

Company: Joyoung International Trading Co., Limited

Atten: Smiling

Mobile/WhatsApp/Wechat: +86 18050035902

Email: info@htechplc.com

Website: <https://www.joyoungintl.com/>



Where Automation Connects.



inRAX[®]

MVI56-MNET

ControlLogix Platform

Modbus TCP/IP Interface Module

October 1, 2010

USER MANUAL

Your Feedback Please

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

How to Contact Us

ProSoft Technology

5201 Truxtun Ave., 3rd Floor

Bakersfield, CA 93309

+1 (661) 716-5100

+1 (661) 716-5101 (Fax)

www.prosoft-technology.com

support@prosoft-technology.com

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

MVI56-MNET User Manual

October 1, 2010

ProSoft Technology[®], ProLinx[®], inRAX[®], ProTalk[®], and RadioLinx[®] 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[®] Product Documentation

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

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

North America: +1.661.716.5100

Asia Pacific: +603.7724.2080

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

Latin America: +1.281.298.9109

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

Hardware Ratings

- Backplane Current Load: 800 mA @ 5 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

<cULus>

E183151

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

Temp Code T6

-30°C ≤ Ta ≤ 60°C

<Ex>

II 3 G

EEx nA IIC T6

0°C ≤ Ta ≤ 60°C

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

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

G – Equipment protected against explosive gasses.

Agency Approvals and Certifications

Agency	Applicable Standards
RoHS	
ATEX EN60	079-15:2003
CSA IEC610	10
CE EMC-EN61	326-1:2006 EN61000-6-4:2007
CSA CB Safety	CA/10533/CSA IEC 61010-1 Ed. 2 CB 243333-2056722 (2090408)
cULus	UL508, UL1604, CSA 22.2 No. 142 & 213
GOST-R Te	st 2.4
DNV	DET NORSKE VERITAS Test 2.4

RoHS



243



333



E183



151



ME06

Contents

Your Feedback Please.....	2
How to Contact Us	2
ProSoft Technology® Product Documentation	2
Warnings.....	3
Battery Life Advisory	3
Markings.....	4

Guide to the MVI56-MNET 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

2 Configuring the MVI56-MNET Module 19

2.1	Sample Add-On Instruction Import Procedure	19
2.1.1	Creating a New RSLogix 5000 Project.....	20
2.1.2	Creating the Module	20
2.1.3	Importing the Add-On Instruction	23
2.2	Connecting Your PC to the ControlLogix Processor	32
2.3	Downloading the Sample Program to the Processor	33
2.4	Using ProSoft Configuration Builder	34
2.4.1	Setting Up the Project	34
2.4.2	Setting Module Parameters.....	36
2.4.3	Module.....	38
2.4.4	MNET Client x	41
2.4.5	MNET Client x Commands.....	44
2.4.6	MNET Servers.....	51
2.4.7	Static ARP Table	54
2.4.8	Ethernet Configuration	55
2.5	Connecting your PC to the Module	56
2.6	Downloading the Project to the Module Using a Serial COM port	57

3 Ladder Logic 59

3.1	Controller Tags	59
3.1.1	MVI56(E)-MNET Controller Tags	60
3.2	User-Defined Data Types (UDTs)	61
3.2.1	MVI56(E)-MNET User-Defined Data Types	61
3.3	Using Controller Tags.....	62
3.4	Controller Tag Overview.....	63
3.4.1	MNET.DATA.....	63
3.4.2	MNET.STATUS	66
3.4.3	MNET.CONTROL.....	67

3.4.4	MNET.UTIL	68
4	Diagnostics and Troubleshooting	69
4.1	LED Indicators	70
4.1.1	Ethernet LED Indicators	70
4.1.2	Clearing a Fault Condition	71
4.1.3	Troubleshooting	71
4.2	Using ProSoft Configuration Builder (PCB) for Diagnostics	72
4.2.1	Using the Diagnostic Window in ProSoft Configuration Builder	72
4.2.2	Main Menu	75
4.2.3	Modbus Database View Menu	79
4.2.4	Command List Menu	81
4.2.5	Network Menu	82
4.3	Reading Status Data from the Module	84
4.4	Configuration Error Word	85
5	Reference	87
5.1	Product Specifications	87
5.1.1	General Specifications	87
5.1.2	Modbus TCP/IP	88
5.1.3	Functional Specifications	88
5.1.4	Hardware Specifications	89
5.2	About the MODBUS TCP/IP Protocol	90
5.3	Backplane Data Transfer	91
5.3.1	Normal Data Transfer Blocks	94
5.3.2	Special Function Blocks	100
5.4	Data Flow between the MVI56-MNET Module and ControlLogix Processor	111
5.4.1	Server Driver	112
5.4.2	Client Driver	114
5.5	Cable Connections	117
5.5.1	Ethernet Connection	117
5.5.2	RS-232 Configuration/Debug Port	118
5.5.3	DB9 to RJ45 Adaptor (Cable 14)	121
5.6	Adding the Module to an Existing Project	122
5.7	Using the Sample Program	125
5.7.1	Opening the Sample Program in RSLogix	125
5.7.2	Choosing the Controller Type	127
5.7.3	Selecting the Slot Number for the Module	128
5.7.4	Downloading the Sample Program to the Processor	129
5.7.5	Adding the Sample Ladder to an Existing Application	130
6	Support, Service & Warranty	131
	Contacting Technical Support	131
6.1	Return Material Authorization (RMA) Policies and Conditions	133
6.1.1	Returning Any Product	133
6.1.2	Returning Units Under Warranty	134
6.1.3	Returning Units Out of Warranty	134
6.2	LIMITED WARRANTY	135
6.2.1	What Is Covered By This Warranty	135
6.2.2	What Is Not Covered By This Warranty	136

6.2.3	Disclaimer Regarding High Risk Activities	136
6.2.4	Intellectual Property Indemnity	137
6.2.5	Disclaimer of all Other Warranties	137
6.2.6	Limitation of Remedies **	138
6.2.7	Time Limit for Bringing Suit	138
6.2.8	No Other Warranties	138
6.2.9	Allocation of Risks	138
6.2.10	Controlling Law and Severability	139

Index	141
--------------	------------

Guide to the MVI56-MNET 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 69)	This section describes Diagnostic and Troubleshooting procedures.
Reference Product Specifications Functional Overview	→	Reference (page 87) Product Specifications (page 87) Functional Overview (page 80)	These sections contain general references associated with this product, Specifications, and the Functional Overview.
Support, Service, and Warranty Index	→	Support, Service and Warranty (page 131) 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

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-MNET module’s application port(s)

1.1 System Requirements

The MVI56-MNET 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-MNET 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-MNET 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-MNET Module	MVI56-MNET	Modbus TCP/IP Interface Module
1 Cabl	e	Cable #15 - RS232 Null Modem	For RS232 between a Personal Computer (PC) and the CFG port of the module
1 Cabl	e	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-MNET 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

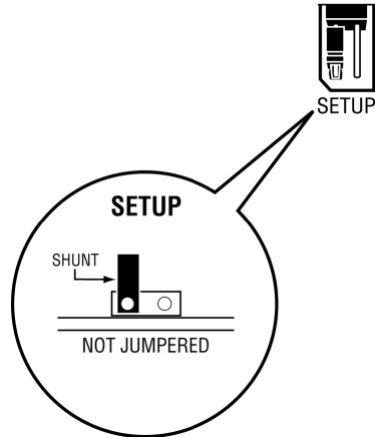
- 1 Insert the *ProSoft Solutions Product CD-ROM* into the CD-ROM drive of your PC. Wait for the startup screen to appear.
- 2 On the startup screen, click **PRODUCT DOCUMENTATION**. This action opens a *Windows Explorer* file tree window.
- 3 Click to open the **UTILITIES** folder. This folder contains 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-MNET 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

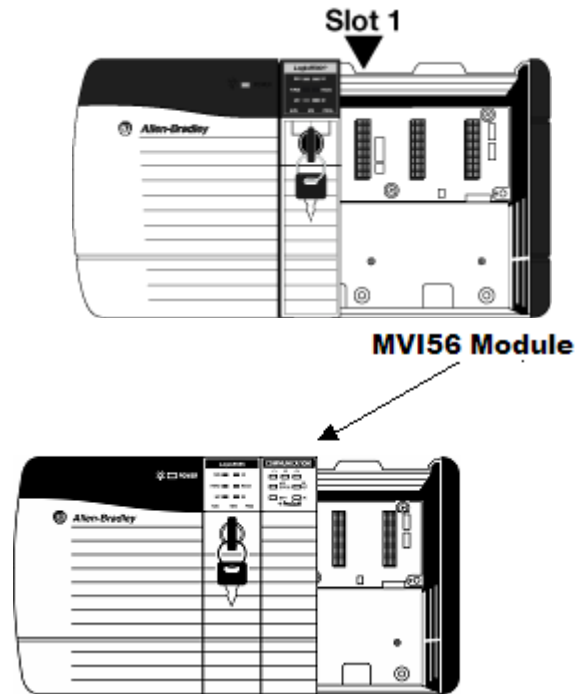
If you have not already installed and configured your ControlLogix processor and power supply, please do so before installing the MVI56-MNET module. Refer to your Rockwell Automation product documentation for installation instructions.

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-MNET 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.

2 Configuring the MVI56-MNET Module

In This Chapter

- ❖ Sample Add-On Instruction Import Procedure..... 19
- ❖ Connecting Your PC to the ControlLogix Processor..... 32
- ❖ Downloading the Sample Program to the Processor 33
- ❖ Using ProSoft Configuration Builder 34
- ❖ Connecting your PC to the Module..... 56
- ❖ Downloading the Project to the Module Using a Serial COM port 57

2.1 Sample Add-On Instruction Import Procedure

Note: This section only applies if your processor is using RSLogix 5000 version 16 or higher. If you have an earlier version, please see Using the Sample Program (page 125).

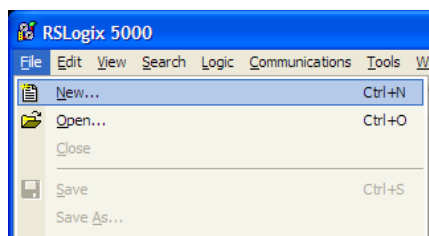
Before You Begin

The following file is required before you start this procedure. Copy the file from the *ProSoft Solutions CD-ROM*, or download it from www.prosoft-technology.com.

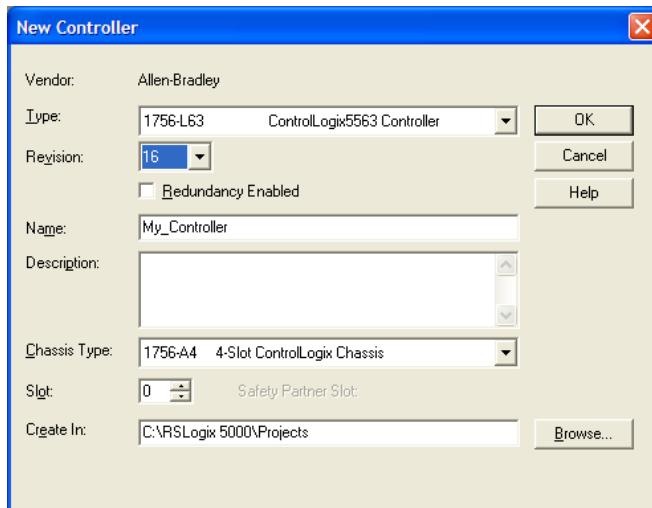
File Name	Description
MVI56(E)MNET_AddOn_Rung_v1_4.L5X	L5X file containing Add-On instruction, user defined data types, data objects and ladder logic required to set up the MVI56-MNET module

2.1.1 Creating a New RSLogix 5000 Project

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



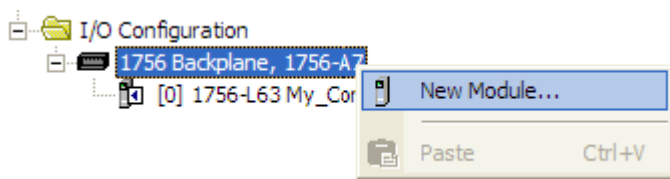
- 2 Select your ControlLogix controller model.
- 3 Select **REVISION 16**.
- 4 Enter a name for your controller, such as *My_Controller*.
- 5 Select your ControlLogix chassis type.
- 6 Select **SLOT 0** for the controller.



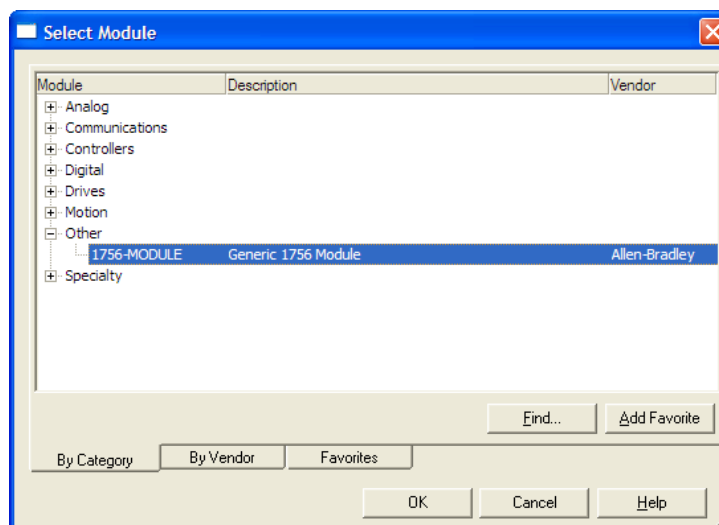
2.1.2 Creating the Module

- 1 Add the MVI56-MNET 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.



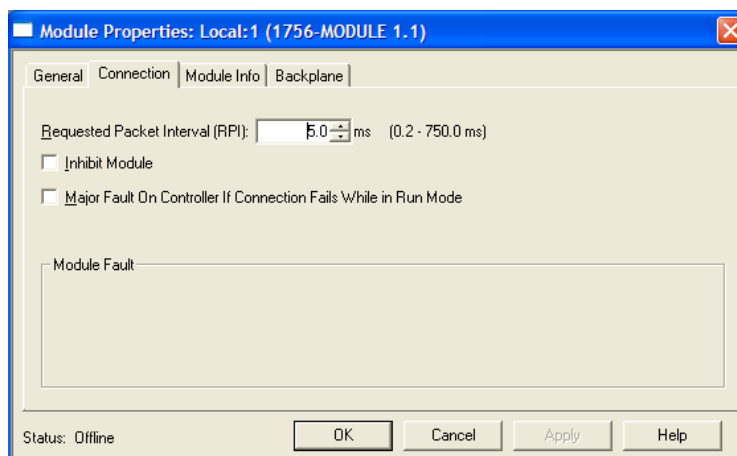
- 2 Select the **1756-MODULE (GENERIC 1756 MODULE)** from the list and click **OK**. This action opens the *New Module* dialog box.
- 3 In the *New Module* dialog box, enter the following values.

Parameter	Value
Name	Enter a module identification string. Example: MNET .
Description	Enter a description for the module. Example: MODBUS TCP/IP INTERFACE MODULE
Comm Format	Select DATA-INT .
Slot	Enter the slot number in the rack where the MVI56-MNET module is located.
Input Assembly Instance	1
Input Size	250
Output Assembly Instance	2
Output Size	248
Configuration Assembly Instance	4
Configuration Size	0

Important: 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.

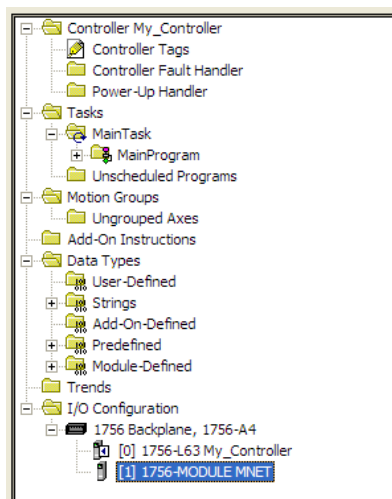
- 4 Click **OK** to continue.

- 5 Edit the Module Properties. Select the **REQUESTED PACKET INTERVAL** value for scanning the I/O on the module. This value represents the minimum frequency at which 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.



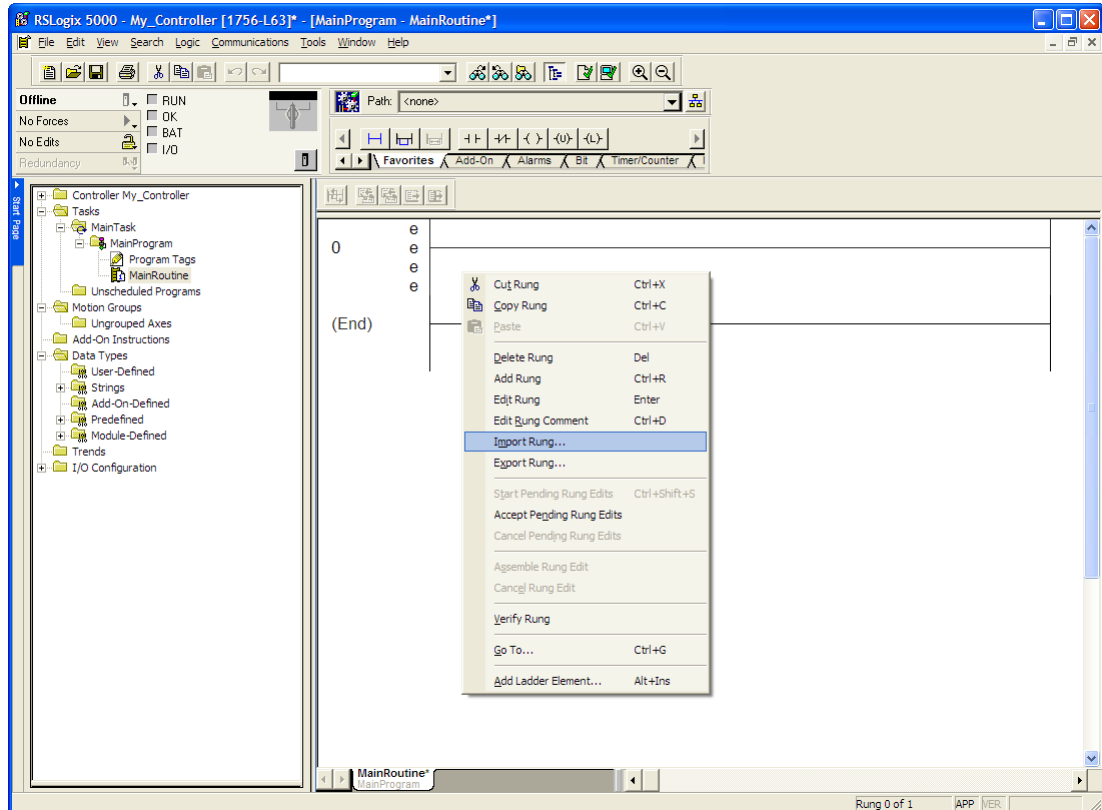
- 6 Save the module.

Click **OK** to close the dialog box. Notice that the module now appears in the *Controller Organization* window.

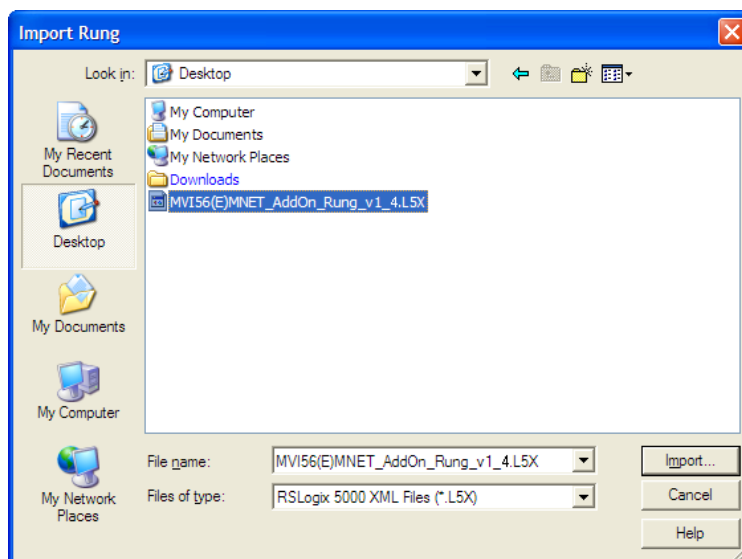


2.1.3 Importing the Add-On Instruction

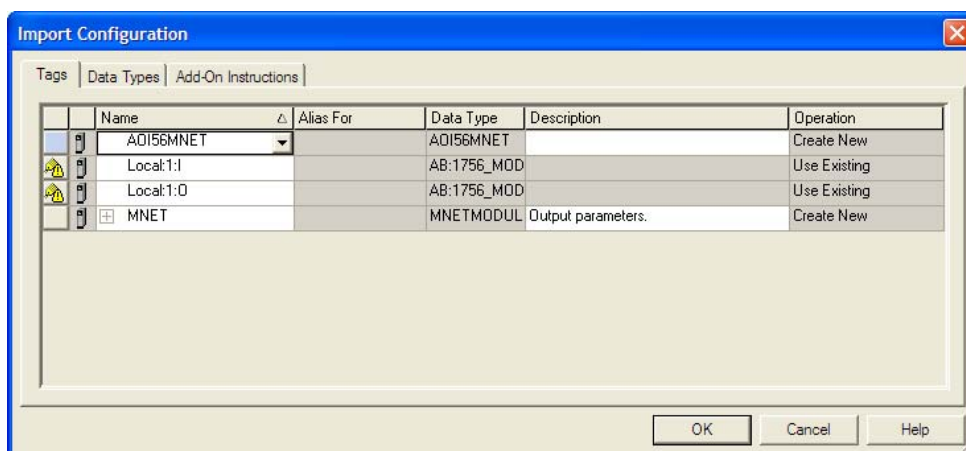
- 1 In the *Controller Organization* window, expand the *Tasks* folder and subfolder until you reach the *MainProgram* folder.
- 2 In the *MainProgram* folder, double-click to open the **MAINROUTINE** ladder.
- 3 Select an empty rung in the new routine, and then click the right mouse button to open a 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 **MVI56(E)MNET_AddOn_Rung_v1_4.L5X** file.

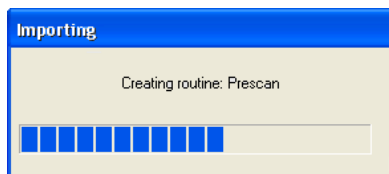


This action opens the *Import Configuration* dialog box, showing the controller tags that will be created.

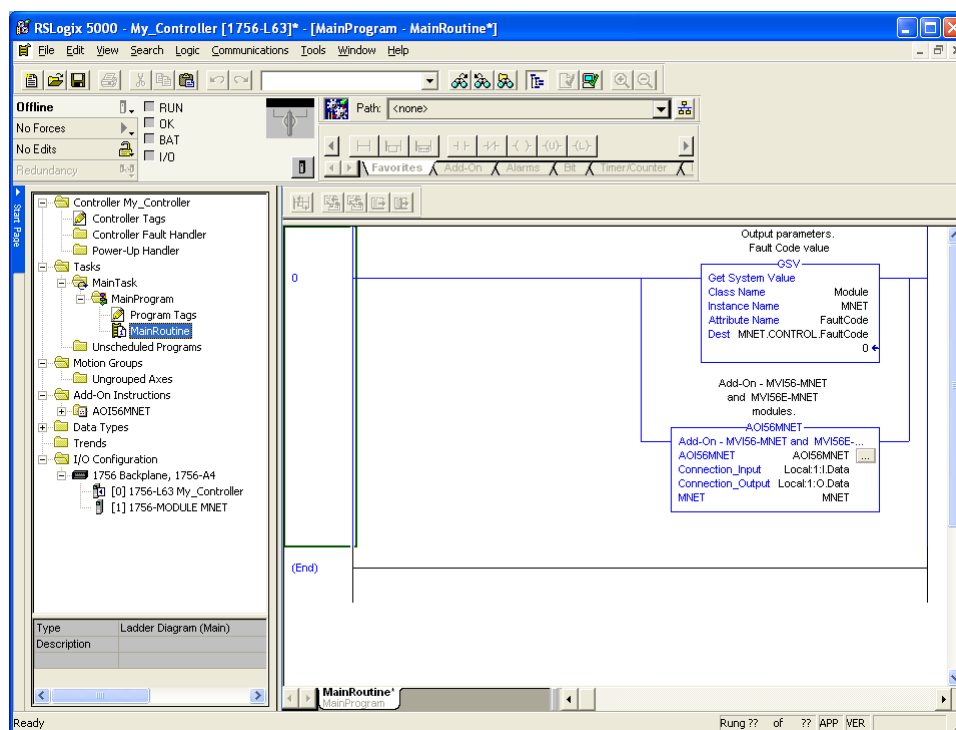


- 5 If you are using the module in a different slot (or remote rack), select the correct connection input and output variables that define the path to the module. If your module is located in Slot 1 of the local rack, this step is not required.

- 6 Click **OK** to confirm the import. RSLogix 5000 will indicate that the import is in progress:



When the import is completed, the new rung with the Add-On Instruction will be visible as shown in the following illustration.



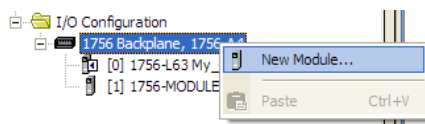
The procedure has also imported new user-defined data types, data objects and the Add-On instruction for your project.



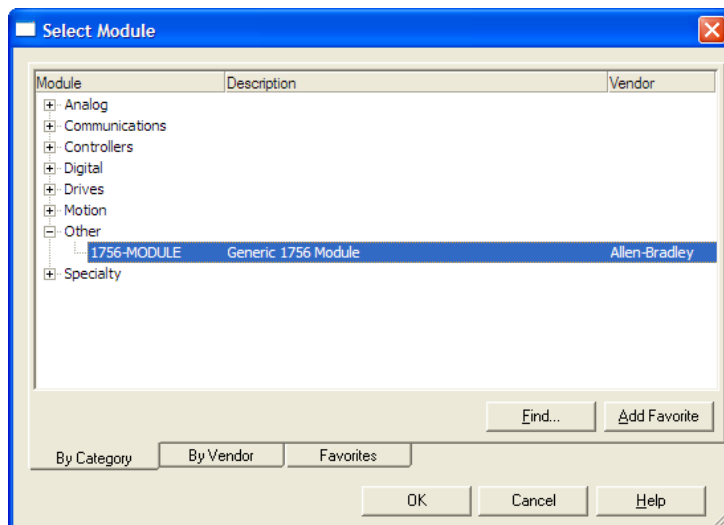
- 7 **SAVE** the application and then download the sample ladder logic into the processor.

Adding Multiple Modules (Optional)

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



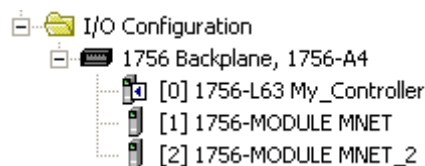
- 2 Select **1756-MODULE**.



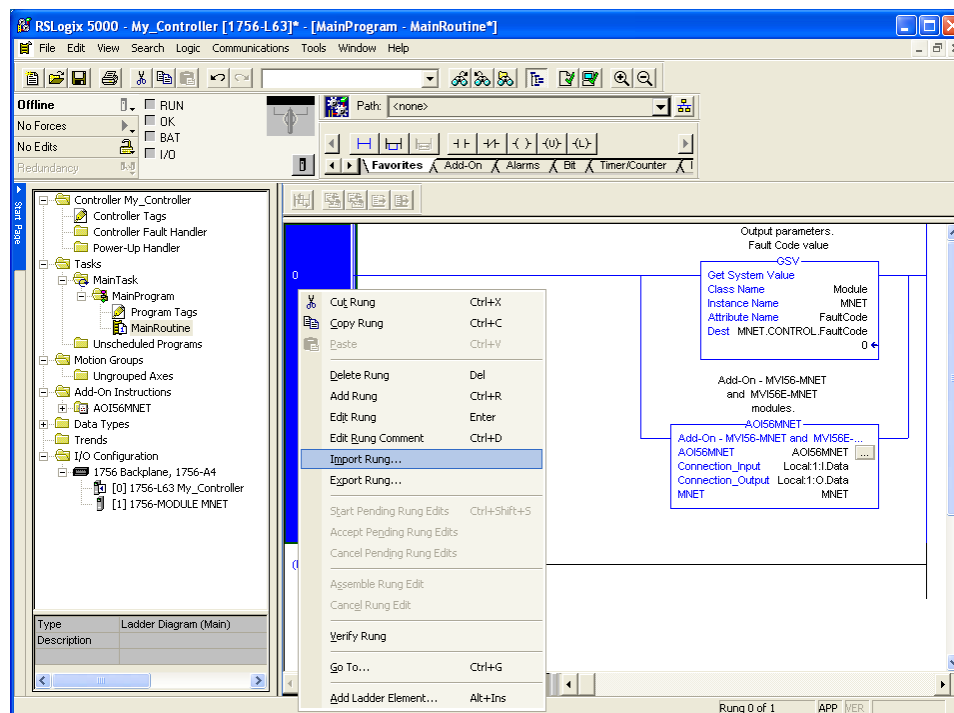
- 3 Fill the module properties as follows:

Parameter	Value
Name	Enter a module identification string. Example: MNET_2
Description	Enter a description for the module. Example: MODBUS TCP/IP INTERFACE MODULE
Comm Format	Select DATA-INT .
Slot	Enter the slot number in the rack where the MVI56-MNET module is located.
Input Assembly Instance	1
Input Size	250
Output Assembly Instance	2
Output Size	248
Configuration Assembly Instance	4
Configuration Size	0

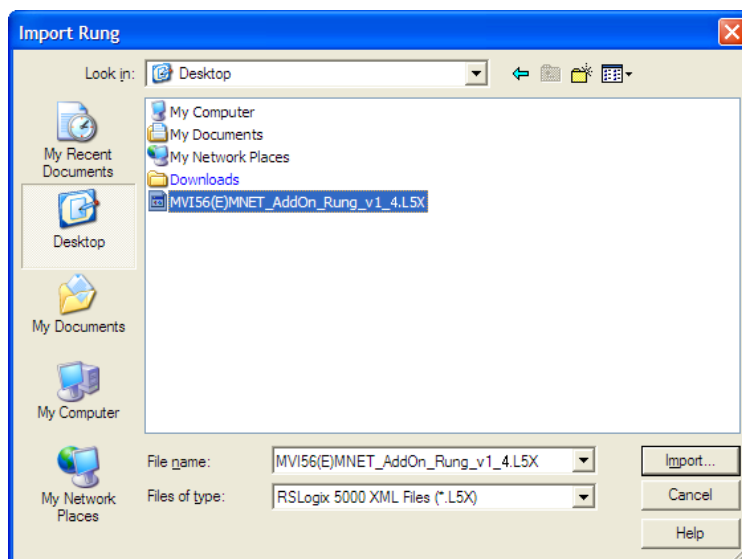
- 4 Click **OK** to confirm. The new module is now visible:



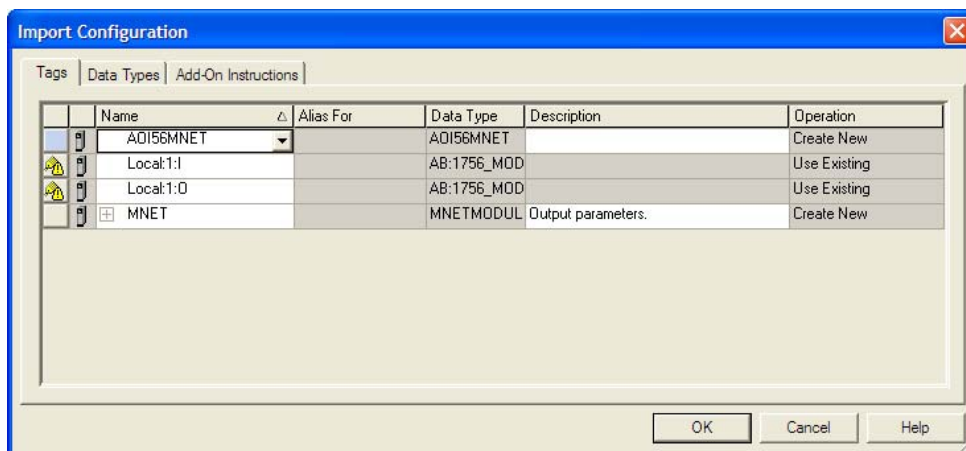
- 5 Expand the *Tasks* folder, and then expand the *MainTask* folder.
- 6 On the *MainProgram* folder, click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW ROUTINE**. As an alternative to creating a separate **NEW ROUTINE**, you could skip to Step 8 and import the AOI for the second module into the same routine you created for the first module.
- 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 or an existing routine, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT RUNG**.



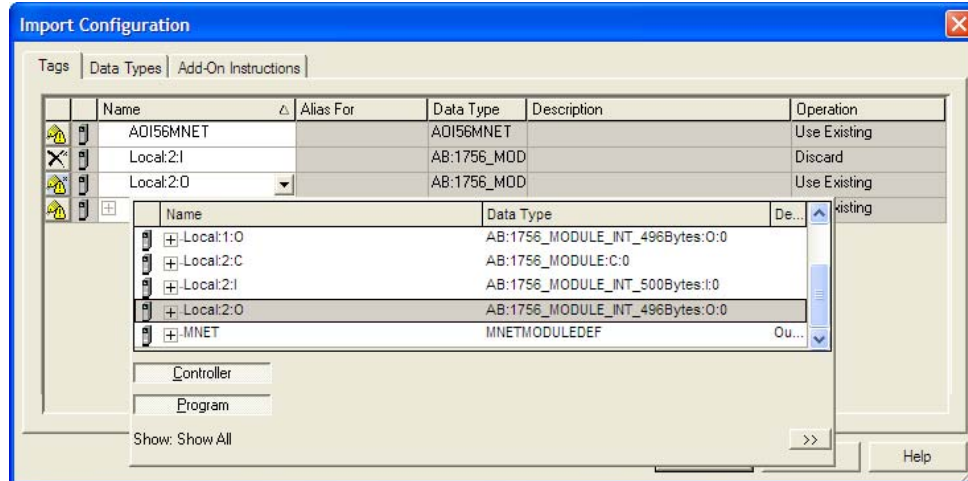
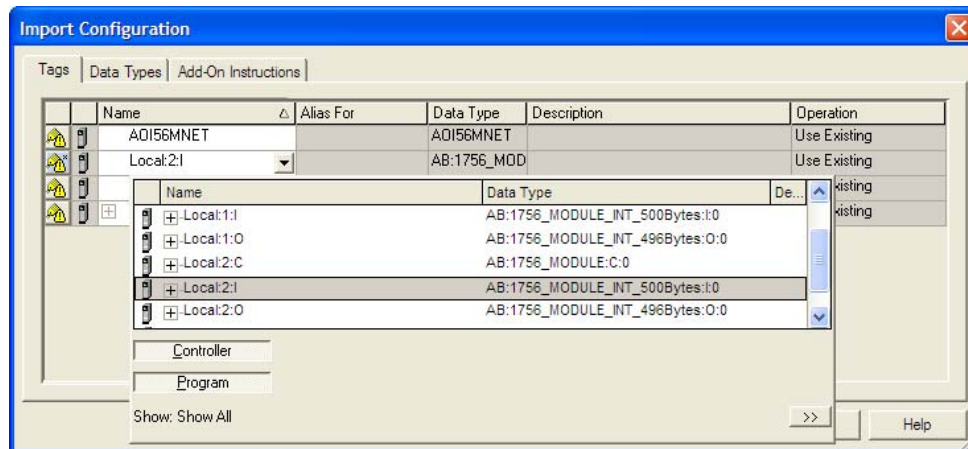
9 Select the file **MVI56(E)MNET_AddOn_Rung_v1_4.L5X**.



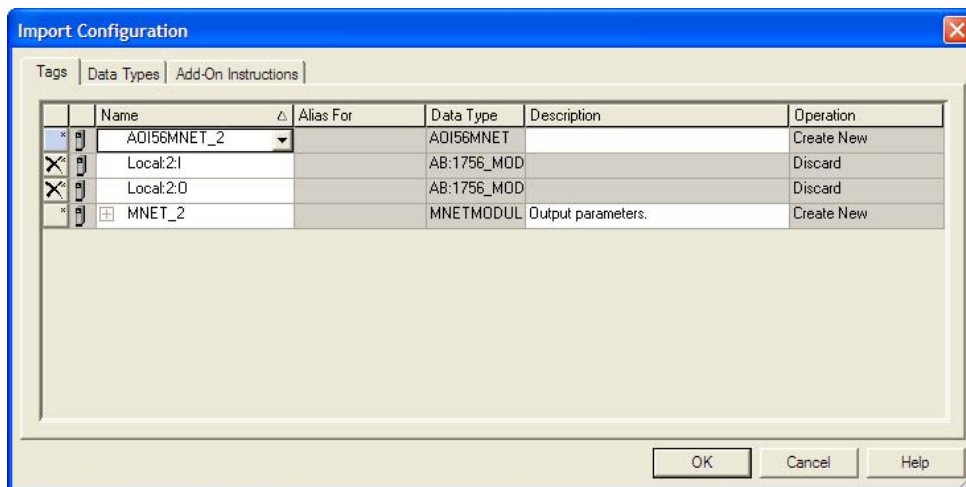
10 The following window will be displayed showing the tags to be imported:



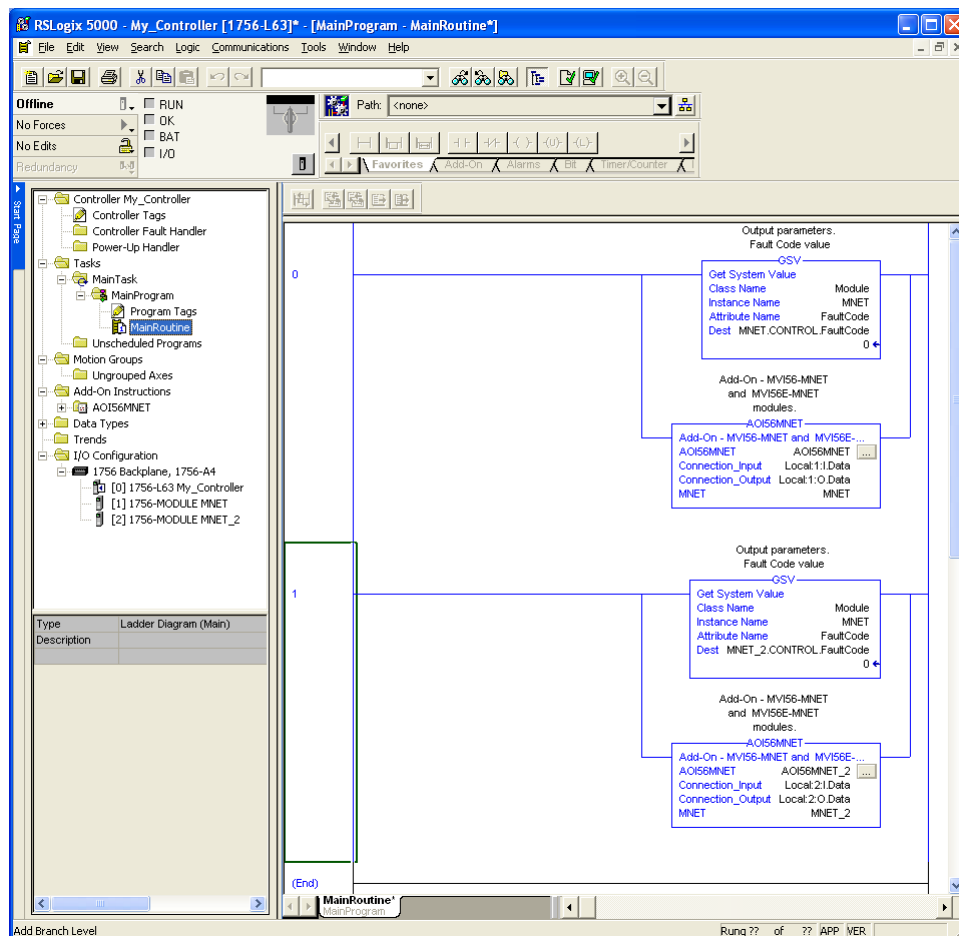
- 11 Associate the I/O connection variables to the correct module. The default values are **LOCAL:1:I** and **LOCAL:1:O** so these require change.



Change the default tags *MNET* and *AOI56MNET* to avoid conflict with existing tags. This procedure will append the string "_2" as follows:



12 Click **OK** to confirm.



Adjusting the Input and Output Array Sizes (Optional)

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

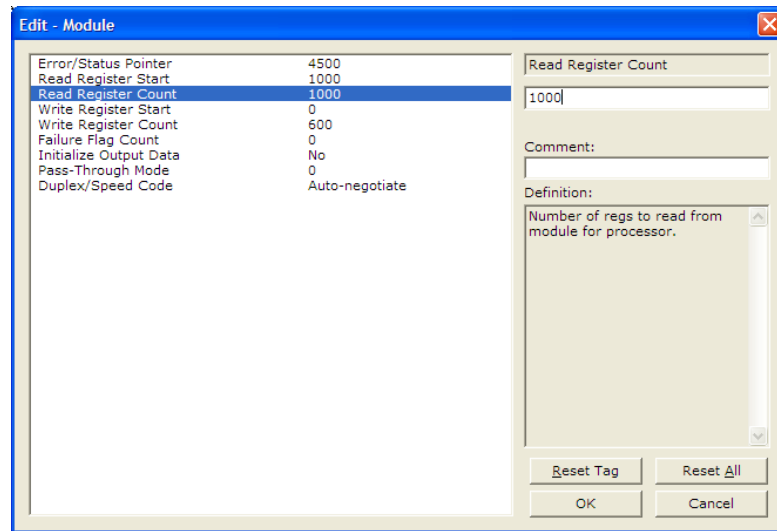
- Read Data
- Write Data

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-MNET 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*.
- 3 Save and download the configuration to the module (page 57) and reboot.

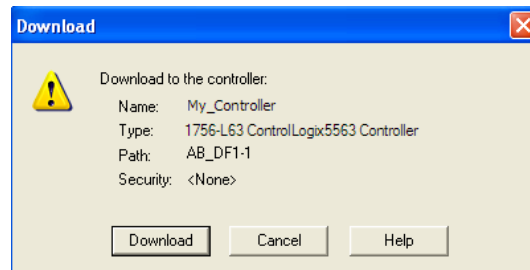
Make sure that the *ReadData* and *WriteData* arrays do not overlap in the module memory. For example, if your application requires 2000 words of *WriteData* starting at register 0, then your *Read Register Start* parameter must be set to a value of 2000 or greater.

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 from *ProSoft Configuration Builder*.

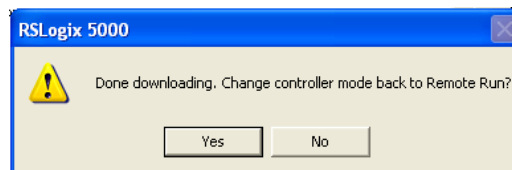
2.3 Downloading the Sample Program to the Processor

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.



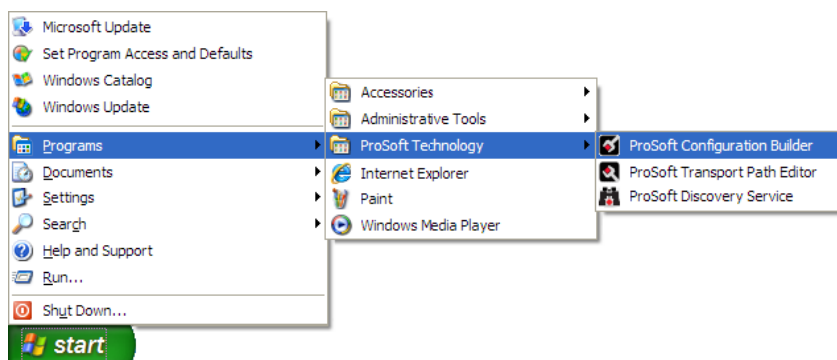
Note: If you receive an error message during these steps, refer to your RSLogix documentation to interpret and correct the error.

2.4 Using ProSoft Configuration Builder

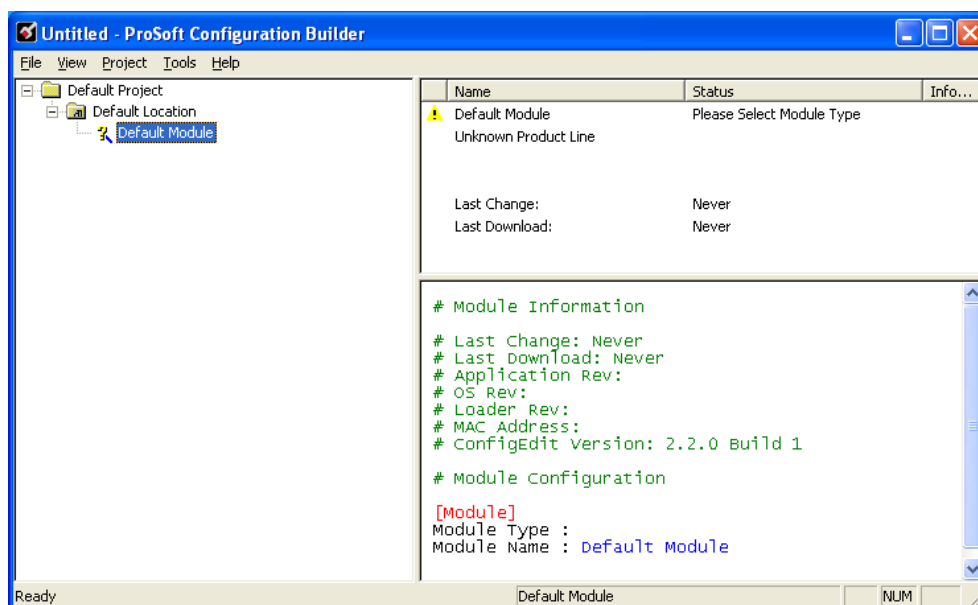
ProSoft Configuration Builder (PCB) provides a quick and easy way to manage module configuration files customized to meet your application needs. *PCB* is not only a powerful solution for new configuration files, but also allows you to import information from previously installed (known working) configurations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

2.4.1 Setting Up the Project

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

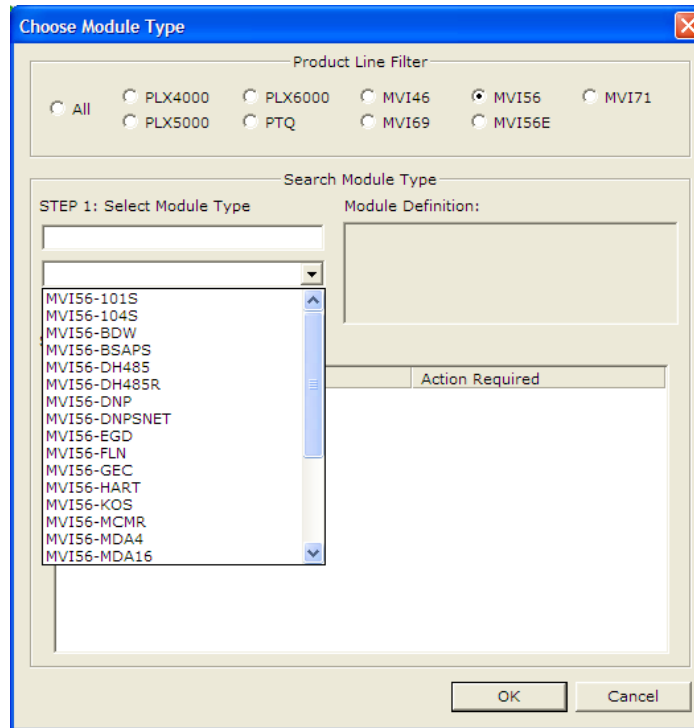


If you have used other Windows configuration tools before, you will find the screen layout familiar. *PCB*'s window consists of a tree view on the left, and an information pane and a configuration pane on the right side of the window. When you first start *PCB*, the tree view consists of folders for *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-MNET module to the project

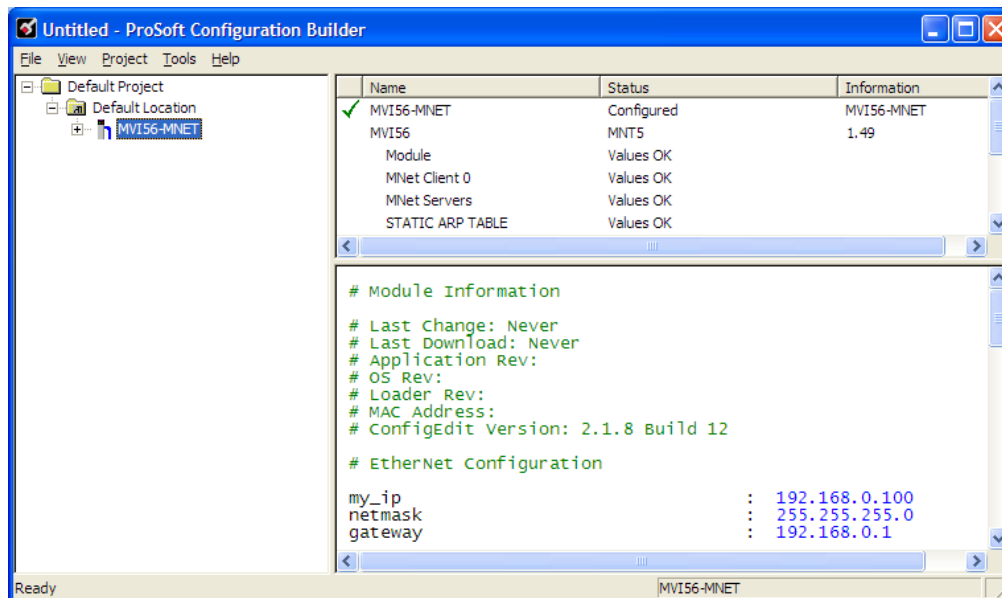
- 1 Use the mouse to select **DEFAULT MODULE** in the tree view, and then click the right mouse button to open a shortcut menu.
- 2 On the shortcut menu, choose **CHOOSE MODULE TYPE**. This action opens the *Choose Module Type* dialog box.



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

2.4.2 Setting Module Parameters

Notice that the contents of the information pane and the configuration pane changed when you added the MVI56-MNET module to the project.



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

Renaming an Object

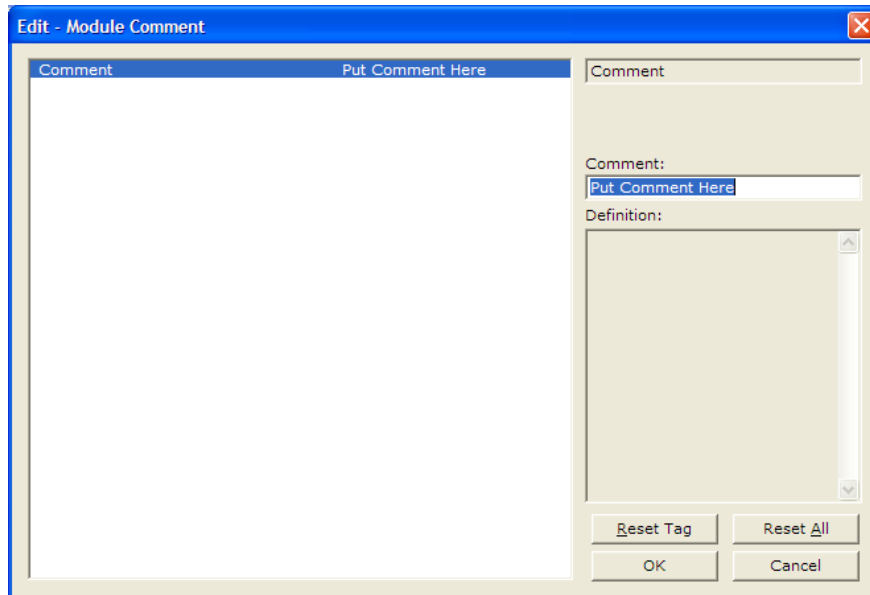
- 1 Select the object, and then click the right mouse button to open a shortcut menu. From the shortcut menu, choose **RENAME**.
- 2 Type the name to assign to the object.
- 3 Click away from the object to save the new name.

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**.

2.4.3 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

This parameter sets the address in the internal database where the error/status 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 error/status data will not be stored in the module's internal database and will not be transferred to the processor's *ReadData* array.

Enabling the error/status pointer is optional. The error/status data already exists as part of the Read Data block, which is continually being transferred from the module to the processor. For more information, see Read Block (page 94).

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 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.

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.

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.

2.4.4 MNET Client x

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

Error/Status Pointer

-1 to 4990

This parameter sets the address in the internal database where the error/status data for this Client 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 error/status data will not be stored in the module's internal database and will not be transferred to the processor's *ReadData* array.

Enabling the error/status pointer is optional. The error/status data already exists as part of the Read Data block, which is continually being transferred from the module to the processor. For more information, see Read Block (page 94).

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

If the remote Server 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 Client driver will send values from two consecutive 16-bit internal memory registers (32 total bits) for each count in a write command, or receive 32 bits per count from the Server for read commands. Example: Count = **10**, Client driver will send 20 16-bit registers for 10 total 32-bit floating-point values.

If, however, the remote Server expects to use a count of two (2) for each 32-bit floating-point value it sends or receives, 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

0 to 65535

Whenever the *Float Flag* parameter is set to **YES**, this parameter determines the lowest Modbus Address, used in commands to a remote Server, to consider as commands to read or write floating-point data. All commands with address values greater than or equal to this value will be considered floating-point data commands. All commands with address values less than this value will be considered normal 16-bit register data commands.

This parameter is used only if the *Float Flag* is set to **YES**. For example, if a value of 7000 is entered, all commands sent with addresses of 47001 (or 407001) and above will be considered as floating-point data commands and 32-bits of data will be sent or received for each count of one in the command.

You will also need to set the *Float Offset* parameter to an appropriate value whenever the *Float Flag* parameter is set to **YES**.

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.

2.4.5 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-MNET 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-MNET 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).

NOTE: If you are using only Event Commands or issuing commands from the Command List using Command Control from ladder logic, it is likely that the module will not leave any inactive TCP/IP socket connections open for more than 60-seconds. To maintain an open socket connection, your configuration or application must be designed so that at least one command is issued to each server connection at less than 60-second intervals. The 60-second connection timeout is not user-configurable and was put in place to prevent long delays between commands.

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	Diagnostic		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.

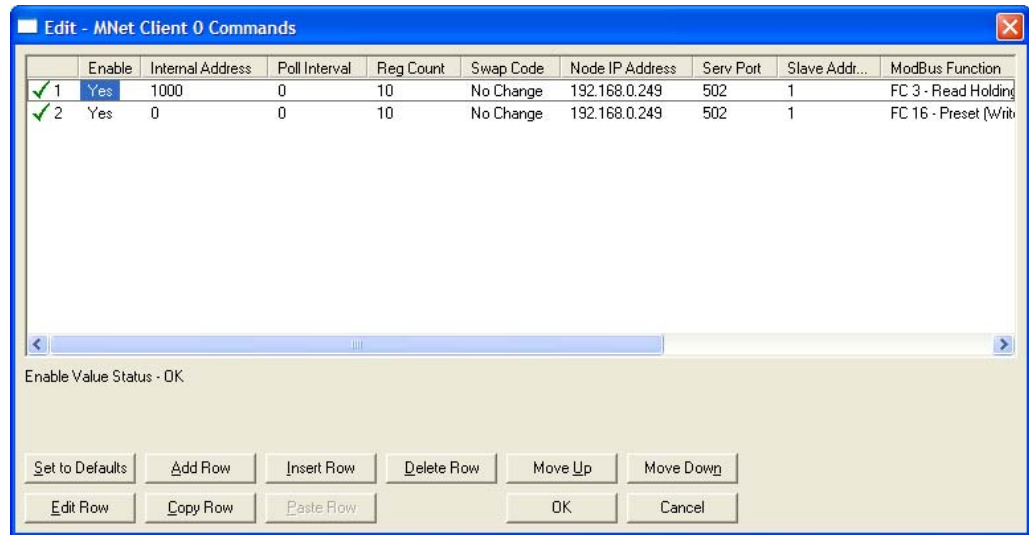
Command Entry Formats

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
No (0)	The command is disabled and will not be executed in the normal polling sequence.
YES (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.
CONDITIONAL (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.

Swap Code	Description
NONE	No change is made in the byte ordering (1234 = 1234)
SWAP WORDS	The words are swapped (1234=3412)
SWAP WORDS & BYTES	The words are swapped, then the bytes in each word are swapped (1234=4321)
SWAP BYTES	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 48) 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.

Modbus Function Code	Description
1	Read Coil Status
2	Read Input Status
3 Read	d 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.
($01101 - 00001 = 1100$)
($11101 - 10001 = 1100$)
($31101 - 30001 = 1100$)
($41101 - 40001 = 1100$)

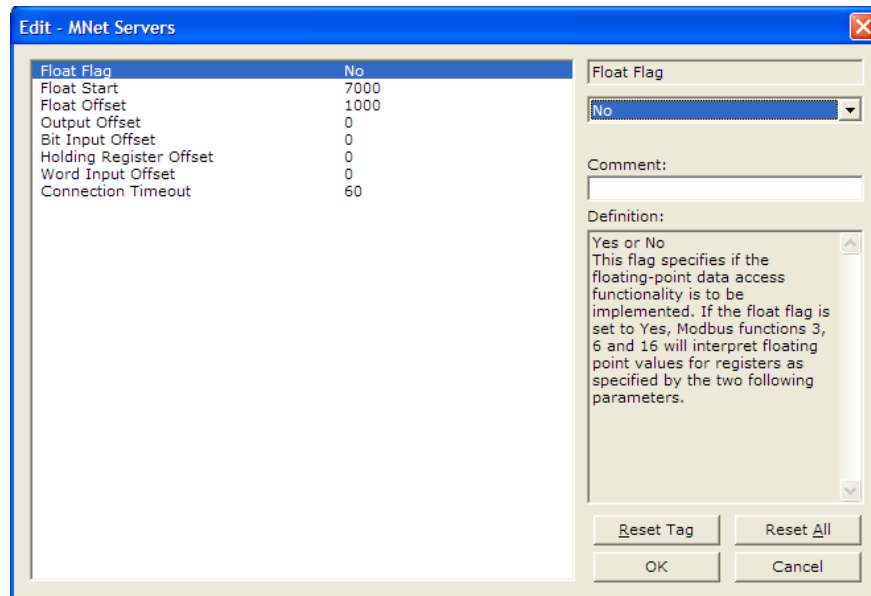
Note: If the documentation for a particular Modbus Server device lists data addresses in hexadecimal (base16) notation, you will need to convert the hexadecimal value to a decimal value to enter in this parameter. In such cases, it is not usually necessary to subtract 1 from the converted decimal number,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

Comment

0 to 35 alphanumeric characters

2.4.6 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

0 to 65535

Whenever the *Float Flag* parameter is set to **YES**, this parameter determines the lowest Modbus Address, received in commands from a remote Client, to consider as requests to read or write floating-point data. All commands with address values greater than or equal to this value will be considered floating-point data requests. All commands with address values less than this value will be considered normal 16-bit register data requests.

This parameter is used only if the *Float Flag* is set to **YES**. For example, if a value of 7000 is entered, all commands received with addresses of 47001 (or 407001) and above will be considered as requests for floating-point data and 32-bits of data will be returned for each count of one in the command.

You will also need to set the *Float Offset* parameter to an appropriate value whenever the *Float Flag* parameter is set to **YES**.

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.

Connection Timeout

0 to 1200 seconds

This is the number of seconds the server will wait to receive new data. If the server does not receive any new data during this time, it will close the connection.

2.4.7 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-MNET 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.

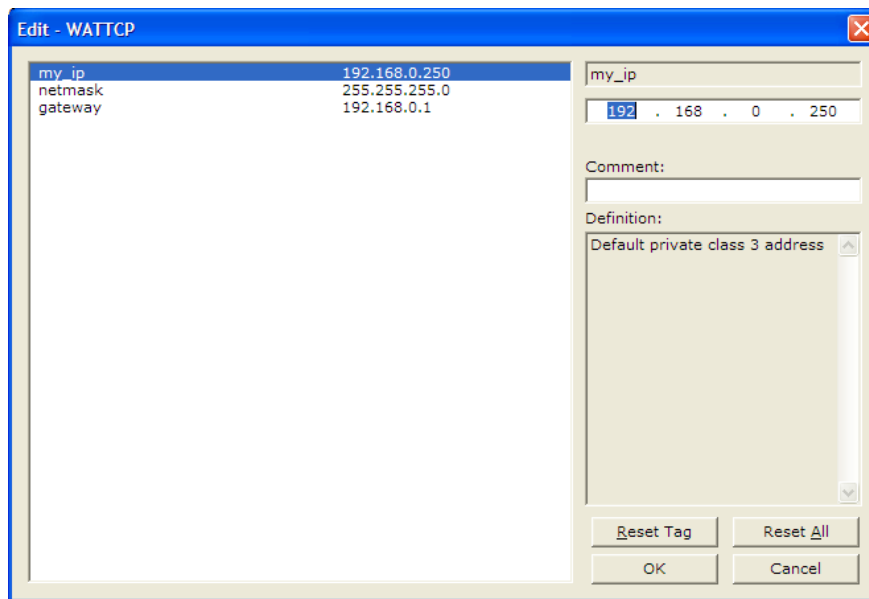
2.4.8 Ethernet Configuration

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:
 - o IP address (fixed IP required) _____ . _____ . _____ . _____
 - o Subnet mask _____ . _____ . _____ . _____
 - o 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.

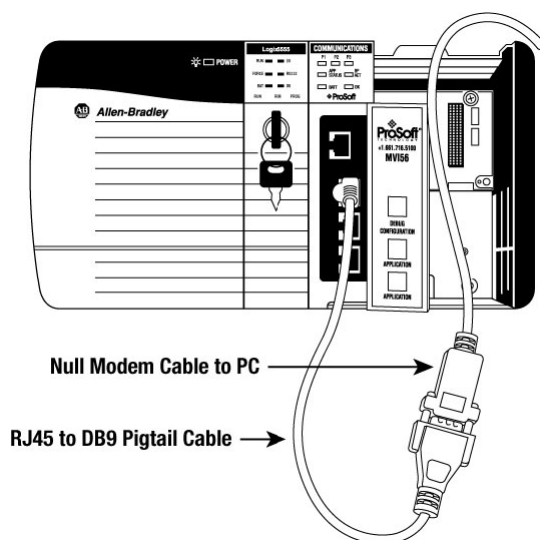


- 3 Edit the values for *my_ip*, *netmask* (subnet mask) and *gateway* (default gateway).
- 4 When you are finished editing, click **OK** to save your changes and return to the *ProSoft Configuration Builder* window.

2.5 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.

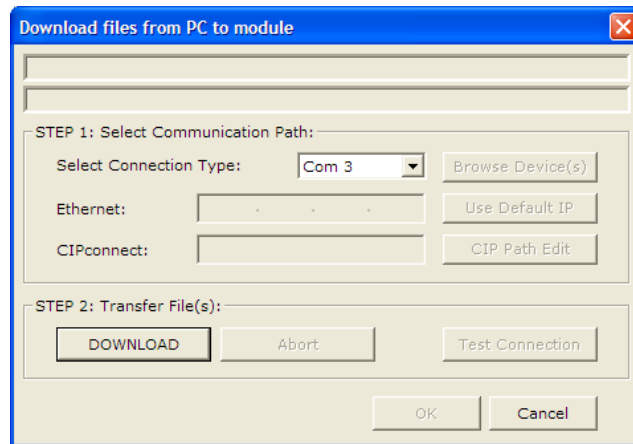
- 1 Attach both cables as shown.
- 2 Insert the RJ45 cable connector into the Configuration/Debug port of the module.
- 3 Attach the other end to the serial port on your PC.



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

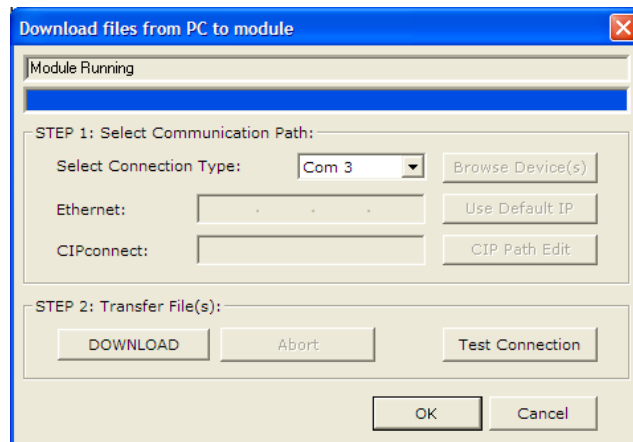
For the module to use the settings you configured, you must download (copy) the updated *Project* file from your PC to the module.

- 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

❖ Controller Tags	59
❖ User-Defined Data Types (UDTs)	61
❖ Using Controller Tags.....	62
❖ Controller Tag Overview.....	63

Ladder logic is required for managing communication between the MVI56-MNET module and the processor. The ladder logic handles tasks such as:

- Module backplane data transfer
- Special block handling
- Status data receipt

Additionally, a power-up handler may be needed to initialize the module's database and may clear some processor fault conditions.

The sample Import Rung with Add-On Instruction is extensively commented to provide information on the purpose and function of each user-defined data type and controller tag. For most applications, the Import Rung with Add-On Instruction will work without modification.

3.1 Controller Tags

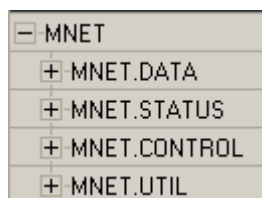
Data related to the MVI56-MNET is stored in variables or variable groupings called controller tags. The controller tags for the module are pre-programmed into the Add-On Instruction Import Rung ladder logic. You can find them in the *Controller Tags* subfolder, located in the *Controller* folder in the *Controller Organizer* pane of the main RSLogix 5000 window.

The controller tags are arranged in a tree structure, with groupings of related controller tags assembled under aggregate controller tags.

Individual controller tags are found at the lowest level of the tree structure. Each individual controller tag is defined to hold a specific data type, such as integers or floating-point numbers. An individual controller tag can also be a member of a controller tag array of a single data type.

3.1.1 MVI56(E)-MNET Controller Tags

The main controller tag, *MNET*, is broken down into four lower-level controller tags:



The four lower-level controller tags contain other controller tags and controller tag groups. Click the **[+]** sign next to each controller tag to expand it and view more controller tags.

For example, if you expand the *MNET.DATA* controller tag, you will see that it contains two controller tag arrays, *MNET.DATA.ReadData* and *MNET.DATA.WriteData*, which are 600-element integer arrays.

Scope: My_Controller Show... Show All				
	Name	Value	Data Type	Description
	+ AOI56MNET	{ ... }	AOI56MNET	Add-On - MVI56-MN
	- MNET	{ ... }	MNETMODULEDEF	Output parameters.
	- MNET.DATA	{ ... }	MNETDATA	Output parameters.
	+ MNET.DATA.ReadData	{ ... }	INT[600]	Output parameters.
	+ MNET.DATA.WriteData	{ ... }	INT[600]	Output parameters.
	+ MNET.STATUS	{ ... }	MNETSTATUS	Output parameters.
	+ MNET.CONTROL	{ ... }	MNETCONTROL	Output parameters.
	+ MNET.UTIL	{ ... }	MNETUTIL	Output parameters.

Each controller tag in the Add-On Instruction is commented in the *Description* column.

Notice that controller tags that are not at the lowest level (i.e. they have lower-level controller tags below them in the tree structure) have the name of a user-defined data type (UDT) in the *Data Type* column.

3.2 User-Defined Data Types (UDTs)

User-defined data types (UDTs) allow users to organize collections of data types into groupings. These groupings can then be used to declare the data type for aggregate controller tags. Another advantage of defining a UDT is that it may be re-used in other controller tags that use the same data types.

The Add-On Instruction Import Rung ladder logic for the module has pre-defined UDTs. You can find them in the *User-Defined* subfolder, located in the *Data Types* folder in the *Controller Organizer* pane of the main RSLogix window.

3.2.1 MVI56(E)-MNET User-Defined Data Types

Twelve different UDTs are defined for the MVI56(E)-MNET Add-On Instruction. The main UDT, *MNETMODULEDEF*, contains all the data types for the module and was used to create the main controller tag, *MNET*. There are four UDTs one level below *MNETMODULEDEF*. These lower-level UDTs were used to create the *MNET.DATA*, *MNET.STATUS*, *MNET.CONTROL*, and *MNET.UTIL* controller tags.

Name: MNETMODULEDEF

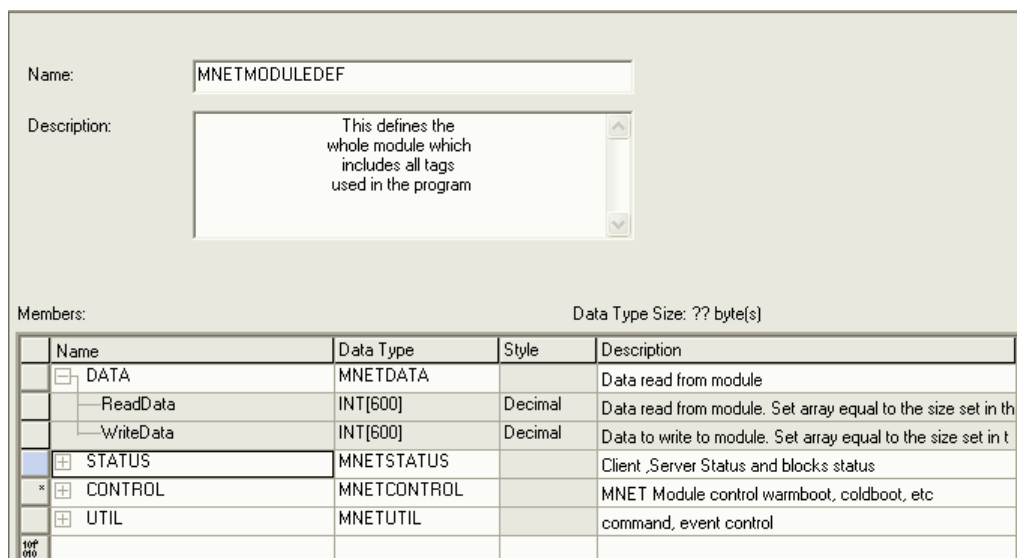
Description: This defines the whole module which includes all tags used in the program

Members: Data Type Size: ?? byte(s)

	Name	Data Type	Style	Description
+	DATA	MNETDATA		Data read from module
+	STATUS	MNETSTATUS		Client ,Server Status and blocks status
* +	CONTROL	MNETCONTROL		MNET Module control warmboot, coldboot, etc
+	UTIL	MNETUTIL		command, event control

Click the **[+]** signs to expand the UDT groupings and view the UDTs on the next level.

For example, if you expand *MNETDATA*, you will see that it contains two UDTs, *ReadData* and *WriteData*. Both of these are 600-element integer arrays.



The screenshot shows the 'MNETMODULEDEF' configuration window. The 'Name' field is 'MNETMODULEDEF'. The 'Description' field contains the text: 'This defines the whole module which includes all tags used in the program'. The 'Members' section displays a table with the following data:

Members:		Data Type Size: ?? byte(s)		
	Name	Data Type	Style	Description
	DATA	MNETDATA		Data read from module
	ReadData	INT[600]	Decimal	Data read from module. Set array equal to the size set in th
	WriteData	INT[600]	Decimal	Data to write to module. Set array equal to the size set in t
	STATUS	MNETSTATUS		Client ,Server Status and blocks status
x	CONTROL	MNETCONTROL		MNET Module control warmboot, coldboot, etc
	UTIL	MNETUTIL		command, event control

Notice that these UDTs are the data types in the *MNET.DATA.ReadData* and *MNET.DATA.WriteData* controller tags.

Each UDT is commented in the *Description* column.

3.3 Using Controller Tags

You can use controller tags to

- View Read and Write data that is being transferred between the module and the processor
- View status data for the module
- Set up and trigger Special Functions
- Initiate module restarts (Warm Boot or Cold Boot)

3.4 Controller Tag Overview

Controller Tag	Description
MNET.DATA	MNET input and output data transferred between the processor and the module
MNET.STATUS Status	information
MNET.CONTROL	Governs the data movement between the PLC rack and the module
MNET.UTIL	Generic tags used for internal ladder processing (DO NOT MODIFY)

The following sections describe each of these controller tags in more detail.

3.4.1 MNET.DATA

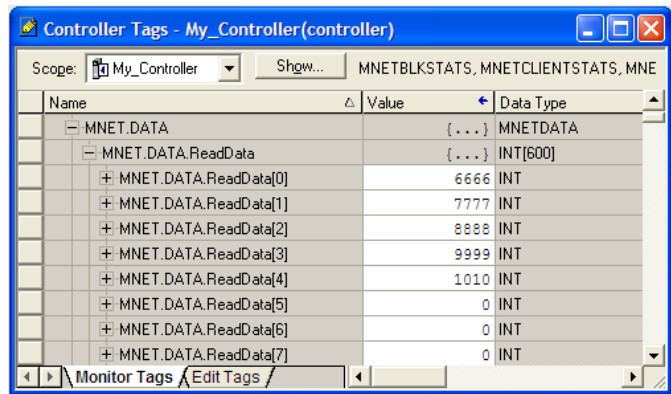
The controller tags in *MNET.DATA* hold data to be transferred between the processor and the MVI56-MNET module. The user data is the read and write data transferred between the processor and the module as "pages" of data up to 200 words long.

The data type for the MNET.DATA controller tag is an integer array containing a variable number of elements.

Controller Tag	Data Type	Description
ReadData INT[x]		Data read from module. Array size is equal to the size set in the configuration.
WriteData INT	[x]	Data to write to module. Array size is equal to the size set in the configuration.

MNET.DATA.ReadData

ReadData is an array that automatically adjusts to match the value entered in the *Read Register Count* (page 38) parameter of the configuration. For ease of use, this array should be dimensioned as an even increment of 200 words. This data is paged up to 200 words at a time from the module to the processor. The *ReadData* task places the data received into the proper position in the *ReadData* array. Use this data for status and control in the processor ladder logic.



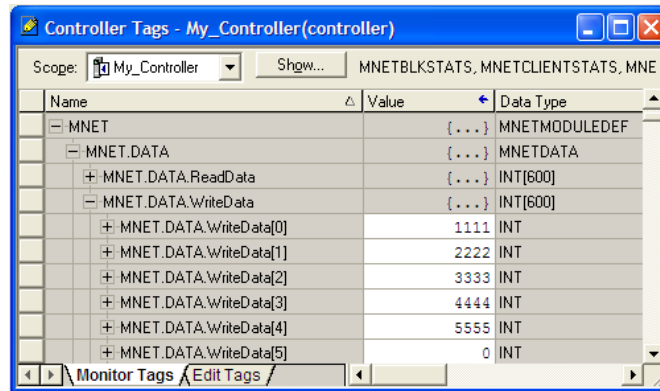
The *ReadData* array is related to the contents of the Read Data area of the module's internal database. To view the actual registers in the module's internal database, access the database display from *ProSoft Configuration Builder's Diagnostics* menu. For more information, see the section on *PCB Diagnostics* (page 72).

DATABASE DISPLAY 0 TO 99 (DECIMAL)

6666	7777	8888	9999	1010	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MNET.DATA.WriteData

WriteData is an array that automatically adjusts to match the value entered in the *Write Register Count* (page 39) parameter of the configuration. For ease of use, this array should be dimensioned as even increments of 200 words. This data is paged up to 200 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.



The *WriteData* array is related to the contents of the Write Data area of the module's internal database. To view the actual registers in the module's internal database, access the database display from *ProSoft Configuration Builder's Diagnostics* menu. For more information, see the section on *PCB Diagnostics* (page 72).

DATABASE DISPLAY 1000 TO 1099 (DECIMAL)

1111	2222	3333	4444	5555	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

3.4.2 MNET.STATUS

The *MNET.STATUS* controller tag has several lower-level controller tags. A few of them are described below.

MNET.STATUS.PassCnt

This is the Program Scan Counter value. It is incremented by a count of 1 each time a module's program cycle is complete.

MNET.STATUS.ClientStats

Controller Tag	Description
CmdReq	This value in this tag is incremented each time a Command Request is issued by the Client.
CmdResp	This value in this tag is incremented each time a Command Response is received by the Client.
CmdErr	This value in this tag is incremented each time an error message is received from a remote unit or a local error is generated for a command.
Requests	This value in this tag is incremented each time a request message is issued.
Responses	This value in this tag is incremented each time a response message is received.
ErrSent Reserv	ed.
ErrRec Reserv	ed.
CfgErrWord	Applicable to Clients and servers. This word encodes several potential errors using a bitmap. For more information on the meanings of the various bits when set, see Configuration Error Word (page 85).
CurrErr	Applicable to Clients and servers. Current error code number detected by the module.
LastErr	Applicable to Clients and servers. Previous error detected by the module.

MNET.STATUS.BlockStats

These tags display the backplane Block Transfer statistics.

Controller Tag	Description
Read	This tag contains the total number of Read blocks transferred from the module to the processor.
Write	This tag contains the total number of Write blocks transferred from the processor to the module.
Parse	This tag contains the total number of blocks successfully parsed that were received from the processor.
Event	This tag contains the total number of Event Command blocks received from the processor.
Cmd	This tag contains the total number of Command Control blocks received from the processor.
Err	This tag contains the total number of block Errors recognized by the module.

3.4.3 MNET.CONTROL

These controller tags are a 'scratchpad' area of intermediate data storage variables used by the ladder logic to keep track of various logic processing functions.

Controller Tag	Description
WarmBoot	Setting this tag to 1 will cause the module to reload, refreshing configuration parameters that must be set on program initialization. Only use this command if you must cause the module to re-boot.
ColdBoot	Set this tag to 1 when the module is required to perform the cold boot (hardware reset) operation. Do this when the module is experiencing a hardware problem requiring a hardware reset.
BPLastRead	This tag stores the latest Read Block ID received from the module. This value changes frequently.
BPLastWrite	This tag stores the latest Write Block ID to be sent to the module. This value changes frequently.
BlockIndex	This tag is an intermediate variable used during the block calculation.
WBPending Pend	ing message
CBPending Pend	ing message
ReadDataBlkCount	Holds the value of the Block Counts of the Read Data Array. Array size is the <i>Read Register Count</i> divided by 200.
WriteDataBlkCount	Holds the value of the Block Counts of the Write Data Array. Array size is the <i>Write Register Count</i> divided by 200.
RBTsremainder	Holds remainder calculation value from the read array.
WBTSremainder	Holds remainder calculation value from the write array.
ReadDataSizeGet	Holds read data array size.
WriteDataSizeGet	Holds write data array size.
IPAddress	Getting and setting IP address to and from the module.
FaultCode	Fault Code value
CheckInitialization	Check Initialization trigger

3.4.4 MNET.UTIL

Controller Tag	Description
CmdControl	This group of tags is used to control special or irregular execution of the commands listed in the configuration under the MNet Client 0 Commands section, regardless of whether or not such commands are normally enabled or disabled.
EventCmd	This group of tags is used to create and have the Client execute a special ladder logic constructed command that is not included in the MNet Client 0 Commands section of the configuration file.
InitOutputData	This group of tags is used for setting up data values when the module performs a restart operation. It will request the processor's output data and transfer it into the module's Modbus registers. Use the <i>Initialize Output Data</i> parameter in the configuration file to bring the module to a known state after a restart operation.
PassThru	This group of tags is used for transferring a remote Client's write commands through the MNET module straight into the processor's controller tags without first storing the data in the module's Modbus registers.
IPsetPending	Allows setting module IP address
IPgetPending	Allows getting module IP address

For more information, refer to Special Function Blocks (page 100).

4 Diagnostics and Troubleshooting

In This Chapter

❖ LED Indicators	70
❖ Using ProSoft Configuration Builder (PCB) for Diagnostics.....	72
❖ Reading Status Data from the Module	84
❖ Configuration Error Word.....	85

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-MNET Status Data Area (page 95).

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-MNET is working normally.
		ON	The MVI56-MNET 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 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.

4.1.2 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.

4.1.3 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-MNET. 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: <i><Backplane Status></i> 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: ▪ The processor is in RUN or REM RUN mode. ▪ The backplane driver is loaded in the module. ▪ The module is configured for read and write data block transfer. ▪ The ladder logic handles all read and write block situations. ▪ The module is properly configured in the processor I/O configuration and ladder logic.
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

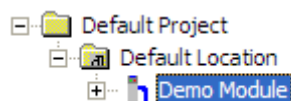
The *Configuration and Debug* menu for this module is arranged as a tree structure, with the *Main* menu at the top of the tree, and one or more submenus for each menu command. The first menu you see when you connect to the module is the *Main* menu.

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.

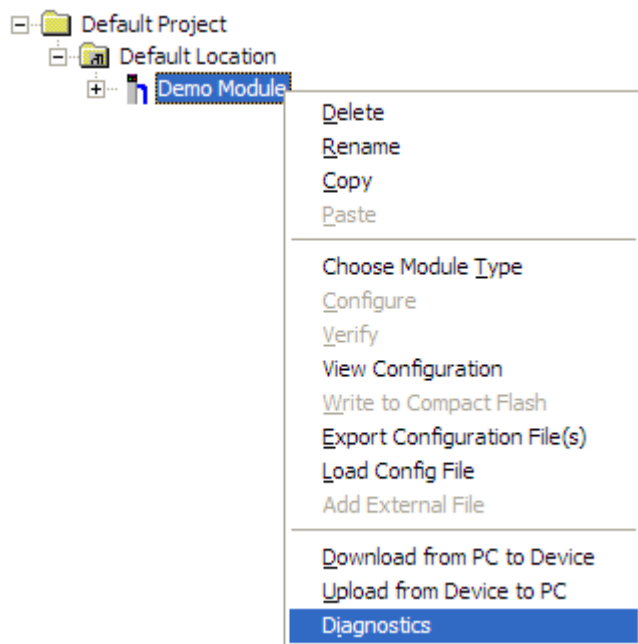
4.2.1 Using the Diagnostic Window in ProSoft Configuration Builder

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.

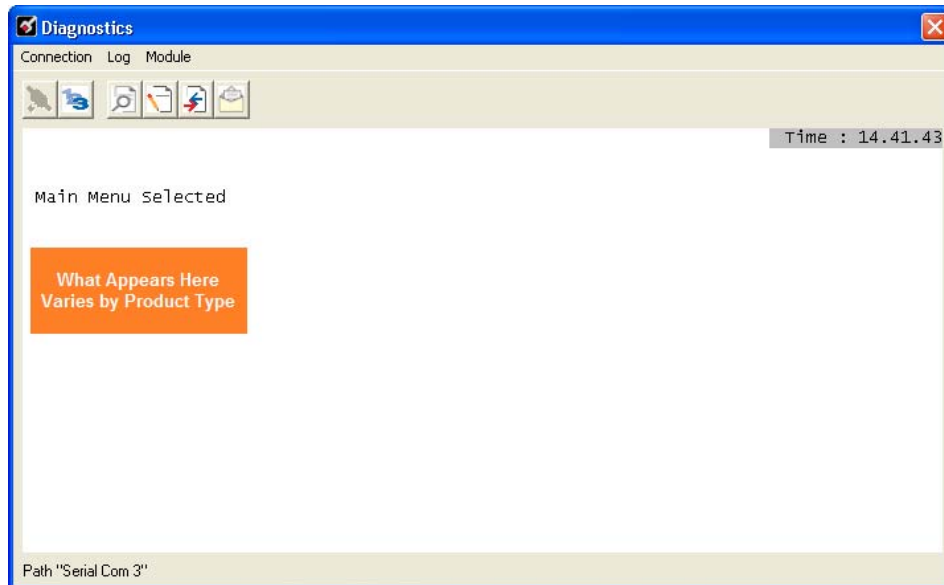


- 2 On the shortcut menu, choose **DIAGNOSTICS**.



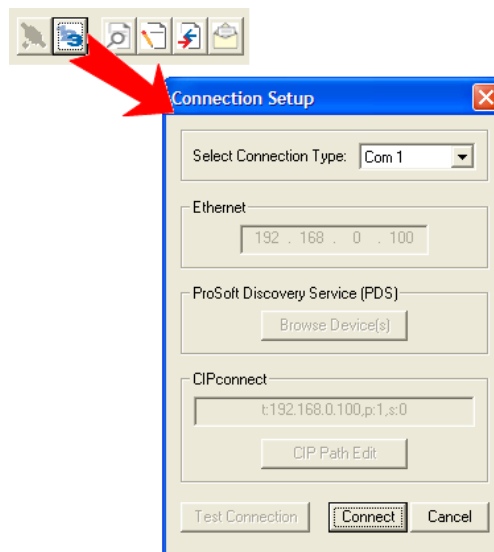
This action opens the *Diagnostics* dialog box.

- 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.



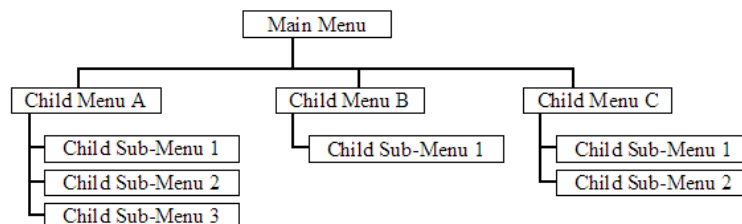
- 2 Verify that the null modem cable is connected properly between your computer's serial port and the module. A regular serial cable will not work.
- 3 On computers with more than one serial port, verify that your communication program is connected to the same port that is connected to the module.

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

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.

4.2.2 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-MNET 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 79).

Opening the Command Error List Menu

Press **[E]** 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.

Opening the Command List Menu

Press **[I]** to open the Command List menu. Use this command to view the configured command list for the module. For more information about this submenu, see Command List Menu (page 81).

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.

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.

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.

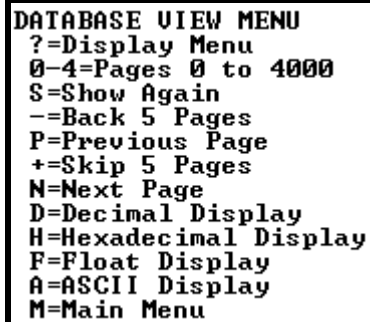
The *Network* menu allows you to send, receive and view the WATTCP.CFG file that contains the IP, gateway and other network specification information. For more information about this submenu, see Network Menu (page 82).

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.

4.2.3 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 for information on the structure of Modbus messages. Each option available on the menu is discussed in the following topics.

Viewing Register Pages

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

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

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

Redisplaying the Current Page

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

Moving Back Through 5 Pages of Registers

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

Viewing the Previous Page of Registers

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

Moving Forward Through 5 Pages of Registers

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

Viewing the Next Page of Registers

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

Viewing Data in Decimal Format

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

Viewing Data in Hexadecimal Format

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

Viewing Data in Floating-Point Format

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

Viewing Data in ASCII (Text) Format

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

Returning to the Main Menu

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

4.2.4 Command List Menu

Use this menu to view the configured command list for the module.

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.

Redisplaying the Current Page

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

Use this command to display the current page of commands. Ten commands are displayed on each page.

If an enabled command has an error, the EN field will contain a value of -1. This indicates that the command will be re-issued every 30 seconds.

Moving Back Through 5 Pages of Registers

Press **[-]** from the *Database View* menu to skip five pages back in the database to see the 10 commands starting 50 commands before the currently displayed page.

Viewing the Previous Page of Commands

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

Moving Forward Through 5 Pages of Registers

Press **[+]** from the *Database View* menu to skip five pages ahead in the database to see 10 commands 50 commands ahead of the currently displayed page.

Viewing the Next Page of Commands

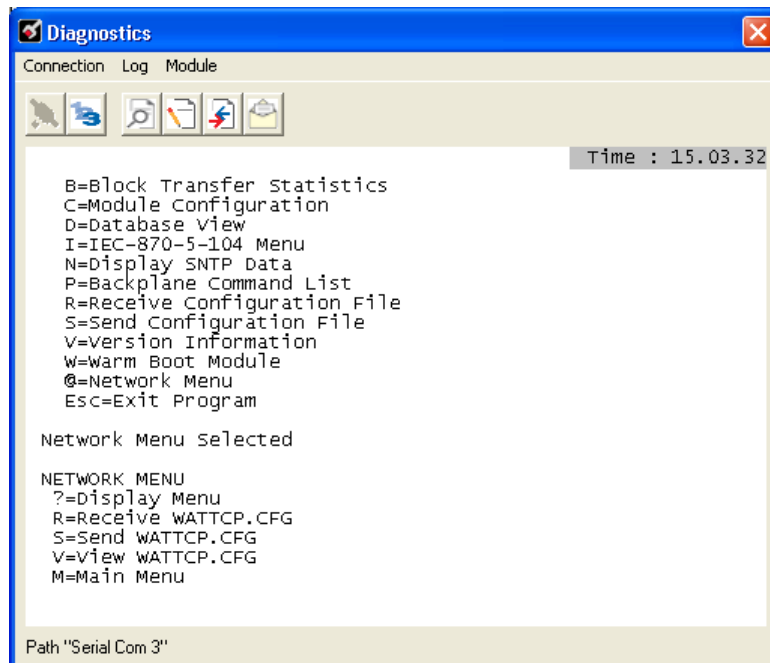
Press **[N]** to display the next page of commands.

Returning to the Main Menu

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

4.2.5 Network Menu

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-MNET module returns a block of status data in the input image that can be used to determine the module's operating status. This data is transferred from the module to the ControlLogix processor continuously. You can view this data in the *MNET.STATUS* controller tag in the ladder logic. For more information, see *MNET.STATUS* (page 66).

If the *Error/Status Pointer* is enabled, the status data can also be found in the Read Data area of the module's database at a location specified by the *Error/Status Pointer* configuration parameter. For more information, see *Error/Status Pointer* (page 38).

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

4.4 Configuration Error Word

The *Configuration Error Word* contains general module, Client, and server configuration error indications, in a bit-mapped format. Specific bits in the module's *Configuration Error Word* are turned on (set to **1**) to indicate various configuration errors. The *Configuration Error Word* appears in three separate *MNET.STATUS* controller tags. Since there is only one *Configuration Error Word* for the whole module, each of these tags contains exactly the same data, even though the tag name might imply otherwise. Multiple copies of the same module error data have been included in these different controller tag locations for your convenience when troubleshooting.

Bits in the *Configuration Error Word* indicate the following errors:

Bit	Description	Hex Value
0	Reserved - not currently used	0001h
1	Reserved - not currently used	0002h
2	Reserved - not currently used	0004h
3	Reserved - not currently used	0008h
4	Invalid retry count parameter (Client only)	0010h
5	The float flag parameter is not valid. (Client or server)	0020h
6	The float start parameter is not valid. (Client or server)	0040h
7	The float offset parameter is not valid. (Client or server)	0080h
8	The ARP Timeout is not in range (ARP Timeout parameter 0 or greater than 60000 milliseconds) and will default to 5000 milliseconds. (Client only)	0100h
9	The Command Error Delay is > 300 and will default to 300. (Client only)	0200h
10	Reserved - not currently used	0400h
11	Reserved - not currently used	0800h
12	Reserved - not currently used	1000h
13	Reserved - not currently used	2000h
14	Reserved - not currently used	4000h
15	Reserved - not currently used	8000h

Combinations of errors will result in more than one bit being set in the error word. Correct any invalid data in the configuration for proper module operation. A value of zero (0) in this word indicates all bits are clear, which means that all module configuration parameters contain valid values. However, this does not mean that the configuration is valid for the user application. Make sure each parameter is set correctly for the intended application.

5 Reference

In This Chapter

❖ Product Specifications	87
❖ About the MODBUS TCP/IP Protocol.....	90
❖ Backplane Data Transfer.....	91
❖ Data Flow between the MVI56-MNET Module and ControlLogix Processor	111
❖ Cable Connections	117
❖ Adding the Module to an Existing Project.....	122
❖ Using the Sample Program	125

5.1 Product Specifications

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

Compatible devices include Modicon Programmable Automation Controllers (PACs), as well as a wide variety of instruments and devices. A 5000-word register space in the module exchanges 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
- Local or remote rack

5.1.2 Modbus TCP/IP

- Single Slot - ControlLogix backplane compatible
- 10/100 MB Ethernet port
- Module I/O data memory mapping supports up to 5000 registers and is user definable
- ProSoft Configuration Builder (PCB) software supported, a Windows-based graphical user interface providing simple product and network configuration
- Sample Ladder Logic and Add-On Instructions (AOI) are used for data transfer between module and processor and module configuration
- Personality Module (non-volatile CF card) used to store module and network configuration allowing for in the field quick product replacement.

5.1.3 Functional Specifications

The MVI56-MNET will operate on a Local or Remote rack. (For remote rack applications with smaller data packet size please refer to the MVI56-MNETR product. For applications requiring up to 30 Client connections please refer to the MVI56-MNETC. The MVI56-MNETCR combines the MNETC and MNETR product functionalities)

- 10/100 MB Ethernet Application port
- CIPconnect[®] enabled for module, network configuration and diagnostics using 1756-ENxT module with EtherNet/IP pass-thru communications
- Supports Enron version of Modbus protocol for floating point data transactions
- 4-digit LED Display for English based status and diagnostics information
- PCB includes a powerful Modbus network analyzer
- Special functions (command control, event commands, status, etc.) are supported by message transfer (unscheduled) using the MSG instruction
- Configurable parameters for the Client including a minimum response delay of 0 to 65535 ms and floating point support
- Supports ten independent server connections for Service Port 502
- Supports ten independent server connections for Service Port 2000
- All data mapping begins at Modbus register 40001.
- Error codes, network error counters, and port status data available in user data memory

Server Specifications

The MVI56-MNET module accepts Modbus function code commands of 1, 2, 3, 4, 5, 6, 8, 15, 16, 17, 22 and 23 from an attached Modbus Client unit. A port configured as a Modbus server permits a remote Client to interact with all data contained in the module. This data can be derived from other Modbus server devices on the network, through a Client port, or from the ControlLogix processor.

Client Specifications

A port configured as a virtual Modbus Client device on the MVI56-MNET module actively issues Modbus commands to other nodes on the Modbus network. One hundred (100) commands are supported on each port. Additionally, the Client ports have an optimized polling characteristic that polls servers with communication problems less frequently. The ControlLogix processor can be programmed to control the activity on the port by actively selecting commands from the command list to execute or issuing commands directly from the ladder logic.

5.1.4 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
Application port (Ethernet)	
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
Debug/Configuration port (CFG)	
CFG Port (CFG)	RJ45 (DB-9M with supplied cable) No hardware handshaking

5.2 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-MNET 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-MNET 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.

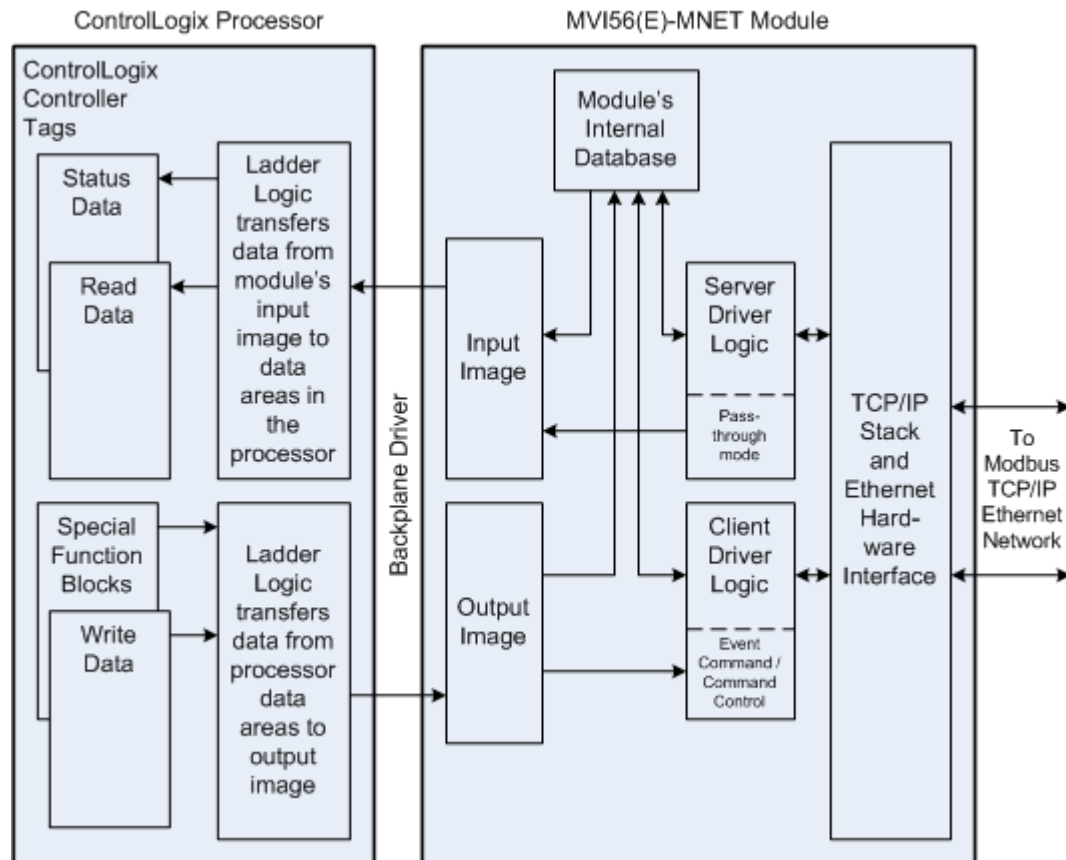
5.3 Backplane Data Transfer

The MVI56-MNET 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 update times range from 1 to 10 milliseconds.

This bi-directional transfer of data is accomplished by the module putting data in the input image to send to the processor. Data in the input image is placed in the processor's controller tags by ladder logic. The input image is set to 250 words.

Processor logic inserts data to the output image to be transferred to the module. The module's firmware program extracts the data and places it in the module's internal database. The output image is set to 248 words.

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

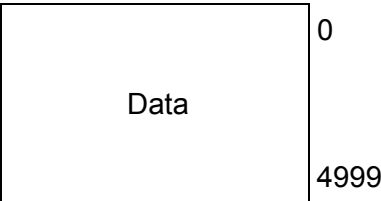


All data transferred between the module and the processor over the backplane is through the input and output images. Ladder logic must be written in the ControlLogix processor to interface the input and output image data with data contained in the controller tags. All data used by the module is stored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

Module's Internal Database Structure

5000 registers for user data

Register



Data contained in this database is transferred in blocks, or pages, using the input and output images. ControlLogix ladder logic and the MVI56-MNET module's program work together to coordinate these block transfers. Up to 200 words of data can be transferred from the module to the processor (read block - input image) or from the processor to the module (write block - output image) in each block transfer. The block structure of each block type depends on the data content and the data transfer function to be performed by the block. The module uses the following block identification numbers.

Block ID Range	Descriptions
-1	Null block
0	For firmware versions earlier than 2.05, this is a null block. For firmware versions 2.05 and newer, block 0 contains the same data as block 1. This feature enhances performance, especially when using less than 200 words of read/write data: - If Read Register Count in the module configuration file is set > 200 words, Block ID 0 is not used. - If Read Register Count in the module configuration file is set >0 and <= 200 words, Block ID contains the same data as block 1 (both read data and status data).
1 to 25	Read or Write blocks
1000 to 1024	Initialize Output Data blocks
2000	Event Command block
5001 to 5006	Command Control blocks
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
9990	Set Module IP Address block
9991	Get Module IP Address block
9996	Unformatted Pass-through block with raw Modbus message
9998	Warm-boot block
9999	Cold-boot block

These block identification codes can be broken down into two groups:

Normal data transfer blocks

- Read and Write blocks (-1 to 25)

Special function blocks

- Initialize Output Data blocks (1000 to 1024)
- Event Command block (2000)
- Command Control blocks (5001 to 5006)
- Pass-through blocks (9956 to 9961, 9970 and 9996)
- Module IP Address blocks (9990 and 9991)
- Warm-boot and Cold-boot blocks (9998 and 9999)

5.3.1 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.

Read Block

These blocks of data transfer information from the module to the ControlLogix processor.

The following table describes the structure of the input image.

Read Block from Module to Processor

Word Offset	Description	Length
0	Reserved	1
1	Write Block ID	1
2 to 201	Read Data	200
202	Program Scan Counter	1
203 to 208	Block Transfer Status	6
209 Product	Code 1	1
210 Product	Code 2	1
211 Versio	n number	1
212 to 218	Not Used	7
219 to 221	Reserved	2
222 to 228	MNet Server Status	7
229 to 231	Reserved	2
232 to 238	MBAP Server Status	7
239 to 248	MNet Client Status	10
249	Read Block ID	1

The Read Block ID is an index value used to determine where the 200 words of data from module memory will be placed in the *ReadData[x]* controller tag array of the ControlLogix processor. Each transfer can move up to 200 words (block offsets 2 to 201) 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.

During 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.

MVI56-MNET Status Data Area

The following table describes in more detail the status information found in the Read Block. The status information can be viewed in

- the *MNET.STATUS* controller tags
- the *Diagnostics* menu of *Prosoft Configuration Builder*

Offset	Content	Description
202	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
203	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
204	Write Block Count	This field contains the total number of write blocks transferred from the processor to the module.
205	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
206	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
207	Command Block Count	This field contains the total number of command blocks received from the processor.
208	Error Block Count	This field contains the total number of block errors recognized by the module.
209 Product	Code 1	This register displays the first word of the product code in ASC format
210 Product	Code 2	This register displays the second word of the product code in ASC format
211 Versio	n number	This register displays the version number in decimal values. For example, if the version number is 1.51, it will display as 151.
212 Reserv	ed	Not used
213 Reserv	ed	Not used
214 Reserv	ed	Not used
215 Reserv	ed	Not used
216 Reserv	ed	Not used
217 Reserv	ed	Not used
218 Reserv	ed	Not used
219 Reserv	ed	Not used
220 Reserv	ed	Not used
221 Reserv	ed	Not used
222	MNet Request Count	This counter increments each time an MNet (port 2000) request is received.
223	MNet Response Count	This counter is incremented each time an MNet (port 2000) response message is sent.
224	MNet Errors Sent Count	This counter increments each time an MNet (port 2000) sends an exception response to Client. Example: Client sent illegal Modbus Data location address.
225	MNet Errors Received Count	This counter increments each time an MNet (port 2000) receives a bad command. Example: Client sent illegal function command.
226	MNet Configuration Error Word	This word contains a bit map that indicates general module configuration errors.

Offset	Content	Description
227	Reserv ed	Not used
228	Reserv ed	Not used
229	Reserv ed	Not used
230	Reserv ed	Not used
231	Reserv ed	Not used
232	MBAP Request Count	This counter increments each time a MBAP (port 502) request is received.
233	MBAP Response Count	This counter is incremented each time a MBAP (port 502) response message is sent.
234	MBAP Errors Sent Count	This counter increments each time an MNet (port 502) sends an exception response to Client. Example: Client sent illegal Modbus Data location address.
235	MBAP Errors Received Count	This counter increments each time an MNet (port 502) receives a bad command. Example: Client sent illegal function command.
236	MBAP Configuration Error Word Count	This word contains a bit map that indicates general module configuration errors.
237	Reserv ed	Not used
238	Reserv ed	Not used
239	Client Cmd Request	This value is incremented each time a command request is issued.
240	Client Cmd Response	This value is incremented each time a command response is received.
241	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.
242	Reserv ed	Not used
243	Reserv ed	Not used
244	Reserv ed	Not used
245	Reserv ed	Not used
246	Client Cfg Error Word	This word contains a bit map that indicates general module configuration errors.
247	Client Current Error Code	This value corresponds to the current error code for the Client.
248	Client Last Error Code	This value corresponds to the last error code recorded for the Client.

Write Block

These blocks of data transfer information from the ControlLogix processor to the module.

The following table describes the structure of the output image.

Write Block from Processor to Module

Word Offset	Description	Length
0	Write Block ID	1
1 to 200	Write Data	200
201 to 247	Spare	46
247	Select Priority Read Block	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 200 words (block offsets 1 to 200) of data.

Select Priority Read Block (Write Block Offset 247)

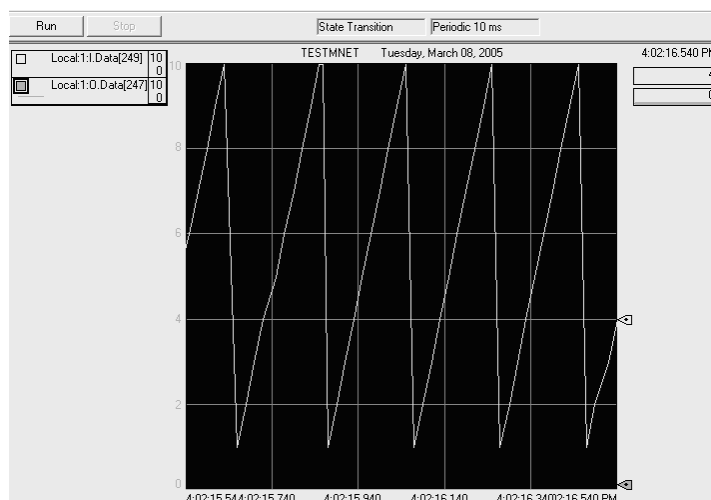
Note: The Select Priority Read Block feature is only available for firmware versions 1.36.000 and newer.

This register allows the processor to select which read blocks will be returned from the module. If this register equals zero, the module will return all read blocks in sequential order.

If this register has a non-zero value, the module will return the read block selected, and the following one.

This feature can be used for applications that require some read blocks to be updated more frequently than other blocks.

The following illustrations show the effect of changing the value of the Select Priority Read Block register (Write Block offset 247). In the following histogram curve, the Select Priority Read Block is equal to 0.



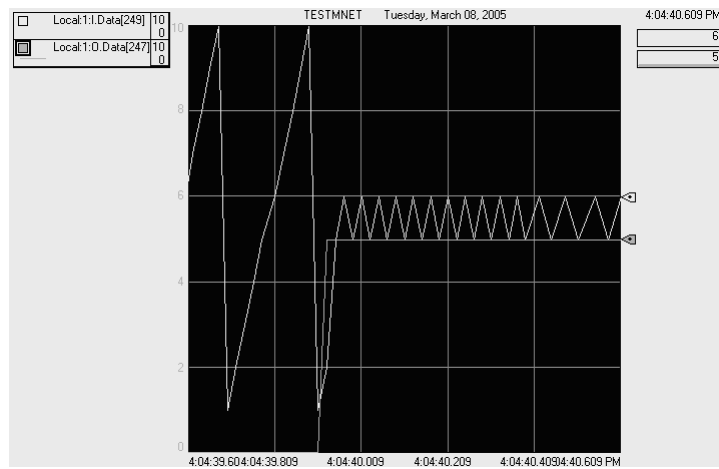
- Local:1.O.Data[247] = Select Priority Read Block.
- Local:1.I.Data[249] = Read Block ID.

In the example above, all read blocks (1 to 10) are returned in sequential order.

Select Priority Read Block = 5

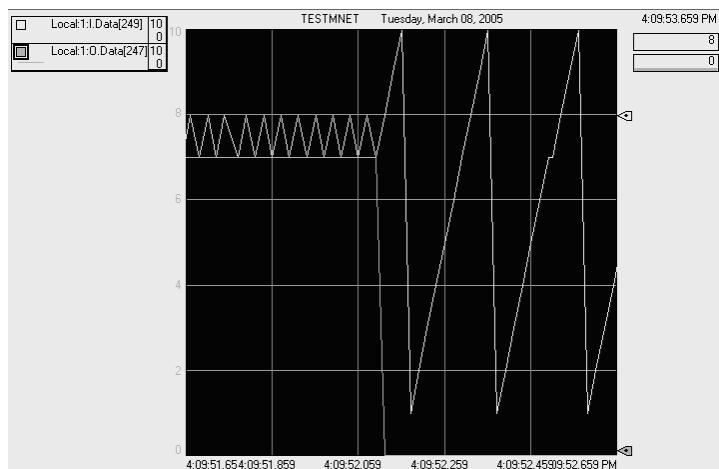
If the ladder logic changes the value of Local:1.O.Data[247] from 0 to 5, note that the Local:1.I.Data[249] value begins to alternate between Block IDs 5 and 6 as long as Local:1.O.Data[247] stays set to 5.

5-6-5-6-5-6-5-6-5-6-...



Select Priority Read Block = 0

After the ladder logic changes the value of Local:1:O.Data[247] from 5 to 0, then the Local:1:I.Data[249] value is updated as before, by returning all blocks 1 through 10 in a repeating sequence.



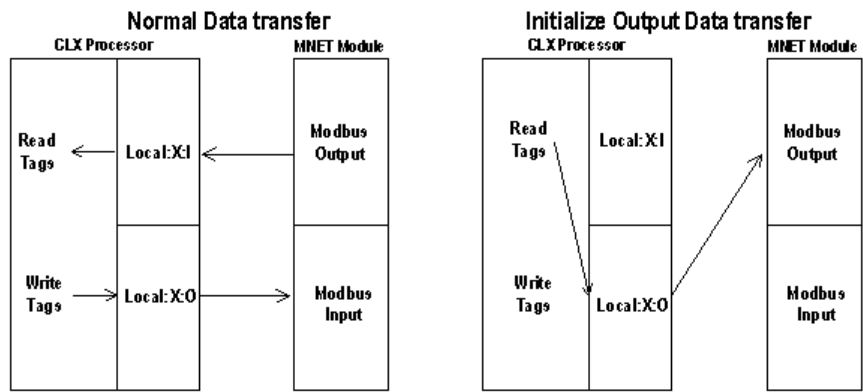
5.3.2 Special Function Blocks

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.

Initialize Output Data Blocks (1000 to 1024)

Use the *Initialize Output Data* parameter in the configuration to bring the module to a known state after a restart operation. If the *Initialize Output Data* parameter is enabled, when the module performs a restart operation, it will request blocks of output data from the *ReadData* array in the processor to initialize the Read Data area of the module’s internal database.



Block Request from Module to Processor

Word Offset	Description	Length
0	Reserved	1
1	1000 to 1024	1
2 to 248	Spare	247
249	1000 to 1024	1

Ladder logic subtracts 1000 from the value contained in word 249 to determine a block index. This block index determines which 200-word block of data will be taken from the *ReadData* array and placed in the output image to be returned to the module.

Block Response from Processor to Module

Word Offset	Description	Length
0	1000 to 1024	1
1 to 200	Output data to preset in module.	200
201 to 247	Spare	47

Event Command Blocks (2000)

During routine operation, the module continuously cycles through the user-defined *MNET Client 0 Command List* (page 44), examining commands in the order they are listed and sending enabled commands on the network. However, the module also has a special command priority queue, which is an internal buffer that holds commands from special function blocks until they can be sent on the network.

When one or more commands appear in the command priority queue:

- 1 The normal polling process is temporarily interrupted.
- 2 The commands in the command priority queue are executed until the queue is empty.
- 3 Then the module goes back to where it left off on the *MNET Client 0 Command List* and continues routine polling.

Event Command blocks send Modbus TCP/IP commands directly from controller tags by ladder logic to the Client command priority queue on the module. Event Commands are not placed in the module's internal database and are not part of the *MNET Client 0 Command List*.

Block Request from Processor to Module

Word Offset	Description	Length
0	<i>Block ID</i> - This word contains block identification code 2000 to indicate that the block contains a command to be executed by the Client driver.	1
1 to 4	<i>IP Address</i> - These four words contain the IP address of the destination server. Each octet value (0 to 255) of the destination server's 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 ensure the values are in the range of 0 to 255.	4
5	<i>Service Port</i> - This word contains the TCP service port used in the message. 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 used. 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 (serial-type) message.	1
6	<i>Slave Address</i> - This word contains the Modbus node address for the message. This field should have a value from 1 to 247 .	1
7	<i>Internal DB Address</i> - 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 .	1
8	<i>Point Count</i> - This word contains the count parameter that determines the number of digital points or registers to associate with the command.	1
9	<i>Swap Code</i> - This parameter specifies the swap type for the data. This option is valid only for function codes 3 and 4.	1
10	<i>Modbus Function Code</i> - This word contains the Modbus function code for the command.	1
11	<i>Device Database Address</i> - This word contains the Modbus address in the server device to be associated with the command.	1
12 to 247	Spare	236

When the module receives this request block, it builds the command, places the command in the command priority queue (if the queue is not already full; maximum capacity is 100 commands), and returns a response block to tell the ladder logic whether or not the command has been successfully added to the queue.

Block Response from Module to Processor

Word Offset	Description
0 Reserved	
1	This word contains the next write request block identification code.
2	This word contains the result of the event request. If a value of one (1) is present, the command was successfully added to the queue. If a value of zero (0) is present, no room was found in the command queue.
3 to 248	Spare
249	This word contains the block identification code 2000 requested by the processor.

Word 2 of the block can be used by the ladder logic to determine whether or not the command was successfully added to the command priority queue. The command will fail if the queue for the port is already full at the time when the Event Command block is received by the module.

Controller Tags

The elements of the *MNET.UTIL.EventCmd* controller tag array contain all the values needed to build one Modbus TCP/IP command, have it sent to the module, and control the processing of the returned response block.

Controller Tag	Description
EventCmdTrigger	Set this tag to 1 to trigger the execution of the Event Command.
EventCmdPending	Temporary variable used to prevent a new Event Command block from being sent to the module until the previously sent Event Command block has been completely processed and a response block has been returned.
IPAddress	Enter the four octet IP address numbers of the target Modbus server into this array tag.
ServicePort	Enter 502 for a MBAP message or 2000 for a MNET message.
SlaveAddress	Enter the Modbus Node Address. Enter 0 , if not needed.
InternalDBAddress	Enter the database address for the Client.
PointCount	Enter the number of words or bits to be transferred by the Client.
SwapCode	Enter the swap type for the data. This function is only valid for function codes 3 and 4.
ModbusFunctionCode	Enter the Modbus function code for the command.
DeviceDBAddress	Enter the database address for the server.
EventCmdStatusReturned	Temporary variable that provides status indication of whether or not the Event Command was successfully added to the command execution queue.
EventBlockID	Temporary variable that provides the identification code number of the Block ID just executed.

Command Control Blocks (5001 to 5006)

During routine operation, the module continuously cycles through the user-defined *MNET Client 0 Command List* (page 44), examining commands in the order they are listed and sending enabled commands on the network. However, the module also has a special command priority queue, which is an internal buffer that holds commands from special function blocks until they can be sent on the network.

When one or more commands appear in the command priority queue:

- 1 The normal polling process is temporarily interrupted.
- 2 The commands in the command priority queue are executed until the queue is empty.
- 3 Then the module goes back to where it left off on the *MNET Client 0 Command List* and continues routine polling.

Like Event Command blocks, Command Control blocks place commands into the module's command priority queue. Unlike Event Commands blocks, which contain all the values needed for one command, Command Control is only used with commands already defined in the *MNET Client 0 Command List*.

Commands in the *MNET Client 0 Command List* may be either enabled for normal polling or disabled and excluded from routine polling. A disabled command has its *Enable* parameter set to **NO** (0) and is skipped during routine polling. An enabled command has its *Enable* parameter set to **YES** (1) and is sent during routine polling. However, Command Control allows any command in the predefined *MNET Client 0 Command List* to be added to the command priority queue, whether it is enabled for routine polling or not.

Command Control also gives you the option to use ladder logic to have commands from the *MNET Client 0 Command List* executed at a higher priority and out of routine order, if such an option might be required in special circumstances.

A single Command Control block request can place up to six commands from the *MNET Client 0 Command List* into the command priority queue.

Block Request from Processor to Module

Word Offset	Description
0	Command Control block identification code of 5001 to 5006 . The rightmost digit indicates the number of commands (1 to 6) to add to the command priority queue.
1	This word contains the Command Index for the first command to be entered into the queue.
2	This word contains the Command Index for the second command to be entered into the queue.
3	This word contains the Command Index for the third command to be entered into the queue.
4	This word contains the Command Index for the fourth command to be entered into the queue.
5	This word contains the Command Index for the fifth command to be entered into the queue.
6	This word contains the Command Index for the sixth command to be entered into the queue.
7 to 247	Spare

The last digit in the block identification code indicates the number of commands to process. For example, a block identification code of **5003** indicates that three commands are to be placed in the queue. In this case, the first three of the six available Command Indexes will be used to determine exactly which three commands will be added to the queue, and to set their order of execution.

Values to enter for the six Command Indexes range from **0** to **99** and correspond to the *MNET Client 0 Command List* entries, which are numbered from 1 to 100. To determine the Command Index value, subtract one (**1**) from the row number of the command in the *MNET Client 0 Command List*, as seen in the *Command Editor* window of *ProSoft Configuration Builder (PCB)*.

The module responds to a Command Control block request with a response block, indicating the number of commands added to the command priority queue.

Block Response from Module to Processor

Word Offset	Description
0	Reserved
1	This word contains the next write block identification code.
2	This word contains the number of commands in the block placed at the front of the command priority queue.
3 to 248	Spare
249	This word contains the block 5001 to 5006 requested by the processor.

Controller Tags

The *MNET.UTIL.CmdControl* controller tag array holds all the values needed to create one Command Control block, have it sent to the module, and control the processing of the returned response block.

Controller Tag	Description
TriggerCmdCntrl	Set this tag to 1 to trigger the execution of a command after all the other parameters have been entered.
NumberOfCommands	Enter a decimal value representing the quantity of commands to be requested in the Command Control block (1 to 6).
CommandIndex[x]	Enter the ROW NUMBER of the command in the <i>MNET Client 0 Command List</i> in <i>Prosoft Configuration Builder</i> minus 1 . This is a six-element array. Each element holds one Command Index.
CmdsAddedToQueue	Returned decimal value representing the quantity of commands added from the <i>MNET Client 0 Command List</i> to the command priority queue by the most recent Command Control block.
CmdControlBlockID	Temporary variable that provides block ID of the Command Control block most recently processed by the module.
CmdCntrlPending	Temporary variable used to prevent a new Command Control block from being sent to the module until the previously sent Command Control block has been completely processed and a response block has been returned.

Pass-Through Blocks (9956-9961 and 9970)

In Pass-Through mode, write messages sent to a server port are passed directly through to the processor. In this mode, the module sends special blocks to the processor when a write request is received from a Client. Ladder logic must handle the receipt of these blocks and place the enclosed data into the proper controller tags in the processor.

There are two basic modes of operation when the pass-through feature is utilized: Unformatted (code 1) and Formatted (code 2 or 3). In the unformatted mode, messages received on the server are passed directly to the processor without any processing. These unformatted blocks require more decoding than the formatted blocks.

The Modbus protocol supports control of binary output (coils - functions 5 and 15) and registers (functions 6 and 16).

Any Modbus function 5, 6, 15 or 16 commands will be passed from the server to the processor using the block identification numbers 9956 to 9961, 9970 and 9996.

Formatted Pass-Through Blocks

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 expected by the processor. 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 used for function 99.

Pass-Through Blocks 9956, 9957, 9958, 9960 and 9961 from Module to Processor

Word Offset	Description	Length
0	0	1
1	9956, 9957, 9958, 9960 or 9961	1
2	Number of word registers in Modbus data set	1
3	Starting address for Modbus data set	1
4 to 248	Modbus data set	245
249	9956, 9957, 9958, 9960 or 9961	1

Pass-Through Block 9959 from Module to Processor

Word Offset	Description	Length
0	0	1
1 995	9	1
2	Number of word registers in Modbus data set	1
3	Starting word address for Modbus data set	1
4 to 53	Modbus data set	50
54 to 103	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.	50
104 to 248	Spare data area	145
249	9959	1

Pass-Through Block 9970 from Module to Processor

Word Offset	Description	Length
0	0	1
1 997	0	1
2	1	1
3 0		1
4 to 248	Spare data area	245
249 999	6	1

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

Response Blocks 9956, 9957, 9958, 9959, 9960, 9961, or 9970 from Processor to Module

Word Offset	Description	Length
0	9956, 9957, 9958, 9959, 9960, 9961, or 9970	1
1 to 249	Spare data area	247

Unformatted Pass-Through Blocks

When the unformatted pass-through mode (code 1) is selected, information is passed from the module to the processor with a block identification code of 9996. Word 2 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.

Pass-Through Block 9996 from Module to Processor

Word Offset	Description	Length
0	0	1
1 999	6	1
2	Number of bytes in Modbus msg	1
3	Reserved (always 0)	1
4 to 248	Modbus message received	245
249 999	6	1

The ladder logic should copy and parse the received message and control the processor as expected by the Client device. The processor must respond to the pass-through block with a write block.

Response Block 9996 from Processor to Module

Word Offset	Description	Length
0	9996	1
1 to 247	Spare	247

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

Set Module IP Address Block (9990)

Block Request from Processor to Module

Word Offset	Description	Length
0 999	0	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 247	Reserved	243

Block Response from Module to Processor

Word 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 248	Spare data area	243
249 999	0	1

Get Module IP Address Block (9991)

Block Request from Processor to Module

Word Offset	Description	Length
0 999	1	1
1 to 247	Spare data area	247

Block Response from Module to Processor

Word 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 248	Spare data area	243
249 999	1	1

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. This block is commonly sent to the module any time configuration data modifications are made in the controller tags data area. This will cause the module to read the new configuration information and to restart.

Block Request from Processor to Module

Word Offset	Description	Length
0	9998	1
1 to 247	Spare	247

Cold Boot Block (9999)

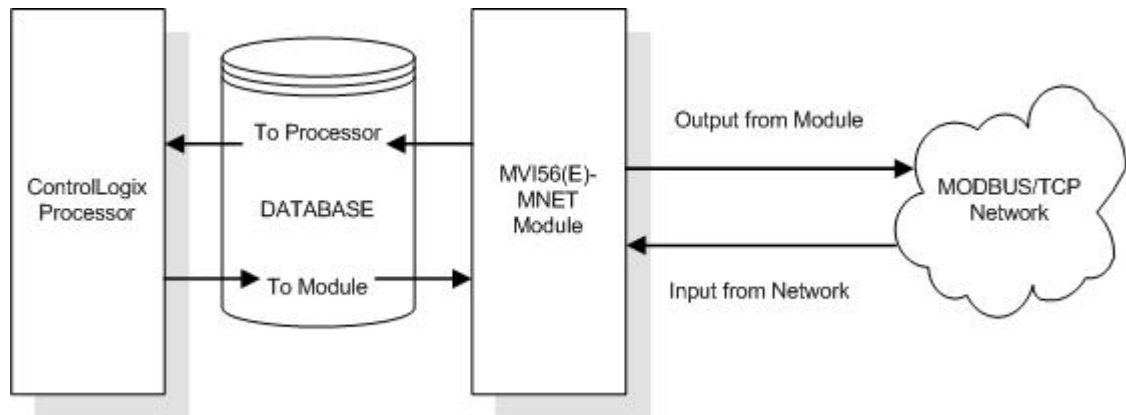
This block is sent from the ControlLogix processor to the module (output image) when the module is required to perform the cold boot (hardware reset) operation. This block is sent to the module when a hardware problem is detected by the ladder logic that requires a hardware reset.

Block Request from Processor to Module

Word Offset	Description	Length
0	9999	1
1 to 247	Spare	247

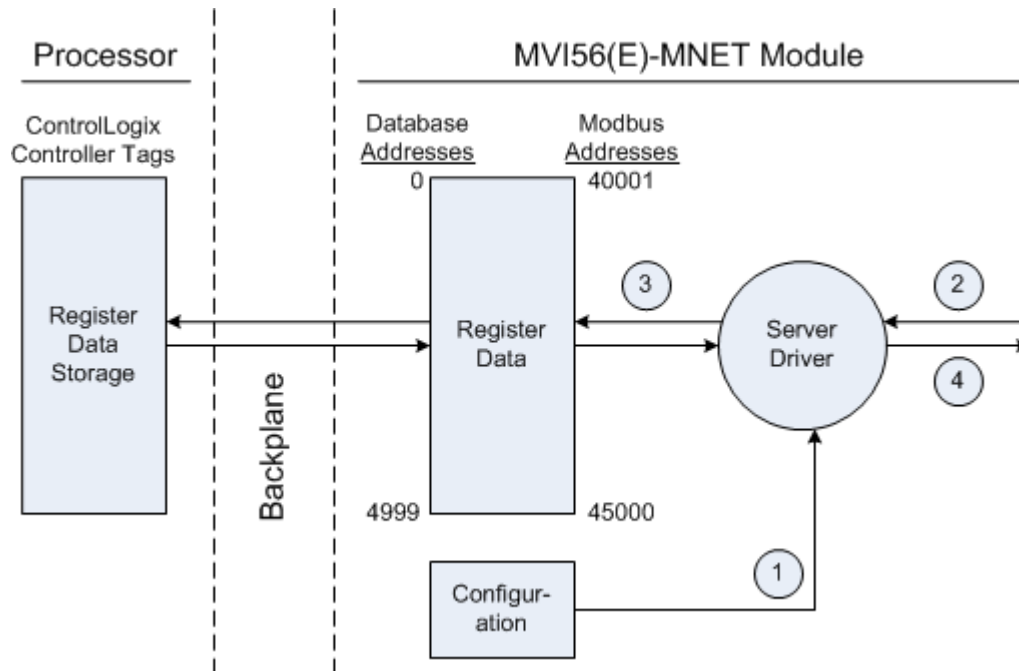
5.4 Data Flow between the MVI56-MNET Module and ControlLogix Processor

The following topics describe the flow of data between the two pieces of hardware (ControlLogix processor and MVI56-MNET module) and other nodes on the Modbus TCP/IP network under the module's different operating modes. The module contains a server and a Client. The server accepts 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.



5.4.1 Server Driver

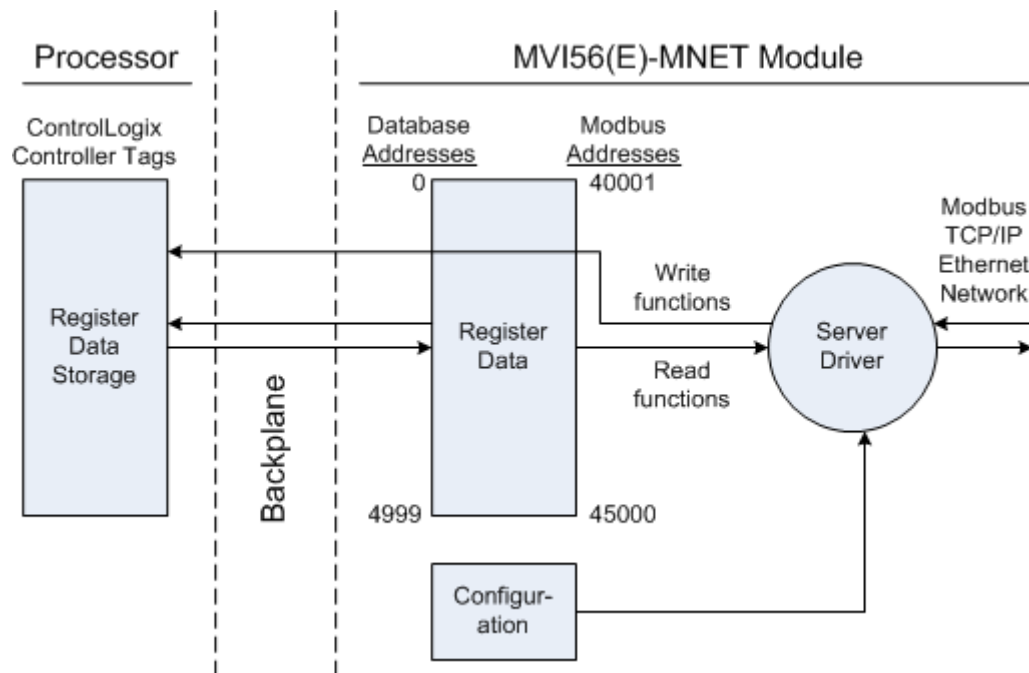
The server driver allows the MVI56-MNET module to respond to data read and write commands issued by Clients on the Modbus TCP/IP network. The following illustration describes the flow of data into and out of the module.



- 1** The server driver receives the configuration information from the configuration file on the Personality Module (compact flash card), and the module initializes the server.
- 2** A host 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 validates the message before accepting it into the module. If the message is considered invalid, an error response is returned to the originating Client node.
- 3** After the module accepts the command, the module processes the data contained in the command.
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 command is a write command and the pass-through feature is utilized, the write message is transferred to the processor ladder logic and is not written directly into the module's database, unless it is returned as a change in the output image that overwrites data in the *WriteData* area as a result of such ladder logic processing.
- 4** After the data processing has been completed in Step 3, a response is issued to the originating Client node.

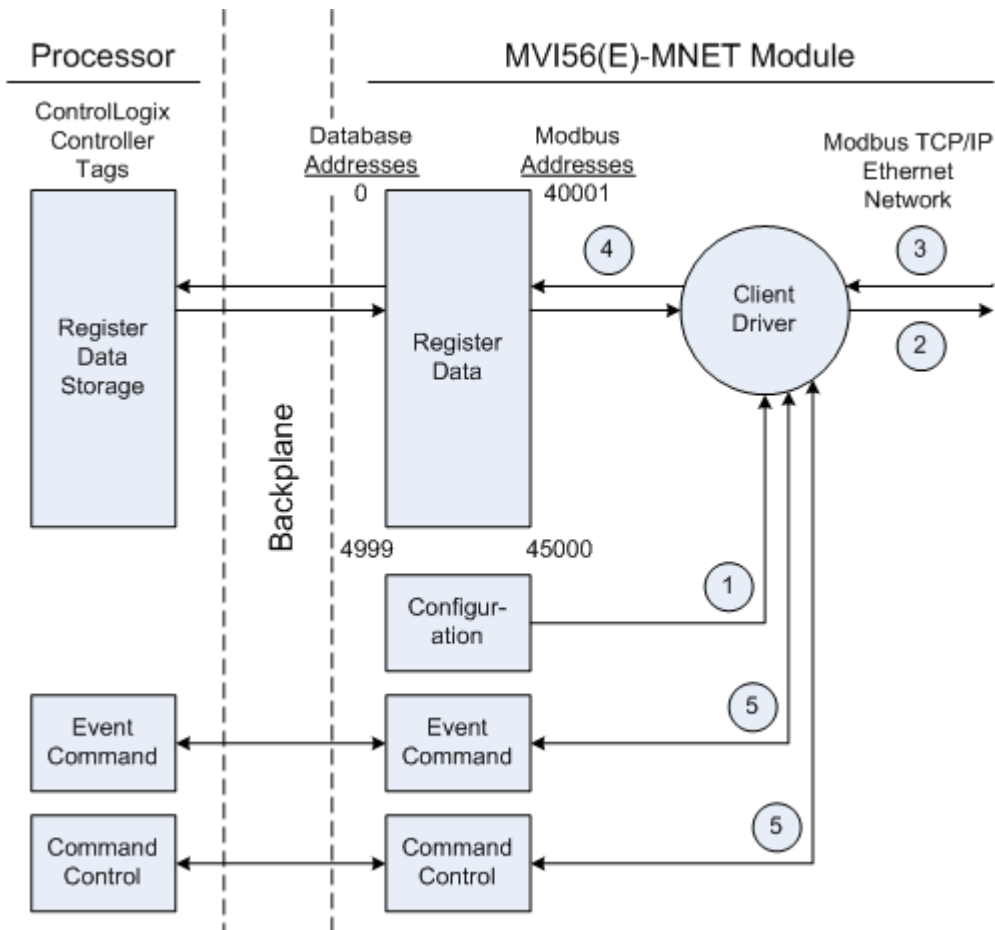
Counters are available in the Status Block that permit the ladder logic program to determine the level of activity of the server driver.

An exception to normal processing is when the pass-through mode is implemented. In this mode, all write requests are passed directly to the processor and are not placed in the database. This permits direct, remote control of the processor without changes in the intermediate database. This mode is especially useful for Client devices that do not send both states of control. For example, a SCADA system may only send a SET command to a digital control point and never send a CLEAR command to that same digital point address because it expects the processor logic to reset the control bit. Pass-through must be used to simulate this mode. The following illustration shows the data flow for a server port with pass-through enabled.



5.4.2 Client Driver

In the Client driver, the MVI56-MNET 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 - 1/10th 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 Illeg	al function
2	Illegal data address
3	Illegal data value
4	Failure in associated device
5 Ackno	wledge
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 T	oo 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.

5.5 Cable Connections

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

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

5.5.1 Ethernet Connection

The MVI56-MNET 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.

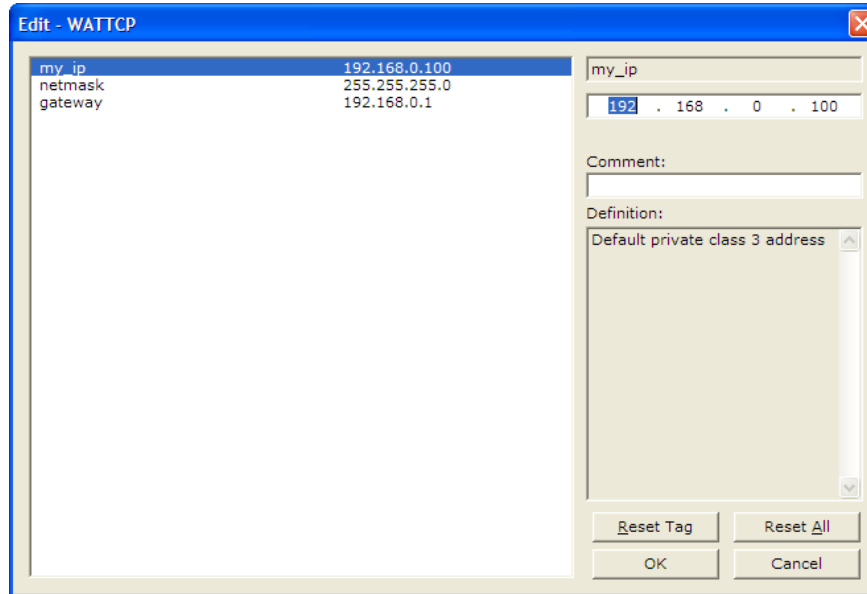
Note: Depending on hardware configuration, you may see more than one RJ45 port on the module. The Ethernet port is labeled *Ethernet*.

Warning: The MVI56-MNET 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

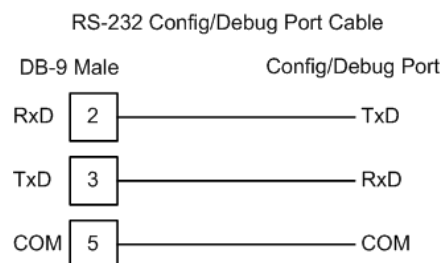
The wattcp.cfg file must be set up properly in order to use a TCP/IP network connection. You can view the current network configuration in *ProSoft Configuration Builder (PCB)*, as shown:



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 69).

5.5.2 RS-232 Configuration/Debug Port

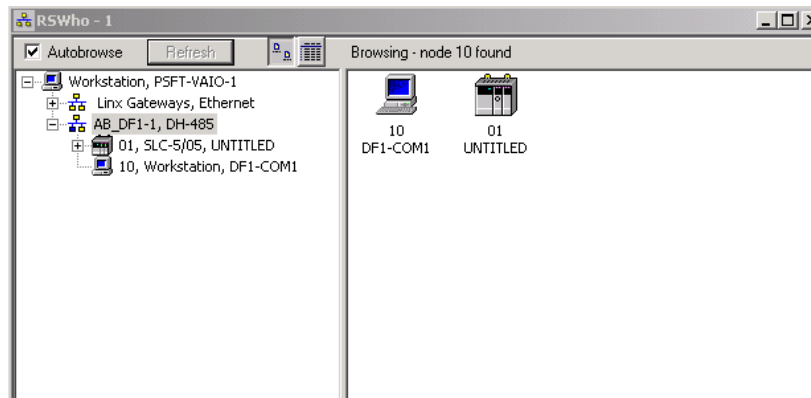
This port is physically an RJ45 connection. An RJ45 to DB-9 adapter cable is included with the module. This port permits a PC based terminal emulation program to view configuration and status data in the module and to control the module. The cable for communications on this port is shown in the following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



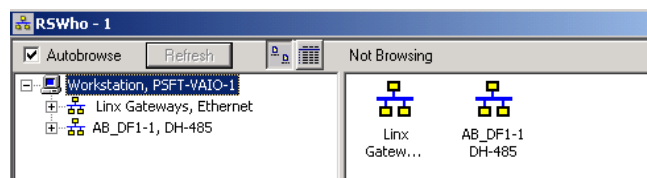
Disabling the RSLinx Driver for the Com Port on the PC

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:



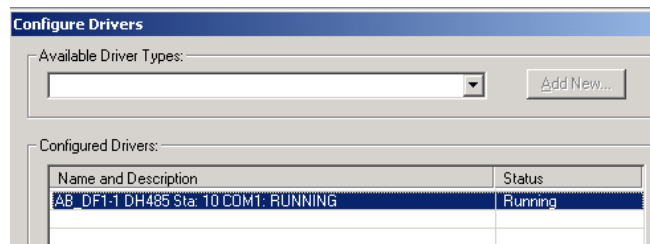
- 3 Notice how the DF1 driver is opened, and the driver is looking for a processor on node 1. If the network is being browsed, then you will not be able to stop this driver. To stop the driver your *RSWho* screen should look like this:



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



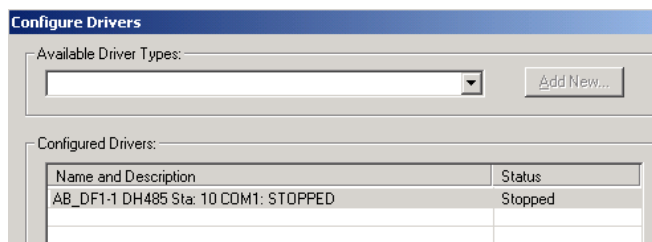
- 4 When you have verified that the driver is not being browsed, go to **COMMUNICATIONS>CONFIGURE DRIVERS**
You may see something like this:



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:



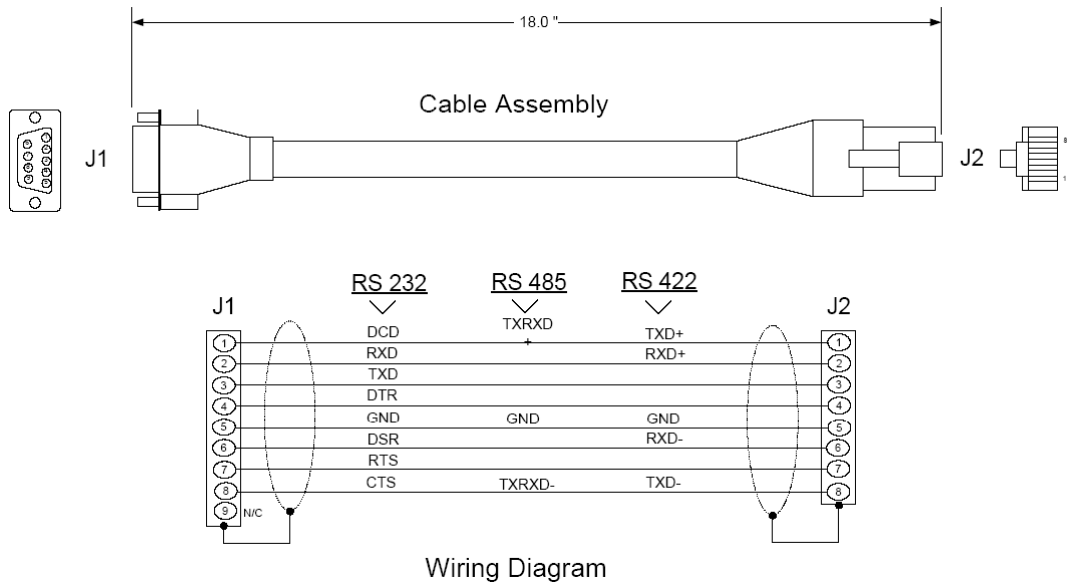
- 5 After you have stopped the driver you will see the following:



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

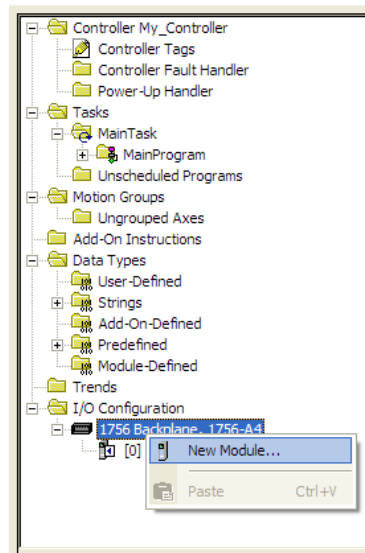
Note: You may need to shut down and restart your PC before it will allow you to stop the driver (usually only on *Windows NT* machines). If you have followed all of the above steps, and it will not stop the driver, then make sure you do not have *RSLogix* open. If *RSLogix* is not open, and you still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

5.5.3 DB9 to RJ45 Adaptor (Cable 14)

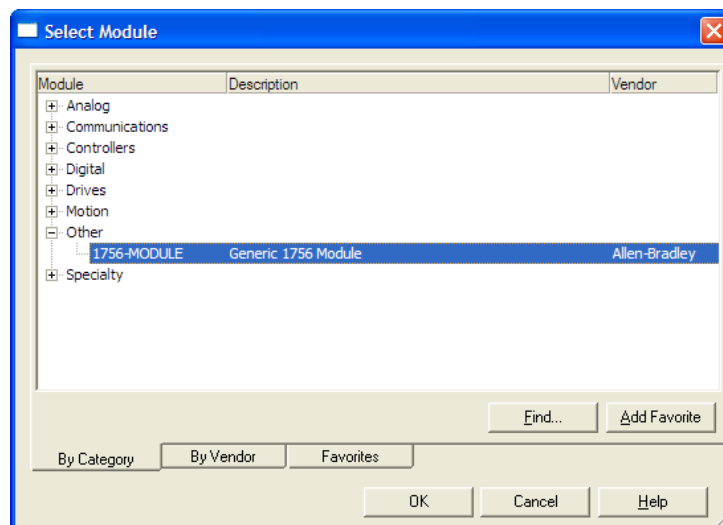


5.6 Adding the Module to an Existing Project

- 1 Select the *I/O Configuration* folder in the *Controller Organization* window of RSLogix 5000, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.



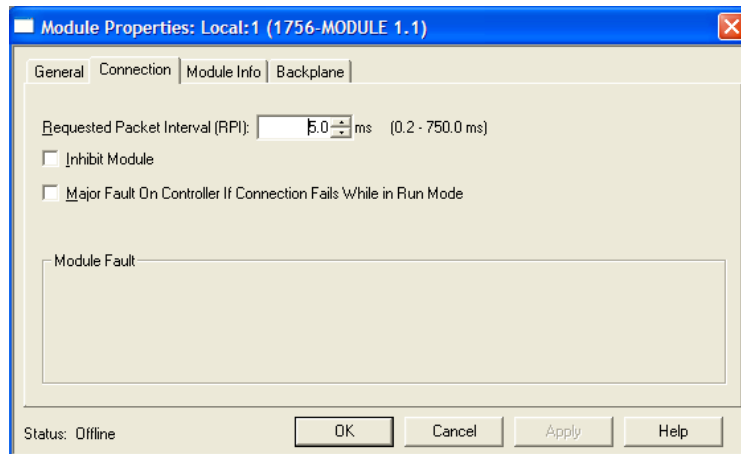
This action opens the *Select Module* dialog box:



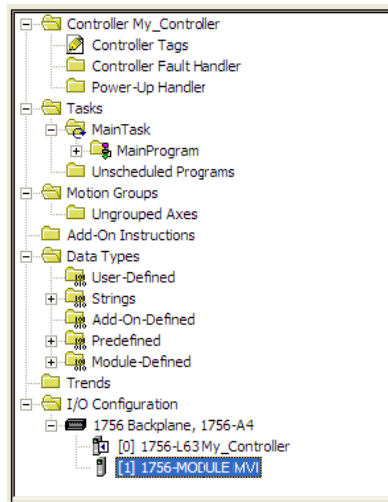
- 2 Select the **1756-MODULE (GENERIC 1756 MODULE)** from the list and click **OK**. This action opens the *New Module* dialog box.
- 3 Enter the *Name*, *Description* and *Slot* options for your application. You must select the *Comm Format* as **DATA - INT** in the dialog box, otherwise the module will not communicate. Click **OK** to continue.

Parameter	Value
Name	Enter a module identification string. Example: MNET_2
Description	Enter a description for the module. Example: MODBUS TCP/IP INTERFACE MODULE
Comm Format	Select DATA-INT.
Slot	Enter the slot number in the rack where the MVI56-MNET module is located.
Input Assembly Instance	1
Input Size	250
Output Assembly Instance	2
Output Size	248
Configuration Assembly Instance	4
Configuration Size	0

- 4 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.



- 5 Save the module. Click **OK** to dismiss the dialog box. The *Controller Organization* window now displays the module's presence.



- 6 Copy the *User-Defined Data Types* from the sample program into your existing RSLogix 5000 project.
- 7 Copy the *Controller Tags* from the sample program into your project.
- 8 Copy the *Ladder Rungs* from the sample program into your project.

5.7 Using the Sample Program

If your processor uses RSLogix 5000 version 15 or earlier, you will not be able to use the Add-On Instruction for your module. Follow the steps below to obtain and use a sample program for your application.

5.7.1 Opening the Sample Program in RSLogix

The sample program for your MVI56-MNET module includes custom tags, data types and ladder logic for data I/O, status and command control. For most applications, you can run the sample program without modification, or, for advanced applications, you can incorporate the sample program into your existing application.

Download the manuals and sample program from the ProSoft Technology web site

You can always download the latest version of the sample ladder logic and user manuals for the MVI56-MNET module from the ProSoft Technology website, at www.prosoft-technology.com/prosoft/support/downloads (<http://www.prosoft-technology.com/prosoft/support/downloads>)

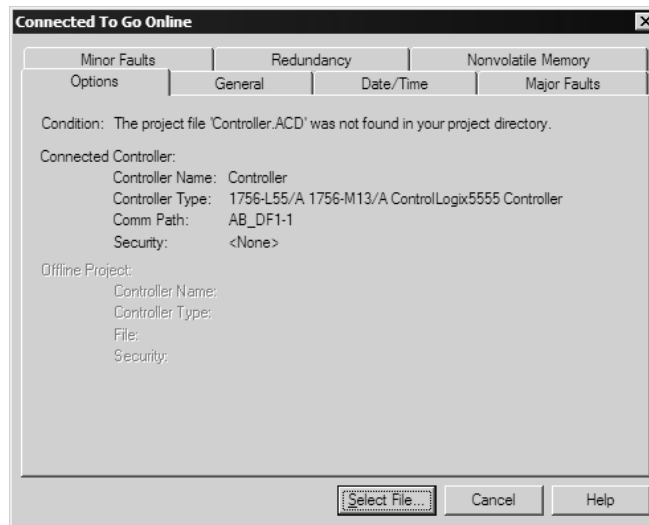
From that link, navigate to the download page for your module and choose the sample program to download for your version of RSLogix 5000 and your processor.

To determine the firmware version of your processor

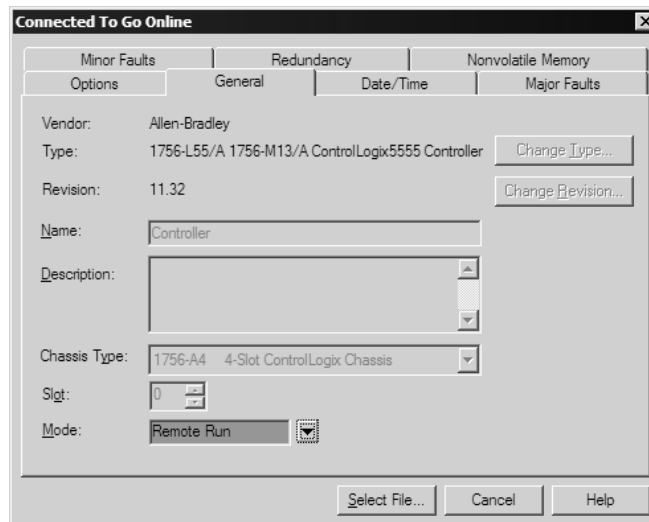
Important: The RSLinx service must be installed and running on your computer in order for RSLogix to communicate with the processor. Refer to your RSLinx and RSLogix documentation for help configuring and troubleshooting these applications.

- 1 Connect an RS-232 serial cable from the COM (serial) port on your PC to the communication port on the front of the processor.
- 2 Start RSLogix 5000 and close any existing project that may be loaded.
- 3 Open the **COMMUNICATIONS** menu and choose **Go ONLINE**. RSLogix will establish communication with the processor. This may take a few moments.

- 4 When RSLogix has established communication with the processor, the *Connected To Go Online* dialog box will open.



- 5 In the *Connected To Go Online* dialog box, click the **GENERAL** tab. This tab shows information about the processor, including the Revision (firmware) version. In the following illustration, the firmware version is 11.32



- 6 Select the sample ladder logic file for your firmware version.

To open the sample program

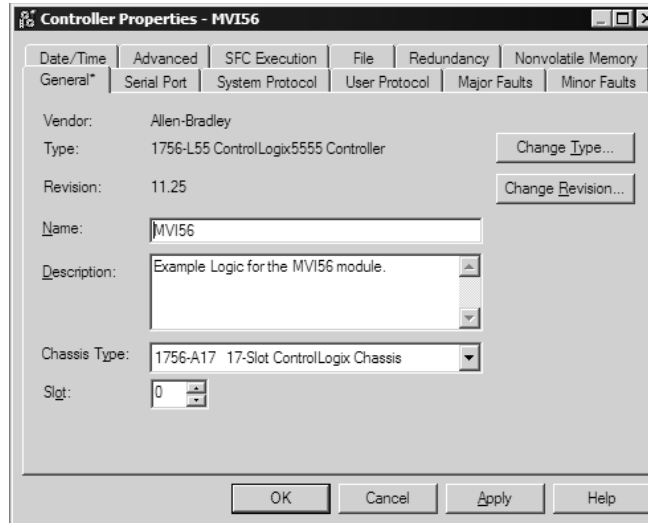
- 1 On the *Connected to Go Online* dialog box, click the **SELECT FILE** button.
- 2 Choose the sample program file that matches your firmware version, and then click the **SELECT** button.
- 3 RSLogix will load the sample program.

The next step is to configure the correct controller type and slot number for your application.

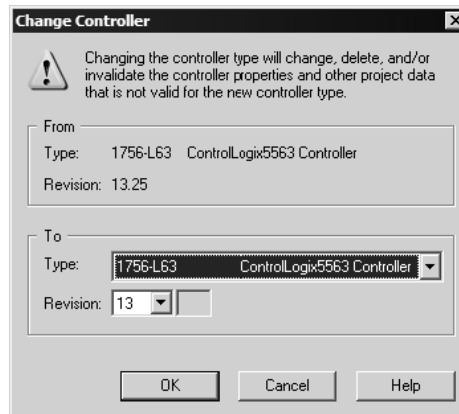
5.7.2 Choosing the Controller Type

The sample application is for a 1756-L63 ControlLogix 5563 Controller. If you are using a different model of the ControlLogix processor, you must configure the sample program to use the correct processor model.

- 1 In the *Controller Organization* list, select the folder for the controller and then click the right mouse button to open a shortcut menu.
- 2 On the shortcut menu, choose **PROPERTIES**. This action opens the *Controller Properties* dialog box.



- 3 Click the **CHANGE TYPE** or **CHANGE CONTROLLER** button. This action opens the *Change Controller* dialog box.



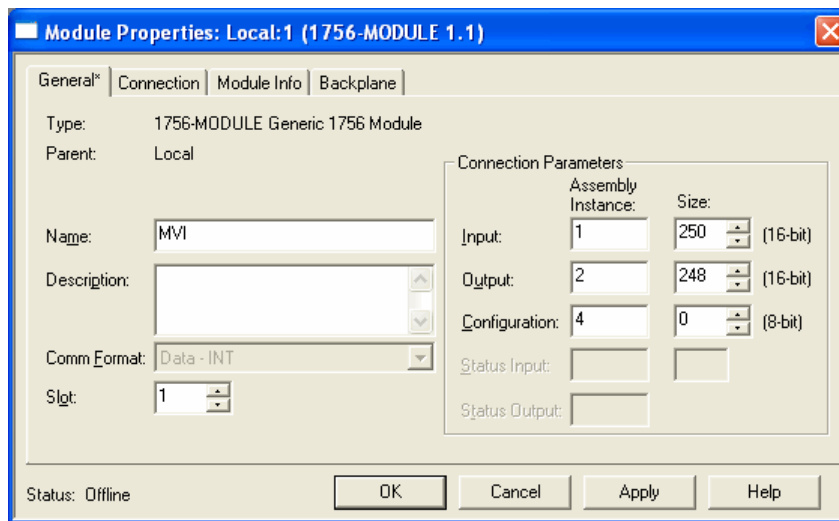
- 4 Open the **TYPE** dropdown list, and then select your ControlLogix controller.
- 5 Select the correct firmware revision for your controller, if necessary.
- 6 Click **OK** to save your changes and return to the previous window.

5.7.3 Selecting the Slot Number for the Module

The sample application is for a module installed in Slot 1 in a ControlLogix rack. The ladder logic uses the slot number to identify the module. If you are installing the module in a different slot, you must update the ladder logic so that program tags and variables are correct, and do not conflict with other modules in the rack.

To change the slot number

- 1 In the *Controller Organization* list, select the module, and then click the right mouse button to open a shortcut menu.
- 2 On the shortcut menu, choose **PROPERTIES**. This action opens the *Module Properties* dialog box.



- 3 In the **SLOT** field, use the up and down arrows on the right side of the field to select the slot number where the module will reside in the rack, and then click **OK**.

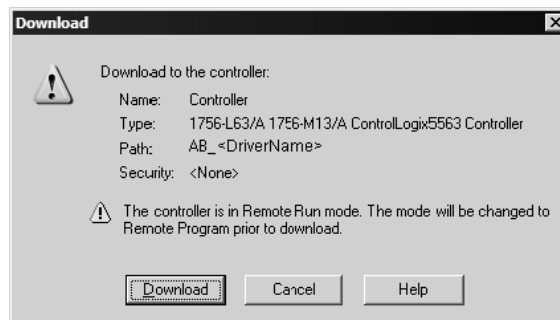
RSLogix will automatically apply the slot number change to all tags, variables and ladder logic rungs that use the MVI56-MNET slot number for computation.

5.7.4 Downloading the Sample Program to the Processor

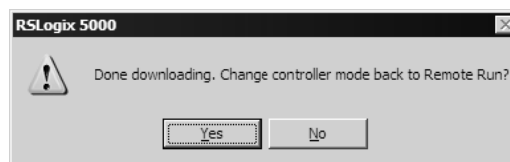
To download the sample program from RSLogix 5000 to the ControlLogix processor

Note: The key switch on the front of the ControlLogix module must be in the REM position.

- 1 If you are not already online to the processor, open the **COMMUNICATIONS** menu, and then choose **DOWNLOAD**. RSLogix will establish communication with the processor.
- 2 When communication is established, RSLogix will open a confirmation dialog box. Click the **DOWNLOAD** button to transfer the sample program to the processor.



- 3 RSLogix will compile the program and transfer it to the processor. This process may take a few minutes.
- 4 When the download is complete, RSLogix will open another confirmation dialog box. 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.

5.7.5 Adding the Sample Ladder to an Existing Application

- 1** Copy the Controller Tags (page 59) from the sample program.
- 2** Copy the User-Defined Data Types (page 61) from the sample program.
- 3** Copy the Ladder Rungs from the sample program.
- 4** Save and Download (page 33, page 129) the new application to the controller and place the processor in RUN mode.

6 Support, Service & Warranty

In This Chapter

- ❖ Contacting Technical Support 131
- ❖ Return Material Authorization (RMA) Policies and Conditions..... 133
- ❖ LIMITED WARRANTY 135

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.

Internet	Web Site: www.prosoft-technology.com/support E-mail address: support@prosoft-technology.com
Asia Pacific (location in Malaysia)	Tel: +603.7724.2080, E-mail: asiapc@prosoft-technology.com Languages spoken include: Chinese, English
Asia Pacific (location in China)	Tel: +86.21.5187.7337 x888, E-mail: asiapc@prosoft-technology.com Languages spoken include: Chinese, English
Europe (location in Toulouse, France)	Tel: +33 (0) 5.34.36.87.20, E-mail: support.EMEA@prosoft-technology.com Languages spoken include: French, English

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

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 135). 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 131). 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:

- o 3150 - All
- o 3750
- o 3600 - All
- o 3700
- o 3170 - All
- o 3250
- o 1560 - Can be repaired, only if defect is the power supply
- o 1550 - Can be repaired, only if defect is the power supply
- o 3350
- o 3300
- o 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 135) 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

A

About the MODBUS TCP/IP Protocol • 90
Adding Multiple Modules (Optional) • 26
Adding the Module to an Existing Project • 122
Adding the Sample Ladder to an Existing Application • 130
Adjusting the Input and Output Array Sizes (Optional) • 31
Allocation of Risks • 138
ARP Timeout • 43

B

Backplane Data Transfer • 91
Battery Life Advisory • 3
Bit Input Offset • 52

C

Cable Connections • 117
Choosing the Controller Type • 127
Clearing a Fault Condition • 71
Client Command Errors • 115
Client Command List • 115
Client Driver • 114
Cold Boot Block (9999) • 110
Command Control Blocks (5001 to 5006) • 103
Command Entry Formats • 45
Command Error Delay • 43
Command Error Pointer • 41
Command List Entry Errors • 116
Command List Menu • 76, 81
Command List Overview • 44
Commands Supported by the Module • 45
Comment • 50
Configuration Error Word • 66, 85
Configuring Module Parameters • 36
Configuring the MVI56-MNET Module • 19
Connecting Your PC to the ControlLogix Processor • 32
Connecting your PC to the Module • 56
Connection Timeout • 53
Contacting Technical Support • 131, 133
Controller Tag Overview • 63
Controller Tags • 59, 130
Controlling Law and Severability • 139
Creating a New RSLogix 5000 Project • 20
Creating Optional Comment Entries • 37
Creating the Module • 20

D

Data Flow between the MVI56-MNET Module and ControlLogix Processor • 111

DB9 to RJ45 Adaptor (Cable 14) • 121
Diagnostics and Troubleshooting • 9, 69, 118
Disabling the RSLinx Driver for the Com Port on the PC • 119
Disclaimer of all Other Warranties • 137
Disclaimer Regarding High Risk Activities • 136
Downloading the Project to the Module Using a Serial COM port • 57
Downloading the Sample Program to the Processor • 33, 129, 130
Duplex/Speed Code • 40

E

Enable • 46
Error/Status Pointer • 38, 41, 84
Ethernet Configuration • 55
Ethernet Connection • 117
Ethernet LED Indicators • 70
Ethernet Port Configuration - wattcp.cfg • 118
Event Command Blocks (2000) • 101
Exiting the Program • 78

F

Failure Flag Count • 39
Float Flag • 42, 51
Float Offset • 43, 52
Float Start • 42, 52
Functional Specifications • 88

G

General Specifications • 87
Get Module IP Address Block (9991) • 109
Guide to the MVI56-MNET User Manual • 9

H

Hardware MAC Address • 54
Hardware Specifications • 89
Holding Register Offset • 53
How to Contact Us • 2

I

Importing the Add-On Instruction • 23
Initialize Output Data • 39
Initialize Output Data Blocks (1000 to 1024) • 100
Installing ProSoft Configuration Builder Software • 14
Installing the Module in the Rack • 16
Intellectual Property Indemnity • 137
Internal Address • 47
IP Address • 54

K

Keystrokes • 74

L

Ladder Logic • 59
LED Indicators • 70
Limitation of Remedies ** • 138

LIMITED WARRANTY • 133, 135

M

Main Menu • 75
Markings • 4
MB Address in Device • 50
Minimum Command Delay • 41
MNET Client Specific Errors • 116
MNET Client x • 41
MNET Client x Commands • 44, 101, 103
MNET Servers • 51
MNET.CONTROL • 67
MNET.DATA • 63
MNET.STATUS • 66, 84
MNET.UTIL • 68
Modbus Database View Menu • 76, 79
Modbus Function • 49
Modbus TCP/IP • 88
Module • 38
Module Communication Error Codes • 116
Moving Back Through 5 Pages of Registers • 79, 81
Moving Forward Through 5 Pages of Registers • 80, 81
MVI56(E)-MNET Controller Tags • 60
MVI56(E)-MNET User-Defined Data Types • 61
MVI56-MNET Status Data Area • 95

N

Navigation • 74
Network Menu • 78, 82
No Other Warranties • 138
Node IP Address • 48, 49
Normal Data Transfer • 94

O

Opening the Command Error List Menu • 76
Opening the Command List Menu • 76
Opening the Database View Menu • 76
Opening the Network Menu • 78
Opening the Sample Program in RSLogix • 125
Output Offset • 52

P

Package Contents • 13
Pass-Through Blocks (9956-9961 and 9970) • 106
Pass-Through Mode • 40
Pinouts • 3, 117, 121
Poll Interval • 47
Printing a Configuration File • 37
Product Specifications • 9, 87
ProSoft Technology® Product Documentation • 2

R

Read Block • 38, 41, 94
Read Register Count • 38, 64
Read Register Start • 38
Reading Status Data from the Module • 84
Receiving the Configuration File • 76

Redisplaying the Current Page • 79, 81
Redisplaying the Menu • 81
Reference • 9, 87
Reg Count • 47
Renaming an Object • 36
Resetting Diagnostic Data • 76
Response Timeout • 41
Retry Count • 42
Return Material Authorization (RMA) Policies and Conditions • 133
Returning Any Product • 133
Returning to the Main Menu • 80, 81, 83
Returning Units Out of Warranty • 134
Returning Units Under Warranty • 134
RS-232 Configuration/Debug Port • 118

S

Sample Add-On Instruction Import Procedure • 19
Select Priority Read Block (Write Block Offset 247) • 98
Selecting the Slot Number for the Module • 128
Sending the Configuration File • 76
Server Driver • 112
Service Port • 48
Set Module IP Address Block (9990) • 109
Setting Jumpers • 15
Setting Module Parameters • 36
Setting Up the Project • 34
Slave Address • 49
Special Function Blocks • 68, 100
Standard Modbus Exception Code Errors • 116
Start Here • 9, 11
Static ARP Table • 54
Support, Service & Warranty • 9, 131
Swap Code • 48
System Requirements • 12

T

Time Limit for Bringing Suit • 138
Transferring WATTCP.CFG to the Module • 82
Transferring WATTCP.CFG to the PC • 82
Troubleshooting • 71

U

User-Defined Data Types (UDTs) • 61, 130
Using Controller Tags • 62
Using ProSoft Configuration Builder • 34
Using ProSoft Configuration Builder (PCB) for Diagnostics • 72
Using the Diagnostic Window in ProSoft Configuration Builder • 72
Using the Sample Program • 19, 125

V

Viewing Block Transfer Statistics • 75
Viewing Client Configuration • 77
Viewing Client Status • 77
Viewing Data in ASCII (Text) Format • 80

Viewing Data in Decimal Format • 9, 80
Viewing Data in Floating-Point Format • 80
Viewing Data in Hexadecimal Format • 80
Viewing Module Configuration • 75
Viewing Network Status • 77
Viewing NIC Status • 77
Viewing Register Pages • 79
Viewing Server Configuration • 77
Viewing the Next Page of Commands • 81
Viewing the Next Page of Registers • 80
Viewing the Previous Page of Commands • 81
Viewing the Previous Page of Registers • 80
Viewing the Static ARP Table • 78
Viewing the WATTCP.CFG File on the module • 83
Viewing Version Information • 76

W

Warm Boot Block (9998) • 110
Warm Booting the Module • 77
Warnings • 3
What Is Covered By This Warranty • 135, 137
What Is Not Covered By This Warranty • 136
Word Input Offset • 53
Write Block • 97
Write Register Count • 39, 65
Write Register Start • 39

Y

Your Feedback Please • 2