



**inRAx<sup>®</sup>**  
**MVI56-MNETR**

**ControlLogix Platform**

Modbus TCP/IP Interface Module with  
Reduced Data Block

September 24, 2010

## Your Feedback Please

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

## How to Contact Us

### **ProSoft Technology**

5201 Truxtun Ave., 3rd Floor

Bakersfield, CA 93309

+1 (661) 716-5100

+1 (661) 716-5101 (Fax)

[www.prosoft-technology.com](http://www.prosoft-technology.com)

[support@prosoft-technology.com](mailto:support@prosoft-technology.com)

**Copyright © 2010 ProSoft Technology, Inc., all rights reserved.**

MVI56-MNETR User Manual

September 24, 2010

ProSoft Technology<sup>®</sup>, ProLinX<sup>®</sup>, inRAX<sup>®</sup>, ProTalk<sup>®</sup>, and RadioLinX<sup>®</sup> are Registered Trademarks of ProSoft Technology, Inc. All other brand or product names are or may be trademarks of, and are used to identify products and services of, their respective owners.

## **ProSoft Technology<sup>®</sup> 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](http://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

## 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 NON-HAZARDOUS.
- D** THIS DEVICE SHALL BE POWERED BY CLASS 2 OUTPUTS ONLY.

## MVI (Multi Vendor Interface) Modules

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

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

## Warnings

### North America Warnings

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, Division 2.
- B** Warning - Explosion Hazard - When in hazardous locations, turn off power before replacing or rewiring modules.
- C** Warning - Explosion Hazard - Do not disconnect equipment unless power has been switched off or the area is known to be non-hazardous.

*Avertissement - Risque d'explosion - Avant de déconnecter l'équipement, couper le courant ou s'assurer que l'emplacement est désigné non dangereux.*

- D** Suitable for use in Class I, Division 2 Groups A, B, C and D Hazardous Locations or Non-Hazardous Locations.

### ATEX Warnings and Conditions of Safe Usage

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

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

## Battery Life Advisory

The MVI46, MVI56, MVI56E, MVI69, and MVI71 modules use a rechargeable Lithium Vanadium Pentoxide battery to backup the real-time clock and CMOS. The battery should last for the life of the module. The module must be powered for approximately twenty hours before the battery becomes fully charged. After it is fully charged, the battery provides backup power for the CMOS setup and the real-time clock for approximately 21 days. When the battery is fully discharged, the module will revert to the default BIOS and clock settings.

**Note:** The battery is not user replaceable.

## Markings

### Electrical Ratings

- Backplane Current Load: 800 mA @ 5.1 Vdc; 3 mA @ 24 Vdc
- Operating Temperature: 0°C to 60°C (32°F to 140°F)
- Storage Temperature: -40°C to 85°C (-40°F to 185°F)
- Shock: 30 g, operational; 50 g, non-operational; Vibration: 5 g from 10 Hz to 150 Hz
- Relative Humidity: 5% to 95% (without condensation)
- All phase conductor sizes must be at least 1.3 mm(squared) and all earth ground conductors must be at least 4mm(squared).

### Label Markings

#### ATEX

II 3 G

EEx nA IIC T6

0°C <= Ta <= 60°C

#### cULus

E183151

Class I Div 2 Groups A,B,C,D

T6

-30°C <= Ta <= 60°C

### Agency Approvals and Certifications

Agency	Applicable Standard
RoHS	
CE	EMC-EN61326-1:2006; EN61000-6-4:2007
ATEX	EN60079-15:2003
cULus	UL508; UL1604; CSA 22.2 No. 142 & 213
CB Safety	CA/10533/CSA IEC 61010-1 Ed.2; CB 243333-2056722 (2090408)
GOST-R	EN 61010
CSA	EN 61010

RoHS



243333



ME06



E18315

# Contents

Your Feedback Please .....	2
How to Contact Us .....	2
ProSoft Technology® Product Documentation .....	2
Important Installation Instructions .....	3
MVI (Multi Vendor Interface) Modules .....	3
Warnings .....	3
Battery Life Advisory .....	3
Markings.....	4

## **Guide to the MVI56-MNETR User Manual 9**

### **1 Start Here 11**

1.1	System Requirements .....	12
1.2	Package Contents .....	13
1.3	Installing ProSoft Configuration Builder Software .....	14
1.4	Setting Jumpers .....	15
1.5	Installing the Module in the Rack .....	16
1.6	Creating a New RSLogix 5000 Project.....	18
1.6.1	Create the Remote Network.....	19
1.6.2	Import Add-On Instruction .....	26
1.6.3	Connecting Your PC to the ControlLogix Processor.....	29
1.6.4	Adding Multiple Modules (Optional) .....	30
1.6.5	Adjusting the Input and Output Array Sizes (Optional) .....	36
1.6.6	Downloading the Sample Program to the Processor .....	37

### **2 Configuring the MVI56-MNETR Module 39**

2.1	Connecting your PC to the Module .....	40
2.2	Using ProSoft Configuration Builder .....	41
2.2.1	Setting Up the Project .....	41
2.2.2	Renaming PCB Objects .....	43
2.2.3	Module.....	45
2.2.4	MNET Client x .....	47
2.2.5	MNET Client x Commands.....	49
2.2.6	MNET Servers.....	56
2.2.7	Static ARP Table .....	58
2.2.8	Ethernet Configuration .....	60
2.3	Downloading the Project to the Module Using a Serial COM port .....	61

### **3 Ladder Logic 63**

3.1	MNETRMODULEDEF .....	64
3.1.1	MNETRDATA .....	64
3.1.2	MNETRSTATUS .....	65
3.1.3	MNETRCONTROL .....	66
3.1.4	MNETRUTIL.....	70
3.2	Modbus Message Data .....	71

3.3	Using the Sample Program - RSLogix 5000 Version 15 and earlier .....	72
3.3.1	Adding the Module to an Existing Project .....	72
<b>4</b>	<b>Diagnostics and Troubleshooting</b>	<b>75</b>
4.1	LED Indicators .....	76
4.1.1	Client Configuration Error Word .....	77
4.1.2	Ethernet LED Indicators .....	77
4.1.3	Clearing a Fault Condition .....	78
4.1.4	Troubleshooting .....	79
4.2	Using ProSoft Configuration Builder (PCB) for Diagnostics .....	80
4.2.1	Using the Diagnostic Window in ProSoft Configuration Builder .....	80
4.2.2	Navigation .....	82
4.2.3	Main Menu .....	83
4.2.4	Modbus Database View Menu .....	86
4.2.5	Command List Menu .....	87
4.2.6	Master Command Error List Menu .....	88
4.2.7	Network Menu .....	89
4.3	Reading Status Data from the Module .....	91
<b>5</b>	<b>Reference</b>	<b>93</b>
5.1	Product Specifications .....	93
5.1.1	General Specifications .....	93
5.1.2	Hardware Specifications .....	94
5.1.3	Functional Specifications .....	95
5.2	Functional Overview .....	96
5.2.1	General Concepts .....	96
5.2.2	Data Flow between MVI56-MNETR Module and ControlLogix Processor .....	109
5.3	Cable Connections .....	115
5.3.1	Ethernet Connection .....	115
5.3.2	RS-232 Configuration/Debug Port .....	116
5.3.3	DB9 to RJ45 Adaptor (Cable 14) .....	118
5.4	Status Data Definition .....	119
5.5	Modbus Protocol Specification .....	120
5.5.1	Read Coil Status (Function Code 01) .....	120
5.5.2	Read Input Status (Function Code 02) .....	122
5.5.3	Read Holding Registers (Function Code 03) .....	123
5.5.4	Read Input Registers (Function Code 04) .....	124
5.5.5	Force Single Coil (Function Code 05) .....	125
5.5.6	Preset Single Register (Function Code 06) .....	126
5.5.7	Diagnostics (Function Code 08) .....	127
5.5.8	Force Multiple Coils (Function Code 15) .....	129
5.5.9	Preset Multiple Registers (Function Code 16) .....	130
5.5.10	Modbus Exception Responses .....	131
<b>6</b>	<b>Support, Service &amp; Warranty</b>	<b>135</b>
	Contacting Technical Support .....	135
6.1	Return Material Authorization (RMA) Policies and Conditions .....	137
6.1.1	Returning Any Product .....	137
6.1.2	Returning Units Under Warranty .....	138
6.1.3	Returning Units Out of Warranty .....	138

---

6.2	LIMITED WARRANTY .....	139
6.2.1	What Is Covered By This Warranty .....	139
6.2.2	What Is Not Covered By This Warranty .....	140
6.2.3	Disclaimer Regarding High Risk Activities .....	140
6.2.4	Intellectual Property Indemnity .....	141
6.2.5	Disclaimer of all Other Warranties .....	141
6.2.6	Limitation of Remedies ** .....	142
6.2.7	Time Limit for Bringing Suit .....	142
6.2.8	No Other Warranties .....	142
6.2.9	Allocation of Risks .....	142
6.2.10	Controlling Law and Severability .....	143

**Index**

**145**





## Guide to the MVI56-MNETR User Manual

Function		Section to Read	Details
Introduction (Must Do)	→	Start Here (page 11)	This section introduces the customer to the module. Included are: package contents, system requirements, hardware installation, and basic configuration.
Diagnostic and Troubleshooting	→	Diagnostics and Troubleshooting (page 75)	This section describes Diagnostic and Troubleshooting procedures.
Reference  Product Specifications  Functional Overview	→	Reference (page 93)  Product Specifications (page 93)  Functional Overview (page 96, page 87)	These sections contain general references associated with this product, Specifications, and the Functional Overview.
Support, Service, and Warranty  Index	→	Support, Service and Warranty (page 135)  Index	This section contains Support, Service and Warranty information.  Index of chapters.



# 1 Start Here

## *In This Chapter*

❖ System Requirements .....	12
❖ Package Contents .....	13
❖ Installing ProSoft Configuration Builder Software.....	14
❖ Setting Jumpers .....	15
❖ Installing the Module in the Rack.....	16
❖ Creating a New RSLogix 5000 Project .....	18

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

- **Rockwell Automation® RSLogix™ software:** launch the program, configure ladder logic, and transfer the ladder logic to the processor
- **Microsoft Windows:** install and launch programs, execute menu commands, navigate dialog boxes, and enter data
- **Hardware installation and wiring:** install the module, and safely connect Modbus TCP/IP and ControlLogix devices to a power source and to the MVI56-MNETR module's application port(s)

## 1.1 System Requirements

The MVI56-MNETR module requires the following minimum hardware and software components:

- Rockwell Automation ControlLogix™ processor, with compatible power supply and one free slot in the rack, for the MVI56-MNETR module. The module requires 800 mA of available power.
- Rockwell Automation RSLogix 5000 programming software version 2.51 or higher
- Rockwell Automation RSLinx communication software
- Pentium® II 450 MHz minimum. Pentium III 733 MHz (or better) recommended
- Supported operating systems:
  - Microsoft Windows XP Professional with Service Pack 1 or 2
  - Microsoft Windows 2000 Professional with Service Pack 1, 2, or 3
  - Microsoft Windows Server 2003
- 128 Mbytes of RAM minimum, 256 Mbytes of RAM recommended
- 100 Mbytes of free hard disk space (or more based on application requirements)
- 256-color VGA graphics adapter, 800 x 600 minimum resolution (True Color 1024 × 768 recommended)
- CD-ROM drive
- ProSoft Configuration Builder, HyperTerminal or other terminal emulator program.

**Note:** You can install the module in a local or remote rack. For remote rack installation, the module requires EtherNet/IP or ControlNet communication with the processor.

## 1.2 Package Contents

The following components are included with your MVI56-MNETR module, and are all required for installation and configuration.

**Important:** Before beginning the installation, please verify that all of the following items are present.

Qty.	Part Name	Part Number	Part Description
1	MVI56-MNETR Module	MVI56-MNETR	Modbus TCP/IP Interface Module with Reduced Data Block
1	Cable	Cable #15 - RS232 Null Modem	For RS232 between a Personal Computer (PC) and the CFG port of the module
1	Cable	Cable #14 - RJ45 to DB9 Male Adapter	For connecting the module's port to Cable #15 for RS-232 connections
1	inRAX Solutions CD		Contains sample programs, utilities and documentation for the MVI56-MNETR module.

If any of these components are missing, please contact ProSoft Technology Support for replacement parts.

### 1.3 Installing ProSoft Configuration Builder Software

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

#### **Installing ProSoft Configuration Builder from the ProSoft website**

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

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

#### **Installing ProSoft Configuration Builder from the Product CD-ROM**

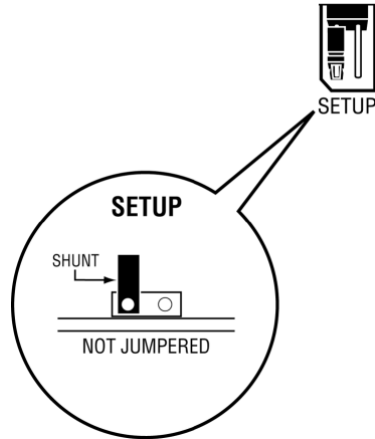
- 1 Insert the *ProSoft Solutions Product CD-ROM* into the CD-ROM drive of your PC. Wait for the startup screen to appear.
- 2 On the startup screen, click **PRODUCT DOCUMENTATION**. This action opens a *Windows Explorer* file tree window.
- 3 Click to open the **UTILITIES** folder. This folder contains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

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

## 1.4 Setting Jumpers

The Setup Jumper acts as "write protection" for the module's flash memory. In "write protected" mode, the Setup pins are not connected, and the module's firmware cannot be overwritten. Do not jumper the Setup pins together unless you are directed to do so by ProSoft Technical Support.

The following illustration shows the MVI56-MNETR jumper configuration.



**Note:** If you are installing the module in a remote rack, you may prefer to leave the Setup pins jumpered. That way, you can update the module's firmware without requiring physical access to the module.

## 1.5 Installing the Module in the Rack

If you have not already installed and configured your ControlLogix processor and power supply, please do so before installing the MVI56-MNETR module. Refer to your Rockwell Automation product documentation for installation instructions.

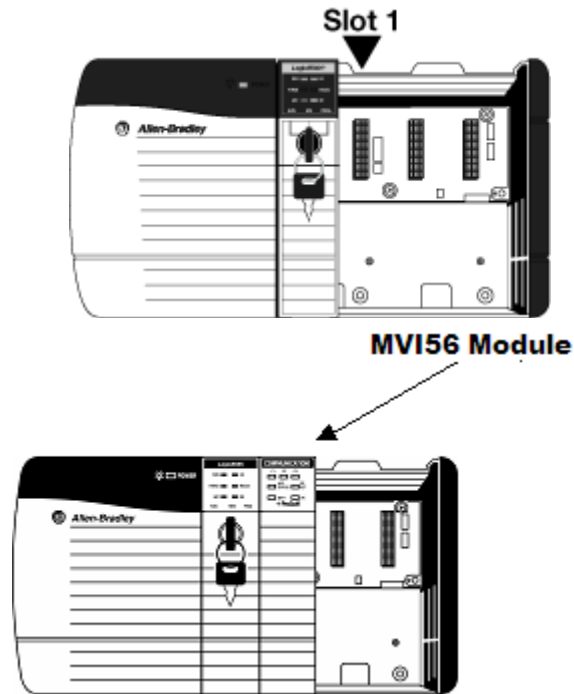
**Warning:** You must follow all safety instructions when installing this or any other electronic devices. Failure to follow safety procedures could result in damage to hardware or data, or even serious injury or death to personnel. Refer to the documentation for each device you plan to connect to verify that suitable safety procedures are in place before installing or servicing the device.

After you have checked the placement of the jumpers, insert MVI56-MNETR into the ControlLogix chassis. Use the same technique recommended by Rockwell Automation to remove and install ControlLogix modules.

**Warning:** When you insert or remove the module while backplane power is on, an electrical arc can occur. This could cause an explosion in hazardous location installations. Verify that power is removed or the area is non-hazardous before proceeding. Repeated electrical arcing causes excessive wear to contacts on both the module and its mating connector. Worn contacts may create electrical resistance that can affect module operation.

- 1 Turn power OFF.
- 2 Align the module with the top and bottom guides, and slide it into the rack until the module is firmly against the backplane connector.



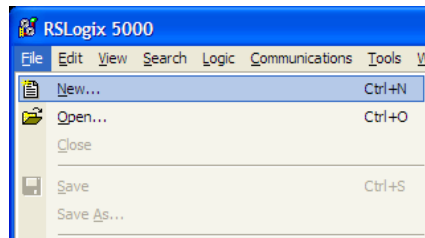


- 3 With a firm but steady push, snap the module into place.
- 4 Check that the holding clips on the top and bottom of the module are securely in the locking holes of the rack.
- 5 Make a note of the slot location. You must identify the slot in which the module is installed in order for the sample program to work correctly. Slot numbers are identified on the green circuit board (backplane) of the ControlLogix rack.
- 6 Turn power ON.

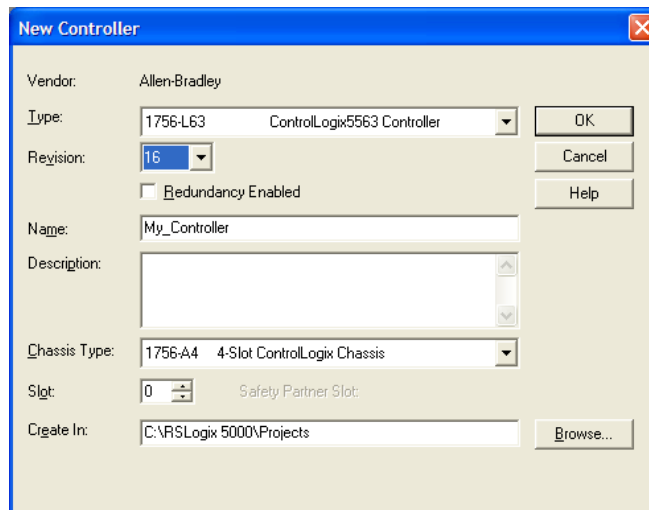
**Note:** If you insert the module improperly, the system may stop working, or may behave unpredictably.

## 1.6 Creating a New RSLogix 5000 Project

- 1 Open the **FILE** menu, and then choose **NEW**.



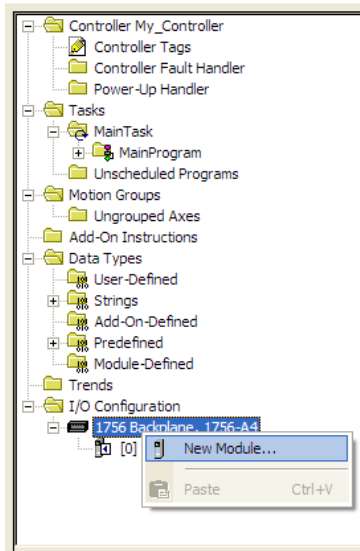
- 2 Select your ControlLogix controller model.
- 3 Select **REVISION 16**.
- 4 Enter a name for your controller, such as *My\_Controller*.
- 5 Select your ControlLogix chassis type.
- 6 Select **SLOT 0** for the controller.



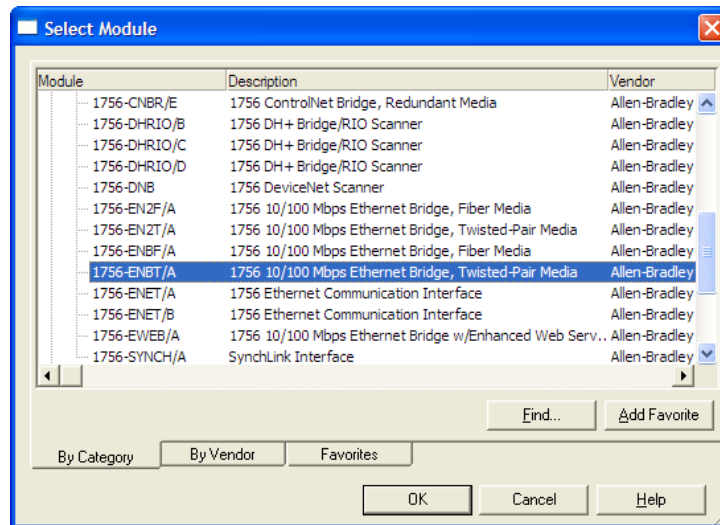
**Note:** If you are installing the MVI56-MNETR module in a remote rack, follow these next few steps. If you are installing the module in a local rack, follow the steps in Create the Module - Local Rack (page 23).

### 1.6.1 Create the Remote Network

- 1 Right-click **I/O CONFIGURATION** and choose **NEW MODULE...**

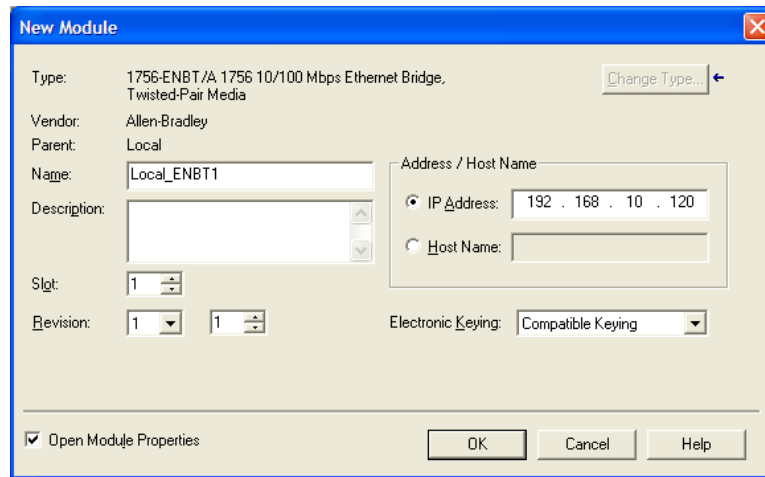


- 2 Expand the **COMMUNICATIONS** module selections and then select the Ethernet Bridge module that matches your hardware. This example uses a 1756-ENBT/A module.

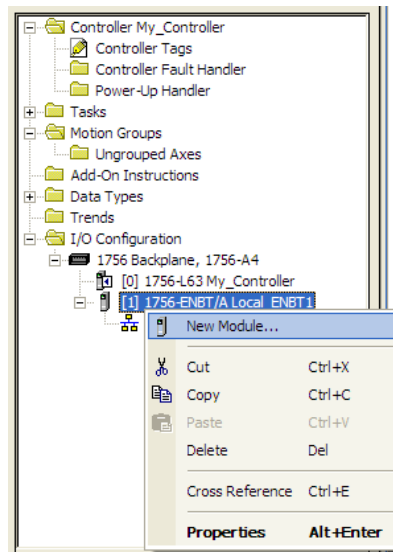


**Note:** If you are prompted to "Select Major Revision", choose the lower of the available revision numbers.

- 3 Name the ENBT/A module, then set the IP Address and slot location in the local rack with the ControlLogix processor.



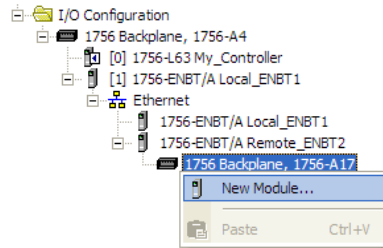
- 4 Click **OK**.
- 5 Next, select the **1756-ENBT** module that you just created in the Controller Organization pane and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.



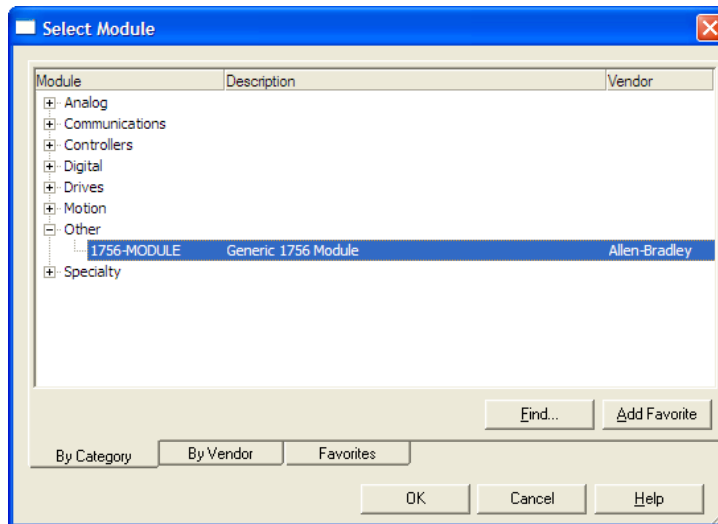
- 6 Repeat steps 2 and 3 to add the second EtherNet/IP module to the remote rack.

Create the Module - Remote Rack

- 1 Next, select the remote **1756 BACKPLANE** node in the Controller Organization pane underneath the remote rack EtherNet/IP module you just created and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.



This action opens the **SELECT MODULE** dialog box.

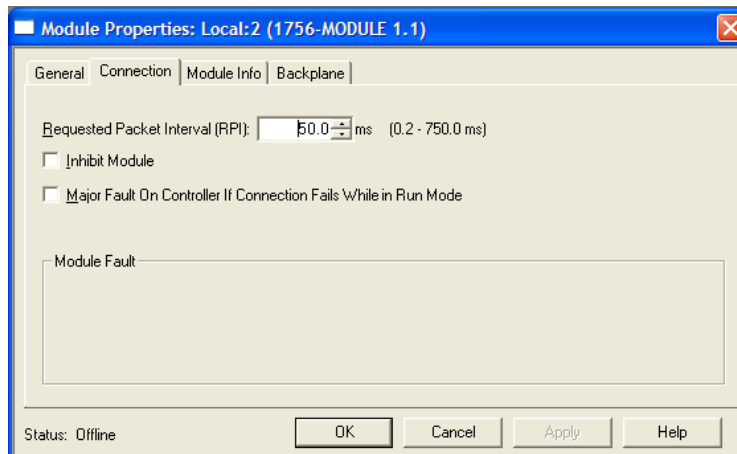


- 2 Select the **1756-MODULE (GENERIC 1756 MODULE)** from the list and click **OK**. This action opens the **NEW MODULE** dialog box.

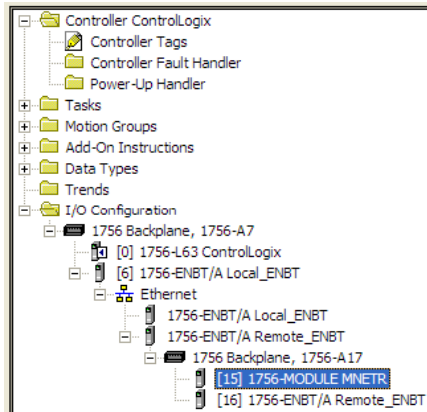
**3** Set the Module Properties values as follows:

Parameter	Value
Name	Enter a module identification string. The recommended value is MNET.
Description	Enter a description for the module. Example: Modbus TCP/IP Interface Module with Reduced Data Block.
Comm Format	Select <b>DATA-INT (Very Important)</b>
Slot	Enter the slot number in the rack where the MVI56-MNETR module will be installed.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

**4** On the **CONNECTION** tab, set the **RPI** value for your project. Fifty (50) milliseconds is usually a good starting value.



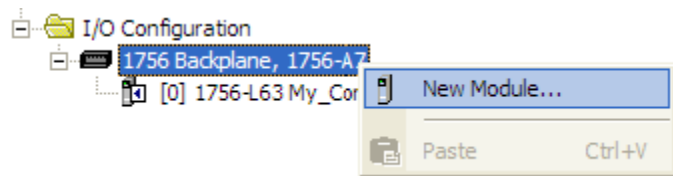
The **MVI56-MNETR** module is now visible in the **I/O CONFIGURATION** section



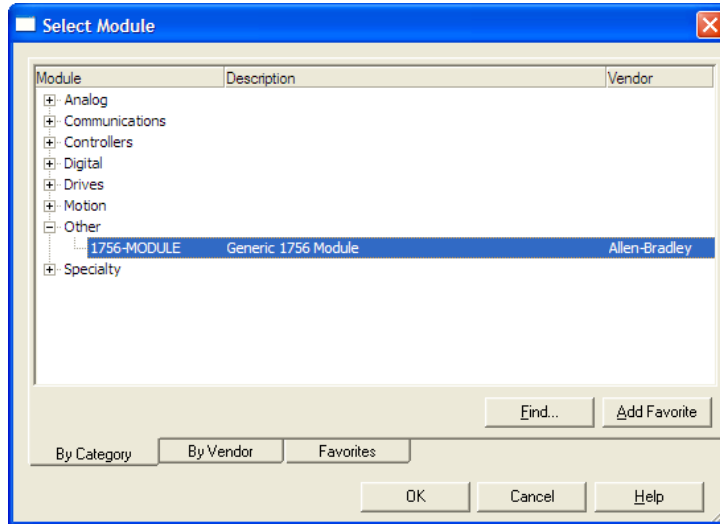
**Note:** If you are installing the MVI56-MNETR module in a local rack, follow these next few steps. If you are installing the module in a remote rack, follow the steps in Create the Module - Remote Rack (page 19).

Create the Module - Local Rack

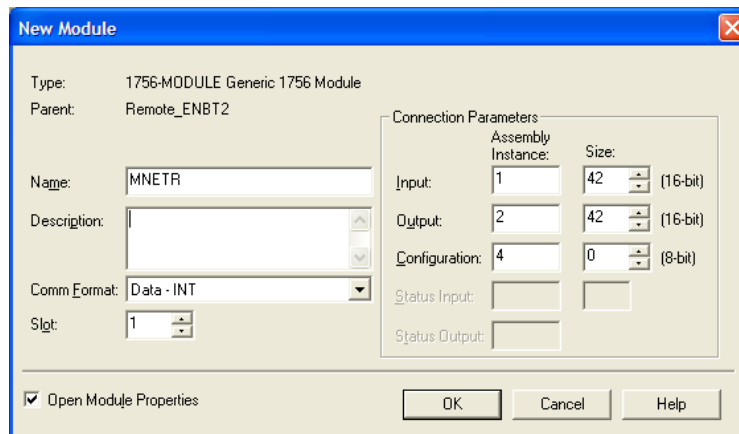
- 1 Add the MVI56-MNETR module to the project.  
In the **CONTROLLER ORGANIZATION** window, select **I/O CONFIGURATION** and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE...**



This action opens the **SELECT MODULE** dialog box.



- 2 Select the **1756-MODULE (GENERIC 1756 MODULE)** from the list and click **OK**. This action opens the **NEW MODULE** dialog box.

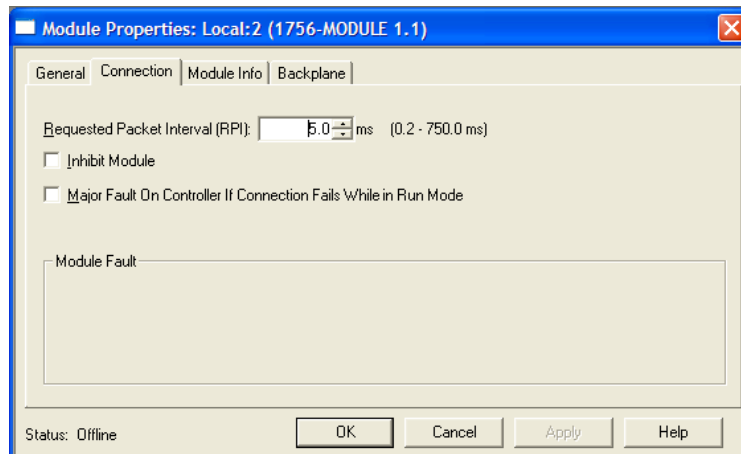




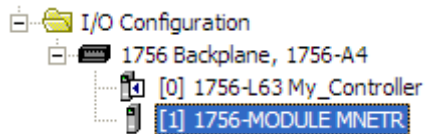
**3** Set the Module Properties values as follows:

Parameter	Value
Name	Enter a module identification string. The recommended value is MNET.
Description	Enter a description for the module. Example: Modbus TCP/IP Interface Module with Reduced Data Block.
Comm Format	Select <b>DATA-INT (Very Important)</b>
Slot	Enter the slot number in the rack where the MVI56-MNETR module is to be installed.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

**4** On the **CONNECTION** tab, set the **RPI** value for your project. Five (5) milliseconds is usually a good starting value. Click **OK** to confirm.

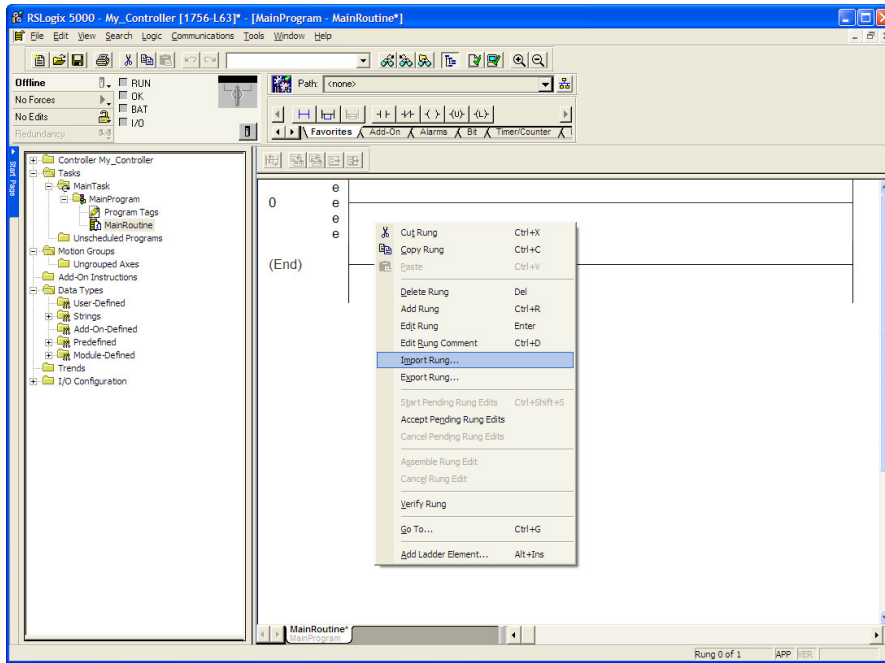


The **MVI56-MNETR** module is now visible in the **I/O CONFIGURATION** section

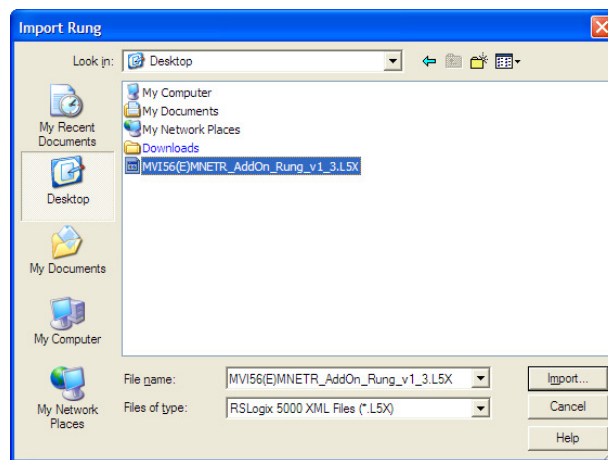


### 1.6.2 Import Add-On Instruction

- 1 In the **CONTROLLER ORGANIZATION** window, expand the **TASKS** folder and subfolder until you reach the **MAINPROGRAM** folder.
- 2 In the **MAINPROGRAM** folder, double-click to open the **MAINROUTINE** ladder.
- 3 Select an empty rung in the new routine, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT RUNG...**

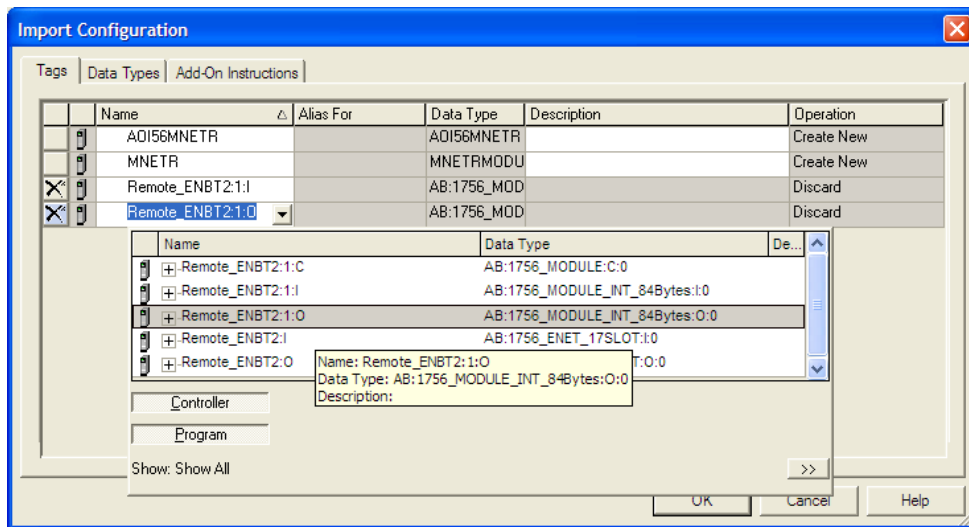
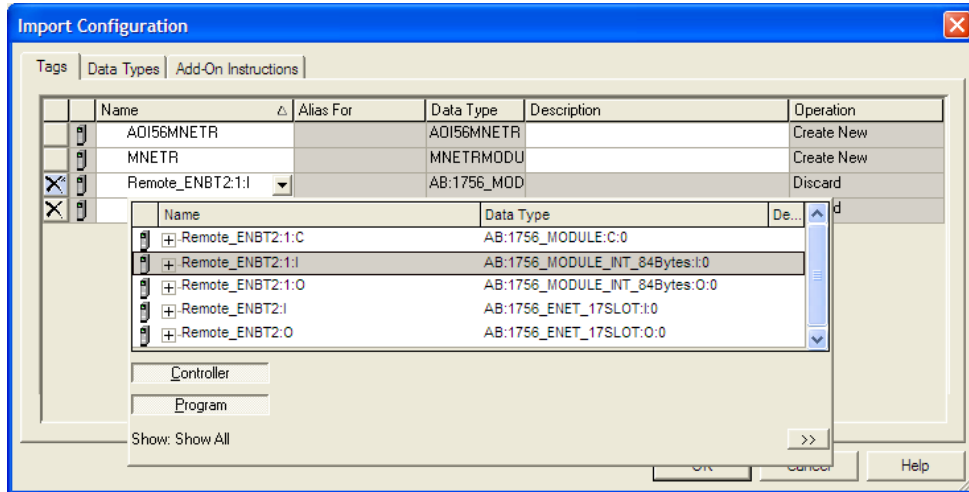


- 4 Navigate to the location on your PC where you saved the Add-On Instruction (for example, "My Documents" or "Desktop"). Select the **MVI56MNET\_ADDON\_RUNG\_v1\_3.L5X** file

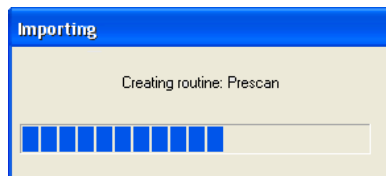


This action opens the **IMPORT CONFIGURATION** dialog box, showing the controller tags that will be created.

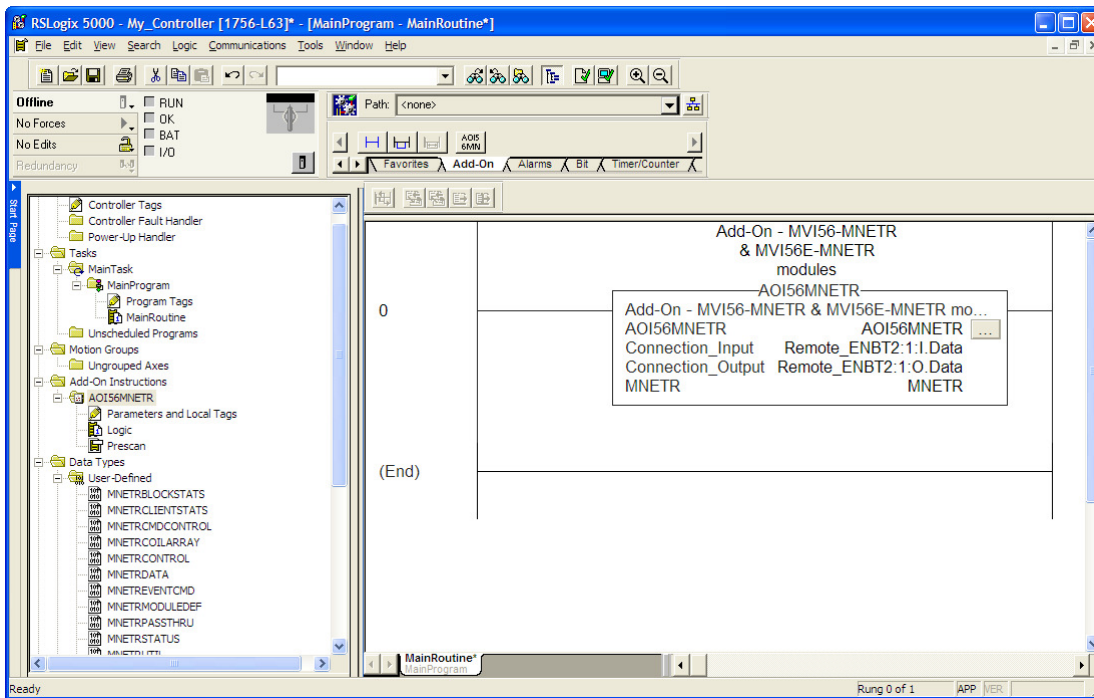
- If you are installing the module in a Remote Rack, open the dropdown menus for the Input and Output tags, and select the MNET module in the remote rack.



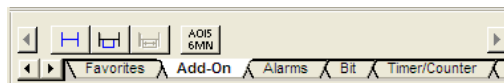
- 5 Click **OK** to confirm the import. RSLogix will indicate that the import is in progress:



When the import is complete, you will see the new Add-On Instruction rung in the ladder.



The procedure has also imported new User Defined Data Types, data objects and the Add-On instruction for your project.

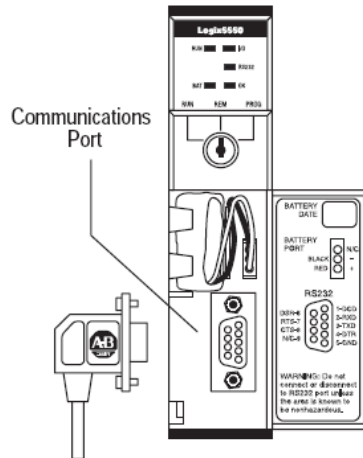


- 6 Save the application and then download the sample ladder logic into the processor.

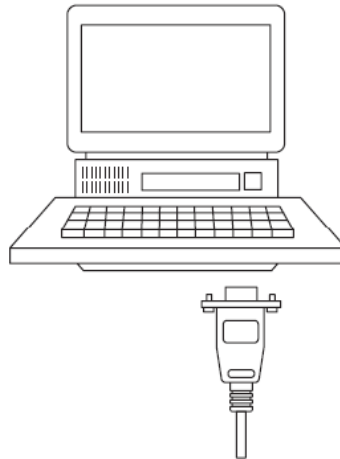
### 1.6.3 Connecting Your PC to the ControlLogix Processor

There are several ways to establish communication between your PC and the ControlLogix processor. The following steps show how to establish communication through the serial interface. It is not mandatory that you use the processor's serial interface. You may access the processor through whatever network interface is available on your system. Refer to your Rockwell Automation documentation for information on other connection methods.

- 1 Connect the right-angle connector end of the cable to your controller at the communications port.



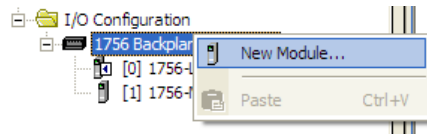
- 2 Connect the straight connector end of the cable to the serial port on your computer.



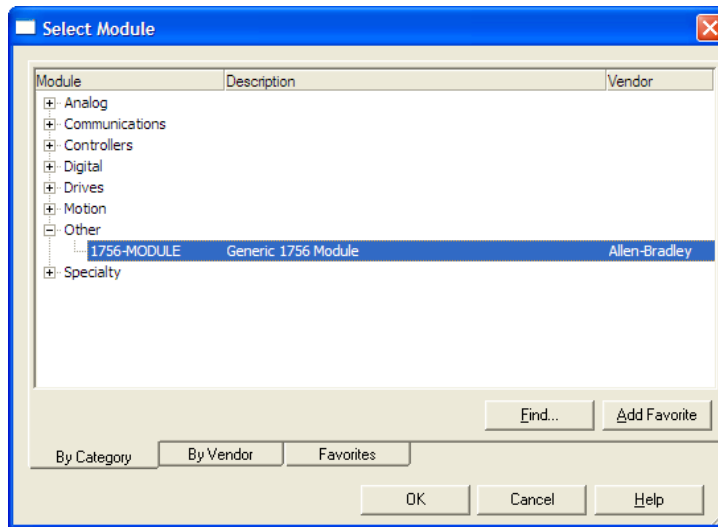
### 1.6.4 Adding Multiple Modules (Optional)

**Important:** If your application requires more than one MVI56-MNETR module into the same project, follow the steps below.

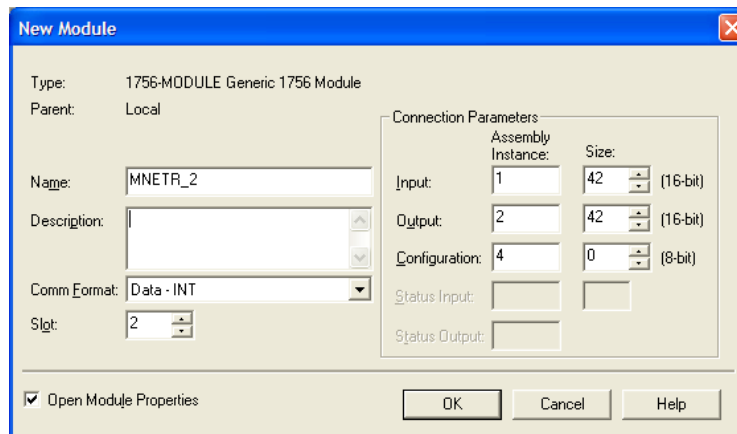
- 1 In the *I/O Configuration* folder, click the right mouse button to open a shortcut menu, and then choose **NEW MODULE**.



- 2 Select **1756-MODULE**



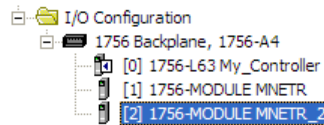
This action opens the *New Module* dialog box.



**3** Fill in the module properties as follows:

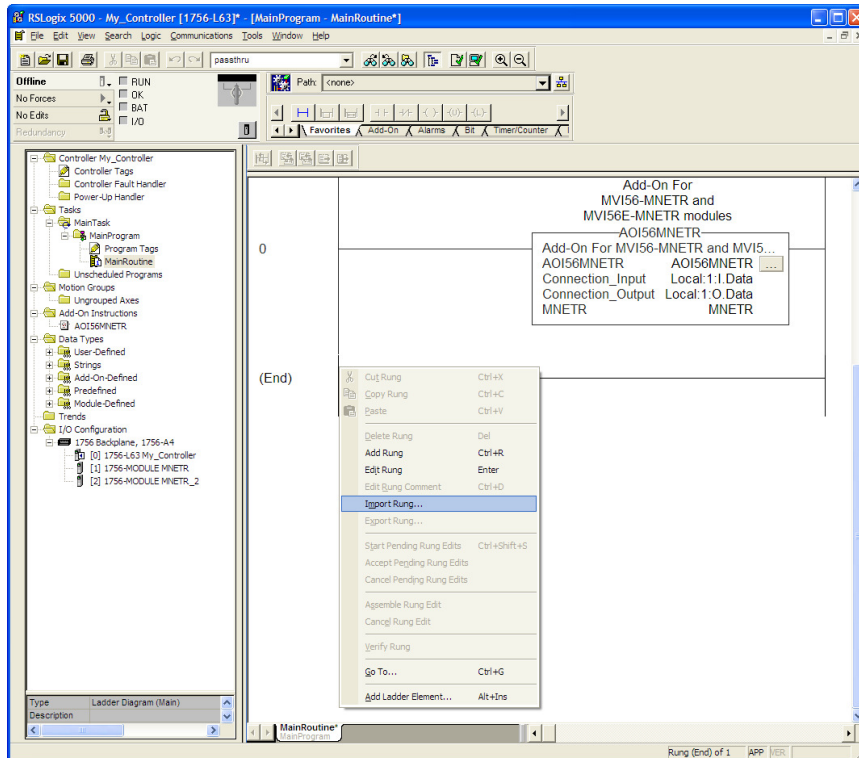
Parameter	Value
Name	Enter a module identification string. Example: MNET_2
Description	Enter a description for the module. Example: Modbus TCP/IP Interface Module with Reduced Data Block
Comm Format	Select DATA-INT
Slot	Enter the slot number in the rack where the MVI56-MNETR module is located.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

**4** Click **OK** to confirm. The new module is now visible:

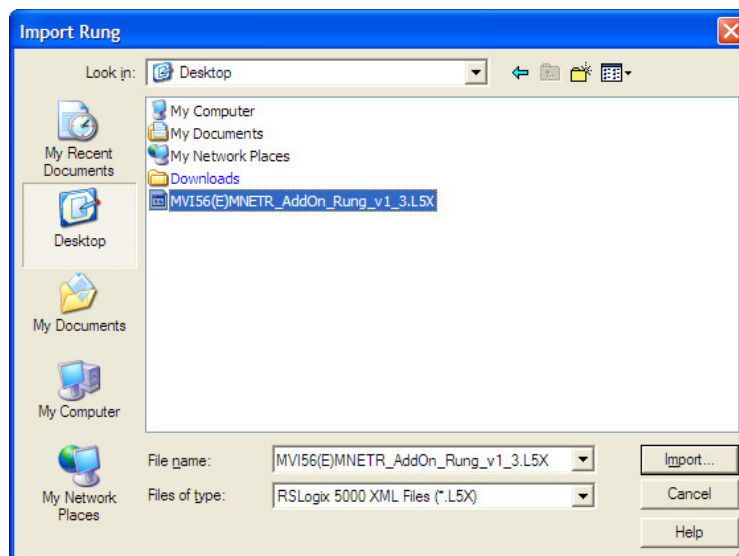


- 5** Expand the *Tasks* folder, and then expand the *MainTask* folder.
- 6** On the *MainProgram* folder, click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW ROUTINE**.
- 7** In the *New Routine* dialog box, enter the name and description of your routine, and then click **OK**.

- 8 Select an empty rung in the new routine, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **"IMPORT RUNG..."**.

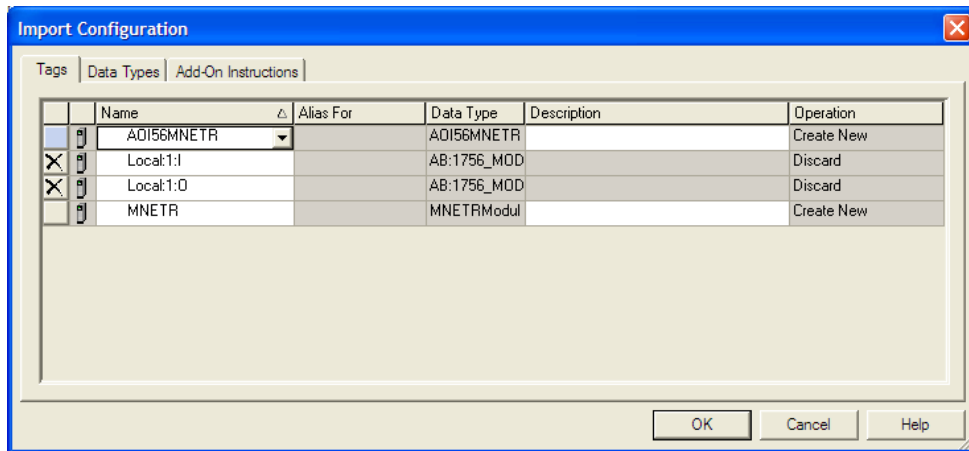


- 9 Select the file MVI56(E)MNET\_AddOn\_Rung\_<Version#>.L5X

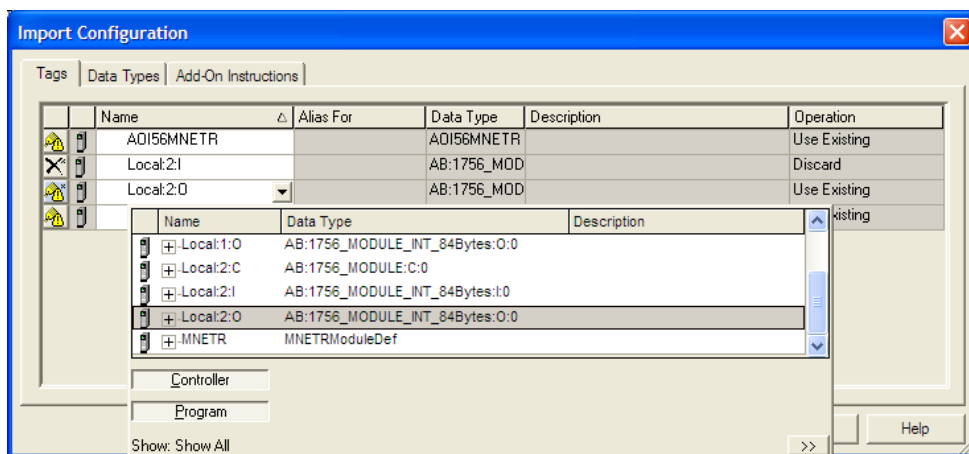
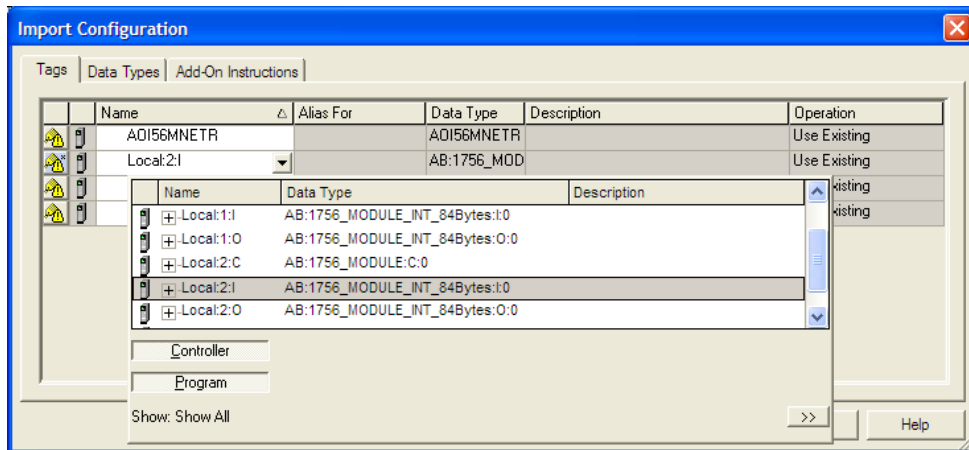




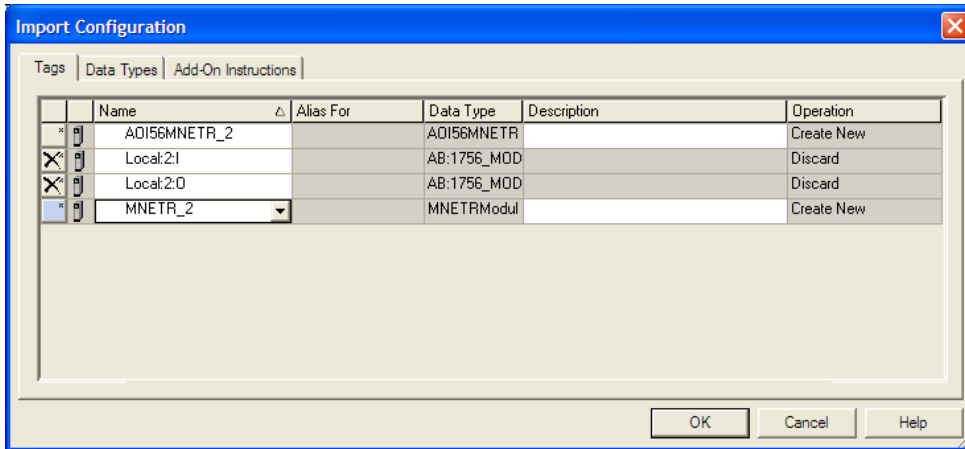
This action opens the **IMPORT CONFIGURATION** dialog box, showing the controller tags that will be created.



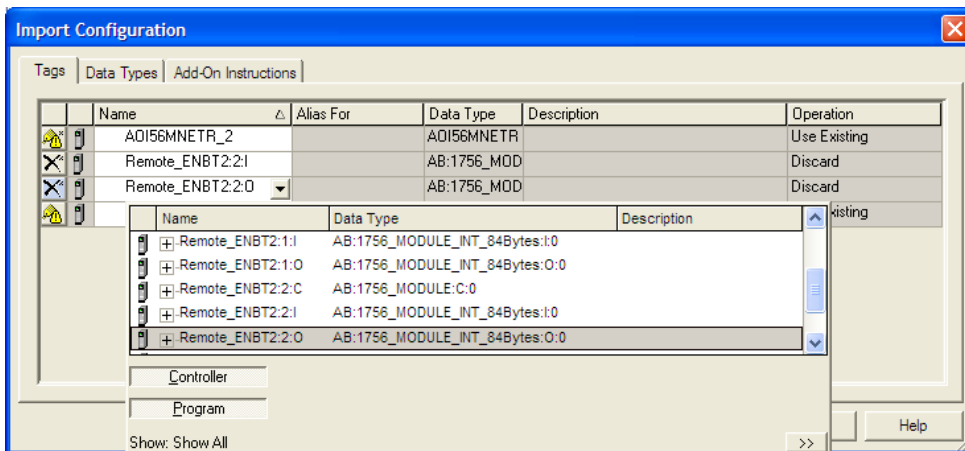
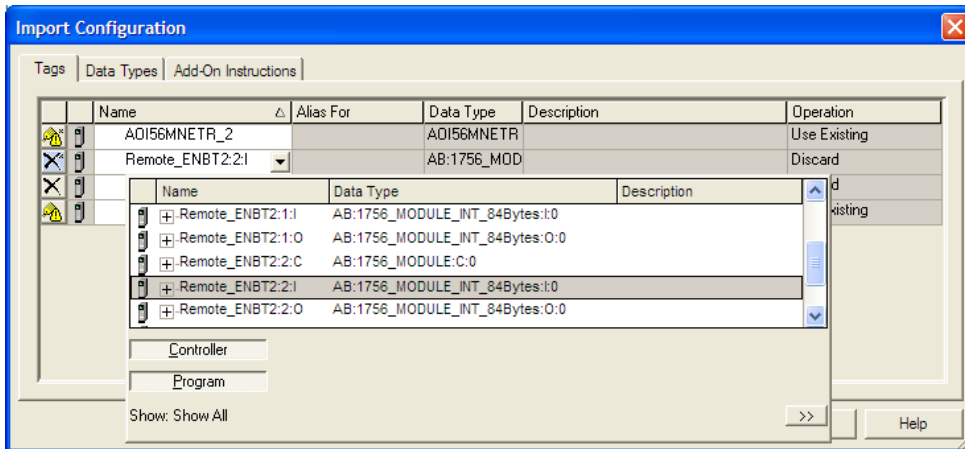
**10** Associate the I/O connection variables to the correct module. The default values are Local:1:I and Local:1:O so these require change.



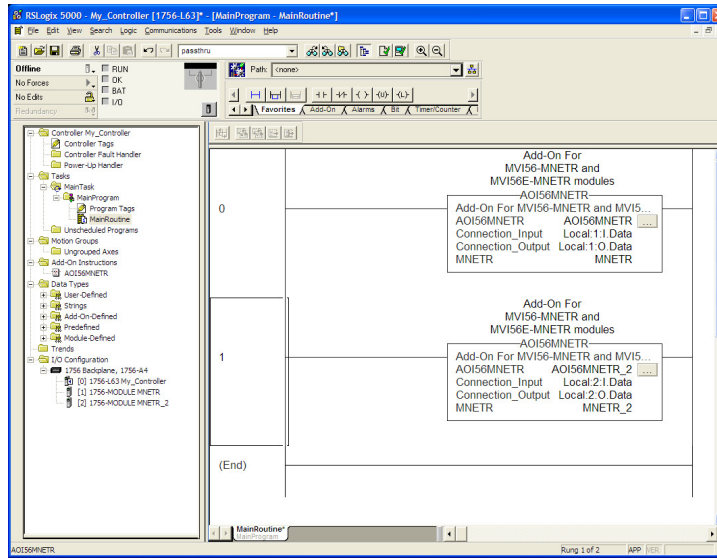
Change the default tags *MNETR* and *AOI56MNETR* to avoid conflict with existing tags. This procedure will append the string "\_2" as follows:



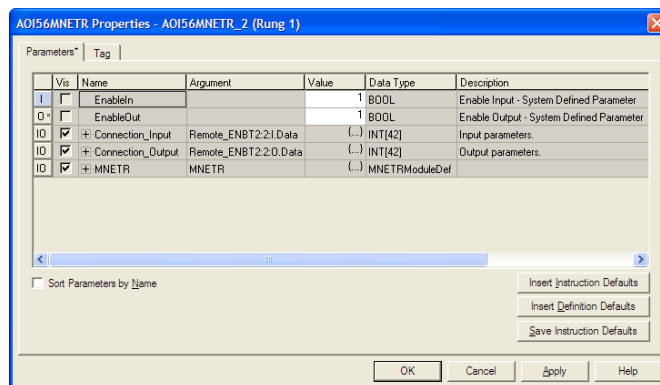
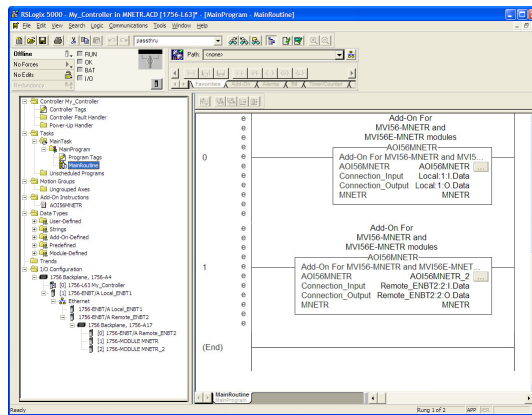
Or, in a Remote Rack application...



11 Click **OK** to confirm.



Or, in a Remote Rack application...



### 1.6.5 Adjusting the Input and Output Array Sizes (Optional)

The module internal database is divided into two user-configurable areas:

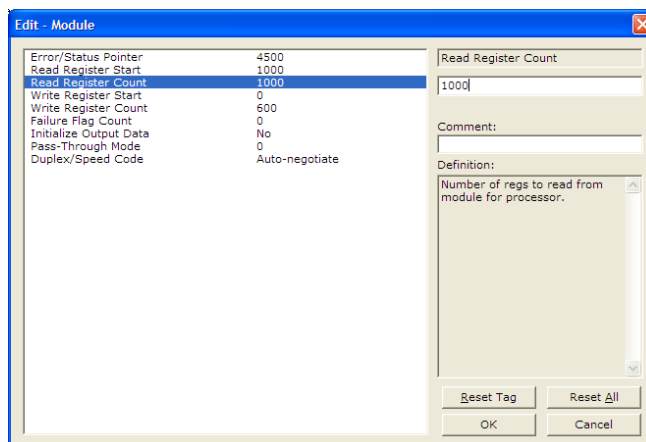
- Read Data
- Write Data

The Read Data area is moved from the module to the processor, while the Write Data area is moved from the processor to the module.

The MVI56-MNETR Add-On Instruction rung is configured for 600 registers of Read Data and 600 registers of Write Data, which is sufficient for most applications. However, you can configure the sizes of these data areas to meet the needs of your application.

- 1 In *ProSoft Configuration Builder*, expand the *Module* icon in the tree view and double-click **MODULE** to open an *Edit* window. Change the **READ REGISTER COUNT** to contain the number of words for your Read Data area.

**Important:** Because the module pages data in blocks of 200 registers at a time, you must configure your user data in multiples of 200 registers.



- 2 To modify the *WriteData* array, follow the above steps, substituting *WriteData* for *ReadData*. Also, make sure that the *ReadData* and *WriteData* arrays do not overlap in the module memory. For example, if your application requires 2000 words of *WriteData* starting at register 0, then your *Read Register Start* parameter must be set to a value of 2000 or greater in *ProSoft Configuration Builder*.

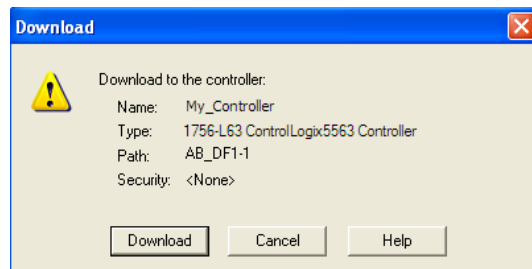
- 3 Save and download the configuration to the module. (page 61).

It is unnecessary to manually edit the *ReadData* and *WriteData* user-defined data types in the ladder logic, as these are automatically updated to match the changed array sizes in *ProSoft Configuration Builder*.

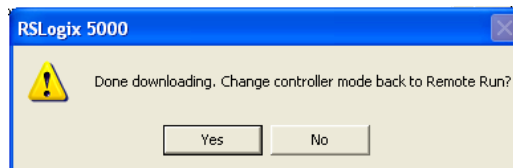
## 1.6.6 Downloading the Sample Program to the Processor

**Note:** The key switch on the front of the ControlLogix processor must be in the REM or PROG position.

- 1 If you are not already online with the processor, open the *Communications* menu, and then choose **DOWNLOAD**. RSLogix 5000 will establish communication with the processor. You do not have to download through the processor's serial port, as shown here. You may download through any available network connection.
- 2 When communication is established, RSLogix 5000 will open a confirmation dialog box. Click the **DOWNLOAD** button to transfer the sample program to the processor.



- 3 RSLogix 5000 will compile the program and transfer it to the processor. This process may take a few minutes.
- 4 When the download is complete, RSLogix 5000 will open another confirmation dialog box. If the key switch is in the REM position, click **OK** to switch the processor from PROGRAM mode to RUN mode.



**Note:** If you receive an error message during these steps, refer to your RSLogix documentation to interpret and correct the error.



## 2 Configuring the MVI56-MNETR Module

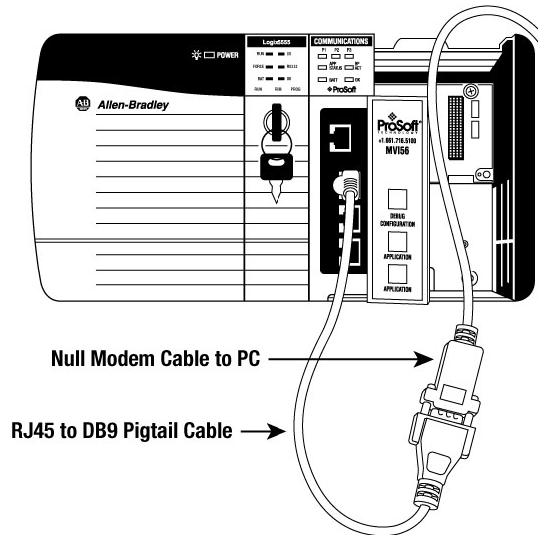
### *In This Chapter*

- ❖ Connecting your PC to the Module..... 40
- ❖ Using ProSoft Configuration Builder..... 41
- ❖ Downloading the Project to the Module Using a Serial COM port ..... 61

## 2.1 Connecting your PC to the Module

With the module securely mounted, connect your PC to the *Configuration/Debug* port using an RJ45-DB-9 Serial Adapter Cable and a Null Modem Cable.

- 1 Attach both cables as shown.
- 2 Insert the RJ45 cable connector into the Configuration/Debug port of the module.
- 3 Attach the other end to the serial port on your PC.



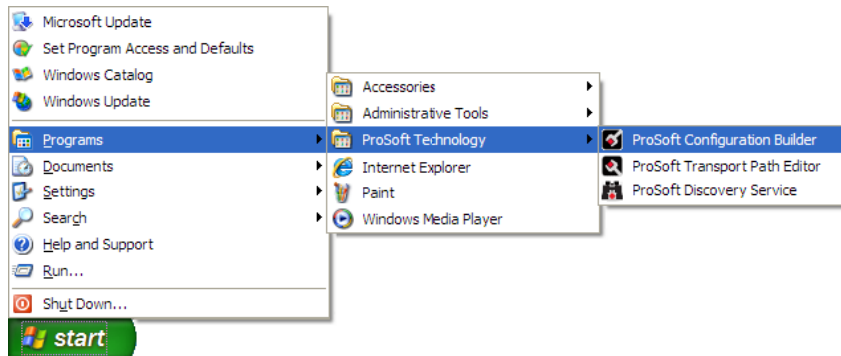


## 2.2 Using ProSoft Configuration Builder

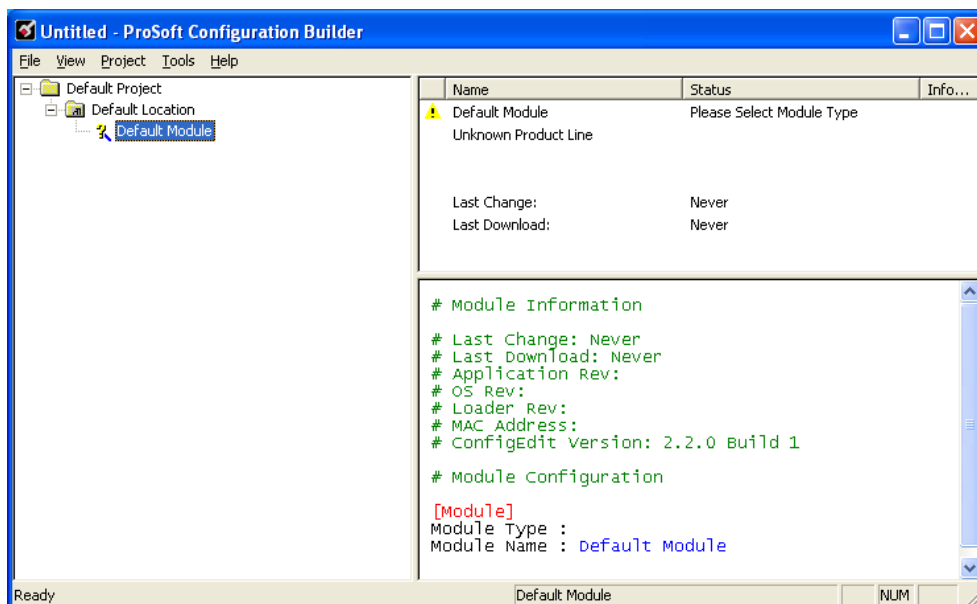
*ProSoft Configuration Builder (PCB)* provides a quick and easy way to manage module configuration files customized to meet your application needs. *PCB* is not only a powerful solution for new configuration files, but also allows you to import information from previously installed (known working) configurations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

### 2.2.1 Setting Up the Project

To begin, start **PROSOFT CONFIGURATION BUILDER (PCB)**.

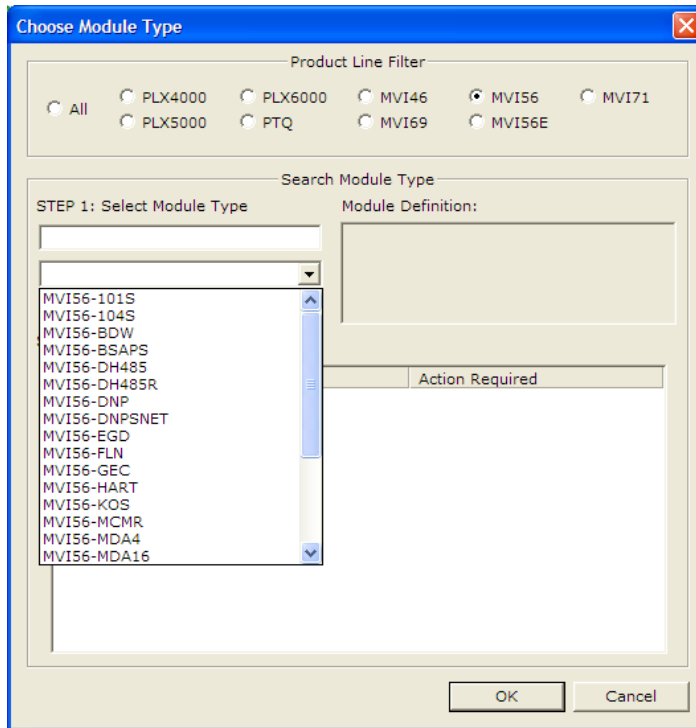


If you have used other Windows configuration tools before, you will find the screen layout familiar. *PCB*'s window consists of a tree view on the left, and an information pane and a configuration pane on the right side of the window. When you first start *PCB*, the tree view consists of folders for *Default Project* and *Default Location*, with a *Default Module* in the *Default Location* folder. The following illustration shows the *PCB* window with a new project.



### Adding the MVI56-MNETR module to the project

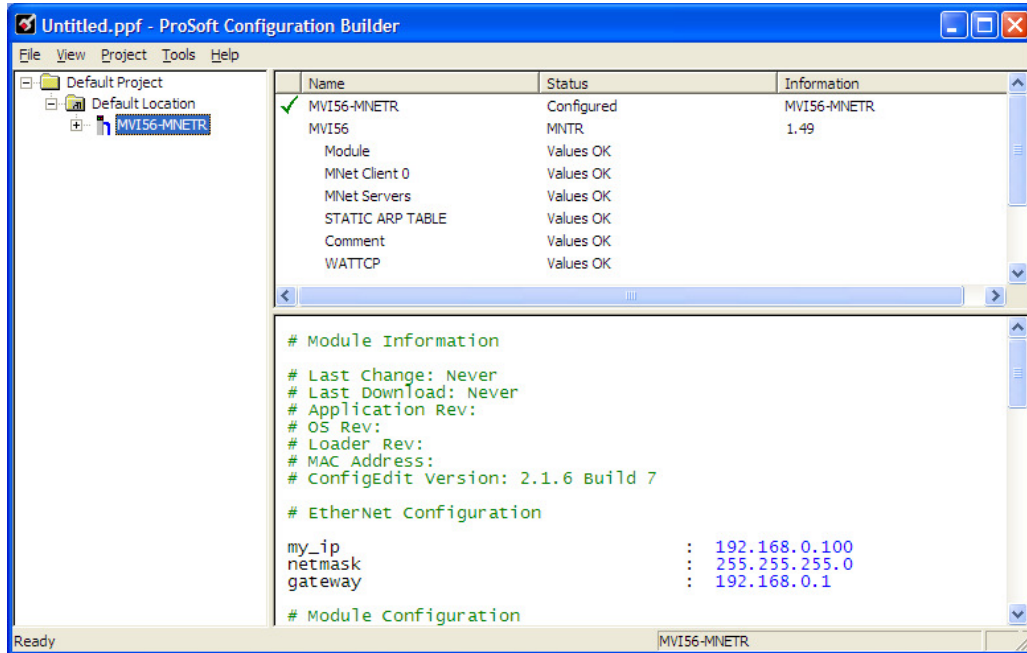
- 1 Use the mouse to select **DEFAULT MODULE** in the tree view, and then click the right mouse button to open a shortcut menu.
- 2 On the shortcut menu, choose **CHOOSE MODULE TYPE**. This action opens the *Choose Module Type* dialog box.



- 3 In the *Product Line Filter* area of the dialog box, select **MVI56**. In the *Select Module Type* dropdown list, select **MVI56-MNETR**, and then click **OK** to save your settings and return to the *ProSoft Configuration Builder* window.

## 2.2.2 Renaming PCB Objects



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





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

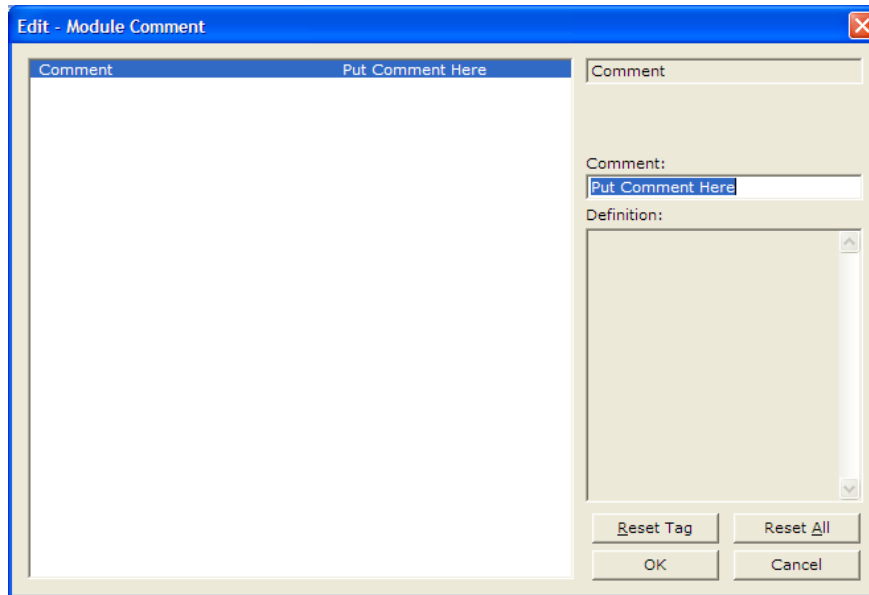
- 1 Select the object, and then click the right mouse button to open a shortcut menu. From the shortcut menu, choose **RENAME**.
- 2 Type the name to assign to the object.
- 3 Click away from the object to save the new name.

### Configuring Module Parameters

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

### Creating Optional Comment Entries

- 1 Click the **[+]** to the left of the  Comment icon to expand the module comments.
- 2 Double-click the  Module Comment icon. The *Edit - Module Comment* dialog box appears.



- 3 Enter your comment and click **OK** to save your changes.

### 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 In the *View Configuration* window, open the **FILE** menu, and choose **PRINT**. This action opens the *Print* dialog box.
- 4 In the *Print* dialog box, choose the printer to use from the drop-down list, select printing options, and then click **OK**.

### **2.2.3 Module**

This section of the configuration describes the database setup and module level parameters. This section provides the module with a unique name, identifies the method of failure for the communications for the module if the processor is not in RUN mode, and describes how to initialize the module upon startup.

#### Error/Status Pointer

**-1 to 4955**

Starting register location in virtual Modbus database for the error/status table. If a value of -1 is entered, the error/status data will not be placed in the database. All other valid values determine the starting location of the data. This data area includes the module version information and all error/status data.

#### Read Register Start

**0 to 4999**

The *Read Register Start* parameter specifies the start of the Read Data area in module memory. Data in this area will be transferred from the module to the processor.

**Note:** Total user database memory space is limited to the first 5000 registers of module memory, addresses 0 through 4999. Therefore, the practical limit for this parameter is 4999 minus the value entered for *Read Register Count*, so that the Read Data Area does not try to extend above address 4999. Read Data and Write Data Areas must be configured to occupy separate address ranges in module memory and should not be allowed to overlap.

#### Read Register Count

**0 to 5000**

The *Read Register Count* parameter specifies the size of the Read Data area of module memory and the number of registers to transfer from this area to the processor, up to a maximum of 5000 words.

**Note:** Total *Read Register Count* and *Write Register Count* cannot exceed 5000 total registers. Read Data and Write Data Areas must be configured to occupy separate address ranges in module memory and should not be allowed to overlap.

### Write Register Start

**0 to 4999**

The *Write Register Start* parameter specifies the start of the Write Data area in module memory. Data in this area will be transferred in from the processor.

**Note:** Total user database memory space is limited to the first 5000 registers of module memory, addresses 0 through 4999. Therefore, the practical limit for this parameter is 4999 minus the value entered for *Write Register Count*, so that the Write Data Area does not try to extend above address 4999. Read Data and Write Data Areas must be configured to occupy separate address ranges in module memory and should not be allowed to overlap.

### Write Register Count

**0 to 5000**

The *Write Register Count* parameter specifies the size of the Write Data area of module memory and the number of registers to transfer from the processor to this memory area, up to a maximum value of 5000 words.

**Note:** Total *Read Register Count* and *Write Register Count* cannot exceed 5000 total registers. Read Data and Write Data Areas must be configured to occupy separate address ranges in module memory and should not be allowed to overlap.

### Failure Flag Count

**0 through 65535**

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

### Initialize Output Data

**0 = No, 1 = Yes**

This parameter is used to determine if the output data for the module should be initialized with values from the processor. If the value is set to **0**, the output data will be initialized to 0. If the value is set to **1**, the data will be initialized with data from the processor. Use of this option requires associated ladder logic to pass the data from the processor to the module.

### Pass-Through Mode

**0, 1, 2 or 3**

This parameter specifies the pass-through mode for write messages received by the MNET and MBAP server ports.

- If the parameter is set to **0**, all write messages will be placed in the module's virtual database.
- If a value of **1** is entered, write messages received will be sent to the processor as unformatted messages.
- If a value of **2** is entered, write messages received will be sent to the processor as formatted messages.
- If a value of **3** is entered, write messages received will be sent to the processor with the bytes swapped in a formatted message.

### Duplex/Speed Code

**0, 1, 2, 3 or 4**

This parameter allows you to cause the module to use a specific duplex and speed setting.

- Value = **1**: Half duplex, 10 MB speed
- Value = **2**: Full duplex, 10 MB speed
- Value = **3**: Half duplex, 100 MB speed
- Value = **4**: Full duplex, 100 MB speed
- Value = **0**: Auto-negotiate

Auto-negotiate is the default value for backward compatibility. This feature is not implemented in older software revisions.

## **2.2.4 MNET Client x**

This section defines general configuration for the MNET Client (Master).

### Error/Status Pointer

**-1 to 4990**

Starting register location in virtual database for the error/status table for this Client. If a value of **-1** is entered, the error/status data will not be placed in the database. All other valid values determine the starting location of the data.

### Command Error Pointer

**-1 to 4999**

This parameter sets the address in the internal database where the Command Error List data will be placed so that it may be moved to the processor and placed into the *ReadData* array. Therefore, the value entered should be a module memory address in the Read Data area. If the value is set to **-1**, the Command Error List data will not be stored in the module's internal database and will not be transferred to the processor's *ReadData* array.

### Minimum Command Delay

**0** to **65535** milliseconds

This parameter specifies the number of milliseconds to wait between the initial issuances of a command. This parameter can be used to delay all commands sent to Servers to avoid "flooding" commands on the network. This parameter does not affect retries of a command as they will be issued when failure is recognized.

### Response Timeout

**0** to **65535** milliseconds

This is the time in milliseconds that a Client will wait before re-transmitting a command if no response is received from the addressed server. The value to use depends upon the type of communication network used, and the expected response time of the slowest device on the network.

### Retry Count

**0** to **10**

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

### Float Flag

**YES** or **NO**

This flag specifies how the Server driver will respond to Function Code 3, 6, and 16 commands (read and write Holding Registers) from a remote Client when it is moving 32-bit floating-point data.

If the remote Client expects to receive or will send one complete 32-bit floating-point value for each count of one (1), then set this parameter to **YES**. When set to **YES**, the Server driver will return values from two consecutive 16-bit internal memory registers (32 total bits) for each count in the read command, or receive 32-bits per count from the Client for write commands. Example: Count = **10**, Server driver will send 20 16-bit registers for 10 total 32-bit floating-point values. If, however, the remote Client sends a count of two (2) for each 32-bit floating-point value it expects to receive or send, or, if you do not plan to use floating-point data in your application, then set this parameter to **No**, which is the default setting.

You will also need to set the *Float Start* and *Float Offset* parameters to appropriate values whenever the *Float Flag* parameter is set to **YES**.

### Float Start

**F0** to **65535**

This parameter defines the first register of floating-point data. All requests with register values greater than or equal to this value will be considered floating-point data requests. This parameter is only used if the *Float Flag* is enabled. For example, if a value of 7000 is entered, all requests for registers 7000 and above will be considered as floating-point data.



Float Offset

**0 to 9999**

This parameter defines the start register for floating-point data in the internal database. This parameter is used only if the *Float Flag* is enabled. For example, if the *Float Offset* value is set to **3000** and the *Float Start* parameter is set to **7000**, data requests for register 7000 will use the internal Modbus register 3000.

ARP Timeout

**1 to 60**

This parameter specifies the number of seconds to wait for an ARP reply after a request is issued.

Command Error Delay

**0 to 300**

This parameter specifies the number of 100 millisecond intervals to turn off a command in the error list after an error is recognized for the command. If this parameter is set to **0**, there will be no delay.

### **2.2.5 MNET Client x Commands**

The MNET Client x Commands section of the configuration sets the Modbus TCP/IP Client command list. This command list polls Modbus TCP/IP server devices attached to the Modbus TCP/IP Client port. The module supports numerous commands. This permits the module to interface with a wide variety of Modbus TCP/IP protocol devices.

The function codes used for each command are those specified in the Modbus protocol. Each command list record has the same format. The first part of the record contains the information relating to the MVI56-MNETR communication module, and the second part contains information required to interface to the Modbus TCP/IP server device.

Command List Overview

In order to interface the MVI56-MNETR module with Modbus TCP/IP server devices, you must construct a command list. The commands in the list specify the server device to be addressed, the function to be performed (read or write), the data area in the device to interface with and the registers in the internal database to be associated with the device data. The Client command list supports up to 100 commands.

The command list is processed from top (command #1) to bottom. A poll interval parameter is associated with each command to specify a minimum delay time in tenths of a second between the issuances of a command. If the user specifies a value of 10 for the parameter, the command will be executed no more frequently than every 1 second.

Write commands have a special feature, as they can be set to execute only if the data in the write command changes. If the register data values in the command have not changed since the command was last issued, the command will not be executed.

If the data in the command has changed since the command was last issued, the command will be executed. Use of this feature can lighten the load on the network. To implement this feature, set the enable code for the command to **CONDITIONAL** (2).

Commands Supported by the Module

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

The following table lists the functions supported by the module.

Function Code	Definition	Supported in Client	Supported in Server
1	Read Coil Status	X	X
2	Read Input Status	X	X
3	Read Holding Registers	X	X
4	Read Input Registers	X	X
5	Set Single Coil	X	X
6	Single Register Write	X	X
7	Read Exception Status		X
8	Diagnostics		X
15	Multiple Coil Write	X	X
16	Multiple Register Write	X	X
22	Mask Write 4X		X
23	Read/Write		X

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

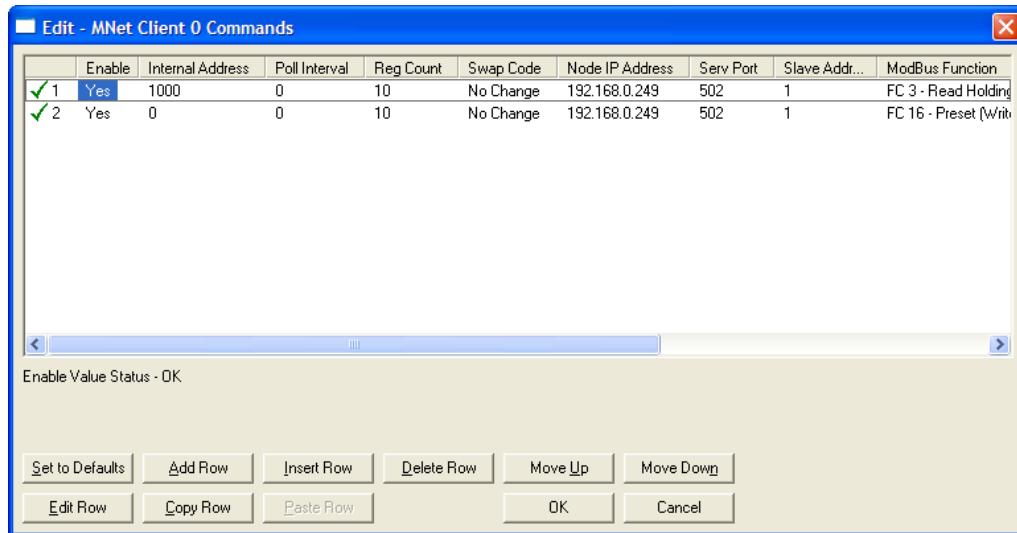
**Command Entry Formats**

The following table shows the structure of the configuration data necessary for each of the supported commands.

1	2	3	4	5	6	7	8	9	10
Enable Code	Internal Address	Poll Interval Time	Count	Swap Code	IP Address	Serv Port	Slave Node	Function Code	Device Modbus Address
Code	Register (bit)	1/10th Seconds	Bit Count	0	IP Address	Port #	Address	Read Coil (0x)	Register
Code	Register (bit)	1/10th Seconds	Bit Count	0	IP Address	Port #	Address	Read Input (1x)	Register
Code	Register	1/10th Seconds	Word Count	Code	IP Address	Port #	Address	Read Holding Registers (4x)	Register
Code	Register	1/10th Seconds	Word Count	0	IP Address	Port #	Address	Read Input Registers (3x)	Register
Code	1 bit	1/10th Seconds	Bit Count	0	IP Address	Port #	Address	Force (Write) Single Coil (0x)	Register
Code	1 bit	1/10th Seconds	Word Count	0	IP Address	Port #	Address	Preset (Write) Single Register (4x)	Register
Code	Register (bit)	1/10th Seconds	Bit Count	0	IP Address	Port #	Address	Force (Write) Multiple Coil (0x)	Register
Code	Register	1/10th Seconds	Word Count	0	IP Address	Port #	Address	Preset (Write) Multiple Register (4x)	Register

The first part of the record is the module information, which relates to the MVI56 module and the second part contains information required to interface to the server device.

**Command list example:**



Enable

**NO** (0), **YES** (1), or **CONDITIONAL** (2)

This field defines whether the command is to be executed and under what conditions.

Value	Description
<b>No</b> (0)	The command is disabled and will not be executed in the normal polling sequence.
<b>YES</b> (1)	The command is executed each scan of the command list if the <i>Poll Interval</i> time is set to zero. If the <i>Poll Interval</i> time is set to a nonzero value, the command will be executed when the interval timer expires.
<b>CONDITIONAL</b> (2)	The command will execute only if the internal data associated with the command changes. This value is valid only for write commands.

Internal Address

**0** to **4999** (for word-level addressing)

or

**0** to **65535** (for bit-level addressing)

This field specifies the database address in the module's internal database to use as the destination for data brought in by a read command or as the source for data to be sent out by a write command. The database address is interpreted as a bit address or a 16-bit word (register) address, depending on the Modbus Function Code used in the command.

- For Modbus functions 1, 2, 5, and 15, this parameter is interpreted as a bit-level address.
- For Modbus functions 3, 4, 6, and 16, this parameter is interpreted as a word- or register-level address.

Poll Interval

**0** to **65535**

This parameter specifies the minimum interval to execute continuous commands (*Enable* code of **1**). The parameter is entered in tenths of a second. Therefore, if a value of **100** is entered for a command, the command executes no more frequently than every 10 seconds.

Reg Count

Regs: **1** to **125**

Coils: **1** to **800**

This parameter specifies the number of 16-bit registers or binary bits to be transferred by the command.

- Functions 5 and 6 ignore this field as they apply only to a single data point.
- For functions 1, 2, and 15, this parameter sets the number of bits (inputs or coils) to be transferred by the command.
- For functions 3, 4, and 16, this parameter sets the number of registers to be transferred by the command.

Swap Code

**NONE**

**SWAP WORDS**

**SWAP WORDS & BYTES**

**SWAP BYTES**

This parameter defines if and how the order of bytes in data received or sent is to be rearranged. This option exists to allow for the fact that different manufacturers store and transmit multi-byte data in different combinations. This parameter is helpful when dealing with floating-point or other multi-byte values, as there is no one standard method of storing these data types. The parameter can be set to rearrange the byte order of data received or sent into an order more useful or convenient for other applications. The following table defines the valid *Swap Code* values and the effect they have on the byte-order of the data.

<b>Swap Code</b>	<b>Description</b>
<b>NONE</b>	No change is made in the byte ordering (1234 = 1234)
<b>SWAP WORDS</b>	The words are swapped (1234=3412)
<b>SWAP WORDS &amp; BYTES</b>	The words are swapped, then the bytes in each word are swapped (1234=4321)
<b>SWAP BYTES</b>	The bytes in each word are swapped (1234=2143)

These swap operations affect 4-byte (or 2-word) groups of data. Therefore, data swapping using these *Swap Codes* should be done only when using an even number of words, such as when 32-bit integer or floating-point data is involved.

Node IP Address

xxx.xxx.xxx.xxx

The IP address of the device being addressed by the command.

Service Port

**502** or other supported ports on server

Use a value of **502** when addressing Modbus TCP/IP servers that are compatible with the Schneider Electric MBAP specifications (this will be most devices). If a server implementation supports another service port, enter the value here.

Slave Address

**0** - Broadcast to all nodes

**1** to **255**

Use this parameter to specify the slave address of a remote Modbus Serial device through a Modbus Ethernet to Serial converter.

**Note:** Use the *Node IP Address* parameter (page 53) to address commands to a remote Modbus TCP/IP device.

**Note:** Most Modbus devices accept an address in the range of only 1 to 247, so check with the slave device manufacturer to see if a particular slave can use addresses 248 to 255. If the value is set to zero, the command will be a broadcast message on the network. The Modbus protocol permits broadcast commands for **write** operations. **Do not** use node address 0 for **read** operations.

Modbus Function

**1, 2, 3, 4, 5, 6, 15, or 16**

This parameter specifies the Modbus Function Code to be executed by the command. These function codes are defined in the Modbus protocol. The following table lists the purpose of each function supported by the module. More information on the protocol is available from [www.modbus.org](http://www.modbus.org).

<b>Modbus Function Code</b>	<b>Description</b>
1	Read Coil Status
2	Read Input Status
3	Read Holding Registers
4	Read Input Registers
5	Force (Write) Single Coil
6	Preset (Write) Single Register
15	Force Multiple Coils
16	Preset Multiple Registers

### MB Address in Device

This parameter specifies the starting Modbus register or bit address in the Server to be used by the command. Refer to the documentation of each Modbus Server device for the register and bit address assignments valid for that device.

The Modbus Function Code determines whether the address will be a register-level or bit-level OFFSET address into a given data type range. The offset will be the target data address in the Server minus the base address for that data type. Base addresses for the different data types are:

- 00001 or 000001 (0x0001) for bit-level Coil data (Function Codes 1, 5, and 15).
- 10001 or 100001 (1x0001) for bit-level Input Status data (Function Code 2)
- 30001 or 300001 (3x0001) for Input Register data (Function Code 4)
- 40001 or 400001 (4x0001) for Holding Register data (Function Codes 3, 6, and 16).

Address calculation examples:

- For bit-level Coil commands (FC 1, 5, or 15) to read or write a Coil 0X address 00001, specify a value of 0 (00001 - 00001 = 0).
- For Coil address 00115, specify 114  
(00115 - 00001 = 114)
- For register read or write commands (FC 3, 6, or 16) 4X range, for 40001, specify a value of 0  
(40001 - 40001 = 0).
- For 01101, 11101, 31101 or 41101, specify a value of 1100.  
(01101 - 00001 = 1100)  
(11101 - 10001 = 1100)  
(31101 - 30001 = 1100)  
(41101 - 40001 = 1100)

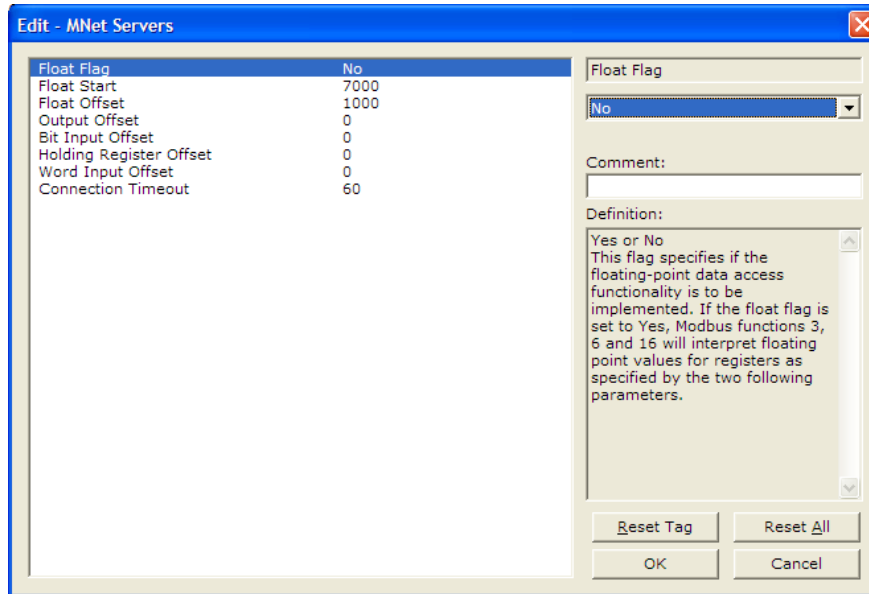
**Note:** If the documentation for a particular Modbus Server device lists data addresses in hexadecimal (base16) notation, you will need to convert the hexadecimal value to a decimal value to enter in this parameter. In such cases, it is not usually necessary to subtract 1 from the converted decimal number, as this addressing scheme typically uses the exact offset address expressed as a hexadecimal number.

### Comment

0 to 35 alphanumeric characters

### 2.2.6 MNET Servers

This section contains database offset information used by the servers when accessed by external Clients. These offsets can be utilized to segment the database by data type.



#### Float Flag

##### **YES or NO**

This flag specifies how the Server driver will respond to Function Code 3, 6, and 16 commands (read and write Holding Registers) from a remote Client when it is moving 32-bit floating-point data.

If the remote Client expects to receive or will send one complete 32-bit floating-point value for each count of one (1), then set this parameter to **YES**. When set to **YES**, the Server driver will return values from two consecutive 16-bit internal memory registers (32 total bits) for each count in the read command, or receive 32-bits per count from the Client for write commands. Example: Count = **10**, Server driver will send 20 16-bit registers for 10 total 32-bit floating-point values.

If, however, the remote Client sends a count of two (2) for each 32-bit floating-point value it expects to receive or send, or, if you do not plan to use floating-point data in your application, then set this parameter to **NO**, which is the default setting.

You will also need to set the *Float Start* and *Float Offset* parameters to appropriate values whenever the *Float Flag* parameter is set to **YES**.



### Float Start

**F0 to 65535**

This parameter defines the first register of floating-point data. All requests with register values greater than or equal to this value will be considered floating-point data requests. This parameter is only used if the *Float Flag* is enabled. For example, if a value of 7000 is entered, all requests for registers 7000 and above will be considered as floating-point data.

### Float Offset

**0 to 9999**

This parameter defines the start register for floating-point data in the internal database. This parameter is used only if the *Float Flag* is enabled. For example, if the *Float Offset* value is set to **3000** and the *Float Start* parameter is set to **7000**, data requests for register 7000 will use the internal Modbus register 3000.

### Output Offset

**0 to 4999**

This parameter defines the start register for the Modbus command data in the internal database. This parameter is enabled when a value greater than **0** is set. For example, if the *Output Offset* value is set to **3000**, data requests for Modbus Coil Register address 00001 will use the internal database register 3000, bit 0. If the *Output Offset* value is set to **3000**, data requests for Modbus Coil register address 00016 will use the internal database register 3000, bit 15. Function codes affected are 1, 5, and 15.

### Bit Input Offset

**0 to 4999**

This parameter defines the start register for Modbus command data in the internal database. This parameter is enabled when a value greater than **0** is set. For example, if the *Bit Input Offset* value is set to **3000**, data requests for Modbus Input Register address 10001 will use the internal database register 3000, bit 0. If the *Bit Input Offset* is set to **3000**, data requests for Modbus Coil register address 10016 will use the internal database register 3000, bit 15. Function code 2 is affected.

### Holding Register Offset

**0 to 4999**

This parameter defines the start register for the Modbus Command data in the internal database. This parameter is enabled when a value greater than **0** is set. For example, if the *Holding Register Offset* value is set to **4000**, data requests for Modbus Word register 40001 will use the internal database register 4000. Function codes affected are 3, 6, 16, & 23.

### Word Input Offset

**0 to 4999**

This parameter defines the start register for Modbus Command data in the internal database. This parameter is enabled when a value greater than **0** is set. For example, if the *Word Input Offset* value is set to **4000**, data requests for Modbus Word register address 30001 will use the internal database register 4000. Function code 4 is affected.

### **2.2.7 Static ARP Table**

The Static ARP Table defines a list of static IP addresses that the module will use when an ARP (Address Resolution Protocol) is required. The module will accept up to 40 static IP/MAC address data sets.

Use the Static ARP table to reduce the amount of network traffic by specifying IP addresses and their associated MAC (hardware) addresses that the MVI56-MNETR module will be communicating with regularly.

**Important:** If the device in the field is changed, this table must be updated to contain the new MAC address for the device and downloaded to the module. If the MAC is not changed, no communications with the module will be provided.

### IP Address

Dotted notation

This table contains a list of static IP addresses that the module will use when an ARP is required. The module will accept up to 40 static IP/MAC address data sets.

**Important:** If the device in the field is changed, this table must be updated to contain the new MAC address for the device and downloaded to the module. If the MAC is not changed, no communications with the module will occur.

### Hardware MAC Address

Hex value

This table contains a list of static MAC addresses that the module will use when an ARP is required. The module will accept up to 40 static IP/MAC address data sets.

**Important:** If the device in the field is changed, this table must be updated to contain the new MAC address for the device and downloaded to the module. If the MAC is not changed, no communications with the module will occur.

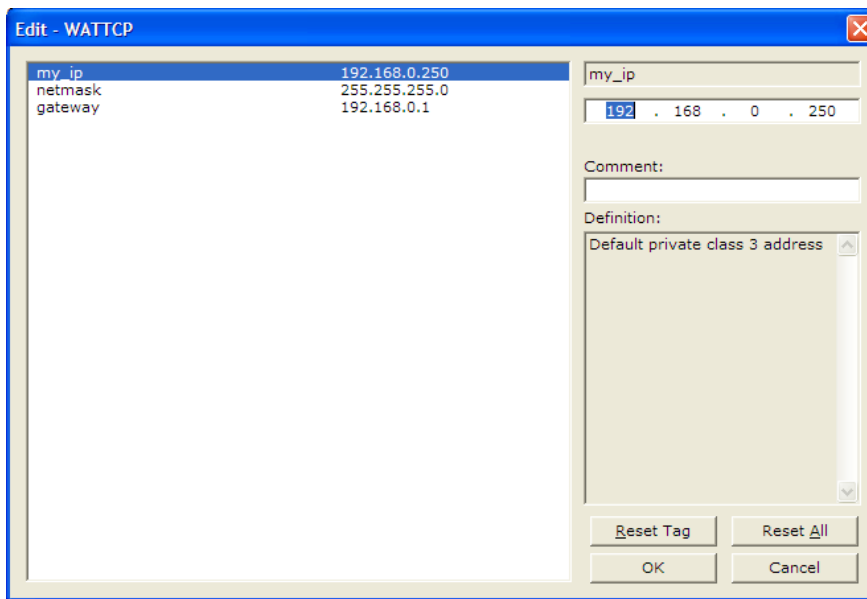
### 2.2.8 Ethernet Configuration

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

- 1 Determine the network settings for your module, with the help of your network administrator if necessary. You will need the following information:
  - IP address (fixed IP required) \_\_\_\_\_ . \_\_\_\_\_ . \_\_\_\_\_ . \_\_\_\_\_
  - Subnet mask \_\_\_\_\_ . \_\_\_\_\_ . \_\_\_\_\_ . \_\_\_\_\_
  - Gateway address \_\_\_\_\_ . \_\_\_\_\_ . \_\_\_\_\_ . \_\_\_\_\_

**Note:** The gateway address is optional, and is not required for networks that do not use a default gateway.

- 2 Double-click the **ETHERNET CONFIGURATION** icon. This action opens the *Edit* dialog box.

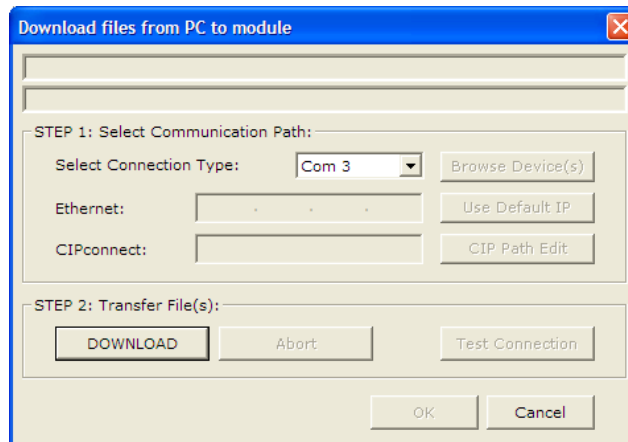


- 3 Edit the values for *my\_ip*, *netmask* (subnet mask) and *gateway* (default gateway).
- 4 When you are finished editing, click **OK** to save your changes and return to the *ProSoft Configuration Builder* window.

### 2.3 Downloading the Project to the Module Using a Serial COM port

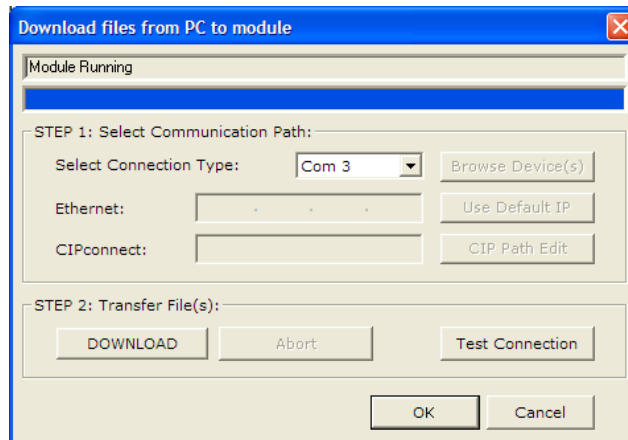
For the module to use the settings you configured, you must download (copy) the updated *Project* file from your PC to the module.

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



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

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





## 3 Ladder Logic

### In This Chapter

- ❖ MNETRMODULEDEF ..... 64
- ❖ Modbus Message Data..... 71
- ❖ Using the Sample Program - RSLogix 5000 Version 15 and earlier..... 72

Ladder logic is required for application of the MVI56-MNETR module. Tasks that must be handled by the ladder logic are module data transfer, special block handling, and status data receipt. Additionally, a power-up handler may be needed to handle the initialization of the module's data and to clear any processor fault conditions.

The sample ladder logic, on the *ProSoft Solutions CD-ROM*, is extensively commented, to provide information on the purpose and function of each rung. For most applications, the sample ladder will work without modification.

### 3.1 MNETRMODULEDEF

All data related to the MVI56-MNETR is stored in a user defined data type. An instance of the data type is required before the module can be used. This is done by declaring a variable of the data type in the **CONTROLLER TAGS EDIT TAGS** dialog box.

The following table describes the structure of the object.

Name	Data Type	Description
DATA	MNETRDATA (page 64)	These objects hold data to be transferred between the processor and the MVI56-MNETR module
STATUS	MNETRSTATUS (page 65)	This object views the status of the module.
CONTROL	MNETRCONTROL (page 66)	This object contains the data structure required for the processor to request special tasks from the module
UTIL	MNETRUTIL (page 70)	This data object stores the variables required for the data transfer between the processor and the module.

This object contains objects that define the configuration, user data, status and command control data related to the module. Each of these object types is discussed in the following topics of the document.

#### 3.1.1 MNETRDATA

This object holds data to be transferred between the processor and the MVI56-MNETR module. The user data is the read and write data transferred between the processor and the module as "pages" of data up to 40 words long.

Name	Data Type	Description
ReadData	INT[600]	Data read from module
WriteData	INT[600]	Data to write to module

The read data (**READDATA**) is an array set to match the value entered in the Read Register Count parameter of the MNET.CFG file. For ease of use, this array should be dimensioned as an even increment of 40 words. This data is paged up to 50 words at a time from the module to the processor. The ReadData task places the data received into the proper position in the read data array. Use this data for status and control in the ladder logic of the processor.

The write data (**WRITEDATA**) is an array set to match the value entered in the Write Register Count parameter of the MNET.CFG file. For ease of use, this array should be dimensioned as even increments of 40 words. This data is paged up to 40 words at a time from the processor to the module. The WriteData task places the write data into the output image for transfer to the module. This data is passed from the processor to the module for status and control information for use in other nodes on the network.



### 3.1.2 MNETRSTATUS

This object views the status of the module. The **MNETRSTATUS** object shown below is updated each time a read block is received by the processor. Use this data to monitor the state of the module at a "real-time rate".

The following table describes the structure of this object.

Name	Data Type	Description
PassCnt	INT	Program cycle counter
ProductVersion	INT	Shows the module software version
ProductCode	INT[2]	This identifies the module product code
BlockStats	MNETRBLOCKSTATS (page 65)	Status information for the data transfer operations between the processor and the module
Reserved1	INT	Reserved for future use
Reserved2	INT	Reserved for future use
MNETReq	INT	The number of MNET (Port 2000) requests received
MNETResp	INT	The number of MNET (Port 2000) responses sent
MBAPReq	INT	The number of MBAP (Port 502) requests received
MBAPResp	INT	The number of MBAP (Port 502) responses sent
ClientStatus	MNETRCLIENTSTATS (page 66)	Client Status Data

#### MNETRBLOCKSTATS

This status object contains a structure that includes the status information for the data transfer operations between the processor and the module (**MNETRBLOCKSTATS**). The following table describes the structure of this object.

Name	Data Type	Description
Read	INT	Total number of read block transfers
Write	INT	Total number of write block transfers
Parse	INT	Total number of blocks parsed
Event	INT	Total number of event blocks received
Cmd	INT	Total number of command blocks received
Err	INT	Total number of block transfer errors

***MNETRCLIENTSTATS***

The status object contains a structure for the MNET Client Status (**MNETRCLIENTSTATS**). The following table describes the structure of this object.

<b>Name</b>	<b>Data Type</b>	<b>Description</b>
CmdReq	INT	Total number of command list requests sent
CmdResp	INT	Total number of command list responses received
CmdErr	INT	Total number of command list errors
Requests	INT	Total number of requests for port
Responses	INT	Total number of responses for port
ErrSent	INT	Total number of errors sent
ErrRec	INT	Total number of errors received
CfgErrWord	INT	Configuration Error Word
CurErr	INT	Current Error code
LastErr	INT	Last recorded error code

Refer to MVI56-MNETR Status Data Definition for a complete listing of the data stored in the status object.

**3.1.3 MNETRCONTROL**

Contains the data structure required for the processor to request special tasks from the module. The command control task allows the processor to dynamically enable commands configured in the port command list. The event command task allows the processor to dynamically build any commands to be sent by the MNET Client to a remote Server.

The following table describes the structure of this object.

<b>Name</b>	<b>Data Type</b>	<b>Description</b>
BootTimer	TIMER	Timer to clear warmboot and coldboot
WarmBoot	BOOL	Triggers a Cold Boot Command
ColdBoot	BOOL	Triggers a Warm Boot Command
EventCmdTrigger	BOOL	Triggers the Event Command.
EventCmd	MNETREVENTCMD (page 67)	This object contains the attributes to define a Master command. An array of these objects is used for each port.
CmdControl	MNETRCMDCONTROL (page 68)	Controls the execution of the commands listed in the configuration under the [MNET Client 0 Commands] section.
PassThru	MNETRPASSTHRU (page 69)	Transfers a remote Client's commands through the MNET Module straight into the Processor's Controller tags.
IPAddress	MNETIPADDRESS	Getting and Setting IP address to and from Module

**MNETREVENTCMD**

The **MNETREVENTCMD** structure holds the information required for an event command. An array of this object should be defined and should hold the event command set to be employed in the application. The following table describes the structure of this object.

<b>Name</b>	<b>Data Type</b>	<b>Description</b>
IP0	INT	First digit of IP address
IP1	INT	Second digit of IP address
IP2	INT	Third digit of IP address
IP3	INT	Last digit of IP address
ServPort	INT	TCP Service Port number (0-65535), 502 for MBAP, 2000 for MNET
Node	INT	Modbus slave node address (0 to 247)
DBAddress	INT	Module internal database for message
Count	INT	Register or data point count
Swap	INT	Swap code for functions 3 and 4
Function	INT	Modbus function code for message
Address	INT	Address to interface with in device
Result	INT	Shows the result of the event that was sent

**MNETRCMDCONTROL**

When the command bit (**MNETR.CONTROL.CMDCONTROL.CMDCONTROLTRIGGER**) is set in the example ladder logic, the module will build a block 9901 with the number of commands set through: **MNETR.CONTROL.CMDCONTROL.NUMBEROFCOMMANDS[0]**.

The command indexes will be set through the controller tags starting from **MNETR.CONTROL.CMDCONTROL.CMDINDEX[0]** to **MNETR.CONTROL.CMDCONTROL.CMDINDEX[5]**

For example, in order to enable commands 0, 2 and 5 the following values would be set:

- **MNETR.CONTROL.CMDCONTROL.CMDINDEX[0] = 3**
- **MNETR.CONTROL.CMDCONTROL.CMDINDEX[1] = 0**
- **MNETR.CONTROL.CMDCONTROL.CMDINDEX[2] = 2**
- **MNETR.CONTROL.CMDCONTROL.CMDINDEX[3] = 5**

The module will receive this block and build and send the command to the specified control device using a MSG block.

The following table describes the data for the command element in MNETRCmdControl.

<b>Name</b>	<b>Data Type</b>	<b>Description</b>
CmdIndex	INT[6]	The position of the initial command to execute from the Client command list.
NumberOfCommands	INT	The number of commands to execute from the Client command list
CommandsAddedtoQueue	INT	Number of commands added to queue
CmdControlTrigger	BOOL	Trigger Command Control. User application will activate this trigger.

**MNETRPASSTHRU**

During pass-through operation, write messages received at the MVI56-MNETR server write messages through to the processor. It is the responsibility of the ladder logic to process the message received using this feature. Two data objects are required for this mode: a variable to hold the length of the message and a buffer to hold the message.

This information is passed from the module to the processor using a block identification code of 9996 if the unformatted pass-through mode (code 1) is selected as the pass through mode in the configuration file. Word one of this block contains the length of the message and the message starts at word 3. Other controller tags are required to store the controlled values contained in these messages. The Modbus protocol supports control of binary output (coils - functions 5 and 15) and registers (functions 6 and 16).

Additionally, formatted message blocks can be sent from the module to the processor when the pass-through option is selected using the format selection (codes 2 or 3 in the MNET.CFG file). These blocks require less decoding than the unformatted blocks. Refer to the user manual for a full discussion on utilizing the pass-through option in an application.

The following table describes the structure of this object.

<b>Name</b>	<b>Data Type</b>	<b>Description</b>
MBControl1	CONTROL	Modbus pass thru message control
MBControl2	CONTROL	Modbus pass thru message control
MBMsg	SINT[500]	Message array
MBScratch	INT[3]	Temporary used ints
MBOffsetBit	INT	Offset bit in the message
MBOffset	INT	Start offset in the message
MBMsgLen	INT	The length of the Modbus message in bytes
mbdouble	DINT	Modbus double int tag
MBCoil	MNETRCOILARRAY	Conversion from Bool to INT data types

### 3.1.4 MNETRUTIL

This data object stores the variables required for the data transfer between the processor and the MVI56-MNETR module.

**Caution:** These variables are for internal ladder usage only. Do not use these variables in your own application, otherwise unpredictable results could occur.

The following table describes the structure of this object.

Name	Data Type	Description
LastRead	INT	Index of last read block
LastWrite	INT	Index of last write block
BlockIndex	INT	Computed block offset for data table
ReadDataSizeGet	INT	Gets ReadData Array Length.
WriteDataSizeGet	INT	Gets WriteData Array Length.
ReadDataBlkCount	INT	Holds the value of the Block Counts of the Read Data Array. Array Size is divided by 40.
WriteDataBlkCount	INT	Holds the value of the Block Counts of the Write Data Array. Array Size is divided by 40.
RBTSremainder	INT	Holds remainder calculation value from the read array.
WBTSremainder	INT	Holds remainder calculation value from the write array.
IPsetPending	BOOL	Allows Setting module IP address
IPgetPending	BOOL	Allows Getting module IP address
InitOutputData	MNETRINITOUTDATA	This is to initialize output data

The LastRead tag stores the latest Read Block ID received from the module. The LastWrite tag stores the latest Write Block ID to be sent to the module. The Block Index tag is an intermediate variable used during the block calculation.

### 3.2 Modbus Message Data

During pass-through operation, write messages received at the MVI56-MNETR server write messages through to the processor. It is the responsibility of the ladder logic to process the message received using this feature. Two data objects are required for this mode: a variable to hold the length of the message and a buffer to hold the message.

This information is passed from the module to the processor using a block identification code of 9996 if the unformatted pass-through mode (code 1) is selected as the pass through mode in the configuration file. Word one of this block contains the length of the message and the message starts at word 3. Other controller tags are required to store the controlled values contained in these messages. The Modbus protocol supports control of binary output (coils - functions 5 and 15) and registers (functions 6 and 16).

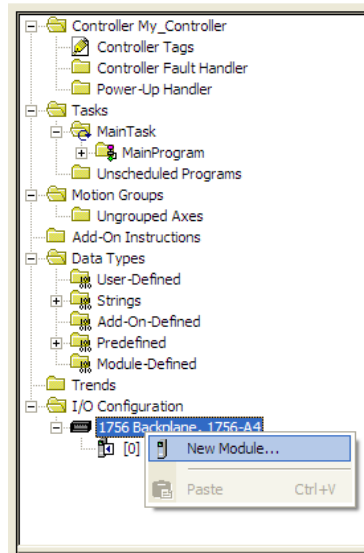
Additionally, formatted message blocks can be sent from the module to the processor when the pass-through option is selected using the format selection (codes 2 or 3 in the MNET.CFG file). These blocks require less decoding than the unformatted blocks. Refer to Pass-Through Control Blocks (page 106) for a full discussion on utilizing the pass-through option in an application.

### 3.3 Using the Sample Program - RSLogix 5000 Version 15 and earlier

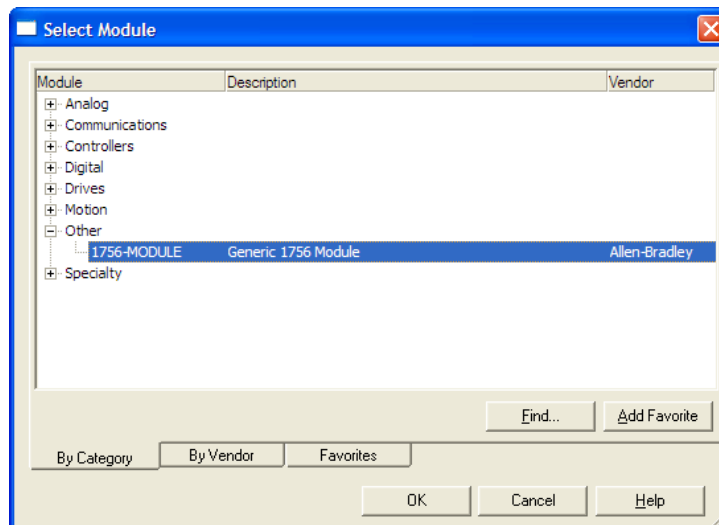
The sample program included with your MVI56-MNETR module contains predefined controller tags, configuration information, data types, and ladder logic that allow the module to communicate between the ControlLogix processor and a network of Modbus TCP/IP devices. For most applications, the sample program will work without modification.

#### 3.3.1 Adding the Module to an Existing Project

- 1 **Add the MVI56-MNETR module to the project.** Select the **I/O CONFIGURATION** folder in the **CONTROLLER ORGANIZATION** window, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.

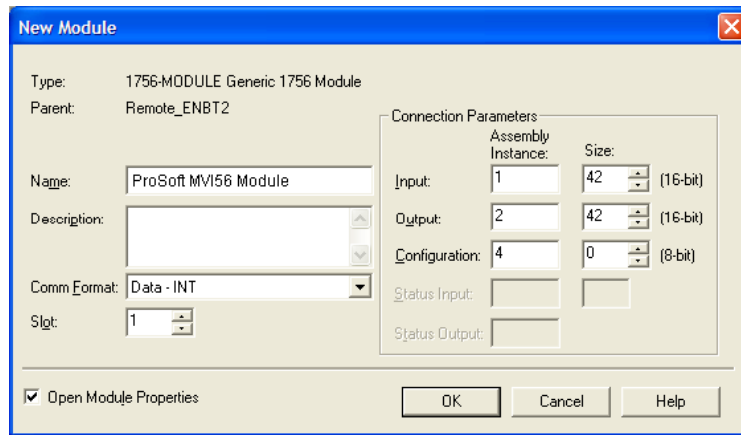


This action opens the **SELECT MODULE** dialog box:





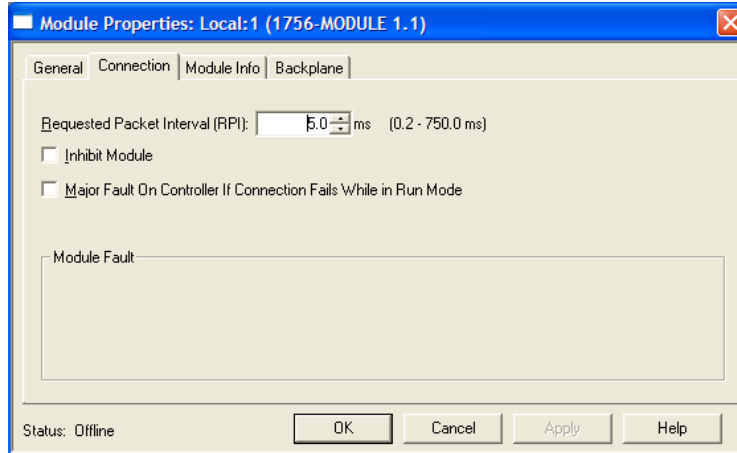
Select the **1756-MODULE** (Generic 1756 Module) from the list and click **OK**. This action opens the **NEW MODULE** dialog box.



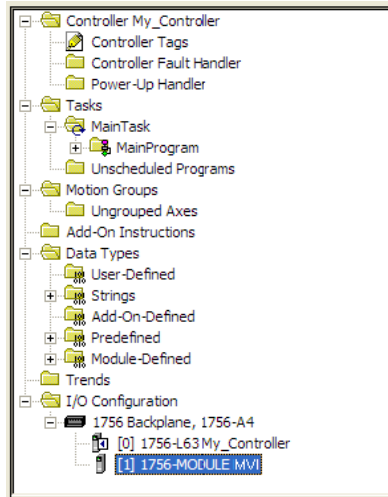
Parameter	Value
Name	Enter a module identification string. The recommended value is MNET.
Description	Enter a description for the module. Example: Modbus TCP/IP Interface Module with Reduced Data Block.
Comm Format	Select <b>DATA-INT (Very Important)</b>
Slot	Enter the slot number in the rack where the MVI56-MNETR module will be installed.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

Enter the Name, Description and Slot options for your application. You must select the **COMM FORMAT AS DATA - INT** in the dialog box, otherwise the module will not communicate over the backplane of the ControlLogix rack. Click OK to continue.

- 2 Edit the Module Properties.** Select the **REQUESTED PACKET INTERVAL** value for scanning the I/O on the module. This value represents the minimum frequency that the module will handle scheduled events. This value should not be set to less than 1 millisecond. The default value is 5 milliseconds. Values between 1 and 10 milliseconds should work with most applications.



- 3 Save the module.** Click **OK** to dismiss the dialog box. The **CONTROLLER ORGANIZATION** window now displays the module's presence.



- 4 Copy the Controller Tags** from the sample program.
- 5 Copy the User Defined Data Types** from the sample program.
- 6 Copy the Ladder Rungs** from the sample program.
- 7 Save and Download** (page 37) the new application to the controller and place the processor in run mode.

## 4 Diagnostics and Troubleshooting

### In This Chapter

❖ LED Indicators.....	76
❖ Using ProSoft Configuration Builder (PCB) for Diagnostics.....	80
❖ Reading Status Data from the Module .....	91

The module provides information on diagnostics and troubleshooting in the following forms:

- LED status indicators on the front of the module provide general information on the module's status.
- Status data contained in the module can be viewed through the Configuration/Debug port, using the troubleshooting and diagnostic capabilities of *ProSoft Configuration Builder (PCB)*.
- Status data values can be transferred from the module to processor memory and can be monitored there manually or by customer-created logic. For details on Status Data values, see MVI56-MNETR Status Data Area.

## 4.1 LED Indicators

The LEDs indicate the module's operating status as follows:

LED	Color	Status	Indication
CFG	Green	ON	Data is being transferred between the module and a remote terminal using the Configuration/Debug port.
		OFF	No data is being transferred on the Configuration/Debug port.
P1	Green	ON	Port not used
		OFF	Port not used
P2	Green	ON	Port not used
		OFF	Port not used
APP	Amber	OFF	The MVI56-MNETR is working normally.
		ON	The MVI56-MNETR module program has recognized a communication error.
BP ACT	Amber	ON	The LED is ON when the module is performing a write operation on the backplane.
		OFF	The LED is OFF when the module is performing a read operation on the backplane. Under normal operation, the LED should blink rapidly ON and OFF.
OK	Red / Green	OFF	The card is not receiving any power and is not securely plugged into the rack.
		GREEN	The module is operating normally.
		RED	The program has detected an error or is being configured. If the LED remains RED for more than 10 seconds, the program has probably halted. Remove the card from the rack and re-insert the card to restart the module's program.
BAT	Red	OFF	The battery voltage is OK and functioning.
		ON	The battery voltage is low or battery is not present. Allow battery to charge by keeping module plugged into rack for 24 hours. If BAT LED still does not go OFF, contact ProSoft Technology, as this is not a user serviceable item.

If the APP, BP ACT and OK LEDs blink at a rate of every one-second, this indicates a serious problem with the module. Call ProSoft Technology support to arrange for repairs.

### 4.1.1 Client Configuration Error Word

If a configuration error is found for the Client, the *Client Configuration Error Word* will have a value other than zero. The *Configuration Error Word* is a controller tag in *MNET.STATUS.ClientStats* in the ladder logic.

The bits in the *Configuration Error Word* have the following definitions:

Bit	Description	Value
0		0x0001
1		0x0002
2		0x0004
3		0x0008
4	Invalid retry count parameter	0x0010
5	The float flag parameter is not valid.	0x0020
6	The float start parameter is not valid.	0x0040
7	The float offset parameter is not valid.	0x0080
8	The ARP Timeout is not in range (ARP Timeout parameter 0 or greater than 60000 milliseconds) and will default to 5000 milliseconds.	0x0100
9	The Command Error Delay is > 300 and will default to 300.	0x0200
10		0x0400
11		0x0800
12		0x1000
13		0x2000
14		0x4000
15		0x8000

Correct any invalid data in the configuration for proper module operation. When the configuration contains a valid parameter set, all the bits in the configuration word will be clear. This does not indicate that the configuration is valid for the user application. Make sure each parameter is set correctly for the specific application.

### 4.1.2 Ethernet LED Indicators

LED	State	Description
Data	OFF	No activity on the Ethernet port.
	GREEN Flash	The Ethernet port is actively transmitting or receiving data.
Link	OFF	No physical network connection is detected. No Ethernet communication is possible. Check wiring and cables.
	GREEN Solid	Physical network connection detected. This LED must be ON solid for Ethernet communication to be possible.

### **4.1.3 Clearing a Fault Condition**

Typically, if the OK LED on the front of the module turns RED for more than ten seconds, a hardware problem has been detected in the module or the program has exited.

To clear the condition, follow these steps:

- 1** Turn off power to the rack.
- 2** Remove the card from the rack.
- 3** Verify that all jumpers are set correctly.
- 4** If the module requires a Compact Flash card, verify that the card is installed correctly.
- 5** Re-insert the card in the rack and turn the power back on.
- 6** Verify correct configuration data is being transferred to the module from the ControlLogix controller.

If the module's OK LED does not turn GREEN, verify that the module is inserted completely into the rack. If this does not cure the problem, contact ProSoft Technology Technical Support.

#### 4.1.4 Troubleshooting

Use the following troubleshooting steps if you encounter problems when the module is powered up. If these steps do not resolve your problem, please contact ProSoft Technology Technical Support.

##### Processor Errors

Problem description	Steps to take
Processor fault	Verify that the module is plugged into the slot that has been configured for the module in the I/O Configuration of RSLogix. Verify that the slot location in the rack has been configured correctly in the ladder logic.
Processor I/O LED flashes	This indicates a problem with backplane communications. A problem could exist between the processor and any installed I/O module, not just the MVI56-MNETR. Verify that all modules in the rack are correctly configured in the ladder logic.

##### Module Errors

Problem description	Steps to take
BP ACT LED (not present on MVI56E modules) remains OFF or blinks slowly MVI56E modules with scrolling LED display: <Backplane Status> condition reads ERR	This indicates that backplane transfer operations are failing. Connect to the module's Configuration/Debug port to check this. To establish backplane communications, verify the following items: <ul style="list-style-type: none"> <li>▪ The processor is in RUN or REM RUN mode.</li> <li>▪ The backplane driver is loaded in the module.</li> <li>▪ The module is configured for read and write data block transfer.</li> <li>▪ The ladder logic handles all read and write block situations.</li> <li>▪ The module is properly configured in the processor I/O configuration and ladder logic.</li> </ul>
OK LED remains RED	The program has halted or a critical error has occurred. Connect to the Configuration/Debug port to see if the module is running. If the program has halted, turn off power to the rack, remove the card from the rack and re-insert it, and then restore power to the rack.

## 4.2 Using ProSoft Configuration Builder (PCB) for Diagnostics

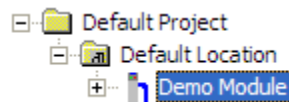
The *Configuration and Debug* menu for this module is arranged as a tree structure, with the *Main* menu at the top of the tree, and one or more submenus for each menu command. The first menu you see when you connect to the module is the *Main* menu.

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

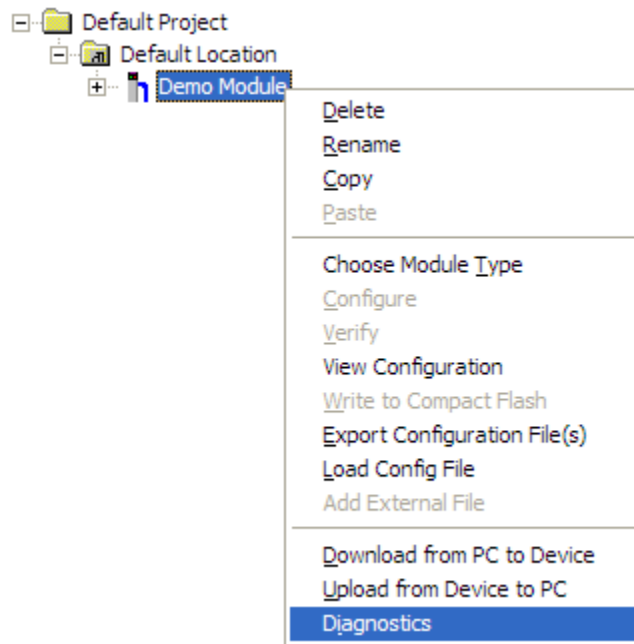
### 4.2.1 Using the Diagnostic Window in ProSoft Configuration Builder

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

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



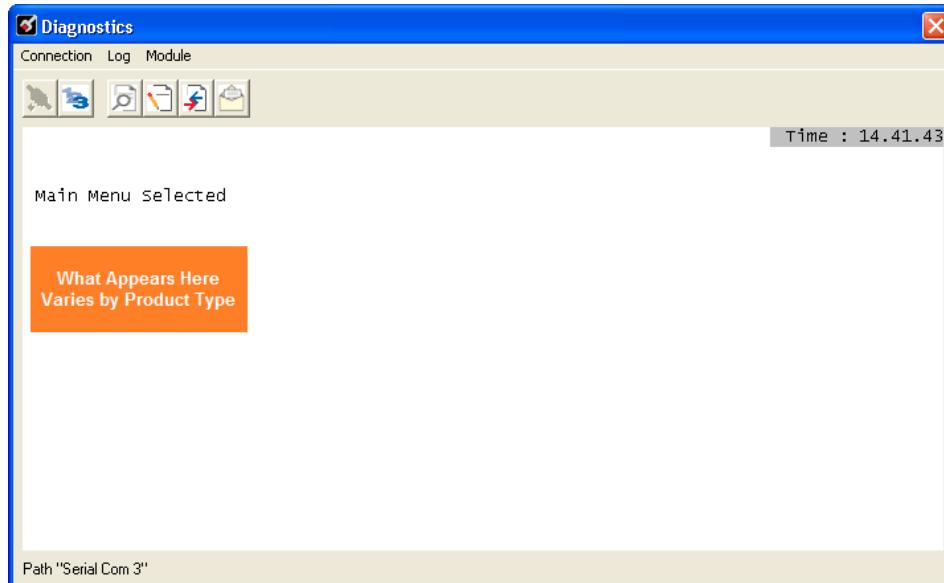
- 2 On the shortcut menu, choose **DIAGNOSTICS**.



This action opens the *Diagnostics* dialog box.

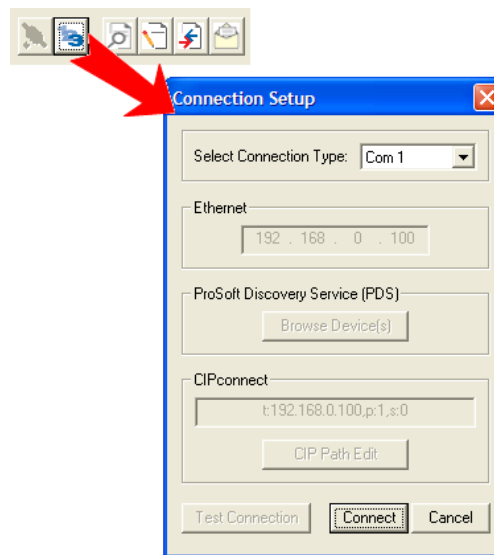


- 3 Press [?] to open the *Main* menu.



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

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



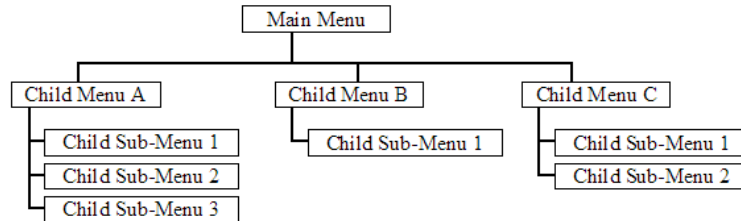
- 2 Verify that the null modem cable is connected properly between your computer's serial port and the module. A regular serial cable will not work.
- 3 On computers with more than one serial port, verify that your communication program is connected to the same port that is connected to the module.

If you are still not able to establish a connection, contact ProSoft Technology for assistance.

### 4.2.2 Navigation

All of the submenus for this module contain commands to redisplay the menu or return to the previous menu. You can always return from a submenu to the next higher menu by pressing **[M]** on your keyboard.

The organization of the menu structure is represented in simplified form in the following illustration:



The remainder of this section shows the menus available for this module, and briefly discusses the commands available to you.

#### Keystrokes

The keyboard commands on these menus are usually not case sensitive. You can enter most commands in lowercase or uppercase letters.

The menus use a few special characters (**?**, **-**, **+**, **@**) that must be entered exactly as shown. Some of these characters will require you to use the **SHIFT**, **CTRL**, or **ALT** keys to enter them correctly. For example, on US English keyboards, enter the **?** command as **SHIFT** and **/**.

Also, take care to distinguish the different uses for uppercase letter "eye" (**I**), lowercase letter "el" (**L**), and the number one (**1**). Likewise, uppercase letter "oh" (**O**) and the number zero (**0**) are not interchangeable. Although these characters look alike on the screen, they perform different actions on the module and may not be used interchangeably.

### 4.2.3 Main Menu

When you first connect to the module from your computer, your terminal screen will be blank. To activate the main menu, press the **[?]** key on your computer's keyboard. If the module is connected properly, the following menu will appear.

```
MVI56-MNETR COMMUNICATION MODULE MENU
?=Display Menu
B=Block Transfer Statistics
C=Module Configuration
D=Modbus Database View
Command List Errors:  E=Client 0
Command List:        I=Client 0
R=Transfer Configuration from PC to MVI Unit
S=Transfer Configuration from MVI Unit to PC
U=Reset diagnostic data
V=Version Information
W=Warm Boot Module
Communication Status: 1=Network  0=Client 0  4=NIC Status
Configuration:       5=Client 0  6=Servers  7=Static ARP Table
@=Network Menu       Esc=Exit Program
```

**Caution:** Some of the commands available to you from this menu are designed for advanced debugging and system testing only, and can cause the module to stop communicating with the processor or with other devices, resulting in potential data loss or other failures. Only use these commands if you are specifically directed to do so by ProSoft Technology Technical Support staff. Some of these command keys are not listed on the menu, but are active nevertheless. Please be careful when pressing keys so that you do not accidentally execute an unwanted command.

#### Viewing Block Transfer Statistics

Press **[B]** from the *Main* menu to view the *Block Transfer Statistics* screen. Use this command to display the configuration and statistics of the backplane data transfer operations between the module and the processor. The information on this screen can help determine if there are communication problems between the processor and the module.

**Tip:** To determine the number of blocks transferred each second, mark the numbers displayed at a specific time. Then some seconds later activate the command again. Subtract the previous numbers from the current numbers and divide by the quantity of seconds passed between the two readings.

#### Viewing Module Configuration

Press **[C]** to view the *Module Configuration* screen. Use this command to display the current configuration and statistics for the module.

#### Opening the Database View Menu

Press **[D]** to open the *Database View* menu. Use this menu command to view the current contents of the module's database. For more information about this submenu, see Database View Menu (page 86).

Opening the Command List Menu

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

Opening the Command Error List Menu

Press **[I]** to open the Command Error List. This list consists of multiple pages of command list error/status data. Press **[?]** to view a list of commands available on this menu.

Receiving the Configuration File

Press **[R]** to download (receive) the current configuration file from the module.

Sending the Configuration File

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

Resetting Diagnostic Data

Press **[U]** to reset the status counters for the Client and/or servers in the module.

Viewing Version Information

Press **[V]** to view version information for the module.

Use this command to view the current version of the software for the module, as well as other important values. You may be asked to provide this information when calling for technical support on the product.

Values at the bottom of the display are important in determining module operation. The *Program Scan Counter* value is incremented each time a module's program cycle is complete.

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

Viewing Network Status

Press **[1]** to view statistics for the network server ports. The Network Server Ports Status screen shows the number of requests, responses, and errors for each network server.

```
NETWORK SERVER PORTS STATUS:
MNET SERVER (Port 2000):
  Number of Requests : 0
  Number of Responses : 0
  Number of Errors Received : 0
  Number of Errors Sent : 0
MBAP SERVER (Port 502):
  Number of Requests : 30230
  Number of Responses : 30230
  Number of Errors Received : 0
  Number of Errors Sent : 0
HTTP SERVER (Port 80):
  Number of Requests : 984
  Number of Responses : 1968
  Number of Errors Received : 0
  Number of Errors Sent : 0
```

### Viewing Client Status

Press **[0]** (zero) to display the statistics of the Client.

### Viewing NIC Status

Press **[4]** to view NIC status. Use this command to view the communication status for the Network Interface Card.

### Viewing Client Configuration

Press **[5]** to display the configuration information for the Client.

### Viewing Server Configuration

Press **[6]** to display the configuration information for the servers.

### Viewing the Static ARP Table

Press **[7]** to view the Static ARP Table. Use this command to view the list of IP and MAC addresses that are configured not to receive ARP messages from the module.

```
STATIC ARP TABLE DEFINED (Count=4)
105.102.0.15  00:00:8D:B0:0A:16  105.102.0.16  00:00:8D:B0:0A:16
105.102.0.17  00:00:8D:B0:0A:16  105.102.0.18  00:00:8D:B0:0A:16
```

### Opening the Network Menu

Press **[@]** to open the *Network* menu.

The *Network* menu allows you to send, receive and view the WATTCP.CFG file that contains the IP, gateway and other network specification information. For more information about this submenu, see Network Menu (page 89).

### Warm Booting the Module

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

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

### Exiting the Program

Press **[ESC]** to restart the module and force all drivers to be loaded. The module will use the configuration stored in the module's Flash memory to configure the module.

#### 4.2.4 Modbus Database View Menu

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

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

All data contained in the module's database is available for viewing using the commands. Refer to the Modbus Protocol Specification (page 120) for information on the structure of Modbus messages. Each option available on the menu is discussed in the following topics.

##### Viewing Register Pages

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

Command	Description
<b>[0]</b>	Display registers 0 to 99
<b>[1]</b>	Display registers 1000 to 1099
<b>[2]</b>	Display registers 2000 to 2099

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

##### Redisplaying the Current Page

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

##### Moving Back Through 5 Pages of Registers

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

##### Viewing the Previous Page of Registers

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

### Moving Forward Through 5 Pages of Registers

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

### Viewing the Next Page of Registers

Press **[N]** from the *Database View* menu to display the next page of data.

### Viewing Data in Decimal Format

Press **[D]** from the *Database View* menu to display the data on the current page in decimal format.

### Viewing Data in Hexadecimal Format

Press **[H]** from the *Database View* menu to display the data on the current page in hexadecimal format.

### Viewing Data in Floating-Point Format

Press **[F]** from the *Database View* menu to display the data on the current page in floating-point format. The program assumes that the values are aligned on even register boundaries. If floating-point values are not aligned as such, they are not displayed properly.

### Viewing Data in ASCII (Text) Format

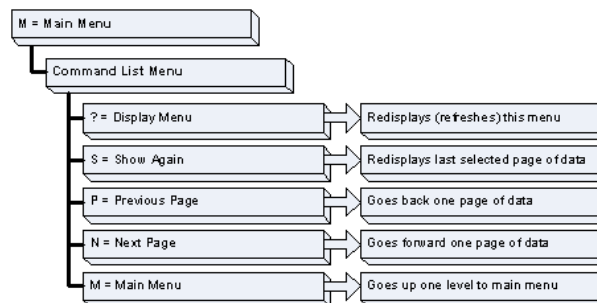
Press **[A]** from the *Database View* menu to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

### Returning to the Main Menu

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

## **4.2.5 Command List Menu**

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



Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Viewing the Previous Page of Commands

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

Viewing the Next Page of Commands

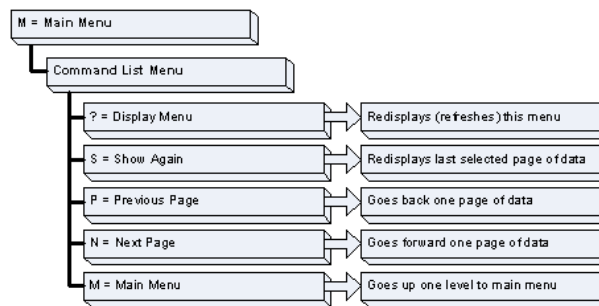
Press **[N]** to display the next page of commands.

Returning to the Main Menu

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

**4.2.6 Master Command Error List Menu**

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



Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Viewing the Previous Page of Commands

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

Viewing the Next Page of Commands

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

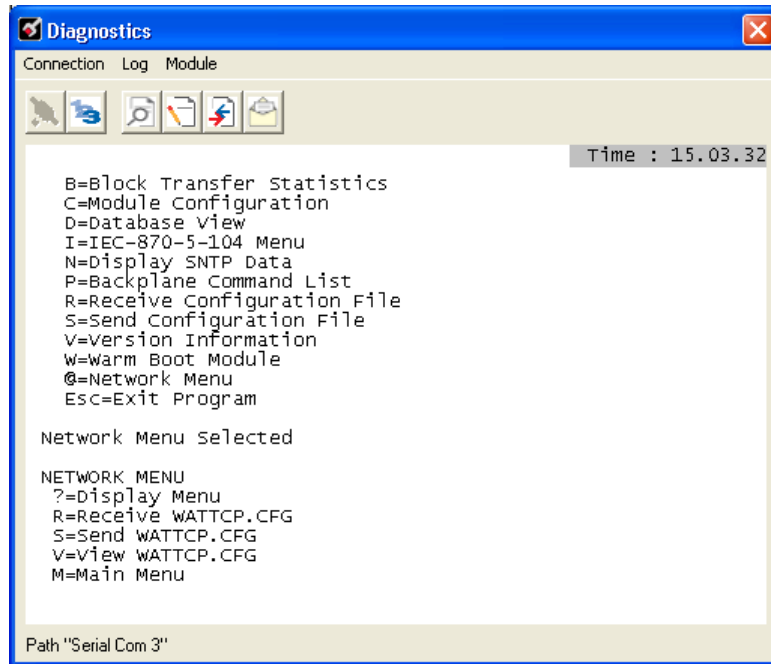
Returning to the Main Menu

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



### 4.2.7 Network Menu

From the *IEC-870-5-104 Server* menu press [**@**] to display the *IEC-870-5-104 Network* menu screen. The *Network* menu allows you to send, receive, and view the WATTCP.CFG file that contains the IP and module addresses, and other network information.



#### Transferring WATTCP.CFG to the Module

Press [**R**] to transfer a new WATTCP.CFG file from the PC to the module. Use this command to change the network configuration for the module (for example, the module's IP address).

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

#### Transferring WATTCP.CFG to the PC

Press [**S**] to transfer the WATTCP.CFG file from the module to your PC.

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

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

Viewing the WATTCP.CFG File on the module

Press **[V]** to view the module's WATTCP.CFG file. Use this command to confirm the module's current network settings.

```
WATTCP.CFG FILE:
# ProLinx Communication Gateways, Inc.
# Default private class 3 address
my_ip=192.168.0.75
# Default class 3 network mask
netmask=255.255.255.0
# name server 1 up to 9 may be included
# nameserver=xxx.xxx.xxx.xxx
# name server 2
# nameserver=xxx.xxx.xxx.xxx
# The gateway I wish to use
gateway=192.168.0.1
# some networks (class 2) require all three parameters
# gateway,network,subnetmask
# gateway 192.168.0.1,192.168.0.0,255.255.255.0
# The name of my network
# domainlist="mynetwork.name"
```

Returning to the Main Menu

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

### 4.3 Reading Status Data from the Module

The MVI56-MNETR module returns a Status Data block that can be used to determine the module's operating status. This data is located in the module's database at a location specified by the Error Status Pointer configuration parameter. This data is transferred to the ControlLogix processor continuously.

The Configuration/Debug port provides the following functionality:

- Full view of the module's configuration data
- View of the module's status data
- Complete display of the module's internal database (registers 0 to 4999)
- Version Information
- Control over the module (warm boot, cold boot, transfer configuration)
- Facility to upload and download the module's configuration file



## 5 Reference

### *In This Chapter*

❖ Product Specifications .....	93
❖ Functional Overview .....	96
❖ Cable Connections .....	115
❖ Status Data Definition .....	119
❖ Modbus Protocol Specification .....	120

### 5.1 Product Specifications

The MVI56-MNETR Modbus TCP/IP Client/Server Communication Module allows Rockwell Automation® ControlLogix® processors to interface easily with other Modbus TCP/IP compatible devices.

Compatible devices include Modicon PAC's as well as a wide variety of instruments and devices. This module uses a small I/O data area for data transfer between the module and the ControlLogix processor, making it ideal for ControlNet™ or Ethernet applications with the module in a remote rack. The module exchanges up to 5000-words of data between the processor and the Modbus TCP/IP network.

#### **5.1.1 General Specifications**

- Single Slot - 1756 backplane compatible
- The module is recognized as an Input/Output module and has access to processor memory for data transfer between processor and module
- Ladder Logic is used for data transfer between module and processor. Sample ladder file included.
- Configuration data obtained from configuration text file downloaded to module. Sample configuration file included
- This module uses a small I/O data area for 40 words data block transfer between the module and the ControlLogix processor, for applications with the module in a remote rack.

### 5.1.2 Hardware Specifications

Specification	Description
Backplane Current Load	800 mA @ 5 Vdc 3 mA @ 24 Vdc
Operating Temperature	32°F to 140°F (0°C to 60°C)
Storage Temperature	-40°F to 185°F (-40°C to 85°C)
Shock	30 g operational 50 g non-operational Vibration: 5 g from 10 Hz to 150 Hz
Relative Humidity	5% to 95% (without condensation)
LED Indicators	Module Status Backplane Transfer Status Application Status Serial Activity
<b>Application port (Ethernet)</b>	
Ethernet Port (Ethernet modules)	10/100 Base-T RJ45 Connector Link and activity LED indicators Electrical Isolation 1500 V rms at 50 Hz to 60 Hz for 60 s, applied as specified in section 5.3.2 of IEC 60950: 1991 Ethernet Broadcast Storm Resiliency = less than or equal to 5000 [ARP] frames-per-second and less than or equal to 5 minutes duration
Shipped with Unit	RJ45 to DB-9M cables for each port 6-foot RS-232 configuration cable
<b>Debug/Configuration port (CFG)</b>	
CFG Port (CFG)	RJ45 (DB-9M with supplied cable) No hardware handshaking

### **5.1.3 Functional Specifications**

- Support for the storage and transfer of up to 5000 registers to/from the ControlLogix processor's controller tags
- User-definable module memory usage
- 10/100 Base-T Ethernet compatible interface
- Configurable parameters for the client include
  - Minimum Command Delay
  - Pass-Through Mode
- The ControlLogix processor can be programmed to control the activity on the client by actively selecting commands from the command list to execute or issue commands directly from the ladder logic
- The module supports 10 servers for Modbus TCP/IP (Port 502) and 10 servers for MNET (Port 2000)
- One hundred commands are supported on each port
- A client configured as a virtual Modbus master device on the MVI56-MNETR module will actively issue Modbus TCP/IP commands to other nodes on the Modbus TCP/IP network
- The servers permit remote clients to interact with all data contained in the module. This data can be derived from other Modbus clients on the network through the client on the module or from the ControlLogix processor
- The module can be configured to pass write commands (functions 5, 6, 15 and 16) directly from the remote host to the processor
- Accepts Modbus function code commands of 1, 2, 3, 4, 5, 6, 15, and 16 from an attached Modbus TCP/IP client

## 5.2 Functional Overview

### 5.2.1 General Concepts

The following discussion explains several concepts that are important for understanding module operation.

#### About the MODBUS TCP/IP Protocol

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

The original MODBUS specification uses a serial connection to communicate commands and data between Client and server devices on a network. Later enhancements to the protocol allow communication over Ethernet networks using TCP/IP as a "wrapper" for the MODBUS protocol. This protocol is known as MODBUS TCP/IP.

MODBUS TCP/IP is a Client/server protocol. The Client establishes a connection to the remote server. When the connection is established, the Client sends the MODBUS TCP/IP commands to the server. The MVI56-MNETR module works both as a Client and as a server.

Aside from the benefits of Ethernet versus serial communications (including performance, distance, and flexibility) for industrial networks, the MODBUS TCP/IP protocol allows for remote administration and control of devices over a TCP/IP network. The efficiency, scalability, and low cost of a MODBUS TCP/IP network make this an ideal solution for industrial applications.

The MVI56-MNETR module acts as an input/output module between devices on a MODBUS TCP/IP network and the Rockwell Automation backplane. The module uses an internal database to pass data and commands between the processor and the Client and server devices on the MODBUS TCP/IP network.



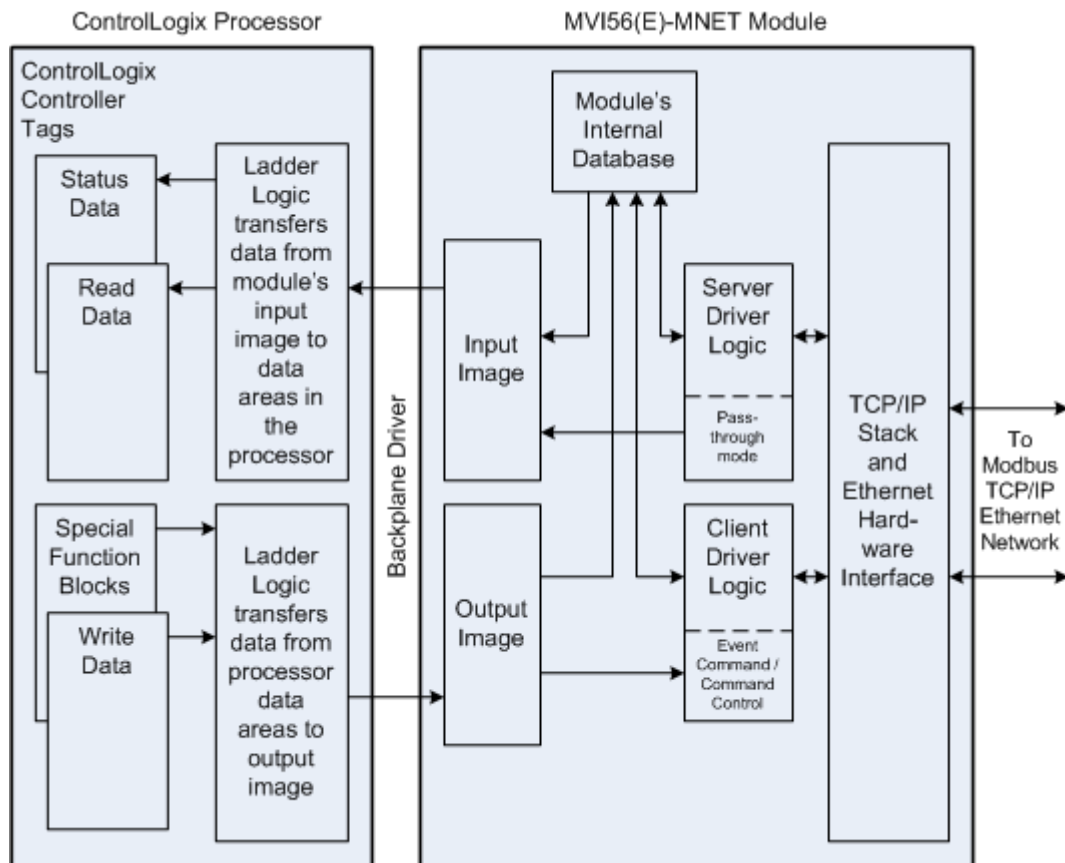
### Backplane Data Transfer

The MVI56-MNETR module communicates directly over the ControlLogix backplane. Data is paged between the module and the ControlLogix processor across the backplane using the module's input and output images. The update frequency of the images is determined by the scheduled scan rate defined by the user for the module and the communication load on the module. Typical updates are in the range of 1 to 10 milliseconds.

This bi-directional transference of data is accomplished by the module filling in data in the module's input image to send to the processor. Data in the input image is placed in the Controller Tags in the processor by the ladder logic. The input image for the module is set to 42 words. This data is transferred in the scheduled I/O timeslot.

The processor inserts data to the module's output image to transfer to the module. The module's program extracts the data and places it in the module's internal database. The output image for the module is set to 42 words. This data is transferred in the scheduled I/O timeslot.

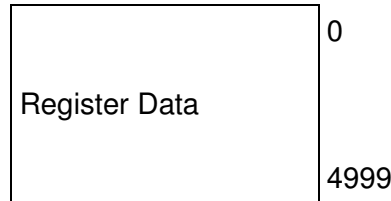
The following illustration shows the data transfer method used to move data between the ControlLogix processor, the MVI56-MNETR module and the Modbus TCP/IP Network.



All data transferred between the module and the processor over the backplane is through the input and output images. Ladder logic must be written in the ControlLogix processor to interface the input and output image data with data defined in the Controller Tags. All data used by the module is stored in its internal database. This database is defined as a virtual Modbus data table with addresses from 0 (40001 Modbus) to 4999 (45000 Modbus). The following illustration shows the layout of the database:

**Module’s Internal Database Structure**

5000 registers for user data



Data contained in this database is paged through the input and output images by coordination of the ControlLogix ladder logic and the MVI56-MNETR module’s program. Up to 40 words of data can be transferred from the module to the processor at a time. Up to 40 words of data can be transferred from the processor to the module. Each image has a defined structure depending on the data content and the function of the data transfer. The module uses the following block numbers:

<b>Block Range</b>	<b>Descriptions</b>
-1	Status block
0	Status block
1 to 125	Read or write data
1000 to 1124	Output Initialization Blocks
2000	Event Command Block
5001 to 5006	Command Control
9956	Formatted pass-through block from function 6 or 16 with word data.
9957	Formatted pass-through block from function 6 or 16 with floating-point data.
9958	Formatted pass-through block from function 5.
9959	Formatted pass-through block from function 15.
9960	Formatted pass-through block from function 22.
9961	Formatted pass-through block from function 23.
9970	Function 99 indication block.
9996	Unformatted Pass-through block with raw Modbus message.
9998	Warm-boot control block
9999	Cold-boot control block

These block identification codes can be broken down into a few groups: Normal data transfer blocks (-1 to 125), Initialization blocks (1000 to 1124), Command control blocks (2000, 5001 to 5006, 9998 and 9999) and pass-through function blocks (9956 to 9961, 9970 and 9996).

**Normal Data Transfer Blocks**

Normal data transfer includes the paging of user data from the module’s internal database (registers 0 to 4999), as well as paging of status data. These data are transferred through read (input image) and write (output image) blocks.

The following topics describe the function and structure of each block.

**Status Read Data Block**

This block is automatically copied from the module and contains status information about the module.

Offset	Description	Length
0	Write Block ID	1
1	Program Scan Counter	1
2 to 7	Block Transfer Status	6
8 to 9	Reserved Server Status	2
10 to 11	MNET Server Status	2
12 to 13	MBAP Server Status	2
14 to 23	MNET Client Status	10
24 to 25	Product Name	2
26	Product Version	1
27 to 40	Reserved	14
41	Read Block ID (-1 or 0)	1

Client Status Data

Word Offset	Client Status
3	Total number of command list requests
4	Total number of command list responses
5	Total number of command list errors
6	Total number of requests of slave
7	Total number of responses
8	Total number of errors sent
9	Total number of errors received
10	Configuration Error Word
11	Current Error
12	Last Error

**Block Response from Module to Processor**

These blocks of data transfer information from the module to the ControlLogix processor. The following table describes the structure of the input image.

Offset	Description	Length
0	Write Block ID	1
1 to 40	Read Data	40
41	Read Block ID	1

The Read Block ID is an index value used to determine the location of where the data will be placed in the ControlLogix processor controller tag array of module read data. Each transfer can move up to 40 words (block offsets 1 to 40) of data. In addition to moving user data, the block also contains status data for the module.

The Write Block ID associated with the block requests data from the ControlLogix processor. Under normal program operation, the module sequentially sends read blocks and requests write blocks.

For example, if the application uses three read and two write blocks, the sequence will be as follows:

R1W1→R2W2→R3W1→R1W2→R2W1→R3W2→R1W1→

This sequence will continue until interrupted by other write block numbers sent by the controller or by a command request from a node on the Modbus network or operator control through the module’s Configuration/Debug port.

**Block Request from Processor to Module**

These blocks of data transfer information from the ControlLogix processor to the module. The following table describes the structure of the output image.

Offset	Description	Length
0	Write Block ID	1
1 to 40	Write Data	40
41	Spare	1

The Write Block ID is an index value used to determine the location in the module’s database where the data will be placed. Each transfer can move up to 40 words (block offsets 1 to 40) of data.

### Initialize Output Data

When the module performs a restart operation, it will request blocks of output data from the processor to initialize the module's output data (Read Data Area). Use the **Initialize Output Data** parameter in the configuration file to bring the module to a known state after a restart operation. The following table describes the structure of the request block.

Offset	Description	Length
0	1000 to 1124	1
1 to 40	Spare	40
41	1000 to 1124	1

The block number in word 20 of the block determines the data set of up to 40 output words to transfer from the processor. Ladder logic in the processor must recognize these blocks and place the correct information in the output image to be returned to the module. The following table describes the structure of the response block.

Offset	Description	Length
0	1000 to 1124	1
1 to 40	Output Data to preset in module.	40
41	Spare	1

### Special Function Blocks

Special function blocks are optional blocks used to request special tasks from the module.

**Note:** Event Commands and Command Control are not needed for normal Modbus command list polling operations and are needed only occasionally for special circumstances.

**Important:** Each command defined in the command list is controlled by the ladder logic. The Write Command Bits parameter must be set in ladder logic to allow the command to be sent out on the Modbus TCP/IP network.

Event Command Block (2000)

Event command control blocks send Modbus TCP/IP commands directly from the ladder logic to one of the clients on the module. The following table describes the format of these blocks.

Offset	Description	Length
0	2000	1
1 to 4	IP Address	4
5	Service Port	1
6	Slave Address	1
7	Internal DB Address	1
8	Point Count	1
9	Swap Code	1
10	Modbus Function Code	1
11	Device Database Address	1
12 to 41	Spare	30

Use the parameters passed with the block to construct the command. The **IP Address** for the node to reach on the network is entered in four registers (1 to 4). Each digit of the IP address is entered in the appropriate register.

For example, to interface with node 192.168.0.100, enter the values 192, 168, 0 and 100 in registers 1 to 4. The **Service Port** field selects the TCP service port on the server to connect. If the parameter is set to 502, a standard MBAP message will be generated. All other service port values will generate a Modbus command message encapsulated in a TCP/IP packet.

The **Internal DB Address** parameter specifies the module's database location to associate with the command. The **Point Count** parameter defines the number of points or registers for the command. The **Swap Code** is used with Modbus functions 3 and 4 requests to change the word or byte order. The **Modbus Function Code** has one of the following values 1, 2, 3, 4, 5, 6, 15 or 16. The **Device Database Address** is the Modbus register or point in the remote slave device to be associated with the command.

When the module receives the block, it will process it and place it in the command queue. The following table describes the format of this block.

Word	Description
0	This word contains the block 2000 identification code to indicate that this block contains a command to execute by the Client Driver.
1 to 4	These words contain the IP address for the server the message is intended. Each digit (0 to 255) of the IP address is placed in one of the four registers. For example, to reach IP address 192.168.0.100, enter the following values in words 1 to 4 →192, 168, 0 and 100. The module will construct the normal dotted IP address from the values entered. The values entered will be anded with the mask 0x00ff to insure the values are in the range of 0 to 255.
5	This word contains the TCP service port the message will be interfaced. For example, to interface with a MBAP device, the word should contain a value of 502. To interface with a MNET device, a value of 2000 should be utilized. Any value from 0 to 65535 is permitted. A value of 502 will cause a MBAP formatted message to be generated. All other values will generate an encapsulated Modbus message.
6	This word contains the Modbus node address for the message. This field should have a value from 0 to 41.
7	This word contains the internal Modbus address in the module to use with the command. This word can contain a value from 0 to 4999.
8	This word contains the count parameter that determines the number of digital points or registers to associate with the command.
9	The parameter specifies the swap type for the data. This function is only valid for function codes 3 and 4.
10	This word contains the Modbus function code for the command.
11	This word contains the Modbus address in the slave device to be associated with the command.
12 to 41	Spare

The module will respond to each command block with a read block. The following table describes the format of this block.

Offset	Description	Length
0	Write Block ID	1
1	0=Fail, 1=Success	1
2 to 40	Spare	39
41	2000	1

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

Command Control Blocks (5001 to 5006)

Command control blocks place commands in the command list into the command queue. The client has a command queue of up to 100 commands. The module services commands in the queue before the user defined command list. This gives high priority to commands in the queue. Commands placed in the queue through this mechanism must be defined in the module's command list. Under normal command list execution, the module will only execute commands with the Enable parameter set to one or two. If the value is set to zero, the command is skipped. Commands may be placed in the command queue with an Enable parameter set to zero using this feature. These commands can then be executed using the command control blocks.

One to six commands can be placed in the command queue with a single request. The following table describes the format for this block.

Offset	Description	Length
0	5001 to 5006	1
1	Command index	1
2	Command index	1
3	Command index	1
4	Command index	1
5	Command index	1
6	Command index	1
7 to 41	Spare	35

The last digit in the block code defines the number of commands to process in the block. For example, a block code of 5003 contains 3 command indexes that are to be placed in the command queue. The Command index parameters in the block have a range of 0 to 99 and correspond to the module's command list entries.

The module responds to a command control block with a block containing the number of commands added to the command queue for the port. The following table describes the format for this block.

Offset	Description	Length
0	Write Block ID	1
1	Number of commands added to command queue	1
2 to 40	Spare	39
41	5001 to 5006	1



Block 9990: Set Module IP Address

**IP Set Request (Write Block)**

Offset	Description	Length
0	9990	1
1	First digit of dotted IP address	1
2	Second digit of dotted IP address	1
3	Third digit of dotted IP address	1
4	Last digit of dotted IP address	1
5 to 41	Reserved	36

**IP Set Response (Read Block)**

Offset	Description	Length
0	0	1
1	Write Block ID	1
2	First digit of dotted IP address	1
3	Second digit of dotted IP address	1
4	Third digit of dotted IP address	1
5	Last digit of dotted IP address	1
6 to 41	Spare data area	35

Block 9991: Get Module IP Address

**IP Get Request (Write Block)**

Offset	Description	Length
0	9991	1
1 to 41	Spare data area	40

**IP Get Response (Read Block)**

Offset	Description	Length
0	0	1
1	Write Block ID	1
2	First digit of dotted IP address	1
3	Second digit of dotted IP address	1
4	Third digit of dotted IP address	1
5	Last digit of dotted IP address	1
6 to 41	Spare data area	35

Warm Boot Block (9998)

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

Offset	Description	Length
0	9998	1
1 to 41	Spare	41

Cold Boot Block (9999)

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

Offset	Description	Length
0	9999	1
1 to 41	Spare	41

**Pass-Through Control Blocks**

If the module is set for pass-through operation by placing a value of 1 to 3 in the configuration file parameter **Pass-Through Mode**, the module will send special blocks to the module when a write request is received from a client. Any Modbus function 5, 6, 15 or 16 commands will be passed from the server to the processor using this block identification numbers 9956 to 9961, 9970 and 9996. Ladder logic must handle the receipt of these blocks and to place the enclosed data into the proper controller tags in the module.

There are two basic modes of operation when the pass-through feature is utilized: Unformatted (code 1) and Formatted (code 2 or 3). The unformatted mode will pass the message received on the server directly to the processor without any processing. The following table describes the format of the read block.

Unformatted

**Unformatted Pass-Through Command (Read Block)**

Offset	Description	Length
0	9996	1
1	Number of bytes in Modbus msg	1
2	Reserved (always 0)	1
3 to 40	Modbus message received	38
41	9996	1

The ladder logic should copy and parse the received message and control the processor as expected by the master device. The processor must respond to the pass-through control block with a write block. The following table describes the format of the write block.

**Unformatted Pass-Through Command (Write Block)**

Offset	Description	Length
0	9996	1
1 to 41	Spare	41

This informs the module that the command has been processed and can be cleared from the pass-through queue.

In formatted pass-through mode, the module processes the received write request and generates a special block dependent on the function received. There are two modes of operation when the formatted pass-through mode is selected. If code 2 is utilized (no swap), the data received in the message is presented in the order received by the module. If code 3 is utilized (swap mode), the bytes in the data area of the message will be swapped. This selection is applied to all received write requests. The block identification code used with the request depends on the Modbus function requested. Block 9956 passes word type data for functions 6 and 16. Block 9957 passes a floating-point message for functions 6 and 16. Block 9958 is utilized when Modbus function 5 data is received. Block 9959 is employed when function 15 is recognized. Block 9960 is used for function 22 and Block 9961 is used for function 23 requests. Block 9970 is utilized for function 99. The following tables describe the format for the read blocks.

Formatted

**Formatted Pass-Through Command Blocks (Read Block)**

Offset	Description	Length
0	9956, 9957, 9958, 9960 or 9961	1
1	Number of word registers in Modbus data set	1
2	Starting address for Modbus data set	1
3 to 40	Modbus data set	38
41	9956, 9957, 9958, 9960 or 9961	1

**Formatted Pass-Through Command Blocks (Read Block)**

Offset	Description	Length
0	9959	1
1	Number of word registers in Modbus data set	1
2	Starting word address for Modbus data set	1
3 to 21	Modbus data set	19
22 to 40	Bit mask for the data set. Each bit to be considered with the data set will have a value of 1 in the mask. Bits to ignore in the data set will have a value of 0 in the mask.	19
41	9959	1

**Formatted Pass-Through Command Blocks (Read Block)**

Offset	Description	Length
0	9970	1
1	1	1
2	0	1
3 to 40	Spare data area	38
41	9996	1

The ladder logic should copy and parse the received message and control the processor as expected by the master device. The processor must respond to the formatted pass-through control blocks with a write block. The following tables describe the format of the write blocks.

**Formatted Pass-Through Response (Write Block)**

Offset	Description	Length
0	9956, 9957, 9958, 9960 or 9961	1
1 to 41	Spare data area	41

**Formatted Pass-Through Response (Write Block)**

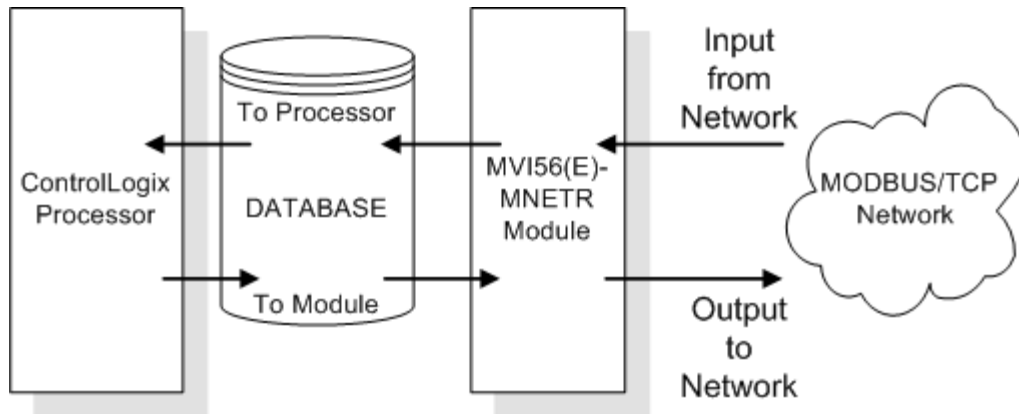
Offset	Description	Length
0	9959	1
1 to 41	Spare data area	41

**Formatted Pass-Through Response (Write Block)**

Offset	Description	Length
0	9970	1
1 to 41	Spare data area	41

**5.2.2 Data Flow between MVI56-MNETR Module and ControlLogix Processor**

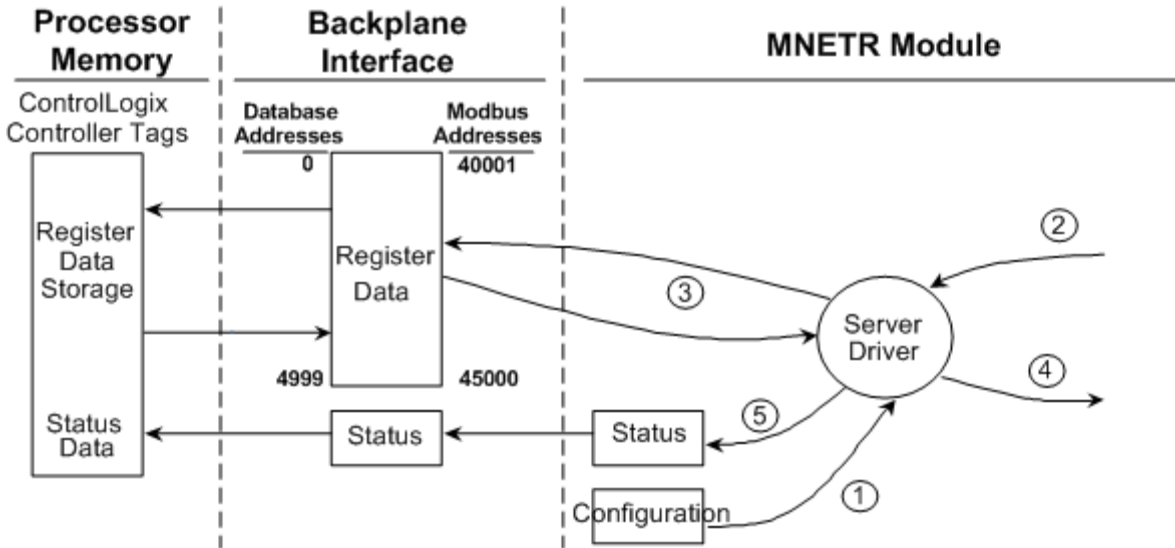
The following topics describe the flow of data between the two pieces of hardware (ControlLogix processor and MVI56-MNETR module) and other nodes on the Modbus TCP/IP network under the module's different operating modes. The module has both server and Client capability. The servers accept TCP/IP connections on service ports 502 (MBAP) (10 server connections) and 2000 (MNET) (10 server connections). The Client can generate either MBAP or MNET requests dependent on the service port selected in the command.



The following topics discuss the operation of the server and Client drivers.

Server Driver

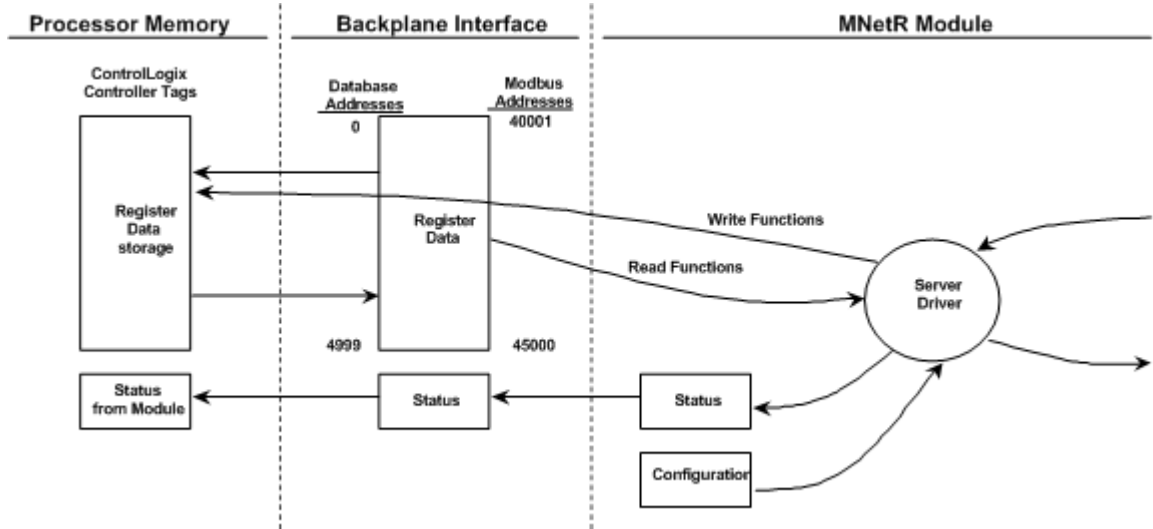
The Server Driver allows the MVI56-MNETR module to respond to data read and write commands issued by clients on the Modbus TCP/IP network. The following illustration and associated table describe the flow of data into and out of the module.



- 1 The server driver receives the configuration information from the configuration file on the Compact Flash Disk, and the module initializes the servers.
- 2 A Client device, such as a Modicon PLC or an HMI application, issues a read or write command to the module's node address. The server driver qualifies the message before accepting it into the module.
- 3 When the module accepts the command, the data is immediately transferred to or from the internal database in the module. If the command is a read command, the data is read out of the database and a response message is built. If the command is a write command, the data is written directly into the database and a response message is built. If the pass-through feature is utilized, the write message is transferred directly to the processor and is not written to the module's database.
- 4 When the data processing has been completed in Step 3, the response is issued to the originating Client node.
- 5 Counters are available in the Status Block that permit the ladder logic program to determine the level of activity of the Server Driver.

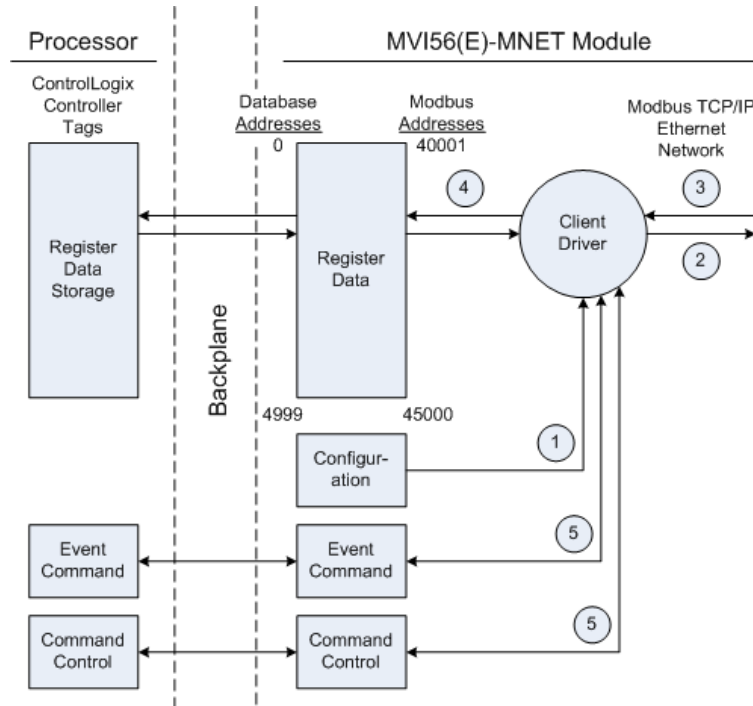
After the server socket is open, it must receive messages within a one minute period, or else it will close the socket. After closing, the socket will be reused.

An exception to this normal mode is when the pass-through mode is implemented. In this mode, all write requests will be passed directly to the processor and will not be placed in the database. This permits direct, remote control of the processor without the intermediate database. This mode is especially useful for Master devices that do not send both states of control. For example, a SCADA system may only send an on command to a digital control point and never send the clear state. The SCADA system expects the local logic to reset the control bit. Pass-through must be used to simulate this mode. The following illustration describes the data flow for a slave port with pass-through enabled:



Client Driver

In the Client driver, the MVI56-MNETR module issues read or write commands to servers on the Modbus TCP/IP network. These commands are user-configured in the module via the Client Command List received from the module's configuration or issued directly from the ControlLogix processor (Event Command). Command status is returned to the processor for each individual command in the command list status block. The location of this status block in the module's internal database is user-defined. The following flowchart describes the flow of data into and out of the module's Client driver.



- 1 The Client driver obtains configuration data when the module restarts. This includes the timeout parameters and the Command List. These values are used by the driver to determine the type of commands to be issued to servers on the Modbus TCP/IP network.
- 2 When configured, the Client driver begins transmitting read and/or write commands to servers on the network. The data for write commands is obtained from the module's internal database.
- 3 Assuming successful processing by the server specified in the command, a response message is received into the Client driver for processing.
- 4 Data received from the server is passed into the module's internal database, if the command was a read command. Status information is routinely returned to the processor in the input images.
- 5 Special functions, such as Event Commands and Command Control options, can be generated by the processor and sent to the Client driver for action.



### Client Command List

In order for the Client to function, the module's Client Command List must be defined. This list contains up to 100 individual entries, with each entry containing the information required to construct a valid command. This includes the following:

- Command enable mode
  - (0) disabled
  - (1) continuous
  - (2) conditional
- IP address and service port to connect to on the remote server
- Slave Node Address
- Command Type - Read or Write up to 100 words per command
- Database Source and Destination Register Address - Determines where data will be placed and/or obtained
- Count - Select the number of words to be transferred - 1 to 100
- Poll Delay - 1/10<sup>th</sup> seconds

### Client Command Errors

You can use the *MNET Client 0 Command Error Pointer* in the configuration to set the database offset register where all command error codes will be stored. This means that the first register refers to command 1 and so on.

Offset	Description
1	Command 1 Error
2	Command 2 Error
3	Command 3 Error
...	....
...	...

For every command that has an error, the module automatically sets the poll delay parameter to 30 seconds. This instructs the module to wait 30 seconds until it attempts to issue the command again.

As the list is read in from the configuration file and as the commands are processed, an error value is maintained in the module for each command. This error list can be transferred to the processor. The errors generated by the module are displayed in the following table.

Standard Modbus Exception Code Errors

Code	Description
1	Illegal function
2	Illegal data address
3	Illegal data value
4	Failure in associated device
5	Acknowledge
6	Busy; message was rejected

Module Communication Error Codes

Code	Description
-2	Timeout while transmitting message
-11	Timeout waiting for response after request
253	Incorrect slave/server address in response
254	Incorrect function code in response
255	Invalid CRC/LRC value in response

MNET Client Specific Errors

Code	Description
-33	Failed to connect to server specified in command
-36	MNET command response timeout
-37	TCP/IP connection ended before session finished

Command List Entry Errors

Code	Description
-40	Too few parameters
-41	Invalid enable code
-42	Internal address > maximum address
-43	Invalid node address (<0 or >255)
-44	Count parameter set to 0
-45	Invalid function code
-46	Invalid swap code
-47	ARP could not resolve MAC from IP (bad IP address, not part of a network, invalid parameter to ARP routine).
-48	Error during ARP operation: the response to the ARP request did not arrive to the module after a user-adjustable ARP Timeout.

**Note:** When the Client gets error -47 or -48, it uses the adjustable ARP Timeout parameter in the configuration file to set an amount of time to wait before trying again to connect to this non-existent server. This feature allows the Client to continue sending commands and polling other existing servers, while waiting for the non-existent server to appear on the network.

## 5.3 Cable Connections

The MVI56-MNETR module has the following functional communication connections installed:

- One Ethernet port (RJ45 connector)
- One RS-232 Configuration/Debug port (RJ45 connector)

### 5.3.1 Ethernet Connection

The MVI56-MNETR module has an RJ45 port located on the front of the module, labeled *Ethernet*, for use with the TCP/IP network. The module is connected to the Ethernet network using an Ethernet cable between the module's Ethernet port and an Ethernet switch or hub.

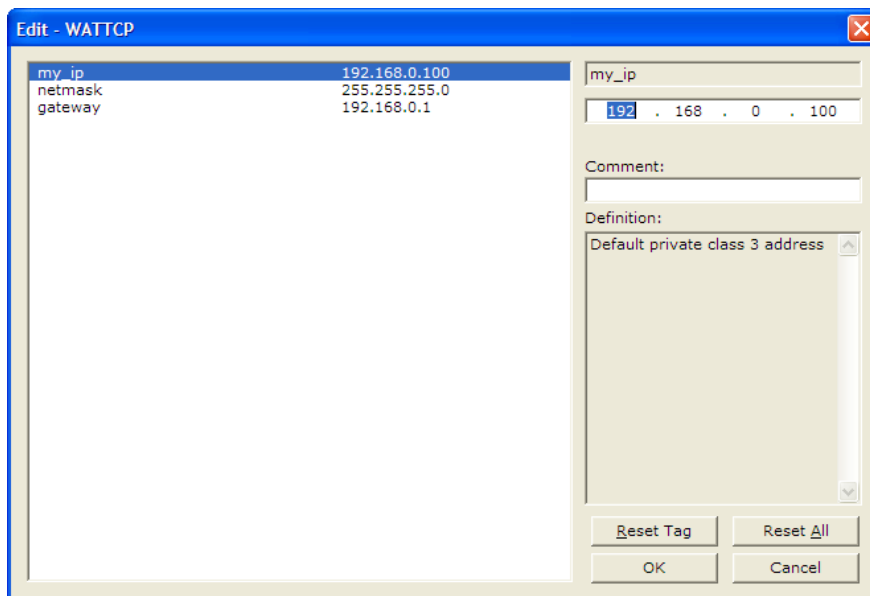
**Note:** Depending on hardware configuration, you may see more than one RJ45 port on the module. The Ethernet port is labeled *Ethernet*.

**Warning:** The MVI56-MNETR module is NOT compatible with Power Over Ethernet (IEEE802.3af / IEEE802.3at) networks. Do NOT connect the module to Ethernet devices, hubs, switches or networks that supply AC or DC power over the Ethernet cable. Failure to observe this precaution may result in damage to hardware, or injury to personnel.

**Important:** The module requires a static (fixed) IP address that is not shared with any other device on the Ethernet network. Obtain a list of suitable IP addresses from your network administrator BEFORE configuring the Ethernet port on this module.

#### Ethernet Port Configuration - wattcp.cfg

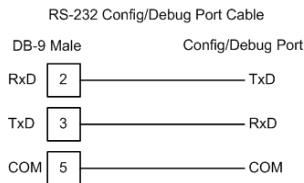
The wattcp.cfg file must be set up properly in order to use a TCP/IP network connection. You can view the current network configuration in *ProSoft Configuration Builder (PCB)*, as shown:



You may also view the network configuration using a PC serial port connection and an ASCII terminal program (like Windows HyperTerminal) by selecting [ @ ] (Network Menu) and [ V ] (View) options when connected to the Debug port. For more information on serial port access, see the chapter on Diagnostics and Troubleshooting (page 75).

### 5.3.2 RS-232 Configuration/Debug Port

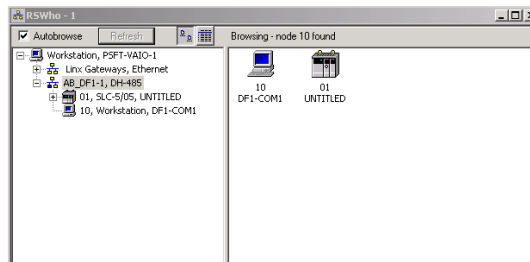
This port is physically an RJ45 connection. An RJ45 to DB-9 adapter cable is included with the module. This port permits a PC based terminal emulation program to view configuration and status data in the module and to control the module. The cable for communications on this port is shown in the following diagram:



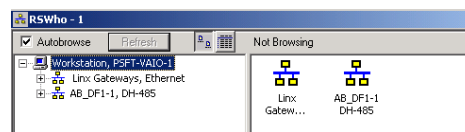
#### Disabling the RSLinx Driver for the Com Port on the PC



The communication port driver in *RSLinx* can occasionally prevent other applications from using the PC's COM port. If you are not able to connect to the module's configuration/debug port using *ProSoft Configuration Builder (PCB)*, *HyperTerminal* or another terminal emulator, follow these steps to disable the *RSLinx* Driver.

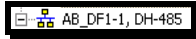
- 1 Open *RSLinx* and go to **COMMUNICATIONS>RSWHO**
- 2 Make sure that you are not actively browsing using the driver that you wish to stop. The following shows an actively browsed network:



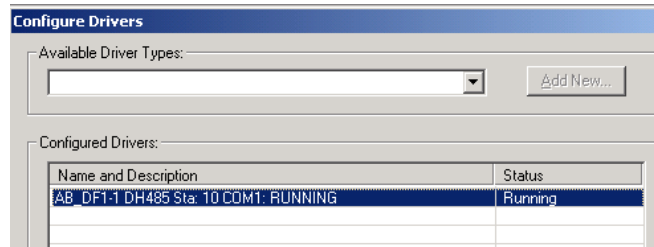
- 3 Notice how the DF1 driver is opened, and the driver is looking for a processor on node 1. If the network is being browsed, then you will not be able to stop this driver. To stop the driver your *RSWho* screen should look like this:



Branches are displayed or hidden by clicking on the  or the  icons.



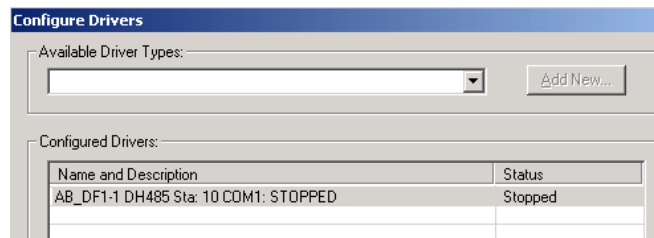
- 4 When you have verified that the driver is not being browsed, go to **COMMUNICATIONS>CONFIGURE DRIVERS**  
You may see something like this:



If you see the status as running, you will not be able to use this com port for anything other than communication to the processor. To stop the driver press the **STOP** button on the side of the window:



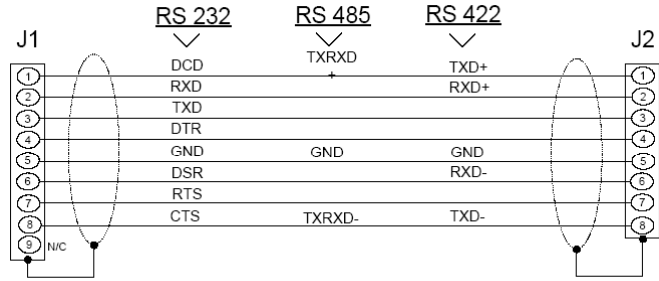
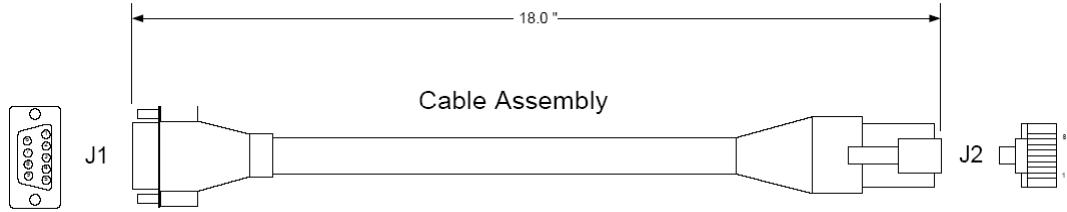
- 5 After you have stopped the driver you will see the following:



- 6 You may now use the com port to connect to the debug port of the module.

**Note:** You may need to shut down and restart your PC before it will allow you to stop the driver (usually only on *Windows NT* machines). If you have followed all of the above steps, and it will not stop the driver, then make sure you do not have *RSLogix* open. If *RSLogix* is not open, and you still cannot stop the driver, then reboot your PC.

### 5.3.3 DB9 to RJ45 Adaptor (Cable 14)



Wiring Diagram

## 5.4 Status Data Definition

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

Content	Description
Pass Count	This value is incremented each time a complete program cycle occurs in the module.
Product Version	Shows the module software version.
Product Code	Identifies the module product code.
Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
Write Block Count	This field contains the total number of write blocks transferred from the processor to the module.
Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
Command Event Block Count	This field contains the total number of command event blocks received from the processor.
Command Block Count	This field contains the total number of command blocks received from the processor.
Error Block Count	This field contains the total number of block errors recognized by the module.
Reserved1	Not used
Reserved2	Not used
MNet Request Count	This counter increments each time a MNet (port 2000) request is received.
MNet Response Count	This counter is incremented each time a MNet (port 2000) response message is sent.
MBAP Request Count	This counter increments each time a MBAP (port 502) request is received.
MBAP Response Count	This counter is incremented each time a MBAP (port 502) response message is sent.
Client Cmd Request	This value is incremented each time a command request is issued.
Client Cmd Response	This value is incremented each time a command response is received.
Client Cmd Error	This value is incremented each time an error message is received from a remote unit or a local error is generated for a command.
Client Request Count	This value is incremented each time a request message is issued.
Client Response Count	This value is incremented each time a response message is received.
Client Error Sent Count	This value is incremented each time an error is sent from the client.
Client Error Received Count	This value is incremented each time an error is received from a remote unit.
Client Cfg Error Word	This word contains a bit map that defines configuration errors in the configuration file for the client.
Client Current Error Code	This value corresponds to the current error code for the client.
Client Last Error Code	This value corresponds to the last error code recorded for the client.

## 5.5 Modbus Protocol Specification

The following pages give additional reference information regarding the Modbus protocol commands supported by the MVI56-MNETR.

### 5.5.1 Read Coil Status (Function Code 01)

#### Query

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

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

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

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

#### Response

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

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

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



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

### 5.5.2 Read Input Status (Function Code 02)

#### Query

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

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

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

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

#### Response

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

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

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

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

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

### 5.5.3 Read Holding Registers (Function Code 03)

#### Query

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

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

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

#### Response

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

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

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

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

### 5.5.4 Read Input Registers (Function Code 04)

#### Query

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

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

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

#### Response

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

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

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

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

### 5.5.5 Force Single Coil (Function Code 05)

#### Query

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

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

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

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

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

#### Response

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

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

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

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

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

### 5.5.6 Preset Single Register (Function Code 06)

#### Query

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

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

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

#### Response

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

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

### 5.5.7 Diagnostics (Function Code 08)

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

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

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

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

#### Sub-function Codes Supported

Only Sub-function 00 is supported by the MVI56-MNETR module.

#### **00 Return Query Data**

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

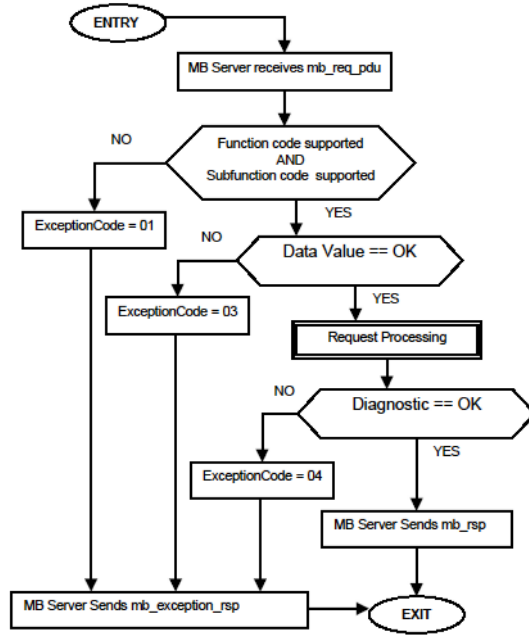
Sub-function	Data Field (Request)	Data Field (Response)
00 00	Any	Echo Request Data

#### **Example and State Diagram**

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

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

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





### 5.5.8 Force Multiple Coils (Function Code 15)

#### Query

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

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

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

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

#### Response

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

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

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

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

### 5.5.9 Preset Multiple Registers (Function Code 16)

#### Query

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

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

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

#### Response

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

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

### 5.5.10 Modbus Exception Responses

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

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

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

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

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

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

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

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

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

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

Modbus Exception Codes

Code	Name	Meaning
01	Illegal Function	The function code received in the query is not an allowable action for the Server. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the Server is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.
02	Illegal Data Address	The data address received in the query is not an allowable address for the Server. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed; a request with offset 96 and length 5 will generate exception 02.
03	Illegal Data Value	A value contained in the query data field is not an allowable value for Server. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program, because the Modbus protocol is unaware of the significance of any particular value of any particular register.
04	Slave Device Failure	An unrecoverable error occurred while the Server was attempting to perform the requested action.
05	Acknowledge	Specialized use in conjunction with programming commands. The Server has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the Client. The Client can next issue a poll program complete message to determine if processing is completed.
06	Slave Device Busy	Specialized use in conjunction with programming commands. The Server is engaged in processing a long-duration program command. The Client should retransmit the message later when the Server is free.
08	Memory Parity Error	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The Server attempted to read record file, but detected a parity error in the memory. The Client can retry the request, but service may be required on the Server device.
0a	Gateway Path Unavailable	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.

---

<b>Code</b>	<b>Name</b>	<b>Meaning</b>
0b	Gateway Target Device Failed To Respond	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

---



## 6 Support, Service & Warranty

### In This Chapter

- ❖ Contacting Technical Support ..... 135
- ❖ Return Material Authorization (RMA) Policies and Conditions..... 137
- ❖ LIMITED WARRANTY..... 139

### Contacting Technical Support

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

- 1 Product Version Number
- 2 System architecture
- 3 Network details

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

- 1 Module configuration and associated ladder files, if any
- 2 Module operation and any unusual behavior
- 3 Configuration/Debug status information
- 4 LED patterns
- 5 Details about the serial, Ethernet or fieldbus devices interfaced to the module, if any.

**Note:** For technical support calls within the United States, an after-hours answering system allows 24-hour/7-days-a-week pager access to one of our qualified Technical and/or Application Support Engineers.

<b>Internet</b>	Web Site: <a href="http://www.prosoft-technology.com/support">www.prosoft-technology.com/support</a> E-mail address: <a href="mailto:support@prosoft-technology.com">support@prosoft-technology.com</a>
<b>Asia Pacific</b> (location in Malaysia)	Tel: +603.7724.2080, E-mail: <a href="mailto:asiapc@prosoft-technology.com">asiapc@prosoft-technology.com</a> Languages spoken include: Chinese, English
<b>Asia Pacific</b> (location in China)	Tel: +86.21.5187.7337 x888, E-mail: <a href="mailto:asiapc@prosoft-technology.com">asiapc@prosoft-technology.com</a> Languages spoken include: Chinese, English

<b>Europe</b> (location in Toulouse, France)	Tel: +33 (0) 5.34.36.87.20, E-mail: support.EMEA@prosoft-technology.com Languages spoken include: French, English
<b>Europe</b> (location in Dubai, UAE)	Tel: +971-4-214-6911, E-mail: mea@prosoft-technology.com Languages spoken include: English, Hindi
<b>North America</b> (location in California)	Tel: +1.661.716.5100, E-mail: support@prosoft-technology.com Languages spoken include: English, Spanish
<b>Latin America</b> (Oficina Regional)	Tel: +1-281-2989109, E-Mail: latinam@prosoft-technology.com Languages spoken include: Spanish, English
<b>Latin America</b> (location in Puebla, Mexico)	Tel: +52-222-3-99-6565, E-mail: soporte@prosoft-technology.com Languages spoken include: Spanish
<b>Brasil</b> (location in Sao Paulo)	Tel: +55-11-5083-3776, E-mail: brasil@prosoft-technology.com Languages spoken include: Portuguese, English



## 6.1 Return Material Authorization (RMA) Policies and Conditions

The following Return Material Authorization (RMA) Policies and Conditions (collectively, "RMA Policies") apply to any returned product. These RMA Policies are subject to change by ProSoft Technology, Inc., without notice. For warranty information, see Limited Warranty (page 139). In the event of any inconsistency between the RMA Policies and the Warranty, the Warranty shall govern.

### 6.1.1 Returning Any Product

- a) In order to return a Product for repair, exchange, or otherwise, the Customer must obtain a Return Material Authorization (RMA) number from ProSoft Technology and comply with ProSoft Technology shipping instructions.
- b) In the event that the Customer experiences a problem with the Product for any reason, Customer should contact ProSoft Technical Support at one of the telephone numbers listed above (page 135). A Technical Support Engineer will request that you perform several tests in an attempt to isolate the problem. If after completing these tests, the Product is found to be the source of the problem, we will issue an RMA.
- c) All returned Products must be shipped freight prepaid, in the original shipping container or equivalent, to the location specified by ProSoft Technology, and be accompanied by proof of purchase and receipt date. The RMA number is to be prominently marked on the outside of the shipping box. Customer agrees to insure the Product or assume the risk of loss or damage in transit. Products shipped to ProSoft Technology using a shipment method other than that specified by ProSoft Technology, or shipped without an RMA number will be returned to the Customer, freight collect. Contact ProSoft Technical Support for further information.
- d) A 10% restocking fee applies to all warranty credit returns, whereby a Customer has an application change, ordered too many, does not need, etc. Returns for credit require that all accessory parts included in the original box (i.e.; antennas, cables) be returned. Failure to return these items will result in a deduction from the total credit due for each missing item.

### **6.1.2 Returning Units Under Warranty**

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

- a) A replacement module will be shipped and invoiced. A purchase order will be required.
- b) Credit for a product under warranty will be issued upon receipt of authorized product by ProSoft Technology at designated location referenced on the Return Material Authorization
  - i. If a defect is found and is determined to be customer generated, or if the defect is otherwise not covered by ProSoft Technology's warranty, there will be no credit given. Customer will be contacted and can request module be returned at their expense;
  - ii. If defect is customer generated and is repairable, customer can authorize ProSoft Technology to repair the unit by providing a purchase order for 30% of the current list price plus freight charges, duties and taxes as applicable.

### **6.1.3 Returning Units Out of Warranty**

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

**The following is a list of non-repairable units:**

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

## 6.2 LIMITED WARRANTY

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

### 6.2.1 What Is Covered By This Warranty

- a) *Warranty On New Products:* ProSoft warrants, to the original purchaser, that the Product that is the subject of the sale will (1) conform to and perform in accordance with published specifications prepared, approved and issued by ProSoft, and (2) will be free from defects in material or workmanship; provided these warranties only cover Product that is sold as new. This Warranty expires three (3) years from the date of shipment for Product purchased **on or after** January 1st, 2008, or one (1) year from the date of shipment for Product purchased **before** January 1st, 2008 (the "Warranty Period"). If the Customer discovers within the Warranty Period a failure of the Product to conform to specifications, or a defect in material or workmanship of the Product, the Customer must promptly notify ProSoft by fax, email or telephone. In no event may that notification be received by ProSoft later than 39 months from date of original shipment. Within a reasonable time after notification, ProSoft will correct any failure of the Product to conform to specifications or any defect in material or workmanship of the Product, with either new or remanufactured replacement parts. ProSoft reserves the right, and at its sole discretion, may replace unrepairable units with new or remanufactured equipment. All replacement units will be covered under warranty for the 3 year period commencing from the date of original equipment purchase, not the date of shipment of the replacement unit. Such repair, including both parts and labor, will be performed at ProSoft's expense. All warranty service will be performed at service centers designated by ProSoft.
- b) *Warranty On Services:* Materials and labor performed by ProSoft to repair a verified malfunction or defect are warranted 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.

### **6.2.2 What Is Not Covered By This Warranty**

- a) ProSoft makes no representation or warranty, expressed or implied, that the operation of software purchased from ProSoft will be uninterrupted or error free or that the functions contained in the software will meet or satisfy the purchaser's intended use or requirements; the Customer assumes complete responsibility for decisions made or actions taken based on information obtained using ProSoft software.
- b) This Warranty does not cover the failure of the Product to perform specified functions, or any other non-conformance, defects, losses or damages caused by or attributable to any of the following: (i) shipping; (ii) improper installation or other failure of Customer to adhere to ProSoft's specifications or instructions; (iii) unauthorized repair or maintenance; (iv) attachments, equipment, options, parts, software, or user-created programming (including, but not limited to, programs developed with any IEC 61131-3, "C" or any variant of "C" programming languages) not furnished by ProSoft; (v) use of the Product for purposes other than those for which it was designed; (vi) any other abuse, misapplication, neglect or misuse by the Customer; (vii) accident, improper testing or causes external to the Product such as, but not limited to, exposure to extremes of temperature or humidity, power failure or power surges; or (viii) disasters such as fire, flood, earthquake, wind and lightning.
- c) The information in this Agreement is subject to change without notice. ProSoft shall not be liable for technical or editorial errors or omissions made herein; nor for incidental or consequential damages resulting from the furnishing, performance or use of this material. The user guide included with your original product purchase from ProSoft contains information protected by copyright. No part of the guide may be duplicated or reproduced in any form without prior written consent from ProSoft.

### **6.2.3 Disclaimer Regarding High Risk Activities**

Product manufactured or supplied by ProSoft is not fault tolerant and is not designed, manufactured or intended for use in hazardous environments requiring fail-safe performance including and without limitation: the operation of nuclear facilities, aircraft navigation or 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.

#### **6.2.4 Intellectual Property Indemnity**

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

- a) Any documentation included with Product purchased from ProSoft is protected by copyright and may not be duplicated or reproduced in any form without prior written consent from ProSoft.
- b) ProSoft's technical specifications and documentation that are included with the Product are subject to editing and modification without notice.
- c) Transfer of title shall not operate to convey to Customer any right to make, or have made, any Product supplied by ProSoft.
- d) Customer is granted no right or license to use any software or other intellectual property in any manner or for any purpose not expressly permitted by any license agreement accompanying such software or other intellectual property.
- e) Customer agrees that it shall not, and shall not authorize others to, copy software provided by ProSoft (except as expressly permitted in any license agreement accompanying such software); transfer software to a third party separately from the Product; modify, alter, translate, decode, decompile, disassemble, reverse-engineer or otherwise attempt to derive the source code of the software or create derivative works based on the software; export the software or underlying technology in contravention of applicable US and international export laws and regulations; or use the software other than as authorized in connection with use of Product.
- f) **Additional Restrictions Relating To Software And Other Intellectual Property**

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

#### **6.2.5 Disclaimer of all Other Warranties**

The Warranty set forth in What Is Covered By This Warranty (page 139) 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.

### **6.2.6 Limitation of Remedies \*\***

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

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

### **6.2.7 Time Limit for Bringing Suit**

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

### **6.2.8 No Other Warranties**

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

### **6.2.9 Allocation of Risks**

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

### ***6.2.10 Controlling Law and Severability***

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





## Index

### O

00 Return Query Data • 127

### A

About the MODBUS TCP/IP Protocol • 96  
Adding Multiple Modules (Optional) • 30  
Adding the Module to an Existing Project • 72  
Adjusting the Input and Output Array Sizes (Optional) • 36  
Allocation of Risks • 142  
ARP Timeout • 49

### B

Backplane Data Transfer • 97  
Battery Life Advisory • 3  
Bit Input Offset • 57  
Block 9990  
    Set Module IP Address • 105  
Block 9991  
    Get Module IP Address • 105  
Block Request from Processor to Module • 100  
Block Response from Module to Processor • 100

### C

Cable Connections • 115  
Clearing a Fault Condition • 78  
Client Command Errors • 113  
Client Command List • 113  
Client Configuration Error Word • 77  
Client Driver • 112  
Client Status Data • 99  
Cold Boot Block (9999) • 106  
Command Control Blocks (5001 to 5006) • 104  
Command Entry Formats • 51  
Command Error Delay • 49  
Command Error Pointer • 47  
Command List Entry Errors • 114  
Command List Menu • 87  
Command List Overview • 50  
Commands Supported by the Module • 50  
Comment • 55  
Configuring Module Parameters • 43  
Configuring the MVI56-MNETR Module • 39  
Connecting Your PC to the ControlLogix Processor • 29  
Connecting your PC to the Module • 40  
Contacting Technical Support • 135, 137  
Controlling Law and Severability • 143  
Create the Module - Local Rack • 18, 23  
Create the Module - Remote Rack • 21  
Create the Remote Network • 19, 23

Creating a New RSLogix 5000 Project • 18  
Creating Optional Comment Entries • 44

### D

Data Flow between MVI56-MNETR Module and ControlLogix Processor • 109  
DB9 to RJ45 Adaptor (Cable 14) • 118  
Diagnostics (Function Code 08) • 127  
Diagnostics and Troubleshooting • 9, 75, 116  
Disabling the RSLinx Driver for the Com Port on the PC • 116  
Disclaimer of all Other Warranties • 141  
Disclaimer Regarding High Risk Activities • 140  
Downloading the Project to the Module Using a Serial COM port • 61  
Downloading the Sample Program to the Processor • 37, 74  
Duplex/Speed Code • 47

### E

Enable • 52  
Error/Status Pointer • 45, 47  
Ethernet Configuration • 60  
Ethernet Connection • 115  
Ethernet LED Indicators • 77  
Ethernet Port Configuration - wattcp.cfg • 115  
Event Command Block (2000) • 102  
Example and State Diagram • 127  
Exiting the Program • 85

### F

Failure Flag Count • 46  
Float Flag • 48, 56  
Float Offset • 49, 57  
Float Start • 48, 57  
Force Multiple Coils (Function Code 15) • 129  
Force Single Coil (Function Code 05) • 125  
Formatted • 108  
Functional Overview • 9, 96  
Functional Specifications • 95

### G

General Concepts • 96  
General Specifications • 93  
Guide to the MVI56-MNETR User Manual • 9

### H

Hardware MAC Address • 59  
Hardware Specifications • 94  
Holding Register Offset • 57  
How to Contact Us • 2

### I

Import Add-On Instruction • 26  
Important Installation Instructions • 3  
Initialize Output Data • 46, 101  
Installing ProSoft Configuration Builder Software • 14

Installing the Module in the Rack • 16  
Intellectual Property Indemnity • 141  
Internal Address • 52  
IP Address • 58

## K

Keystrokes • 82

## L

Ladder Logic • 63  
LED Indicators • 76  
Limitation of Remedies \*\* • 142  
LIMITED WARRANTY • 137, 139

## M

Main Menu • 83  
Markings • 4  
Master Command Error List Menu • 88  
MB Address in Device • 55  
Minimum Command Delay • 48  
MNET Client Specific Errors • 114  
MNET Client x • 47  
MNET Client x Commands • 49  
MNET Servers • 56  
MNETRBLOCKSTATS • 65  
MNETRCLIENTSTATS • 65, 66  
MNETRCMDCONTROL • 66, 68  
MNETRCONTROL • 64, 66  
MNETRDATA • 64  
MNETREVENTCMD • 66, 67  
MNETRMODULEDEF • 64  
MNETRPASSTHRU • 66, 69  
MNETRSTATUS • 64, 65  
MNETRUTIL • 64, 70  
Modbus Database View Menu • 83, 86  
Modbus Exception Codes • 132  
Modbus Exception Responses • 131  
Modbus Function • 54  
Modbus Message Data • 71  
Modbus Protocol Specification • 86, 120  
Module • 45  
Module Communication Error Codes • 114  
Moving Back Through 5 Pages of Registers • 86  
Moving Forward Through 5 Pages of Registers • 87  
MVI (Multi Vendor Interface) Modules • 3

## N

Navigation • 82  
Network Menu • 85, 89  
No Other Warranties • 142  
Node IP Address • 53, 54  
Normal Data Transfer Blocks • 99

## O

Opening the Command Error List Menu • 84  
Opening the Command List Menu • 84  
Opening the Database View Menu • 83  
Opening the Network Menu • 85

Output Offset • 57

## P

Package Contents • 13  
Pass-Through Control Blocks • 71, 106  
Pass-Through Mode • 47  
Pinouts • 3, 115, 118  
Poll Interval • 52  
Preset Multiple Registers (Function Code 16) • 130  
Preset Single Register (Function Code 06) • 126  
Printing a Configuration File • 44  
Product Specifications • 9, 93  
ProSoft Technology® Product Documentation • 2

## R

Read Coil Status (Function Code 01) • 120  
Read Holding Registers (Function Code 03) • 123  
Read Input Registers (Function Code 04) • 124  
Read Input Status (Function Code 02) • 122  
Read Register Count • 45  
Read Register Start • 45  
Reading Status Data from the Module • 91  
Receiving the Configuration File • 84  
Redisplaying the Current Page • 86  
Redisplaying the Menu • 88  
Reference • 9, 93  
Reg Count • 53  
Renaming PCB Objects • 43  
Resetting Diagnostic Data • 84  
Response Timeout • 48  
Retry Count • 48  
Return Material Authorization (RMA) Policies and Conditions • 137  
Returning Any Product • 137  
Returning to the Main Menu • 87, 88, 90  
Returning Units Out of Warranty • 138  
Returning Units Under Warranty • 138  
RS-232 Configuration/Debug Port • 116

## S

Sending the Configuration File • 84  
Server Driver • 110  
Service Port • 54  
Setting Jumpers • 15  
Setting Up the Project • 41  
Slave Address • 54  
Special Function Blocks • 101  
Standard Modbus Exception Code Errors • 114  
Start Here • 9, 11  
Static ARP Table • 58  
Status Data Definition • 119  
Status Read Data Block • 99  
Sub-function Codes Supported • 127  
Support, Service & Warranty • 9, 135  
Swap Code • 53  
System Requirements • 12

## T

Time Limit for Bringing Suit • 142  
Transferring WATTCP.CFG to the Module • 89  
Transferring WATTCP.CFG to the PC • 89  
Troubleshooting • 79

## U

Unformatted • 107  
Using ProSoft Configuration Builder • 41  
Using ProSoft Configuration Builder (PCB) for  
Diagnostics • 80  
Using the Diagnostic Window in ProSoft Configuration  
Builder • 80  
Using the Sample Program - RSLogix 5000 Version 15  
and earlier • 72

## V

Viewing Block Transfer Statistics • 83  
Viewing Client Configuration • 85  
Viewing Client Status • 85  
Viewing Data in ASCII (Text) Format • 87  
Viewing Data in Decimal Format • 9, 87  
Viewing Data in Floating-Point Format • 87  
Viewing Data in Hexadecimal Format • 87  
Viewing Module Configuration • 83  
Viewing Network Status • 84  
Viewing NIC Status • 85  
Viewing Register Pages • 86  
Viewing Server Configuration • 85  
Viewing the Next Page of Commands • 88  
Viewing the Next Page of Registers • 87  
Viewing the Previous Page of Commands • 88  
Viewing the Previous Page of Registers • 86  
Viewing the Static ARP Table • 85  
Viewing the WATTCP.CFG File on the module • 90  
Viewing Version Information • 84

## W

Warm Boot Block (9998) • 106  
Warm Booting the Module • 85  
Warnings • 3  
What Is Covered By This Warranty • 139, 141  
What Is Not Covered By This Warranty • 140  
Word Input Offset • 58  
Write Register Count • 46  
Write Register Start • 46

## Y

Your Feedback Please • 2