea Phone: +603.7724.2080 korea@prosoft-technology.com Languages spoken: English, Korean</p>	<p>Mediterranean Countries Phone: +39.342.8651.595 italy@prosoft-technology.com Languages spoken: Italian, English, Spanish</p>

Latin America	North America
<p>Regional Office Phone: +52.222.264.1814 support.la@prosoft-technology.com Languages spoken: Spanish, English REGIONAL TECH SUPPORT support.la@prosoft-technology.com</p> <p>Brazil Phone: +55.11.5084.5178 brasil@prosoft-technology.com Languages spoken: Portuguese, English REGIONAL TECH SUPPORT support.la@prosoft-technology.com</p> <p>Mexico Phone: +52.222.264.1814 mexico@prosoft-technology.com Languages spoken: Spanish, English REGIONAL TECH SUPPORT support.la@prosoft-technology.com</p> <p>Andean Countries, Central America & Caribbean Phone: +507.6427.48.38 andean@prosoft-technology.com Languages spoken: Spanish, English</p> <p>Southern Cone (Argentina, Bolivia, Chile, Paraguay & Uruguay) Phone: +54.911.4565.8119 scone@prosoft-technology.com Languages spoken: Spanish, English</p>	<p>Regional Office Phone: +1.661.716.5100 info@prosoft-technology.com Languages spoken: English, Spanish REGIONAL TECH SUPPORT support@prosoft-technology.com</p>

9.2 Warranty Information

For complete details regarding ProSoft Technology’s TERMS & CONDITIONS OF SALE, WARRANTY, SUPPORT, SERVICE AND RETURN MATERIAL AUTHORIZATION INSTRUCTIONS please see the documents at:

www.prosoft-technology.com/legal

Documentation is subject to change without notice.

Index

[

[Backplane Configuration] • 56
[MCM Port X] • 58
[Modbus Port 1 Commands] • 64
[Module] • 55

0

00 Return Query Data • 117

3

3x Register Start • 57

4

4x Register Start • 57

A

About the MODBUS Protocol • 91
Adding the PTQ Module to the Project • 16, 29
Analyzing Data for the first application port • 81
Analyzing Data for the second application port • 81

B

Backplane Data Transfer • 92, 95
Backplane Menu • 79
Baud Rate • 60
Bit Input Offset • 61
Block 9998
 Warm Boot • 97
Building the Project • 18

C

Cable Connections • 101
Can I use the module in a hot backup system? • 123
Clearing Diagnostic Data • 76
Cmd Err Pointer • 62
Cold Boot Block (9999) - PTQ 1 to 63 • 97
Command Control Block • 94
Command Control Block Request from Processor to the Module • 96
Command Control Block Response from Module to Processor • 97
Command Count • 62
Command List Entry Errors • 86
Commands Supported by the Module • 110
Configuration Data • 107
Configuring Module Parameters • 54
Configuring the Floating Point Data Transfer • 65
Configuring the Module • 51
Configuring the Processor with Concept • 23
Configuring the Processor with ProWORX • 41
Configuring the Processor with Unity Pro • 13

Connect the PC to the ProTalk Configuration/Debug Port • 48
Connect Your PC to the Processor • 19
Connecting to the Processor with TCP/IP • 21
Contacting Technical Support • 125
Content Disclaimer • 2
Creating a New Project • 14, 26

D

Data Analyzer • 80
Data Bits • 60
Diagnostics (Function Code 08) • 117
Diagnostics and Troubleshooting • 70, 71
Direct Pass-Thru • 98
Displaying Timing Marks in the Data Analyzer • 81
Does the module work in a remote rack? • 123
Downloading the Project to the Module Using a Serial COM port • 69
Downloading the Project to the Processor • 22, 35

E

Edit the Configuration File • 55
Enable • 58
ENRON Floating Point Support • 65
Error Delay Counter • 63
Error Status Table • 86
Error/Status Block Pointer • 57
Event Command Block • 93
Event Command Block Request from Processor to the Module • 93
Event Command Block Response from Module to Processor • 94
Example and State Diagram • 117
Exiting the Program • 77

F

Failure Flag Count • 57
Float Flag • 59
Float Offset • 59
Float Start • 59
Force Multiple Coils (Function Code 15) • 118
Force Single Coil (Function Code 05) • 115
Frequently Asked Questions • 123
Functional Overview • 91
Functional Specifications • 90

G

General Specifications • 88
General Specifications - Modbus Master/Slave • 90

H

Hardware and Software Requirements • 10
Hardware Specifications • 89
Hold Offset • 61
How to Contact Us • 2

I

Information for Concept Version 2.6 Users • 24
Information for ProTalk® Product Users • 3
Initializing Output Data • 57
Inserting the 1454-9F connector • 46
Installing MDC Configuration Files • 24
Installing ProSoft Configuration Builder Software • 12
Installing the ProTalk Module in the Quantum Rack • 46, 47

K

Keystrokes • 75

L

LED Status Indicators • 72

M

Main Menu • 75
Master Command Error List Menu • 84
Min Cmd Delay • 62
Minimum Response Delay • 60
Modbus Database View Menu • 76, 77
Modbus Exception Codes • 122
Modbus Exception Responses • 120
Modbus Protocol Specification • 78, 110
Module Communication Error Codes • 86
Module Name • 56
Module Type Parameter • 55
Moving Back Through 5 Pages of Commands • 85
Moving Back Through 5 Pages of Registers • 78
Moving Forward (Skipping) Through 5 Pages of Commands • 85
Moving Forward Through 5 Pages of Registers • 78

N

Navigation • 75

O

Opening the Backplane Menu • 76
Opening the Command List Menu • 84
Opening the Database View Menu • 76
Opening the Protocol_Serial_Menu • 76
Opening the Serial Port Menu • 84
Output Offset • 61
Overview • 98

P

Package Contents • 10
Parity • 60
Pass Thru Address • 59
Pass-Thru Configuration • 98
Pass-Thru Control Blocks • 98
PC and PC Software • 11
Pinouts • 3, 46, 101
Port 1 Commands • 108
Port 1 Setup • 107
Port 2 Commands • 109

Port 2 Setup • 108
Preset Multiple Registers (Function Code 16) • 119
Preset Single Register (Function Code 06) • 116
Product Specifications • 88
Protocol • 59
Protocol Serial Menu • 83

Q

Quantum Hardware • 10

R

Read Coil Status (Function Code 01) • 111
Read Holding Registers (Function Code 03) • 113
Read Input Registers (Function Code 04) • 114
Read Input Status (Function Code 02) • 112
Read Register Count • 56
Read Register Start • 56
Reading Status Data from the Module • 86
Redisplaying the Current Page • 78, 85
Redisplaying the Menu • 79, 84
Reference • 87
Removing Timing Marks in the Data Analyzer • 81
Renaming PCB Objects • 54
Response Timeout • 62
Retry Count • 62
Returning to the Main Menu • 79, 80, 82, 84, 85
RS-232
 Modem Connection (Hardware Handshaking Required) • 102
 Null Modem Connection (Hardware Handshaking) • 102
 Null Modem Connection (No Hardware Handshaking) • 103
RS-232 Application Port(s) • 101
RS-232 Configuration/Debug Port • 101
RS-422 • 104
RS-485 and RS-422 Tip • 104
RS-485 Application Port(s) • 104
RTS Off • 60
RTS On • 60

S

Set Up the Project • 52
Setting up Data Memory in Project • 32
Setting Up the ProTalk Module • 45
Slave Address • 61
Special Functions • 93
Standard Modbus Protocol Errors • 86
Start Here • 9
Starting the Data Analyzer • 82
Status Data Block Structure • 105
Status Data Definition • 105
Stop Bits • 60
Stopping the Data Analyzer • 82
Sub-function Codes Supported • 117
Support, Service & Warranty • 125

T

- Transferring the Configuration File from The Module to the PC • 76
- Transferring the Configuration File from the PC to the Module • 77
- Type • 59

U

- Use CTS Line • 61
- Using ProSoft Configuration Builder • 52
- Using ProSoft Configuration Builder (PCB) for Diagnostics • 73
- Using the Diagnostic Window in ProSoft Configuration Builder • 73

V

- Verification and Troubleshooting • 70
- Verifying Jumper Settings • 46
- Verifying Successful Download • 37
- Viewing Backplane Diagnostic Information • 80
- Viewing Configuration Information • 80, 84
- Viewing Data in ASCII (Text) Format • 79, 81
- Viewing Data in Decimal Format • 78
- Viewing Data in Floating-Point Format • 79
- Viewing Data in Hexadecimal Format • 78, 81
- Viewing Error and Status Data • 84
- Viewing Register Pages • 78
- Viewing the Next Page of Commands • 85
- Viewing the Next Page of Registers • 78
- Viewing the Previous Page of Commands • 85
- Viewing the Previous Page of Registers • 78
- Viewing Version Information • 76, 79

W

- Warm Booting the Module • 77
- Warnings • 3
- Warranty Information • 127
- What kind of data transfer rates can I expect between the PLC and the module? • 123
- Word Input Offset • 61
- Write Register Count • 56
- Write Register Start • 56

Y

- Your Feedback Please • 2