ProSoft TECHNOLOGY

Where Automation Connects.

ProLinx®

5204SE-MNET-PDPMV1

ProLinx Gateway

Modbus TCP/IP to PROFIBUS DP-V1 Pass-Through Master

12/21/2009

USER MANUAL

Important Installation Instructions

Power, Input, and Output (I/O) wiring must be in accordance with Class I, Division 2 wiring methods, Article 501-4 (b) of the National Electrical Code, NFPA 70 for installation in the U.S., or as specified in Section 18-1J2 of the Canadian Electrical Code for installations in Canada, and in accordance with the authority having jurisdiction. The following warnings must be heeded:

A WARNING - EXPLOSION HAZARD - SUBSTITUTION OF COMPONENTS MAY IMPAIR SUITABILITY FOR CLASS I, DIV. 2;
B WARNING - EXPLOSION HAZARD - WHEN IN HAZARDOUS LOCATIONS, TURN OFF POWER BEFORE REPLACING OR WIRING MODULES
C WARNING - EXPLOSION HAZARD - DO NOT DISCONNECT EQUIPMENT UNLESS POWER HAS BEEN SWITCHED OFF OR THE AREA IS KNOWN TO BE NONHAZARDOUS.
D THIS DEVICE SHALL BE POWERED BY CLASS 2 OUTPUTS ONLY.

All ProLinx® Products

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'EQUIPMENT, COUPER LE COURANT OU S'ASSURER QUE L'EMPLACEMENT EST DÉSIGNÉ NON DANGEREUX.

Markings

UL/cUL	ISA 12.12.01 Class I, Div 2 Groups A, B, C, D
cUL	C22.2 No. 213-M1987

243333 183151

CL I Div 2 GPs A, B, C, D

Temp Code T5

II 3 G

Ex nA nL IIC T5 X

0° C <= Ta <= 60° C

II – Equipment intended for above ground use (not for use in mines).

3 – Category 3 equipment, investigated for normal operation only.

G – Equipment protected against explosive gasses.

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 the product, documentation, or support, please write or call us.

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

5204SE-MNET-PDPMV1 User Manual
12/21/2009

ProSoft Technology® Product Documentation

In an effort to conserve paper, ProSoft Technology no longer includes printed manuals with our product shipments. User Manuals, Datasheets, Sample Ladder Files, and Configuration Files are provided on the enclosed CD-ROM, and are available at no charge from our web site: www.prosoft-technology.com

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

North America: +1.661.716.5100

Asia Pacific: +603.7724.2080

Europe, Middle East, Africa: +33 (0) 5.3436.87.20

Latin America: +1.281.298.9109

Contents

1 Scope

In This Chapter

1.1 Learning Objectives

When you have completed the steps in this User Manual, you will have learned how to:

- Understand data flow through the gateway between a Schneider Electric Modicon M340 or Quantum controller using Modbus® TCP/IP, and slave devices on a PROFIBUS DP network.
- Configure the 5204SE-MNET-PDPMV1 as a PROFIBUS DP version 1 Master station to read cyclic data from and write cyclic data to PROFIBUS slave devices.
- Understand how the Unity™ Pro v 4.0 *Derived Function Blocks (DFBs)* created by ProSoft Configuration Builder (PCB) work to transfer PROFIBUS cyclic data and perform any required PROFIBUS acyclic messaging, allowing the Modicon processor to emulate a PROFIBUS DP version 1 Master.
- Observe that the 5204SE-MNET-PDPMV1 gateway is sending and receiving data on the Ethernet port and PROFIBUS Master port.

1.2 Prerequisites

To get the most benefit from this User Manual, you should have the following skills:

- **Microsoft Windows:** Install and launch programs, execute menu commands, navigate dialog boxes, and enter data.
- **PROFIBUS communication:** Configure a PROFIBUS network using ProSoft Configuration Builder (PCB) software.
- **Ethernet networking:** Connect the 5204SE-Protocol> gateway and a Schneider Electric Modicon M340 or Quantum Programmable Automation Controller (PAC) system to an Ethernet network using valid IP address, subnet mask, and default network gateway settings.
- **Hardware installation and wiring**: Install the gateway, safely connect all devices to a power source, and connect the gateway's PROFIBUS Master port and its Ethernet port to their respective networks.

1.3 System Requirements

The application described in this User Manual requires the following minimum hardware and software components:

- Schneider Electric Telemecanique Modicon PAC system with either:
  - Built-in Modbus TCP/IP Ethernet communication port

    or
  - BMXNOE0100 Ethernet Network Module (NOE)
- Schneider Electric Telemecanique Unity Pro programming software, version 4.0 or higher
- ProSoft Configuration Builder (PCB) software, version 2.1. 9.1 or higher (on the ProLinx Solutions CD-ROM, or can be downloaded from the web site)
- Supported operating systems and PC hardware required:
  - Microsoft Windows VISTA Business Edition 32

    Pentium IV, 2.4 GHz processor minimum, 3 GHz recommended

    1 GB of RAM minimum, 3 GB recommended

    8 GB of hard drive space minimum, 20 GB recommended
  - Microsoft Windows XP Professional with Service Pack 1 or 2

    Pentium IV, 1.5 GHz processor minimum, 3 GHz recommended

    512 MB of RAM minimum, 1 GB recommended

    4GB of hard drive space minimum, 8 GB recommended
- VGA or SVGA graphics adapter, 800 x 600 minimum resolution, 24-bit color resolution (True Color)
- CD-ROM drive, Windows Mouse and Keyboard
- PC with DB9Male RS-232 Serial Port (for full diagnostics using PCB), USB port (for Unity Pro), and Ethernet port (for configuration and PROFIBUS diagnostics using PCB and for Unity Pro)
- 24 vdc power supply (not provided) with at least 500 mA current capacity available to power the gateway

2 Functional Overview

In This Chapter

2.1 General Overview

Automating integration for Schneider Electric (SE) Modicon processors and maximizing ease-of-use are the hallmark design criteria behind the new ProLinx SE line of communication gateways. The first SE gateway, the 5204SE-MNET-PDPMV1 Modbus TCP/IP to PROFIBUS DP-V1 Master gateway, easily turns a Modicon M340 or Quantum Programmable Automation Controller (PAC) into a PROFIBUS DP-V1 Master. The new Application Communication Logic functions built into ProSoft Configuration Builder (PCB) automatically generate all the Unity Pro data types, variables, and logic required for the processor to perform PROFIBUS DP-V1 cyclic and acyclic communication.

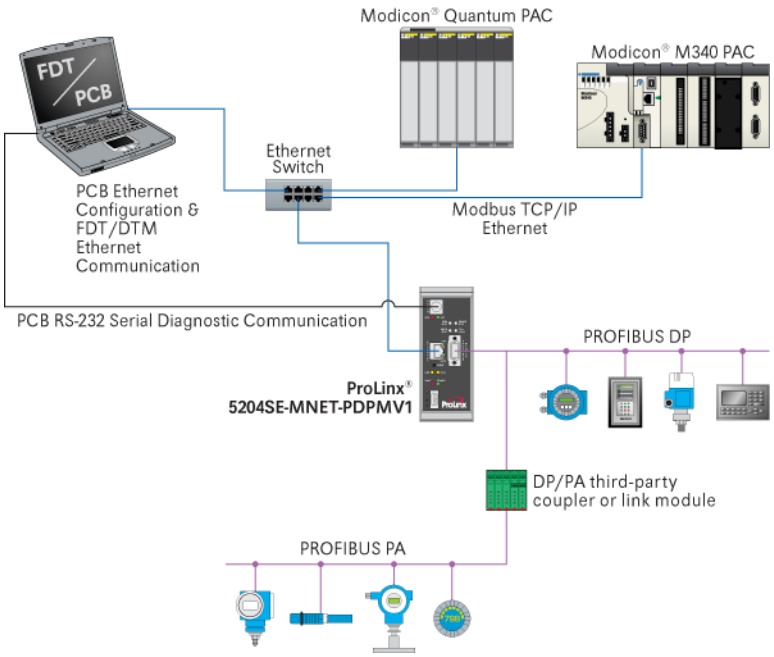
PCB automatically generates customized export files based on the gateway's PROFIBUS DP network configuration. You can import these files into Unity Pro version 4 software without modification, eliminating the need to write additional communication message logic.

Automatically-generated Derived Function Block logic also provides advanced PROFIBUS DP-V1 acyclic message pass-through capability. Acyclic messaging allows the processor to request extended slave data and diagnostics, as well as access slave-specific special functions.

If you change the configuration of your PROFIBUS DP network in ProSoft Configuration Builder, you can easily export new logic files, and then import them into an existing project.

2.2 Architecture

The following diagram shows an example network that connects a Personal Computer (PC) and a Modicon M340 or Quantum Programmable Automation Controller (PAC) to a ProLinx 5204SE-MNET-PDPMV1 gateway.



You configure the gateway using ProSoft Configuration Builder (PCB) software through an Ethernet connection. The gateway also uses its Ethernet port to support the Modbus TCP/IP protocol, allowing it to communicate with the Modicon processor, as well as other Modbus TCP/IP devices.

You can also use the Ethernet connection to manage your PROFIBUS network slaves using ProSoft Technology Field Device Tool/Communications Device Type Manager (FDT/comDTM) drivers for popular plant asset management software, such as PACT*ware*™ and Endress+Hauser FieldCare. For more information on FDT/comDTM drivers for the gateway, please refer to the ProLinx PDPMV1 Driver Manual, on the ProLinx Solutions CD-ROM.

The gateway's PROFIBUS port allows it to act as a PROFIBUS DP-V1 Master. The special Application Communication Logic functions built into PCB create all the Unity *Pro Derived Data Types (DDTs)*, *Variables*, and *Derived Function Blocks (DFBs)* needed by the Modicon processor to be able to send Modbus TCP/IP messages and have those messages turned into PROFIBUS DP-V1 cyclic I/O and acyclic mailbox messages (called *telegrams* in the PROFIBUS protocol). The gateway can also communicate with PROFIBUS PA slaves though a third-party PROFIBUS DP-to-PA Link Coupler device (not supplied by ProSoft Technology).

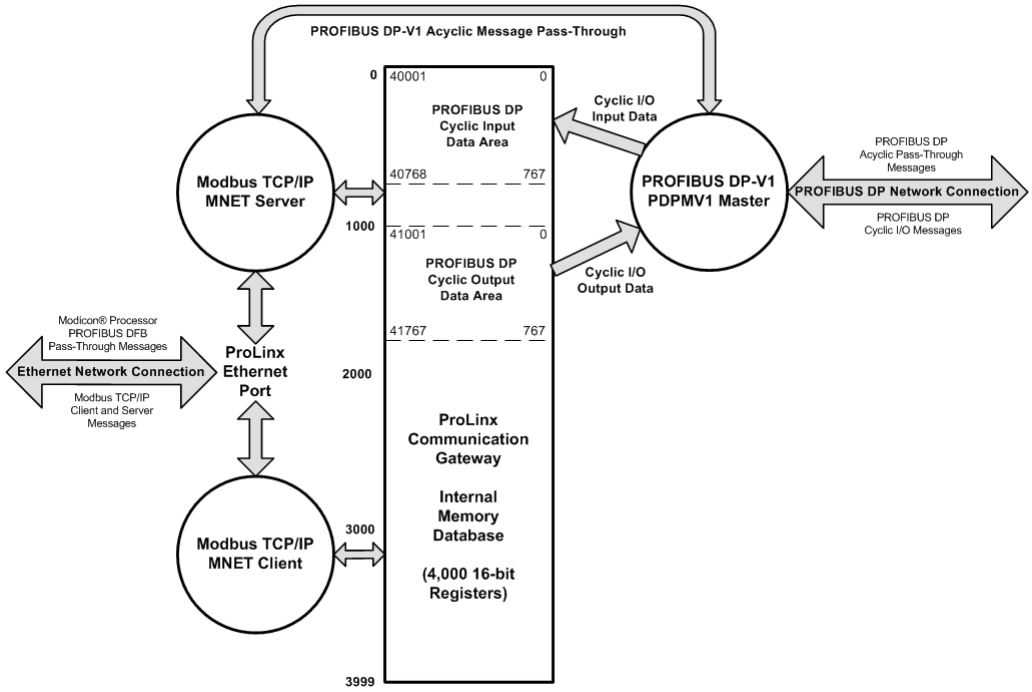
The gateway has an RS-232 serial port. ProSoft Configuration Builder can use this port to view the gateway's diagnostics and troubleshooting menus. You can also use the serial port to upgrade the gateway's firmware.

2.3 Data Flow through the Gateway

The internal database is central to the functionality of the gateway. This database is shared between all the ports on the gateway and is used as a conduit to pass information from one device on one network to one or more devices on either network supported by the gateway. This permits data from devices on one communication port or network to be viewed and controlled by devices on another port or network. In addition to data from the Master port, status and error information generated by the gateway can also be mapped into the internal database.

This special SE implementation of the 5204SE-MNET-PDPMV1 gateway uses the gateway database to store PROFIBUS DP cyclic input and output data. This means this PROFIBUS cyclic data will be available to the MNET Client (Master) and Server (Slave) drivers for use on a Modbus TCP/IP network. The SE version of the MNET Server has also been programmed to "Pass-Through" special PROFIBUS DP-V1 acyclic mailbox message commands from the Modbus TCP/IP Server directly to the PDPMV1 PROFIBUS DP-V1 Master driver for transmission on the PROFIBUS network.

This pass-through capability allows a Modicon processor using its native Modbus TCP/IP protocol to communicate directly with PROFIBUS DP-V1 slaves and receive their responses, if any. Pass-through functions bypass the gateway database, going from Modicon processor to SE MNET Server to PDPMV1 Master to PROFIBUS DP slaves and back.



Note: The normal MNET Server driver will accept and respond normally to remote Modbus TCP/IP Client requests to read or write data to any address in the gateway's internal database. However, this special SE implementation of the MNET Server is slightly different.

The SE MNET Server will accept and respond to read requests from Modbus TCP/IP clients in the same way as the normal MNET Server. However, for write requests, the SE MNET Server will accept and respond normally only if the address or addresses to be written fall in the gateway database area designated for PROFIBUS Cyclic Output Data. This area is internal addresses 1000 through 1767, which have corresponding virtual Modbus addresses of 41001 through 41768 (five-digit addressing) or 401001 through 401768 (six-digit addressing), as shown in the preceding data flow diagram.

Any write requests received by the special SE MNET Server that are outside this specific data address range will be rejected by the SE Server and an exception response containing Exception Code *"02 ILLEGAL DATA ADDRESS"* will be returned to the requesting Client.

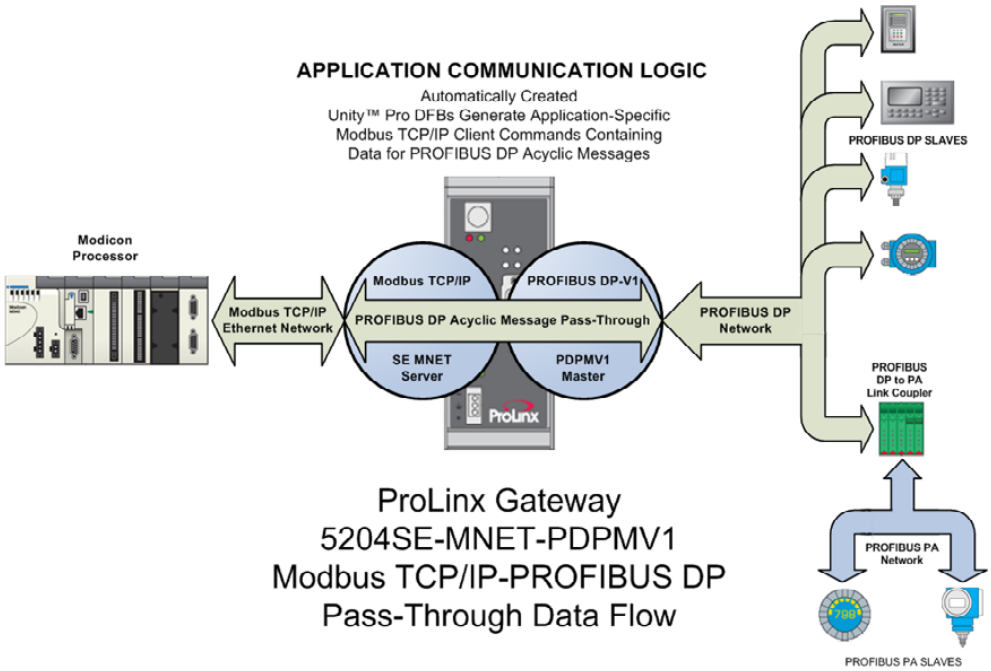
This special modification has been done to preserve the integrity of PROFIBUS Input Data by preventing external Modbus TCP/IP Clients from writing to and thereby corrupting data in this critical area, while allowing any Modbus TCP/IP Client to send data to PROFIBUS slaves by writing it to the PROFIBUS Output Data area.

2.4 PROFIBUS DP Pass-Through Data Flow

The Application Communication Logic functions built into the latest version of ProSoft Configuration Builder (PCB) will automatically create all the *Derived Data Types (DDTs)*, *Variables*, and *Derived Function Blocks (DFBs)* required by the Modicon processor to allow it to act as a PROFIBUS DP-V1 Master for acyclic mailbox message communication. PCB uses the PROFIBUS DP Master configuration to create two files that may be imported into Unity Pro version 4.0 (or higher) to create these data structures and process logic.

PROFIBUS DP-V1 acyclic communication goes beyond normal cyclic I/O data transfers by adding the capability to directly access each PROFIBUS network slave. The types of acyclic data available varies from slave to slave but generally include such data as alarm and status information, extended diagnostic information, extended process data information, and device-specific special commands.

The following illustration shows the basics of acyclic mailbox message data flow from a Modicon processor, through the two gateway drivers, out to PROFIBUS slaves, with responses (if any) returned to the Modicon.



The *DFBs* create Modbus TCP/IP Client (Master) commands, which are transmitted to the ProLinx MNET server (slave) driver. The MNET server recognizes these special Modbus TCP/IP commands as PROFIBUS DP Acyclic mailbox messages, strips out the PROFIBUS DP-V1 acyclic mailbox message parameters contained in the Modbus TCP/IP message, and passes those PROFIBUS message parameters through to the ProLinx PDPMV1 Master driver for transmission to the PROFIBUS slave or slaves indicated in the message. Any response message returned by the a slave to the PDPMV1 Master will be automatically repackaged as a Modbus TCP/IP message and returned to the Modicon processor, where the imported *DFBs* will process it.

This process allows a Modicon processor, using its native Modbus TCP/IP communication ability, to act as a PROFIBUS DP-V1 Master on a PROFIBUS network. By using a third-party PROFIBUS DP to PROFIBUS PA Link Coupler (not supplied by ProSoft Technology), you can extend these PROFIBUS capabilities even further and communicate with PROFIBUS PA slaves.

2.5 Cyclic Polling and Acyclic Messaging Control Logic

The Application Communication Logic functions of ProSoft Configuration Builder (PCB) automatically create *Variables, DDTs* and *DFBs* that have been customized to match the PROFIBUS part of your PCB application. These automatically-created structures and logic give you all the basic building blocks you need to create an effective Modbus TCP/IP to PROFIBUS DP communication application.

To complete your PROFIBUS communication application, all you will need to do is to create your own customized control and sequencing logic to call the PCB-generated *DFBs* in a logical, controlled manner. It will be your sequencing logic that will decide when to call for PROFIBUS cyclic I/O data, status, and diagnostic updates. Your logic will also decide if and when to trigger any acyclic messages you may need to send.

To help you create your control and sequencing logic, here are a few principles to keep in mind.

1 Once you have successfully created and downloaded your PROFIBUS configuration to the 5204SE-MNET-PDPMV1 gateway, the PROFIBUS DP-V1 Master driver will automatically begin and maintain normal PROFIBUS cyclic I/O communications. PROFIBUS cyclic inputs and outputs will constantly be updated on the PROFIBUS network with no action being required from the processor. This means that PROFIBUS cyclic data transfers are asynchronous and separate from whatever communication the processor may or may not be doing between itself and the 5204SE-MNET-PDPMV1 gateway.

2 In order for the processor to 'see' any of the data being received from the PROFIBUS slaves or to send any data to the PROFIBUS slaves, your control and sequencing logic must initiate Modbus TCP/IP Client command messages from the processor to the gateway by using the binary trigger variables provided for each *DFB*. This requirement applies equally for the cyclic *DFBs* and for the acyclic *DFBs*. The main difference will be that the DFBs to update cyclic data will most likely be triggered much more often than the acyclic *DFBs* are triggered. For additional details, see Sample Control and Sequencing Logic for Cyclic Data Polling (page 113).

3 The basic communication cycle between the processor and the gateway is:

- User-created logic in the processor sends a Modbus TCP/IP command message to the gateway by triggering one of the fourteen (14) provided *DFBs*.
- The MNET Server driver on the gateway receives the command.
- The MNET Server driver processes the command.
- The MNET Server driver returns a Modbus TCP/IP response to the processor.
- Some commands cause data to be returned, such as read commands. Some commands, such as write commands, return only an acknowledgement that the command was received and executed. Commands sometimes fail. Any data or error response to a command returned by the MNET Server will be available in the provided *Variables* or *DDTs* after being placed there by the triggered *DFB* that initiated the process cycle. User-created logic in the processor must process data or errors received in the Modbus TCP/IP response, if any.
- If no Modbus TCP/IP response is received within the time value specified in a *Timeout* variable, the triggered *DFB* will set a *Message Error* bit flag to indicate the message sequence failed and should be retried by triggering a new message cycle.

4 In cases where PROFIBUS cyclic I/O data, general gateway status, or standard PROFIBUS slave diagnostic data are concerned, these read or write requests from the processor are handled internally in the gateway and are processed asynchronously from any PROFIBUS DP Master processes that might also be running in the gateway. This means these cyclic requests tend to be responded to much more quickly than requests involving acyclic messages that must be "passed-through" to the PROFIBUS Master for execution before a Modbus TCP/IP response can be created and returned to the processor.

5 In cases where PROFIBUS DP-V1 acyclic messages are concerned, these read or write requests require a longer, more involved process cycle, synchronized with actions on the PROFIBUS DP-V1 Master side of the module. When the processor sends a Modbus TCP/IP read or write request, using an acyclic message *DFB*:

- User-created logic in the processor sends a Modbus TCP/IP command message containing the data needed for the PROFIBUS DP-V1 acyclic message to the gateway by triggering one of the ten (10) provided acyclic *DFBs*.
- The gateway MNET Server must process the incoming command.

- The MNET Server must pass the acyclic message and any associated data through to the PROFIBUS DP-V1 Master.
- The PROFIBUS Master must insert this command in between normal slave data polling messages (send an acyclic message to a particular slave or group of slaves.)
- The PROFIBUS Master must receive a response from the addressed PROFIBUS slave.
- The PROFIBUS Master must return any PROFIBUS slave response data to the MNET Server.
- The MNET Server must create and return a Modbus TCP/IP response to the processor containing the data, if any, from the PROFIBUS slave or slaves.
- Some acyclic messages cause data to be returned. Some acyclic messages return no data. Acyclic messages sometimes fail. Communication failures could happen on either or both the Modbus TCP/IP protocol or the PROFIBUS protocol. Any data or error response to an acyclic message returned by the PROFIBUS Master or the MNET Server will be available in the provided *Variables* or *DDTs* after being placed there by the triggered *DFB* that initiated the process cycle. User-created logic in the processor must process data received in the PROFIBUS DP or Modbus TCP/IP response, if any.
- If no Modbus TCP/IP response is received within the time value specified in a *Timeout* variable, the triggered *DFB* will set a *Message Error* bit flag to indicate the message sequence failed and should be retried by triggering a new message cycle.

You can see from the amount of processing involved that it will take somewhat more time for the gateway to respond to acyclic message commands than it will take to respond to cyclic I/O, status and diagnostics requests.

6 Due to the nature of the communication routines used in the processor and the Unity Pro programming language, only one Modbus TCP/IP command can be "active" or "in process" at any one time. All the provided *DFBs* have internal checks built in to prevent more than one at a time from being active. Therefore, any control or sequencing logic you create must respect and accommodate this processor/language limitation. Part of the accommodations you will have to make is to allow for the differing amounts of time it takes to process cyclic I/O and status commands as well as the increased time it takes to process acyclic messages.

7 All provided DFBs have binary status bits available which can be monitored by your control and sequencing logic to be sure you are not trying to activate more than one *DFB*-created Modbus TCP/IP message at a time. There is a *Message Done* bit to indicate the communication cycle completed successfully; and, there is a *Message Error* bit to indicate the communication cycle did not complete successfully.

3 Procedures

In This Chapter

3.1 ProLinx Reference Guide

The *ProLinx Reference Guide* on the ProSoft Solutions CD-ROM provides detailed information on the entire range of ProLinx modules. If you have any questions that are not answered in the MNET-PDPMV1 User Manual, please refer to the *ProLinx Reference Guide*.

3.2 Install ProSoft Configuration Builder Software

You must install the ProSoft Configuration Builder (PCB) software to configure the gateway. You can always get the newest version of ProSoft Configuration Builder from the ProSoft Technology web site.

To install ProSoft Configuration Builder from the ProSoft Web Site

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

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

To install ProSoft Configuration Builder from the Product CD-ROM

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

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

3.2.1 Using the Online Help

Most of the information needed to help you use ProSoft Configuration Builder is provided in a Help System that is always available whenever you are running ProSoft Configuration Builder. The Help System does not require an Internet connection.

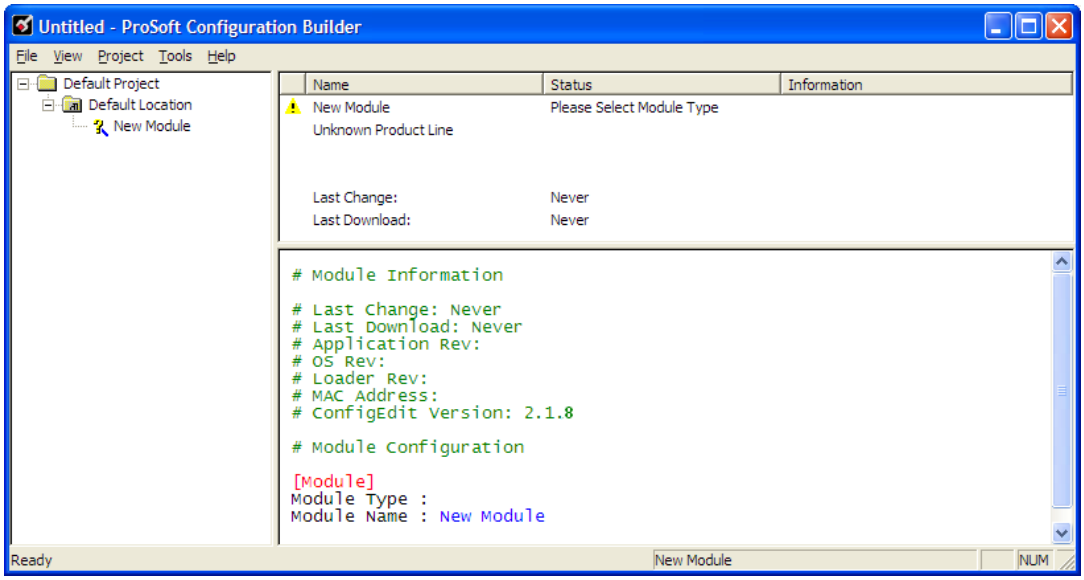
To view the help pages, start ProSoft Configuration Builder, open the **HELP** menu, and then choose **CONTENTS.**

3.2.2 Using ProSoft Configuration Builder

ProSoft Configuration Builder (PCB) provides a quick and easy way to manage gateway 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.

3.2.3 Set Up the Project - SE-MNET-PDPMV1

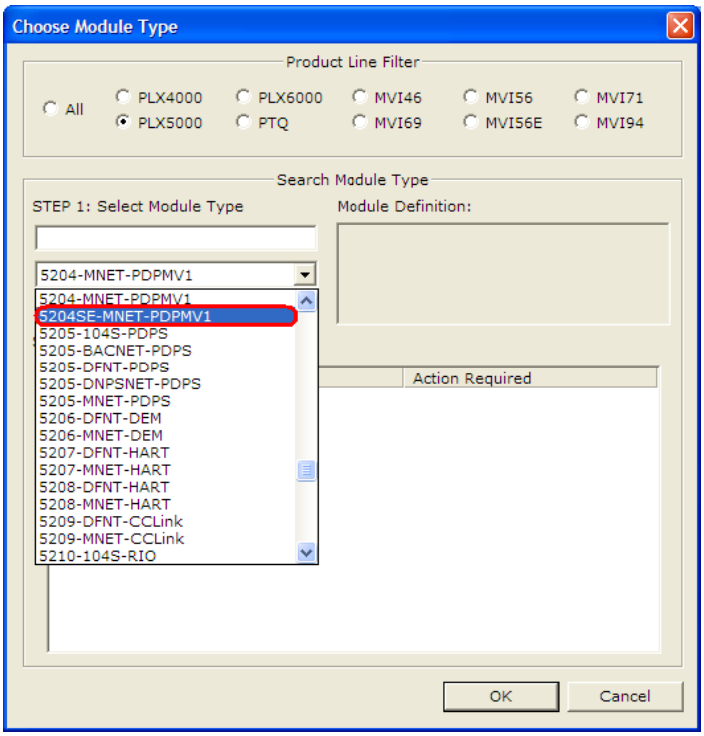
To begin, start ProSoft Configuration Builder. If you have used other Windows configuration tools before, you will find the screen layout familiar. ProSoft Configuration Builder's window consists of a tree view on the left, an information pane, and a configuration pane on the right side of the window. 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 5204SE-MNET-PDPMV1 module to the project.

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

2 On the shortcut menu, choose **CHOOSE MODULE TYPE**. This action opens the **CHOOSE MODULE TYPE** dialog box.

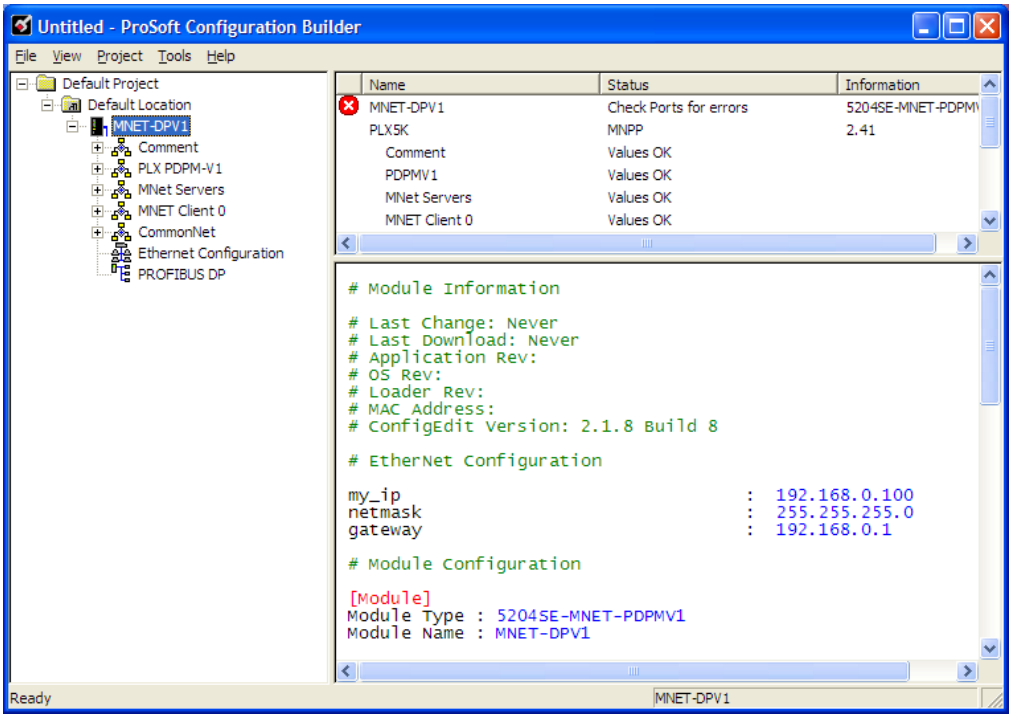


3 In the **PRODUCT LINE FILTER** area of the dialog box, select **5204SE.** In the **SELECT MODULE TYPE** dropdown list, select **5204SE-MNET-PDPMV1**, and then click **OK** to save your settings and return to the **PROSOFT CONFIGURATION BUILDER** window.

Important: Be sure to pick the 5204SE-MNET-PDPMV1 module from the list. The very similar 5204-MNET-PDPMV1 does not support the special Application Communication Logic functions available in the 5204SE-MNET-PDPMV1. These functions are required to integrate the 5204SE-MNET-PDPMV1 with the Modicon processor.

3.3 Set Module Parameters

Notice that the contents of the information pane and the configuration pane changed when you added the 5204SE-MNET-PDPMV1 module to the project.



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

To rename an object:

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

3.3.1 To Configure Module Parameters

1. Click on the plus sign next to the icon to expand gateway information.
2. Double-click the icon to open the **EDIT** dialog box.
3. To edit a parameter, select the parameter in the left pane and make your changes in the right pane.
4. Click **OK** to save your changes.

3.3.2 Printing a Configuration File

1. Select the **MODULE** icon, and then click the right mouse button to open a shortcut menu.
2. On the shortcut menu, choose **VIEW CONFIGURATION.** This action opens the **VIEW CONFIGURATION** window.
3. On the **VIEW CONFIGURATION** window, open the **FILE** menu, and choose **PRINT.** This action opens the **PRINT** dialog box.
4. On the **PRINT** dialog box, choose the printer to use from the dropdown list, select printing options, and then click **OK.**

3.4 Configure the Gateway

To configure the gateway for your application, follow these steps:

1. Configure the PROFIBUS DP Master Setting and setup all PROFIBUS DP network slave devices (page 23, page 24)
2. Configure the PROFIBUS Master DP-V1 gateway settings (page 32)
3. Configure the MNET Server Settings (Optional - default settings can be used without modification) (page 38)
4. Configure the MNET Client Settings (Optional - default settings can be used without modification) (page 39)
5. Configure the Ethernet Settings (page 40)
6. Download the Project to the gateway (page 43).

3.4.1 Configure the PROFIBUS DP Network

To configure your PROFIBUS DP network you must perform four tasks:

1. Install any PROFIBUS slave-specific device configuration files, typically called .GSD files (page 23).
2. Configure the ProLinx PROFIBUS DP Master (page 23, page 24).
3. Configure the PROFIBUS Slaves.
4. Print the Unity Passthru Memory Map (page 30).

Install the GSD Files

ProSoft Configuration Builder (PCB) uses PROFIBUS slave device definition files (GSD files) to obtain basic configuration information about the PROFIBUS slaves you add to the network. The GSD configuration files identify the slave's capabilities so that the 5204SE-MNET-PDPMV1 can communicate with it correctly.

Follow these steps to install the GSD file or files for your slave device or devices.

Tip: GSD configuration files for popular PROFIBUS slaves and ProSoft Technology solutions are included with PCB. Before installing GSD files, browse the list of available slaves in the Tree View window to see if GSD files for your slave are already installed.
GSD files are often both model number specific as well as model revision specific. Just because you may have an older GSD file from a manufacturer for the particular make and model of your slave device does not guarantee it will work for a newer revision of that device. Be sure you obtain from the device manufacturer the correct GSD file or files for your PROFIBUS slave or slaves.

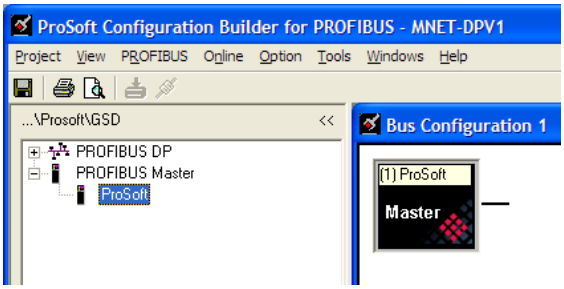
To install GSD files manually:

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

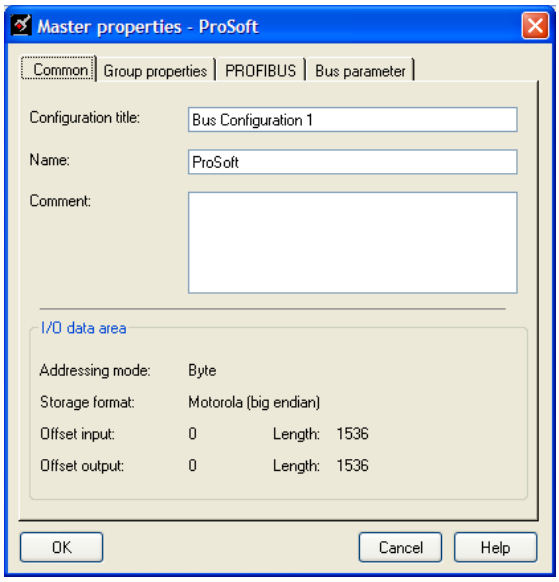
3 Open the **TOOLS** menu, and then choose **INSTALL NEW GS* FILE.** This action opens a dialog box that allows you to browse for the location of the GSD configuration files to install. (Depending on the device and language used in the file, the actual extension may be ".GSD", ".GSE", ".GSS", of other combinations; hence the generic reference to ".GS*" files, where "*" is a wildcard that stands for any letter.)
4 Choose the file to install, and then click **OPEN.** If the file already exists in the configuration file path, you will be prompted to overwrite the file.
5 You will be prompted to associate the GSD configuration file with a bitmap image of the slave device. Use the **FILE / OPEN** dialog box to browse for the location of the image file to use. If you have no device-specific bitmap file, you may **CANCEL** the bitmap upload, and a generic device icon will be used in the Bus Configuration window for this slave device.

Configure the PROFIBUS DP Master

1 In ProSoft Configuration Builder tree view, click [+] to expand the module tree, and then double-click the **PROFIBUS DP** icon. This action opens the *PDPMV1 PROFIBUS Master Setup* dialog box.
2 On the *PDPMV1 PROFIBUS Master Setup* dialog box, click the **CONFIGURE PROFIBUS** button. This action opens the *ProSoft Configuration Builder for PROFIBUS* application.
3 Click [+] to expand the **PROFIBUS MASTER** tree.
4 Drag the **PROSOFT** *PROFIBUS Master* icon into the *Bus Configuration* window.



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



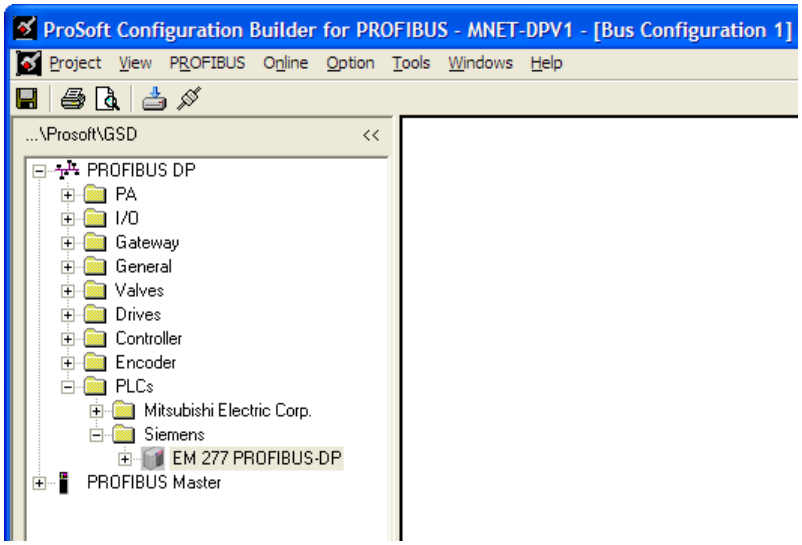
Configure the PROFIBUS Slaves

Important: The GSD file for this example is not included on the ProLinx Solutions CD-ROM, and is used for illustrative purposes only. You can download a variety of example GSD files from the PROFIBUS Trade Organization web site at www.profibus.org, or from the manufacturer's web site for your PROFIBUS slaves.

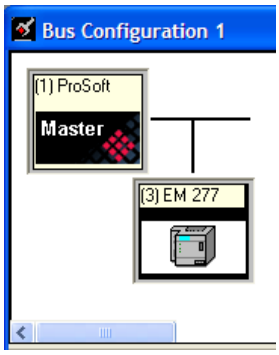
The following steps, describe how to add and configure a Siemens EM 277 I/O chassis to the PROFIBUS network. The configuration information (.GSD file) for this device must be installed according to the procedure found in Install the GSD Files (page 23). Most other PROFIBUS Slaves can be configured in a similar manner.

1 In *ProSoft Configuration Builder for PROFIBUS*, click the plus sign [+] to expand the **PROFIBUS DP** tree.

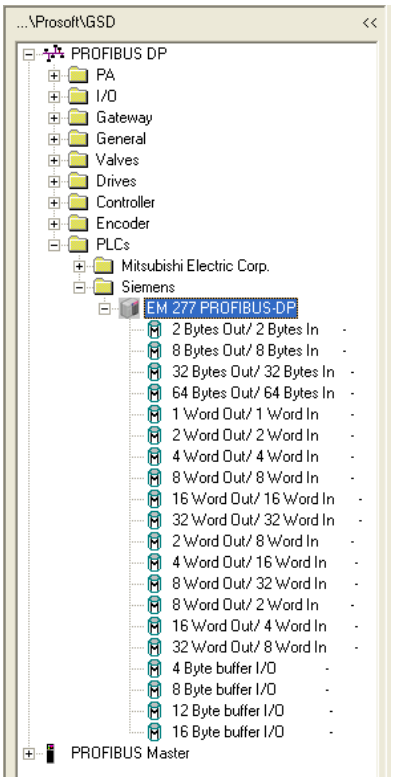
2 Navigate to the folder containing the type of slave device to add (**PLCs/SIEMENS/EM 277**, in this example), and then click the plus sign **[+]** to expand the folder.



3 Click on the **EM 277 PROFIBUS-DP** icon in the tree view and drag and drop the icon into the Bus Configuration view. This action adds the slave device and connects it to the Master in a network relationship.

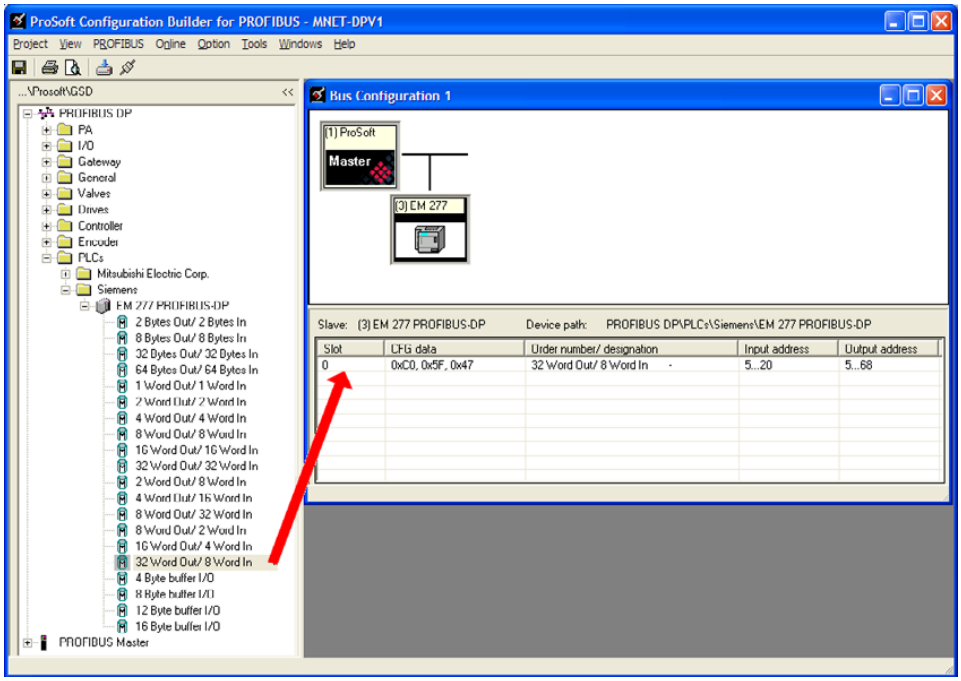


4 In the tree view, click the plus sign **[+]** to expand the slave device you added. This action opens a list of device configuration values. The following illustration shows the possible input/output configuration values for a Siemens EM 277. The selections available for other devices may be different, so you should review the specifications for the product you are installing in order to determine the correct values to use.



5 Drag the input and output parameters to the slot location grid (*Subscriber List*) below the *Bus Configuration* window. The slot view displays the slot number, configuration data, and input and output addresses. The PROFIBUS DP Master uses this information to identify and communicate with individual slaves on the network.

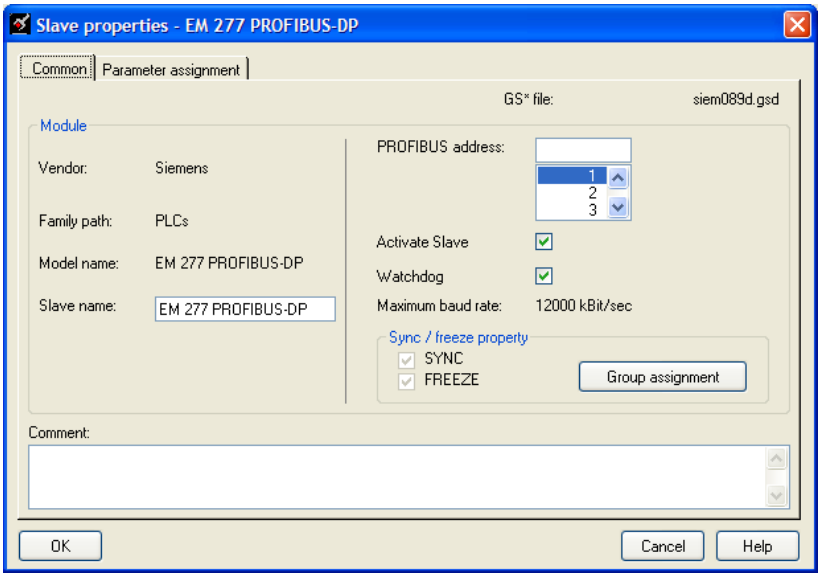
For this example, we will configure 8 words of input and 32 words of output. These input and output words are assigned to addresses within the gateway's internal database.



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

Tip: To make it easier to view the data from individual slaves, you can create a spreadsheet with all added slaves and input and output data offsets, or you can view and print the data map.

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



ProSoft Configuration Builder automatically assigns a PROFIBUS address to each new slave. The slave address assignment begins at address 3 for the first slave added to the network (addresses 0, 1, and 2 are reserved for use with PROFIBUS Masters), and is incremented by 1 for each new slave added to the network. You may, however, assign any address, 0-125 to any Master or slave node as long as you do not assign the same address to more than one device. You can change the address in the **COMMON** tab of the *Slave properties* dialog box. ProSoft Configuration Builder will not allow you to assign a PROFIBUS address that is already in use by another device on this network.

Leave the remaining settings unchanged for now, and click **OK** to close the *Slave properties* dialog box.

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

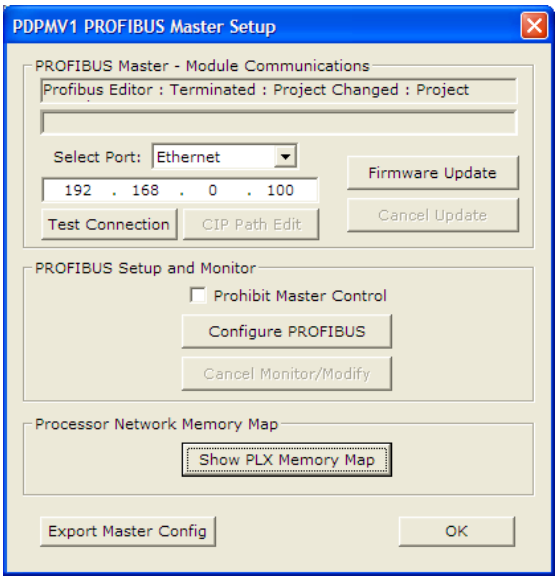
8 When you are finished adding slaves, open the **PROJECT** menu and choose **EXIT.** Click **YES** to save the project and return to the PROFIBUS Master Setup dialog box.

Print the Unity Passthru Memory Map

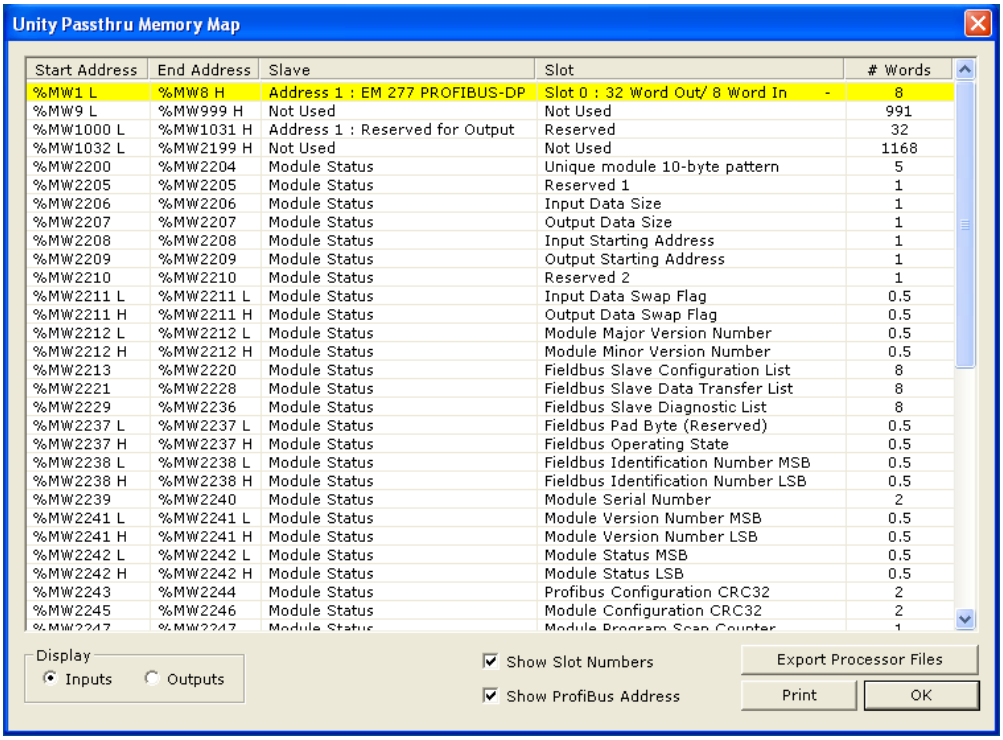
The Unity Passthru Memory Map dialog box uses the information about your PROFIBUS Master and slave configuration to display where the data may appear in your Modicon processor's *Memory Word (%MW)* data registers. You need to know where in the Modicon processor *%MW* area you want the PROFIBUS data to appear before you will be able to input configuration parameters that will make this Memory Map display valid addresses.

To view or print the ProLinx Memory Map

1 On the *PDPMV1 PROFIBUS Master Setup* dialog box, click the **SHOW PLX MEMORY MAP** button, near the bottom of the window.



This action opens *Unity Passthru Memory Map* window.



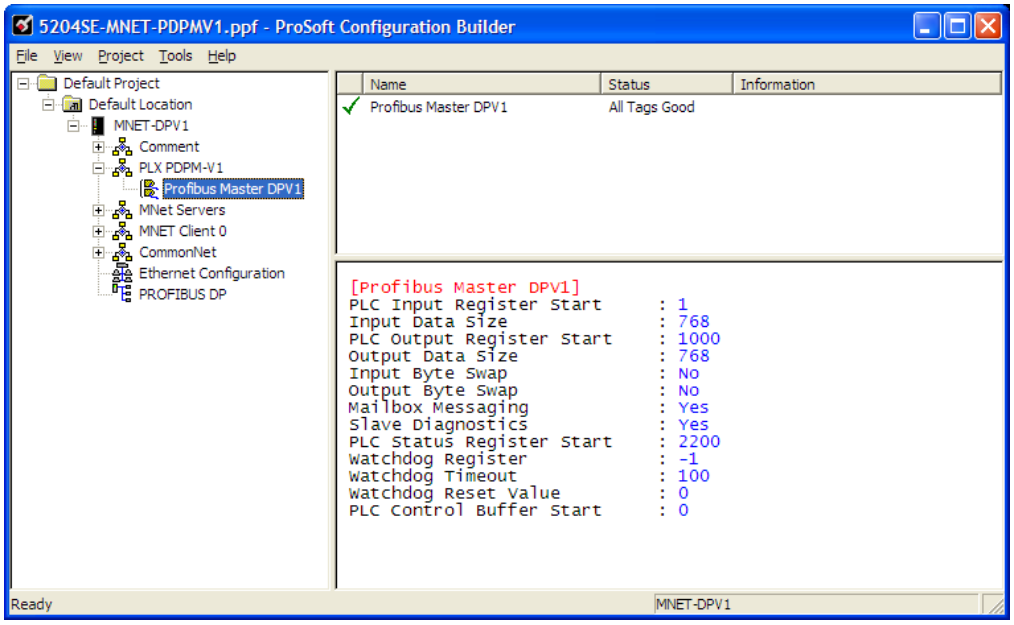
2 Notice that there are two radio buttons in the *Display* area of the dialog box to select Inputs or Outputs. These Input and Output maps correspond to the Input and Output data you configured for the PROFIBUS Slaves (page 25). Notice also that there are check boxes to display **SLOT NUMBERS** and **PROFIBUS ADDRESSES.**

3 Click **PRINT** to print the maps for reference. Note that you must do this once for the input map and once for the output map. Use the *Display* area radio buttons to select which map you wish to print.

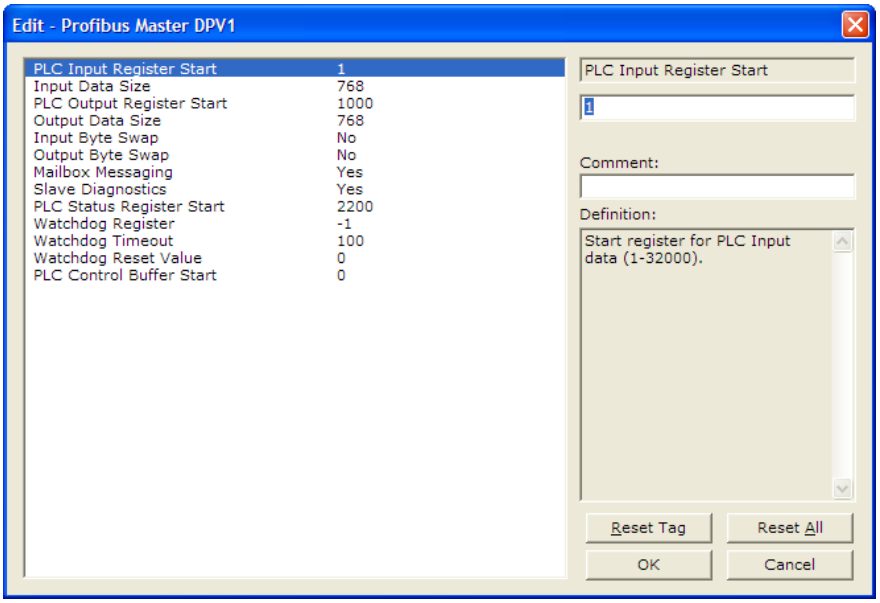
4 When you have finished printing the ProLinx memory maps, click **OK** to close the dialog box. Click **OK** again to close the *Master Setup* dialog box.

3.4.2 [Profibus Master DPV1]

1 Click the plus sign **[+]** next to the module to expand the module tree, and then expand the **PLX PDPM-V1** tree.



2 Double-click the **PROFIBUS MASTER DPV1** object. This action opens the *Edit* dialog box.



PLC Input Start Register

The value you enter here will be used by PCB to assign Modicon processor's *%MW* memory addresses to the *Unity Passthru Input Memory Map*.

Valid values are *%MW* address 00001 to highest possible address (32000 minus **INPUT DATA SIZE** value, depending on the processor's memory configuration.)

Set this parameter to the Unity Pro *Memory Word (%MW) address* in the Modicon processor that will hold PROFIBUS Slave device cyclic input data. This start address and the **INPUT DATA SIZE** value will determine the area of processor memory to reserve exclusively for incoming PROFIBUS cyclic input data.

Caution: To avoid corruption of PROFIBUS data, you must make sure that this memory area is only read from and never written into by other parts of your application logic. Refer to the Unity Passthru Memory Map for the addresses to use (page 30).

PROFIBUS cyclic input data will always be stored in the gateway's memory addresses 0000 through 0767. These are 16-bit word-sized registers, which can hold two (2) 8-bit bytes per register, for a total PROFIBUS cyclic input capacity of 1536 bytes.

Input Data Size

0 to 768

Total number of *PROFIBUS Input Words* (one word equals two bytes) from all PROFIBUS slaves. These *Input Words* will be the data received from slave devices on the PROFIBUS network.

PLC Output Register Start

The value you enter here will be used by PCB to assign the processor's *%MW* memory addresses to the *Unity Passthru Output Memory Map*.

Valid values are *%MW* address 00001 to highest possible address (32000 minus **OUTPUT DATA SIZE** value, depending on processor memory configuration.)

Set this parameter to the Unity Pro *Memory Word (%MW)* address in the processor you intend to use for holding PROFIBUS slave device cyclic output data. This start address and the **OUTPUT DATA SIZE** value will determine the area of processor memory to reserve exclusively for outgoing PROFIBUS cyclic output data. Be sure other parts of your application logic put data into this area only if it should be sent to PROFIBUS slaves and be sure to put the data into the correct part of this data area, so that the data goes to the correct slave.

For more information on using these memory registers, refer to Print the Unity Passthru Memory Map (page 30).

You will need to create your own custom control and sequencing logic to place data into the proper places in the processor memory in this address range before you send the data to the gateway with the *WriteCyclicData DFB*.

PROFIBUS cyclic output data will always be stored in the gateway's memory addresses 1000 through 1767. These are 16-bit word-sized registers, which can hold two (2) 8-bit bytes per register, for a total PROFIBUS cyclic output capacity of 1536 bytes.


Warning: Inadvertant overwriting of data in the PROFIBUS memory areas could cause unexpected behavior on your PROFIBUS DP network, leading to unintended equipment operation. Such unintended operation could cause injury to personnel or damage to equipment.

Output Data Size

0 to 768

Total number of *PROFIBUS Output Words* (one word equals two bytes) to be sent to all PROFIBUS Slaves. These *Output Words* will be the data sent to Slave devices on the PROFIBUS network.

Input Byte Swap

Yes or No

This parameter determines if the bytes in the *PROFIBUS Input Data* area are swapped before being stored in the gateway memory database. If the parameter is set to **NO**, no swapping will be applied. If the parameter is set to **YES**, the order of bytes in each word will be swapped before being stored in memory.

Example:

- With Input Byte Swap set to **NO**, incoming order is unchanged - ABCDEF
- With Input Byte Swap set to **YES**, each byte pair is swapped - BADCFE

Output Byte Swap

Yes or No

This parameter determines if the bytes in the *PROFIBUS Output Data* area are swapped before being transmitted to slaves on the PROFIBUS network. If the parameter is set to **NO**, no swapping will be applied. If the parameter is set to **YES**, the order of bytes in each word will be swapped before being transmitted.

Example:

- With Output Byte Swap set to **NO**, outgoing output order is unchanged - ABCDEF
- With Output Byte Swap set to **YES**, each output byte pair is swapped - BADCFE

Mailbox Messaging

Yes or No

This parameter controls whether or not special files will be created for import into your Unity Pro project for PROFIBUS acyclic messaging support.

Set the parameter to **YES** if you plan to use the special acyclic messaging capabilities of PROFIBUS DP version 1. When set to **YES**, the gateway will use your PROFIBUS DP Master/Slave configuration to create the required *Derived Function Blocks (DFBs)*, *Derived Data Types (DDTs)*, and *Variables* needed for processor application logic.

If your PROFIBUS application uses only cyclic I/O data (PROFIBUS Input and Output Data) and you will not be using any acyclic messaging, set this parameter to **NO**, so that unnecessary files will not be created.

Slave Diagnostics

Yes or No

If the parameter is set to **YES**, then the gateway will poll data from all slaves on the PROFIBUS network and place it in the gateway database addresses1800 through 2177. If it is set to **NO**, then the gateway will not poll any slave diagnostics data over the network and the gateway database will show zeros in this area.

Each PROFIBUS slave can report six bytes (three words) of standard diagnostic data. A total of 378 words (756 bytes; 3 words or 6 bytes times 126 possible nodes) will have to be reserved to hold this *Slave Diagnostic Data* in processor %*MW* memory to use this feature.

PLC Status Data Register Start

Modicon %MW address 00001 to highest possible address (32000 minus 152).

Set this parameter to the Unity Pro Memory Word (%MW) address in the processor that you intend to use for holding general gateway status data. This data consists of 76 words (152 bytes) of gateway status registers, error counters, and general gateway diagnostic information. This start address will determine the area of the processor memory to reserve exclusively for incoming gateway status data. Be sure this memory area is only read from and never written into by other parts of your application logic to avoid corruption of this status data.

General gateway status data will always be stored in the gateway's memory addresses 2200 through 2275. For more details about the contents of these registers, refer to DFB Get Module Status (page 109).

Watchdog Register

The Watchdog function allows the gateway to monitor a database register, the *Watchdog Register*, to check for loss of communication with the non-PROFIBUS communication protocol. If this function is used, the other gateway protocol is expected to change the value in the *Watchdog Register* at an interval less than the amount of time specified in the *Watchdog Timeout* parameter. If the value in the *Watchdog Register* does not change within this amount of time, a communication loss is assumed and the Watchdog function will set the PROFIBUS outputs to the default value specified in the *Watchdog Reset Value* parameter. To disable this function, set this parameter to a value of -1.

Watchdog Timeout

Sets the period of time (in 0.1s increments) for the gateway to wait for communication loss detection. For example, set this parameter to 100 to set a waiting period of 10 seconds. To disable this function, set this parameter to a value of -1.

Watchdog Reset Value

Sets the value that will be sent to the PROFIBUS output byte registers upon communication loss as detected by the Watchdog function. To disable this function, set this parameter to a value of -1.

PLC Control Buffer Start

0 for M340 processors only

1 to 32000 for Quantum processors only

This parameter serves two purposes. First, it tells the gateway what kind of processor will be used. This affects what kind of export files are created by the Application Communication Logic (ACL) features of the gateway. If set to zero (0), export files will be created for M340 processors. If set to a non-zero value in the range of 1 to 32000, export files will be created for Quantum processors. Second, it tells the gateway what addresses to show in the *Unity Passthru Memory Map* for Quantum communication control and data buffers when a Quantum processor is used.

Set this parameter to zero (0) when the module will be used with an M340 processor, which does not require special communication control and data buffer space be reserved in processor memory. No Control Buffer information will be shown in the *Unity Passthru Memory Map* when this parameter is set to zero (0).

Set this parameter to a memory address (1 to 32000) to reserve memory space in a Quantum processor for communication control and data buffers. To send and receive Modbus TCP/IP messages with a Quantum processor, an area of processor memory must be set aside and reserved for exclusive use as communication control and data buffer space. The amount of memory that must be reserved varies with the amount of communication tasks in the Unity Pro program. The address you enter in this parameter will be the starting address of these buffers in *%MW* processor memory.

How much memory to reserve depends on whether or not you plan to use the PROFIBUS Acyclic Messaging (Mailbox Messaging) capabilities of the gateway. The communication control and data buffers are related to the special *Defined Function Blocks (DFBs)* created for you by the gateway's ACL. The four (4) basic *DFBs* used to transfer PROFIBUS Cyclic data, general module status, and standard slave diagnostic data require 436, 16-bit registers of communication control and data buffer space. This is the least amount of space that will need to be reserved. The ten (10) DFBs used to accomplish Mailbox Messaging require as much as an additional 1635 registers, for a total of up to 2071 registers, that should be reserved for these special function buffers.

How you set the **MAILBOX MESSAGING** parameter will determine how much memory will be shown for Control Buffers in the *Unity Passthru Memory Map*. Setting **MAILBOX MESSAGING** to **NO** will show 436 registers reserved for Control Buffers. Setting **MAILBOX MESSAGING** to **YES** will show 2071 registers reserved for Control Buffers.

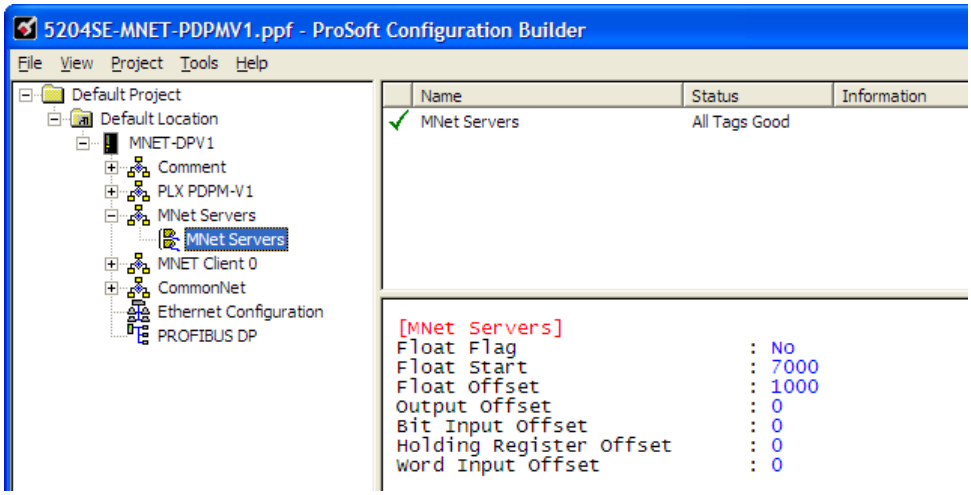
3.4.3 Configure the MNET Server Settings (Optional)

All 4000 of the gateway's 16-bit internal memory registers can be read by remote Modbus TCP/IP Clients. The gateway database registers can be read by remote Modbus TCP/IP Clients (Masters) using virtual Modbus addresses 40001 through 44000 (for five-digit addressing) or 400001 through 404000 (for six digit addressing).

However, not all 4000 registers can be written with Modbus TCP/IP write requests from remote Clients. To maintain compatibility with the PROFIBUS DP-V1 Master protocol on this gateway, the MNET Server accepts write requests from Clients only if the register address and range (count) used in the command will place data in the PROFIBUS Output data area. This is the gateway database area that holds data which will be sent to Slaves on the PROFIBUS DP network. The allowable range of Modbus TCP/IP addresses for acceptable write requests is 41001 through 41768 (for five-digit addressing) or 401001 through 401768 (for six-digit addressing). This address range is equivalent to the gateway's database registers 1000 through 1767.

Care must be taken if remote Clients are allowed to write to this data area because, once data values are written into those registers, those data values will be passed to PROFIBUS slaves. Which slave or slaves receive the data will be determined by the PROFIBUS Master/slave configuration.

The default values contained in *MNET Servers* section of PCB should work for most applications and should not need to be modified. If you wish to view the MNET Server settings, you may do so by expanding the *MNet Servers* section of the gateway configuration tree.



3.4.4 Configure the MNET Client (Optional)

MNET Client Commands can affect the data contained in any of the gateway's 4000-register internal memory database. Be aware that the processor logic will also be reading from and writing to the PROFIBUS areas of the gateway's memory through the MNET Servers. If you decide to use Client commands in your application, be careful that you do not interfere with or overwrite data in the PROFIBUS areas of the gateway unless your application requires it. For more details on gateway memory areas, see Memory Maps (page 30).

Warning: Inadvertant overwriting of data in the PROFIBUS memory areas could cause unexpected behavior on your PROFIBUS DP network, leading to unintended equipment operation. Such unintended operation could cause injury to personnel or damage to equipment.

Refer to the MNET Driver Manual, on the ProLinx Solutions CD-ROM, for more information about the MNET Client, its functions and capabilities.

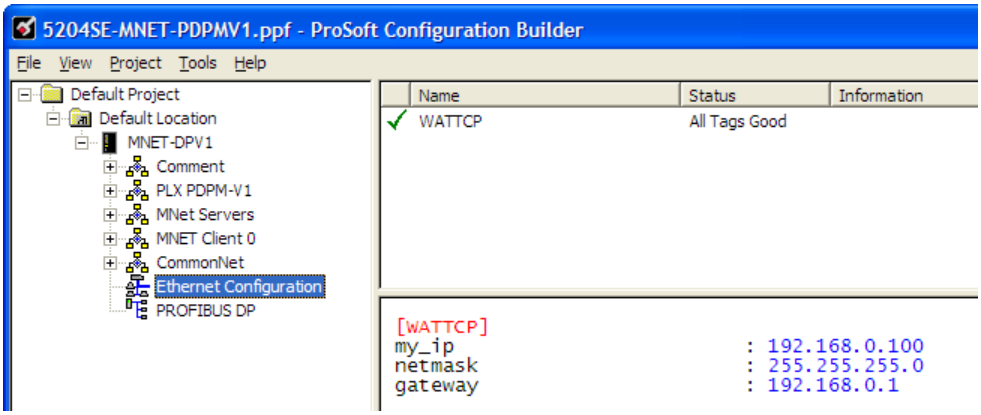
3.4.5 Configure Ethernet Settings

Use this procedure to configure the Ethernet settings for your module. You must assign an *IP address, Subnet Mask* and you may also want to assign a *Default Gateway Address* if one exists on your network. After you complete this step, you can connect to the gateway with an Ethernet cable.

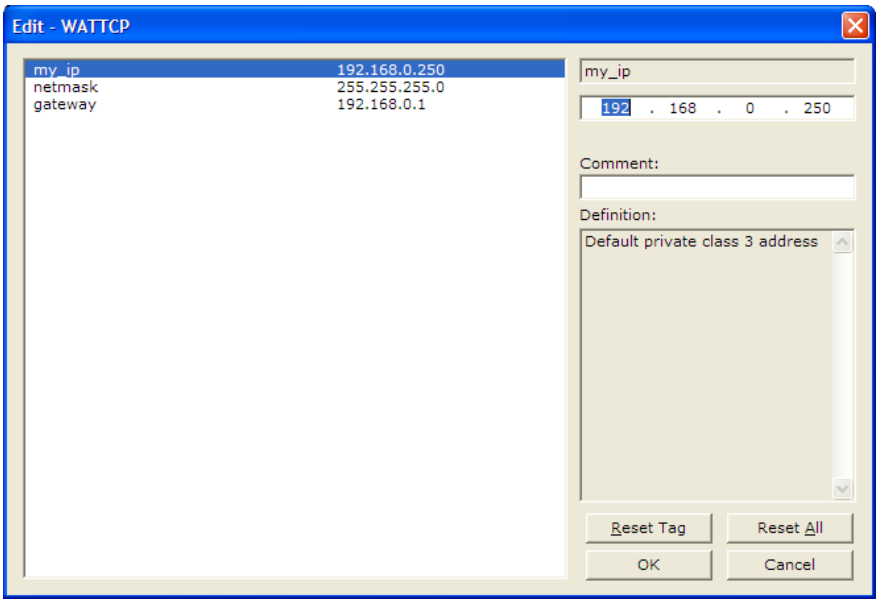
1 Determine the network settings for your module, with the help of your network administrator if necessary. You will need the following information:

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

2 Click [+] next to the gateway name to expand the tree for the 5204SE-MNET-PDPMV1 module. Click [+] next to the **COMMONNET** option to reach the **ETHERNET CONFIGURATION**.



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



4 Edit the values for **MY_IP** (*IP Address*), **NETMASK** (*Subnet mask*) and **GATEWAY** (*Default Gateway*).

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

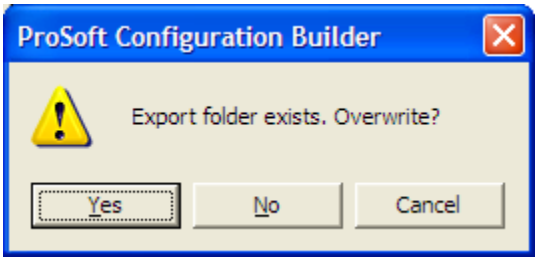
3.4.6 Back Up the PCB Project

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

To save your project and configuration files:

1. In the ProSoft Configuration Builder, open the *File* menu, and then choose **SAVE AS.**
2. Name the project file, and click **SAVE.** The recommended location for this file is your "My Documents" or "Desktop" folder.

A complete backup consists of the Project and Master Configuration files, plus the GSD files. PCB does this complete backup for you automatically. The default location for these backup files is *C:\PCBExportFiles*. All the files associated with your PCB configuration will be stored in a folder with the same name as the name you used to save your PCB configuration (.ppf) file. When you exit PCB, you will be prompted to overwrite your Export folder files.



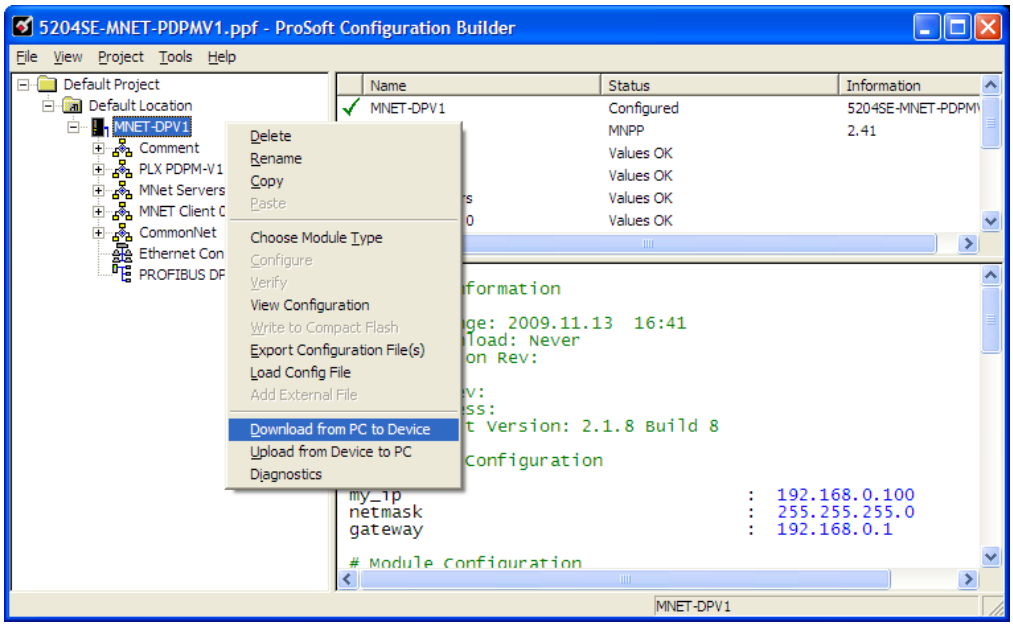
You should normally click the **YES** button every time you see this dialog box to have the backup files updated to match your latest configuration settings. Having all the files for your PCB configuration stored in one folder makes it easier to transfer the application from one system to the other or to send your files to ProSoft Technical Support when you need assistance.

3.4.7 Download the Project to the Module

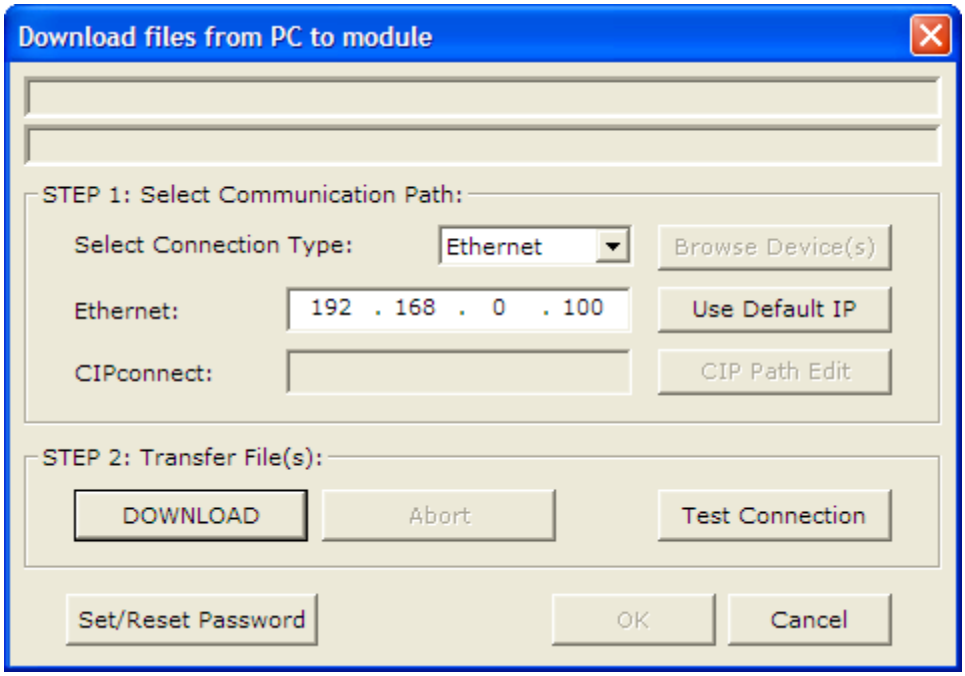
In order for the gateway to use the settings you configured, you must download (copy) the updated Project file from your PC to the gateway.

To Download the Project File

1. In the tree view in ProSoft Configuration Builder, click once to select the 5204SE-MNET-PDPMV1 module.
2. Right-click on the module name and select **DOWNLOAD FROM PC TO DEVICE** from the context menu.

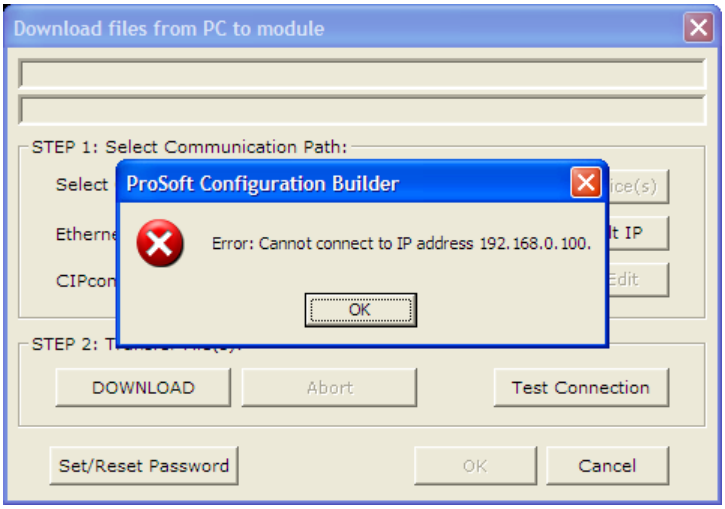


3 This action opens the *Download files* dialog box. Notice that the Ethernet address field contains the gateway default IP address. ProSoft Configuration Builder will use this default IP address to connect to the module.



Click **TEST CONNECTION** to verify that the default IP address is correct.

4 If the Test Connection procedure fails, you will see an error message.



Several factors might cause or contribute to your receiving this error. To correct the two most common errors and complete the download, check and verify the following:

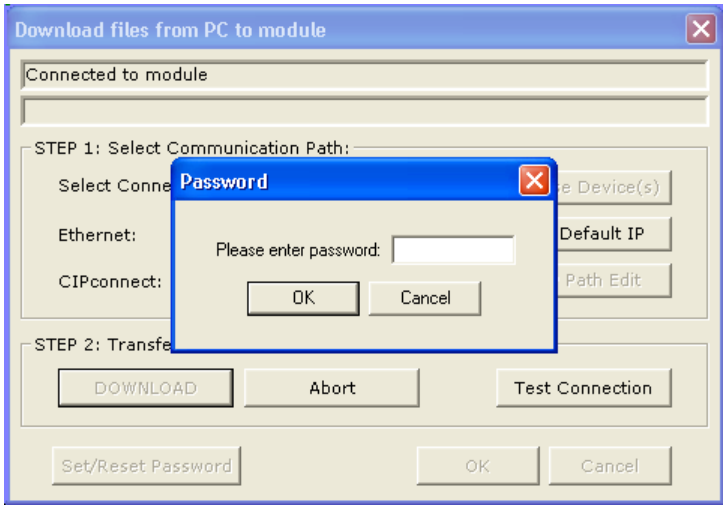
- Is the PC you are using to configure the gateway on the same subnet as the gateway?

  The subnet is determined by a combination of the *IP Address* and the *Subnet Mask*. If two devices are not on the same subnet, they will not be able to connect with each other. To correct this problem you may need to temporarily change the *IP address* and/or *Subnet Mask* on your PC to allow it to be on the same subnet as the gateway. If there is an Ethernet Gateway Server on your network, putting its IP Address in the *Gateway* parameter of the gateway's Ethernet configuration might also solve this problem.

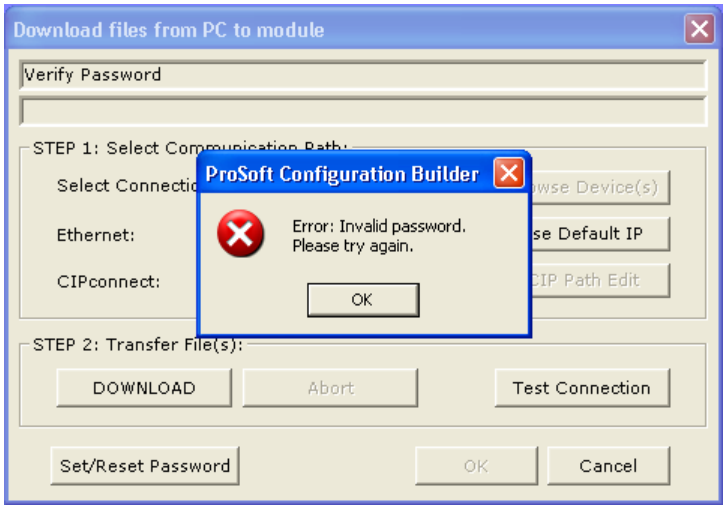
- Are there any switches, hubs, routers, or other network hardware in between your PC and the gateway which might be blocking the messages?

  If your network equipment is not configured properly, your PC may not be able to connect to the gateway. To correct this problem, you could ask your Information Technology (IT) personnel to check your network configuration. Another possible solution would be to connect your PC directly to the gateway using an Ethernet cross-over cable. This cable is different from standard Ethernet connection cables in that it has been specially wired for direct connection between two Ethernet devices. Ethernet cross-over cables are readily available from most computer parts suppliers or may be custom-made.

5 If the connection succeeds, click **DOWNLOAD** button to transfer the configuration to the module. If you do not have the configuration download password protection feature enabled on the gateway, which is the factory default condition, the download will begin. However, If you have set a password, the configuration download password protection feature will be enabled, and you will be prompted to enter your password before the download will be allowed to begin.



6 If you incorrectly enter the password or if you enter the wrong password, you will see the invalid password window and be prompted to re-enter the password. Click on the OK button, click on the Download button, and try again to correctly enter the correct password.

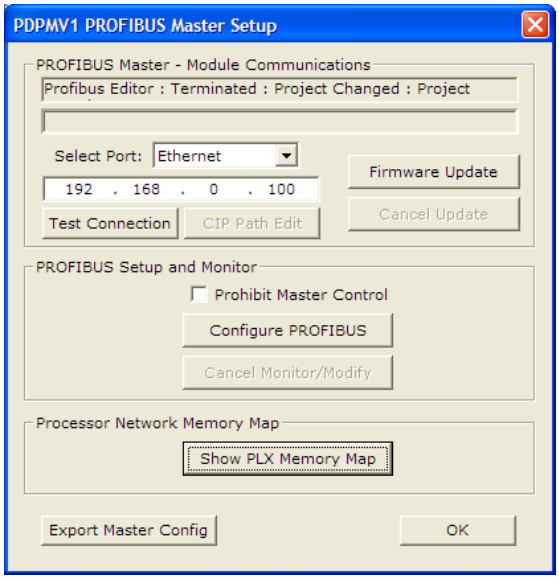


3.4.8 Export the Unity Pro v 4.0 Logic Support Files from PCB

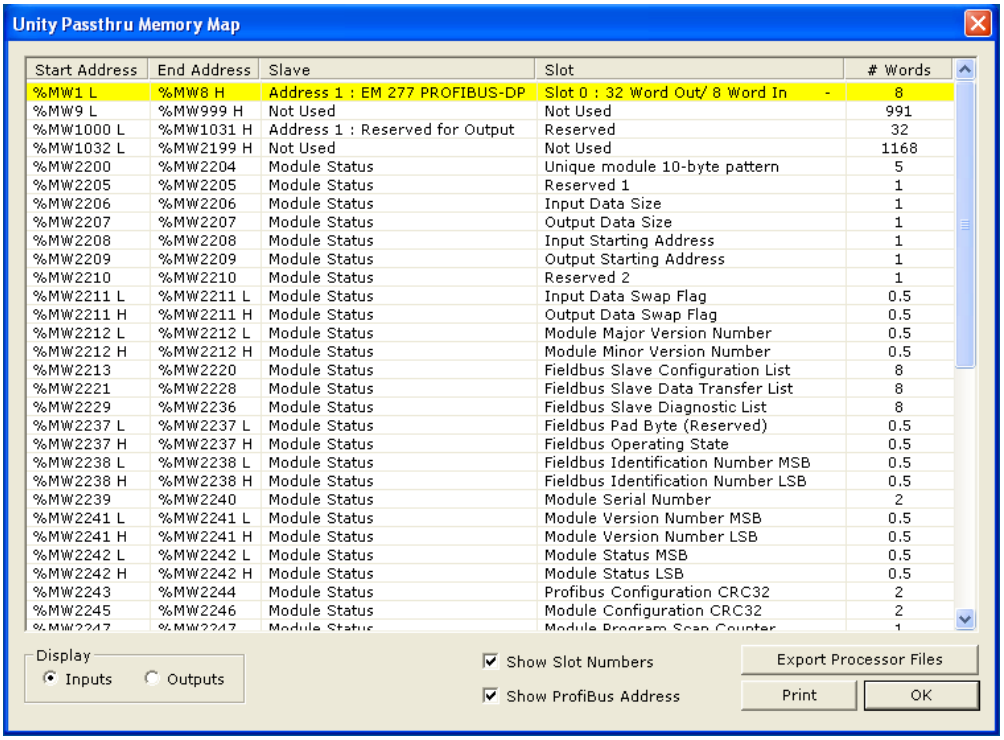
The Unity Pro import files that you create in this step use the information in the *Unity Passthru Memory Map* to build the *Derived Data Types (DDTs), Variables, and Derived Function Blocks (DFBs)* for the slave devices on your PROFIBUS DP network. These files may be uploaded and used without modification to allow the Modicon processor to act as a PROFIBUS DP Master.

To export the processor memory map:

1 In the *Master Setup* dialog box, click **SHOW PLX MEMORY MAP**.



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



3 On the Memory Map dialog box, click **EXPORT PROCESSOR FILES**. This will create two Unity Pro import files, an .XSY file and a .XFM file. Both files must be imported into the Unity Pro project for the application to work successfully.

4 Name the files (or accept the default names given by PCB), choose a location on your hard drive where you wish the files to be stored, and then click **SAVE.**

3.5 Password Protecting the Configuration

You can create password protection for the configuration that can prevent unauthorized persons from downloading configuration files.

Here are some points to remember about the password protection implementation on this gateway:

- The gateway is shipped from the factory with password protection disabled.
- Whenever password protection is disabled, you may freely download configuration files to the gateway without password protection checking until you decide you need to enable this feature.
- To begin password protection, follow the *Creating a Password* procedure to create your password. Once you create a password, each configuration download attempt will be preceded by a password check. If you enter the correct password, the download will begin. If you do not enter the correct password, the download will not be allowed to start and you will have to try again to enter the password.
- If you enter the password incorrectly, there is no limit on the number of times you may retry to enter the correct password. There is no automatic lock-out after a certain number of retries.
- Remember your password! Once password protection is enabled, if you forget your password, there is no easy way for you to recover it from the gateway, clear it, or reset it without special instructions from ProSoft Technology Technical support.
- You can change your password at any time by following the *Changing a Password* procedure.
- If you wish to discontinue using password protection, you can disable this feature by using the *Removing Password Protection* procedure.

NOTE: The original version of the gateway did not provide password protection. To use the configuration download password protection feature, you will need to be sure your hardware and software have the following version numbers:

- *ProSoft Configuration Builder (PCB) software, version* 2.1.9.1*, or higher*
- *5204SE-MNET-PDPMV1 gateway firmware, version* 2.41*, or higher*
- *5204SE-MNET-PDPMV1 gateway operating system (OS), version* 2.50*, or higher*

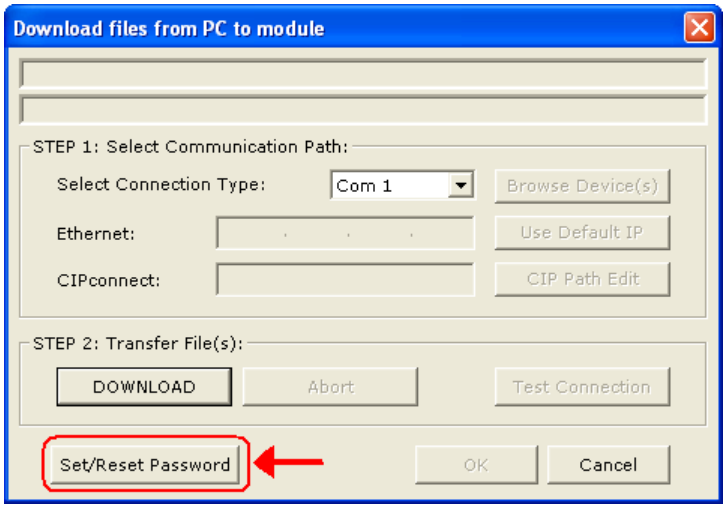
You can find the gateway firmware version and OS version numbers on the label on the back of the gateway. If the version numbers you see on the label are lower than those shown here, please contact ProSoft Technology Technical Support for upgrade information. In most cases, the gateway firmware can be upgraded in the field in a just a few minutes. If your gateway requires an operating system upgrade, it may need to be returned to the factory.

The latest version of PCB may be downloaded from:
http://www.prosoft-technology.com/content/view/full/10018

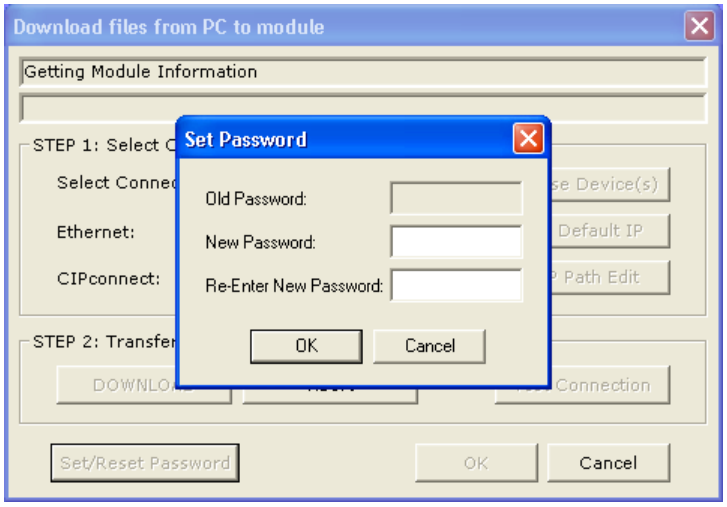
3.5.1 Creating a Password

To begin using password protection, follow these steps.

1 On the Download Files dialog box, click **SET/RESET PASSWORD** button.



This action opens the *Set Password* dialog box.

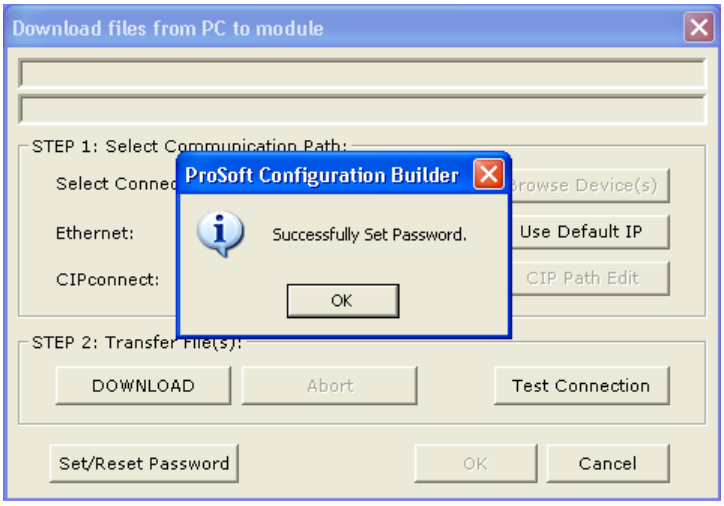


2 You will notice the *Old Password* entry box is greyed out. This indictes that password protection is currently not enabled on this gateway. Enter a password in the *New Password* box and then retype the password in the *Re-Enter New Password* box. The password can be any combination of four (4) to twelve (12) letter and number (alphanumeric) characters. The password is case sensitive, meaning:

- password1
- PASSWORD1
- PaSsWoRd1

are three different passwords, not the same password typed three different ways.

3 Click **OK**. You will see a download progress bar near the top of the download dialog box. This indicates that password protection is being enabled on the gateway. When you see the success dialog window appear, password protection has been successfully enabled. You will now be required to enter your password whenever you wish to download a configuration to the gateway.

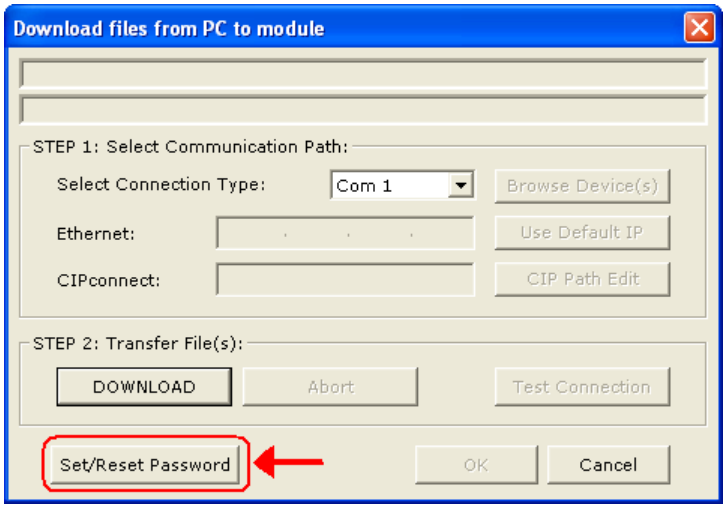


If the progress bar indicates the process has finished, but you do not see the success window, then password protection is not properly enabled on the gateway. This problem can usually be solved by upgrading the gateway firmware, gateway operating system loader, and/or ProSoft Configuration Builder to the latest versions that support password protection. For further information, contact ProSoft Technology Technical Support (page 223).

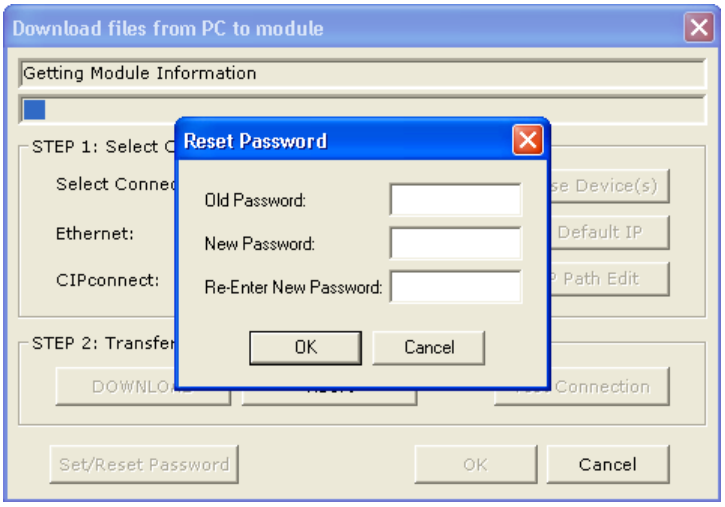
3.5.2 Changing a Password

To change the password, follow these steps.

1 On the Download Files dialog box, click **SET/RESET PASSWORD** button.



This action opens the *Reset Password* dialog box.

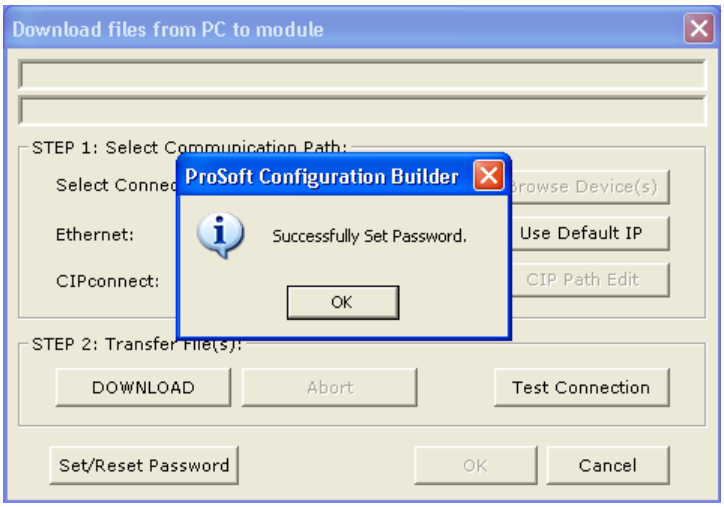


2 You will notice the *Old Password* entry box is white, just like the other boxes. This indicates that password protection is currently enabled on this gateway. To change the password, you will need to enter the current passwork in the *Old Password* entry box, enter a new password in the *New Password* box, and then retype the new password in the *Re-Enter New Password* box. The password can be any combination of four (4) to twelve (12) letter and number (alphanumeric) characters. The password is case sensitive, meaning:

- password1
- PASSWORD1
- PaSsWoRd1

are three different passwords, not the same password typed three different ways.

3 Click **OK**. You will see a download progress bar near the top of the download dialog box indicate that the new password is being downloaded to the gateway. When you see the success dialog sceen appear, the new password has been successfully installed. You will now be required to enter this new password whenever you wish to download a configuration to the gateway.

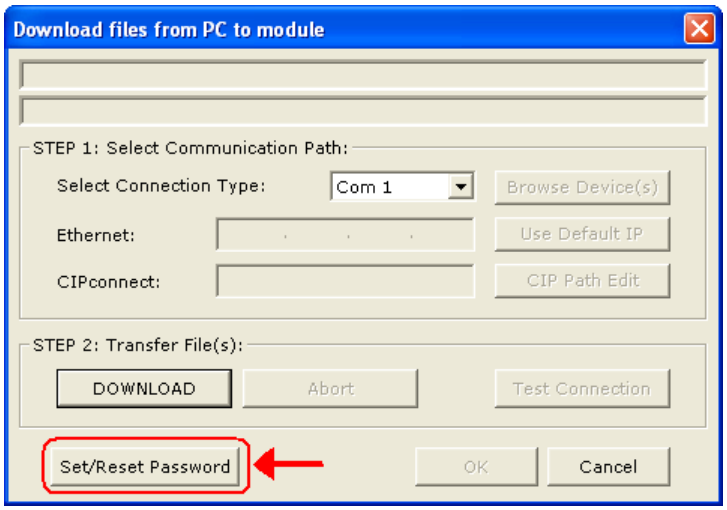


If the progress bar indicates the process has finished but you do not see the success window, then the new password has not been successfully changed on the gateway and the old password is still in effect. Make sure that the length of the new password you are trying to set is no less than 4 alphanumberic characters and no more than 12 alphanumeric characters, and try the procedure again.

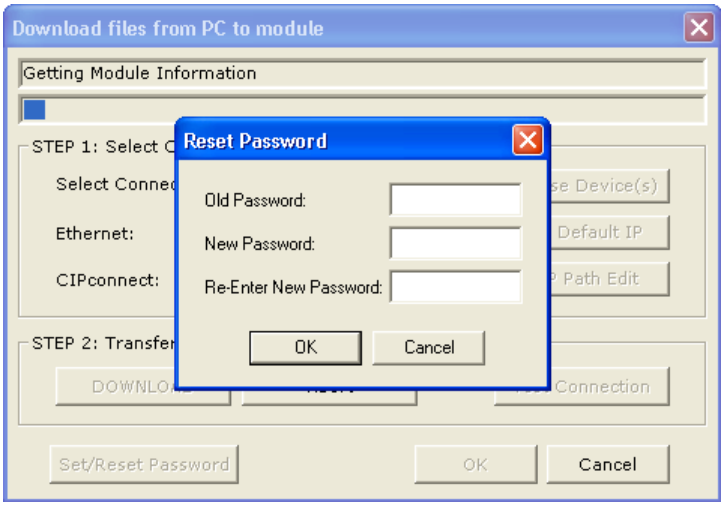
3.5.3 Removing Password Protection

To remove the password and disable configuration download password protection checking on the gateway, follow these steps.

1 On the Download Files dialog box, click **SET/RESET PASSWORD** button.



This action opens the *Reset Password* dialog box.

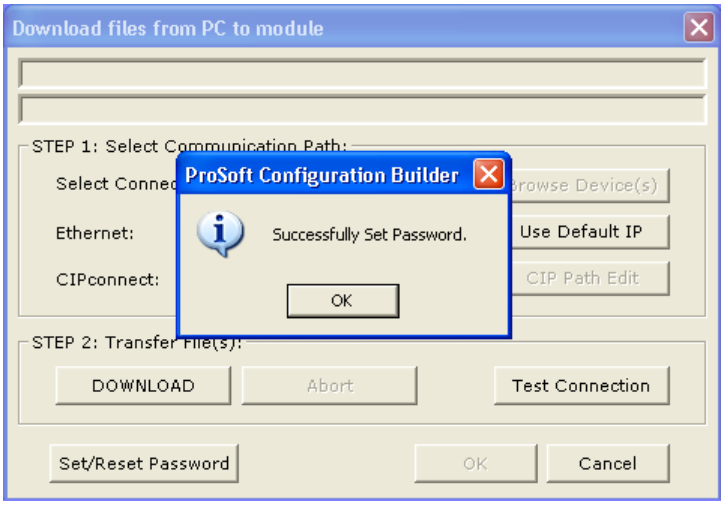


2 You will notice the *Old Password* entry box is white, just like the other boxes. This indicates that password protection is currently enabled on this gateway. To remove the password, you will need to:

- Enter the current passwork in the *Old Password* entry box
- Enter nothing in the *New Password* box
- Enter nothing in the *Re-Enter New Password* box.

Leaving the New Password and Re-Enter New Password boxes blank will clear out the existing current password and disable password checking on the gateway, once this change has been successfully completed.

3 Click **OK**. You will see a download progress bar near the top of the download dialog box indicate that the existing password is being deleted from the gateway. When you see the success dialog sceen appear, the existing password has been successfully erased. You will no longer be required to enter a password whenever you wish to download a configuration to the gateway. No further password checks will be done before a configuration is allowed to download.



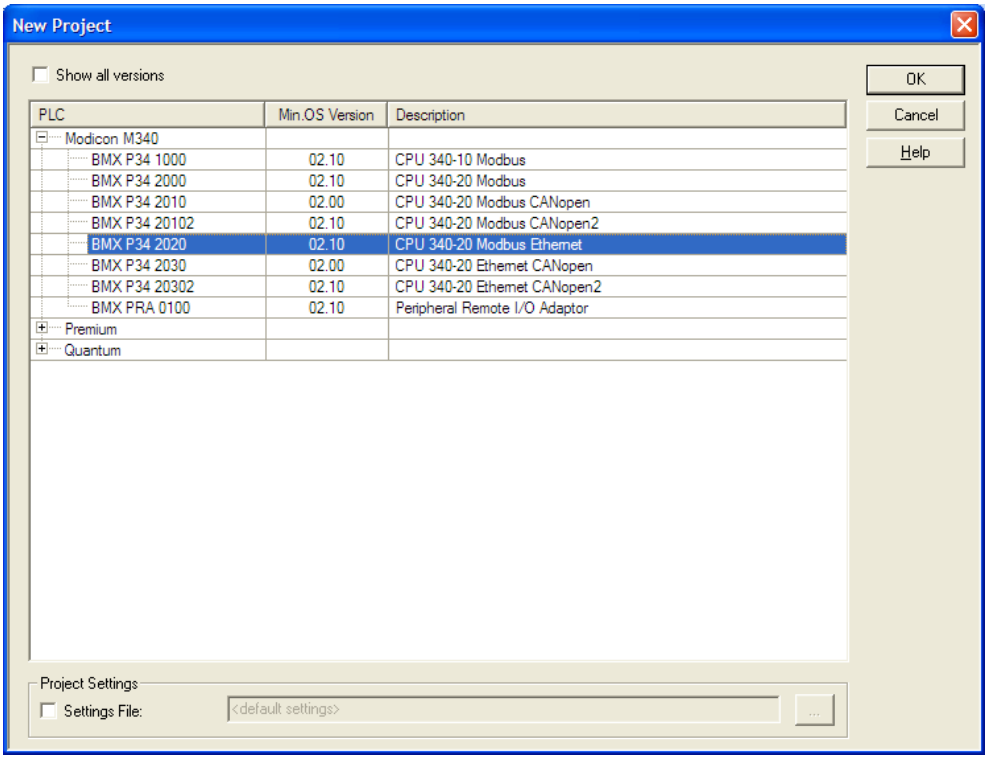
If the progress bar indicates the process has finished but you do not see the success window, then the existing password has not been successfully deleted from the gateway and password protection is still enabled. Try the procedure again.

3.6 Configure the Modicon M340 Processor with Unity Pro

3.6.1 Create a New M340 Project

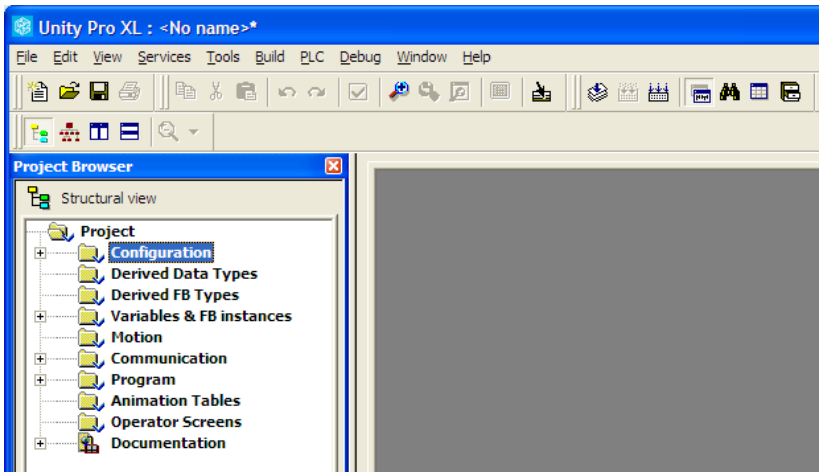
The first step is to open Unity Pro and create a new project.

1. Start Unity Pro. Open the **FILE** menu, and then select **NEW**. This action opens the *New Project* dialog box.
2. In the New Project dialog box, choose the CPU type that matches your Modicon M340 processor.

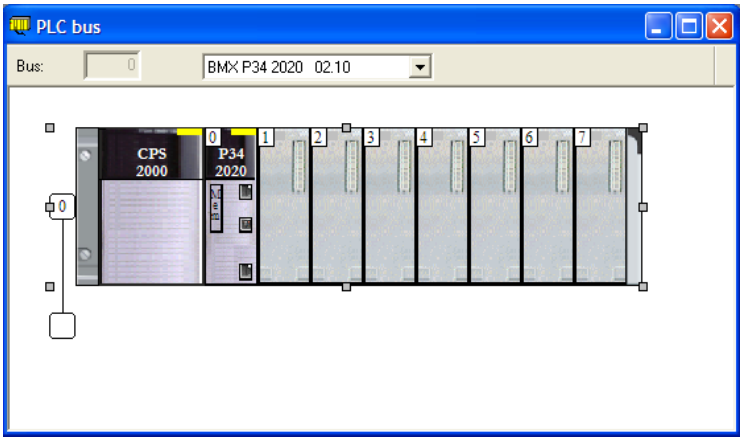


Click OK. This action opens the *Project Browser* pane.

3 In the Project Browser, double-click **CONFIGURATION** to open the *PLC Bus* window.



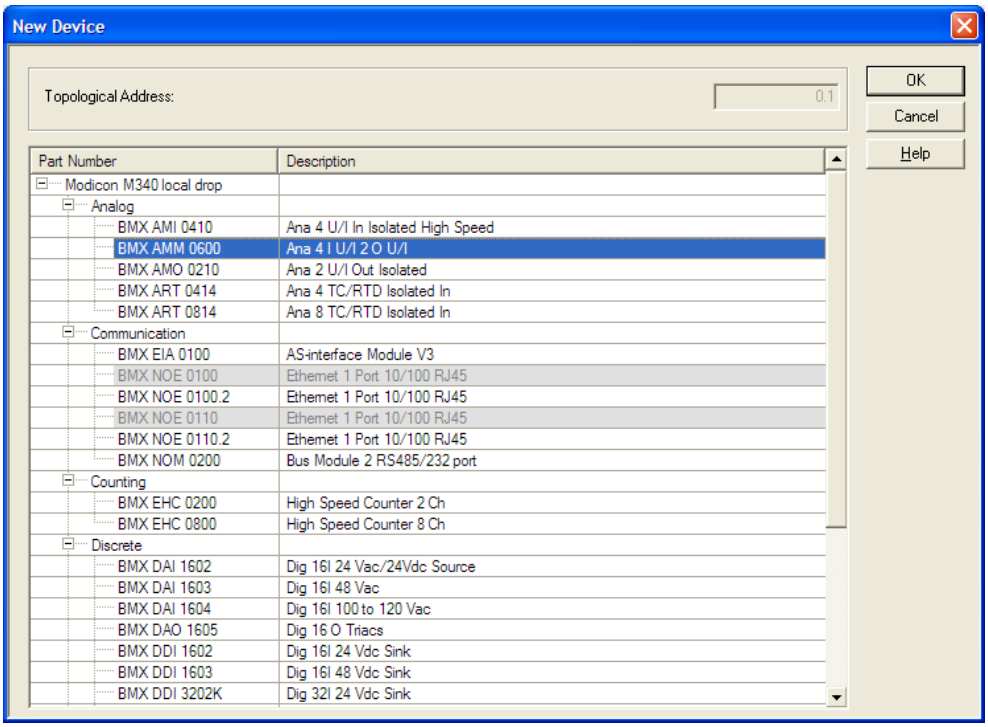
Notice that the image in the window shows the processor in the second position in the rack (the first position is for the power supply.)



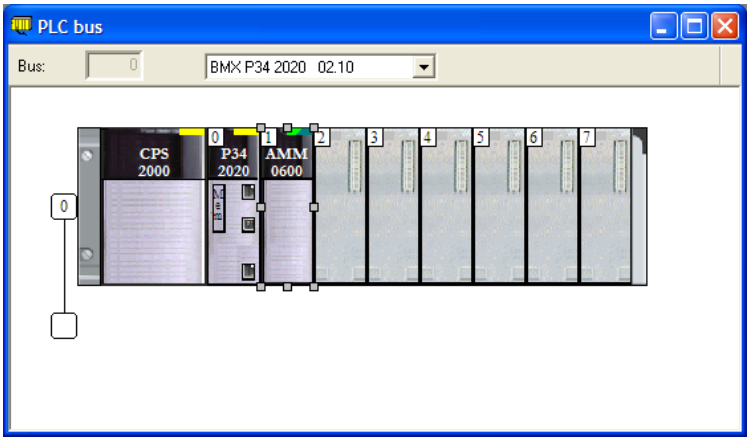
4 For this example, you will populate the rack with a combination of modules that represent all the possible Modbus data types:

- Coil bits
- Input Status Bits
- Input Registers
- Holding Registers.

To add devices to the rack, double-click the location (slot) in the rack where the device will be installed. This action opens the *New Device* dialog box.

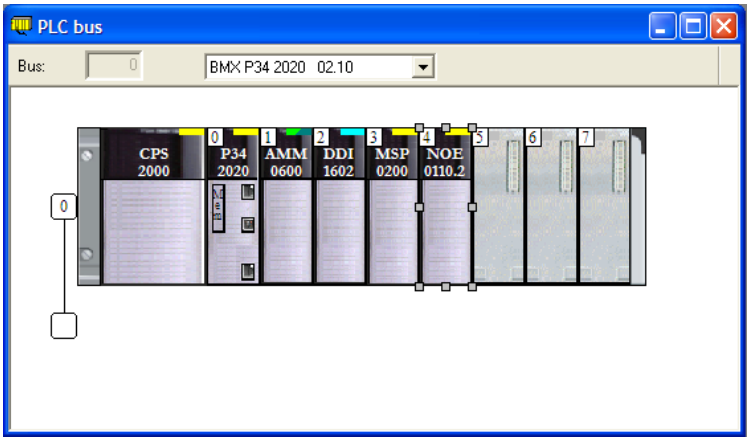


5 Click the **[+]** sign next to module types to open the list of devices. Select a module from the list, and then click OK. This action adds a module to the *PLC Bus* image.

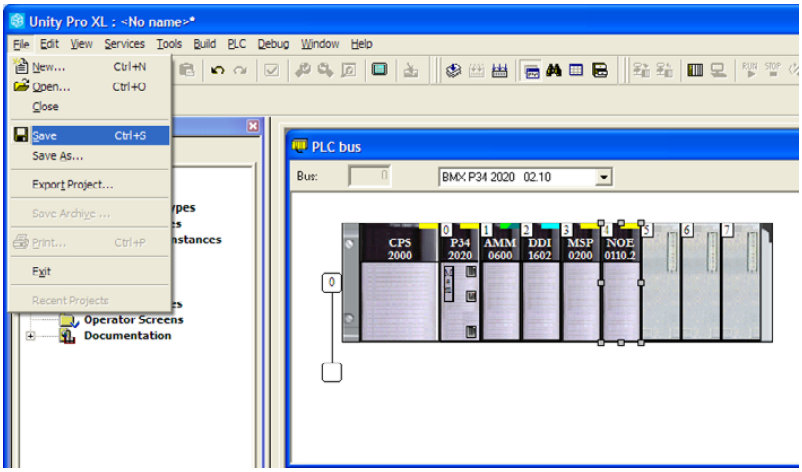


6 Repeat steps 4 and 5 to add the following modules to the project:

- Analog: BMX AMM 0600
- Discrete: BMX DDI 1602
- Motion: BMX MSP 0200
- Communication: BMX NOE 0110.2



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



3.6.2 Configure the Memory Size for the Processor

The processor memory maps that you viewed in and exported from ProSoft Configuration Builder (PCB) will be imported into the Unity Pro project. These processor State RAM maps are calculated from the starting memory addresses and register counts entered into PCB for the module's input and output data images. For more information on configuring memory addresses in PCB, refer to Configure the Gateway (page 32).

Allocating processor memory to store input and output data is part of the processor configuration process. You should view the memory configuration in the PCB Processor Memory Maps before you begin to allocate memory addresses in Unity Pro.

Some points to keep in mind are:

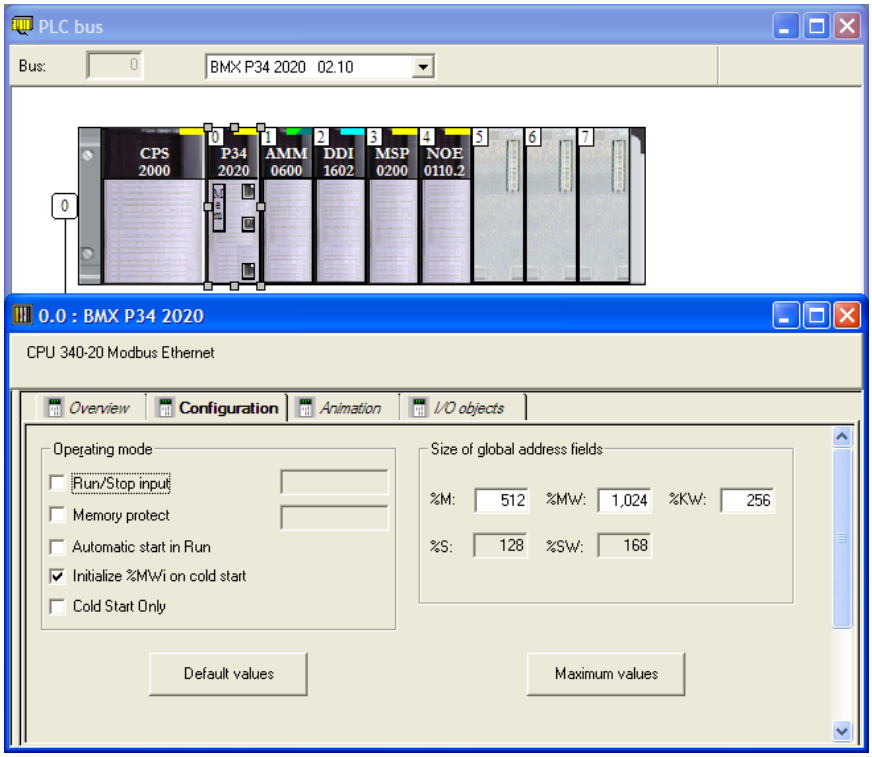
- As the programmer, you must be aware of the memory spaces that are available when deploying in an existing system and assign values to the Modicon processor and 5204SE-MNET-PDPMV1 configurations accordingly.
- The Modicon M340 processor has a maximum *%MW* memory allocation size limit of 32,464 16-bit registers. The default size is 1024 registers. While setting the *%MW* memory allocation, you must allocate enough total memory to accommodate the amount required for the 5204SE-MNET-PDPMV1 gateway as well as for the rest of your application.
- The total number of data registers allocated for PROFIBUS data must at least equal or exceed the number needed, which can be calculated by taking the starting register configured in PCB and adding the register count configured, plus any additional registers required for the rest of the application process logic. The memory map from PCB can help you determine these numbers.
- The gateway can use up to 768 words of cyclic input data, 768 words of cyclic output data, 76 words of status data, and 378 words of standard PROFIBUS slave diagnostic data. Therefore, the total *%MW* memory requirement for just the PROFIBUS application could be as much as 1990 words. Round this up to an even 2000 registers as the amount of *%MW* memory to allocate for PROFIBUS data.
- You must allocate at least this much memory space as a continuous, uninterrupted block of processor memory that will not be used by any I/O modules, processes, or variables.

WARNING: Failure to properly map your processor memory will likely cause corruption of PROFIBUS data and can create potentially hazardous situations resulting from unexpected equipment operation; which can result in injury to personnel or damage to equipment.

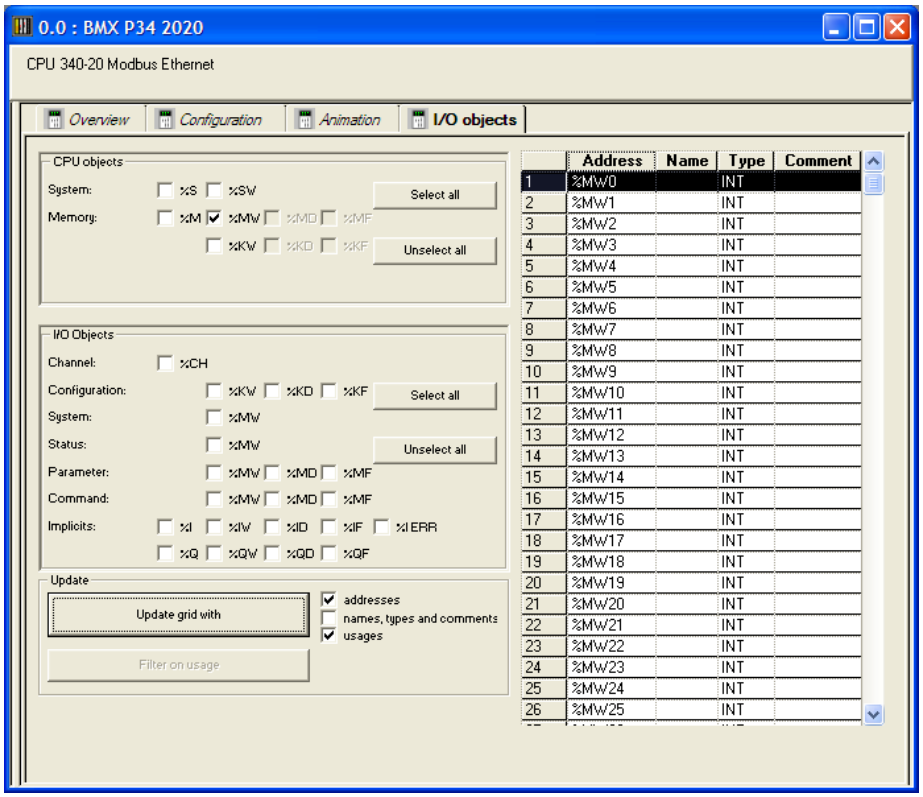
The following steps will help you determine the correct memory addresses to assign.

To view memory usage in the processor:

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



5 To view detailed information about the processor's memory configuration, click the **I/O OBJECTS** tab. These selections offers tools to view the types of data stored at specific addresses in the processor. Make note of memory areas that are already allocated, and select an area of contiguous memory that can be allocated to the gateway.



3.6.3 Import the M340 Functional Module (.XFM File)

To simplify the task of programming the processor when communicating with the 5204SE-MNET-PDPMV1, the Application Communication Logic functions of ProSoft Configuration Builder (PCB) create a Unity Pro Functional Module (XFM).

Note: The Functional Module is intended only for new installations of the gateway. If you have an existing installation, the following procedure will overwrite your settings, and may cause loss of functionality. DO NOT overwrite a working application until you have thoroughly reviewed the rest of the topics in this manual.

The Functional Module provides easy access to:

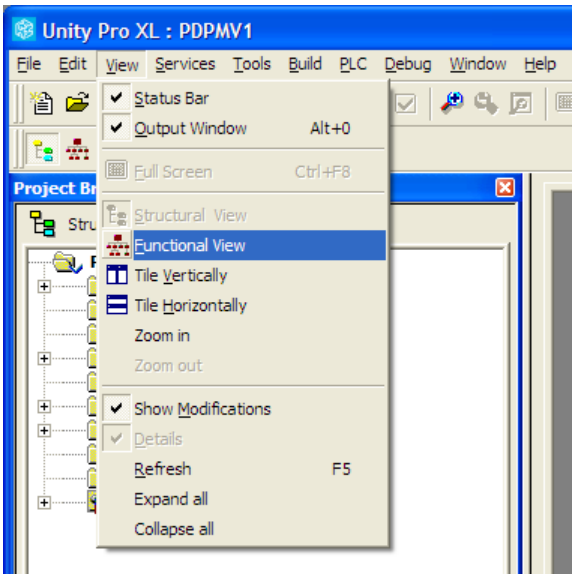
- PROFIBUS DP-V0 or DP-V1 cyclic input and output data
- gateway input/output status data
- Standard PROFIBUS slave diagnostic data (six bytes per slave)
- PROFIBUS DP-V1 acyclic message data, such as "Get Live List", "Get (Extended) Slave Diagnostics", perform Freeze and Sync commands, or perform any slave device-specific commands or functions.

The Functional Module file name matches the gateway name you defined in PCB and will have the extension ".XFM". This file is created by PCB when you export the processor file from the Unity Passthru Memory Map box.

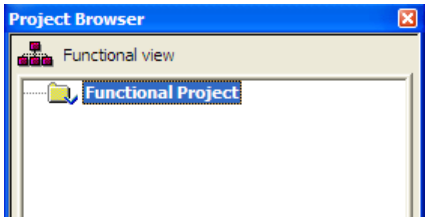
To import the Functional Module:

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

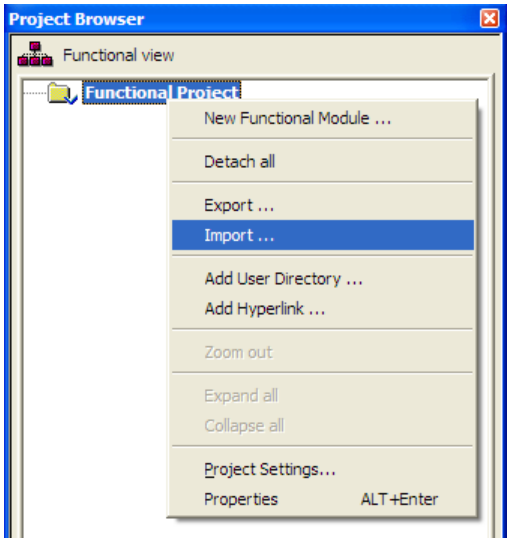
1 Open the **VIEW** menu, and then choose **FUNCTIONAL VIEW.**



This action populates the *Project Browser* with a **FUNCTIONAL PROJECT** icon.

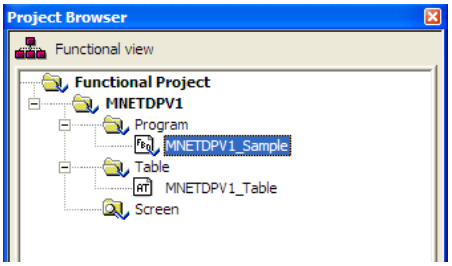


2 Select **FUNCTIONAL PROJECT** and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT.**

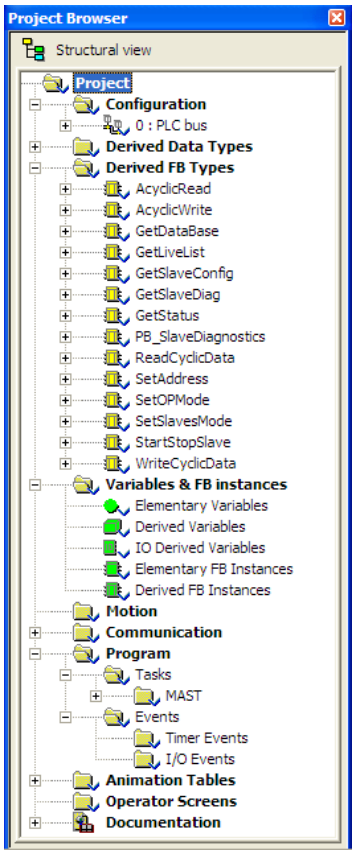


3 In the *Import* dialog box, choose **FUNCTIONAL MODULE (*.XFM)** in the Files of Type dropdown list and then select the XFM file to import. The XFM file name matches the gateway name you defined in PCB and exported (page 47).

Click **IMPORT** to import the file. Notice that the *Project Browser* is now populated with the *Functional Module*.



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



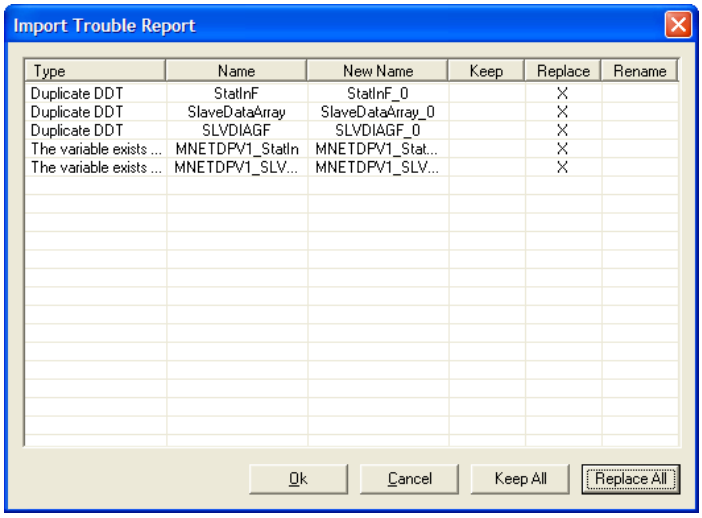
3.6.4 Import the M340 Variables (.XSY file)

The Application Communication Logic functions of ProSoft Configuration Builder (PCB) also create a list of *variables* and *variable structures* customized to the particular PROFIBUS DP-V1 Master configuration you created. These *variables* are contained in the "{ProjectName}.XSY" file you exported in Export the Unity Pro v 4.0 Logic Support Files (page 47).

The .XSY file contains all the cyclic input and output variables configured by the PCB master configuration software. This file includes gateway status data and will also include slave diagnostics data if the **SLAVE DIAGNOSTICS** parameter was set to Yes.

To import the Variables:

1. In the Project Browser, select **VARIABLES & FB INSTANCES**, and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT**.
2. In the **FILES OF TYPE:** dropdown list, choose **DATA EXCHANGE FILE (*.XSY)**. Select the .xsy file created when you exported the processor files from PCB (page 47) and then click **IMPORT.**
3. If you see an *Import Trouble Report* window, click **REPLACE ALL**, then click **OK.**



At this point, the *DDTs*, *DFBs*, and *Variables* have been imported to the application.

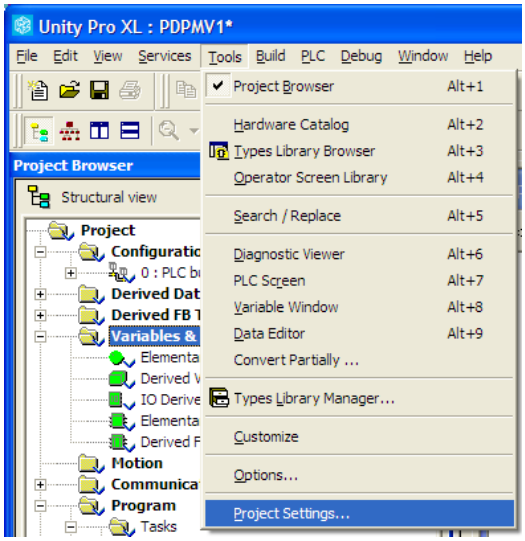
3.6.5 Build the M340 Project

Whenever you update the configuration of your gateway, the PROFIBUS network, or the processor, you must import the changed configuration from ProSoft Configuration Builder (PCB) 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.

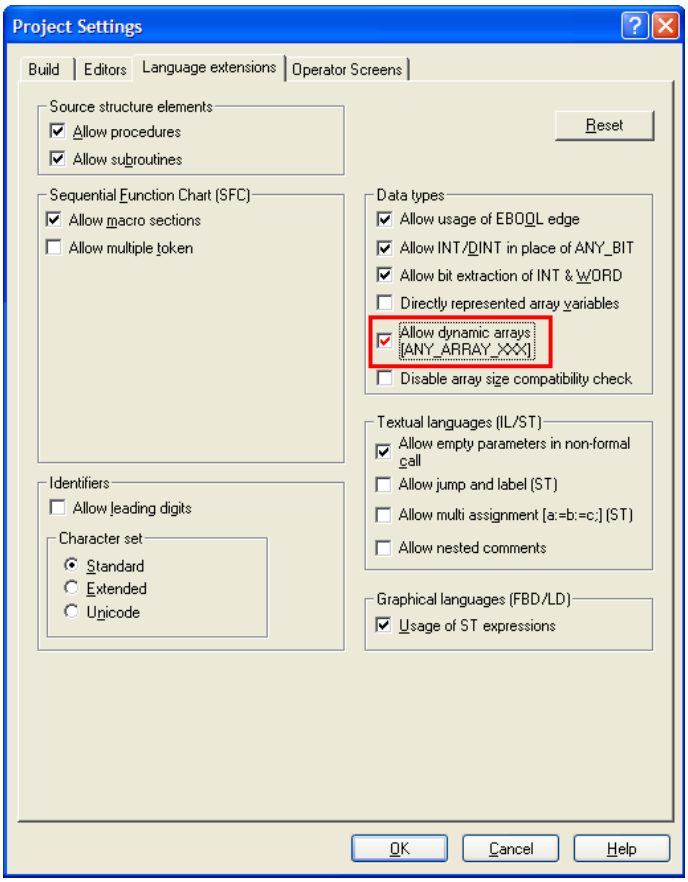
To build (compile) the project:

1. Review the elements of the project in the *Project Browser*.
2. Make sure you have configured sufficient *%MW* memory space for your entire project and PROFIBUS data.
3. To avoid build errors, you will need to enable the **DYNAMIC ARRAY LANGUAGE EXTENSION** option. From the Unity Pro menu bar, select **TOOLS,** and then choose **PROJECT SETTINGS**.



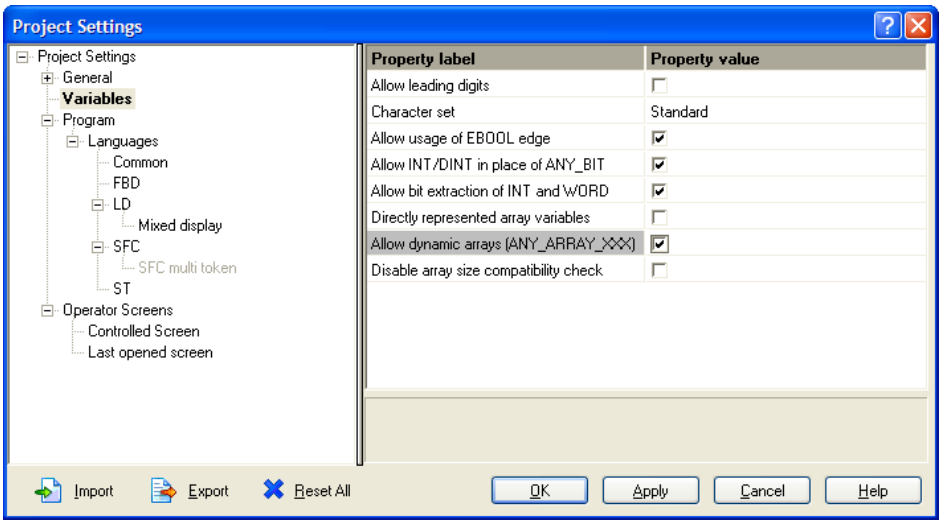
- For UnityPro version 4.0

  In the *Project Settings* box, click the **LANGUAGE EXTENSIONS** tab and select (check) **ALLOW DYNAMIC ARRAYS [ANY_ARRAY_XXX]**.



- For UnityPro version 4.1:

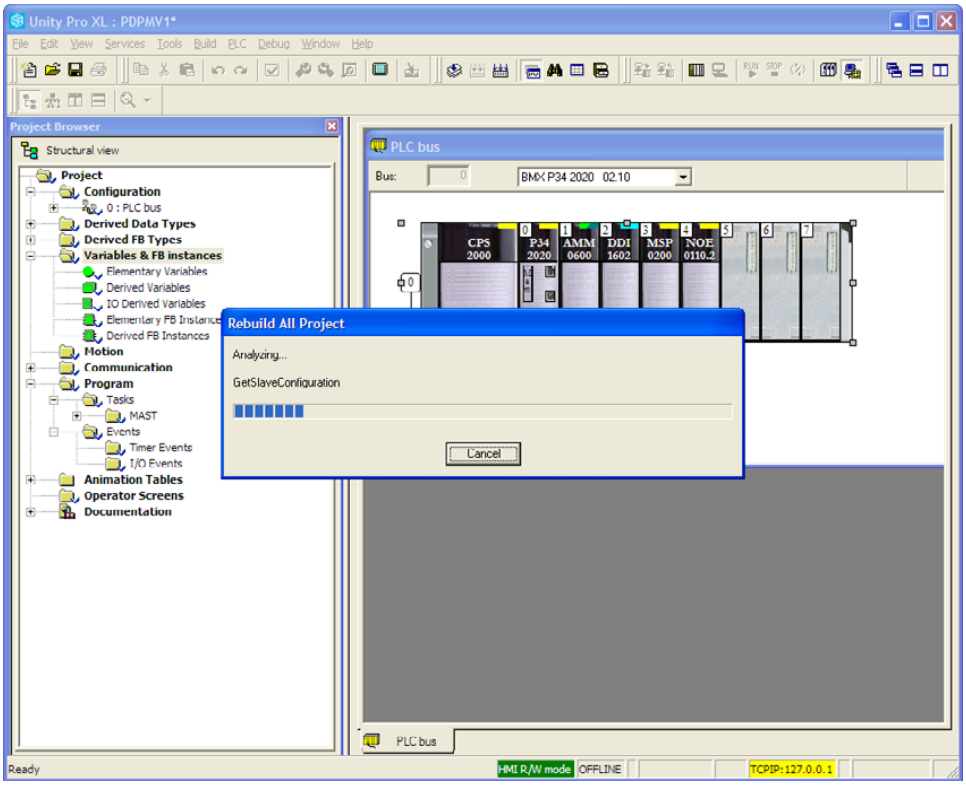
  Select Variables in the left pane, and then select (check) **ALLOW DYNAMIC ARRAYS [ANY_ARRAY_XXX]**



Click **OK** to save your changes and dismiss the dialog box.

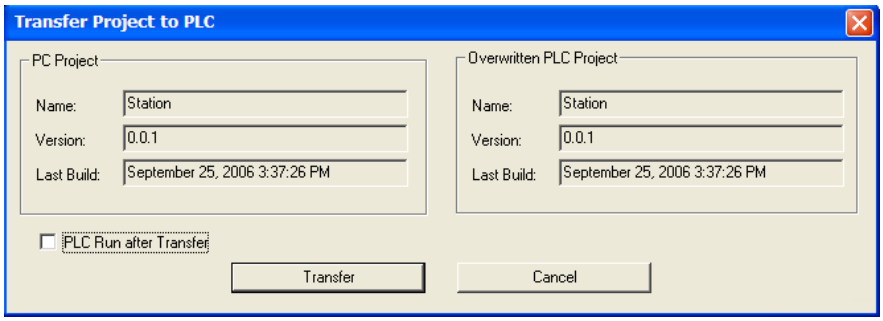
4 When you are satisfied that you are ready to download the project, open the **BUILD** menu, and then choose **REBUILD ALL PROJECT**. This action builds (compiles) the project into a form that the processor can use to execute the instructions in the project file. This task may take several minutes,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

5 As the project is built, Unity Pro displays a Progress dialog box, with details appearing in a pane at the bottom of the window. If you are using the files from PCB and have your memory and processor configuration set up correctly, the project should build without errors. The following illustration shows the build process under way.



3.6.6 Download the Project to the Quantum Processor

1. Open the **PLC** menu and then choose **CONNECT.** This action opens a connection between the Unity Pro software and the processor, using the address and media type settings you configured in the previous step.
2. On the **PLC** menu, choose **TRANSFER PROJECT TO PLC**. This action opens the **TRANSFER PROJECT TO PLC** dialog box. If you would like the PLC to go to "Run" mode immediately after the transfer is complete, select (check) the **PLC RUN AFTER TRANSFER** check box.



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

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

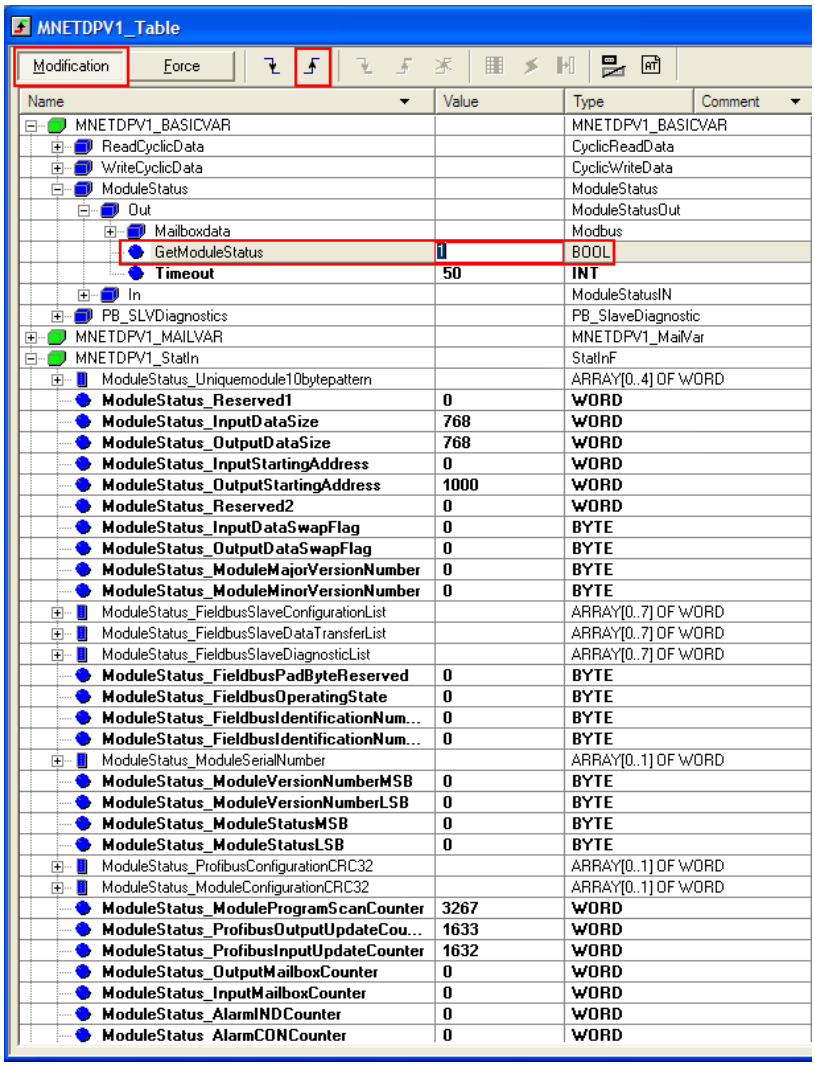
3.6.7 Verify Communication between the M340 Processor and the Gateway

In this step, you will verify that the processor and the gateway are communicating with each other over the Modbus TCP/IP Ethernet network. The sample project includes an animation table called **MNETDPV1_Table**. When the processor is in **RUN** mode and communicating with the gateway, the values in this animation table are updated whenever you trigger a *GetModuleStatus* read message to request general gateway status from the gateway.

To verify communication between the processor and the gateway:

1. Place the processor in **RUN** mode, if you have not already done so.
2. In the Unity Pro *Project Browser* pane, click **[+]** to open the **ANIMATION TABLES** tree, and then double-click **MNETDPV1_TABLE.**
3. In the *MNETDPV1_Table*, you will see three main *variables* which can be expanded to see many more *sub-variables*.
4. Look for the variable, *MNETDPV1_StatIn*. Click the **[+]** next to it to open it and see all the *sub-variables*. These will include configuration information and version/revision information, which will remain static, as well as program scan counters, input/output update counters, request/response counters, and others *variables* that should change whenever you trigger a status calls.

5 Now, look for and expand the *variable*, **MNETDPV1_BASICVAR.** From there, look for and expand the *sub-variable*, **MODULESTATUS.** Expand the **OUT** *sub-variable*.



Force a one (1) into the sub-variable, *GetModuleStatus*. Notice that the **MODIFICATION** button must be engaged and you must use the **SET TO 1** icon option to actually have the variable value changed so the update request will be sent.

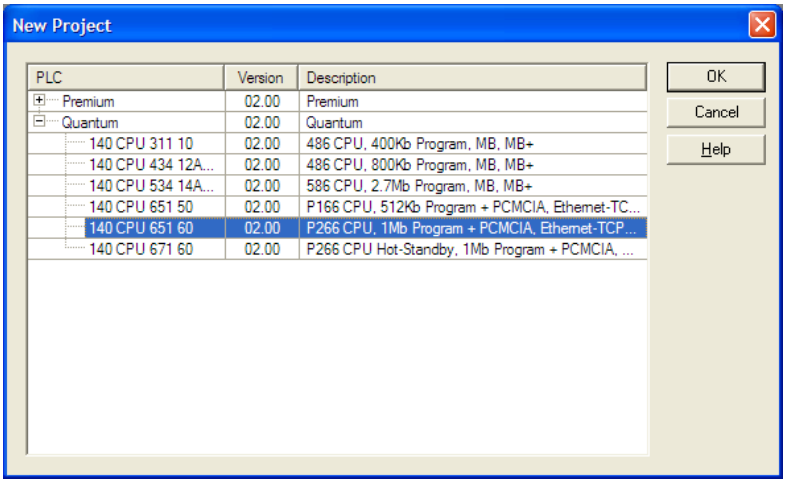
6 Scroll within *MNETDPV1_StatIn*. Notice that whenever you force the **GETMODULESTATUS** update, the numbers in the **Value** column for items such as *ModuleStatus_ModuleProgramScanCounter* are updated.

3.7 Configure the Modicon Quantum Processor with Unity Pro

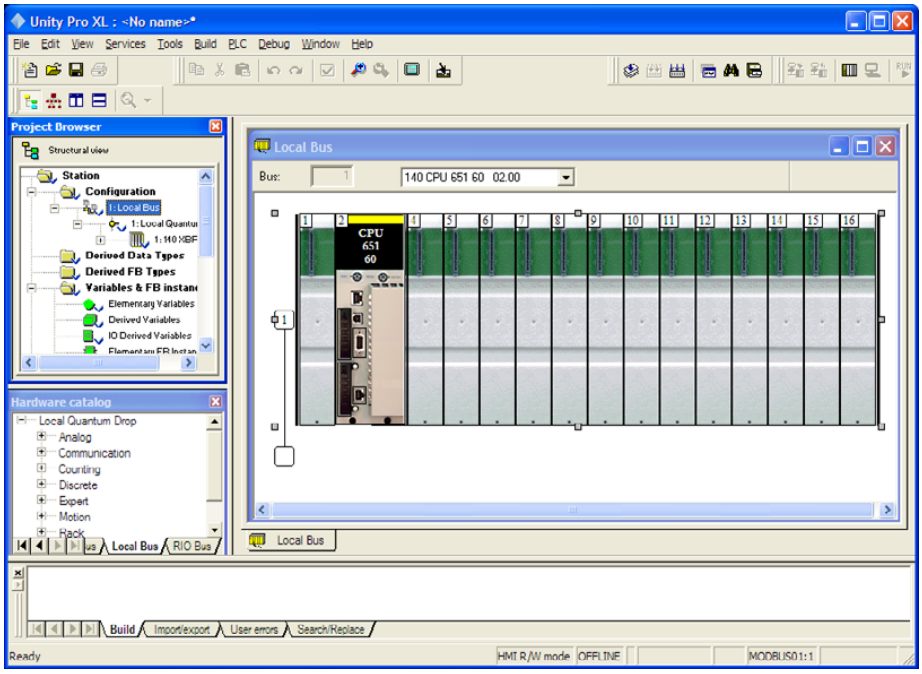
3.7.1 Create a New Quantum Project

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

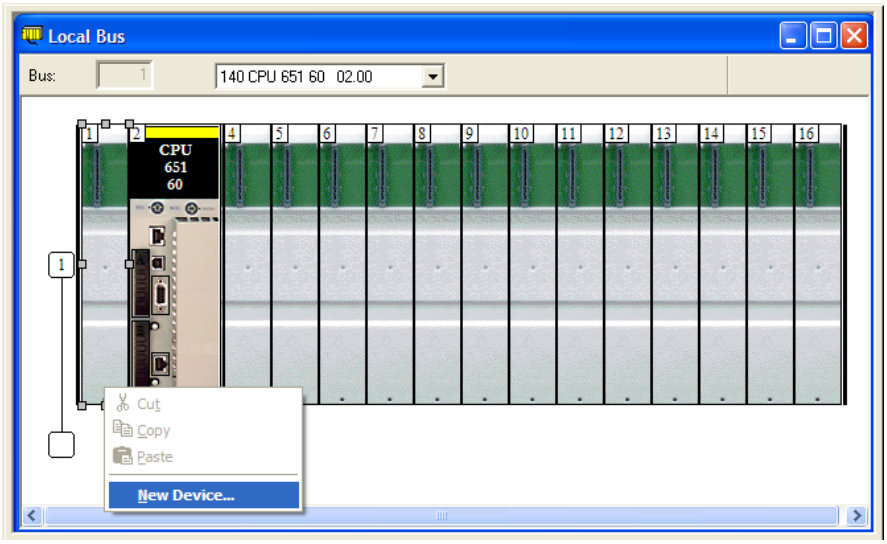
1 In the **NEW PROJECT** dialog box, choose the CPU type. In the following illustration, the CPU is 140 CPU 651 60. Choose the processor type that matches your own hardware configuration, if it differs from the example. Click **OK** to continue.



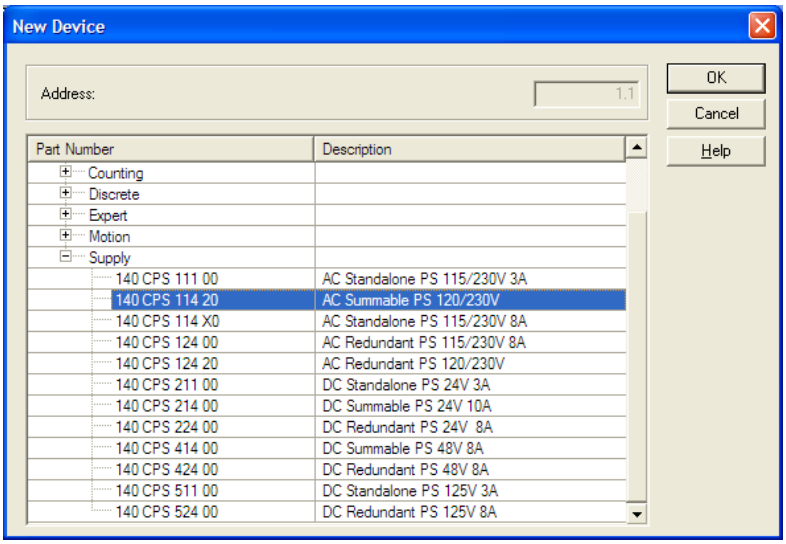
2 Next, add a power supply to the project. In the **PROJECT BROWSER**, expand the **CONFIGURATION** folder, and then double-click the **1:LOCALBUS** icon. This action opens a graphical window showing the arrangement of devices in your Quantum rack.

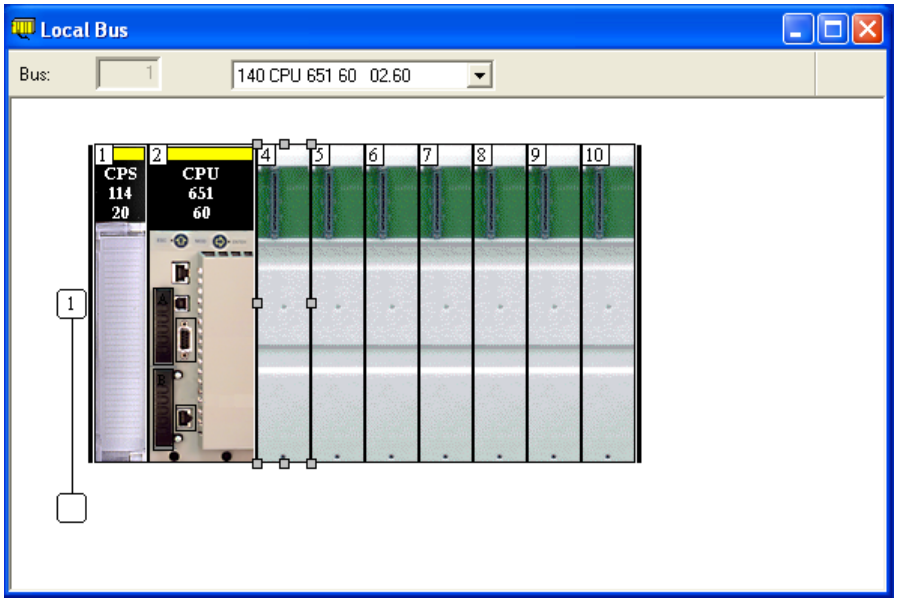


3 Select the rack position for the power supply, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW DEVICE**.



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

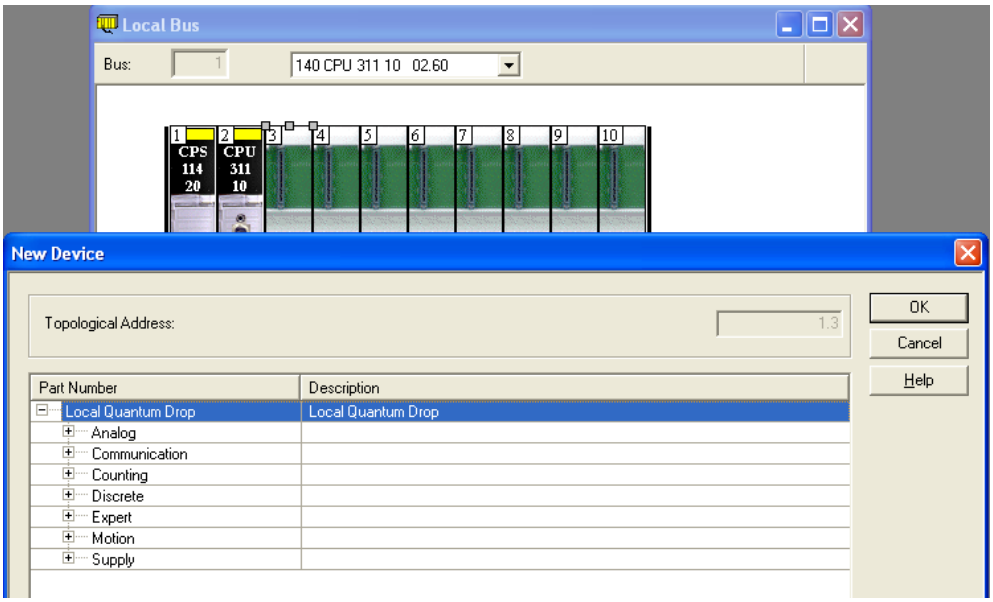




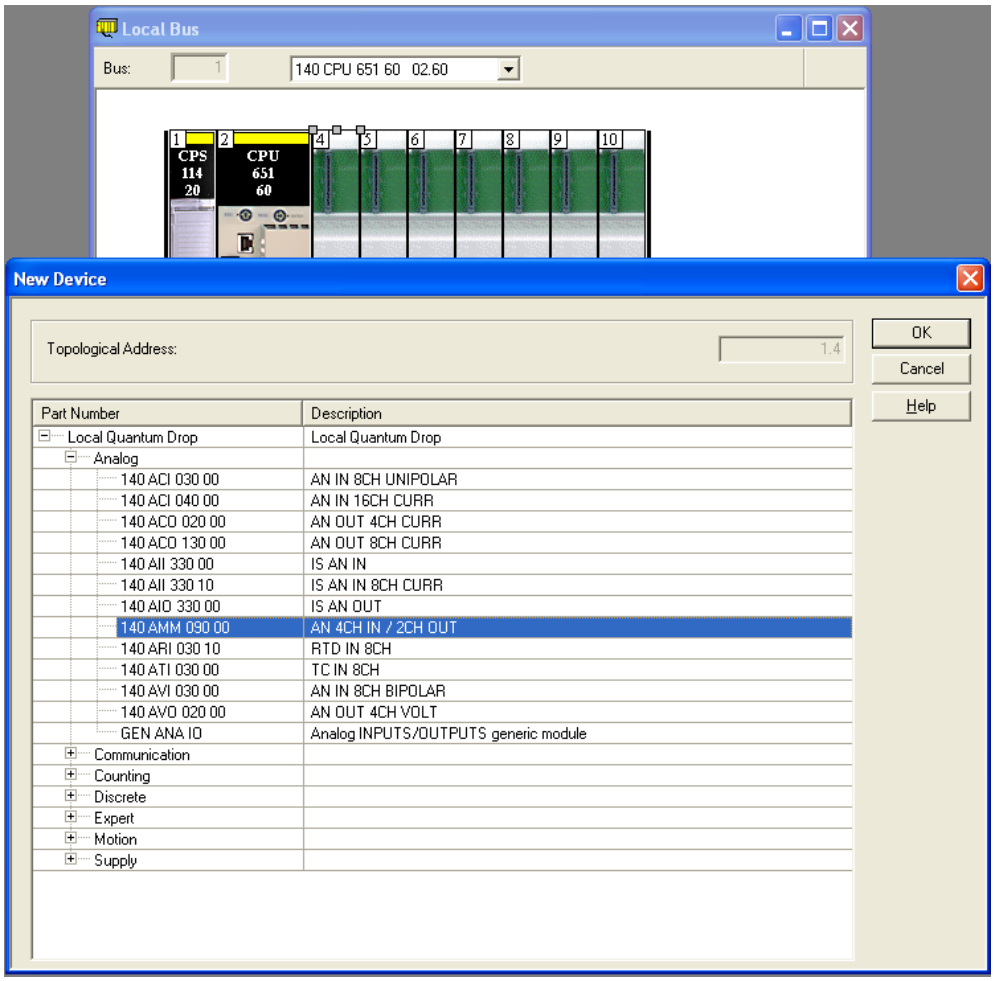
5 For this example, you will populate the rack with a combination of modules that represent all the possible Modbus data types:

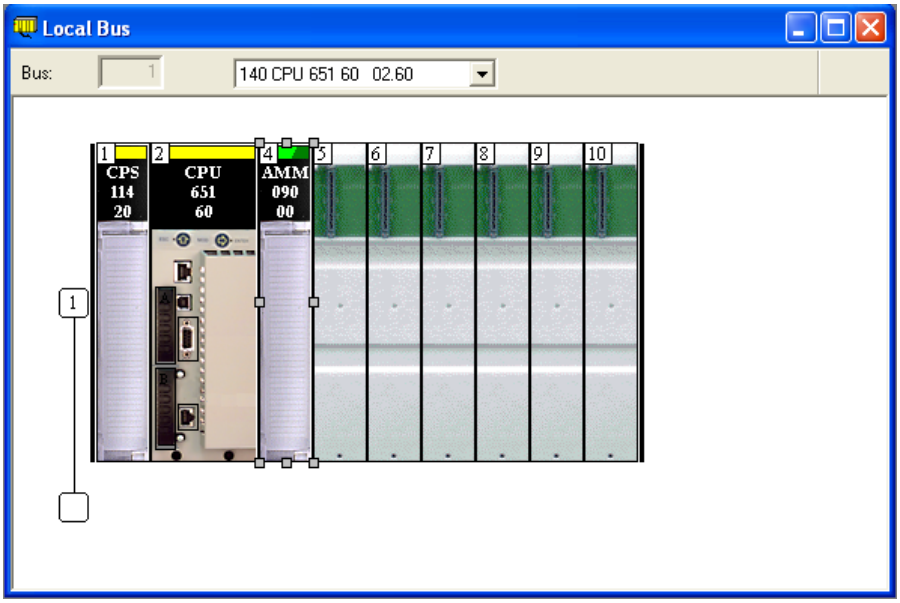
- Coil Bits
- Input Status Bits
- Input Registers
- Holding Registers.

To add devices to the rack, double-click the location (slot) in the rack where the device will be installed. This action opens the *New Device* dialog box.



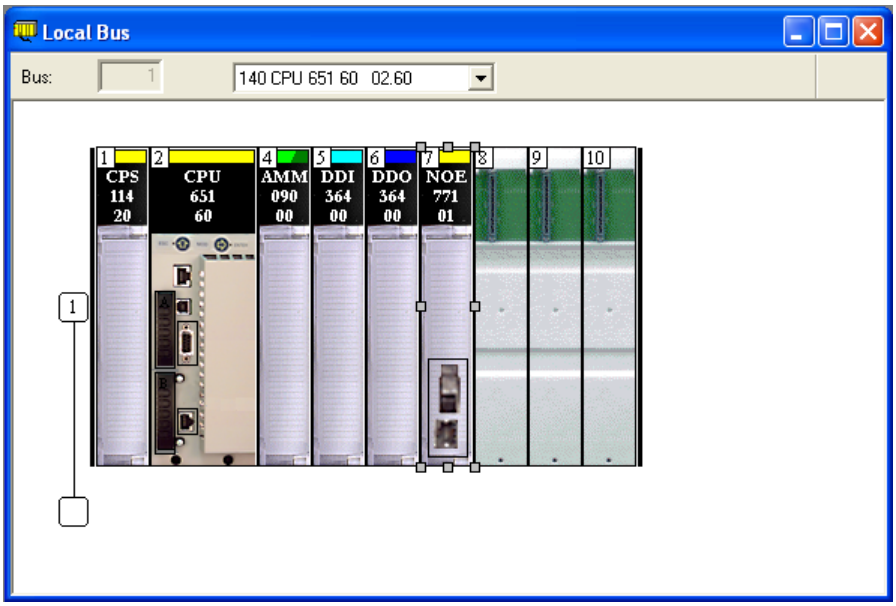
6 Click the **[+]** sign next to module types to open the list of devices. Select a module from the list, such as the 140 AMM 090 00 in the examples below, and then click OK. This action adds a module to the *PLC Bus* image.





7 Repeat steps 5 and 6 to add the following modules to the project:

- Discrete: 140 DDI 364 00
- Motion: 140 DDO 364 00
- Communication: 140 NOE 771 01



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

3.7.2 Configure the Memory Size for the Quantum Processor

The processor memory maps that you viewed in and exported from ProSoft Configuration Builder (PCB) will be imported into the Unity Pro project. These processor State RAM maps are calculated from the starting memory addresses and register counts entered into PCB for the module's input and output data images. For more information on configuring memory addresses in PCB, refer to Configure the Gateway (page 32).

Allocating processor memory to store input and output data is part of the processor configuration process. You should view the memory configuration in the PCB Processor Memory Maps before you begin to allocate memory addresses in Unity Pro.

Some points to keep in mind are:

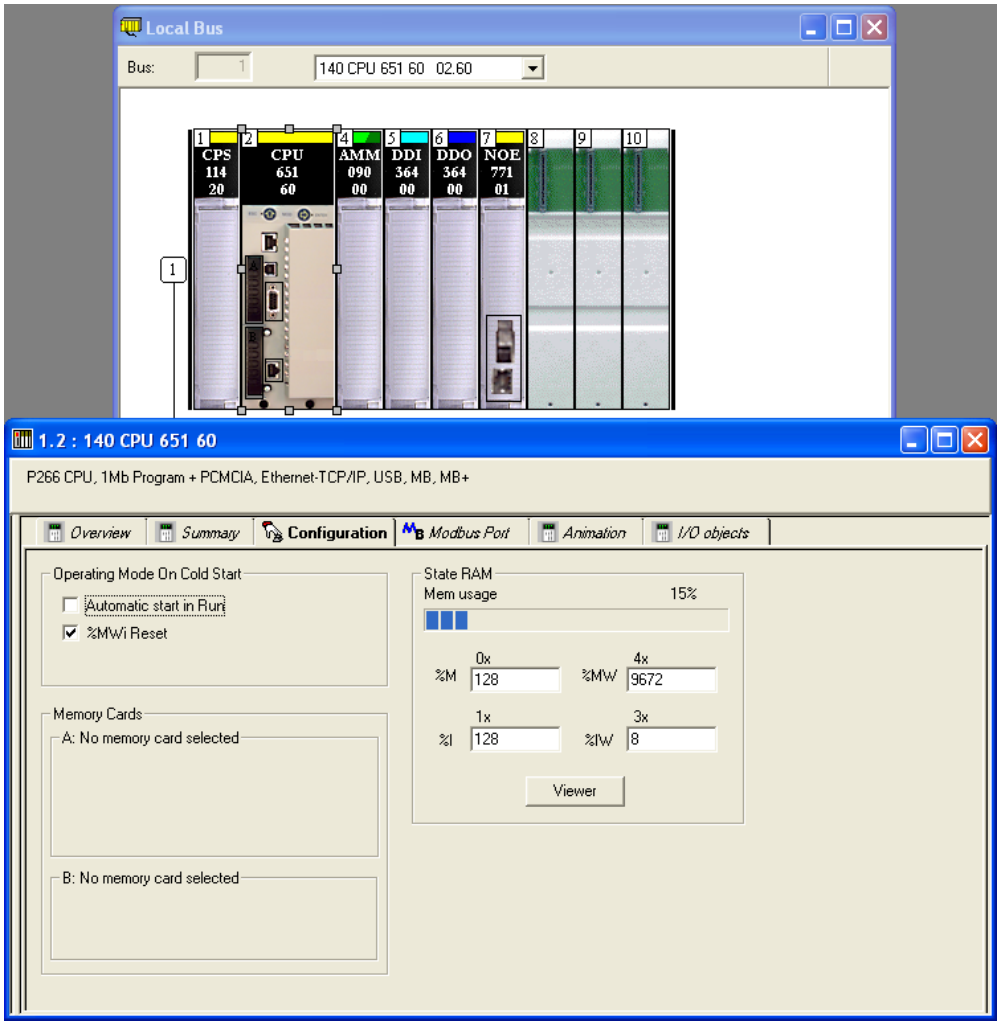
- As the programmer, you must be aware of the memory spaces that are available when deploying in an existing system and assign values to the Modicon processor and 5204SE-MNET-PDPMV1 configurations accordingly.
- The maximum number of 16-bit *%MW* memory registers that can be configured in a Quantum processor vary based on the model. When setting the *%MW* memory allocation, you must allocate enough total memory to accommodate the amount required for the gateway as well as for the rest of your application.
- The total number of data registers allocated for PROFIBUS data must at least equal or exceed the number needed, which can be calculated by taking the starting register configured in PCB and adding the register count configured, plus any additional registers required for the rest of the application process logic. The memory map from PCB can help you determine these numbers.
- The gateway can use up to 768 words of cyclic input data, 768 words of cyclic output data, 76 words of status data, 378 words of standard PROFIBUS slave diagnostic data and up to 2071 words for communication control and data buffers. Therefore, the total *%MW* memory requirement for just the PROFIBUS application could be as much as 4061 words. Round this up to an even 4100 registers as the amount of *%MW* memory to allocate for PROFIBUS data.
- You must allocate at least this much memory space as a continuous, uninterrupted block of processor memory that will not be used by any I/O modules, processes, or variables.

WARNING: Failure to properly map your processor memory will likely cause corruption of PROFIBUS data and can create potentially hazardous situations resulting from unexpected equipment operation; which can result in injury to personnel or damage to equipment.

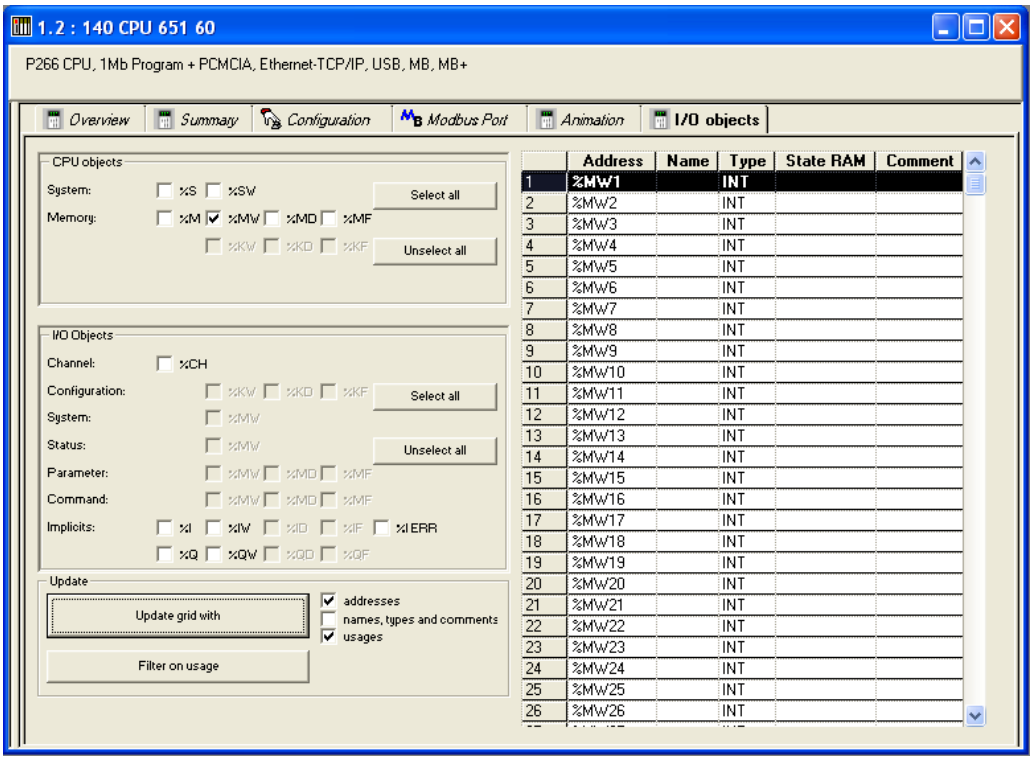
The following steps will help you determine the correct memory addresses to assign.

To view memory usage in the processor:

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



5 To view detailed information about the processor's memory configuration, click the **I/O OBJECTS** tab. These selections offers tools to view the types of data stored at specific addresses in the processor. Make note of memory areas that are already allocated, and select an area of contiguous memory that can be allocated to the gateway.



3.7.3 Import the Quantum Functional Module (.XFM File)

To simplify the task of programming the processor when communicating with the 5204SE-MNET-PDPMV1, the Application Communication Logic functions of ProSoft Configuration Builder (PCB) create a Unity Pro Functional Module (XFM).

Note: The Functional Module is intended only for new installations of the gateway. If you have an existing installation, the following procedure will overwrite your settings, and may cause loss of functionality. DO NOT overwrite a working application until you have thoroughly reviewed the rest of the topics in this manual.

The Functional Module provides easy access to:

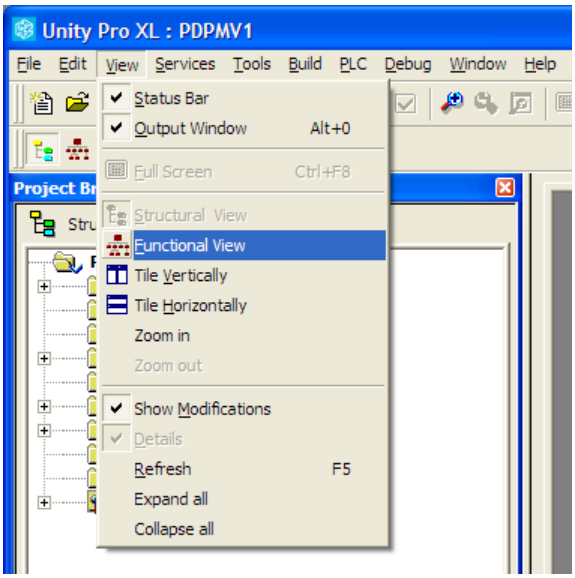
- PROFIBUS DP-V0 or DP-V1 cyclic input and output data
- gateway input/output status data
- Standard PROFIBUS slave diagnostic data (six bytes per slave)
- PROFIBUS DP-V1 acyclic message data, such as "Get Live List", "Get (Extended) Slave Diagnostics", perform Freeze and Sync commands, or perform any slave device-specific commands or functions.

The Functional Module file name matches the gateway name you defined in PCB and will have the extension ".XFM". This file is created by PCB when you export the processor file from the Unity Passthru Memory Map box.

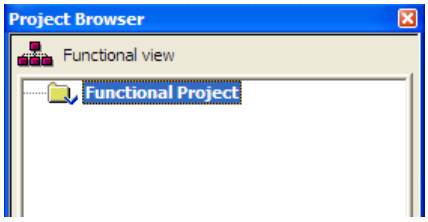
To import the Functional Module:

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

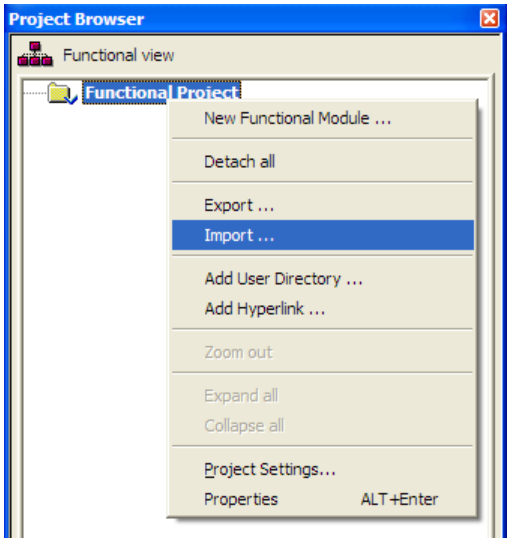
1 Open the **VIEW** menu, and then choose **FUNCTIONAL VIEW.**



This action populates the *Project Browser* with a **FUNCTIONAL PROJECT** icon.

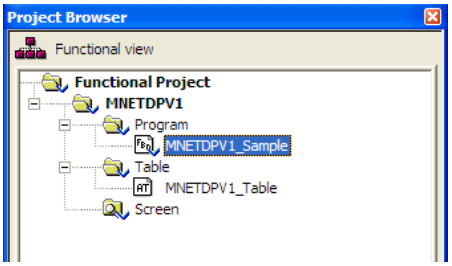


2 Select **FUNCTIONAL PROJECT** and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT.**

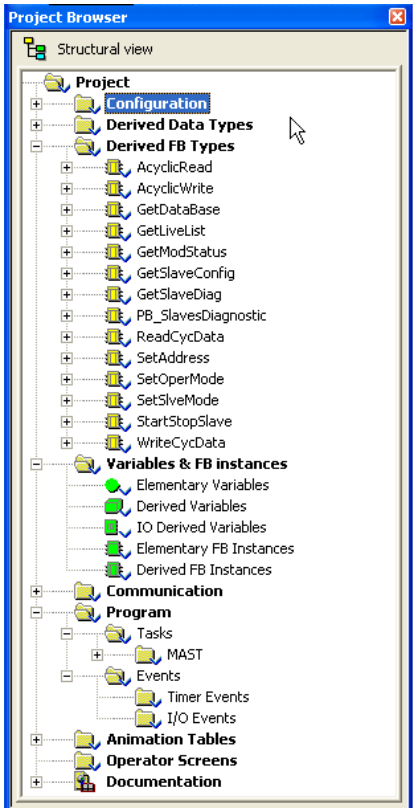


3 In the *Import* dialog box, choose **FUNCTIONAL MODULE (*.XFM)** in the Files of Type dropdown list and then select the XFM file to import. The XFM file name matches the gateway name you defined in PCB and exported (page 47).

Click **IMPORT** to import the file. Notice that the *Project Browser* is now populated with the *Functional Module*.



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



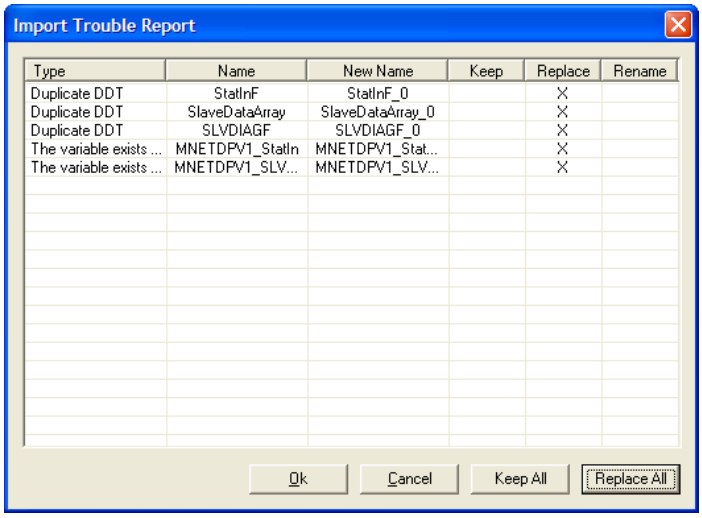
3.7.4 Import the Quantum Variables (.XSY file)

The Application Communication Logic functions of ProSoft Configuration Builder (PCB) also create a list of *variables* and *variable structures* customized to the particular PROFIBUS DP-V1 Master configuration you created. These *variables* are contained in the "{ProjectName}.XSY" file you exported in Export the Unity Pro v 4.0 Logic Support Files (page 47).

The .XSY file contains all the cyclic input and output variables configured by the PCB master configuration software. This file includes gateway status data and will also include slave diagnostics data if the **SLAVE DIAGNOSTICS** parameter was set to Yes.

To import the Variables:

1. In the Project Browser, select **VARIABLES & FB INSTANCES**, and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT**.
2. In the **FILES OF TYPE:** dropdown list, choose **DATA EXCHANGE FILE (*.XSY)**. Select the .xsy file created when you exported the processor files from PCB (page 47) and then click **IMPORT.**
3. If you see an *Import Trouble Report* window, click **REPLACE ALL**, then click **OK.**



At this point, the *DDTs*, *DFBs*, and *Variables* have been imported to the application.

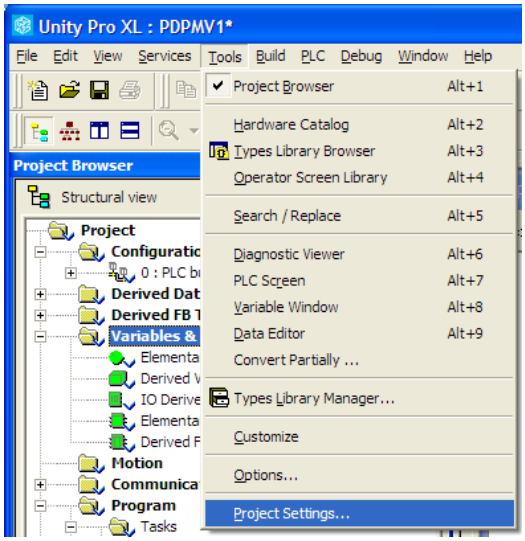
3.7.5 Build the Quantum Project

Whenever you update the configuration of your gateway, the PROFIBUS network, or the processor, you must import the changed configuration from ProSoft Configuration Builder (PCB) 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.

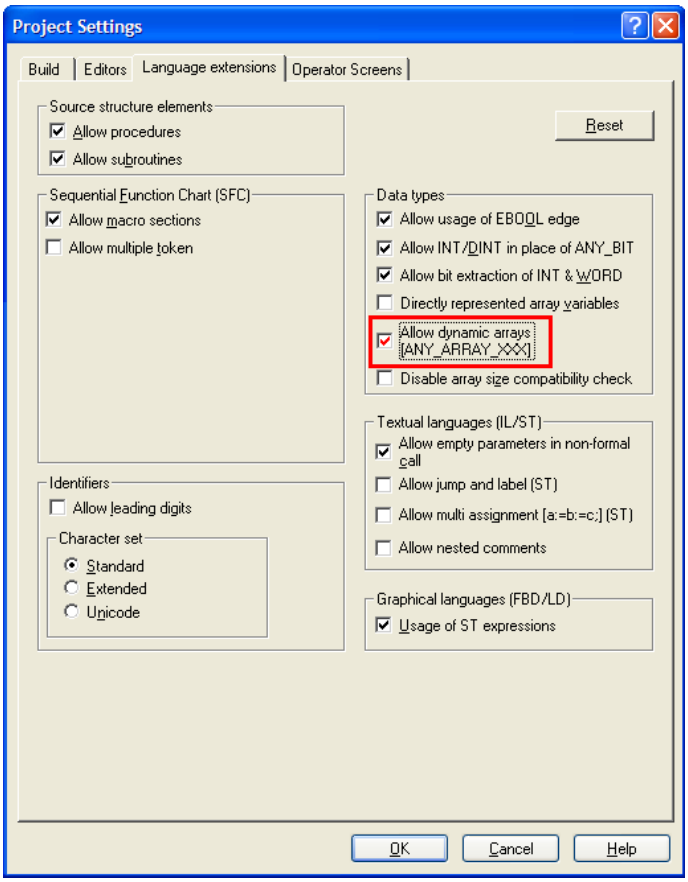
To build (compile) the project:

1. Review the elements of the project in the *Project Browser*.
2. Make sure you have configured sufficient *%MW* memory space for your entire project and PROFIBUS data.
3. To avoid build errors, you will need to enable the **DYNAMIC ARRAY LANGUAGE EXTENSION** option. From the Unity Pro menu bar, select **TOOLS,** and then choose **PROJECT SETTINGS**.



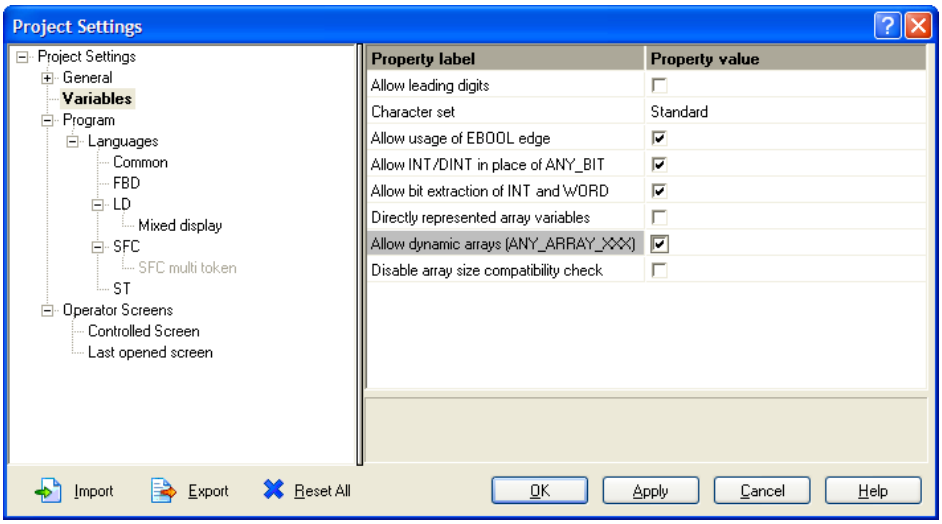
- For UnityPro version 4.0

  In the *Project Settings* box, click the **LANGUAGE EXTENSIONS** tab and select (check) **ALLOW DYNAMIC ARRAYS [ANY_ARRAY_XXX]**.



- For UnityPro version 4.1:

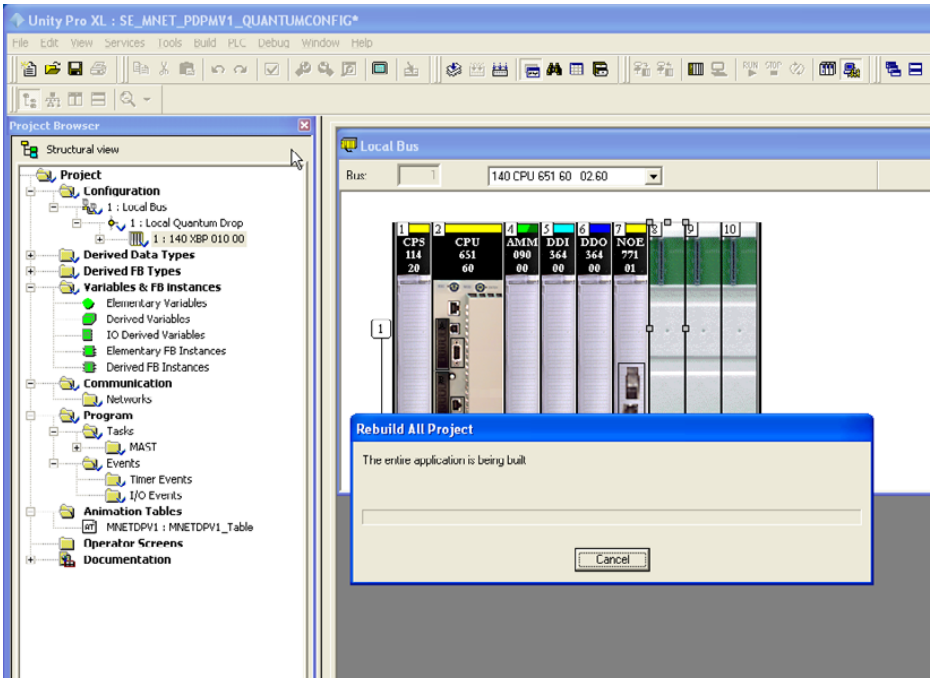
  Select Variables in the left pane, and then select (check) **ALLOW DYNAMIC ARRAYS [ANY_ARRAY_XXX]**



Click **OK** to save your changes and dismiss the dialog box.

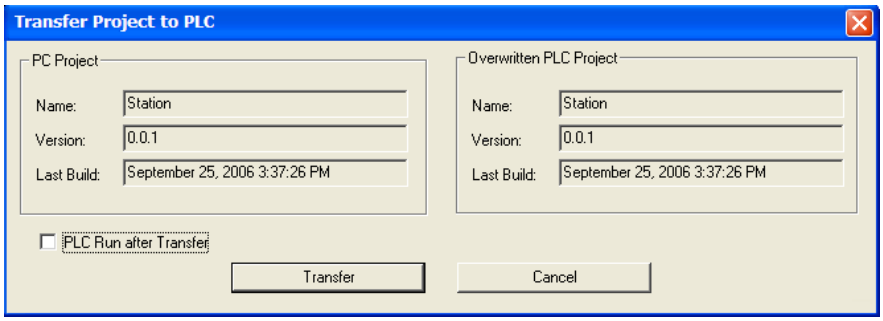
4 When you are satisfied that you are ready to download the project, open the **BUILD** menu, and then choose **REBUILD ALL PROJECT**. This action builds (compiles) the project into a form that the processor can use to execute the instructions in the project file. This task may take several minutes,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

5 As the project is built, Unity Pro displays a Progress dialog box, with details appearing in a pane at the bottom of the window. If you are using the files from PCB and have your memory and processor configuration set up correctly, the project should build without errors. The following illustration shows the build process under way.



3.7.6 Download the Project to the M340 Processor

1. Open the **PLC** menu and then choose **CONNECT.** This action opens a connection between the Unity Pro software and the processor, using the address and media type settings you configured in the previous step.
2. On the **PLC** menu, choose **TRANSFER PROJECT TO PLC**. This action opens the **TRANSFER PROJECT TO PLC** dialog box. If you would like the PLC to go to "Run" mode immediately after the transfer is complete, select (check) the **PLC RUN AFTER TRANSFER** check box.



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

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

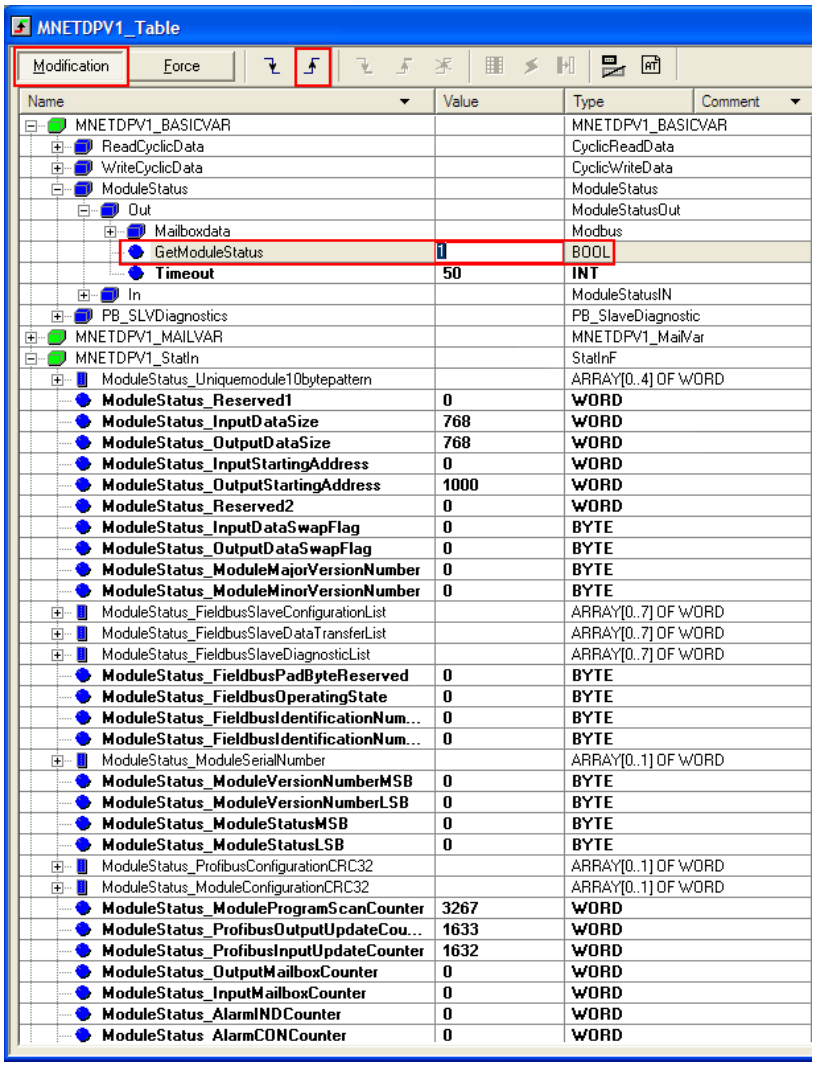
3.7.7 Verify Communication between the Quantum Processor and the Gateway

In this step, you will verify that the processor and the gateway are communicating with each other over the Modbus TCP/IP Ethernet network. The sample project includes an animation table called **MNETDPV1_Table**. When the processor is in **RUN** mode and communicating with the gateway, the values in this animation table are updated whenever you trigger a *GetModuleStatus* read message to request general gateway status from the gateway.

To verify communication between the processor and the gateway:

1. Place the processor in **RUN** mode, if you have not already done so.
2. In the Unity Pro *Project Browser* pane, click **[+]** to open the **ANIMATION TABLES** tree, and then double-click **MNETDPV1_TABLE.**
3. In the *MNETDPV1_Table*, you will see three main *variables* which can be expanded to see many more *sub-variables*.
4. Look for the variable, *MNETDPV1_StatIn*. Click the **[+]** next to it to open it and see all the *sub-variables*. These will include configuration information and version/revision information, which will remain static, as well as program scan counters, input/output update counters, request/response counters, and others *variables* that should change whenever you trigger a status calls.

5 Now, look for and expand the *variable*, **MNETDPV1_BASICVAR.** From there, look for and expand the *sub-variable*, **MODULESTATUS.** Expand the **OUT** *sub-variable*.



Force a one (1) into the sub-variable, *GetModuleStatus***.** Notice that the **MODIFICATION** button must be engaged and you must use the **SET TO 1** icon option to actually have the variable value changed so the update request will be sent.

6 Scroll within *MNETDPV1_StatIn*. Notice that whenever you force the **GETMODULESTATUS** update, the numbers in the **Value** column for items such as *ModuleStatus_ModuleProgramScanCounter* are updated.

4 Reference

In This Chapter

4.1 Basics of Working with Unity Pro

Before launching into descriptions of the *Variables, Derived Data Types (DDTs) and Derived Function Blocks(DFBs)* that are automatically created by the Application Communication Logic functions of ProSoft Configuration Builder (PCB), it might be helpful to give a quick overview of these Unity Pro structures.

Derived Data Types (DDTs) provide the basic building blocks for more complex Unity Pro data structures. They are used by both *Variables* and *Function Blocks* as a way to organize and define the characteristics of individual pieces of data. These characteristics will be shared by all instances of the data type. *DDTs* specify a data item's:

- Structure
- Format
- List of attributes
- Behavior

Variables are the basic data storage unit in Unity Pro programming software. *Variables* allow a processor to hold and manipulate data values using application process logic. *Variables* will be identified by a unique name (sometimes referred to as *symbols*) and will be assigned to hold a particular type of data, like binary or Boolean values (zeros and ones), signed or unsigned integer values of various sizes (8-, 16-, or 32-bit data), floating point values, alpha-numeric strings, arrays (numbered groups of the same type), DDTs, and more.

Variables can be stored in fixed, non-changeable memory locations within the processor's memory. Such *Variables* are called *located variables* and may be referenced by their variable name (symbol) or by their memory address. However, *Variables* are not required to be assigned to specific, fixed memory locations. If *Variables* are not assigned to specific memory addresses, they are called *unlocated variables* and may be referenced only by their unique variable names (symbols).

DDTs and *Variables* are then used to create *Derived Function Blocks (DFBs)*. These *DFBs* have input data, internal storage data, and output data variables and *DDTs* associated with them. They use the data stored in *DDTs* and *Variables* in process logic algorithms, which are also part of the *DFBs*. Unity Pro offers several programming options to create the logic contained in the *DFBs*. All logic in the *DFBs* created by ProSoft Configuration Builder (PCB) is in the form of Structured Text (ST) program language sections, called *Implements* in Unity Pro.

The Application Communication Logic functions built into ProSoft Configuration Builder (PCB) automatically create all the *Variables, DDTs* and *DFBs* required to allow the processor to use its native Modbus TCP/IP communication protocol capability and act as a PROFIBUS DP Master on a PROFIBUS network.

The following sections in this Reference chapter give brief descriptions of the *Variables, DDTs* and *DFBs* created by PCB. Each *DFB* will be described and grouped with its associated *Variables* and *DDTs*.

NOTE: Thorough understanding of all of this reference material is not required to successfully use the gateway. This information is given for those users who wish to have a deeper understanding of the inner workings of the system and to make it easier to troubleshoot potential communication or programming problems.

4.2 Unity Pro Program Objects and Organizing Structures

The following sections outline basic *Variables* and *Derived Data Types (DDTs)* that organize and centralize control of all the custom application logic created by the Application Communication Logic functions of PCB. These variables and structures will be the main ones used to allow the rest of the processor logic and control application to interact with the 5204SE-MNET-PDPMV1 gateway, as well as slaves on the PROFIBUS network. By manipulating values within these main variables, you can perform all the actions required for effective Modbus TCP/IP to PROFIBUS communication.

Based on the configuration options selected, PCB will export one set of files for use with Modicon M340 Programmable Automation Controllers (PACs) and will export a different set of files for use with Modicon Quantum PACs. There are many similarities between these two sets of files. They have the same variable, DDT, and Derived Function Block (DFB) names and have similar data structures. However, Ethernet communication messages are created differently in Unity Pro logic for the M340 and Quantum platforms. Therefore, the .XFM and .XSY export files for M340 and Quantum configurations are mutually exclusive and not interchangeable.

4.3 Modbus TCP/IP Communication Control in M340 and Quantum PACs

Unity Pro programming software (version 4.0 and higher) provides special *Messaging Service Communication Functions* that enable Modicon PACs to accomplish communication using multiple protocols, including Modbus TCP/IP. The special functions provided for the M340 platform are different from those provided for the Quantum platform. The special functions for each platform require similar, but different, input parameters to create a Modbus TCP/IP messages and produce similar, but different, output parameters that hold the results from message responses, if any.

The M340 platform uses the *DATA_EXCH* function for Modbus TCP/IP messaging. PCB Application Communication Logic (ACL) creates a Derived Data Type (DDT), called *Modbus*, that contains all the parameters and structures need to use the *DATA_EXCH* function to send and receive Modbus TCP/IP messages between the M340 processor and the ProLinx gateway.

The Quantum platform uses the *MBP_MSTR* function. PCB ACL creates a DDT, called ControlAndBufferArrays, that contains all the parameters and structures need to use the *MBP_MSTR* function to send and receive Modbus TCP/IP messages between the Quantum processor and the ProLinx gateway.

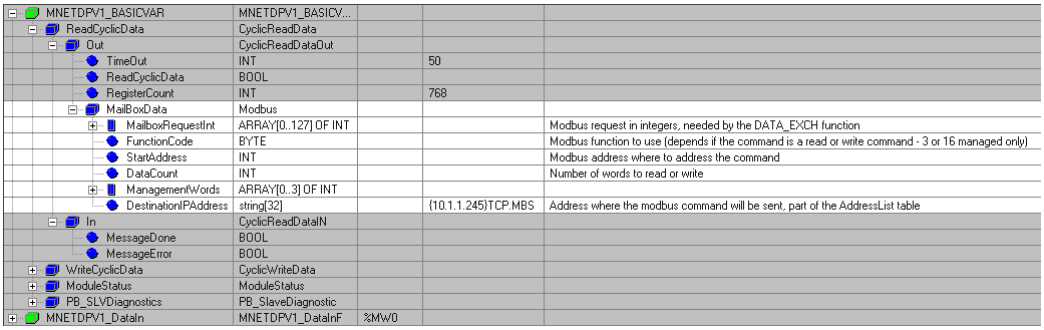
Details about the *DATA_EXCH* and *MBP_MSTR* functions can be found in the Unity Pro Help files. From the Unity Pro Help Index menu, type in *DATA_EXCH* or *MBP_MSTR* in the keyword search window; or, click on the **CONTENTS** tab and look in the path, **UNITY|EF/EFB/DFB LIBRARIES|COMMUNICATION LIBRARY|EXTENDED** to find folders on the *DATA_EXCH* and the *MBP_MSTR* functions.

4.4 Modicon M340 Variables, Derived Data Types, and Derived Function Blocks

4.4.1 M340 Modbus Variables and Derived Data Types (DDTs)

The *Modbus Derived Data Type (DDT)* is a special *DDT* used by all other *DDTs* and *Derived Function Blocks (DFBs)*. This *DDT* holds all the variables needed to create a Modbus TCP/IP Client command which the processor can then transmit to servers on the network. The specific purpose of this *DDT* is to allow any parameters or data needed to create a valid PROFIBUS DP Master request to be passed from the processor to the 5204SE-MNET-PDPMV1 as a Modbus TCP/IP message, which can then be re-transmitted on the PROFIBUS DP network, if required.

You can think of the variables in this structure as temporary holding registers. They will be used by all the other *DDTs* and *DFBs* to build and transmit each required Modbus TCP/IP message. No direct manipulation of these variables should be required or attempted by any other parts of your application code. The other imported *DDTs* and *DFBs* will make all necessary changes to these variables for you, as required. The information here is given for reference only.



MNETDPV1_BASICVAR	MNETDPV1_BASICV...			
ReadCyclicData	CyclicReadData			
Out	CyclicReadDataOut			
TimeOut	INT		50	
ReadCyclicData	BOOL			
RegisterCount	INT		768	
MailBoxData	Modbus			
MailboxRequestInt	ARRAY[0..127] OF INT			Modbus request in integers, needed by the DATA_EXCH function
FunctionCode	BYTE			Modbus function to use (depends if the command is a read or write command - 3 or 16 managed only)
StartAddress	INT			Modbus address where to address the command
DataCount	INT			Number of words to read or write
ManagementWords	ARRAY[0..3] OF INT			
DestinationIPAddress	string[32]		{10.1.1.245}TCP.MBS	Address where the modbus command will be sent, part of the AddressList table
In	CyclicReadDataIN			
MessageDone	BOOL			
MessageError	BOOL			
WriteCyclicData	CyclicWriteData			
ModuleStatus	ModuleStatus			
PB_SLVDiagnostics	PB_SlaveDiagnostic			
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW0		

MNETDPV1_MailVar	<Struct>	
Modbus	<Struct>	
MailboxRequestInt	ARRAY[0..127] OF INT	Modbus request in integers, needed by the DATA_EXCH function
FunctionCode	BYTE	Modbus function to use (depends if the command is a read or write command - 3 or 16 managed only)
StartAddress	INT	Modbus address where to address the command
DataCount	INT	Number of words to read or write
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	Address where the modbus command will be sent, part of the AddressList table
ModuleStatus	<Struct>	

Variable Name	Size/Type	Description
MailboxRequestInt[ ]	128-element 16-bit integer array	Used to hold the raw byte values of a Modbus TCP/IP message as it is being assembled by the active DFB. Once assembly is complete, the contents of this array will be passed to the Unity Pro DATA_EXCH function to be transmitted on the Modbus TCP/IP network as a Client Request Message.
FunctionCode	1 8-bit byte	Will hold a value of 3 for Read Holding Register messages to request data from the 5204SE-MNET-PDPMV1 or a value of 16 for Preset (Write) Multiple Registers to send data to the 5204SE-MNET-PDPMV1
StartAddress	1 16-bit integer	Will hold the address where data will begin to be read from or written to the gateway For Read Commands to bring in PROFIBUS cyclic input data, this address will be in the range of 0-767. For Write Commands that send PROFIBUS cyclic output data, this address will be in the range of 1000-1767. These address ranges are fixed in the gateway's memory and may not be changed or re-configured by the user.
DataCount	1 16-bit integer	Will hold the number of 16-bit registers what will be affected by the command. The maximum allowable count is 125. Most messages will use lower values. A value of 0 is invalid and will cause an error.
ManagementWords[ ]	4-element 16-bit integer array	These are four standard Unity Pro communication message status words. Refer to the topic "Structure of the management parameters" in your Unity Pro documentation for more information.
DestinationIPAddress [ ]	32-character alpha-numeric string array	Since an IP address contains a combination of numbers and period characters, it must be passed to the message functions as a string of alpha-numeric characters. Please see your Unity Pro documentation for the DATA_EXCH and ADDM functions for additional details.

4.4.2 MNETDPV1_BASICVAR Variables and DDTs - M340

These structures hold all the *Variables* and *DDTs* required to send and receive PROFIBUS DP-V0 or DP-V1 cyclic data messages and handle the responses. Cyclic data is all the data coming from and going to slaves on the PROFIBUS network on a regular, recurring cycle. Cyclic data transfers are accomplished at a very rapid, fixed-interval rate in a repeating cycle. The process of completing and repeating these data transfer cycles is called "polling".

As you can see below, there are four major types of cyclic data:

1. *Cyclic input data* - data from PROFIBUS Slaves sent to the Master
2. *Cyclic output data* - data from the PROFIBUS Master sent to the Slaves
3. General Gateway *(Module) Status Data* - created and reported by the gateway. (Although this data is not PROFIBUS protocol-specific data, it is updated along with all the other polling data and, therefore, will be treated as cyclic data by the automatically-created Application Communication Logic *DDTs* and *DFBs*.)
4. PROFIBUS *Slave Diagnostic Data* - the PROFIBUS protocol specifies that each slave send six (6) 8-bit bytes of status and diagnostic data in a fixed format to the Master as part of the regular polling cycle.

All the *DDTs* and *variables* required to use, control, and manage these four types of cyclic data are contained in the *MNETDPV1_BASICVAR* structures and sub-structures.

Cyclic input and output (I/O) data is the data to be transferred based on the PROFIBUS Master/Slave configuration you did in ProSoft Configuration Builder (PCB) when you configured specific amounts of inputs and outputs for each slave on the network.

The *ReadCyclicData* sub-structures handle PROFIBUS cyclic input data. For more information, see DFB Read Cyclic Data (page 103).

The *WriteCyclicData* sub-structures handle PROFIBUS cyclic output data. For more information, see DFB Write Cyclic Data (page 106).

The ModuleStatus sub-structures handle general gateway status data. For more information, see DFB Get Module Status (page 109).

The PB_SLVDiagnostics sub-structures handle the standard PROFIBUS slave diagnostic data. For more information, see DFB Get PROFIBUS Standard Slave Diagnostics (page 111).

4.4.3 MNETDPV1_MailVar Variables and DDTs - M340

These structures hold the all the *Variables* and *DDTs* required to send and receive PROFIBUS DP-V1 acyclic messages, also called *Mailbox Messages*. Note that acyclic messaging is available only on devices using PROFIBUS DP Version 1 or above. PROFIBUS Version 0 devices do not support acyclic messaging.

Acyclic messages are PROFIBUS Master commands that are sent in addition to normal cyclic polling. Acyclic messages are sent at irregular intervals, interspersed in between regular cyclic polling messages. Cyclic polling is deterministic and happens at predictable intervals. Acyclic messaging is not deterministic and not guaranteed to happen at any predictable interval. For this reason, acyclic messages are used for special functions more than for normal data transfer operations.

There are ten major types of acyclic messages supported by the gateway:

1. *Read Acyclic Data* - There are limits to the amount of cyclic input data that can be transferred from PROFIBUS slaves. Some devices can provide more data than can fit within these limits. Acyclic Read messages give the PROFIBUS Master a way to request this additional slave data. For detailed information, see DFB Acyclic Mailbox Message: Read Class 1 Acyclic Data (page 132).
2. *Write Acyclic Data* - There are limits to the amount of cyclic output data that can be transferred to PROFIBUS slaves. Some devices require more data can fit within these limits. Acyclic Write messages give the PROFIBUS Master a way to send this additional data to the slaves. For detailed information, see DFB Acyclic Mailbox Message: Write Class 1 Acyclic Data (page 134).
3. *Get Slave Configuration* - These structures allow the Master to read the actual configuration (identifier bytes) of a specified slave. For detailed information, see DFB Acyclic Mailbox Message: Get Slave Configuration (page 122).
4. *Get* (Extended) *Slave Diagnostic Data* - Some PROFIBUS DP-V1 devices can provide additional diagnostic and alarm data in addition to the six standard diagnostic bytes provided by all slaves. The Get Slave Diagnostic Data message allows the PROFIBUS DP Master to retrieve this extra data from slaves that can provide it. For detailed information, see DFB Acyclic Mailbox Message: Get Slave Diagnostics (page 120).
5. *Get Live List* - A PROFIBUS network can have up to 126 total nodes. The *Live List* is a way for the Master to know which node addresses have active slaves associated with them and which do not. This is a way to see what nodes are 'alive' and 'living' on the network, attached, and ready to transfer data. For detailed information, see DFB Acyclic Mailbox Message: Get Live List (page 118).
6. *Set Slave Address* - For Slaves that support this capability, this structure allows the PROFIBUS DP Master to change the Slave address number of a particular slave. For detailed information, see DFB Acyclic Mailbox Message: Set Address (page 128).

7 *Set Slave Mode* - Some PROFIBUS Slaves support capabilities called *Sync* and *Freeze*. These are special command features which allow a PROFIBUS Master to control when and how a slave updates its internal cyclic inputs and outputs. These structures give the Master the ability to send these special kinds of control messages. For detailed information, see DFB Acyclic Mailbox Message: Set Slave Mode (page 125).

8 *Start/Stop Slaves* - These structures allow the Master to stop or start cyclic data transfers with a slave or or group of slaves. For detailed information, see DFB Acyclic Mailbox Message: Start/Stop Slave (page 124).

9 *Set Operate Mode* - These structures allow the Master to suspend or restart all cyclic polling activity on the network. For detailed information, see DFB Acyclic Mailbox Message: Set Operating Mode (page 116).

10 *Get Database* - These structures allow the Master to obtain and report database configuration information about the PROFIBUS Master hardware. For detailed information, see DFB Acyclic Mailbox Message: Get Database Information (page 130).

All the *DDTs* and *variables* required to use, control, and manage these ten types of acyclic messages are contained in the *MNETDPV1_MAILVAR* structures and sub-structures.

4.4.4 Cyclic I/O Variables, DDTs and DFBs - M340

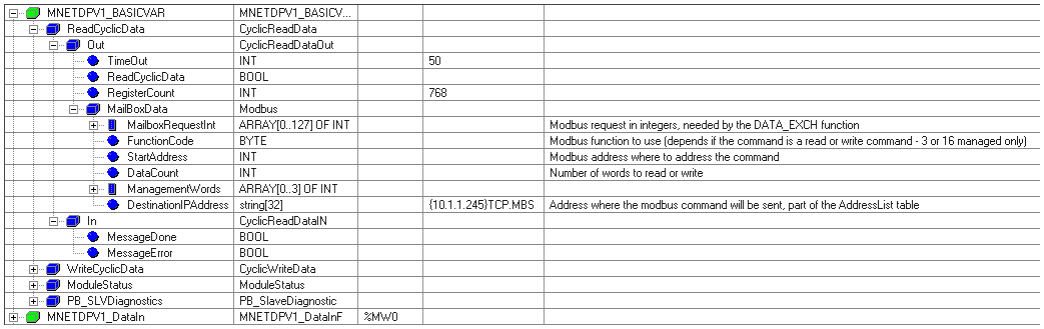
The following five sections provide a more detailed breakdown of the *Variables, DDTs* and *DFBs* used for transferring PROFIBUS cyclic data.

DFB Read Cyclic Data - M340

The *Read Cyclic Data DFB* is used to retrieve PROFIBUS cyclic input data from the 5204SE-MNET-PDPMV1 gateway and bring it back into the processor. This is the data being received by the PROFIBUS DP-V1 Master from the slave or slaves on the PROFIBUS network.

MNETDPV1_BASICVAR_ReadCyclicData Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a request to read PROFIBUS cyclic input data.



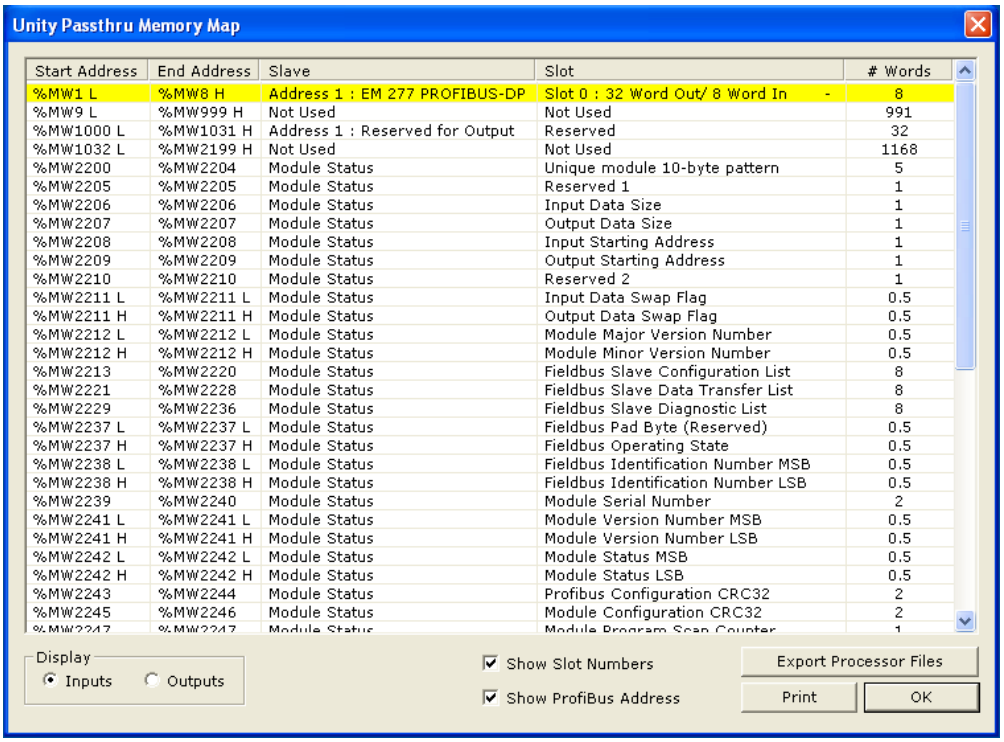
MNETDPV1_BASICVAR	MNETDPV1_BASICV...			
ReadCyclicData	CyclicReadData			
Out	CyclicReadDataOut			
TimeOut	INT		50	
ReadCyclicData	BOOL			
RegisterCount	INT		768	
MailBoxData	Modbus			
MailboxRequestInt	ARRAY[0..127] OF INT			Modbus request in integers, needed by the DATA_EXCH function
FunctionCode	BYTE			Modbus function to use (depends if the command is a read or write command - 3 or 16 managed only)
StartAddress	INT			Modbus address where to address the command
DataCount	INT			Number of words to read or write
ManagementWords	ARRAY[0..3] OF INT			
DestinationIPAddress	string[32]		{10.1.1.245}TCP.MBS	Address where the modbus command will be sent, part of the AddressList table
In	CyclicReadDataIN			
MessageDone	BOOL			
MessageError	BOOL			
WriteCyclicData	CyclicWriteData			
ModuleStatus	ModuleStatus			
PB_SLVDiagnostics	PB_SlaveDiagnostic			
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW0		

Variable Name	Size/Type	Description
Out - Timeout	1 16-bit integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
Out - ReadCyclicData	1 Single-bit Boolean	This is the bit your control and sequencing logic will use to trigger a cyclic read message. Set this bit to one (1) whenever no other messages are active and when you want to update the PROFIBUS cyclic slave input data.
Out - Register Count	1 16-bit integer	Will hold the total number of 16-bit register words of PROFIBUS cyclic input data that need to be read. This value will be the same as what you entered in the PCB configuration for the [PROFIBUS Master DPV1] Input Data Size parameter
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98).
In - Message Done	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.
In - Message Error	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

MNETDPV1_Inputs Variable - M340

This *variable* is an array of 1536 bytes. It is used to receive up to 768 words (1536 bytes) of PROFIBUS cyclic input data from slaves on the PROFIBUS network.

The order of data in this array will match the order in the PCB memory maps you exported and/or printed. The following screen shot shows a typical memory map.



The PCB table lists usage in words rather than bytes, where one (1) word equals two (2) bytes. In this example, there are only eight (8) total words or 16 total bytes of PROFIBUS cyclic input data configured (highlighted in yellow) of the available 1536 bytes that could be used. The 16 bytes (8 words) of PROFIBUS inputs from the device assigned to Slave Address 1 will be stored in the first 16 bytes of this array.

You should also notice that the native storage size in the module's memory is 16-bit or 2-byte word registers. If the number of inputs from the first configured device is an odd number of bytes, you will see that memory register hold one byte from the first device in its low-order byte. The higher-order byte of this register will hold the first byte of data for the next configured slave device. In other words, this data is *byte-packed* with no extra blank bytes inserted just so the data for each slave address can begin on a low-order byte boundary.

If you wish to put gaps into the memory map to give more separation between data blocks from different slave addresses, you may do so in the PROFIBUS Master configuration in PCB by editing the starting address of the data for each slave so that it falls on whatever byte or register address you prefer.

MNETDPV1_DataIn Variables - M340

These variables allow you to take advantage of the *MNETDPV1_DataIn DDT* structure.

MNETDPV1_BASICVAR	MNETDPV1_BASICVAR	
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW1
Slave01Slot00	ARRAY[0..15] OF BYTE	%MW1
MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MailVar	

There is no direct link or logic provided to populate this array with the data received in the *MNETDPV1_Inputs* variable. If you wish to use these variables for your application, you will need to create the logic to link the individual bytes of the *MNETDPV1_Inputs* variable to the word array variables in this structure.

The *%MW* addresses shown are for illustration only. If you decide to use these variables, your application may require that you map them to addresses other than the ones shown. You may assign these variables to any valid *%MW* addresses that exist in your processor configuration.

Sample Procedure for Copying from the MNETDPV1_Inputs array to the MNENTDPV1_DataIn Variables

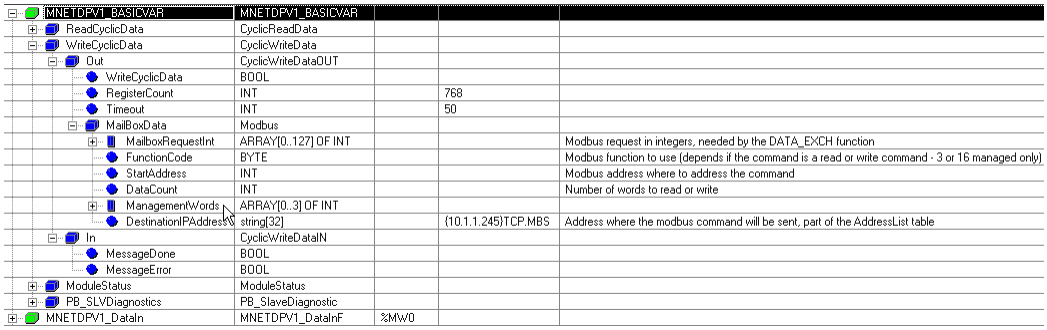
1. Create an INT variable to use as the control variable in a copy loop. This example uses the variable "i".
2. Assign a specific *%MW* address to the variables in the *MNETDPV1_Inputs* variable array. This example uses address *%MW 200*.
3. Assign a specific *%MW* address to the *MNETDPV1_DataIn* variable structure. This example uses address *%MW 1*.
4. Use logic to copy from one set of *%MW* memory addresses to the other for the amount of data you need to copy. In our sample configuration, we have 16 bytes of PROFIBUS cyclic input data. So, the logic needed would look something like this:

```
FOR i:=0 to 15 DO
   %MW1[i]:=%MW200[i] ;
END_FOR ;
```

DFB Write Cyclic Data - M340

MNETDPV1_BASICVAR_WriteCyclicData Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a request to write PROFIBUS cyclic output data.



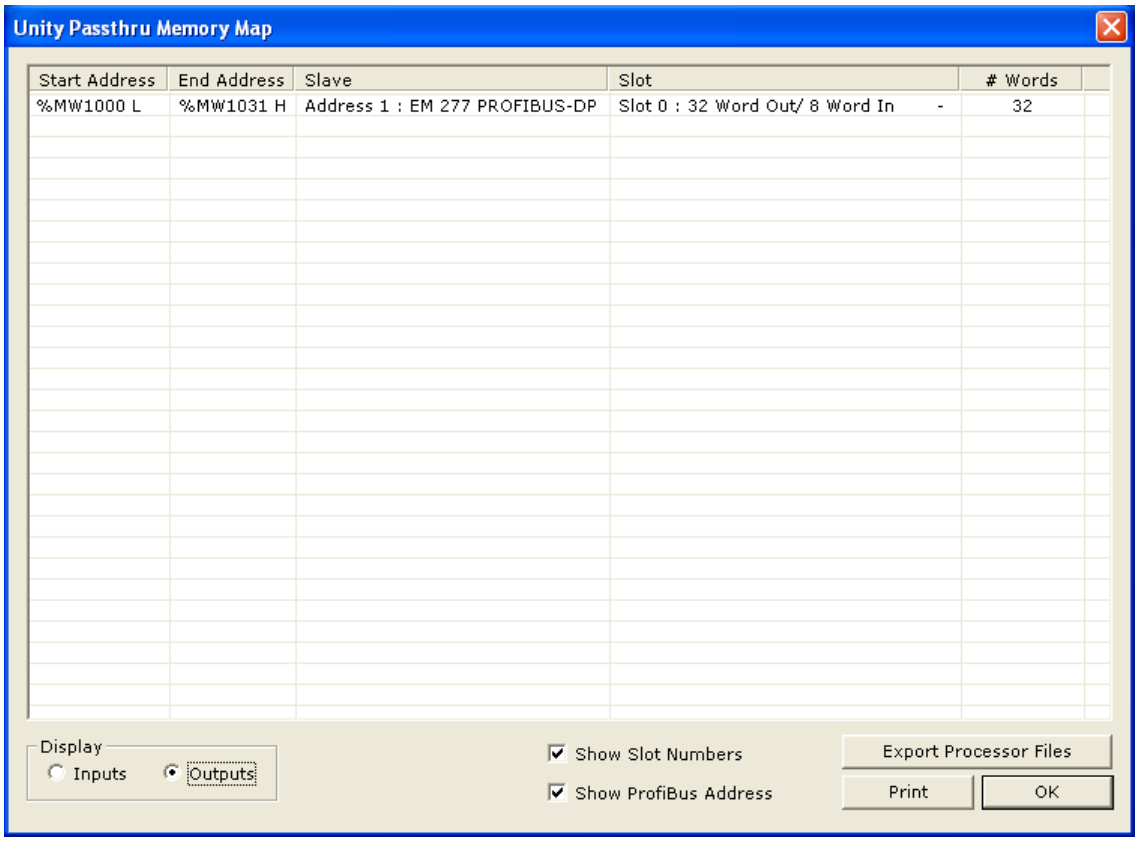
MNETDPV1_BASICVAR	MNETDPV1_BASICVAR			
ReadCyclicData	CyclicReadData			
WriteCyclicData	CyclicWriteData			
Out	CyclicWriteDataOUT			
WriteCyclicData	BOOL			
RegisterCount	INT		768	
Timeout	INT		50	
MailBoxData	Modbus			
MailboxRequestInt	ARRAY[0..127] OF INT			Modbus request in integers, needed by the DATA_EXCH function
FunctionCode	BYTE			Modbus function to use (depends if the command is a read or write command - 3 or 16 managed only)
StartAddress	INT			Modbus address where to address the command
DataCount	INT			Number of words to read or write
ManagementWords	ARRAY[0..3] OF INT			
DestinationIPAddress	string[32]		(10.1.1.245)TCP.MBS	Address where the modbus command will be sent, part of the AddressList table
In	CyclicWriteDataIN			
MessageDone	BOOL			
MessageError	BOOL			
ModuleStatus	ModuleStatus			
PB_SLVDiagnostics	PB_SlaveDiagnostic			
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW0		

Variable Name	Size/Type	Description
Out - WriteCyclicData	1 Single-bit Boolean	This is the bit your control and sequencing logic will use to trigger a cyclic write message. Set this bit to one (1) whenever no other messages are active and when you want to send PROFIBUS cyclic slave output data.
Out - RegisterCount	1 16-bit integer	Will hold the total number of 16-bit register words of PROFIBUS cyclic output data that need to be written. This value will be the same as what you entered in the PCB configuration for the [PROFIBUS Master DPV1] Output Data Size parameter.
Out - Timeout	1 16-bit integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit.
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98).
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic output data memory area in the gateway has been updated.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

MNETDPV1_Outputs Variable - M340

This *variable* is an array of 1536 bytes. It is used to hold up to 768 words (1536 bytes) of PROFIBUS cyclic output data to be sent to slaves on the PROFIBUS network.

The order of data in this array will match the order in the PCB memory maps you exported and/or printed. The following illustration shows a typical memory map.



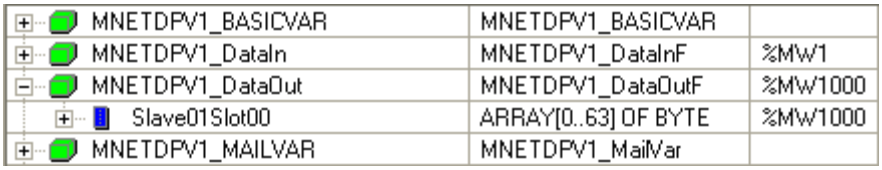
The PCB table lists usage in words rather than bytes, where one (1) word equals two (2) bytes. In this example, there are only 32 total words or 64 total bytes of PROFIBUS output data configured of the 1536 bytes available that could be used. The first 64 bytes (32 words) this array will hold data to be sent to Slave Address 1, Slot 0.

You should also notice that the native storage size in the module's memory is 16-bit or 2-byte word registers. When you have more than one slave device, this data is 'byte-packed' with no extra blank bytes inserted just so the data for each slave address can begin on an even-numbered, low-order byte boundary.

If you wish to put gaps into the memory map to give more separation between data blocks from different slave addresses, you may do so in the PROFIBUS Master configuration in PCB by editing the starting address of the data for each slave.

MNETDPV1_DataOut Variable - M340

These variables allow you to take advantage of the *MNETDPV1_DataOut DDT* structure.



There is no direct link or logic provided to populate data in the *MNETDPV1_Outputs* variable from the data in these variables. If you wish to use these variables for your application, you will need to create the logic to link the variables in this structure to the *MNETDPV1_Outputs* array variable.

The *%MW* addresses shown are for illustration only. If you decide to use these variables, your application may require that you map them to addresses other than the ones shown here. You may assign these variables to any valid *%MW* addresses that exist in your processor configuration.

Sample Procedure for Copying from the MNETDPV1_DataOut variables to the MNENTDPV1_Outputs array

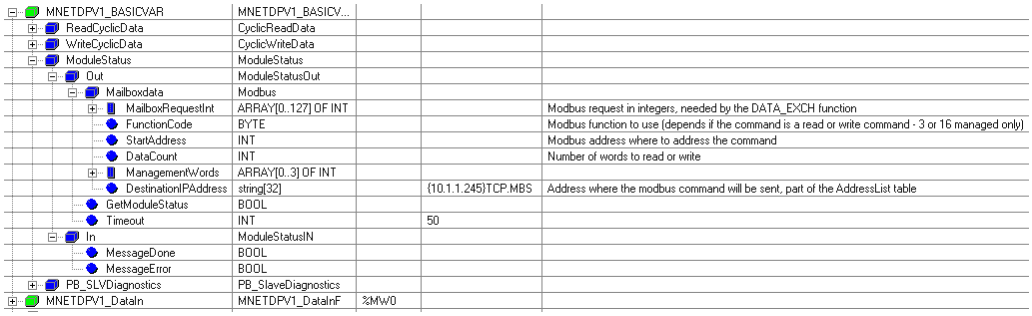
1. Create an INT variable to use as the control variable in a copy loop. This example uses the variable "j".
2. Assign a specific *%MW* address to the variables in the *MNETDPV1_DataOut* variable structure. This example uses address *%MW 1000*.
3. Assign a specific *%MW* address to the *MNETDPV1_Outputs* variable array. This example uses address *%MW 1200*.
4. Use logic to copy from one set of *%MW* memory addresses to the other for the amount of data you need to copy. In our sample configuration, we have 64 bytes of PROFIBUS cyclic output data. So, the logic needed would look something like this:

```
FOR j:=0 to 63 DO
   %MW1000[j]:=%MW1200[j] ;
END_FOR ;
```

DFB Get Module Status - M340

MNETDPV1_BASICVAR_ModuleStatus Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a request to read general gateway status data.



MNETDPV1_BASICVAR	MNETDPV1_BASICV...			
ReadCyclicData	CyclicReadData			
WriteCyclicData	CyclicWriteData			
ModuleStatus	ModuleStatus			
Out	ModuleStatusOut			
Mailboxdata	Modbus			
MailboxRequestInt	ARRAY[0..127] OF INT			Modbus request in integers, needed by the DATA_EXCH function
FunctionCode	BYTE			Modbus function to use (depends if the command is a read or write command - 3 or 16 managed only)
StartAddress	INT			Modbus address where to address the command
DataCount	INT			Number of words to read or write
ManagementWords	ARRAY[0..3] OF INT			
DestinationIPAddress	string[32]		(10.1.1.245)TCP.MBS	Address where the modbus command will be sent, part of the AddressList table
GetModuleStatus	BOOL			
Timeout	INT		50	
In	ModuleStatusIN			
MessageDone	BOOL			
MessageError	BOOL			
PB_SLVDiagnostics	PB_SlaveDiagnostics			
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW0		

Variable Name	Size/Type	Description
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98).
Out - GetModuleStatus	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to get general gateway status data. Set this bit to one (1) whenever no other messages are active and when you want to update the StatInF variable table.
Out - Timeout	1 16-bit integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the general module status data variables have been updated.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

MNETDPV1_StatIn Variables - M340

These variables take advantage of the *StatInF DDT* structure. The *GetStatus DFB* will automatically populate this variable list with general gateway status information received in the Modbus TCP/IP response to the *GetModuleStatus* command.

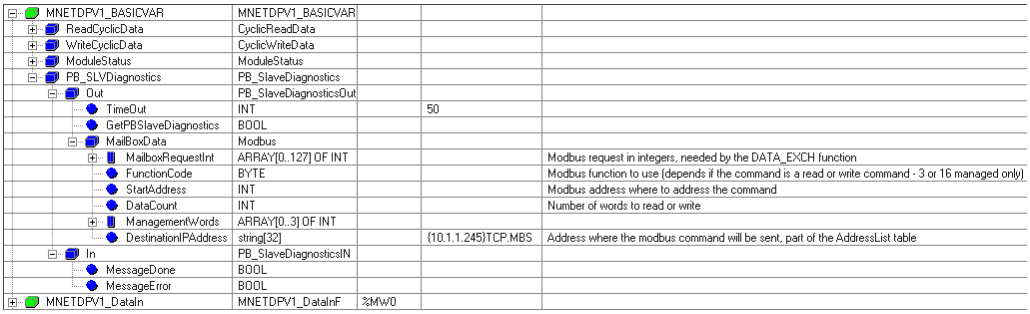
MNETDPV1_StatIn	StatInF	%MW2200
ModuleStatus_Uniquemodule10bytepattern	ARRAY[0..4] OF WORD	%MW2200
ModuleStatus_Reserved1	WORD	%MW2205
ModuleStatus_InputDataSize	WORD	%MW2206
ModuleStatus_OutputDataSize	WORD	%MW2207
ModuleStatus_InputStartingAddress	WORD	%MW2208
ModuleStatus_OutputStartingAddress	WORD	%MW2209
ModuleStatus_Reserved2	WORD	%MW2210
ModuleStatus_InputDataSwapFlag	BYTE	%MW2211
ModuleStatus_OutputDataSwapFlag	BYTE	%MW2211
ModuleStatus_ModuleMajorVersionNumber	BYTE	%MW2212
ModuleStatus_ModuleMinorVersionNumber	BYTE	%MW2212
ModuleStatus_FieldbusSlaveConfigurationList	ARRAY[0..7] OF WORD	%MW2213
ModuleStatus_FieldbusSlaveDataTransferList	ARRAY[0..7] OF WORD	%MW2221
ModuleStatus_FieldbusSlaveDiagnosticList	ARRAY[0..7] OF WORD	%MW2229
ModuleStatus_FieldbusPadByteReserved	BYTE	%MW2237
ModuleStatus_FieldbusOperatingState	BYTE	%MW2237
ModuleStatus_FieldbusIdentificationNumberMSB	BYTE	%MW2238
ModuleStatus_FieldbusIdentificationNumberLSB	BYTE	%MW2238
ModuleStatus_ModuleSerialNumber	ARRAY[0..1] OF WORD	%MW2239
ModuleStatus_ModuleVersionNumberMSB	BYTE	%MW2241
ModuleStatus_ModuleVersionNumberLSB	BYTE	%MW2241
ModuleStatus_ModuleStatusMSB	BYTE	%MW2242
ModuleStatus_ModuleStatusLSB	BYTE	%MW2242
ModuleStatus_ProfibusConfigurationCRC32	ARRAY[0..1] OF WORD	%MW2243
ModuleStatus_ModuleConfigurationCRC32	ARRAY[0..1] OF WORD	%MW2245
ModuleStatus_ModuleProgramScanCounter	WORD	%MW2247
ModuleStatus_ProfibusOutputUpdateCounter	WORD	%MW2248
ModuleStatus_ProfibusInputUpdateCounter	WORD	%MW2249
ModuleStatus_OutputMailboxCounter	WORD	%MW2250
ModuleStatus_InputMailboxCounter	WORD	%MW2251
ModuleStatus_AlarmINDCounter	WORD	%MW2252
ModuleStatus_AlarmCONCounter	WORD	%MW2253
ModuleStatus_AcyclicReadRequestCounter	WORD	%MW2254
ModuleStatus_AcyclicWriteRequestCounter	WORD	%MW2255
ModuleStatus_Reserved3	ARRAY[0..2] OF WORD	%MW2256
ModuleStatus_ModuleFileErrorWordbitmapped	WORD	%MW2259
ModuleStatus_Reserved4	ARRAY[0..5] OF WORD	%MW2260
ModuleStatus_ConfiguredResponseRimeout	WORD	%MW2266
ModuleStatus_Reserved5	ARRAY[0..1] OF WORD	%MW2267
ModuleStatus_MailboxMessagingDBRegister	WORD	%MW2269
ModuleStatus_MailboxMessagingAlarmRegister	WORD	%MW2270
ModuleStatus_SlaveDiagnosticStartReg	WORD	%MW2271
ModuleStatus_StatusStartRegister	WORD	%MW2272
ModuleStatus_MailboxInputQueueMessageCount	WORD	%MW2273
ModuleStatus_MailboxOutputQueueMessageCo...	WORD	%MW2274
ModuleStatus_AlarmqueueMessageCount	WORD	%MW2275

The *%MW* addresses shown are for illustration only. If you decide to use these variables, your application may require that you map them to addresses other than the ones shown here. You may assign these variables to any valid *%MW* addresses that exist in your processor configuration by setting the desired start address in the **PLC STATUS REGISTER START** parameter in the PCB configuration file. This will cause the import files to contain addresses in the range you select and change the values displayed in this array.

DFB Get PROFIBUS Standard Slave Diagnostics - M340

MNETDPV1_BASICVAR_PB_SLVDiagnostics Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a request to read standard PROFIBUS slave diagnostic data.

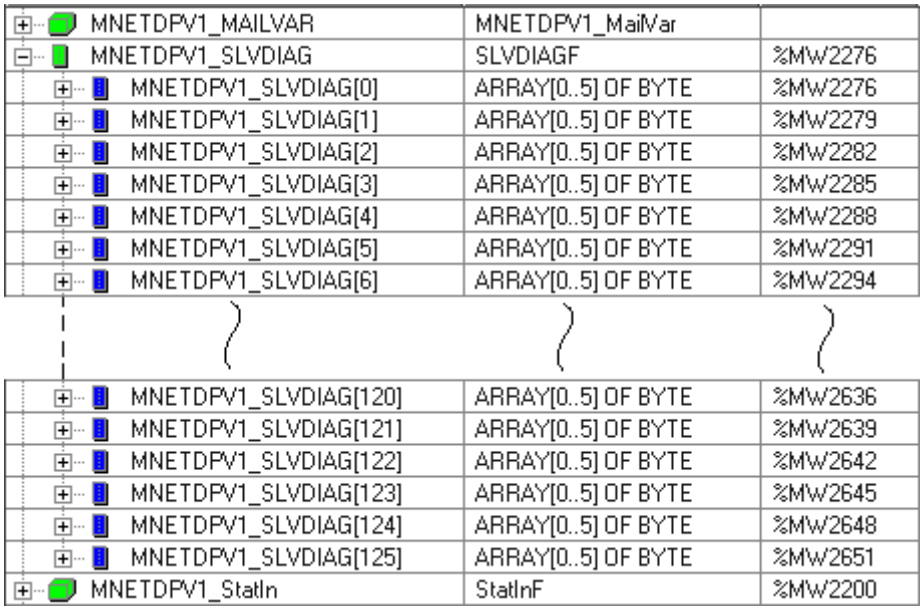


MNETDPV1_BASICVAR	MNETDPV1_BASICVAR			
ReadCyclicData	CyclicReadData			
WriteCyclicData	CyclicWriteData			
ModuleStatus	ModuleStatus			
PB_SLVDiagnostics	PB_SlaveDiagnostics			
Out	PB_SlaveDiagnosticsOut			
TimeOut	INT		50	
GetPBSlaveDiagnostics	BOOL			
MailBoxData	Modbus			
MailboxRequestInt	ARRAY[0..127] OF INT			Modbus request in integers, needed by the DATA_EXCH function
FunctionCode	BYTE			Modbus function to use (depends if the command is a read or write command - 3 or 16 managed only)
StartAddress	INT			Modbus address where to address the command
DataCount	INT			Number of words to read or write
ManagementWords	ARRAY[0..3] OF INT			
DestinationIPAddress	string[32]		(10.1.1.245)TCP.MBS	Address where the modbus command will be sent, part of the AddressList table
In	PB_SlaveDiagnosticsIN			
MessageDone	BOOL			
MessageError	BOOL			
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW0		

Variable Name	Size/Type	Description
Out - TimeOut	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
Out - GetPBSlaveDiagnostics	1 Single-bit Boolean	This is the bit your control and sequencing logic will use to trigger a read message that will retrieve PROFIBUS slave diagnostic data. Set this bit to one (1) whenever no other messages are active and when you want to update the MNETDPV1_SLVDIAG data variables.
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98).
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

MNETDPV1_SLVDIAG Variables - M340

This variable structure is a collection of six-byte arrays. Each array element holds the six bytes of standard PROFIBUS slave data reported to the PROFIBUS Master from each slave that exists on the network as part of the regular cyclic data polling scheme. The array element number corresponds to the node address of each slave.



MNETDPV1_MAILVAR	MNETDPV1_MailVar	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276
MNETDPV1_SLVDIAG[0]	ARRAY[0..5] OF BYTE	%MW2276
MNETDPV1_SLVDIAG[1]	ARRAY[0..5] OF BYTE	%MW2279
MNETDPV1_SLVDIAG[2]	ARRAY[0..5] OF BYTE	%MW2282
MNETDPV1_SLVDIAG[3]	ARRAY[0..5] OF BYTE	%MW2285
MNETDPV1_SLVDIAG[4]	ARRAY[0..5] OF BYTE	%MW2288
MNETDPV1_SLVDIAG[5]	ARRAY[0..5] OF BYTE	%MW2291
MNETDPV1_SLVDIAG[6]	ARRAY[0..5] OF BYTE	%MW2294
MNETDPV1_SLVDIAG[120]	ARRAY[0..5] OF BYTE	%MW2636
MNETDPV1_SLVDIAG[121]	ARRAY[0..5] OF BYTE	%MW2639
MNETDPV1_SLVDIAG[122]	ARRAY[0..5] OF BYTE	%MW2642
MNETDPV1_SLVDIAG[123]	ARRAY[0..5] OF BYTE	%MW2645
MNETDPV1_SLVDIAG[124]	ARRAY[0..5] OF BYTE	%MW2648
MNETDPV1_SLVDIAG[125]	ARRAY[0..5] OF BYTE	%MW2651
MNETDPV1_StatIn	StatInF	%MW2200

The *PB_SlaveDiagnostic DFB* will automatically populate these variables with the diagnostic data returned by the Modbus TCP/IP command. The *%MW* addresses shown are for illustration only. If you decide to use these variables, your application may require that you map them to addresses other than the ones shown here. You may assign these variables to any valid *%MW* addresses that exist in your processor configuration.

4.4.5 Sample Control and Sequencing Logic for Cyclic Data Polling - M340

Here is a structured text (ST) logic example of how you might control and sequence the PROFIBUS cyclic data *DFBs*. You may adapt this sample to fit your application or you may choose to create your own control and sequencing scheme that is more suitable for your specific needs.

For this example, start by creating two variables:

LastExecuted	as type INT
Start	as type BOOL

Then, you can use the following ST logic code. Each time you set the variable *Start* equal to 1, it will begin executing a sequence to read cyclic inputs, write cyclic outputs, get general gateway status, and get standard PROFIBUS slave-specific diagnostic data. As long as *Start* remains equal to 1, this sequence will roll-over and be repeated until interrupted by setting *Start* = 0.

```
IF Start:=1 THEN

   IF MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics=0 AND
      MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus=0 AND
      MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData=0 AND
      MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadCyclicData=0 THEN

      IF LastExcuted=0 THEN
     MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadCyclicData:=1;
         LastExcuted:=1;
      END_IF;
   END_IF;

   IF MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics=0 AND
      MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus=0 AND
      MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData=0 AND
      MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadCyclicData=0 THEN

      IF LastExcuted=1 THEN
         MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData:=1;
     LastExcuted:=2;
      END_IF;
   END_IF;

   IF MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics=0 AND
      MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus=0 AND
      MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData=0 AND
      MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadCyclicData=0 THEN

      IF LastExcuted=2 THEN
         MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus:=1;
     LastExcuted:=3;
      END_IF;
   END_IF;

  IF MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics=0 AND
     MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus=0 AND
     MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData=0 AND
     MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadCyclicData=0 THEN

     IF LastExcuted=3 THEN
        MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics:=1;
    LastExcuted:=0;
       END_IF;
   END_IF;

END_IF;
```

4.4.6 Acyclic Mailbox Message DFBs -M340

These following eleven sections provide information about the *Derived Data Types (DDTs)* and *Variables* associated with each of the ten (10) *Derived Function Blocks (DFBs)* created by the *Application Communication Logic* functions of *ProSoft Configuration Builder (PCB)* that can be used to send PROFIBUS DP-V1 acyclic messaging. Your application-specific control and sequencing logic will use these variables to activate these special functions, if required, as required, and receive any results that may be returned.

The last item for each *DFB* topic is a breakdown of the PROFIBUS acyclic message structure. Creating these messages and handling the responses, if any, will all be done for you by the provided *DFBs*. Since each message has its own unique set of error codes, the primary reason for including this message structure information is to help you interpret any error codes you may receive as a result of activating one of these acyclic messages. The eleventh section contains acyclic message error code information.

DFB Acyclic Mailbox Message: Set Operating Mode - M340

This command allows setting the operating mode of the PROFIBUS Master (STOP, CLEAR, or OPERATE).

MNETDPV1_MAILVAR_SetOperateMode Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a request to set the PROFIBUS Master/network operating mode. Possible choices are:

- **Operate -** Tells the PROFIBUS Master to begin and continue normal cyclic polling and pass acyclic messages, if requested. The network should be in **Stop** mode when you issue this command message.
- **Stop -** Tells the PROFIBUS Master to interrupt normal cyclic polling. Acyclic messaging can still be accomplished when the network is stopped. The network should be in **Operate** mode when you issue this command.
- **Clear -** Tells the PROFIBUS Master to attempt to clear diagnostic errors and re-initialize the PROFIBUS network. The network should be in **Stop** mode when you issue this command.

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperateMode	SetOperateMode	
Out	SetOperateModeOUT	
Operate	BOOL	
Stop	BOOL	
Clear	BOOL	
TimeOut	INT	
MailBoxData	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
In	SetOperateModeIN	
MessageDone	BOOL	
MessageError	BOOL	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - Operate	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to begin and continue normal network cyclic polling and acyclic messaging. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - Stop	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to stop normal network cyclic polling. Some acyclic messaging can still be accomplished in this mode. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - Clear	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to clear slave diagnostic faults and re-initialize the PROFIBUS network. Set this bit to one (1) whenever no other messages are active, the network mode is currently set to STOP, and you want to send this acyclic message.
Out - TimeOut	1 16-bit integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98).
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Get Live List - M340

This acyclic message returns 127 bytes of information about the nodes on the network. Each byte holds the node type for one bus subscriber (node or device). The position of the byte in the response data corresponds to the address (0 to 125) of the node on the network. The content of each byte tells whether the node is a Master or Slave (multiple PROFIBUS Masters may co-exist on the same physical network).

This acyclic message can be sent in all operation modes (STOP, CLEAR, and OPERATE), however the gateway must be initialized properly.

MNETDPV1_MAILVAR_GetLiveList Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to request a list of PROFIBUS network nodes (bus subscribers).

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
Out	GetListOut	
Mailboxdata	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
GetLiveList	BOOL	
Timeout	INT	
In	GetListIn	
StationsStatus	ARRAY[0..127] OF BYTE	
Fault	INT	
ReturnCode	INT	
MessageDone	BOOL	
MessageError	BOOL	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperateMode	SetOperateMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98).
Out - GetLiveList	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master provide a list of active slave nodes. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - Timeout	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
In - StationsStatus[ ]	127 element Byte Array	Each byte will contain one of the following codes indicating type of node present at that node address: 00h: Slave Station 01h: Master Station not yet ready for Token ring (station physically attached to the bus but not configured or polling) 02h: Master Station ready to enter Token ring (station is configured but not polling; there is not yet any Token transmission) 03h: Master Station in Token Ring (Token transmission through the station; station is fully operational) 04h: Station does not exist
In - Fault	1 16-bit Integer	For details on Fault Codes, see Acyclic Message Status Word (page 218).
In - Return Code	1 16-bit Integer	For details, see Return Codes (page 219).
In - Message Done	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - Message Error	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Get Slave Diagnostics - M340

This acyclic message reads extended diagnostic data from a specified slave.

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

MNETDPV1_MAILVAR_GetDiag Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to retrieve extended slave diagnostic data from a specific PROFIBUS network slave.

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
Out	GetDiagnosticOut	
GetSlaveDiagnostic	BOOL	
SlaveAddress	BYTE	
RequestType	BYTE	
TimeOut	INT	
MailBoxData	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
In	GetDiagnosticIn	
ResponseData	ARRAY[0..244] OF BYTE	
Error1	BYTE	
ReturnCode	INT	
Fault	INT	
Length	BYTE	
MessageDone	BOOL	
MessageError	BOOL	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperateMode	SetOperateMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - GetSlaveDiagnostic	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to retrieve extended slave diagnostic data from a specified slave address. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries: 0 - 125 Enter the slave address of the device from which you wish to retrieve extended diagnostic data.
Out - RequestType	1 8-bit Byte	Valid entries: 0 or 1 0 = Get slave extended diagnostic data already stored on the Master. Can be requested only from slaves configured by this Master node. (faster response to the acyclic message; but data may not be current) 1 = Send a special request on the network to read extended diagnostic data directly from the slave at the address specified. Can be requested from any slave on the network. (takes longer to receive response but data is current)
Out - TimeOut	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
Out - MailboxData	Multi-variable Nested Array	For details, see Modbus DDT (page 98)
In - Response Data[ ]	245-element 8-bit Byte Array	For detailed breakdown of the data available in this array, see Get (Extended) Slave Diagnostic Message Structure (page 198). The amount and type of extended diagnostic data returned varies, and depends on the capabilities of the slave device. Refer to the device manufacturer's documentation for slave diagnostic information.
In - Error1	1 8-bit Byte	Error Byte 1 of 4. Bytes 2 - 4 are reserved For details on Error Byte 1, see Acyclic Message Status Word (page 218)
In - ReturnCode	1 16-bit Integer	For details, see Return Codes (page 219)
In - Fault	1 16-bit Integer	If "Invalid Other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. 0001h: Address out of range. 0002h: Incorrect "Request Type" 000Ah: Failed to read diagnostic data from slave. Refer to Return Codes for additional fault information. 000Bh: Remote station failure. For additional fault information, refer to Return Codes 00FEh: Command not possible; gateway operates as a Class 2 master only. 00FFh: Gateway offline (not initialized or no valid database).

Variable Name	Size/Type	Description
In - Length	1 8-bit Byte	Number of diagnostic bytes returned by the slave and help in the *In-Response Data[ ]* variable array
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Get Slave Configuration - M340

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

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

MNETDPV1_MAILVAR_GetConfig Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to retrieve configuration information from a specific PROFIBUS network slave.

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
Out	GetSlaveConfigurationOUT	
GetSlaveConfiguration	BOOL	
SlaveAddress	BYTE	
Timeout	INT	
MailBoxData	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
In	GetSlaveConfigurationIN	
Error1	BYTE	
Length	BYTE	
Fault	INT	
ReturnCode	INT	
ResponseData	ARRAY[0..244] OF BYTE	
MessageDone	BOOL	
MessageError	BOOL	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperateMode	SetOperateMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - GetSlaveConfiguration	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to retrieve slave configuration information from the specified slave address. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries: 0 - 125 Enter the slave address of the device from which you wish to retrieve slave configuration data.
Out - Timeout	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
Out - MailboxData	Multi-variable Nested Array	For details, see Modbus DDT (page 98)
In - Error1	1 8-bit Byte	Error Byte 1 of 4. Bytes 2 - 4 are reserved For details on Error Byte 1, see Acyclic Message Status Word (page 218)
In -Length	1 8-bit Byte	Range 6 to 244 Number of bytes sent by the slave as response data.
In - Fault	1 16-bit Integer	If "Invalid other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Acyclic Message Status Word 0001h: Address out of range. 000Ah: Failed to execute request. Refer to Return Codes for additional information. 000Bh: Remote station failure. Refer to Return Codes for additional information. 00FFh: gateway not initialized.
In - ReturnCode	1 16-bit Integer	For details, see Return Codes (page 219)
In - ResponseData[ ]	245-element 8-bit Byte Array	For detailed breakdown of the data available in this array, see Get Slave Configuration Message Structure (page 200). Response Data size will be from 6 to 244 bytes. Actual amount and type of data returned varies and depends on the capabilities of the slave device. Please see the device manufacturer's documentation for details.
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Start/Stop Slave - M340

This acyclic message stops or starts a selection of slaves. Stopping a slave or group of slaves removes them from the normal cyclic data polling cycle. Starting a slave or group of slaves returns them to the normal polling cycle.

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

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

MNETDPV1_MAILVAR_StartStopSlaves Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a PROFIBUS Master acyclic message to start or stop one or more slaves.

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
Out	StartStopSlaveOUT	
StopSlaves	BOOL	
StartSlaves	BOOL	
TimeOut	INT	
SlaveNumber	ARRAY[0..125] OF BYTE	
Mailboxdata	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
In	StartStopSlaveIN	
MessageDone	BOOL	
MessageError	BOOL	
SetOperateMode	SetOperateMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - StopSlaves	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to send Stop acyclic message to all slaves which have their Out-SlaveNumber array element set to 1. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - StartSlaves	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to send Start acyclic message to all slaves which have their Out-SlaveNumber array element set to 1. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - TimeOut	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
Out - SlaveNumber	126-element 8-bit Byte Array	Enter 0 or 1 for each array element before you trigger the Start or Stop acyclic message 0 = Do not change this node (ignore acyclic message for this node) 1 = Change the state of this node to Stop for Stop acyclic message or Start for Start acyclic message (acyclic message affects this node)
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98)
In - Message Done	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - Message Error	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Set Slave Mode - M340

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

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

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

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

For additional details, see Set Slave Mode Message Structure (page 206).

MNETDPV1_MAILVAR_SetSlaveMode Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a request to the PROFIBUS Master to send Sync and Freeze control messages to a slave or group of slaves.

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
Out	SetSlaveModeOUT	
Timeout	INT	
SetSlaveMode	BOOL	
SlaveAddress	BYTE	
GroupSelect	BYTE	
CommandControl	BYTE	
MailBoxData	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
In	SetSlaveModeIN	
MessageDone	BOOL	
MessageError	BOOL	
StartStopSlaves	StartStopSlaves	
SetOperateMode	SetOperateMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - Timeout	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
Out - SetSlaveMode	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to change the operating mode of a single slave or group of slaves. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries: 0-125, 127 If you enter a value from 0 to 125, the acyclic message will affect only the one slave device at this address. If you enter a value of 127, the acyclic message will affect a group of slaves, specified by the Out-GroupSelect parameter.
Out - GroupSelect	1 8-bit Byte	For details on how to set this parameter, see Set Slave Mode Message Structure (page 206)
Out - CommandControl	1 8-bit Byte	For details on how the set this parameter, see Set Slave Mode Message Structure (page 206)
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98)
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Set Address - M340

This acyclic message makes it possible to set the node address of a specified slave, provided that the slave supports this feature.

NOTE: The message data size depends on the actual slave implementation, range 0-240 bytes.

MNETDPV1_MAILVAR_SetSlaveAdd Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to change the network node address of a slave that supports this feature.

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
Out	SetAddressOut	
SetSlaveAddress	BOOL	
CurrentSlaveAddress	BYTE	
NewSlaveAddress	BYTE	
NoAddressChange	BYTE	
SlaveIdentNumber	INT	
TimeOut	INT	
MailboxData	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
In	SetAddressIn	
Error1	BYTE	
Length	BYTE	
ReturnCode	INT	
Fault	INT	
ConfigurationData	ARRAY[0..244] OF BYTE	
MessageDone	BOOL	
MessageError	BOOL	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperateMode	SetOperateMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - SetSlaveAddress	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to send a new network node number to a specific slave, thereby changing its node address. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - CurrentSlaveAddress	1 8-Bit Byte	Valid entries: 0 - 126 Set this variable to the current address of the slave whose address you wish to change.
Out - NewSlaveAddress	1 8-Bit Byte	Valid entries: 0 - 126 Set this variable to the new address you wish the slave to have.
Out - NoAddressChange	1 8-Bit Byte	Valid entries: 00h Change of address is still possible at a later stage. 01h - FFh Change or address is not possible until after slave reset This parameter specifies whether the slave address can be changed again at a later stage. if this is not allowed, then it is only possible to change the address with this function after initial reset. After the initial reset, the slave takes the default address 126.
Out - SlaveIdentNumber	1 16-bit Integer	Unique PROFIBUS Slave Identifier, assigned by PROFIBUS User Organization
Out - TimeOut	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98)
In - Error1	1 8-bit Byte	Error Byte 1 of 4. Bytes 2 - 4 are reserved For details on Error Byte 1, see Acyclic Message Status Word (page 218)
In - Length	1 8-bit Byte	Range 6 to 244 Number of bytes sent by the slave as response data, if any.
In - ReturnCode	1 16-bit Integer	Refer to Return Codes (page 219) for additional information.
In - Fault	1 16-bit Integer	If "Invalid other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Acyclic Message Status Word (page 218) 0001h: Address out of range. 000Ah: Failed to execute request. Refer to Return Codes (page 219) for additional information. 000Bh: Remote station failure. Refer to Return Codes (page 219) for additional information. 00FFh: gateway not initialized.

Variable Name	Size/Type	Description
In - ConfigurationData	245-element 8-bit Byte Array	Additional message data that may be returned by the slave
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Get Database Information - M340

This acyclic message fetches information about the stored database (user-specific data that was downloaded to the gateway in message data bytes 1 to 32 via mailbox "FB_APPL_END_DATABASE_DOWNLOAD" or from the configuration tool).

This message also returns information about the amount of allocated I/O data space.

MNETDPV1_MAILVAR_GetDataBase Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a request to retrieve information about the PROFIBUS database.

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperateMode	SetOperateMode	
GetDataBase	GetDataBases	
Out	GetDBaseOUT	
Getdatabase	BOOL	
Mailboxdata	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
Timeout	INT	
In	GetDBaseIN	
Fault	INT	
InputLength	INT	
OutputLength	INT	
InputSize	INT	
OutputSize	INT	
DatabaseInformations	ARRAY[0..3] OF INT	
MessageDone	BOOL	
MessageError	BOOL	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - Getdatabase	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to retrieve information about the PROFIBUS database. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98)
Out - Timeout	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit
In - Fault	1 16-bit Integer	If 'Invalid Other' is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found in this variable. 0001h - No database in FLASH memory, or download is in progress.
In - InputLength	1 16-bit Integer	Total Input Length Sum of all Input lengths for all slaves in the database, in bytes
In - OutputLength	1 16-bit Integer	Total Output Length Sum of all Output lengths for all slaves in the database, in bytes
In - InputSize	1 16-bit Integer	Required Initialization Input size for the current database. If the slaves are located in a contiguous block, this size is the same as the Total Input Length.
In - OutputSize	1 16-bit Integer	Required Initialization Output size for the current database. If the slaves are located in a contiguous block, this size is the same as the Total Output Length.
In – Database Informations	4-element 16-bit Integer Array	This array holds two 32-bit values which are the PROFIBUS and Module CRC32 checksums. These values also appear in the gateway Status variables and in ProSoft Configuration Builder (PCB).
In - MessageDone	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - MessageError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Read Class 1 Acyclic Data - M340

Some, but not all, PROFIBUS DP-V1 slaves are capable of providing more data that can be configured for normal cyclic polling. Acyclic data read messages allow the PROFIBUS Master to request this additional data from slaves that can provide it. This acyclic message initiates a PROFIBUS DP-V1 *Class 1 Acyclic Read Request*. Refer to protocol specification EN50170 (DP-V1) for more information about this type of acyclic message.

MNETDPV1_MAILVAR_AcyclicRead Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a request to read PROFIBUS *DP-V1 Class 1 Acyclic Data*.

MNETDPV1_MAILVAR	MNETDPV1_MailVar	
AcyclicRead	ReadAcyclicData	
Out	ReadAcyclicDataOut	
ReadAcyclicData	BOOL	
SlaveAddress	BYTE	
Slot	BYTE	
Index	BYTE	
Length	BYTE	
MailBoxData	Modbus	
MailboxRequestInt	ARRAY[0..127] OF INT	
FunctionCode	BYTE	
StartAddress	INT	
DataCount	INT	
ManagementWords	ARRAY[0..3] OF INT	
DestinationIPAddress	string[32]	
TimeOut	INT	
In	ReadAcyclicDataIn	
ErrorDecode	BYTE	
Error1	BYTE	
Error2	BYTE	
Length	BYTE	
ExtendedFault	INT	
Fault	INT	
AcyclicData	ARRAY[0..242] OF BYTE	
MessageDone	BOOL	
MessageError	BOOL	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperateMode	SetOperateMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - ReadAcyclicData	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to read PROFIBUS DP-V1 Acyclic Data from a specific slave on the network. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries 0 - 125 Enter the node address of the slave device from which you wish to receive Acyclic Data.
Out - Slot	1 8-bit Byte	Valid entries 0 - n (where 'n' is the highest configured slot number on the target slave address - depends on device configuration.) Enter the slot number on the target node from which you wish to retrieve the acyclic data.
Out - Index	1 8-bit Byte	See slave device manufacturer for valid entries. This parameter is used to address the desired data block in the target slave.
Out - Length	1 8-bit Byte	See slave device manufacturer for valid entries. This parameter specifies the number of bytes of the data block that has to be read. If the slave data block length is less than requested, the length of the response will be the actual length of the data block. If the slave data block is greater or equal, then the response will contain the same amount of data.
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98)
Out - TimeOut	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit.
In - ErrorDecode	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation, or the EN50170 (DP-V1) protocol specification.
In - Error1	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation, or the EN50170 (DP-V1) protocol specification.
In - Error2	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation, or the EN50170 (DP-V1) protocol specification.
In - Length	1 8-bit Byte	This value specifies the number of valid acyclic read data bytes returned by the slave.
In - Extended Fault	1 16-bit Integer	Refer to Read Class 1 Acyclic Data Message Structure (page 214)
In - Fault	1 16-bit Integer	Refer to Read Class 1 Acyclic Data Message Structure (page 214)
In - AcyclicData[ ]	243-element 8-bit Byte Array	This array will hold the actual acyclic read data bytes returned by the slave.

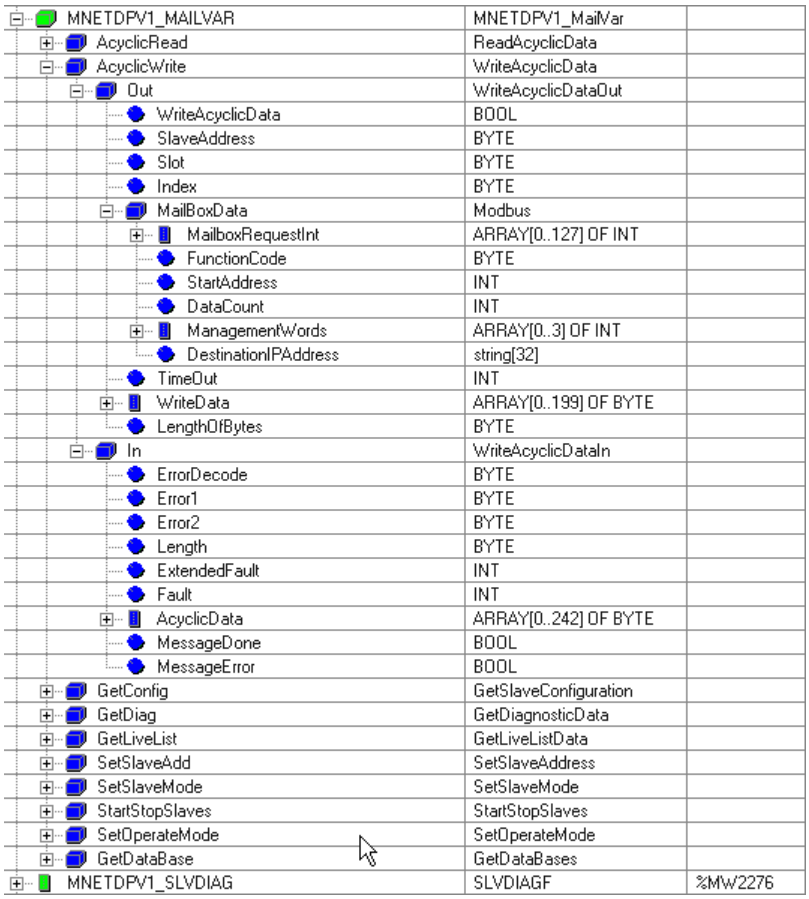
Variable Name	Size/Type	Description
In - Message Done	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - Message Error	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

DFB Acyclic Mailbox Message: Write Class 1 Acyclic Data - M340

Some, but not all, PROFIBUS DP-V1 slaves are capable of receiving more data that can be configured for normal cyclic polling. Acyclic data write messages allow the PROFIBUS Master to send this additional data to slaves that can receive it. This acyclic message initiates a PROFIBUS *DP-V1 Class 1 Acyclic Write Request*. Refer to PROFIBUS DP-V1 specification EN50170 (DP-V1) for more information.

MNETDPV1_MAILVAR_AcyclicWrite Variables - M340

This variable structure is the one to use with your control and sequencing logic when you want to send a PROFIBUS *DP-V1 Class 1 Acyclic Write Request*.



Variable Name	Size/Type	Description
Out - WriteAcyclicData	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to write PROFIBUS DP-V1 Acyclic Data to a specific slave on the network. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries 0 - 125 Enter the node address of the slave device to which you wish to send Acyclic Data.
Out - Slot	1 8-bit Byte	Valid entries 0 - n (where 'n' is the highest configured slot number on the target slave address - depends on device configuration.) Enter the slot number on the target node to which you wish to send the acyclic data.
Out - Index	1 8-bit Byte	See slave device manufacturer for valid entries. This parameter is used to address the desired data block in the target slave.
Out - MailBoxData	Multi-variable nested DDT	For details, see Modbus DDT (page 98)
Out - TimeOut	1 16-bit Integer	Used to hold the amount of time in milliseconds to wait for a response to the Modbus TCP/IP command before assuming a communication error has occurred and setting the In-MessageError status bit.
Out - WriteData[ ]	200-element 8-bit Byte Array	This array will be used to hold the acyclic data, in bytes, that you wish to write to the target slave.
Out - LengthOfBytes	1 8-bit Byte	Valid entries: 1 - 200 Enter the number of acyclic data bytes you wish to write to the target slave.
In - ErrorDecode	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation or the EN50170 (DP-V1) protocol specification.
In - Error1	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation or the EN50170 (DP-V1) protocol specification.
In - Error2	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation or the EN50170 (DP-V1) protocol specification.
In - Length	1 8-bit Byte	The value seen here specifies the number of valid data bytes returned by the slave, if any.
In - Extended Fault	1 16-bit Integer	For details, see Write Class 1 Acyclic Data Message Structure (page 216)
In - Fault	1 16-bit Integer	For details, see Write Class 1 Acyclic Data Message Structure (page 216)

Variable Name	Size/Type	Description
In - AcyclicData[ ]	243-element 8-bit Byte Array	This array will hold any acyclic data bytes that may be returned by the slave. Please consult the slave manufacturer's documentation for information about any data you may find here.
In - Message Done	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received.
In - Message Error	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.

4.5 Modicon Quantum Variables, Derived Data Types and Derived Function Blocks

4.5.1 Quantum Communication Control and Data Buffer Variables and Derived Data Types (DDTs)

The *MBP_MSTR* function used by the Quantum platform performs a similar function to that of the *DATA_EXCH* function in the M340 platform. However, the *MBP_MSTR* function uses a different structure which requires the use of two variable arrays: a data buffer array, called *BufferArray*, and a control parameter array, called *ControlArray*.

To make it more convenient for you, the two DDTs and variable arrays have been combined into one mid-level structure, called *ControAndBufferArrays DDT*.

BufferArray	ARRAY[1..100] OF INT
ControlAndBufferArrays	<Struct>
Control	ControlArray
Control[1]	INT
Control[2]	INT
Control[3]	INT
Control[4]	INT
Control[5]	INT
Control[6]	INT
Control[7]	INT
Control[8]	INT
Control[9]	INT
Buffer	BufferArray
ControlArray	ARRAY[1..9] OF INT
CyclicReadData	<Struct>

The *BufferArray* is used to hold only the data to be sent or the data received in the Modbus message created by the *MBP_MSTR* function. The *BufferArray* is an array of 100, 16-bit integer registers. Since this array holds only the data contained in a Modbus message, the significance of the values in this array depends on how the data is to be used in the application.

The *ControlArray* holds all the parameters needed to determine the type of message to be sent (read or write message), the amount of data to transfer in the message (register count), and the address of the device that will be the message target (the IP address of the ProLinx gateway). This array has a fixed structure, which each element contains the value for a specific *MBP_MSTR* parameter. Here are the definitions of the parameters held in the *ControlArray*.

Variable Name	Size/Type	Description
Control[1]	1 16-bit Integer Array Element	Holds the *MBP_MSTR* Block Function Code for the communication operation to be performed. 1 = Write Operation 2 = Read Operation

Variable Name	Size/Type	Description
Control[2]	1 16-bit Integer Array Element	Holds the *MBP_MSTR* Block Error Status A value will be placed in this register after the message has completed, failed, or timed out. 0 = message completed successfully Any non-zero value indicates an unsuccessful message attempt. For details, see *MBP_MSTR* Error Codes
Control[3]	1 16-bit Integer Array Element	Holds the length (number of 16-bit registers) to be sent or requested by the *MBP_MSTR* Block.
Control[4]	1 16-bit Integer Array Element	Holds *MSTR* operation-dependent information. Typically, this will be used to hold the starting register address in the target device where data will be written by a write command or read by a read command.
Control[5]	1 16-bit Integer Array Element	Holds *MBP_MSTR* Routing Register information. Used to specify a message source node. Most Significant Byte (MSB) = Source Node Address, which could be the slot number of an NOE module or a Quantum processors built-in Ethernet port. Use a default value of **254** (16#**FE00** hex) to use a Quantum processor's built-in Ethernet port. Least Significant Byte (LSB) = MBP on Ethernet Transporter (MET) mapping index. Use a default value of **0** to use a Quantum processor's built-in Ethernet port. For more details, see Unity Pro Help index: "MBP_MSTR, Ethernet (Quantum)" - Network Control Block Structures -Control Block for TCP/IP Ethernet
Control[6]	1 16-bit Integer Array Element	Byte 4, MSB or first octet of the ProLinx gateway destination IP address Example: the value **192** for IP address **192**.168.0.100
Control[7]	1 16-bit Integer Array Element	Byte3, second octet of the ProLinx gateway destination IP address Example: the value **168** for IP address 192.**168**.0.100
Control[8]	1 16-bit Integer Array Element	Byte2, third octet of the ProLinx gateway destination IP address Example: the value **0** for IP address 192.168.**0**.100
Control[9]	1 16-bit Integer Array Element	Byte1, LSB last octet of the ProLinx gateway destination IP address Example: the value **100** for IP address 192.168.0.**100**

Quantum MBP_MSTR TCP/IP Ethernet Error Codes

An error in an *MSTR* routine via TCP/IP Ethernet may produce one of the following errors in the *MSTR* control block. The error code appears as:

Mmss

where: **M** is the high code

m is the low code

ss is a subcode

Hexadecimal Error Codes For TCP/IP Ethernet Message Errors:

Hex Error Code	Meaning
1001	Abort by user
2001	An operation type that is not supported has been specified in the control block
2002	One or more control block parameters were modified while the *MSTR* element was active (this only applies to operations which require several cycles for completion). Control block parameters my only be modified in inactive *MSTR* components.
2003	Invalid value in the length field of the control block
2004	Invalid value in the offset field of the control block
2005	Invalid value in the length and offset fields of the control block
2006	Unauthorized data field on slave
2008	Unauthorized network routing path on slave
200E	The control block is not assigned, or parts of the control block are located outside of the %MW (4x) range.
3000	Generic Modbus failure code
30ss	Exception response by Modbus slave
4001	Inconsistent response by Modbus slave

ss Hexadecimal Subcode in 30ss Error Code

Hex Error Subcode	Meaning
01	Slave does not support requested operation
02	Non-existing slave registers were requested
03	An unauthorized data value was requested
05	Slave has accepted a lengthy program command
06	Function cannot currently be carried out: lengthy command running
07	Slave has rejected lengthy program command

Hexadecimal Error Codes For TCP/IP Ethernet Network Errors

An error on the TCP/IP Ethernet network itself may produce one of the following errors in the *CONTROL[1]* register of the control block.

Hex Error Code	Meaning
5004	Interrupted system invocation
5005	I/O error
5006	No such address
5009	The socket descriptor is not valid
500C	Not enough storage space
500D	Authorization denied
5011	Entry exists
5016	An argument is not valid
5017	An internal table has no more space
5020	There is interference on the connection
5023	This operation was blocked and the socket is non-blocking
5024	The socket is non-blocking and the connection cannot be closed down
5025	The socket is non-blocking and a previous connection attempt has not been concluded
5026	Socket operation on a non-socket
5027	The destination address is not valid
5028	Message too long
5029	Wrong type of protocol for the socket
502A	Protocol not available
502B	Protocol not supported
502C	Socket type not supported
502D	Operation not supported at socket
502E	Protocol family not supported
F502	Address family not supported
5030	Address is already in use
5031	Address not available
5032	Network is out of order
5033	Network cannot be reached
5034	Network shut down the connection during reset
5035	The connection was terminated by the peer
5036	The connection was reset by the peer
5037	An internal buffer is required, but cannot be assigned
5038	The socket is already connected
5039	The socket is not connected
503A	Cannot transmit after the socket has been shut off
503B	Too many references; cannot splice
503C	Connection timed out
503D	The connection attempt was denied
5040	Host is out of order
5041	The destination host could not be reached from this node
5042	Directory not empty
5046	NI_INIT returned -1
5047	The MTU is not valid

Hex Error Code	Meaning
5048	The hardware length is not valid
5049	The route specified cannot be found
504A	Collision when invoking Select; these conditions have already been selected by another job
504B	The job ID is not valid
5050	No Network Resource
5051	Length Error
5052	Addressing Error
5053	Application Error
5054	Client cannot process request
5055	No Network Resource
5056	Non-Operational TCP connection
5057	Incoherent configuration
6003	FIN or RST not expected
F001	In reset mode
F002	Component not fully initialized

4.5.2 MNETDPV1_BASICVAR Variables and DDTs - Quantum

These structures hold all the *Variables* and *DDTs* required to send and receive PROFIBUS DP-V0 or DP-V1 cyclic data messages and handle the responses. Cyclic data is all the data coming from and going to slaves on the PROFIBUS network on a regular, recurring cycle. Cyclic data transfers are accomplished at a very rapid, fixed-interval rate in a repeating cycle. The process of completing and repeating these data transfer cycles is called "polling".

As you can see below, there are four major types of cyclic data:

1. *Cyclic input data* - data from PROFIBUS Slaves sent to the Master
2. *Cyclic output data* - data from the PROFIBUS Master sent to the Slaves
3. General Gateway *(Module) Status Data* - created and reported by the gateway. (Although this data is not PROFIBUS protocol-specific data, it is updated along with all the other polling data and, therefore, will be treated as cyclic data by the automatically-created Application Communication Logic *DDTs* and *DFBs*.)
4. PROFIBUS *Slave Diagnostic Data* - the PROFIBUS protocol specifies that each slave send six (6) 8-bit bytes of status and diagnostic data in a fixed format to the Master as part of the regular polling cycle.

All the *DDTs* and *variables* required to use, control, and manage these four types of cyclic data are contained in the *MNETDPV1_BASICVAR* structures and sub-structures.

Cyclic input and output (I/O) data is the data to be transferred based on the PROFIBUS Master/Slave configuration you did in ProSoft Configuration Builder (PCB) when you configured specific amounts of inputs and outputs for each slave on the network.

The *ReadCyclicData* sub-structures handle PROFIBUS cyclic input data. For more information, see DFB Read Cyclic Data (page 146).

The *WriteCyclicData* sub-structures handle PROFIBUS cyclic output data. For more information, see DFB Write Cyclic Data (page 150).

The ModuleStatus sub-structures handle general gateway status data. For more information, see DFB Get Module Status (page 154).

The PB_SLVDiagnostics sub-structures handle the standard PROFIBUS slave diagnostic data. For more information, see DFB Get PROFIBUS Standard Slave Diagnostics (page 158).

4.5.3 MNETDPV1_MailVar Variables and DDTs - Quantum

These structures hold the all the *Variables* and *DDTs* required to send and receive PROFIBUS DP-V1 acyclic messages, also called *Mailbox Messages*. Note that acyclic messaging is available only on devices using PROFIBUS DP Version 1 or above. PROFIBUS Version 0 devices do not support acyclic messaging.

Acyclic messages are PROFIBUS Master commands that are sent in addition to normal cyclic polling. Acyclic messages are sent at irregular intervals, interspersed in between regular cyclic polling messages. Cyclic polling is deterministic and happens at predictable intervals. Acyclic messaging is not deterministic and not guaranteed to happen at any predictable interval. For this reason, acyclic messages are used for special functions more than for normal data transfer operations.

There are ten major types of acyclic messages supported by the gateway:

1. *Read Acyclic Data* - There are limits to the amount of cyclic input data that can be transferred from PROFIBUS slaves. Some devices can provide more data than can fit within these limits. Acyclic Read messages give the PROFIBUS Master a way to request this additional slave data. For detailed information, see DFB Acyclic Mailbox Message: Read Class 1 Acyclic Data (page 186).
2. *Write Acyclic Data* - There are limits to the amount of cyclic output data that can be transferred to PROFIBUS slaves. Some devices require more data can fit within these limits. Acyclic Write messages give the PROFIBUS Master a way to send this additional data to the slaves. For detailed information, see DFB Acyclic Mailbox Message: Write Class 1 Acyclic Data (page 190).
3. *Get Slave Configuration* - These structures allow the Master to read the actual configuration (identifier bytes) of a specified slave. For detailed information, see DFB Acyclic Mailbox Message: Get Slave Configuration (page 172).
4. *Get* (Extended) *Slave Diagnostic Data* - Some PROFIBUS DP-V1 devices can provide additional diagnostic and alarm data in addition to the six standard diagnostic bytes provided by all slaves. The Get Slave Diagnostic Data message allows the PROFIBUS DP Master to retrieve this extra data from slaves that can provide it. For detailed information, see DFB Acyclic Mailbox Message: Get Slave Diagnostics (page 168).
5. *Get Live List* - A PROFIBUS network can have up to 126 total nodes. The *Live List* is a way for the Master to know which node addresses have active slaves associated with them and which do not. This is a way to see what nodes are 'alive' and 'living' on the network, attached, and ready to transfer data. For detailed information, see DFB Acyclic Mailbox Message: Get Live List (page 166).
6. *Set Slave Address* - For Slaves that support this capability, this structure allows the PROFIBUS DP Master to change the Slave address number of a particular slave. For detailed information, see DFB Acyclic Mailbox Message: Set Address (page 180).

7 *Set Slave Mode* - Some PROFIBUS Slaves support capabilities called *Sync* and *Freeze*. These are special command features which allow a PROFIBUS Master to control when and how a slave updates its internal cyclic inputs and outputs. These structures give the Master the ability to send these special kinds of control messages. For detailed information, see DFB Acyclic Mailbox Message: Set Slave Mode (page 178).

8 *Start/Stop Slaves* - These structures allow the Master to stop or start cyclic data transfers with a slave or or group of slaves. For detailed information, see DFB Acyclic Mailbox Message: Start/Stop Slave (page 176).

9 *Set Operate Mode* - These structures allow the Master to suspend or restart all cyclic polling activity on the network. For detailed information, see DFB Acyclic Mailbox Message: Set Operating Mode (page 164).

10 *Get Database* - These structures allow the Master to obtain and report database configuration information about the PROFIBUS Master hardware. For detailed information, see DFB Acyclic Mailbox Message: Get Database Information (page 184).

All the *DDTs* and *variables* required to use, control, and manage these ten types of acyclic messages are contained in the *MNETDPV1_MAILVAR* structures and sub-structures.

4.5.4 Cyclic I/O Variables, DDTs and DFBs - Quantum

The following sections provide a more detailed breakdown of the *Variables, DDTs* and *DFBs* used for transferring PROFIBUS cyclic data.

DFB ReadCycData - Quantum

The *ReadCycData DFB* is used to retrieve PROFIBUS cyclic input data from the 5204SE-MNET-PDPMV1 gateway and bring it back into the processor. This is the data being received by the PROFIBUS DP-V1 Master from the slave or slaves on the PROFIBUS network.

MNETDPV1_BASICVAR_ReadCyclicData Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a request to read PROFIBUS cyclic input data.

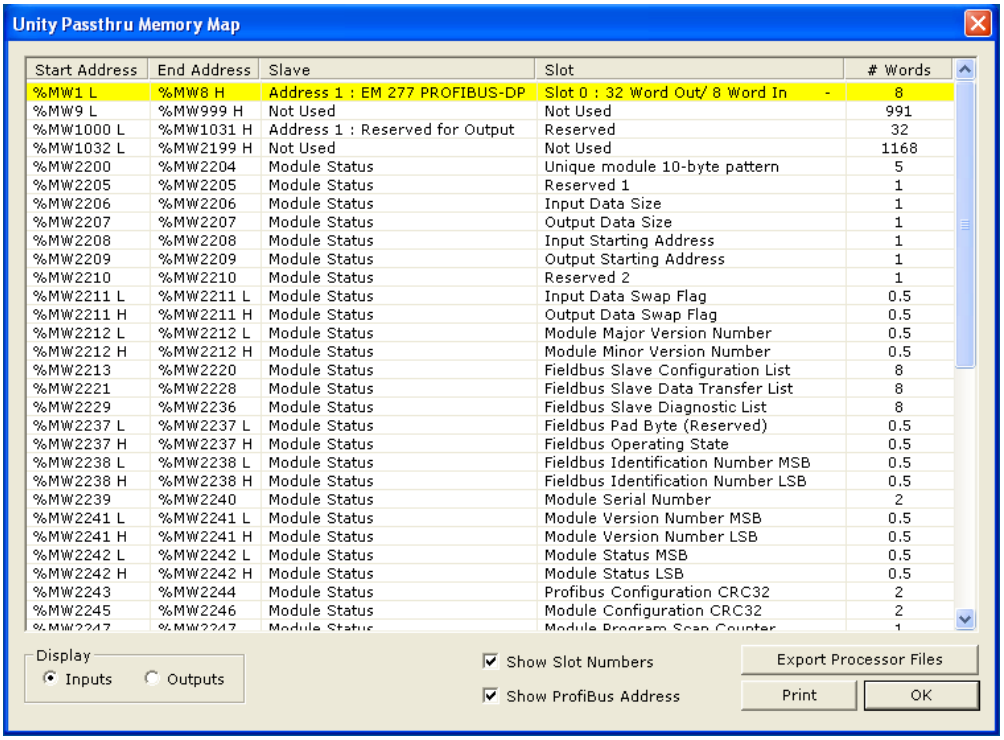
MNETDPV1_BASICVAR	MNETDPV1_BASICVAR		
ReadCyclicData	CyclicReadData		
Out	CyclicReadDataOUT		
ReadcyclicData	BOOL		
RegisterCount	INT		768
IPAddress	ARRAY[1..4] OF INT		
IPAddress[1]	INT		192
IPAddress[2]	INT		168
IPAddress[3]	INT		0
IPAddress[4]	INT		100
RoutingRegister	INT		16#FE00
In	CyclicReadDataIN		
ReadOperationActive	BOOL		
ReadOperationError	BOOL		
ReadOperationSuccess	BOOL		
WriteCyclicData	CyclicWriteData		
ModuleStatus	ModuleStatus		
PB_SLVDiagnostics	PB_SlaveDiagnostic		
MNETDPV1_CNTRL_BUFFS	FunctionBlockList	%MW2700	
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW1	
MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000	
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR		
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276	
MNETDPV1_StatIn	StatInF	%MW2200	

Variable Name	Size/Type	Description
Out - ReadcyclicData	1 Single-bit Boolean	This is the bit your control and sequencing logic will use to trigger a cyclic read message. Set this bit to one (1) whenever no other messages are active and when you want to update the PROFIBUS cyclic slave input data.
Out - RegisterCount	1 16-bit integer	Will hold the total number of 16-bit register words of PROFIBUS cyclic input data that need to be read. This value will be the same as what you entered in the PCB configuration for the **[PROFIBUS MASTER DPV1] INPUT DATA SIZE** parameter
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - Routing Register	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - ReadOperationActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - ReadOperationError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - ReadOperationSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

MNETDPV1_Inputs Variable - Quantum

This *variable* is an array of 1536 bytes. It is used to receive up to 768 words (1536 bytes) of PROFIBUS cyclic input data from slaves on the PROFIBUS network.

The order of data in this array will match the order in the PCB memory maps you exported and/or printed. The following screen shot shows a typical memory map.



The PCB table lists usage in words rather than bytes, where one (1) word equals two (2) bytes. In this example, there are only eight (8) total words or 16 total bytes of PROFIBUS cyclic input data configured (highlighted in yellow) of the available 1536 bytes that could be used. The 16 bytes (8 words) of PROFIBUS inputs from the device assigned to Slave Address 1 will be stored in the first 16 bytes of this array.

You should also notice that the native storage size in the module's memory is 16-bit or 2-byte word registers. If the number of inputs from the first configured device is an odd number of bytes, you will see that memory register hold one byte from the first device in its low-order byte. The higher-order byte of this register will hold the first byte of data for the next configured slave device. In other words, this data is *byte-packed* with no extra blank bytes inserted just so the data for each slave address can begin on a low-order byte boundary.

If you wish to put gaps into the memory map to give more separation between data blocks from different slave addresses, you may do so in the PROFIBUS Master configuration in PCB by editing the starting address of the data for each slave so that it falls on whatever byte or register address you prefer.

MNETDPV1_DataIn Variables - Quantum

These variables allow you to take advantage of the *MNETDPV1_DataIn DDT* structure.

MNETDPV1_BASICVAR	MNETDPV1_BASICVAR	
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW1
Slave01Slot00	ARRAY[0..15] OF BYTE	%MW1
MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MailVar	

There is no direct link or logic provided to populate this array with the data received in the *MNETDPV1_Inputs* variable. If you wish to use these variables for your application, you will need to create the logic to link the individual bytes of the *MNETDPV1_Inputs* variable to the word array variables in this structure.

The *%MW* addresses shown are for illustration only. If you decide to use these variables, your application may require that you map them to addresses other than the ones shown. You may assign these variables to any valid *%MW* addresses that exist in your processor configuration.

Sample Procedure for Copying from the MNETDPV1_Inputs array to the MNENTDPV1_DataIn Variables

1. Create an INT variable to use as the control variable in a copy loop. This example uses the variable "i".
2. Assign a specific *%MW* address to the variables in the *MNETDPV1_Inputs* variable array. This example uses address *%MW 200*.
3. Assign a specific *%MW* address to the *MNETDPV1_DataIn* variable structure. This example uses address *%MW 1*.
4. Use logic to copy from one set of *%MW* memory addresses to the other for the amount of data you need to copy. In our sample configuration, we have 16 bytes of PROFIBUS cyclic input data. So, the logic needed would look something like this:

```
FOR i:=0 to 15 DO
   %MW1[i]:=%MW200[i] ;
END_FOR ;
```

DFB Write Cyclic Data - Quantum

MNETDPV1_BASICVAR_WriteCyclicData Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a request to write PROFIBUS cyclic output data.

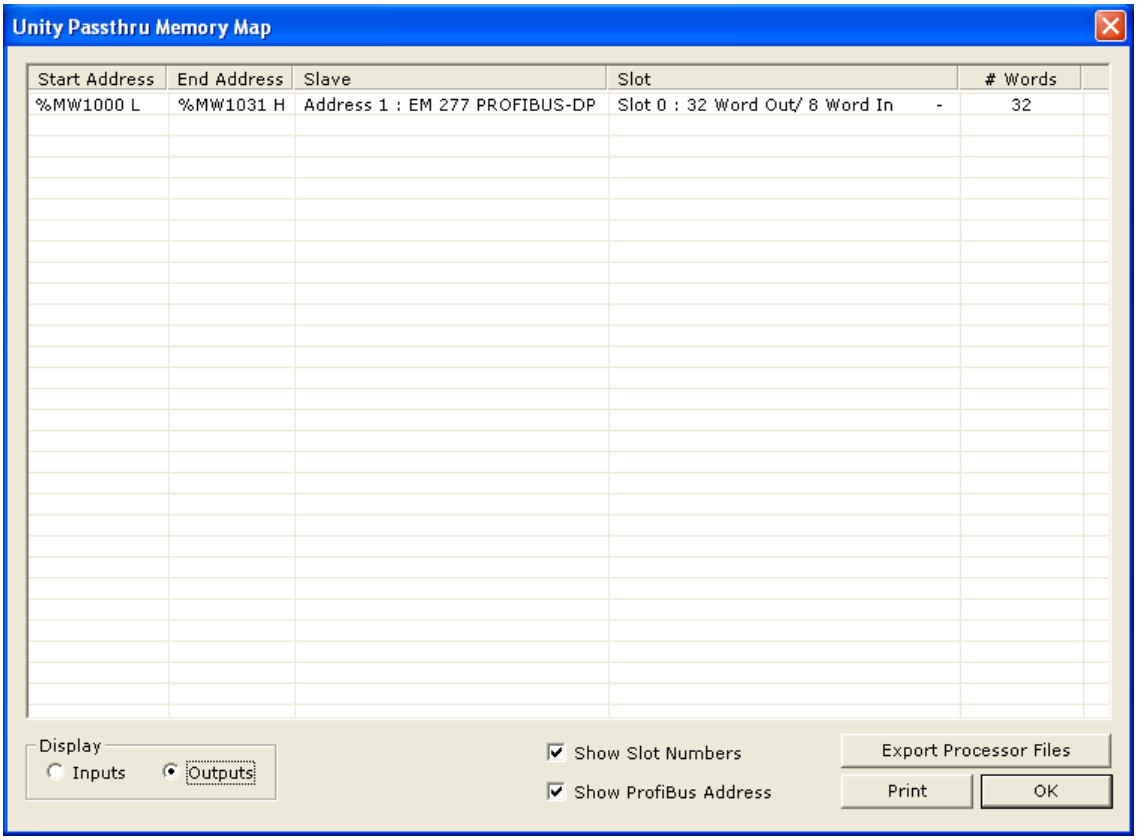
MNETDPV1_BASICVAR	MNETDPV1_BASICVAR	
ReadCyclicData	CyclicReadData	
WriteCyclicData	CyclicWriteData	
Out	CyclicWriteDataOUT	
WriteCyclicData	BOOL	
RegisterCount	INT	
IPAddress	ARRAY[1..4] OF INT	
IPAddress[1]	INT	
IPAddress[2]	INT	
IPAddress[3]	INT	
IPAddress[4]	INT	
RoutingRegister	INT	
In	CyclicWriteDataIN	
WriteOperationActive	BOOL	
WriteOperationError	BOOL	
WriteOperationSuccess	BOOL	
ModuleStatus	ModuleStatus	
PB_SLVDiagnostics	PB_SlaveDiagnostic	
MNETDPV1_CNTRL_BUFFS	FunctionBlockList	%MW2700
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW1
MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000

Variable Name	Size/Type	Description
Out - WriteCyclicData	1 Single-bit Boolean	This is the bit your control and sequencing logic will use to trigger a cyclic write message. Set this bit to one (1) whenever no other messages are active and when you want to send PROFIBUS cyclic slave output data.
Out - RegisterCount	1 16-bit integer	Will hold the total number of 16-bit register words of PROFIBUS cyclic output data that need to be written. This value will be the same as what you entered in the PCB configuration for the **[PROFIBUS MASTER DPV1] OUTPUT DATA SIZE** parameter.
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - Routing Register	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - WriteOperation Active	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - WriteOperation Error	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.
In - WriteOperation Success	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic output data variables have been updated.

MNETDPV1_Outputs Variable - Quantum

This *variable* is an array of 1536 bytes. It is used to hold up to 768 words (1536 bytes) of PROFIBUS cyclic output data to be sent to slaves on the PROFIBUS network.

The order of data in this array will match the order in the PCB memory maps you exported and/or printed. The following illustration shows a typical memory map.



The PCB table lists usage in words rather than bytes, where one (1) word equals two (2) bytes. In this example, there are only 32 total words or 64 total bytes of PROFIBUS output data configured of the 1536 bytes available that could be used. The first 64 bytes (32 words) this array will hold data to be sent to Slave Address 1, Slot 0.

You should also notice that the native storage size in the module's memory is 16-bit or 2-byte word registers. When you have more than one slave device, this data is 'byte-packed' with no extra blank bytes inserted just so the data for each slave address can begin on an even-numbered, low-order byte boundary.

If you wish to put gaps into the memory map to give more separation between data blocks from different slave addresses, you may do so in the PROFIBUS Master configuration in PCB by editing the starting address of the data for each slave.

MNETDPV1_DataOut Variable - Quantum

These variables allow you to take advantage of the *MNETDPV1_DataOut DDT* structure.

MNETDPV1_BASICVAR	MNETDPV1_BASICVAR	
MNETDPV1_CNTRL_BUFFS	FunctionBlockList	%MW2700
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW1
MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
Slave01Slot00	ARRAY[0..63] OF BYTE	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276
MNETDPV1_StatIn	StatInF	%MW2200

There is no direct link or logic provided to populate data in the *MNETDPV1_Outputs* variable from the data in these variables. If you wish to use these variables for your application, you will need to create the logic to link the variables in this structure to the *MNETDPV1_Outputs* array variable.

The *%MW* addresses shown are for illustration only. If you decide to use these variables, your application may require that you map them to addresses other than the ones shown here. You may assign these variables to any valid *%MW* addresses that exist in your processor configuration.

Sample Procedure for Copying from the MNETDPV1_DataOut variables to the MNENTDPV1_Outputs array

1. Create an INT variable to use as the control variable in a copy loop. This example uses the variable "j".
2. Assign a specific *%MW* address to the variables in the *MNETDPV1_DataOut* variable structure. This example uses address *%MW 1000*.
3. Assign a specific *%MW* address to the *MNETDPV1_Outputs* variable array. This example uses address *%MW 1200*.
4. Use logic to copy from one set of *%MW* memory addresses to the other for the amount of data you need to copy. In our sample configuration, we have 64 bytes of PROFIBUS cyclic output data. So, the logic needed would look something like this:

```
FOR j:=0 to 63 DO
   %MW1000[j]:=%MW1200[j] ;
END_FOR ;
```

DFB Get Module Status - Quantum

MNETDPV1_BASICVAR_ModuleStatus Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a request to read general gateway status data.

MNETDPV1_BASICVAR	MNETDPV1_BASICVAR	
ReadCyclicData	CyclicReadData	
WriteCyclicData	CyclicWriteData	
ModuleStatus	ModuleStatus	
Out	ModuleStatusOUT	
GetModuleStatus	BOOL	
IPAddress	ARRAY[1..4] OF INT	
IPAddress[1]	INT	
IPAddress[2]	INT	
IPAddress[3]	INT	
IPAddress[4]	INT	
RoutingRegister	INT	
In	ModuleStatusIN	
StatusOperationActive	BOOL	
StatusOperationError	BOOL	
StatusOperationSuccess	BOOL	
Fault	INT	
ReturnCode	INT	
PB_SLVDiagnostics	PB_SlaveDiagnostic	
MNETDPV1_CNTRL_BUFFS	FunctionBlockList	%MW2700
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW1

Variable Name	Size/Type	Description
Out - GetModuleStatus	1 Single-bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to get general gateway status data. Set this bit to one (1) whenever no other messages are active and when you want to update the StatInF variable table.
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - RoutingRegister	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - StatusOperationActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - StatusOperationError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - StatusOperationSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.
In - Fault	1 16-bit Integer	If "Invalid other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Acyclic Message Status Word (page 218) **0001h:** Address out of range. **000Ah**: Failed to execute request. Refer to Return Codes (page 219) for additional information. **000Bh**: Remote station failure. Refer to Return Codes (page 219) for additional information. **00FFh**: gateway not initialized.
In - ReturnCode	1 16-bit Integer	Refer to Return Codes (page 219) for additional information.

MNETDPV1_StatIn Variables - Quantum

These variables take advantage of the *StatInF DDT* structure. The *GetStatus DFB* will automatically populate this variable list with general gateway status information received in the Modbus TCP/IP response to the *GetModuleStatus* command.

MNETDPV1_StatIn	StatInF	%MW2200
ModuleStatus_Uniquemodule10bytepattern	ARRAY[0..4] OF WORD	%MW2200
ModuleStatus_Reserved1	WORD	%MW2205
ModuleStatus_InputDataSize	WORD	%MW2206
ModuleStatus_OutputDataSize	WORD	%MW2207
ModuleStatus_InputStartingAddress	WORD	%MW2208
ModuleStatus_OutputStartingAddress	WORD	%MW2209
ModuleStatus_Reserved2	WORD	%MW2210
ModuleStatus_InputDataSwapFlag	BYTE	%MW2211
ModuleStatus_OutputDataSwapFlag	BYTE	%MW2211
ModuleStatus_ModuleMajorVersionNumber	BYTE	%MW2212
ModuleStatus_ModuleMinorVersionNumber	BYTE	%MW2212
ModuleStatus_FieldbusSlaveConfigurationList	ARRAY[0..7] OF WORD	%MW2213
ModuleStatus_FieldbusSlaveDataTransferList	ARRAY[0..7] OF WORD	%MW2221
ModuleStatus_FieldbusSlaveDiagnosticList	ARRAY[0..7] OF WORD	%MW2229
ModuleStatus_FieldbusPadByteReserved	BYTE	%MW2237
ModuleStatus_FieldbusOperatingState	BYTE	%MW2237
ModuleStatus_FieldbusIdentificationNumberMSB	BYTE	%MW2238
ModuleStatus_FieldbusIdentificationNumberLSB	BYTE	%MW2238
ModuleStatus_ModuleSerialNumber	ARRAY[0..1] OF WORD	%MW2239
ModuleStatus_ModuleVersionNumberMSB	BYTE	%MW2241
ModuleStatus_ModuleVersionNumberLSB	BYTE	%MW2241
ModuleStatus_ModuleStatusMSB	BYTE	%MW2242
ModuleStatus_ModuleStatusLSB	BYTE	%MW2242
ModuleStatus_ProfibusConfigurationCRC32	ARRAY[0..1] OF WORD	%MW2243
ModuleStatus_ModuleConfigurationCRC32	ARRAY[0..1] OF WORD	%MW2245
ModuleStatus_ModuleProgramScanCounter	WORD	%MW2247
ModuleStatus_ProfibusOutputUpdateCounter	WORD	%MW2248
ModuleStatus_ProfibusInputUpdateCounter	WORD	%MW2249
ModuleStatus_OutputMailboxCounter	WORD	%MW2250
ModuleStatus_InputMailboxCounter	WORD	%MW2251
ModuleStatus_AlarmINDCounter	WORD	%MW2252
ModuleStatus_AlarmCONCounter	WORD	%MW2253
ModuleStatus_AcyclicReadRequestCounter	WORD	%MW2254
ModuleStatus_AcyclicWriteRequestCounter	WORD	%MW2255
ModuleStatus_Reserved3	ARRAY[0..2] OF WORD	%MW2256
ModuleStatus_ModuleFileErrorWordbitmapped	WORD	%MW2259
ModuleStatus_Reserved4	ARRAY[0..5] OF WORD	%MW2260
ModuleStatus_ConfiguredResponseTimeout	WORD	%MW2266
ModuleStatus_Reserved5	ARRAY[0..1] OF WORD	%MW2267
ModuleStatus_MailboxMessagingDBRegister	WORD	%MW2269
ModuleStatus_MailboxMessagingAlarmRegister	WORD	%MW2270
ModuleStatus_SlaveDiagnosticStartReg	WORD	%MW2271
ModuleStatus_StatusStartRegister	WORD	%MW2272
ModuleStatus_MailboxInputQueueMessageCo...	WORD	%MW2273
ModuleStatus_MailboxOutputQueueMessageC...	WORD	%MW2274
ModuleStatus_AlarmqueueMessageCount	WORD	%MW2275

The *%MW* addresses shown are for illustration only. If you decide to use these variables, your application may require that you map them to addresses other than the ones shown here. You may assign these variables to any valid *%MW* addresses that exist in your processor configuration by setting the desired start address in the **PLC STATUS REGISTER START** parameter in the PCB configuration file. This will cause the import files to contain addresses in the range you select and change the values displayed in this array.

{This page intentionally left blank.}

DFB Get PROFIBUS Standard Slave Diagnostics - Quantum

MNETDPV1_BASICVAR_PB_SLVDiagnostics Variables - Quantum

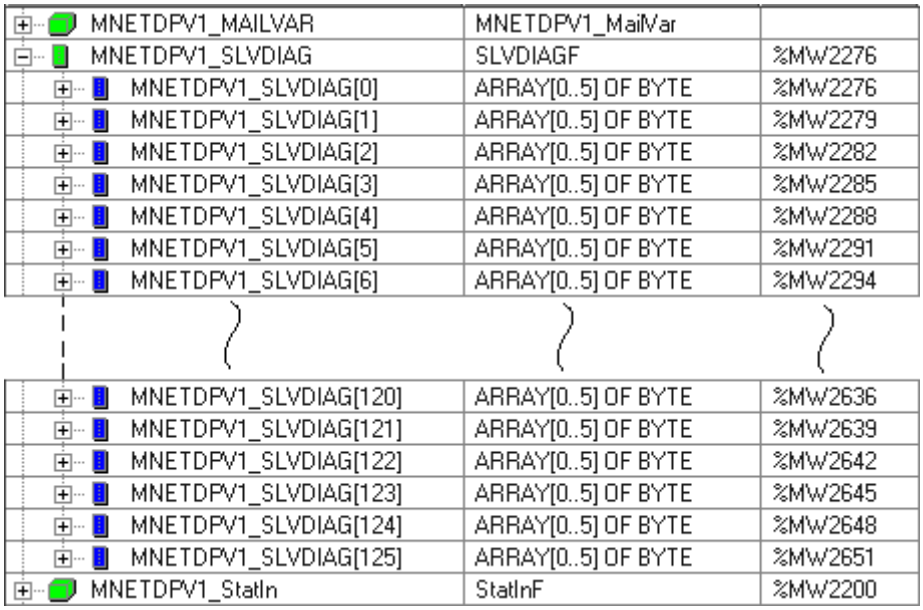
This variable structure is the one to use with your control and sequencing logic when you want to send a request to read standard PROFIBUS slave diagnostic data.

MNETDPV1_BASICVAR	MNETDPV1_BASICVAR	
ReadCyclicData	CyclicReadData	
WriteCyclicData	CyclicWriteData	
ModuleStatus	ModuleStatus	
PB_SLVDiagnostics	PB_SlaveDiagnostic	
Out	PB_SlaveDiagnosticsOUT	
GetPBSlaveDiagnostics	BOOL	
IPAddress	ARRAY[1..4] OF INT	
IPAddress[1]	INT	
IPAddress[2]	INT	
IPAddress[3]	INT	
IPAddress[4]	INT	
RoutingRegister	INT	
In	PB_SlaveDiagnosticsIN	
DiagOperationActive	BOOL	
DiagOperationError	BOOL	
DiagOperationSuccess	BOOL	
MNETDPV1_CNTRL_BUFFS	FunctionBlockList	%MW2700
MNETDPV1_DataIn	MNETDPV1_DataInF	%MW1
MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276
MNETDPV1_StatIn	StatInF	%MW2200

Variable Name	Size/Type	Description
Out - GetPBSlaveDiagnostics	1 Single-bit Boolean	This is the bit your control and sequencing logic will use to trigger a read message that will retrieve PROFIBUS slave diagnostic data. Set this bit to one (1) whenever no other messages are active and when you want to update the MNETDPV1_SLVDIAG data variables.
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - RoutingRegister	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - DiagOperationActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - DiagOperationError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - DiagOperationSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

MNETDPV1_SLVDIAG Variables - Quantum

This variable structure is a collection of six-byte arrays. Each array element holds the six bytes of standard PROFIBUS slave data reported to the PROFIBUS Master from each slave that exists on the network as part of the regular cyclic data polling scheme. The array element number corresponds to the node address of each slave.



MNETDPV1_MAILVAR	MNETDPV1_MailVar	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276
MNETDPV1_SLVDIAG[0]	ARRAY[0..5] OF BYTE	%MW2276
MNETDPV1_SLVDIAG[1]	ARRAY[0..5] OF BYTE	%MW2279
MNETDPV1_SLVDIAG[2]	ARRAY[0..5] OF BYTE	%MW2282
MNETDPV1_SLVDIAG[3]	ARRAY[0..5] OF BYTE	%MW2285
MNETDPV1_SLVDIAG[4]	ARRAY[0..5] OF BYTE	%MW2288
MNETDPV1_SLVDIAG[5]	ARRAY[0..5] OF BYTE	%MW2291
MNETDPV1_SLVDIAG[6]	ARRAY[0..5] OF BYTE	%MW2294
MNETDPV1_SLVDIAG[120]	ARRAY[0..5] OF BYTE	%MW2636
MNETDPV1_SLVDIAG[121]	ARRAY[0..5] OF BYTE	%MW2639
MNETDPV1_SLVDIAG[122]	ARRAY[0..5] OF BYTE	%MW2642
MNETDPV1_SLVDIAG[123]	ARRAY[0..5] OF BYTE	%MW2645
MNETDPV1_SLVDIAG[124]	ARRAY[0..5] OF BYTE	%MW2648
MNETDPV1_SLVDIAG[125]	ARRAY[0..5] OF BYTE	%MW2651
MNETDPV1_StatIn	StatInF	%MW2200

The *PB_SlaveDiagnostic DFB* will automatically populate these variables with the diagnostic data returned by the Modbus TCP/IP command. The *%MW* addresses shown are for illustration only. If you decide to use these variables, your application may require that you map them to addresses other than the ones shown here. You may assign these variables to any valid *%MW* addresses that exist in your processor configuration.

4.5.5 Sample Control and Sequencing Logic for Cyclic Data Polling - Quantum

Here is a structured text (ST) logic example of how you might control and sequence the PROFIBUS cyclic data *DFBs*. You may adapt this sample to fit your application or you may choose to create your own control and sequencing scheme that is more suitable for your specific needs.

For this example, start by creating two variables:

LastExecuted	as type INT
Start	as type BOOL

Then, you can use the following ST logic code. Each time you set the variable *Start* equal to 1, it will begin executing a sequence to read cyclic inputs, write cyclic outputs, get general gateway status, and get standard PROFIBUS slave-specific diagnostic data. As long as *Start* remains equal to 1, this sequence will roll-over and be repeated until interrupted by setting *Start* = 0.

```
IF Start:=1 THEN

   IF MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics=0 AND
      MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus=0 AND
      MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData=0 AND
      MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadcyclicData=0 THEN

      IF LastExcuted=0 THEN
     MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadcyclicData:=1;
         LastExcuted:=1;
      END_IF;
   END_IF;

   IF MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics=0 AND
      MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus=0 AND
      MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData=0 AND
      MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadcyclicData=0 THEN

      IF LastExcuted=1 THEN
         MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData:=1;
     LastExcuted:=2;
      END_IF;
   END_IF;

   IF MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics=0 AND
      MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus=0 AND
      MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData=0 AND
      MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadcyclicData=0 THEN

      IF LastExcuted=2 THEN
         MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus:=1;
     LastExcuted:=3;
      END_IF;
   END_IF;

  IF MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics=0 AND
     MNETPDPMV1_BASICVAR.ModuleStatus.Out.GetModuleStatus=0 AND
     MNETPDPMV1_BASICVAR.WriteCyclicData.Out.WriteCyclicData=0 AND
     MNETPDPMV1_BASICVAR.ReadCyclicData.Out.ReadcyclicData=0 THEN

     IF LastExcuted=3 THEN
        MNETPDPMV1_BASICVAR.PB_SLVDiagnostics.Out.GetPBSlaveDiagnostics:=1;
    LastExcuted:=0;
       END_IF;
   END_IF;

END_IF;
```

4.5.6 Acyclic Messaging DFBs - Quantum

These following eleven sections provide information about the *Derived Data Types (DDTs)* and *Variables* associated with each of the ten (10) *Derived Function Blocks (DFBs)* created by the *Application Communication Logic* functions of *ProSoft Configuration Builder (PCB)* that can be used to send PROFIBUS DP-V1 acyclic messaging. Your application-specific control and sequencing logic will use these variables to activate these special functions, if required, as required, and receive any results that may be returned.

The last item for each *DFB* topic is a breakdown of the PROFIBUS acyclic message structure. Creating these messages and handling the responses, if any, will all be done for you by the provided *DFBs*. Since each message has its own unique set of error codes, the primary reason for including this message structure information is to help you interpret any error codes you may receive as a result of activating one of these acyclic messages. The eleventh section contains acyclic message error code information.

DFB Acyclic Mailbox Message: Set Operating Mode - Quantum

This command allows setting the operating mode of the PROFIBUS Master (STOP, CLEAR, or OPERATE).

MNETDPV1_MAILVAR_SetOperatMode Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a request to set the PROFIBUS Master/network operating mode. Possible choices are:

- **Operate -** Tells the PROFIBUS Master to begin and continue normal cyclic polling and pass acyclic messages, if requested. The network should be in **Stop** mode when you issue this command message.
- **Stop -** Tells the PROFIBUS Master to interrupt normal cyclic polling. Acyclic messaging can still be accomplished when the network is stopped. The network should be in **Operate** mode when you issue this command.
- **Clear -** Tells the PROFIBUS Master to attempt to clear diagnostic errors and re-initialize the PROFIBUS network. The network should be in **Stop** mode when you issue this command.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
Out	SetOperateModeOUT	
IPAddress	ARRAY[1..4] OF INT	
Operate	BOOL	
Stop	BOOL	
Clear	BOOL	
RoutingRegister	INT	
In	SetOperateModeIN	
OperationActive	BOOL	
OperationError	BOOL	
OperationSuccess	BOOL	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - Operate	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to begin and continue normal network cyclic polling and acyclic messaging. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - Stop	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to stop normal network cyclic polling. Some acyclic messaging can still be accomplished in this mode. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - Clear	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to clear slave diagnostic faults and re-initialize the PROFIBUS network. Set this bit to one (1) whenever no other messages are active, the network mode is currently set to **STOP**, and you want to send this acyclic message.
Out - RoutingRegister	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - OperationActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - OperationError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - OperationSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

DFB Acyclic Mailbox Message: Get Live List - Quantum

This acyclic message returns 127 bytes of information about the nodes on the network. Each byte holds the node type for one bus subscriber (node or device). The position of the byte in the response data corresponds to the address (0 to 125) of the node on the network. The content of each byte tells whether the node is a Master or Slave (multiple PROFIBUS Masters may co-exist on the same physical network).

This acyclic message can be sent in all operation modes (STOP, CLEAR, and OPERATE), however the gateway must be initialized properly.

MNETDPV1_MAILVAR_GetLiveList Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to request a list of PROFIBUS network nodes (bus subscribers).

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
Out	GetListOut	
GetLiveList	BOOL	
IPAddress	ARRAY[1..4] OF INT	
RoutingRegister	INT	
In	GetListIN	
ListOperationActive	BOOL	
ListOperationError	BOOL	
ListOperationSuccess	BOOL	
StationStatus	ARRAY[0..127] OF BYTE	
Fault	INT	
ReturnCode	INT	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - GetLiveList	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master provide a list of active slave nodes. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - RoutingRegister	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - ListOperationActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - ListOperationError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - ListOperationSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.
In - StationsStatus[ ]	127 element 8-bit Byte Array	Each byte will contain one of the following codes indicating type of node present at that node address: **00h:** Slave Station **01h:** Master Station not yet ready for Token ring (station physically attached to the bus but not configured or polling) **02h:** Master Station ready to enter Token ring (station is configured but not polling; there is not yet any Token transmission) **03h**: Master Station in Token Ring (Token transmission through the station; station is fully operational) **04h**: Station does not exist
In - Fault	1 16-bit Integer	For details on Fault Codes, see Acyclic Message Status Word (page 218).
In - Return Code	1 16-bit Integer	For details, see Return Codes **(page 219).**

DFB Acyclic Mailbox Message: Get Slave Diagnostics - Quantum

This acyclic message reads extended diagnostic data from a specified slave.

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

MNETDPV1_MAILVAR_GetDiag Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to retrieve extended slave diagnostic data from a specific PROFIBUS network slave.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
Out	GetDiagnosticOUT	
GetSlaveDiagnostics	BOOL	
SlaveAddress	BYTE	
RequestType	BYTE	
IPAddress	ARRAY[1..4] OF INT	
RoutingRegister	INT	
In	GetDiagnosticIN	
Error1	BYTE	
Length	BYTE	
Fault	INT	
ReturnCode	INT	
SlaveDiagnosticsData	ARRAY[0..199] OF BYTE	
RDDiagnostActive	BOOL	
RDDiagnostError	BOOL	
RDDiagnostSuccess	BOOL	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - GetSlaveDiagnostics	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to retrieve extended slave diagnostic data from a specified slave address. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries: 0 - 125 Enter the slave address of the device from which you wish to retrieve extended diagnostic data.
Out - RequestType	1 8-bit Byte	Valid entries: 0 or 1 0 = Get slave extended diagnostic data already stored on the Master. Can be requested only from slaves configured by this Master node. (faster response to the acyclic message; but data may not be current) 1 = Send a special request on the network to read extended diagnostic data directly from the slave at the address specified. Can be requested from any slave on the network. (takes longer to receive response but data is current)
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - RoutingRegister	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - Error1	1 8-bit Byte	Error Byte 1 of 4. Bytes 2 - 4 are reserved For details on Error Byte 1, see Acyclic Message Status Word (page 218)
In - Length	1 8-bit Byte	Number of diagnostic bytes returned by the slave and help in the *In-Response Data[ ]* variable array
In - Fault	1 16-bit Integer	If "Invalid Other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. **0001h:** Address out of range. **0002h:** Incorrect "Type of request" **000Ah:** Failed to read diagnostic data from slave. Refer to Return Codes (page 219) for additional fault information. **000Bh:** Remote station failure. For additional fault information, refer to Return Codes (page 219) **00FEh:** Command not possible; gateway operates as a Class 2 master only. **00FFh:** Gateway offline (not initialized or no valid database).
In - Return Code	1 16-bit Integer	For details, see Return Codes (page 219)

Variable Name	Size/Type	Description
In - SlaveDiagnosticData[ ]	200-element 8-bit Byte Array	For detailed breakdown of the data available in this array, see Get (Extended) Slave Diagnostic Message Structure (page 198). The amount and type of extended diagnostic data returned varies, and depends on the capabilities of the slave device. Refer to the device manufacturer's documentation for slave diagnostic information.
In - RDDiagnostActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - RDDiagnostError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - RDDiagnostSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

NOTE: The MBP_MSTR function used on the Quantum platform has a lower data transfer limit than the DATA_EXCH function for the M340 platform. As a result, on the Quantum platform, the amount of data that can be transferred between a Quantum PAC and the ProLinx Gateway is 200, 8-bit bytes (100, 16-bit words) per message transaction.
This limit is more than enough for most PROFIBUS DP message telegrams. However, there are a few PROFIBUS messages that could contain more data in the PROFIBUS response than can be sent to the Quantum processor using the MBP_MSTR funcution for Modbus TCP/IP. It such cases, the PROFIBUS data is truncated after the first 200-bytes, any PROFIBUS data beyond that limit will not be sent to the Quantum, and the excess data will be lost.
The performance of this *Get Slave Diagnostics* mailbox message may be affected by this limitation. Please consult your device manufacture's documentation to see if any of your PROFIBUS DP slave devices provide more than 200 bytes of extended diagnostic data in per message.

{This page intentionally left blank.}

DFB Acyclic Mailbox Message: Get Slave Configuration - Quantum

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

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

MNETDPV1_MAILVAR_GetConfig Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to retrieve configuration information from a specific PROFIBUS network slave.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
Out	GetSlaveConfigurationOUT	
GetSlaveConfiguration	BOOL	
SlaveAddress	BYTE	
IPAddress	ARRAY[1..4] OF INT	
RoutingRegister	INT	
In	GetSlaveConfigurationIN	
Error1	BYTE	
Length	BYTE	
Fault	INT	
ReturnCode	INT	
SlaveConfigurations	ARRAY[0..199] OF BYTE	
RDConfigActive	BOOL	
RDConfigError	BOOL	
RDConfigSuccess	BOOL	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - GetSlaveConfiguration	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to retrieve slave configuration information from the specified slave address. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries: 0 - 125 Enter the slave address of the device from which you wish to retrieve slave configuration data.
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - RoutingRegister	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - Error1	1 8-bit Byte	Error Byte 1 of 4. Bytes 2 - 4 are reserved For details on Error Byte 1, see Acyclic Message Status Word (page 218)
In -Length	1 8-bit Byte	Range 6 to 244 Number of bytes sent by the slave as response data.
In - Fault	1 16-bit Integer	If "Invalid other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Acyclic Message Status Word (page 218) **0001h:** Address out of range. **000Ah**: Failed to execute request. Refer to Return Codes (page 219) for additional information. **000Bh**: Remote station failure. Refer to Return Codes (page 219) for additional information. **00FFh**: Gateway not initialized.
In - Return Code	1 16-bit Integer	For details, see Return Codes (page 219)
In - SlaveConfigurations[ ]	200-element 8-bit Byte Array	For detailed breakdown of the data available in this array, see Get Slave Configuration Message Structure (page 200). Response Data size will be from 6 to 244 bytes. Actual amount and type of data returned varies and depends on the capabilities of the slave device. Please see the device manufacturer's documentation for details.
In - RDConfigActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.

Variable Name	Size/Type	Description
In - RDConfigError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - RDConfigSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

{This page intentionally left blank.}

DFB Acyclic Mailbox Message: Start/Stop Slave - Quantum

This acyclic message stops or starts a selection of slaves. Stopping a slave or group of slaves removes them from the normal cyclic data polling cycle. Starting a slave or group of slaves returns them to the normal polling cycle.

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

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

MNETDPV1_MAILVAR_StartStopSlaves Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a PROFIBUS Master acyclic message to start or stop one or more slaves.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
Out	StartStopSlaveOUT	
StopSlaves	BOOL	
StartSlaves	BOOL	
IPAddress	ARRAY[1..4] OF INT	
SlaveAddress	ARRAY[0..125] OF BYTE	
RoutingRegister	INT	
In	StartStopSlaveIN	
StStOperationActive	BOOL	
StStOperationError	BOOL	
StStOperationSuccess	BOOL	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - StopSlaves	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to send Stop acyclic message to all slaves which have their **Out-SlaveNumber** array element set to 1. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - StartSlaves	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to send Start acyclic message to all slaves which have their **Out-SlaveNumber** array element set to 1. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - SlaveNumber	126-element 8-bit Byte Array	Enter 0 or 1 for each array element before you trigger the Start or Stop acyclic message 0 = Do not change this node (ignore acyclic message for this node) 1 = Change the state of this node to Stop for Stop acyclic message or Start for Start acyclic message (acyclic message affects this node)
Out - Routing Register	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - StStOperationActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - StStOperationError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - StStOperationSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

DFB Acyclic Mailbox Message: Set Slave Mode - Quantum

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

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

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

MNETDPV1_MAILVAR_SetSlaveMode Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a request to the PROFIBUS Master to send Sync and Freeze control messages to a slave or group of slaves.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
Out	SetSlaveModeOUT	
SetSlaveMode	BOOL	
SlaveAddress	BYTE	
GroupSelect	BYTE	
CommandControl	BYTE	
IPAddress	ARRAY[1..4] OF INT	
RoutingRegister	INT	
In	SetSlaveModeIN	
SlaveModeActive	BOOL	
SlaveModeError	BOOL	
SlaveModeSuccess	BOOL	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - SetSlaveMode	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to change the operating mode of a single slave or group of slaves. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries: 0-125, 127 If you enter a value from 0 to 125, the acyclic message will affect only the one slave device at this address. If you enter a value of 127, the acyclic message will affect a group of slaves, specified by the **Out-GroupSelect** parameter.
Out - GroupSelect	1 8-bit Byte	For details on how to set this parameter, see Set Slave Mode Message Structure (page 206)
Out - CommandControl	1 8-bit Byte	For details on how the set this parameter, see Set Slave Mode Message Structure (page 206)
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - Routing Register	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - SlaveModeActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - SlaveModeError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - SlaveModeSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

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

DFB Acyclic Mailbox Message: Set Address - Quantum

This acyclic message makes it possible to set the node address of a specified slave, provided that the slave supports this feature.

NOTE: The message data size depends on the actual slave implementation, range 0-240 bytes.

MNETDPV1_MAILVAR_SetSlaveAdd Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to change the network node address of a slave that supports this feature.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
Out	SetAddressOUT	
SetSlaveAddress	BOOL	
CurrentSlaveAddress	BYTE	
NewSlaveAddress	BYTE	
NoAddressChange	BYTE	
SlaveIdentNumber	INT	
IPAddress	ARRAY[1..4] OF INT	
RoutingRegister	INT	
In	SetAddressIN	
Error1	BYTE	
Length	BYTE	
ReturnCode	INT	
Fault	INT	
SetAddressData	ARRAY[0..199] OF BYTE	
SetSlaveAddActive	BOOL	
SetSlaveAddError	BOOL	
SetSlaveAddSuccess	BOOL	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - SetSlaveAddress	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to send a new network node number to a specific slave, thereby changing its node address. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - CurrentSlaveAddress	1 8-Bit Byte	Valid entries: 0 - 126 Set this variable to the current address of the slave whose address you wish to change.
Out - NewSlaveAddress	1 8-Bit Byte	Valid entries: 0 - 126 Set this variable to the new address you wish the slave to have.
Out - NoAddressChange	1 8-Bit Byte	Valid entries: 00h Change of address is still possible at a later stage. 01h - FFh Change or address is not possible until after slave reset This parameter specifies whether the slave address can be changed again at a later stage. if this is not allowed, then it is only possible to change the address with this function after initial reset. After the initial reset, the slave takes the default address 126.
Out - SlaveIdentNumber	1 16-bit Integer	Unique PROFIBUS Slave Identifier, assigned by PROFIBUS User Organization
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - Routing Register	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - Error1	1 8-bit Byte	Error Byte 1 of 4. Bytes 2 - 4 are reserved For details on Error Byte 1, see Acyclic Message Status Word (page 218)
In - Length	1 8-bit Byte	Range 6 to 244 Number of bytes sent by the slave as response data, if any.
In - ReturnCode	1 16-bit Integer	Refer to Return Codes (page 219) for additional information.

Variable Name	Size/Type	Description
In - Fault	1 16-bit Integer	If "Invalid other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Acyclic Message Status Word (page 218) **0001h:** Address out of range. **000Ah**: Failed to execute request. Refer to Return Codes (page 219) for additional information. **000Bh**: Remote station failure. Refer to Return Codes (page 219) for additional information. **00FFh**: gateway not initialized.
In - SetAddressData	200-element 8-bit Byte Array	Additional message data that may be returned by the slave
In - SetSlaveAddActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - SetSlaveAddError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - SetSlaveAddSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

{This page has been intentionally left blank.}

DFB Acyclic Mailbox Message: Get Database Information - Quantum

This acyclic message fetches information about the stored database (user-specific data that was downloaded to the gateway in message data bytes 1 to 32 via mailbox "FB_APPL_END_DATABASE_DOWNLOAD" or from the configuration tool).

This message also returns information about the amount of allocated I/O data space.

MNETDPV1_MAILVAR_GetDataBase Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a request to retrieve information about the PROFIBUS database.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
Out	GetDBaseOUT	
IPAddress	ARRAY[1..4] OF INT	
GetDatabaseInfo	BOOL	
RoutingRegister	INT	
In	GetDBaseIN	
Fault	INT	
InputLength	INT	
OutputLength	INT	
ProfibusInputSize	INT	
ProfibusOutputSize	INT	
ProfiBusCRCInfo	ARRAY[0..3] OF INT	
DatabaseOperationActive	BOOL	
DatabaseOperationError	BOOL	
DatabaseOperationSuc...	BOOL	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - GetDatabaseInfo	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to retrieve information about the PROFIBUS database. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - Routing Register	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - Fault	1 16-bit Integer	If 'Invalid Other' is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found in this variable. **0001h** - No database in FLASH memory, or download is in progress.
In - Input Length	1 16-bit Integer	Total Input Length Sum of all Input lengths for all slaves in the database, in bytes
In - Output Length	1 16-bit Integer	Total Output Length Sum of all Output lengths for all slaves in the database, in bytes
In - PROFIBUSInput Size	1 16-bit Integer	Required Initialization Input size for the current database. If the slaves are located in a contiguous block, this size is the same as the Total Input Length.
In - PROFIBUSOutput Size	1 16-bit Integer	Required Initialization Output size for the current database. If the slaves are located in a contiguous block, this size is the same as the Total Output Length.
In – PROFIBUSCRCInfo	4-element 16-bit Integer Array	This array holds two 32-bit values which are the PROFIBUS and Module CRC32 checksums. These values also appear in the gateway status variables and in ProSoft Configuration Builder (PCB).
In - DatabaseActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - DatabaseError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - DatabaseSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

DFB Acyclic Mailbox Message: Read Class 1 Acyclic Data - Quantum

Some, but not all, PROFIBUS DP-V1 slaves are capable of providing more data that can be configured for normal cyclic polling. Acyclic data read messages allow the PROFIBUS Master to request this additional data from slaves that can provide it. This acyclic message initiates a PROFIBUS DP-V1 *Class 1 Acyclic Read Request*. Refer to protocol specification EN50170 (DP-V1) for more information about this type of acyclic message.

MNETDPV1_MAILVAR_AcyclicRead Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a request to read PROFIBUS *DP-V1 Class 1 Acyclic Data*.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
Out	ReadAcyclicDataOUT	
ReadAcyclicdata	BOOL	
IPAddress	ARRAY[1..4] OF INT	
SlaveAddress	BYTE	
Slot	BYTE	
Index	BYTE	
Length	BYTE	
RoutingRegister	INT	
In	ReadAcyclicDataIN	
ErrorDecode	BYTE	
Error1	BYTE	
Error2	BYTE	
Length	BYTE	
ExtendedFault	INT	
Fault	INT	
AcyclicData	ARRAY[0..199] OF BYTE	
AcyclicReadActive	BOOL	
AcyclicReadError	BOOL	
AcyclicReadSuccess	BOOL	
AcyclicWrite	WriteAcyclicData	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - ReadAcyclicData	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to read PROFIBUS DP-V1 Acyclic Data from a specific slave on the network. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - SlaveAddress	1 8-bit Byte	Valid entries 0 - 125 Enter the node address of the slave device from which you wish to receive Acyclic Data.
Out - Slot	1 8-bit Byte	Valid entries 0 - n (where 'n' is the highest configured slot number on the target slave address - depends on device configuration.) Enter the slot number on the target node from which you wish to retrieve the acyclic data.
Out - Index	1 8-bit Byte	See slave device manufacturer for valid entries. This parameter is used to address the desired data block in the target slave.
Out - Length	1 8-bit Byte	See slave device manufacturer for valid entries. Maximum value is 200 bytes. See **NOTE** below. This parameter specifies the number of bytes of the data block that has to be read. If the slave data block length is less than requested, the length of the response will be the actual length of the data block. If the slave data block is greater or equal, then the response will contain the same amount of data.
Out - Routing Register	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - ErrorDecode	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation, or the EN50170 (DP-V1) protocol specification.
In - Error1	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation, or the EN50170 (DP-V1) protocol specification.
In - Error2	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation, or the EN50170 (DP-V1) protocol specification.

Variable Name	Size/Type	Description
In - Length	1 8-bit Byte	This value specifies the number of valid acyclic read data bytes returned by the slave.
In - ExtendedFault	1 16-bit Integer	Refer to Read Class 1 Acyclic Data Message Structure (page 214)
In - Fault	1 16-bit Integer	Refer to Read Class 1 Acyclic Data Message Structure (page 214)
In - AcyclicData[ ]	200-element 8-bit Byte Array	This array will hold the actual acyclic read data bytes returned by the slave.
In - AcyclicReadActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - AcyclicReadError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - AcyclicReadSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

NOTE: The MBP_MSTR function used on the Quantum platform has a lower data transfer limit than the DATA_EXCH function for the M340 platform. As a result, on the Quantum platform, the amount of data that can be transferred between a Quantum PAC and the ProLinx Gateway is 200, 8-bit bytes (100, 16-bit words) per message transaction.
This limit is more than enough for most PROFIBUS DP message telegrams. However, there are a few PROFIBUS messages that could contain more data in the PROFIBUS response than can be sent to the Quantum processor using the MBP_MSTR funcution for Modbus TCP/IP. It such cases, the PROFIBUS data is truncated after the first 200-bytes, any PROFIBUS data beyond that limit will not be sent to the Quantum, and the excess data will be lost.
The performance of this *Read Class 1 Acyclic* mailbox message may be affected by this limitation. Please consult your device manufacture's documentation to see if any of your PROFIBUS DP slave devices provide more than 200 bytes of acyclic read data per message.

{This page has been intentionally left blank.}

DFB Acyclic Mailbox Message: Write Class 1 Acyclic Data - Quantum

Some, but not all, PROFIBUS DP-V1 slaves are capable of receiving more data that can be configured for normal cyclic polling. Acyclic data write messages allow the PROFIBUS Master to send this additional data to slaves that can receive it. This acyclic message initiates a PROFIBUS *DP-V1 Class 1 Acyclic Write Request*. Refer to PROFIBUS DP-V1 specification EN50170 (DP-V1) for more information.

MNETDPV1_MAILVAR_AcyclicWrite Variables - Quantum

This variable structure is the one to use with your control and sequencing logic when you want to send a PROFIBUS *DP-V1 Class 1 Acyclic Write Request*.

MNETDPV1_DataOut	MNETDPV1_DataOutF	%MW1000
MNETDPV1_MAILVAR	MNETDPV1_MAILVAR	
AcyclicRead	ReadAcyclicData	
AcyclicWrite	WriteAcyclicData	
Out	WriteAcyclicDataOUT	
WriteAcyclicdata	BOOL	
SlaveAddress	BYTE	
Slot	BYTE	
Index	BYTE	
LengthOfBytes	BYTE	
IPAddress	ARRAY[1..4] OF INT	
WriteData	ARRAY[0..199] OF BYTE	
RoutingRegister	INT	
In	WriteAcyclicDataIN	
ErrorDecode	BYTE	
Error1	BYTE	
Error2	BYTE	
Length	BYTE	
ExtendedFault	INT	
Fault	INT	
AcyclicWriteActive	BOOL	
AcyclicWriteError	BOOL	
AcyclicWriteSuccess	BOOL	
GetConfig	GetSlaveConfiguration	
GetDiag	GetDiagnosticData	
GetLiveList	GetLiveListData	
SetSlaveAdd	SetSlaveAddress	
SetSlaveMode	SetSlaveMode	
StartStopSlaves	StartStopSlaves	
SetOperatMode	SetOperationMode	
GetDataBase	GetDataBases	
MNETDPV1_SLVDIAG	SLVDIAGF	%MW2276

Variable Name	Size/Type	Description
Out - WriteAcyclicData	1 Single-Bit Boolean	This is the bit your control and sequencing logic will use to trigger a message to tell the PROFIBUS Master to write PROFIBUS DP-V1 Acyclic Data to a specific slave on the network. Set this bit to one (1) whenever no other messages are active and you want to send this acyclic message.
Out - SlaveAddress	1 8-bit Byte	Valid entries 0 - 125 Enter the node address of the slave device to which you wish to send Acyclic Data.
Out - Slot	1 8-bit Byte	Valid entries 0 - n (where 'n' is the highest configured slot number on the target slave address - depends on device configuration.) Enter the slot number on the target node to which you wish to send the acyclic data.
Out - Index	1 8-bit Byte	See slave device manufacturer for valid entries. This parameter is used to address the desired data block in the target slave.
Out - LengthOfBytes	1 8-bit Byte	Valid entries: 1 - 200 Enter the number of acyclic data bytes you wish to write to the target slave.
Out - IPAddress[ ]	4-element 16-bit Integer Array	Each of the four integer elements holds one octet of the message destination device's IP Address. For this application, it will be the the ProLinx gateway's Ethernet port address.
Out - WriteData	200-element 8-bit Byte Array	This array will be used to hold the acyclic data, in bytes, that you wish to write to the target slave.
Out - RoutingRegister	1 16-bit Integer	Default value for Quantum processors with built-in Ethernet port is 254 (16#FE00 hex) For details, see "MBP_MSTR, Ethernet (Quantum) - Control Block for TCP/IP Ethernet" in Quantum Help Files, or Quantum Communication Control and Data Buffer Variables and DDTs (page 137)
In - ErrorDecode	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation or the EN50170 (DP-V1) protocol specification.
In - Error1	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation or the EN50170 (DP-V1) protocol specification.
In - Error2	1 8-bit Byte	If 'Fault' contains error code 0010h, more information can be found here. For information on how to interpret these values, refer to the slave device manufacturer's documentation or the EN50170 (DP-V1) protocol specification.
In - Length	1 8-bit Byte	The value seen here specifies the number of valid data bytes returned by the slave, if any.
In - ExtendedFault	1 16-bit Integer	For details, see Write Class 1 Acyclic Data Message Structure (page 216)

Variable Name	Size/Type	Description
In - Fault	1 16-bit Integer	For details, see Write Class 1 Acyclic Data Message Structure (page 216)
In - AcyclicWriteActive	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP message has been initiated and is being processed.
In - AcyclicWriteError	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is not successfully received. When this bit is set, it indicates your control and sequencing logic should retry the command.
In - AcyclicWriteSuccess	1 Single-bit Boolean	The DFB will set this bit when the Modbus TCP/IP response is successfully received and the PROFIBUS cyclic input data variables have been updated.

{This page intentionally left blank.}

4.6 PROFIBUS Acyclic Telegram (Message) Block Structures

4.6.1 Set Operating Mode Message Structure

Parameter	Description
Command Initiator	Application
Command Name	FB_APPL_SET_OPERATION_MODE
Command Number	0002h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Set Operating Mode

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	0002h		0002h		Set Operation Mode
Data size	0000h		0000h		
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Req. Mode	Conf. Req	Act. Mode	Conf. Req	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Appl. Specific Error Code		
Extended word 8	-		Fault Information		

Mode

40h: STOP

80h: CLEAR

C0h: OPERATE

Conf. Req.

00h: Confirmation is not required

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

Fault Information

If "Invalid Other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Return Codes (page 219) for more information.

0001h: Invalid operating mode

0002h Invalid 'Conf.Req.' setting

0003h Timeout or incorrect answering of the 'FB_ABM_SHIFT_OPERATION_MODE_REQ' message

0004h Application did not permit changing the operation mode. More information might be supplied in the 'Application Specific Error Code'.

00FEh Command not possible in 'Class 2-Only' mode.

00FFh: gateway not initialized

4.6.2 Get Live List Message Structure

Parameter	Description
Command Initiator	Application
Command Name	FB_APPL_GET_LIVE_LIST
Command Number	0018h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Get Live List

	Command	Response	
Message ID	(ID)	(ID)	
Acyclic Message Status Word	4002h	0002h	
Command	0018h	0018h	Get Live List
Data size	0000h	007Fh	127 Bytes of Data
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	Return Code	
Extended word 8	-	Fault Information	
Message Data byte 1		Station Type 0	Response Data Byte 1
Message Data byte 2		Station Type 1	Response Data Byte 2
Message Data byte 3		Station Type 2	Response Data Byte 3
...		...	...
Message Data byte "n"		Station Type 126	Response Data Byte 127

Acyclic Message Status Word

Refer to Acyclic Message Status Word (page 218).

Station Type [0 ... 126]

00h: Slave Station

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

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

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

04h: Station does not exist

Fault Information

If "Invalid Other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Acyclic Message Status Word (page 218).

000Ah: Failed to build Live List.

00FFh: gateway offline (not initialized or no valid database)

4.6.3 Get (Extended) Slave Diagnostics Message Structure

Parameter	Description
Command Initiator	Application
Command Name	FB_APPL_GET_SLAVE_DIAG
Command Number	0004h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Get Slave Diagnostics

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	0004h		0004h		Get Slave Diagnostics
Data size	0000h		(Size of data)		
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slave Address	Type of Request	Slave Address	Type of Request	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error Code 1	Error Code 2	
Extended word 6	-		Error Code 3	Error Code 4	
Extended word 7	-		Return Code		
Extended word 8	-		Fault Information		
			Station Status 1	Station Status 2	Response data word 1
			Station Status 3	Master Address	Response data word 2
			Ident Number		Response data word 3
			Extended Diagnostic Data		Response data word 4 ... ... Response data word n

Acyclic Message Status Word

Refer to Acyclic Message Status Word (page 218).

Slave Address

Range 0 to 125, specifies the slave from which to read diagnostics.

Type of request

00h: Internal slave diagnostic request. Returns the diagnostic information stored in the master. Can only be requested for slaves configured by the master.

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

01h: External slave diagnostic request. Sends a diagnostic request on the network to the specified slave. Can be requested for all slaves on the network.

Error code [1 ...4]

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

Return Code

Refer to Mailbox Messaging Error Codes

Fault Information

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

0001h: Address out of range.

0002h: Incorrect "Type of request"

000Ah: Failed to read diagnostic data from slave. Refer to Return Codes (page 219) for additional fault information.

000Bh: Remote station failure. Refer to Return Codes (page 219) for additional fault information.

00FEh: Command not possible; gateway operates as a Class 2 master only.

00FFh: gateway offline (not initialized or no valid database).

Station Status [1 ... 3]

Refer to EN50170 Vol. 2 for more information.

Master Address

Address of the master that parameterized the slave.

Ident Number

Unique ID assigned by the PROFIBUS User Organization.

Extended Diagnostic Data

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

4.6.4 Get Slave Configuration Message Structure

Parameter	Description
Command Initiator	Application
Command Name	FB_APPL_GET_SLAVE_CONFIG
Command Number	0005h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Get Slave Configuration

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	0005h		0005h		Get Slave Configuration
Data size	0000h		(Size of data)		Number of identifier bytes (n)
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slave Address	-	Slave Address	-	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error Code 1	Error Code 2	
Extended word 6	-		Error Code 3	Error Code 4	
Extended word 7	-		Return Code		
Extended word 8	-		Fault Information		
			Identifier byte 1		Response data word 1
			Identifier byte 2		Response data word 2
			Identifier byte 3		Response data word 3
			...		...
			Identifier byte n		Response data word n

Acyclic Message Status Word

Refer to Acyclic Message Status Word (page 218).

Slave Address

Range 0 to 125, specifies the slave from which to read the configuration.

Error Code [1 ... 4]

If "Return Code" equals 8030h ("Negative indication from lower layer"), status values according to the DP-specification may be available in "Error Code 1", Error Codes 2 through 3 are reserved. Refer to Mailbox Messaging Error Codes.

Return Code

Refer to Mailbox Messaging Error Codes.

Fault Information

If "Invalid other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Acyclic Message Status Word (page 218).

0001h: Address out of range.

000Ah: Failed to execute request. Refer to Return Codes (page 219) for additional information.

000Bh: Remote station failure. Refer to Return Codes (page 219) for additional information.

00FFh: gateway not initialized.

Identifier Bytes [1 ... n]

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

4.6.5 Stop Slave Message Structure

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

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

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

Command and Response Layout: Stop Slave

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	000Ch		000Ch		Stop Slave
Data size	007Eh		007Eh		
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	-		-		
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Additional Fault Information		
Extended word 8	-		Fault Information		
Message data word 1	Slave 0	Slave 1	Slave 0	Slave 1	
Message data word 2	Slave 2	Slave 3	Slave 2	Slave 3	
Message data word 3 to 62	...	...	...	...	
Message data word 63	Slave 124	Slave 125	Slave 124	Slave 125	

Command:

- Message data word 1-63

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

Response:

- Acyclic Message Status Word (in response header)

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

- Additional Fault information (Extended word 7)

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

- Fault information (Extended word 8)

  0001h: Invalid setting in Message data word 1-63 of the command.
  0002h: At least one slave reports a warning. Refer to Message data word 1-63.
  000Ah: Failed to execute request. Additional fault information is to be found in Extended word 7.
  00FEh: Command not possible, gateway operates as class 2 master only.
  00FFh: gateway not initialized (this command is only possible after END_INIT).

- Message data word 1-63

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

4.6.6 Start Slave Message Structure

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

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

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

Command and Response Layout: Start Slave

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	000Bh		000Bh		Start Slave
Data size	007Eh		007Eh		
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1					
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Additional Fault Information		
Extended word 8	-		Fault Information		
Message data word 1	Slave 0	Slave 1	Slave 0	Slave 1	
Message data word 2	Slave 2	Slave 3	Slave 2	Slave 3	
Message data word 3 to 62	...	...	...	...	
Message data word 63	Slave 124	Slave 125	Slave 124	Slave 125	

Command:

- Message data word 1-63

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

Response:

- Acyclic Message Status Word (in response header)

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

- Additional Fault information (Extended word 7)

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

- Fault information (Extended word 8)

  0001h: Invalid setting in Message data word 1-63 of the command.
  0002h: At least one slave reports a warning. Refer to Message data word 1-63.
  000Ah: Failed to execute request. Additional fault information is to be found in Extended word 7.
  00FEh: Command not possible, gateway operates as class 2 master only.
  00FFh: gateway not initialized (this command is only possible after END_INIT).

- Message data word 1-63

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

4.6.7 Set Slave Mode Message Structure

Parameter	Description
Command Initiator	Application
Command Name	FB_APPL_SET_SLAVE_MODE
Command Number	0003h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Set Slave Mode

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	0003h		0003h		Set Slave Mode
Data size	0000h		0000h		
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slave Address	Group Select	Slave Address	Group Select	
Extended word 2	Control Command		Control Command		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Extended Fault Information		
Extended word 8			Fault Information		

Acyclic Message Status Word

Refer to Acyclic Message Status Word (page 218).

Slave Address

Range 0 to 125; 127

If the request applies for only one slave, that Slave Address must be entered in the range 1 to 125. If a slave group is to be addressed, Slave Address should be 127 (Multicast address).

Group Select

Range 01h to FFh (Bit Coded)

This parameter determines which group to address. Refer to the following example:

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

Example: To address Group 1, 2, and 4, the Group Select value should be 0Dh. If an individual slave should be addressed, the correct group selection must also be made, because the slave will ignore the message if it does not belong to the requested group(s).

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

Control Command

This parameter specifies the command to send:

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

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

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

Fault Information and Extended Fault Information

"Fault Information" Contents		"Extended Fault Information" Contents	
0001h	Address out of range	-	
0002h	Group number 0 not permitted	-	
000Ah	Failed to send Global Control request	000Ah	Incorrect operation mode (Clear/Operate Only)
		5001h	Invalid Freeze Group (Group is not initiated to be Freeze Group)
		5002h	Invalid Sync Group (Group is not initiated to be Sync Group)
		5003h	Incorrect Control Command
		5004h	No Sync or Freeze groups enabled in Master configuration
00FEh	Command not possible in Class 2 only mode	-	
00FFh	Module not initialized	-	

{This page intentionally left blank.}

4.6.8 Set Slave Address Message Structure

Parameter	Description
Command Initiator	Application
Command Name	FB_APPL_SET_SLAVE_ADDRESS
Command Number	0006h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Set Slave Address

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	0006h		0006h		Set Slave Address
Data size	(Size of data bytes in hex)		(Size of data bytes in hex)		Number of Slave Data bytes (n)
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Current Slave Address	New Slave Address	Current Slave Address	New Slave Address	
Extended word 2	Slave Ident Number		Slave Ident Number		
Extended word 3	No_Add_Chg		No_Add_Chg		
Extended word 4	-		-		
Extended word 5	-		Error Code 1	Error Code 2	
Extended word 6	-		Error Code 3	Error Code 4	
Extended word 7	-		Return Code		
Extended word 8			Fault Information		
Message Data Byte 1	Slave Data 1		Slave Data 1		
Message Data Byte 2	Slave Data 2		Slave Data 2		
Message Data Byte 3	Slave Data 3		Slave Data 3		
...					
Message Data Byte 'n' (where 'n' <= 240)	Slave Data 'n'		Slave Data 'n'		

Acyclic Message Status Word

Refer to Acyclic Message Status Word (page 218).

Current Slave Address

Range 0 to 125

Specifies the current address of the slave.

New Slave Address

Range 0 to 125

Specifies the new address of the slave.

Slave Ident Number

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

No_Add_Change

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

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

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

Error Code [1 ... 4]

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

(See Return Codes (page 219) and Error Codes)

Fault Information and Extended Fault Information

"Fault Information" Contents		"Extended Fault Information" Contents
0001h	Current Slave Address out of range	-
0002h	New Slave Address out of range	-
000Ah	Failed to send Global Control request	For additional fault information, see Return Codes (page 219)
000Bh	Remote Station Failure	For additional fault information, see Return Codes (page 219)
00FFh	Module offline	(not initialized or no valid database)

Slave Data

With this parameter it is possible to deliver user specific data. The data is stored in the slave if possible (EEPROM, FLASH, etc.)

4.6.9 Get Database Information Message Structure

Parameter	Description
Command initiator	Application
Command Name	FB_APPL_GET_DATABASE_INFO
Command number	0017h
Fragmented	No
Firmware Revision	All

Command and response layout: Get Database Information

	Command	Response	
Message ID	(ID)	(ID)	
Acyclic Message Status Word	4002h	0002h	
Command	0017h	0017h	Get Database Info
Data size	0000h	0040h	
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1		Total Output Length	
Extended word 2	-	Total Input Length	
Extended word 3	-	Init Output Size	
Extended word 4	-	Init Input Size	
Extended word 5	-	No. of Slaves / -	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	Fault Information	
		Database Description (ASCII, 64 characters)	Response data word 1 to 32

Total Input Length, Total Output Length: The sum of Input/Output lengths for all slaves in the database (in bytes).

Init Input size, Init Output size: Required initialization Input/Output sizes for the current database. If the slaves are located in a contiguous block, these sizes are the same as total Input/Output lengths.

Note: The input/output direction refers to the directions and naming used for the DPRAM areas, not to the input/output directions used in the bus database or the NetTool-PB.

Database Description: String of ASCII characters that describes the data base file. This is the string that was written to the database by the "FB_APPL_END_DATABASE_DOWNLOAD" command.

No. of Slaves: Number of configured slaves in the database.

Fault Information

If "Invalid Other" is returned in the Acyclic Message Status Word in the header of the response, information about the fault can be found here. Refer to Return Codes (page 219) for more information.

0001h: No database in flash, or download in progress.

4.6.10 Read Class 1 Acyclic Data Message Structure

Parameter	Description
Command Initiator	Application
Command Name	FB_APPL_MSAC1_READ
Command Number	0020h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Acyclic Read

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	0020h		0020h		Acyclic Class 1 Read
Data size	0000h		(Size of data)		Number of data bytes (n)
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slave Addr.	Slot Number	Slave Addr.	Slot Number	
Extended word 2	Index	Length	Index	Length	
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-			Error Decode	
Extended word 6	-		Error Code 1	Error Code 2	
Extended word 7	-		Extended Fault information		
Extended word 8	-		Fault Information		
			Data 1		Response Data byte 1
			Data 2		Response Data byte 1
			Data 3		Response Data byte 1
			...		...
			Data n		Response Data byte 1

Acyclic Message Status Word

Refer to Acyclic Message Status Word (page 218).

Slave Address

Station address of the slave responder.

Slot Number and Slot Index

Used in the slave to address the desired data block.

Length

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

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

Data [1 ... n]

Returned data.

Fault Information and Extended Fault Information

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

"Fault Information"		"Extended Fault Information" Contents
0001h	Address out of range	-
000Ah	Failed to execute request	Refer to Return Codes (page 219).
000Bh	Remote station failure	
0010h	Remote station DP-V1 failure	Function_Number
0011h	Length out of range (>240 bytes)	-
0012h	Slave does not support DP-V1	-
0013h	Slave not active or not present in configuration	-
00FEh	Command not possible in "Class 2-Only" mode	-
00FFh	Module offline (not initialized or no valid database)	-

Error Decode, Error Code 1 and Error Code 2

If "Fault Information" contains error code 0010h, refer to the slave device manufacturer's documentation, or the EN50170 (DP-V1) protocol specification.

4.6.11 Write Class 1 Acyclic Data Message Structure

Parameter	Description
Command Initiator	Application
Command Name	FB_APPL_MSAC1_WRITE
Command Number	0021h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout: Acyclic Write

	Command		Response		
Message ID	(ID)		(ID)		
Acyclic Message Status Word	4002h		0002h		
Command	0021h		0021h		Acyclic Write
Data size	(Size of data)		(Size of data)		Number of data bytes (n)
Frame count	0001h		0001h		
Frame number	0001h		0001h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slave Addr.	Slot Number	Slave Addr.	Slot Number	
Extended word 2	Index	Length	Index	Length	
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-			Error Decode-	
Extended word 6	-		Error Code 1	Error Code 2	
Extended word 7	-		Extended Fault information		
Extended word 8	-		Fault Information		
Message Data byte 1	Data 1		Data 1		
Message Data byte 2	Data 2		Data 2		
Message Data byte 3	Data 3		Data 3		
...	...		...		
Message Data byte n	Data n		Data n		

Acyclic Message Status Word

Refer to Acyclic Message Status Word (page 218).

Slave Address

Station address of the slave responder.

Slot Number and Slot Index

Used in the slave to address the desired data block.

Length

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

Data [1 ... n]

Data that should be written.

Fault Information and Extended Fault Information

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

"Fault Information"		"Extended Fault Information" Contents
0100h	Address out of range	-
0A00h	Failed to execute request	Refer to Return Codes (page 219).
0B00h	Remote station failure	
1000h	Remote station DP-V1 failure	Function_Number
1100h	Length out of range (>240 bytes)	-
1200h	Slave does not support DP-V1	-
1300h	Slave not active or not present in configuration	-
FE00h	Command not possible in "Class 2-Only" mode	-
FF00h	Module offline (not initialized or no valid database)	-

Error Decode, Error Code 1, and Error Code 2

If "Fault Information" contains error code 1000h, more information according to the DP-V1 specification can be found here.

4.7 Mailbox Messaging Error Codes

4.7.1 Acyclic Message Status Word

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

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

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

4.7.2 Return Codes

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

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

4.7.3 Error Codes

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

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

4.7.4 DP-V1 Error Codes

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

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

4.7.5 Command Error Codes

Errors reported from the command list of the gateway require 8-bytes or 4-words per command. If the first 7 bytes of the error are 0xFF, this is a gateway generated error as follows:

Value of Last Byte	Error Definition
0x00	Too few parameters for command in command list section of configuration file.
0x01	Invalid type value specified for command.
0x02	Invalid database offset specified for command.
0x03	Invalid swap type code specified for command.
0x04	Invalid database trigger address
0x05	Invalid database address and count combination
0x10	Invalid function code specified for command.
0xFF	Response timeout for command recognized.

Refer to Error Codes (page 219) for an explanation of other error codes.

The 8-bytes (4-words) represent the extended words 5 to 8 in response messages.

5 Conclusion

In This Chapter

5.1 ProSoft Technology Support

Information outside the scope of this manual can be obtained from ProSoft Technology in several ways:

- **Web Site Support:** You can visit our web site and download documents from the product web pages at : www.prosoft-technology.com.
- **Driver Manuals:** These are detailed reference guides to the protocol implementations, including configuration options, functional overview, diagnostics and troubleshooting procedures, and product specifications. There will be one manual for the MNET protocol and one for the PROFIBUS DP-V1 Master protocol.
- **Datasheet:** Contains a brief description of the 5204SE gateway hardware and protocol implementations, general, and functional specifications in a condensed form for easy inclusion in sales materials, proposals, technical specifications documents and other such requirements.
- **Telephone and Email Support:** For contact information, refer to How to Get Help (page 224).

5.2 How to Get Help

ProSoft Technology has several ways for customers to quickly and easily acquire knowledge about our solutions. Also, if you have any comments, recommendations, or suggestions regarding our solutions, please let us know how we may better serve you.

Contact Us: You can always call or email us with your comments and questions.

- **Telephone Technical Support:** You can call ProSoft Technical Support, worldwide.
  - In North America: 661- 716-5100 (English, Spanish, and Japanese)
  - In Malaysia: +603.7724.2080 (Chinese, English, and Japanese)
  - In China: +86.21.5109.7557 (Chinese and English)
  - In Europe: +33.(0)5.34.36.87.30 (French and English)
  - In the Middle East and Africa: +971.(0).4.214.6911 (English and Hindi)
  - In Brasil: +55.11.5083.3776 (Portuguese and English)
  - In Mexico and Central America: +52.222.264.18.14 (Spanish and English)
- **Email Technical Support:** You can email your support questions and requests.
  - From Anywhere in the World: Support@prosoft-technology.com
  - From the Asia Pacific area: asiapc@prosoft-technology.com
  - From Europe: support.emea@prosoft-technology.com
  - From the Middle East or Africa: mea@prosoft-technology.com
  - From Brasil: brasil@prosoft-technology.com
  - From Mexico and Central America: latinam@prosoft-technology.com

Web-based Support: Available through our corporate web site, *www.prosoft-technology.com/support*

- **Live Chat:** (6am to 5pm PST): Communicate with a Technical Support Engineer on-line. This is just one more way to get one-on-one support from our knowledgeable support staff.
- **Downloads:** Get manuals, datasheets, configuration utilities, and more.
- **Knowledgebase:** Type a question into our knowledgebase search engine. Answers come from a technical support knowledge database built from helping inquisitive customers like you.
- **Bulletin Board:** Here's a public forum just for you. Make comments, ask questions, and get to know ProSoft's automation community. Register, login, and join the discussion.
- **Frequently Asked Questions**: Viewing our FAQ pages could get you the answers you need immediately. Check back regularly for updates.

6 Support, Service & Warranty

In This Chapter

ProSoft Technology, Inc. (ProSoft) is committed to providing the most efficient and effective support possible. Before calling, please gather the following information to assist in expediting this process:

1 Product Version Number
2 System architecture
3 Network details

If the issue is hardware related, we will also need information regarding:

1 Module configuration and contents of file
   - Module Operation
   - Configuration/Debug status information
   - LED patterns

2 Information about the processor and user data files as viewed through and LED patterns on the processor.
3 Details about the serial devices interfaced, if any.

6.1 How to Contact Us: Technical Support

Internet	Web Site: www.prosoft-technology.com/support
	E-mail address: support@prosoft-technology.com

Asia Pacific

+603.7724.2080, support.asia@prosoft-technology.com
Languages spoken include: Chinese, English

Europe (location in Toulouse, France)

+33 (0) 5.34.36.87.20, support.EMEA@prosoft-technology.com
Languages spoken include: French, English

North America/Latin America (excluding Brasil) (location in California)

+1.661.716.5100, support@prosoft-technology.com
Languages spoken include: English, Spanish
For technical support calls within the United States, an after-hours answering system allows pager access to one of our qualified technical and/or application support engineers at any time to answer your questions.

Brasil (location in Sao Paulo)

+55-11-5084-5178, eduardo@prosoft-technology.com
Languages spoken include: Portuguese, English

6.2 Return Material Authorization (RMA) Policies and Conditions

The following RMA Policies and Conditions (collectively, "RMA Policies") apply to any returned Product. These RMA Policies are subject to change by ProSoft without notice. For warranty information, see "Limited Warranty". In the event of any inconsistency between the RMA Policies and the Warranty, the Warranty shall govern.

6.2.1 All Product Returns:

a) In order to return a Product for repair, exchange or otherwise, the Customer must obtain a Returned Material Authorization (RMA) number from ProSoft and comply with ProSoft shipping instructions.
b) In the event that the Customer experiences a problem with the Product for any reason, Customer should contact ProSoft Technical Support at one of the telephone numbers listed above (page 225). A Technical Support Engineer will request that you perform several tests in an attempt to isolate the problem. If after completing these tests, the Product is found to be the source of the problem, we will issue an RMA.
c) All returned Products must be shipped freight prepaid, in the original shipping container or equivalent, to the location specified by ProSoft, and be accompanied by proof of purchase and receipt date. The RMA number is to be prominently marked on the outside of the shipping box. Customer agrees to insure the Product or assume the risk of loss or damage in transit. Products shipped to ProSoft using a shipment method other than that specified by ProSoft or shipped without an RMA number will be returned to the Customer, freight collect. Contact ProSoft Technical Support for further information.
d) A 10% restocking fee applies to all warranty credit returns whereby a Customer has an application change, ordered too many, does not need, and so on.

6.2.2 Procedures for Return of Units Under Warranty:

A Technical Support Engineer must approve the return of Product under ProSoft's Warranty:

a) A replacement module will be shipped and invoiced. A purchase order will be required.
b) Credit for a product under warranty will be issued upon receipt of authorized product by ProSoft at designated location referenced on the Return Material Authorization.

6.2.3 Procedures for Return of Units Out of Warranty:

a) Customer sends unit in for evaluation
b) If no defect is found, Customer will be charged the equivalent of $100 USD, plus freight charges, duties and taxes as applicable. A new purchase order will be required.
c) If unit is repaired, charge to Customer will be 30% of current list price (USD) plus freight charges, duties and taxes as applicable. A new purchase order will be required or authorization to use the purchase order submitted for evaluation fee.

The following is a list of non-repairable units:

- 3150 - All
- 3750
- 3600 - All
- 3700
- 3170 - All
- 3250
- 1560 - Can be repaired, only if defect is the power supply
- 1550 - Can be repaired, only if defect is the power supply
- 3350
- 3300
- 1500 - All

6.3 LIMITED WARRANTY

This Limited Warranty ("Warranty") governs all sales of hardware, software and other products (collectively, "Product") manufactured and/or offered for sale by ProSoft, and all related services provided by ProSoft, including maintenance, repair, warranty exchange, and service programs (collectively, "Services"). By purchasing or using the Product or Services, the individual or entity purchasing or using the Product or Services ("Customer") agrees to all of the terms and provisions (collectively, the "Terms") of this Limited Warranty. All sales of software or other intellectual property are, in addition, subject to any license agreement accompanying such software or other intellectual property.

6.3.1 What Is Covered By This Warranty

a) *Warranty On New Products*: ProSoft warrants, to the original purchaser, that the Product that is the subject of the sale will (1) conform to and perform in accordance with published specifications prepared, approved and issued by ProSoft, and (2) will be free from defects in material or workmanship; provided these warranties only cover Product that is sold as new. This Warranty expires three years from the date of shipment (the "Warranty Period"). If the Customer discovers within the Warranty Period a failure of the Product to conform to specifications, or a defect in material or workmanship of the Product, the Customer must promptly notify ProSoft by fax, email or telephone. In no event may that notification be received by ProSoft later than 39 months. Within a reasonable time after notification, ProSoft will correct any failure of the Product to conform to specifications or any defect in material or workmanship of the Product, with either new or used replacement parts. Such repair, including both parts and labor, will be performed at ProSoft's expense. All warranty service will be performed at service centers designated by ProSoft.
b) *Warranty On Services*: Materials and labor performed by ProSoft to repair a verified malfunction or defect are warranteed in the terms specified above for new Product, provided said warranty will be for the period remaining on the original new equipment warranty or, if the original warranty is no longer in effect, for a period of 90 days from the date of repair.

6.3.2 What Is Not Covered By This Warranty

a) ProSoft makes no representation or warranty, expressed or implied, that the operation of software purchased from ProSoft will be uninterrupted or error free or that the functions contained in the software will meet or satisfy the purchaser's intended use or requirements; the Customer assumes complete responsibility for decisions made or actions taken based on information obtained using ProSoft software.
b) This Warranty does not cover the failure of the Product to perform specified functions, or any other non-conformance, defects, losses or damages caused by or attributable to any of the following: (i) shipping; (ii) improper installation or other failure of Customer to adhere to ProSoft's specifications or instructions; (iii) unauthorized repair or maintenance; (iv) attachments, equipment, options, parts, software, or user-created programming (including, but not limited to, programs developed with any IEC 61131-3, "C" or any variant of "C" programming languages) not furnished by ProSoft; (v) use of the Product for purposes other than those for which it was designed; (vi) any other abuse, misapplication, neglect or misuse by the Customer; (vii) accident, improper testing or causes external to the Product such as, but not limited to, exposure to extremes of temperature or humidity, power failure or power surges; or (viii) disasters such as fire, flood, earthquake, wind and lightning.

c) The information in this Agreement is subject to change without notice. ProSoft shall not be liable for technical or editorial errors or omissions made herein; nor for incidental or consequential damages resulting from the furnishing, performance or use of this material. The user guide included with your original product purchase from ProSoft contains information protected by copyright. No part of the guide may be duplicated or reproduced in any form without prior written consent from ProSoft.

6.3.3 Disclaimer Regarding High Risk Activities

Product manufactured or supplied by ProSoft is not fault tolerant and is not designed, manufactured or intended for use in hazardous environments requiring fail-safe performance including and without limitation: the operation of nuclear facilities, aircraft navigation of communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly or indirectly to death, personal injury or severe physical or environmental damage (collectively, "high risk activities"). ProSoft specifically disclaims any express or implied warranty of fitness for high risk activities.

6.3.4 Intellectual Property Indemnity

Buyer shall indemnify and hold harmless ProSoft and its employees from and against all liabilities, losses, claims, costs and expenses (including attorney's fees and expenses) related to any claim, investigation, litigation or proceeding (whether or not ProSoft is a party) which arises or is alleged to arise from Buyer's acts or omissions under these Terms or in any way with respect to the Products. Without limiting the foregoing, Buyer (at its own expense) shall indemnify and hold harmless ProSoft and defend or settle any action brought against such Companies to the extent based on a claim that any Product made to Buyer specifications infringed intellectual property rights of another party. ProSoft makes no warranty that the product is or will be delivered free of any person's claiming of patent, trademark, or similar infringement. The Buyer assumes all risks (including the risk of suit) that the product or any use of the product will infringe existing or subsequently issued patents, trademarks, or copyrights.

a) Any documentation included with Product purchased from ProSoft is protected by copyright and may not be duplicated or reproduced in any form without prior written consent from ProSoft.
b) ProSoft's technical specifications and documentation that are included with the Product are subject to editing and modification without notice.
c) Transfer of title shall not operate to convey to Customer any right to make, or have made, any Product supplied by ProSoft.
d) Customer is granted no right or license to use any software or other intellectual property in any manner or for any purpose not expressly permitted by any license agreement accompanying such software or other intellectual property.

e) Customer agrees that it shall not, and shall not authorize others to, copy software provided by ProSoft (except as expressly permitted in any license agreement accompanying such software); transfer software to a third party separately from the Product; modify, alter, translate, decode, decompile, disassemble, reverse-engineer or otherwise attempt to derive the source code of the software or create derivative works based on the software; export the software or underlying technology in contravention of applicable US and international export laws and regulations; or use the software other than as authorized in connection with use of Product.

f) **Additional Restrictions Relating To Software And Other Intellectual Property**

   In addition to compliance with the Terms of this Warranty, Customers purchasing software or other intellectual property shall comply with any license agreement accompanying such software or other intellectual property. Failure to do so may void this Warranty with respect to such software and/or other intellectual property.

6.3.5 Disclaimer of all Other Warranties

The Warranty set forth in What Is Covered By This Warranty (page 228) are in lieu of all other warranties, express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

*6.3.6 Limitation of Remedies ***

In no event will ProSoft or its Dealer be liable for any special, incidental or consequential damages based on breach of warranty, breach of contract, negligence, strict tort or any other legal theory. Damages that ProSoft or its Dealer will not be responsible for included, but are not limited to: Loss of profits; loss of savings or revenue; loss of use of the product or any associated equipment; loss of data; cost of capital; cost of any substitute equipment, facilities, or services; downtime; the claims of third parties including, customers of the Purchaser; and, injury to property.

** Some areas do not allow time limitations on an implied warranty, or allow the exclusion or limitation of incidental or consequential damages. In such areas, the above limitations may not apply. This Warranty gives you specific legal rights, and you may also have other rights which vary from place to place.

6.3.7 Time Limit for Bringing Suit

Any action for breach of warranty must be commenced within 39 months following shipment of the Product.

6.3.8 No Other Warranties

Unless modified in writing and signed by both parties, this Warranty is understood to be the complete and exclusive agreement between the parties, suspending all oral or written prior agreements and all other communications between the parties relating to the subject matter of this Warranty, including statements made by salesperson. No employee of ProSoft or any other party is authorized to make any warranty in addition to those made in this Warranty. The Customer is warned, therefore, to check this Warranty carefully to see that it correctly reflects those terms that are important to the Customer.

6.3.9 Allocation of Risks

This Warranty allocates the risk of product failure between ProSoft and the Customer. This allocation is recognized by both parties and is reflected in the price of the goods. The Customer acknowledges that it has read this Warranty, understands it, and is bound by its Terms.

6.3.10 Controlling Law and Severability

This Warranty shall be governed by and construed in accordance with the laws of the United States and the domestic laws of the State of California, without reference to its conflicts of law provisions. If for any reason a court of competent jurisdiction finds any provisions of this Warranty, or a portion thereof, to be unenforceable, that provision shall be enforced to the maximum extent permissible and the remainder of this Warranty shall remain in full force and effect. Any cause of action with respect to the Product or Services must be instituted in a court of competent jurisdiction in the State of California.

Index