ProSoft TECHNOLOGY

Where Automation Connects.

inRAx®

MVI56E-MNETR

ControlLogix Platform

Modbus TCP/IP Interface Module with Reduced Data Block

August 31, 2009

USER MANUAL

Important Safety Information - MVI56E Modules

North America Warnings

A Warning - Explosion Hazard - Substitution of components may impair suitability for Class I, Division 2.
B Warning - Explosion Hazard - When in Hazardous Locations, turn off power before replacing or rewiring modules.
Warning - Explosion Hazard - Do not disconnect equipment unless power has been switched off or the area is known to be nonhazardous.
C Suitable for use in Class I, Division 2 Groups A, B, C, and D, T5 Hazardous Locations or Non-Hazardous Locations.

ATEX Warnings and Conditions of Safe Usage

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

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

Electrical Ratings

- Backplane Current Load: 800 mA @ 5 V DC; 3mA @ 24V DC
- Operating Temperature: 0 to 60°C (32 to 140°F)
- Storage Temperature: -40 to 85°C (-40 to 185°F)
- Shock: 30g Operational; 50g non-operational; Vibration: 5 g from 10 to 150 Hz
- Relative Humidity 5% to 95% (non-condensing)
- All phase conductor sizes must be at least 1.3 mm (squared) and all earth ground conductors must be at least 4mm (squared).

Markings

ANSI / ISA	ISA 12.12.01 Class I Division 2, GPs A, B, C, D
CSA/cUL	C22.2 No. 213-M1987
CSA CB Certified	IEC61010
ATEX	EN60079-0 Category 3, Zone 2 EN60079-15

243333 E183151

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

Temp Code T5

II 3 G

Ex nA nL IIC T5 X

0° C <= Ta <= 60° C

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

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

G – Equipment protected against explosive gasses.

Battery Life Advisory

The module uses a rechargeable Lithium Vanadium Pentoxide battery to backup the real-time clock and CMOS settings. The battery itself should last for the life of the module. However, if left in an unpowered state for 14 to 21 days, the battery may become fully discharged and require recharging by being placed in a powered-up ControlLogix chassis. The time required to fully recharge the battery may be as long as 24hours.

Once it is fully charged, the battery provides backup power for the CMOS setup and the real-time clock for approximately 21 days. Before you remove a module from its power source, ensure that the battery within the module is fully charged (the BATT LED on the front of the module goes OFF when the battery is fully charged. If the battery is allowed to become fully discharged, the module will revert to the default BIOS and clock settings.

Note: The battery is not user-replaceable or serviceable.

Your Feedback Please

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

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

MVI56E-MNETR User Manual
August 31, 2009

ProSoft Technology® Product Documentation

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

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

North America: +1.661.716.5100

Asia Pacific: +603.7724.2080

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

Latin America: +1.281.298.9109

Contents

Important Safety Information - MVI56E Modules 2
Battery Life Advisory 3
Your Feedback Please 3
ProSoft Technology® Product Documentation 3

Guide to the MVI56E-MNETR User Manual 7

1 Start Here 9

1.1 What's Different? 10
1.2 System Requirements 10
1.3 Package Contents 11
1.4 Setting Jumpers 12
1.5 Install the Module in the Rack 12
1.6 Install the Configuration Tools 14
1.7 Connect your PC to the Module 15
1.8 Set Temporary IP Address 15
1.9 Connect to the Module's Web Page 23
1.10 Upload the Add-On Instruction from the Module 25
1.11 Create a new RSLogix 5000 project 26

2 Configuring the MVI56E-MNETR Module 49

2.1 Using ProSoft Configuration Builder Software 49
2.2 Download the Project to the Module 67
2.3 Using CIPconnect® to Connect to the Module 68

3 Ladder Logic 79

3.1 MNETRModuleDef 79
3.2 Modbus Message Data 84

4 Diagnostics and Troubleshooting 85

4.1 Reading Status Data from the Module 85
4.2 The Diagnostics Menu 86
4.3 Monitoring Module Information 89
4.4 Monitoring Backplane Information 90
4.5 Monitoring Database Information 91
4.6 Monitoring MNET Client Information 92
4.7 Monitoring MNET Server Information 93
4.8 Data Analyzer 93
4.9 LED Status Indicators 97

5 Reference 101

5.1 Product Specifications 101

5.2 Functional Overview 103
5.3 Ethernet Cable Specifications 118
5.4 MVI56E-MNETR Status Data Definition 119
5.5 Modbus Protocol Specification 121
5.6 Using the Rung Import with Utility Add-On Instruction 132
5.7 Adding the Module to an Existing Project 141
5.8 Using the Sample Program - RSLogix Version 15 and earlier 144

6 Support, Service & Warranty 145

6.1 How to Contact Us: Technical Support 145
6.2 Return Material Authorization (RMA) Policies and Conditions 146
6.3 LIMITED WARRANTY 147

Index 151

Guide to the MVI56E-MNETR User Manual

Function		Section to Read	Details
Introduction (Must Do)	→	Start Here (page 9)	This Section introduces the customer to the module. Included are: package contents, system requirements, hardware installation, and basic configuration.
Verify Communication, Diagnostic and Troubleshooting	→	Verifying Communication (page 97) Diagnostics and Troubleshooting (page 85)	This section describes how to verify communications with the network. Diagnostic and Troubleshooting procedures.
Reference Product Specifications Functional Overview	→	Reference (page 101) Functional Overview (page 103) Product Specifications (page 101)	These sections contain general references associated with this product, Specifications, and the Functional Overview.
Support, Service, and Warranty Index	→	Support, Service and Warranty (page 145)	This section contains Support, Service and Warranty information. Index of chapters.

1 Start Here

In This Chapter

- What's Different? 10
- System Requirements 10
- Package Contents 11
- Setting Jumpers 12
- Install the Module in the Rack 12
- Install the Configuration Tools 14
- Connect your PC to the Module 15
- Set Temporary IP Address 15
- Connect to the Module's Web Page 23
- Upload the Add-On Instruction from the Module 25
- Create a new RSLogix 5000 project 26

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 MVI56E-MNETR module's application port(s).

Caution: You must be able to complete the application without exposing personnel or equipment to unsafe or inappropriate working conditions.

1.1 What's Different?

Modbus TCP/IP Interface Module with Reduced Data Block products are **backward compatible** with existing MVI56 products in the field, including ladder logic and module configuration files. Easily swap and upgrade products while benefiting from an array of new features designed to improve interoperability and enhance the user experience.

- **Web Page:** The web page allows access to manuals and other tools previously provided on a product CD.
- **ProSoft Configuration Builder (PCB):** New Windows software for diagnostics, connecting via the module's Ethernet port or CIPconnect® to upload/download module configuration information and access troubleshooting features and functions.
- **ProSoft Discovery Service (PDS):** Utility software to find and display a list of all MVI56E modules on the network and to temporarily change the IP address to connect with the module's web page.
- **CIPconnect®-enabled:** Allows PC-to-module configuration and diagnostics from the Ethernet network through a ControlLogix 1756-ENBT EtherNet/IP® module.
- **Personality Card:** An industrial compact flash memory card storing the module's complete configuration and Ethernet settings, allowing quick and easy replacement.
- **LED Scrolling Diagnostic Display:** 4-character, alphanumeric display, providing English messages for status and alarm data, and for processor and network communication status.

1.2 System Requirements

The MVI56E-MNETR module requires the following minimum hardware and software components:

- Rockwell Automation® ControlLogix™ processor (firmware version 10 or higher), with compatible power supply, and one free slot in the rack for the MVI56E-MNETR module. The module requires 800mA of available 5 VDC power
- Rockwell Automation RSLogix 5000 programming software
  - Version 16 or higher required for Add-On Instruction
  - Version 15 or lower must use Sample Ladder, available from www.prosoft-technology.com
- Rockwell Automation RSLinx communication software version 2.51 or higher
- ProSoft Configuration Builder (PCB) (included)
- ProSoft Discovery Service (PDS) (included in PCB)
- Pentium® II 450 MHz minimum. Pentium III 733 MHz (or better) recommended
- Supported operating systems:
  - Microsoft Windows Vista
  - 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

Note: The Hardware and Operating System requirements in this list are the minimum recommended to install and run software provided by ProSoft Technology. Other third party applications may have different minimum requirements. Refer to the documentation for any third party applications for system requirements.
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.

1.3 Package Contents

The following components are included with your MVI56E-MNETR module, and are all required for installation and configuration.

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

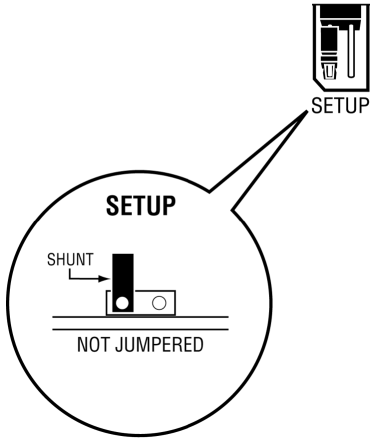
Qty.	Part Name	Part Number	Part Description
1	MVI56E-MNETR Module	<ModelNumbeXTr>	Modbus TCP/IP Interface Module with Reduced Data Block
1	Cable	RL-CBL025	5 foot Ethernet Straight-Through Cable (Gray)
1	ProSoft Solutions CD	CD-013	Contains configuration tools for the MVI56E-MNETR module
1	Insert		MVI56E-MNETR Quick Start Guide

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

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 MVI56E-MNETR jumper configuration.



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

1.5 Install 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 MVI56E-MNETR module. Refer to your Rockwell Automation product documentation for installation instructions.

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

After you have checked the placement of the jumpers, insert the MVI56E-MNETR into the ControlLogix chassis. Use the same technique recommended by Rockwell Automation to remove and install ControlLogix modules.

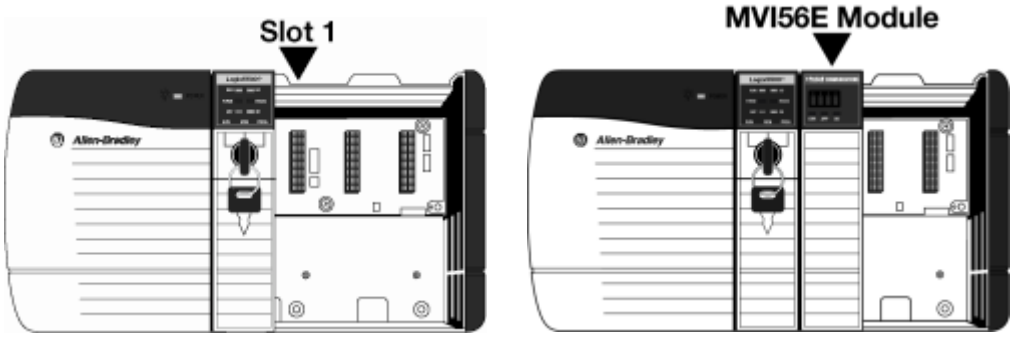
You can install or remove ControlLogix system components while chassis power is applied and the system is operating. However, please note the following warning.

Warning: When you insert or remove the module while backplane power is on, an electrical arc can occur. An electrical arc can cause personal injury or property damage by:

- sending an erroneous signal to your system's actuators causing unintended machine motion or loss of process control
- causing an explosion in a hazardous environment

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. Align the module with the top and bottom guides, and then slide it into the rack until the module is firmly against the backplane connector.



2. With a firm, steady push, snap the module into place.
3. Check that the holding clips on the top and bottom of the module are securely in the locking holes of the rack.
4. 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.
5. Turn power ON.

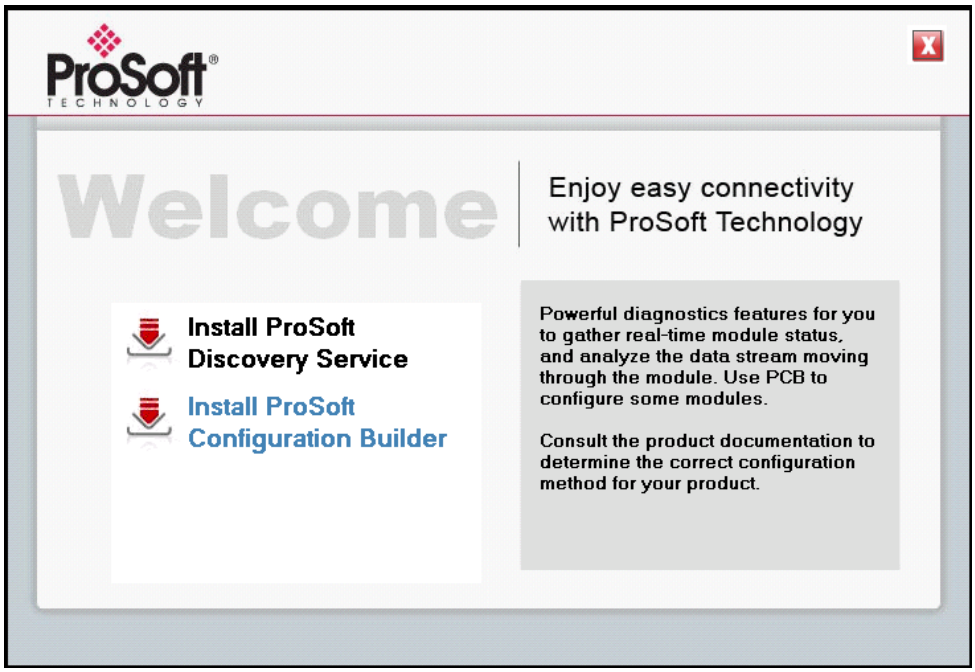
Note: If you insert the module improperly, the system may stop working, or may behave unpredictably.

1.6 Install the Configuration Tools

1.6.1 Install ProSoft Configuration Builder

To install ProSoft Configuration Builder from the CD-ROM

1 Insert the ProSoft Solutions CD-ROM into the CD drive of your PC. Wait for the startup screen to appear.

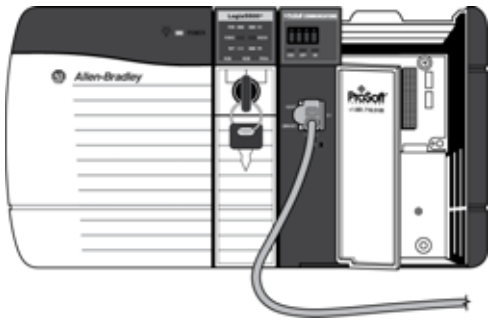


2 On the startup screen, click **INSTALL PROSOFT CONFIGURATION BUILDER.** This action starts the installation wizard for *ProSoft Configuration Builder*.

3 Click **NEXT** on each page of the installation wizard. Click **FINISH** on the last page of the wizard.

1.7 Connect your PC to the Module

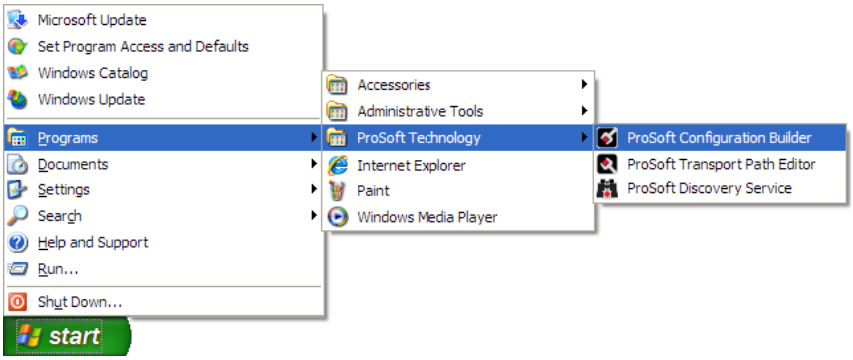
With the module securely mounted, connect one end of the Ethernet cable to the **CONFIG (E1)** Port, and the other end to an Ethernet hub or switch accessible from the same network as your PC. Or, you can connect directly from the Ethernet Port on your PC to the **CONFIG (E1)** Port on the module.



1.8 Set Temporary IP Address

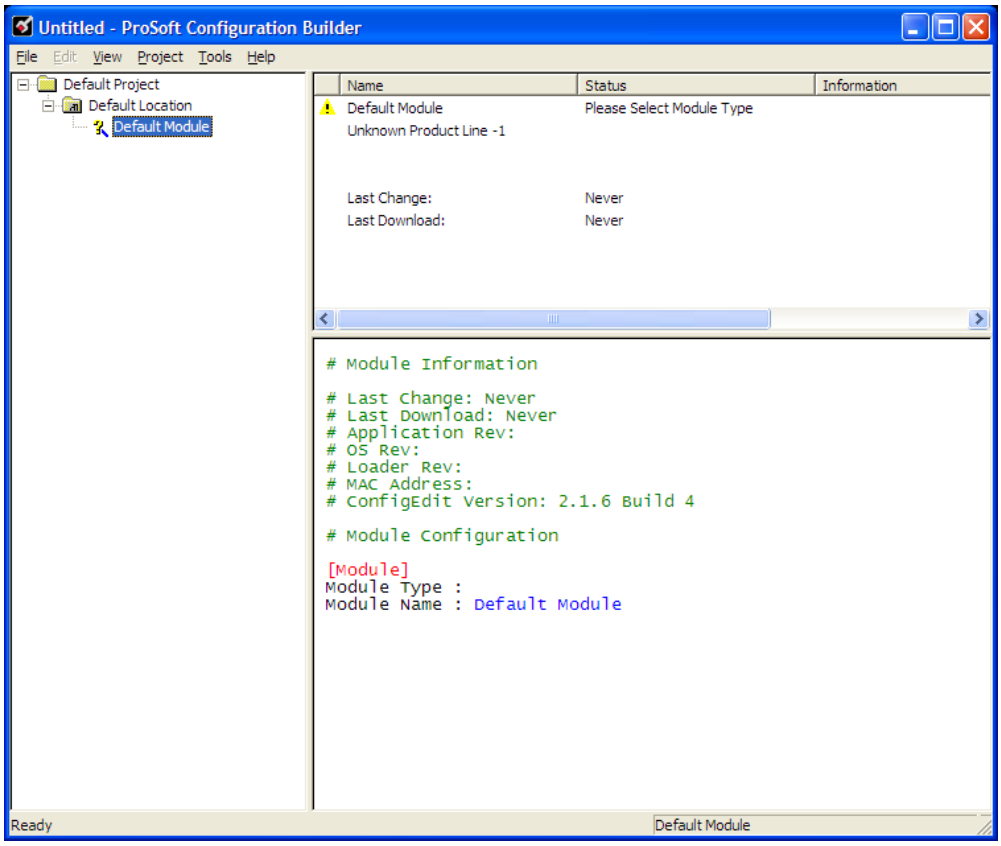
Important: ProSoft Configuration Builder locates MVI56E-MNETR modules through UDP broadcast messages. These messages may be blocked by routers or layer 3 switches. In that case, ProSoft Discovery Service will be unable to locate the modules.
To use ProSoft Configuration Builder, arrange the Ethernet connection so that there is no router/ layer 3 switch between the computer and the module OR reconfigure the router/layer 3 switch to allow the routing of the UDP broadcast messages.

1 Click the **START** button, and then navigate to **PROGRAMS / PROSOFT TECHNOLOGY**



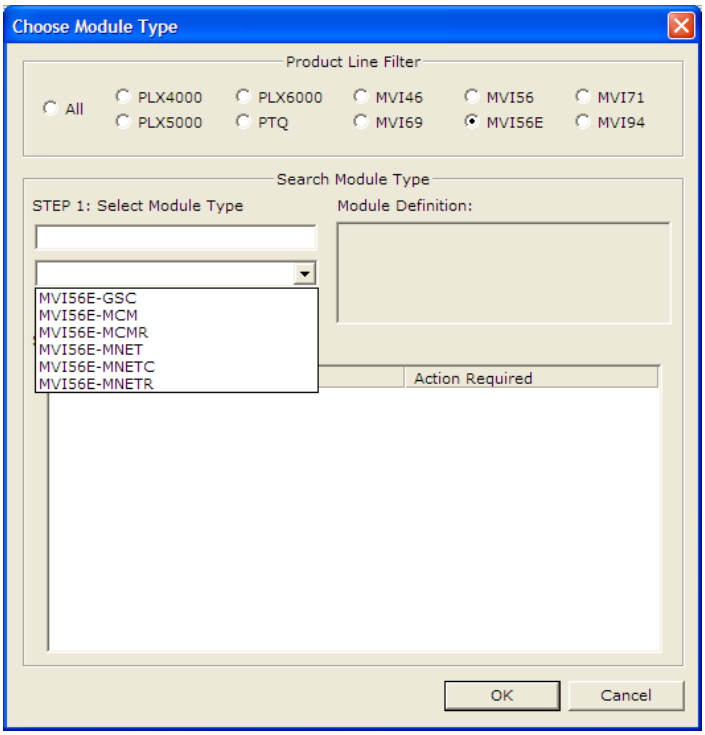
2 Click to start **PROSOFT CONFIGURATION BUILDER**

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.



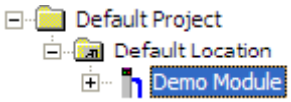
3 Use the mouse to select **DEFAULT MODULE** in the tree view, and then click the right mouse button to open a shortcut menu.

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

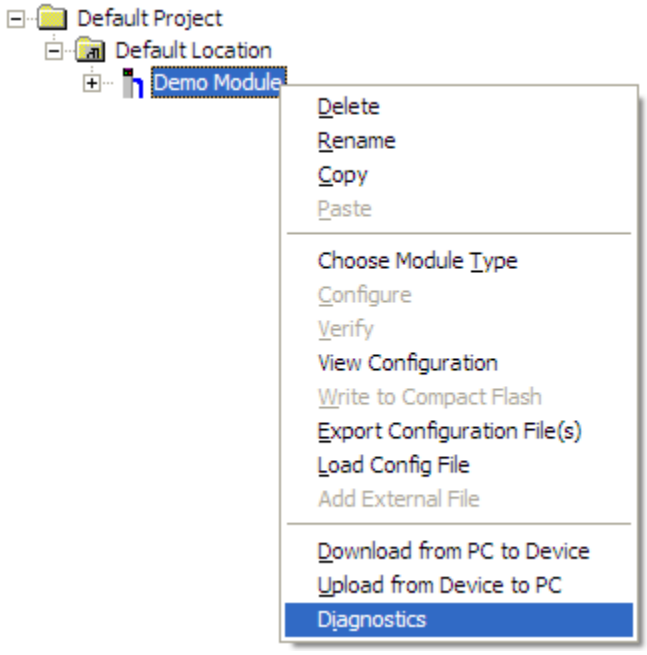


5 In the **PRODUCT LINE FILTER** area of the dialog box, select MVI56E. In the **SELECT MODULE TYPE** dropdown list, select MVI56E-MNETR, and then click **OK** to save your settings and return to the ProSoft Configuration Builder window.

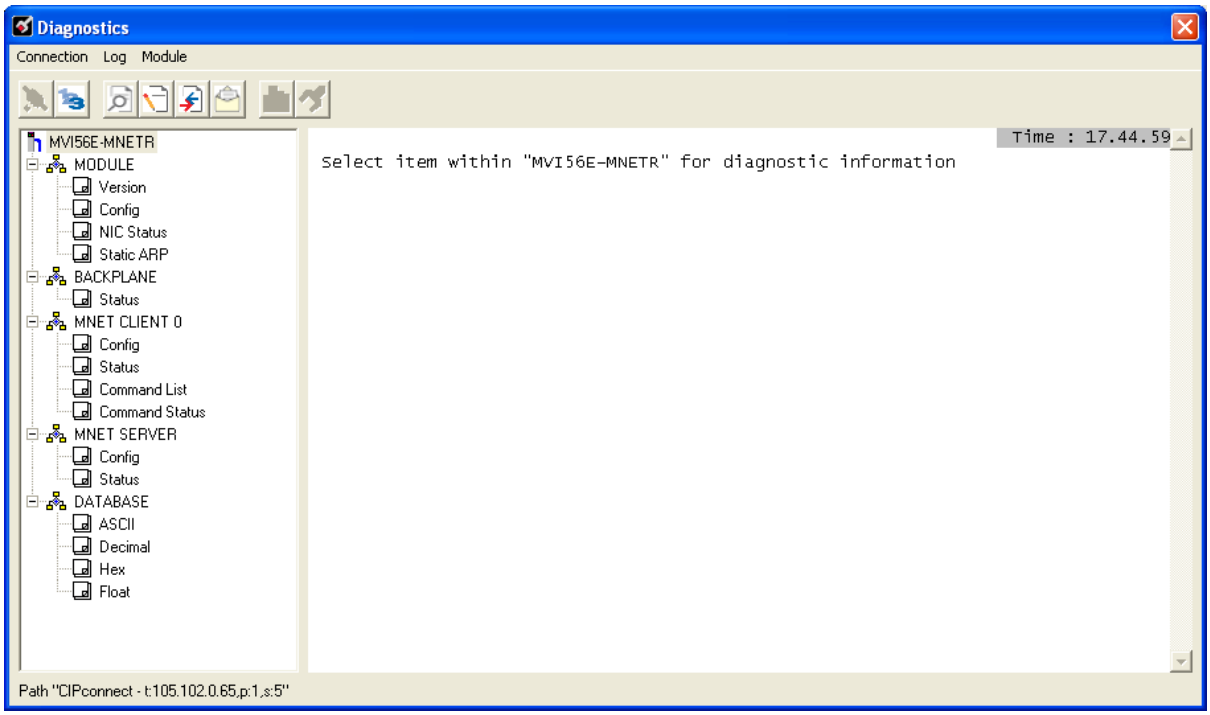
6 Right click over the module icon.



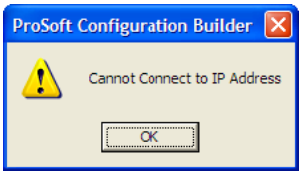
7 On the shortcut menu, choose **DIAGNOSTICS.**



This action opens the **DIAGNOSTICS** dialog box.

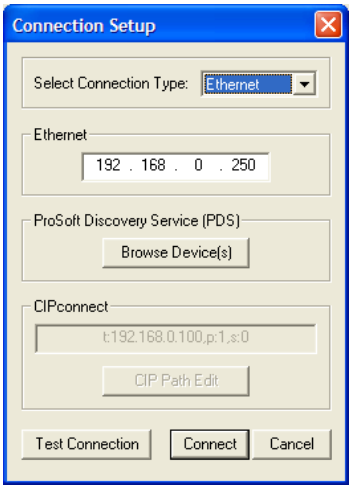


If there is no response from the module,



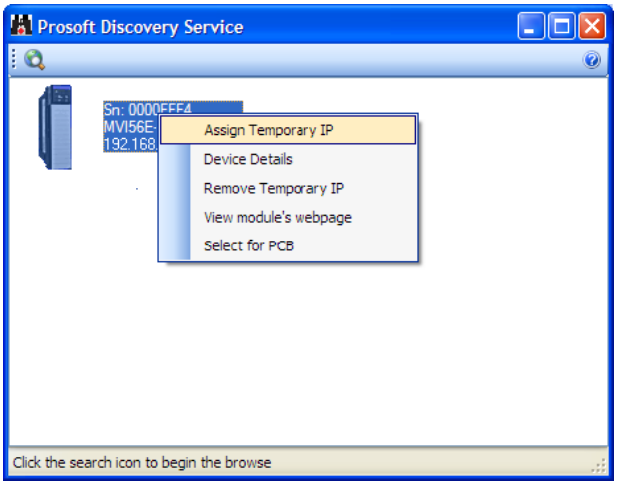
1 Click the **SET UP CONNECTION** button to browse for the module's IP address.

2 On the **CONNECTION SETUP** dialog box, click the **TEST CONNECTION** button to verify if the module is accessible with the current settings

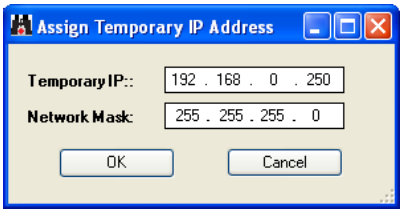


3 If PCB is still unable to connect to the module, click the **BROWSE DEVICE(S)** button to open the **PROSOFT DISCOVERY SERVICE**.

4 Select the module, then right click and choose **ASSIGN TEMPORARY IP**.



5 The module's default IP address is 192.168.0.250.



6 Choose an unused IP within your subnet, and then click **OK.**

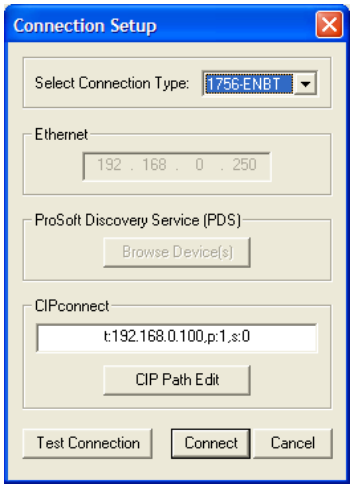
1.8.1 CIPconnect

You can use CIPconnect® to connect a PC to the MVI56E-MNETR module over Ethernet using Rockwell Automation's 1756-ENBT EtherNet/IP® module. This allows you to configure the MVI56E-MNETR module and network, upload and download files, and view network and module diagnostics from a PC. RSLinx is not required when you use CIPconnect. All you need are:

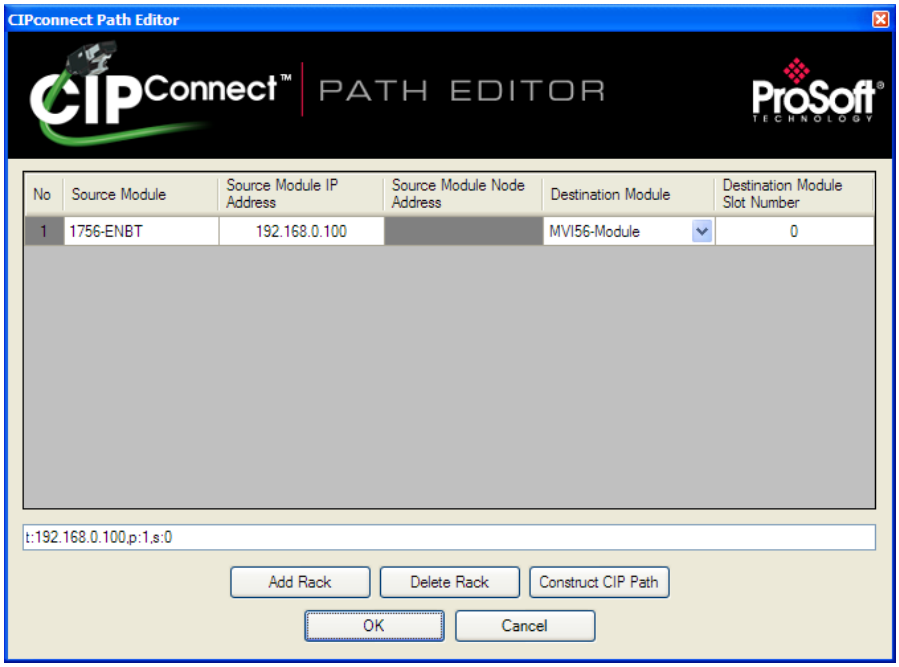
- The IP addresses and slot numbers of any 1756-ENBT modules in the path
- The ControlNet node numbers and slot numbers of any 1756-CNBx ControlNet Bridge modules in the path
- The slot number of the MVI56E-MNETR in the destination ControlLogix chassis (the last ENBT/CNBx and chassis in the path).

To use CIPconnect, follow these steps.

1 In the **SET CONNECTION TYPE** dropdown list, choose 1756-ENBT. The default path appears in the text box, as shown in the following illustration.



2 Click **CIP PATH EDIT** to open the **CIPCONNECT PATH EDITOR** dialog box.



The **CIPCONNECT PATH EDITOR** allows you to define the path between the PC and the MVI56E-MNETR module. The first connection from the PC is always a 1756-ENBT (Ethernet/IP) module.

Each row corresponds to a physical rack in the CIP path.

- If the MVI56E-MNETR module is located in the same rack as the first 1756-ENBT module, select **RACK NO. 1** and configure the associated parameters.
- If the MVI56E-MNETR is available in a remote rack (accessible through ControlNet or Ethernet/IP), include all racks (by using the **ADD RACK** button).

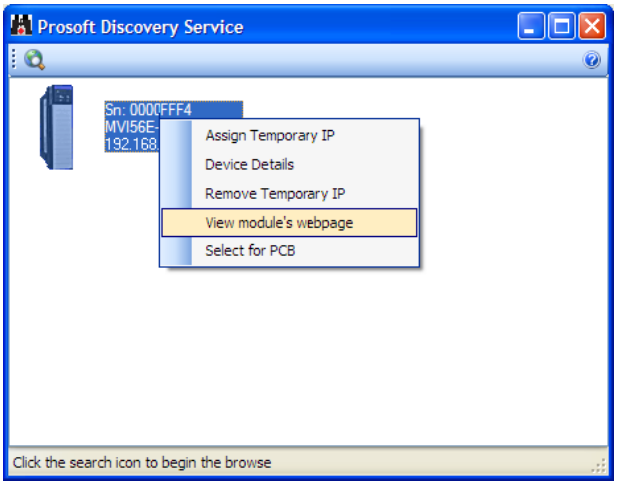
Parameter	Description
Source Module	Source module type. This field is automatically selected depending on the destination module of the last rack (1756-CNB or 1756-ENBT).
Source Module IP Address	IP address of the source module (only applicable for 1756-ENBT)
Source Module Node Address	Node address of the source module (only applicable for 1756-CNB)
Destination Module	Select the destination module associated to the source module in the rack. The connection between the source and destination modules is performed through the backplane.
Destination Module Slot Number	The slot number where the destination MVI56E module is located.

To use the CIPconnect Path Editor, follow these steps.

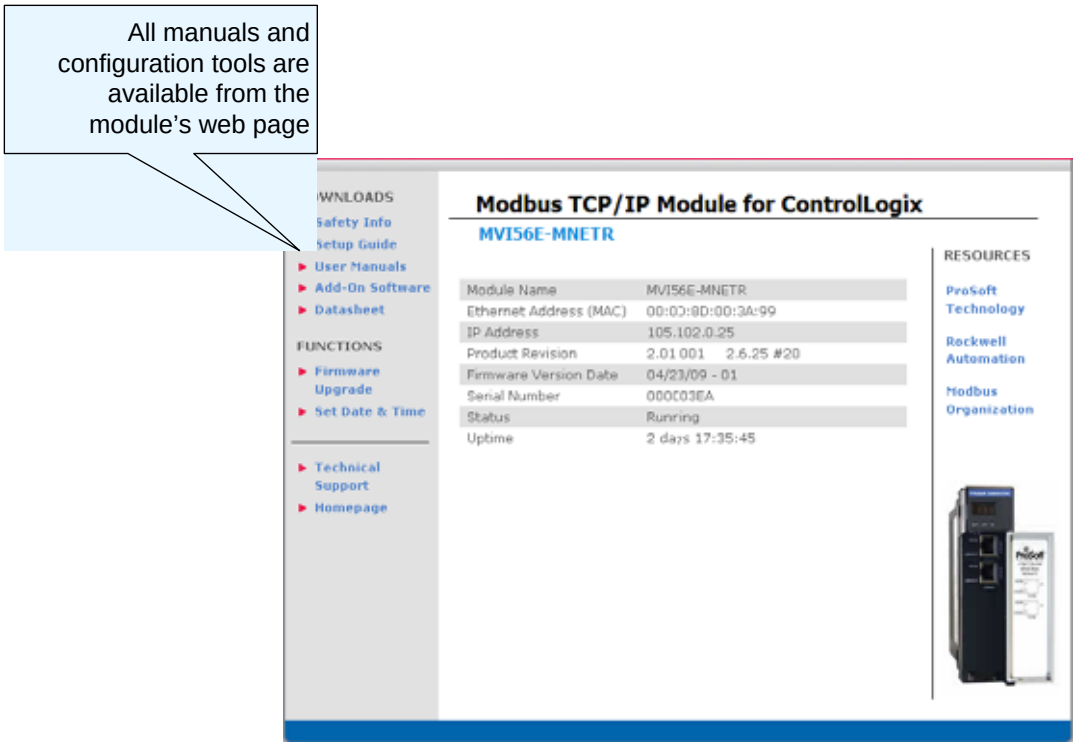
1. Configure the path between the 1756-ENBT connected to your PC and the MVI56E-MNETR module.
   - If the module is located in a remote rack, add more racks to configure the full path.
   - The path can only contain ControlNet or Ethernet/IP networks.
   - The maximum number of supported racks is six.
2. Click **CONSTRUCT CIP PATH** to build the path in text format
3. Click **OK** to confirm the configured path.

1.9 Connect to the Module's Web Page

1. In ProSoft Discovery Service, select the module to configure, and then click the right mouse button to open a shortcut menu.
2. On the shortcut menu, choose **VIEW MODULE'S WEBPAGE**.



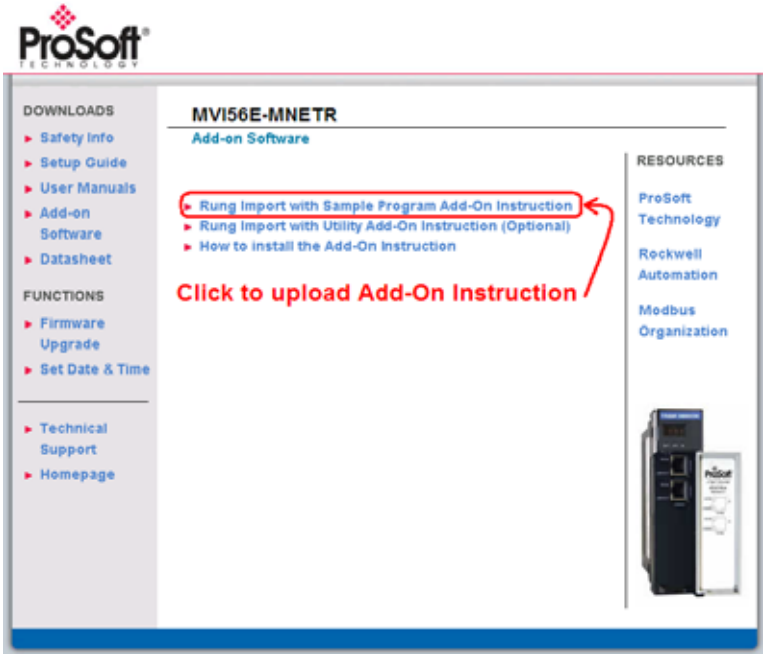
The web page contains the product documentation and sample programs.



Important: The temporary IP address is only valid until the next time the module is initialized. Please refer to the MVI56E-MNETR User Manual for information on how to set the module's permanent IP address.

1.10 Upload the Add-On Instruction from the Module

Configuration and control information for the MVI56E-MNETR module is provided as an Add-On Instruction for RSLogix 5000, version 16 or higher.

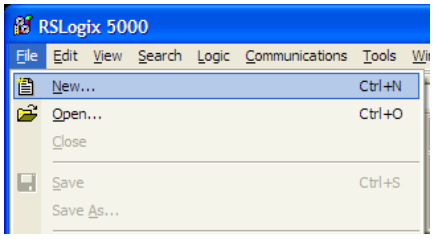


Two Add-On Instructions are provided:

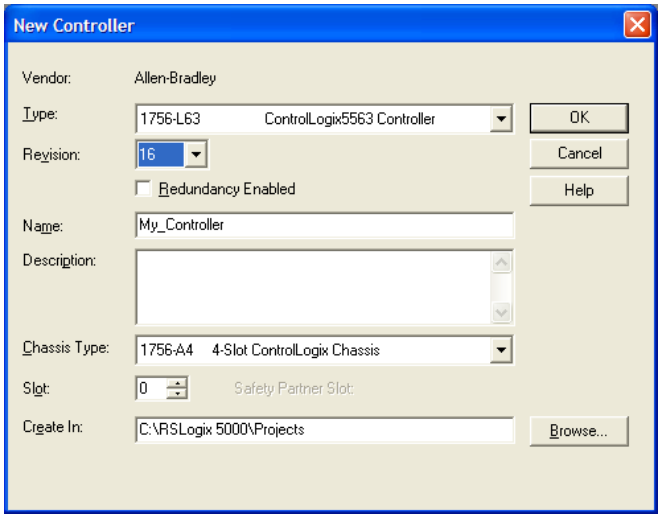
- The **RUNG IMPORT WITH SAMPLE PROGRAM ADD-ON INSTRUCTION** (MVI56(E)MNETR_AddOn_Rung_<Version #>.L5X), which includes the User Defined Data Types, data objects and ladder logic required to configure the MVI56E-MNETR module.
- The **RUNG IMPORT WITH UTILITY ADD-ON INSTRUCTION (OPTIONAL)** (MVI56(E)MNETR_Optional_AddOn_Rung_<version #>.L5X), which includes the data types and controller tags that allow you to update the IP address, date and time on the module. (page 25)

1.11 Create a new RSLogix 5000 project

1 Open the **FILE** menu, and then choose **NEW…**



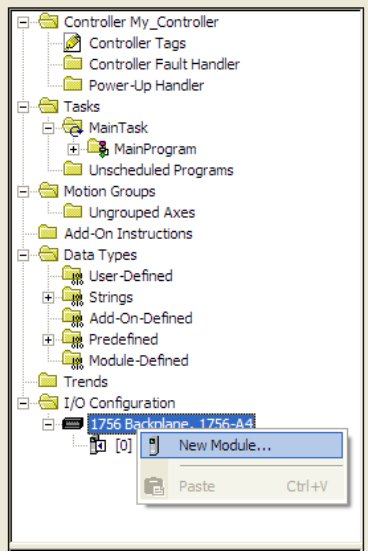
2 Select **REVISION 16**.



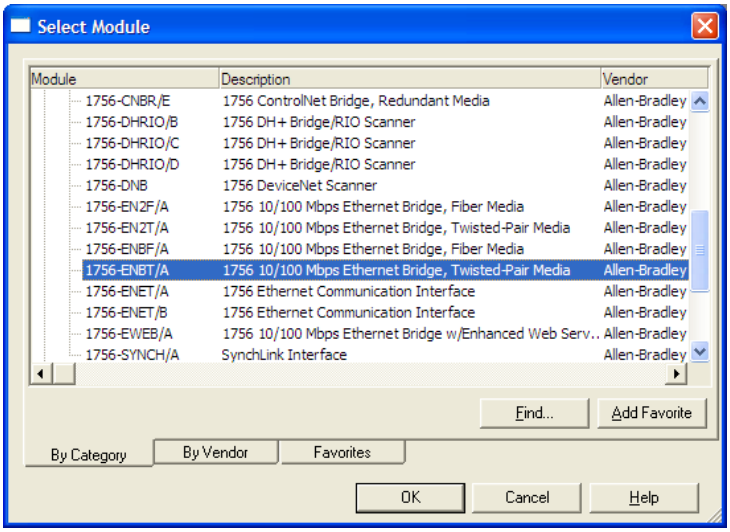
Note. If you are installing the MVI56E-MNETR module in a remote rack, follow these steps. If you are installing the module in a local rack, refer to Create the Module - Local Rack (page 31).

1.11.1 Create the Remote Network

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

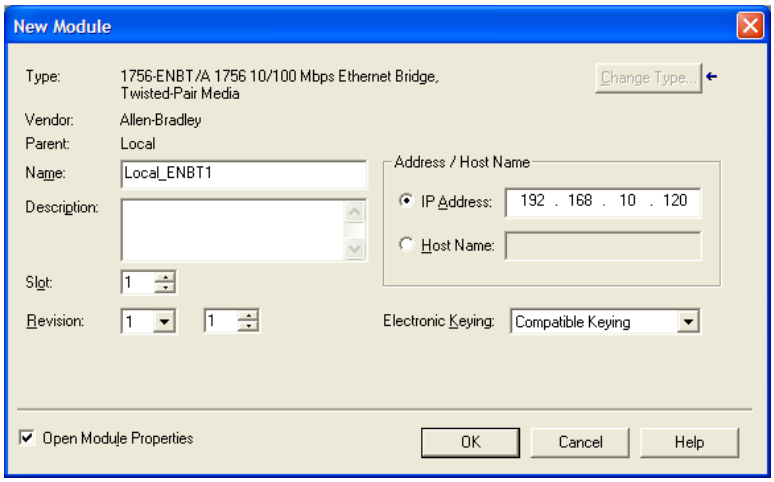


2 Expand the **COMMUNICATIONS** module selections and then select the Ethernet Bridge module that matches your hardware. This example uses a 1756-ENBT/A module.



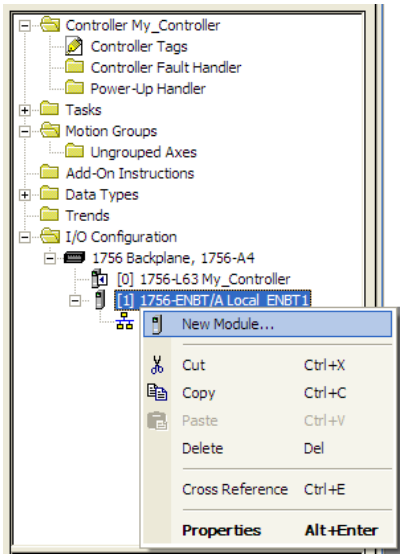
Note: If you are prompted to "Select Major Revision", choose the lower of the available revision numbers.

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



4 Click **OK**.

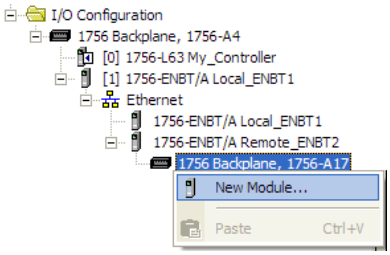
5 Next, select the **1756-ENBT** module that you just created in the Controller Organization pane and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.



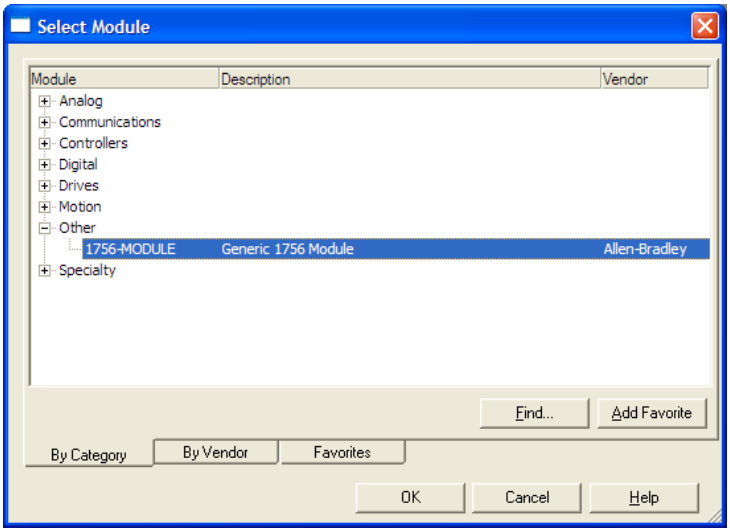
6 Repeat steps 2 and 3 to add the second EtherNet/IP module to the remote rack.

1.11.2 Create the Module - Remote Rack

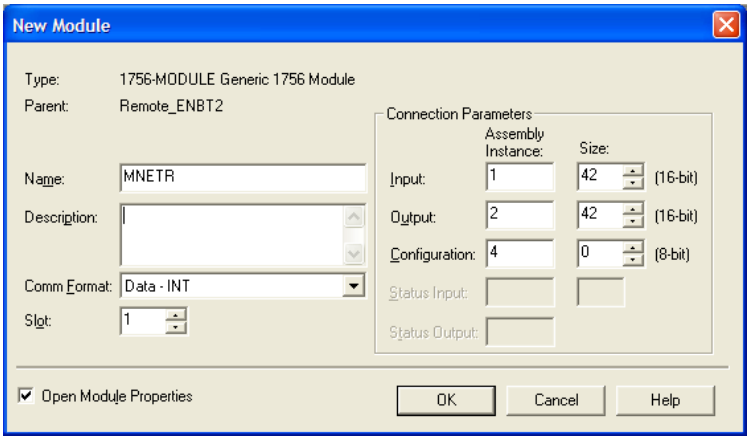
1 Next, select the remote **1756 BACKPLANE** node in the Controller Organization pane underneath the remote rack EtherNet/IP module you just created and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.



2 Expand the **OTHER** modules selection and then select **1756-MODULE (GENERIC 1756 MODULE)**

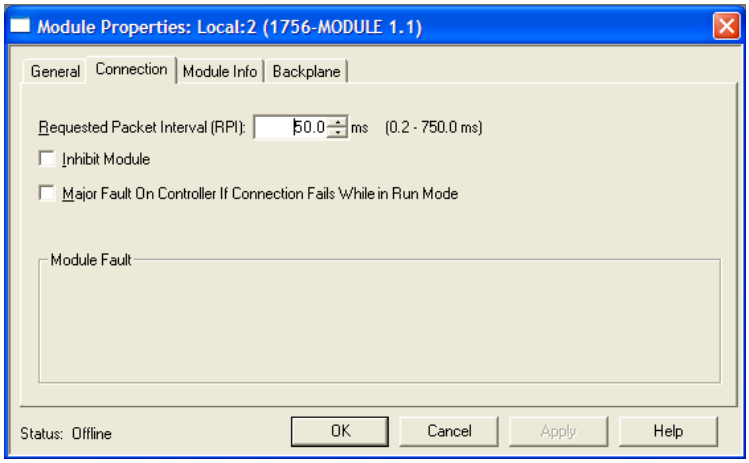


3 Set the Module Properties values as follows:

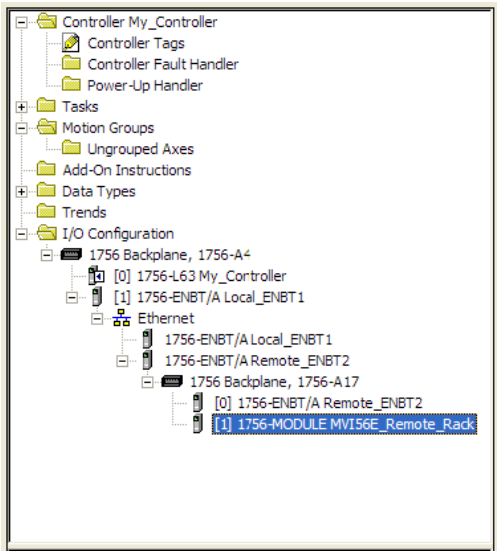


Parameter	Value
Name	Enter a module identification string. The recommended value is MNETR.
Description	Enter a description for the module. Example: Modbus TCP/IP Interface Module with Reduced Data Block.
Comm Format	Select **DATA-INT (Very Important)**
Slot	Enter the slot number in the rack where the MVI56E-MNETR module is or will be located.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

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



The MVI56E-MNETR module is now visible in the **I/O CONFIGURATION** section

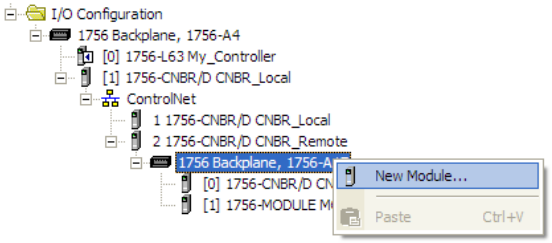


Note. If you are installing the MVI56E-MNETR module in a local rack, follow these steps. If you are installing the module in a remote rack, refer to Create the Module - Remote Rack (page 27).

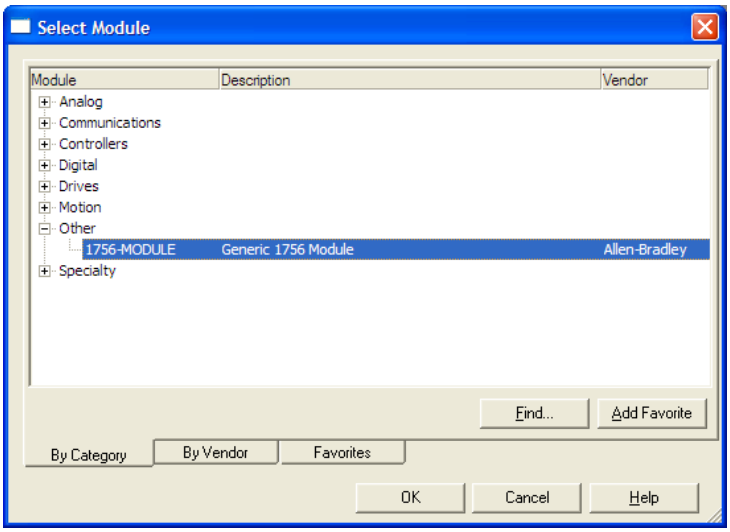
1.11.3 Create the Module - Local Rack

1 Add the MVI56E-MNETR module to the project.

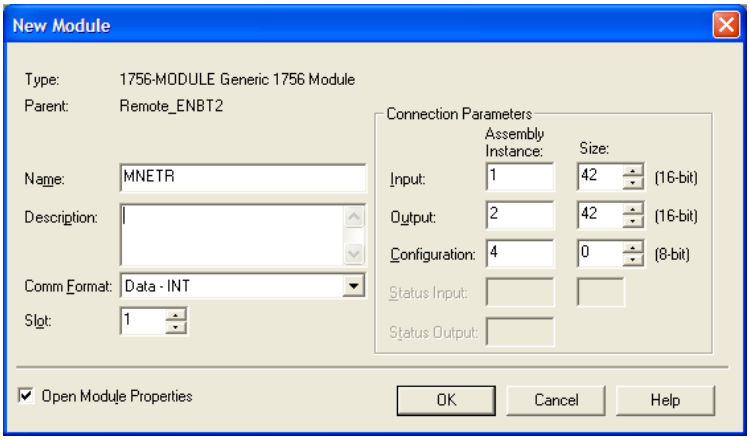
In the **CONTROLLER ORGANIZATION** window, select **I/O CONFIGURATION** and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE...**



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



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

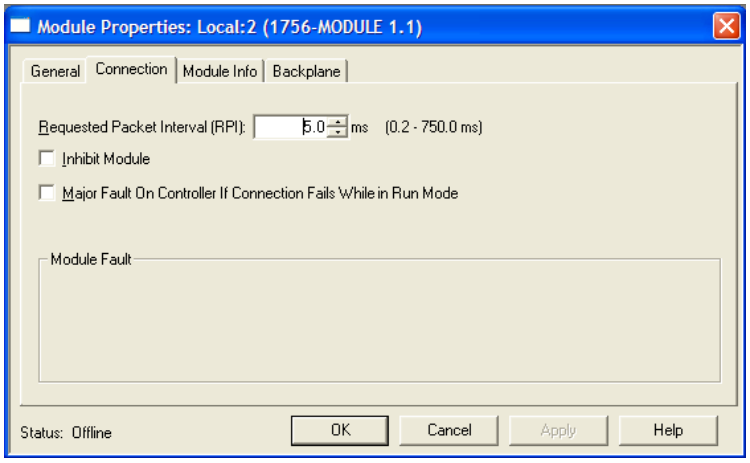


3 Set the Module Properties values as follows:

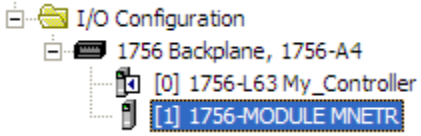
Parameter	Value
Name	Enter a module identification string. The recommended value is MNETR.
Description	Enter a description for the module. Example: Modbus TCP/IP Interface Module with Reduced Data Block.
Comm Format	Select **DATA-INT (Very Important)**
Slot	Enter the slot number in the rack where the MVI56E-MNETR module is to be installed.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2

Parameter	Value
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

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

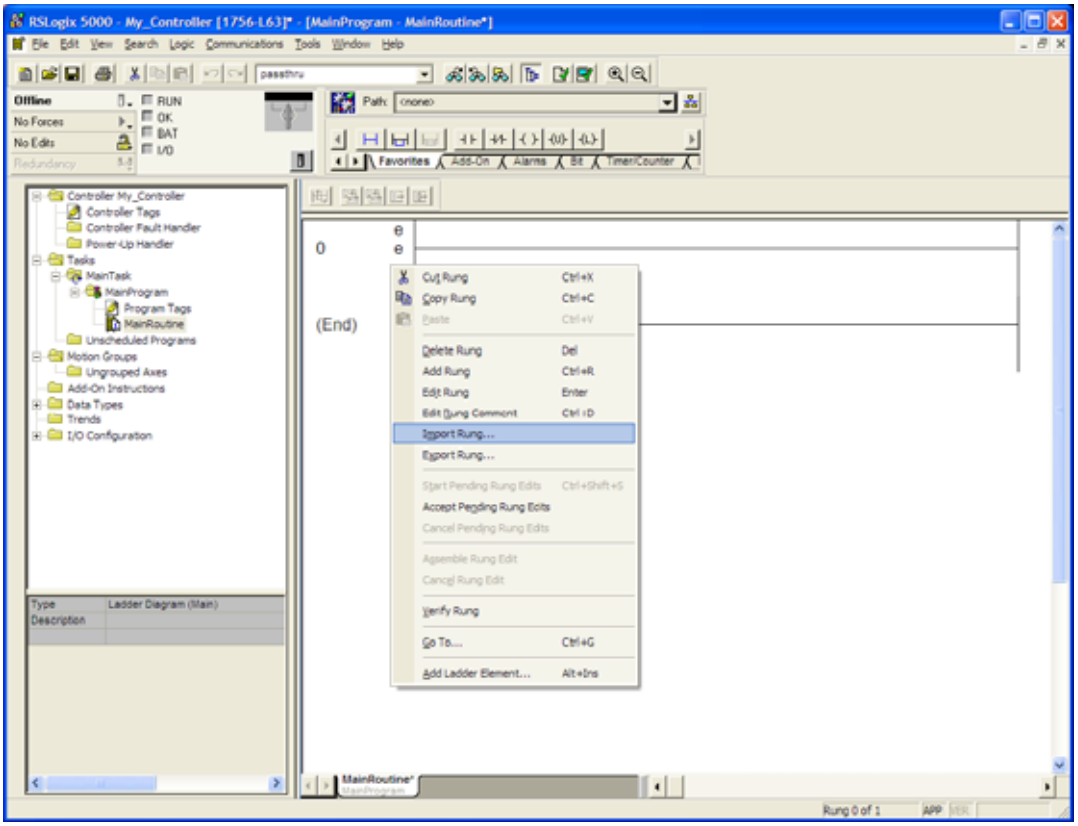


The MVI56E-MNETR module is now visible in the **I/O CONFIGURATION** section

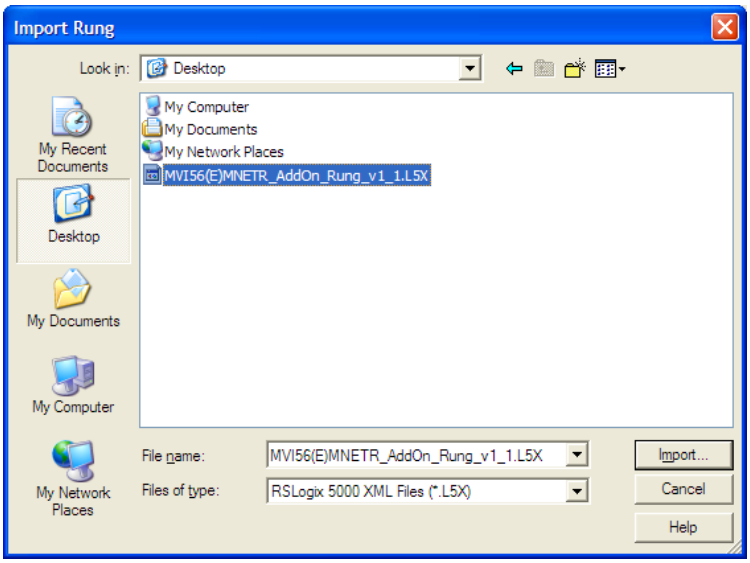


1.11.4 Import Add-On Instruction

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

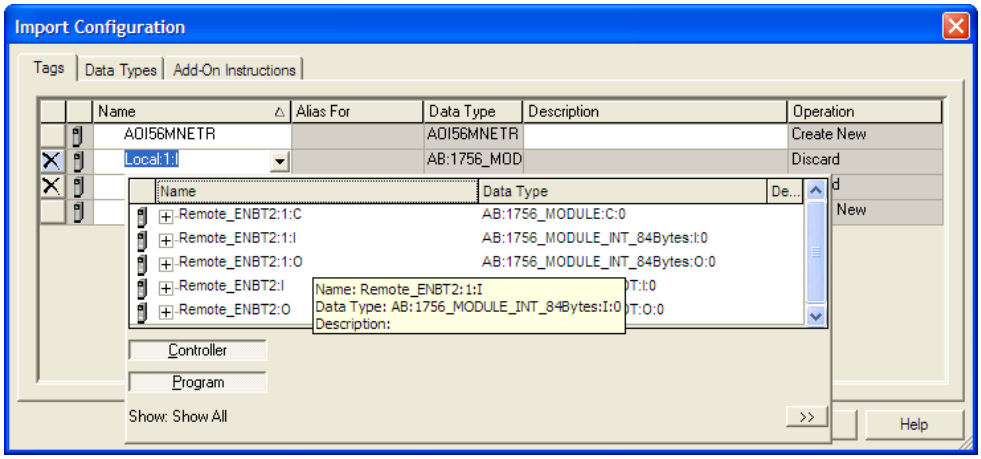


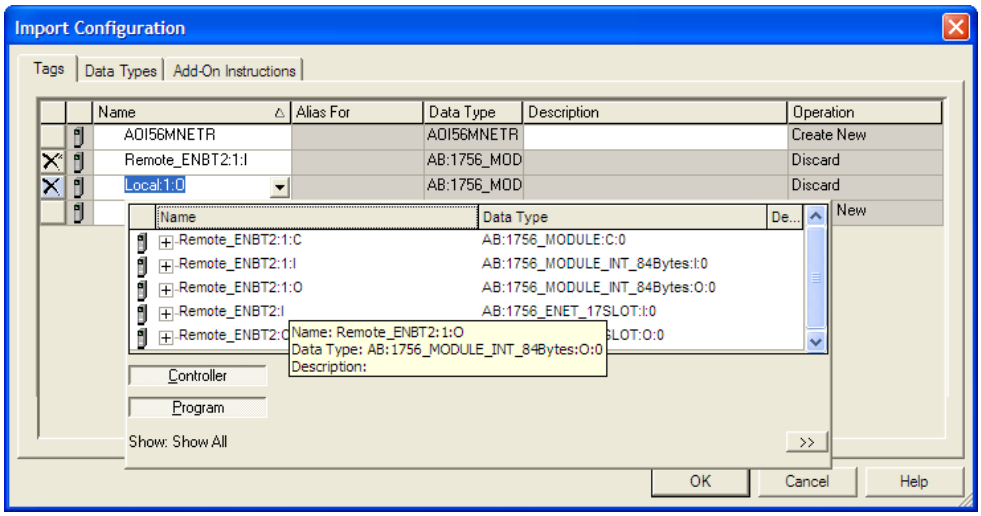
4 Navigate to the location on your PC where you saved (page 25) the Add-On Instruction (for example, "My Documents" or "Desktop"). Select the **MVI56(E)MNETR_ADDON_RUNG_<VERSION #>.L5X** file



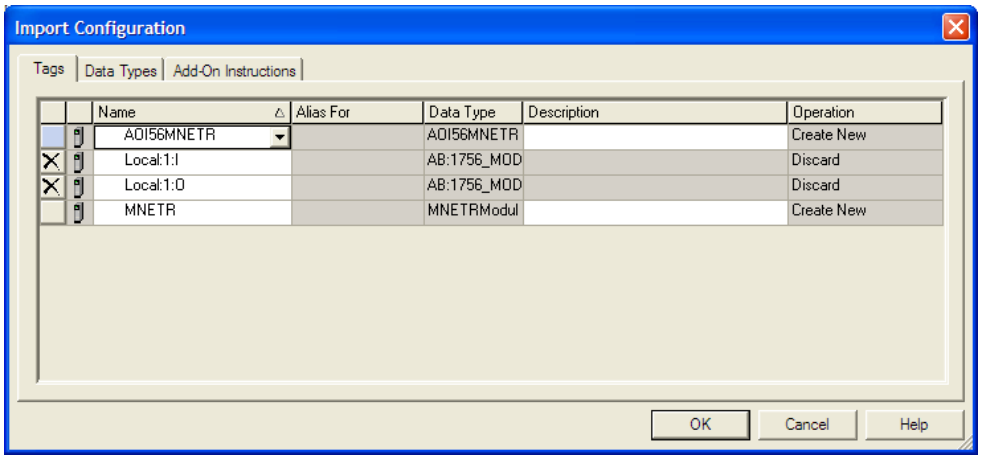
This action opens the **IMPORT CONFIGURATION** dialog box, showing the controller tags that will be created.

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

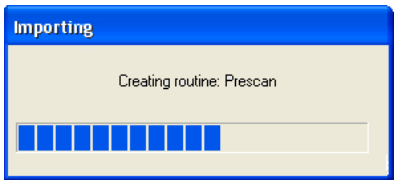




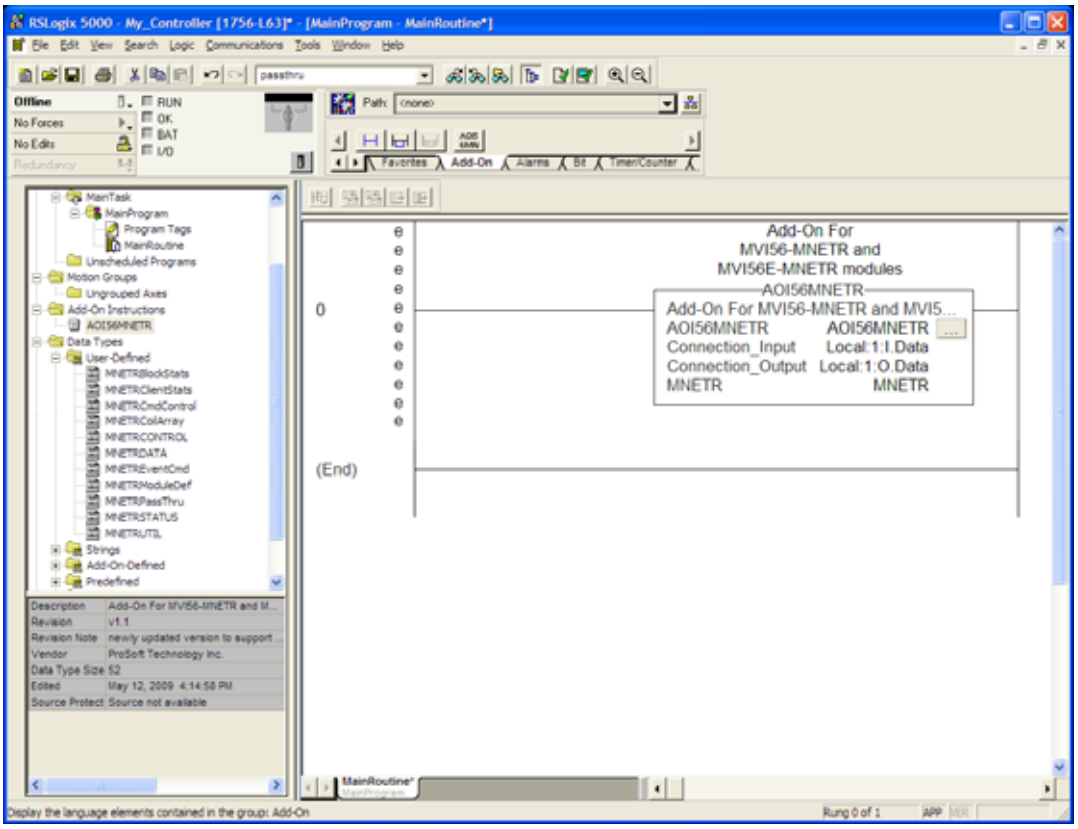
- If you are installing the module in a Local Rack, verify that the slot number is correct for the module.



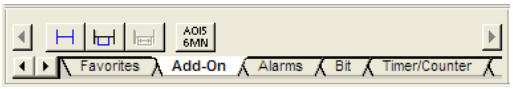
5 Click **OK** to confirm the import. RSLogix will indicate that the import is in progress:



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



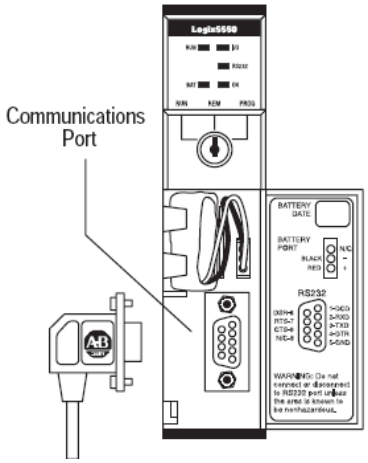
The procedure has also imported new User Defined Data Types, data objects and the Add-On instruction for your project.



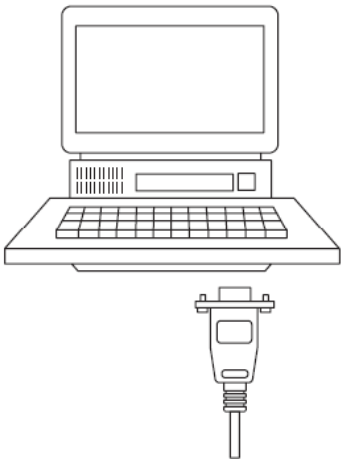
1.11.5 Connect your PC to the ControlLogix Processor

There are several ways to establish communication between your PC and the ControlLogix processor. The following steps show how to establish communication through the serial interface. Refer to your Rockwell Automation documentation for information on other connection methods.

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



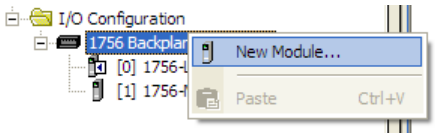
2 Connect the straight connector end of the cable to the serial port on your computer.



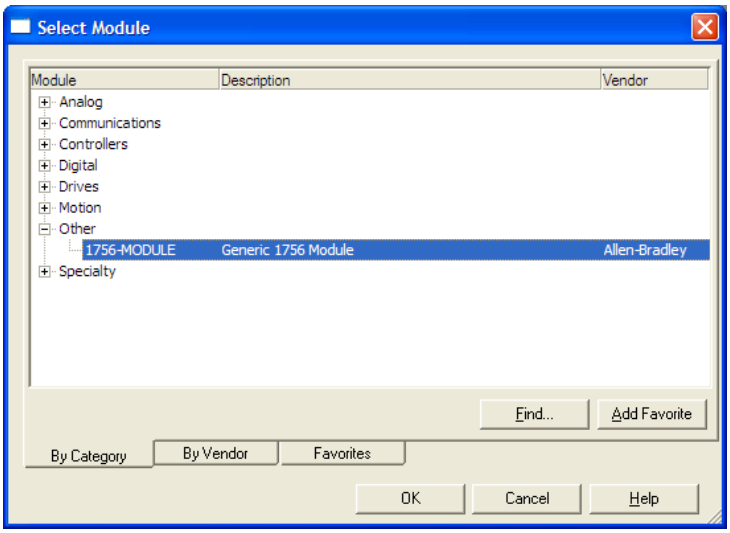
1.11.6 Adding Multiple Modules (Optional)

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

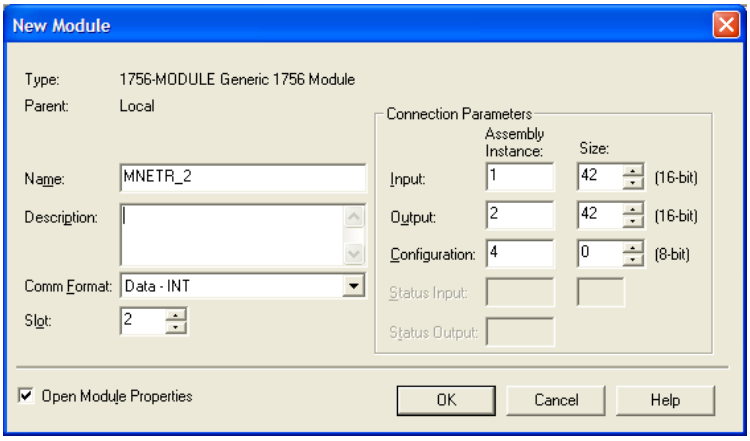
1 In the *I/O Configuration* folder, click the right mouse button to open a shortcut menu, and then choose **NEW MODULE**.



2 Select **1756-MODULE**



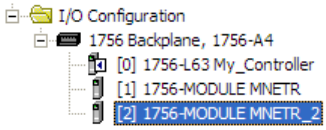
This action opens the *New Module* dialog box.



3 Fill in the module properties as follows:

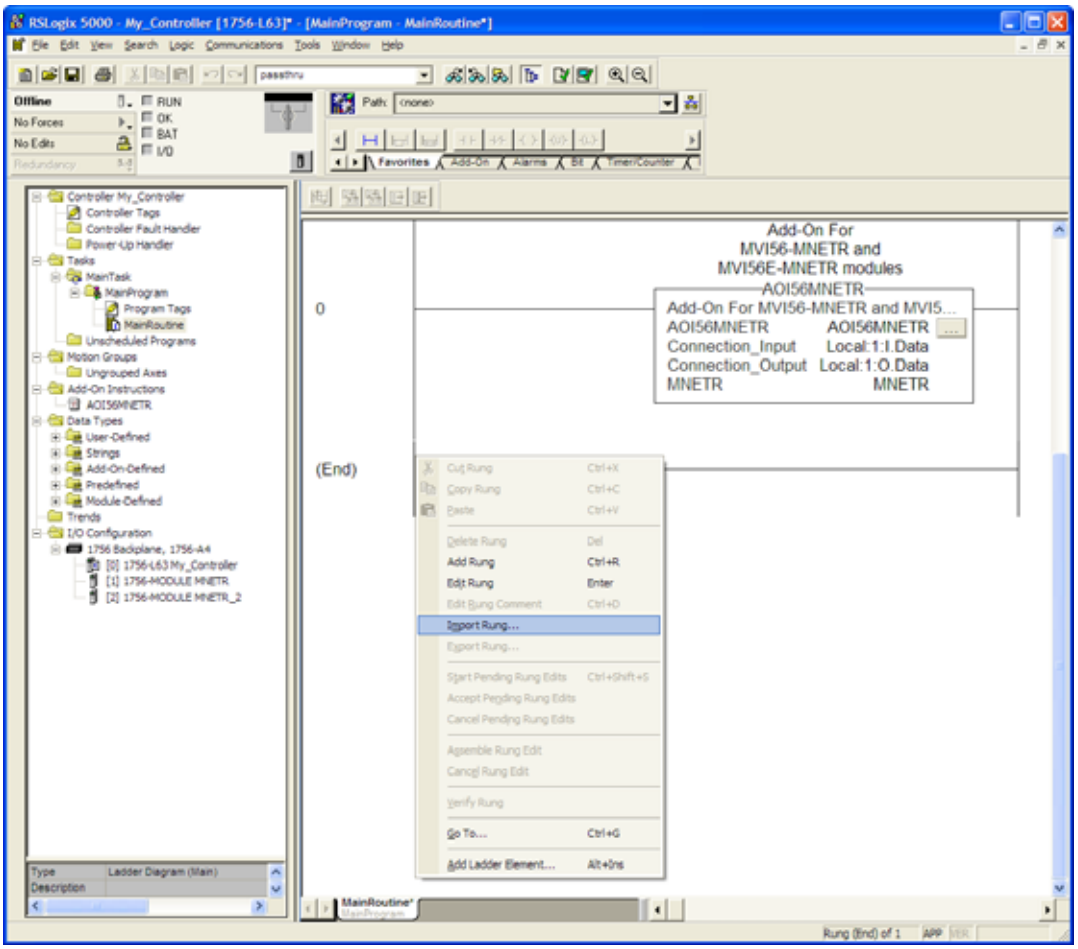
Parameter	Value
Name	Enter a module identification string. Example: MNETR_2
Description	Enter a description for the module. Example: Modbus TCP/IP Interface Module with Reduced Data Block
Comm Format	Select DATA-INT
Slot	Enter the slot number in the rack where the ModelNumber> module is located.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

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

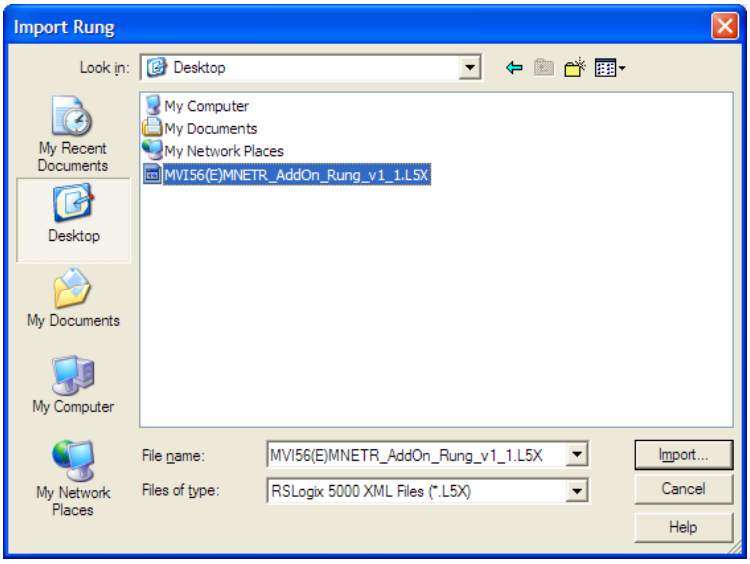


5 Expand the *Tasks* folder, and then expand the *MainTask* folder.
6 On the MainProgram folder, click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW ROUTINE**.
7 In the *New Routine* dialog box, enter the name and description of your routine, and then click **OK**.

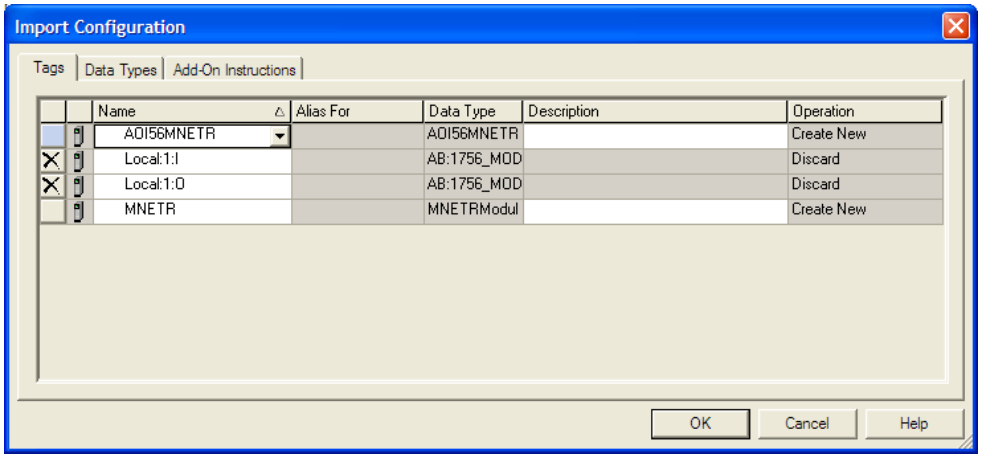
8 Select an empty rung in the new routine, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **"IMPORT RUNG…"**.



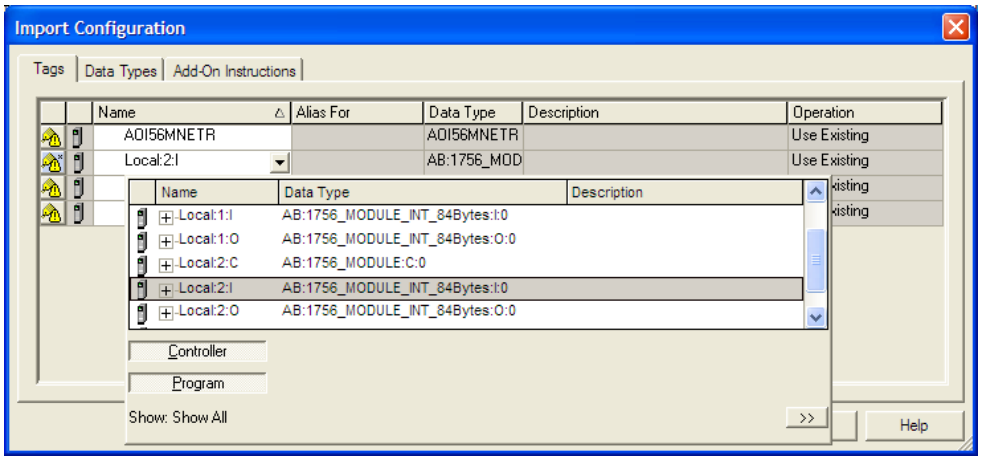
9 Select the file MVI56(E)MNETR_AddOn_Rung_<Version #>.L5X

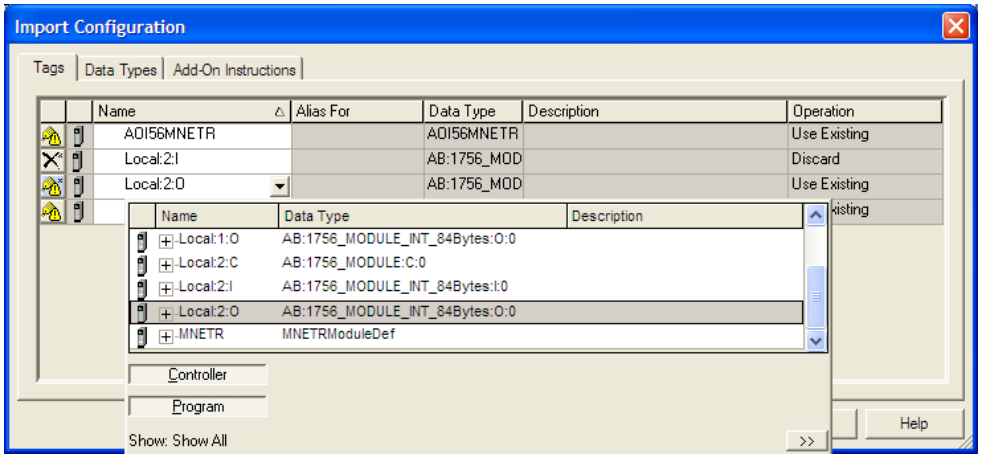


This action opens the **IMPORT CONFIGURATION** dialog box, showing the controller tags that will be created.

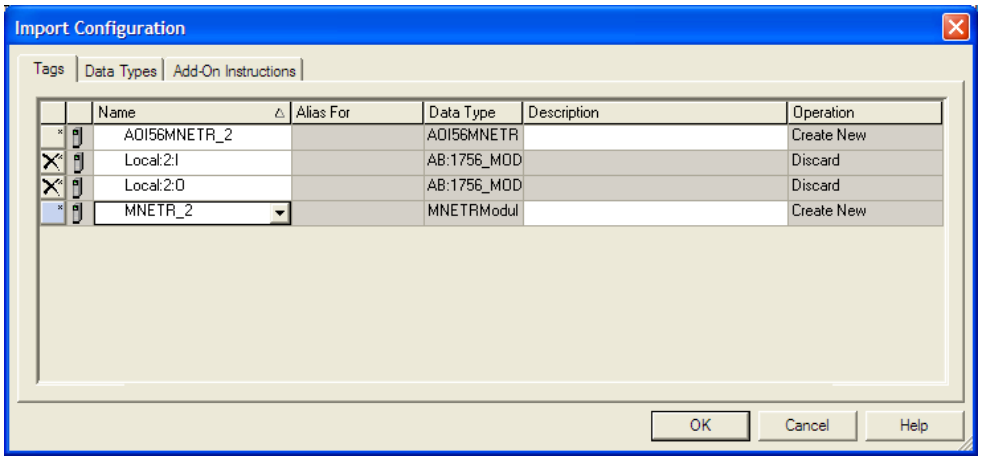


10 Associate the I/O connection variables to the correct module. The default values are Local:1:I and Local:1:O so these require change.

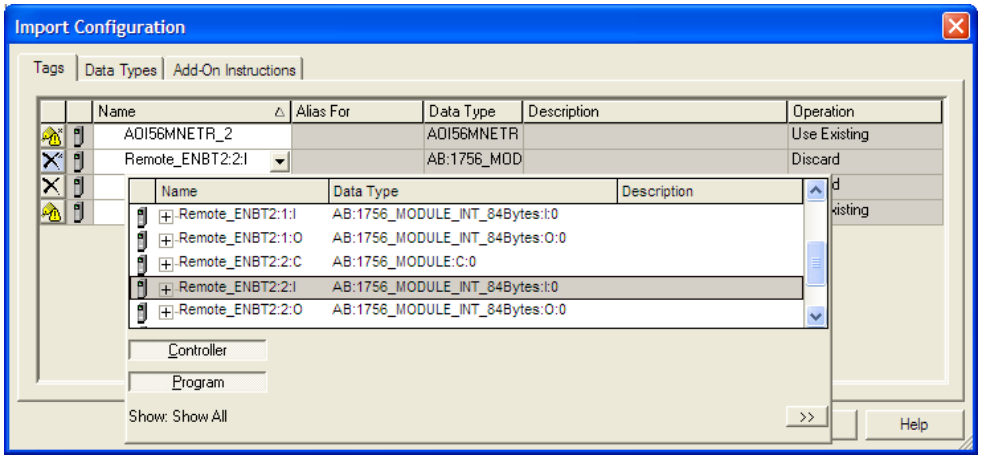


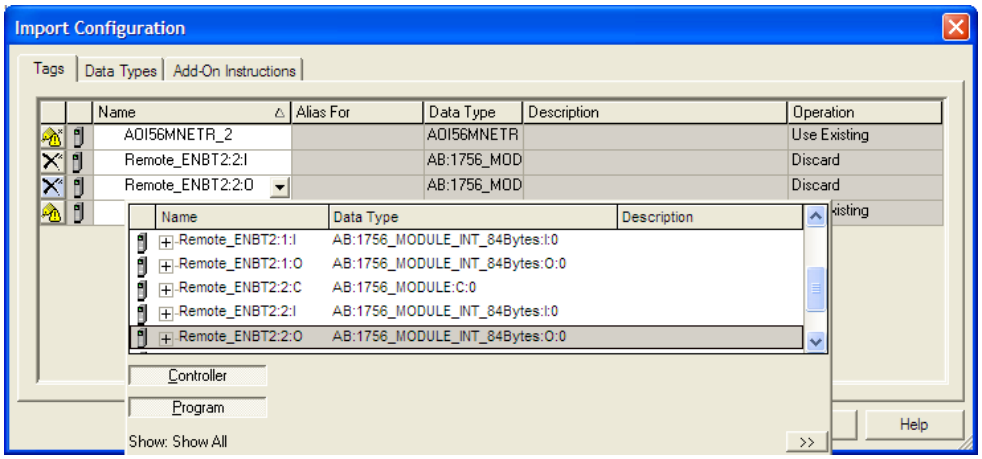


Change the default tags *MNETR* and *AOI56MNETR* to avoid conflict with existing tags. This procedure will append the string "_2" as follows:

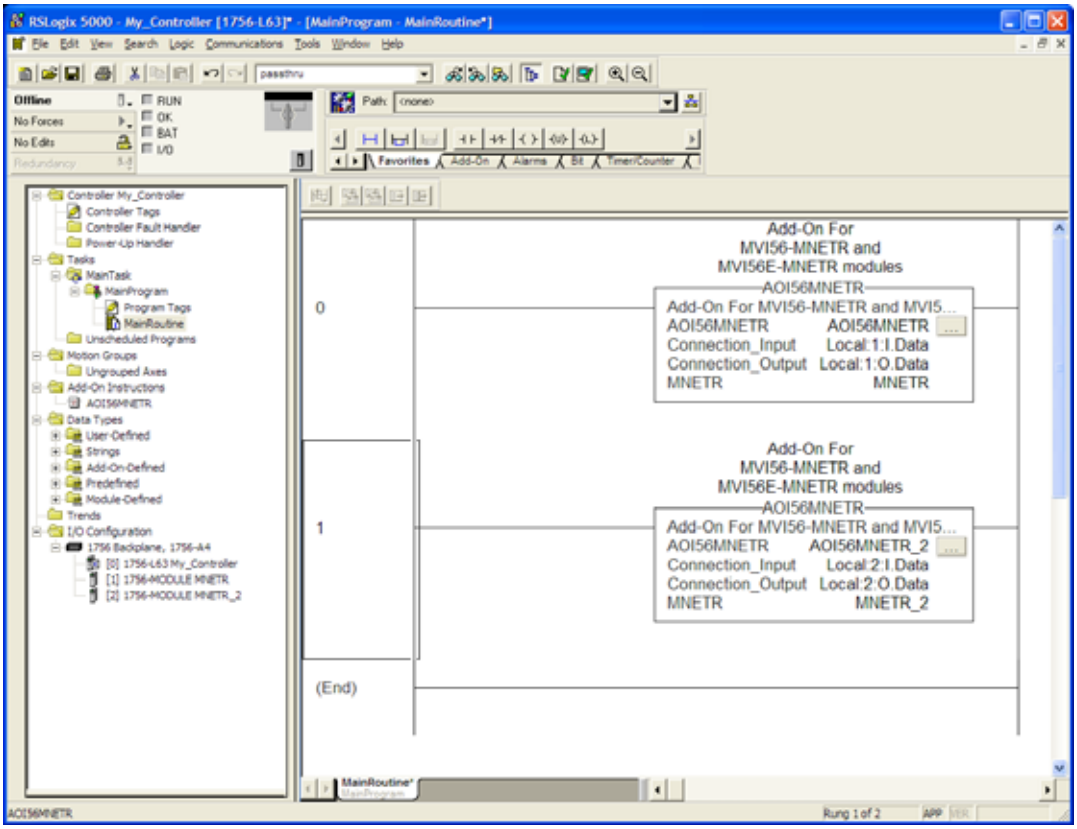


Or, in a Remote Rack application…

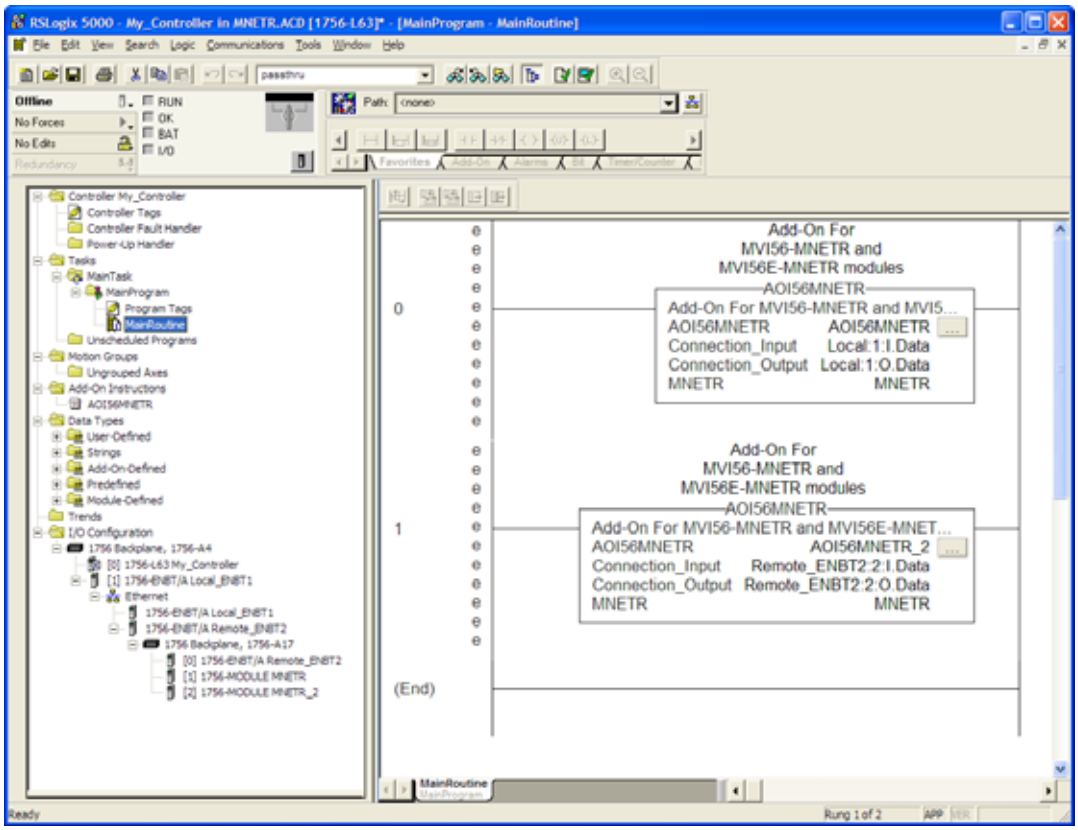


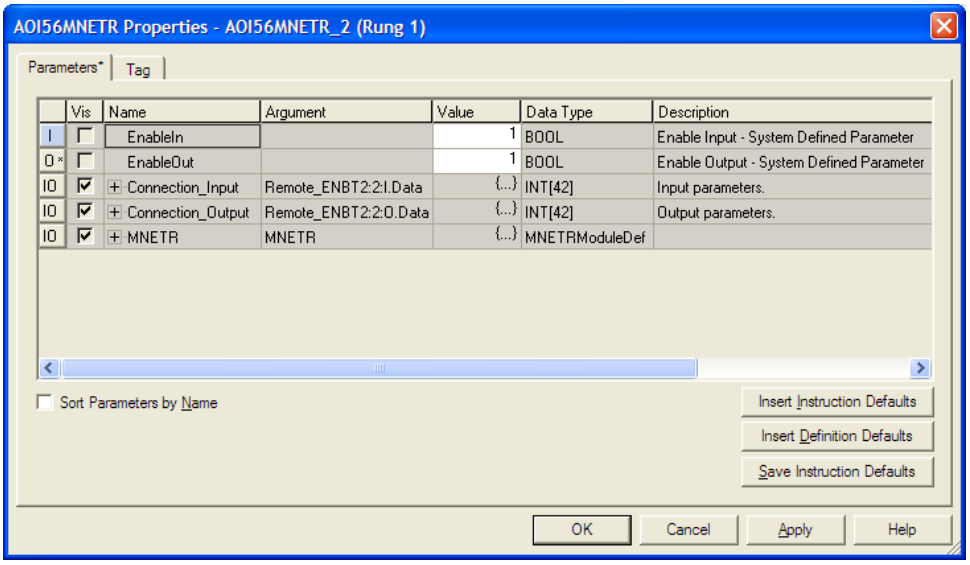


11 Click **OK** to confirm.



Or, in a Remote Rack application…



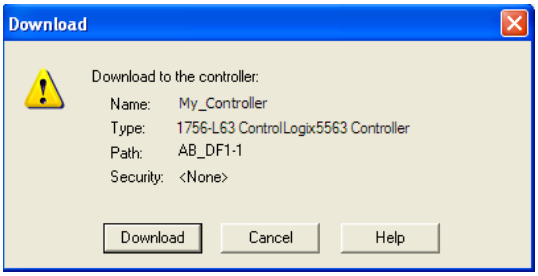


The setup procedure is now complete. Save the project and download the application to your ControlLogix processor.

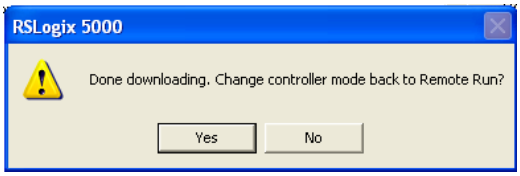
1.11.7 Download the Sample Program to the Processor

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 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. If the keyswitch is in the **REM** position, click **OK** to switch the processor from **PROGRAM** mode to **RUN** mode.



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

2 Configuring the MVI56E-MNETR Module

In This Chapter

- Using ProSoft Configuration Builder Software 49
- Download the Project to the Module 67
- Using CIPconnect® to Connect to the Module 68

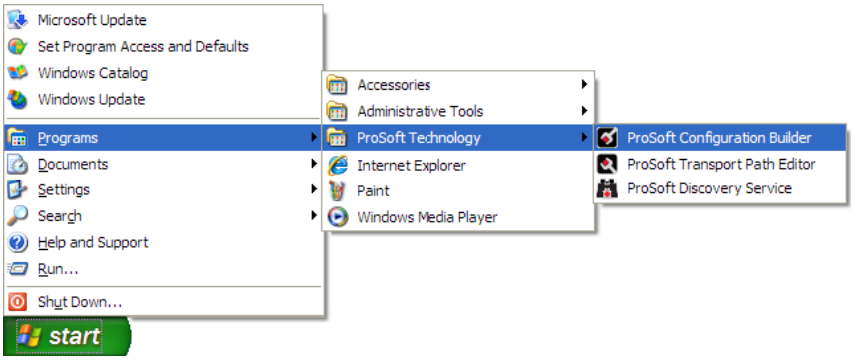
2.1 Using ProSoft Configuration Builder Software

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.

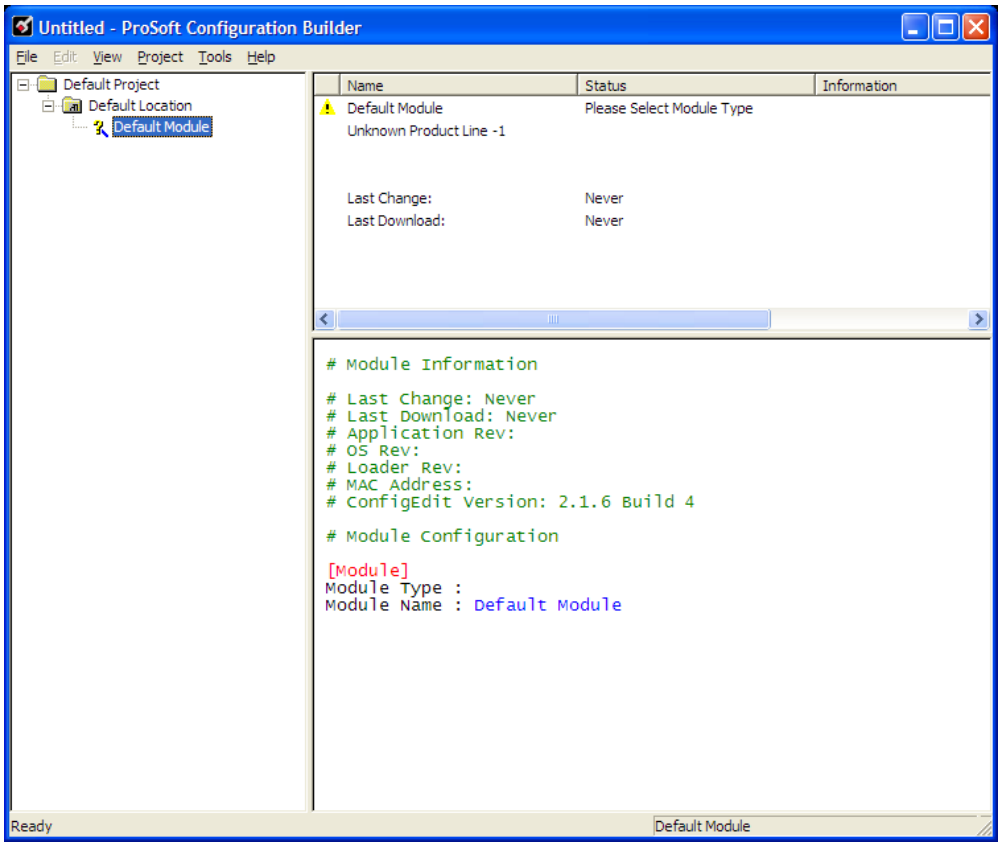
Note: The MVI56E-MNETR module receives its protocol and backplane configuration information from the *Personality Module* (Compact Flash). Use ProSoft Configuration Builder to configure module settings, and to download changes to the Personality Module.

2.1.1 Set Up the Project

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

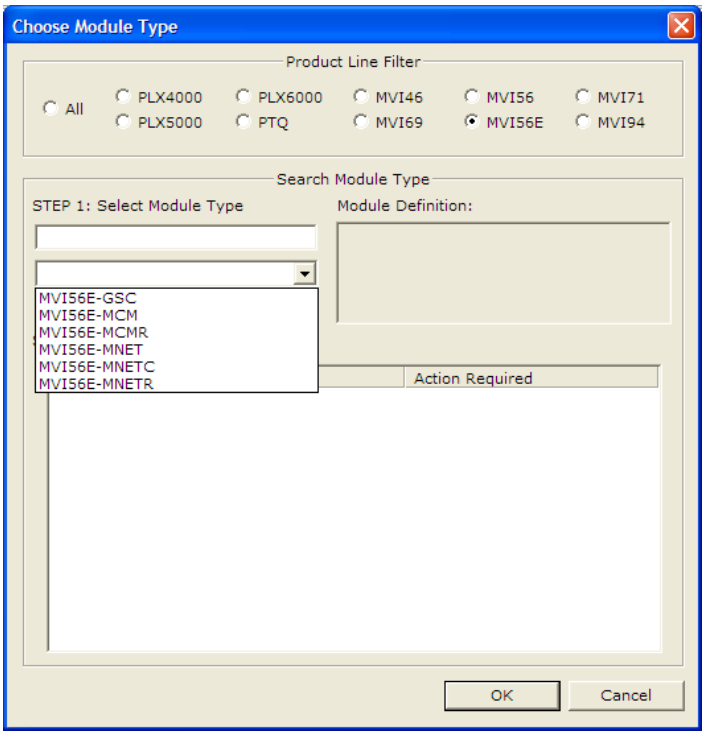


If you have used other Windows configuration tools before, you will find the screen layout familiar. PCB's window consists of a tree view on the left, and an information pane and a configuration pane on the right side of the window. When you first start *PCB*, the tree view consists of folders for **DEFAULT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



Your first task is to add the MVI56E-MNETR module to the project.

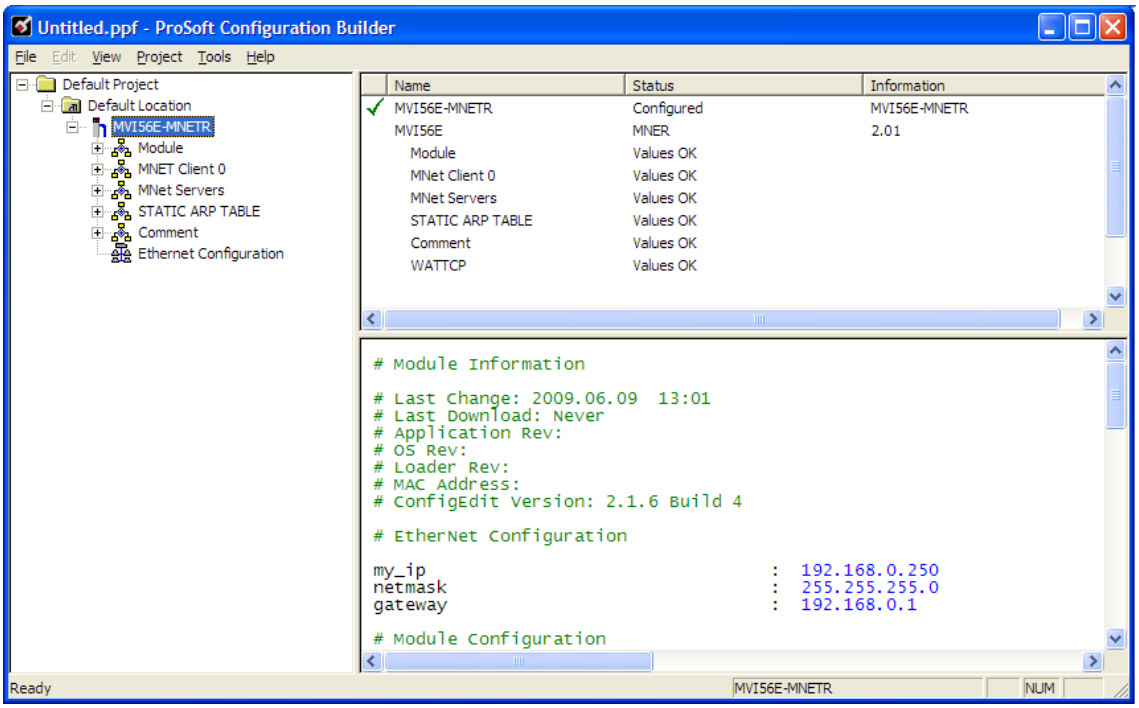
1. Use the mouse to select **DEFAULT MODULE** in the tree view, and then click the right mouse button to open a shortcut menu.
2. On the shortcut menu, choose **CHOOSE MODULE TYPE**. This action opens the **CHOOSE MODULE TYPE** dialog box.



3. In the **PRODUCT LINE FILTER** area of the dialog box, select MVI56E. In the **SELECT MODULE TYPE** dropdown list, select MVI56E-MNETR, and then click **OK** to save your settings and return to the ProSoft Configuration Builder window.

2.1.2 Set Module Parameters

The next task is to configure module parameters. Notice that the contents of the information pane and the configuration pane changed when you added the MVI56E-MNETR module to the project.



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

To rename an object:

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

Configuring Module Parameters

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

Printing a Configuration File

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

2.1.3 [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, and describes how to initialize the module upon startup.

Error/Status Pointer

-1 to 4955

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

Read Register Start

0 to 4999

This parameter specifies the starting register in the module where data will be transferred from the module to the processor. Valid range for this parameter is 0 to 4999.

Read Register Count

0 to 5000

This parameter specifies the number of registers to be transferred from the module to the processor. Valid entry for this parameter is 0 to 5000.

Write Register Start

0 to 4999

This parameter specifies the starting register in the module where the data will be transferred from the processor to the module. Valid range for this parameter is 0 to 4999.

Write Register Count

0 to 5000

This parameter specifies the number of registers to transfer from the processor to the module. Valid entry for this parameter is 0 to 5000 words.

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 force 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.

2.1.4 [MNET Client x]

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

Error/Status Pointer

-1 to 4990

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

Command Error Pointer

-1 to 4999

This parameter sets the address in the internal database where the command error data will be placed. If the value is set to -1, the data will not be transferred to the database.

Minimum Command Delay

0 to 65535

This parameter specifies the number of milliseconds to wait between the initial issuance of a command. This parameter can be used to delay all commands sent to slaves 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 Slave driver will respond to Function Code 3, 6, and 16 commands (read and write Holding Registers) from a remote Master when it is moving 32-bit floating-point data.

If the remote Master 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**, especially if the Master must read or write from Modbus addresses above

gateway address 3999 (virtual Modbus address 44000 or 440000). When set to **YES**, the Slave 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 Master for write commands. Example: Count = 10, Slave driver will send 20 16-bit registers for 10 total 32-bit floating-point values.

If, however, the remote Master 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

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

Float Offset

0 to 9999

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

ARP Timeout

1 to 60

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

Command Error Delay

0 to 300

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

2.1.5 [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 (page 121). Each command list record has the same format. The first part of the record contains the information relating to the MVI56E-MNETR communication module, and the second part contains information required to interface to the Modbus TCP/IP Server device.

Command List Overview

In order to interface the MVI56E-MNETR module with Modbus TCP/IP Server devices, you must construct a command list. The commands in the list specify the Server device to be addressed, the function to be performed (read or write), the data area in the device to interface with and the registers in the internal database to be associated with the device data. The Client command list supports up to 100 commands.

The command list is processed from top (command #0) to bottom. A poll interval parameter is associated with each command to specify a minimum delay time in tenths of a second between the issuance 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. In order to implement this feature; set the enable code for the command to a value of 2.

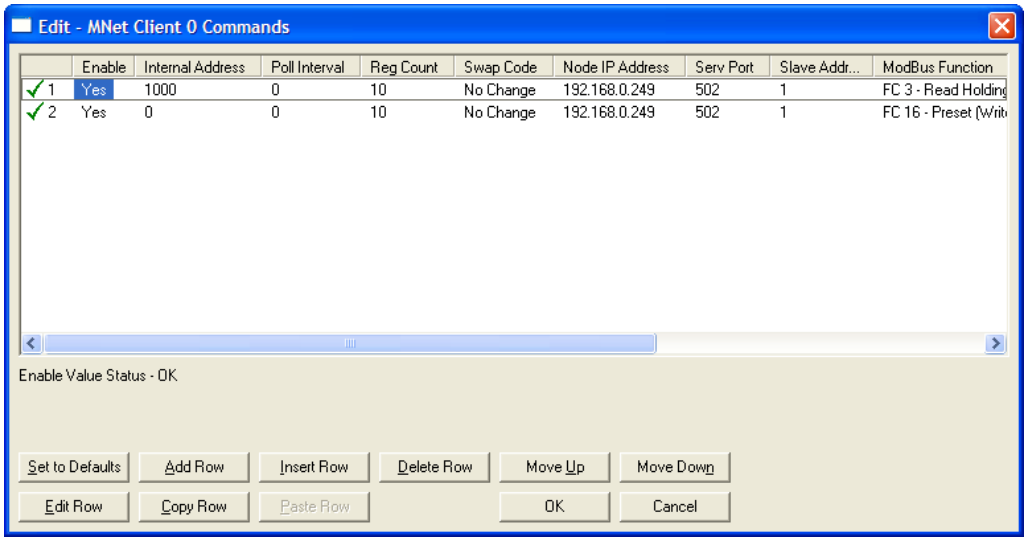
Command Entry Formats

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

Column #	1	2	3	4	5	6	7	8	9	10
Function Code	Enable Code	Internal Address	Poll Interval Time	Count	Swap Code	IP Address	Serv Port	Slave Node	Function Code	Device Modbus Address
fc1	Code	Register (bit)	1/10th Seconds	Bit Count	0	IP Address	Port #	Address	Read Coil (0x)	Register
fc2	Code	Register (bit)	1/10th Seconds	Bit Count	0	IP Address	Port #	Address	Read Input (1x)	Register
fc3	Code	Register	1/10th Seconds	Word Count	Code	IP Address	Port #	Address	Read Holding Registers (4x)	Register
fc4	Code	Register	1/10th Seconds	Word Count	0	IP Address	Port #	Address	Read Input Registers (3x)	Register
fc5	Code	1 bit	1/10th Seconds	Bit Count	0	IP Address	Port #	Address	Force (Write) Single Coil (0x)	Register
fc6	Code	1 bit	1/10th Seconds	Word Count	0	IP Address	Port #	Address	Preset (Write) Single Register (4x)	Register
fc15	Code	Register (bit)	1/10th Seconds	Bit Count	0	IP Address	Port #	Address	Force (Write) Multiple Coil (0x)	Register
fc16	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 MVI56E module and the second part contains information required to interface to the Server device.

Command list example:



Enable

Yes, No, or Conditional

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

Value	Description
0	The command is disabled and will not be executed in the normal polling sequence.
1	The command is executed each scan of the command list if the Poll Interval Time is set to zero. If the Poll Interval time is set, the command will be executed, when the interval timer expires.
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

No Change

Word Swap

Word and Byte Swap

Byte Swap

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 that do other manufacturers. 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. This parameter can be set to rearrange the byte order of data received or sent into 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 61) 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 slave device manufacturer to see if a particular slave can use addresses 248-255.
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 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 slave to be used by the command. Refer to the documentation of each Modbus slave device for the register and bit address assignments valid for that device.

The Modbus Function Code determines whether the address will be a register- or bit- level OFFSET address into a given data type range. The offset will be the target data address in the slave 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)

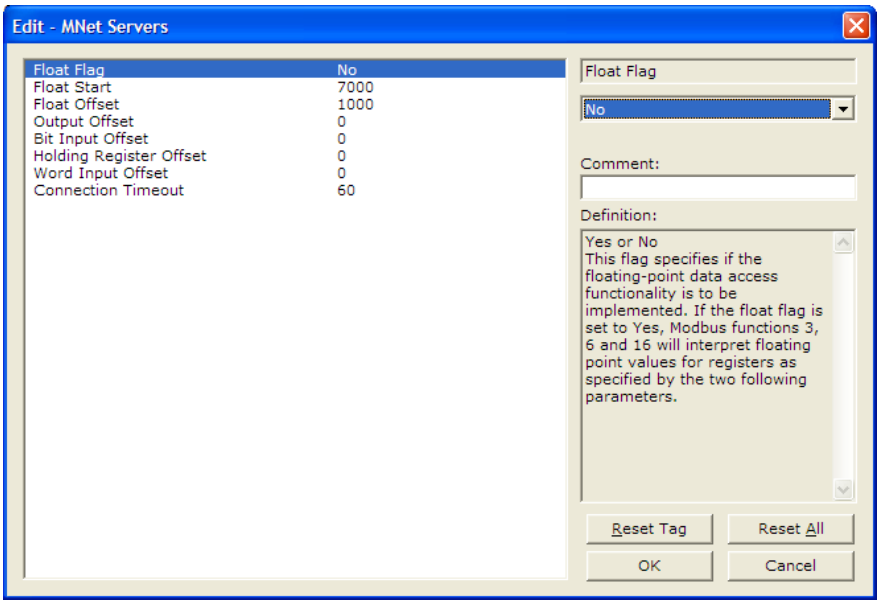
Note: If the documentation for a particular Modbus slave device lists data addresses in hexadecimal(base16) notation, you will need to convert the hexadecimal value to a decimal value to enter in this parameter. In such cases, it is not usually necessary to subtract 1 from the converted decimal number,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

Comment

0 to 35 alphanumeric characters

2.1.6 [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 Slave driver will respond to Function Code 3, 6, and 16 commands (read and write Holding Registers) from a remote Master when it is moving 32-bit floating-point data.

If the remote Master 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**, especially if the Master must read or write from Modbus addresses above gateway address 3999 (virtual Modbus address 44000 or 440000). When set to **YES**, the Slave 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 Master for write commands. Example: Count = 10, Slave driver will send 20 16-bit registers for 10 total 32-bit floating-point values.

If, however, the remote Master 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

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

Float Offset

0 to 9999

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

Output Offset

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 requires 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 3999

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

Holding Register Offset

0 to 4999

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

Word Input Offset

0 to 4999

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

2.1.7 [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 MVI56E-MNETR module will be communicating with regularly.

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

IP Address

Dotted notation

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

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

Hardware MAC Address

Hex Value

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

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

2.1.8 Ethernet Configuration

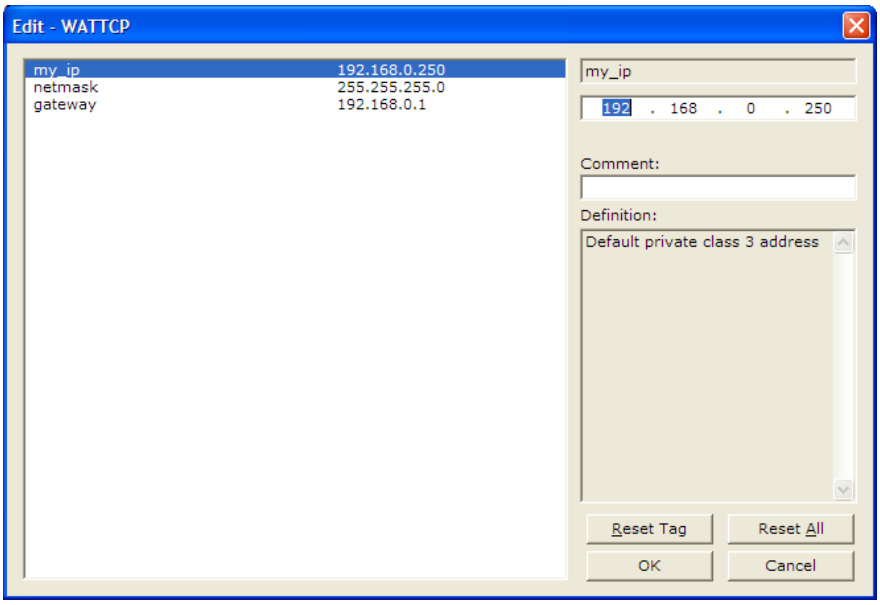
Use this procedure to configure the Ethernet settings for your module. You must assign an IP address, subnet mask and gateway address. After you complete this step, you can connect to the module with an Ethernet cable.

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

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

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

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



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

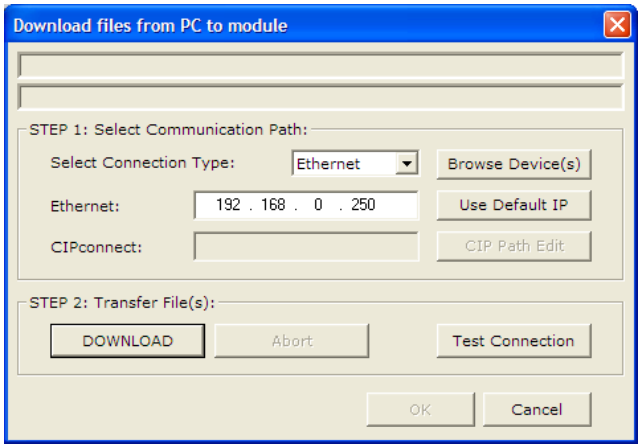
4 When you are finished editing, click **OK** to save your changes and return to the ProSoft Configuration Builder window.

2.2 Download the Project to the Module

In order 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 MVI56E-MNETR module.
2. Open the **PROJECT** menu, and then choose **MODULE / DOWNLOAD.**

   This action opens the **DOWNLOAD** dialog box. Notice that the Ethernet address field contains the temporary IP address you assigned in the previous step. ProSoft Configuration Builder will use this temporary IP address to connect to the module.



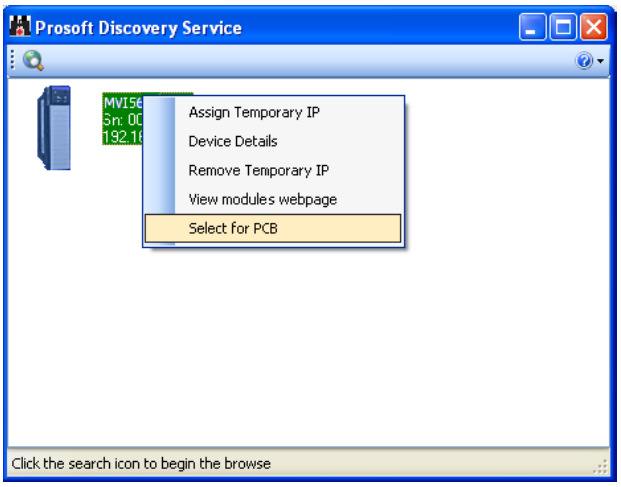
   Click **TEST CONNECTION** to verify that the temporary IP address is correct.

3. If the connection succeeds, click **DOWNLOAD** to transfer the Ethernet configuration to the module.

If the Test Connection procedure fails, you will see an error message. To correct the error, follow these steps.

1. Click **OK** to dismiss the error message.

2 On the **DOWNLOAD** dialog box, click **BROWSE DEVICES** to open **PROSOFT DISCOVERY SERVICE**.



3 Select the module, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **SELECT FOR PCB**.
4 Close ProSoft Discovery Service.
5 Click **DOWNLOAD** to transfer the configuration to the module.

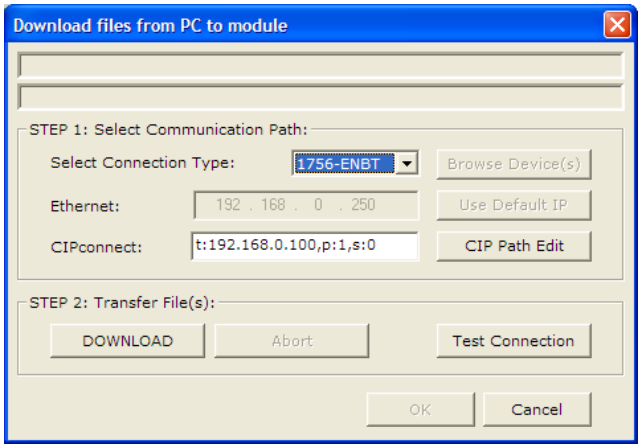
2.3 Using CIPconnect® to Connect to the Module

You can use CIPconnect® to connect a PC to the ProSoft Technology MVI56E-MNETR module over Ethernet using Rockwell Automation's 1756-ENBT EtherNet/IP® module. This allows you to configure the MVI56E-MNETR module and network, upload and download files, and view module diagnostics from a PC. RSLinx is not required when you use CIPconnect. All you need are:

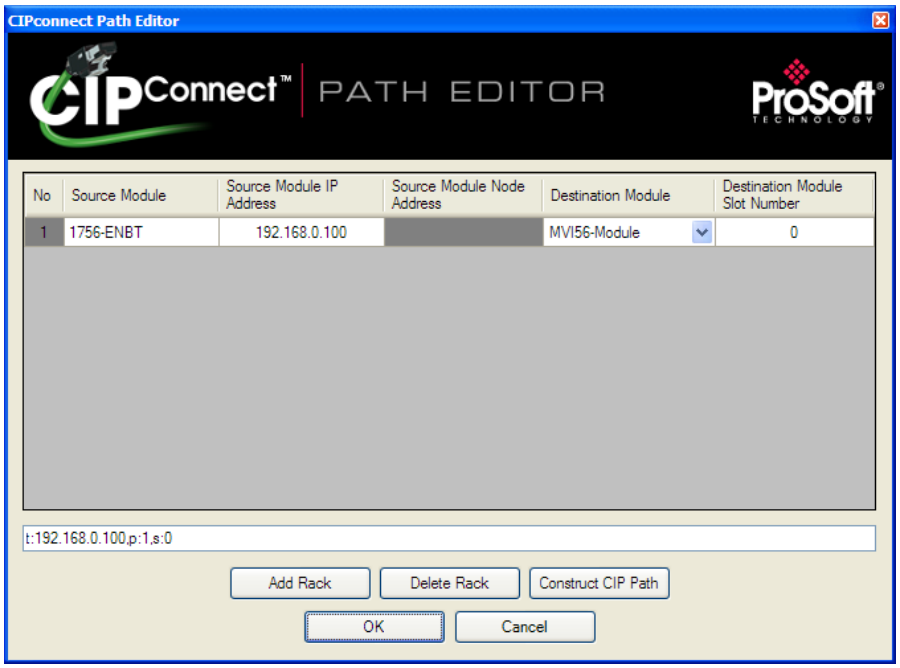
- The IP addresses and slot numbers of any 1756-ENBT modules in the path
- The ControlNet node numbers and slot numbers of any 1756-CNBx ControlNet Bridge modules in the path
- The slot number of the MVI56E-MNETR in the destination ControlLogix chassis (the last ENBT/CNBx and chassis in the path).

To use CIPconnect, follow these steps.

1 In the **SELECT PORT** dropdown list, choose 1756-ENBT. The default path appears in the text box, as shown in the following illustration.



2 Click **CIP PATH EDIT** to open the **CIPCONNECT PATH EDITOR** dialog box.



The **CIPCONNECT PATH EDITOR** allows you to define the path between the PC and the MVI56E-MNETR module. The first connection from the PC is always a 1756-ENBT (Ethernet/IP) module.

Each row corresponds to a physical rack in the CIP path.

- If the MVI56E-MNETR module is located in the same rack as the first 1756-ENBT module, select **RACK NO. 1** and configure the associated parameters.
- If the MVI56E-MNETR is available in a remote rack (accessible through ControlNet or Ethernet/IP), include all racks (by using the **ADD RACK** button).

Parameter	Description
Source Module	Source module type. This field is automatically selected depending on the destination module of the last rack (1756-CNB or 1756-ENBT).
Source Module IP Address	IP address of the source module (only applicable for 1756-ENBT)
Source Module Node Address	Node address of the source module (only applicable for 1756-CNB)
Destination Module	Select the destination module associated to the source module in the rack. The connection between the source and destination modules is performed through the backplane.
Destination Module Slot Number	The slot number where the destination MVI56E module is located.

To use the CIPconnect Path Editor, follow these steps.

1 Configure the path between the 1756-ENBT connected to your PC and the MVI56E-MNETR module.
   - If the module is located in a remote rack, add more racks to configure the full path.
   - The path can only contain ControlNet or Ethernet/IP networks.
   - The maximum number of supported racks is six.

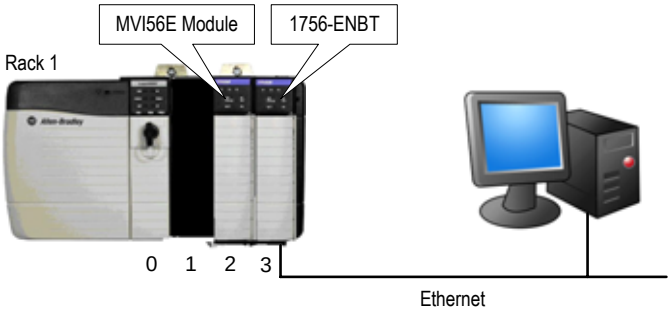
2 Click **CONSTRUCT CIP PATH** to build the path in text format

3 Click **OK** to confirm the configured path.

The following examples should provide a better understanding on how to set up the path for your network.

2.3.1 Example 1: Local Rack Application

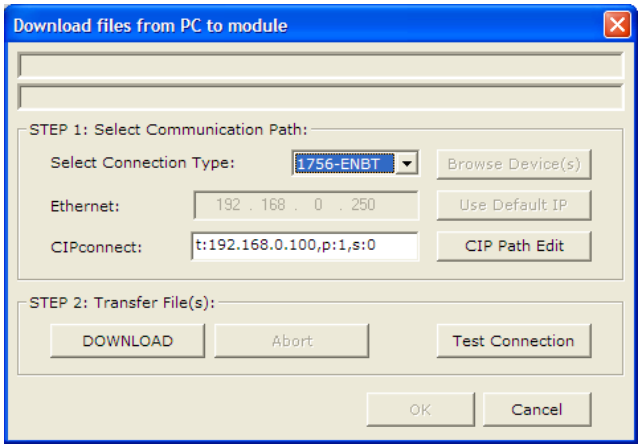
For this example the MVI56E-MNETR module is located in the same rack as the 1756-ENBT that is connected to the PC.



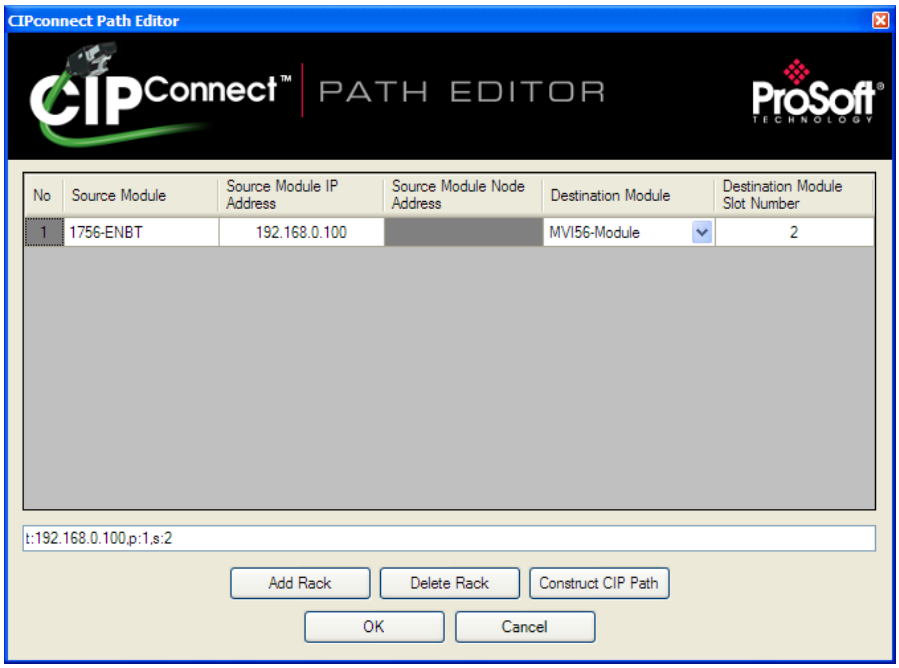
Rack 1

Slot	Module	Network Address
0	ControlLogix Processor	-
1	Any	-
2	MVI56E-MNETR	-
3	1756-ENBT	IP=192.168.0.100

1 In the Download window, click **CIP PATH EDIT.**

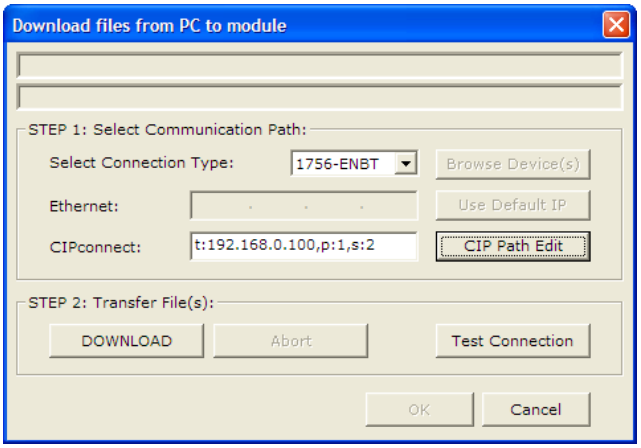


2 Configure the path as shown in the following illustration, and click **CONSTRUCT CIP PATH** to build the path in text format.

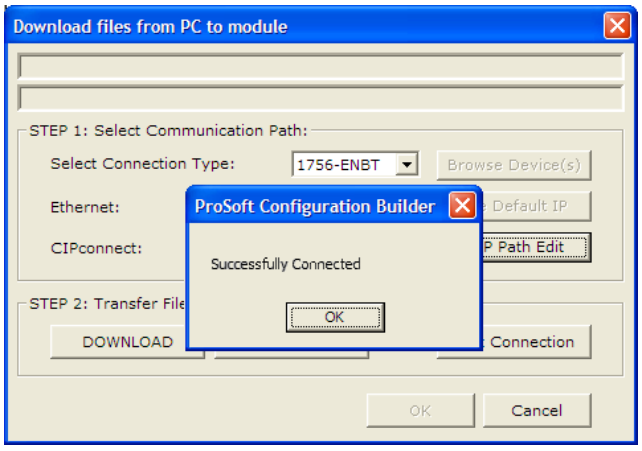


Click **OK** to close the **CIP PATH EDITOR** and return to the **DOWNLOAD** dialog box.

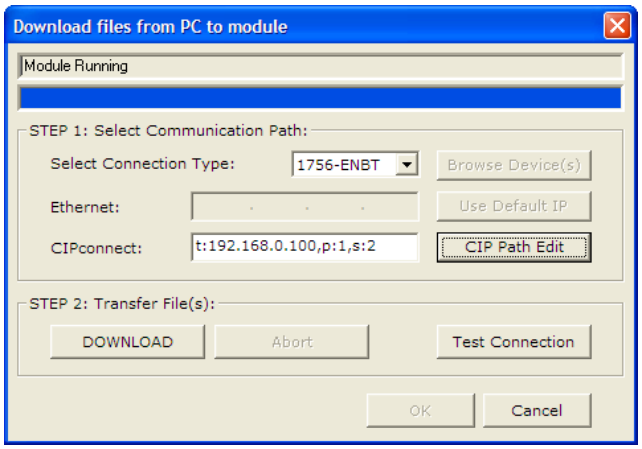
3 Check the new path in the download text box.



4 Click **TEST CONNECTION** to verify that the physical path is available. The following message should be displayed upon success.

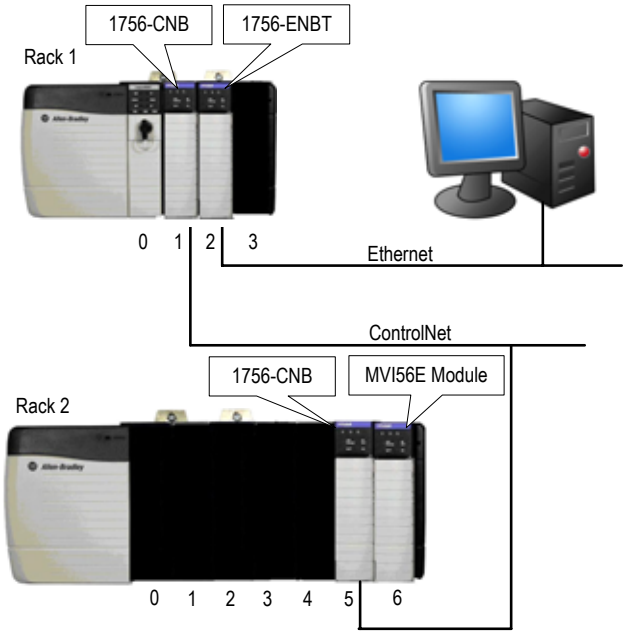


5 Click **OK** to close the Test Connection pop-up and then click **DOWNLOAD** to download the configuration files to the module through the path.



2.3.2 Example 2: Remote Rack Application

For this example, the MVI56E-MNETR module is located in a remote rack accessible through ControlNet, as shown in the following illustration.



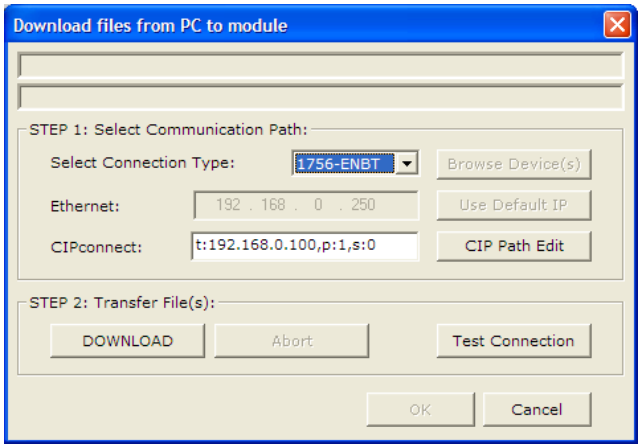
Rack 1

Slot	Module	Network Address
0	ControlLogix Processor	-
1	1756-CNB	Node = 1
2	1756-ENBT	IP=192.168.0.100
3	Any	-

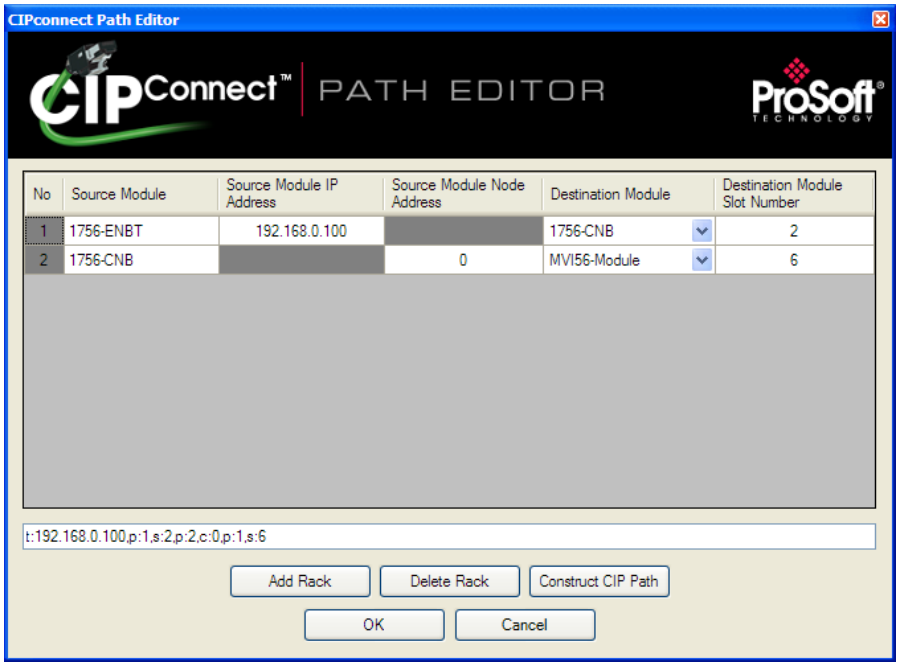
Rack 2

Slot	Module	Network Address
0	Any	-
1	Any	-
2	Any	-
3	Any	-
4	Any	-
5	1756-CNB	Node = 2
6	MVI56E-MNETR	-

1 In the Download window, click **CIP PATH EDIT**.

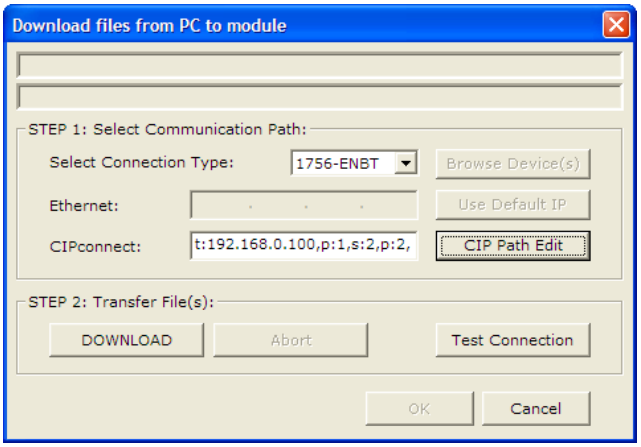


2 Configure the path as shown in the following illustration for this example and click **CONSTRUCT CIP PATH** to build the path in text format.

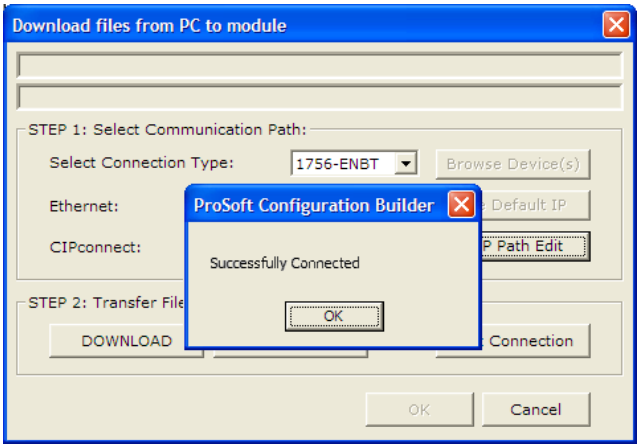


Click **OK** to close the **CIP PATH EDITOR** and return to the **DOWNLOAD** dialog box.

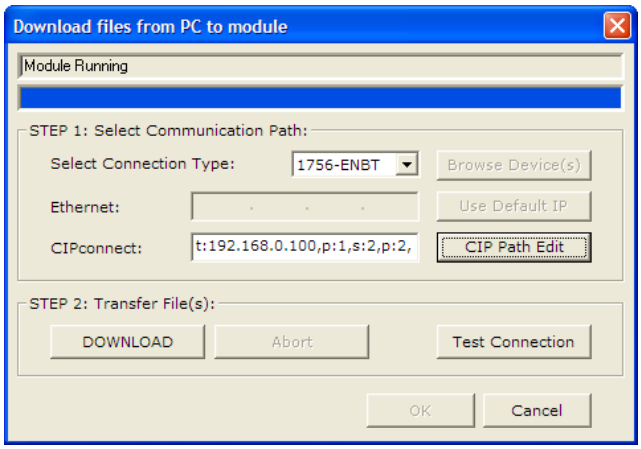
3 Check the new path in the download text box.



4 Click **TEST CONNECTION** to verify that the physical path is available. The following message should be displayed upon success.



5 Click **DOWNLOAD** to download the configuration files to the module through the path.



3 Ladder Logic

In This Chapter

- MNETRModuleDef 79
- Modbus Message Data..... 84

Ladder logic is required for application of the MVI56E-MNETR module. Tasks that must be handled by the ladder logic are module data transfer, special block handling, and status data receipt. Additionally, a power-up handler may be needed to handle the initialization of the module's data and to clear any processor fault conditions.

The sample 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 MNETRModuleDef

All data related to the MVI56E-MNETR is stored in a user defined data type. An instance of the data type is required before the module can be used. This is done by declaring a variable of the data type in the **CONTROLLER TAGS EDIT TAGS** dialog box.

The following table describes the structure of the object.

Name	Data Type	Description
DATA	MNETRDATA	These objects hold data to be transferred between the processor and the MVI56E-MNETR module
STATUS	MNETRSTATUS	This object views the status of the module.
CONTROL	MNETRCONTROL	This object contains the data structure required for the processor to request special tasks from the module
UTIL	MNETRUTIL	This data object stores the variables required for the data transfer between the processor and the module.

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

3.1.1 MNETRDATA

This object holds data to be transferred between the processor and the MVI56E-MNETR module. The user data is the read and write data transferred between the processor and the module as "pages" of data up to 40 words long.

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

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

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

3.1.2 MNETRSTATUS

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

The following table describes the structure of this object.

Name	Data Type	Description
PassCnt	INT	Program cycle counter
BlockStats	MNETRBlockStats	Status information for the data transfer operations between the processor and the module
Reserved1	INT	Reserved for future use
Reserved2	INT	Reserved for future use
MNETReq	INT	The number of MNET (Port 2000) requests received
MNETResp	INT	The number of MNET (Port 2000) responses sent
MBAPReq	INT	The number of MBAP (Port 502) requests received
MBAPResp	INT	The number of MBAP (Port 502) responses sent
ClientStatus	MNETRClientStats	Client Status Data

MNETRBlockStats

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

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

MNETRClientStats

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

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

Refer to MVI56E-MNETR Status Data Definition for a complete listing of the data stored in the status object.

3.1.3 MNETRCONTROL

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

The following table describes the structure of this object.

Name	Data Type	Description
BootTimer	TIMER	Timer to clear warmboot and coldboot
WarmBoot	BOOL	Triggers a Cold Boot Command
ColdBoot	BOOL	Triggers a Warm Boot Command

Name	Data Type	Description
EventCmdTrigger	BOOL	Triggers the Event Command.
EventCmd	MNETREventCmd	This object contains the attributes to define a Master command. An array of these objects is used for each port.
CmdControl	MNETRCmdControl	Controls the execution of the commands listed in the configuration under the [MNET Client 0 Commands] section.
PassThru	MNETRPassThru	Transferrs a remote Client's commands through the MNETR Module straight into the Processor's Controller tags.

MNETREventCmd

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

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

MNETRCmdControl

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

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

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

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

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

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

Name	Data Type	Description
CmdIndex	INT[6]	The position of the initial command to execute from the Client command list.
NumberOfCommands	INT	The number of commands to execute from the Client command list
CommandsAddedtoQueue	INT	
CmdControlTrigger	BOOL	

MNETRPassThru

During pass-through operation, write messages received at the MVI56E-MNETR server write messages through to the processor. It is the responsibility of the ladder logic to process the message received using this feature. Two data objects are required for this mode: a variable to hold the length of the message and a buffer to hold the message.

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

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

The following table describes the structure of this object.

Name	Data Type	Description
MBControl1	CONTROL	Reserved
MBControl2	CONTROL	Reserved
MBMsg	SINT[500]	The Modbus message received
MBScratch	INT[3]	Reserved
MBOffsetBit	INT	The starting bit address for the Modbus data set

Name	Data Type	Description
MBOffset	INT	The starting word address for the Modbus data set
MBMsgLen	INT	The length of the Modbus message in bytes
mbdouble	DINT	Reserved
MBCoil	MNETRCoilArray	Conversion from Bool to INT data types

MNETRCoilArray

Name	Data Type	Description
Boolean	BOOL[416]	Conversion from Bool to INT data types

3.1.4 MNETRUTIL

This data object stores the variables required for the data transfer between the processor and the MVI56E-MNETR module. The following table describes the structure of this object.

Name	Data Type	Description
LastRead	INT	Index of last read block
LastWrite	INT	Index of last write block
BlockIndex	INT	Computed block offset for data table
ReadDataSizeGet	INT	Gets ReadData Array Length.
WriteDataSizeGet	INT	Gets WriteData Array Length.

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

3.2 Modbus Message Data

During pass-through operation, write messages received at the MVI56E-MNETR server write messages through to the processor. It is the responsibility of the ladder logic to process the message received using this feature. Two data objects are required for this mode: a variable to hold the length of the message and a buffer to hold the message.

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

Additionally, formatted message blocks can be sent from the module to the processor when the pass-through option is selected using the format selection (codes 2 or 3 in the MNET.CFG file). These blocks require less decoding than the unformatted blocks. Refer to Pass-Through Control Blocks (page 111) for a full discussion on utilizing the pass-through option in an application.

4 Diagnostics and Troubleshooting

In This Chapter

- Reading Status Data from the Module 85
- The Diagnostics Menu 86
- Monitoring Module Information 89
- Monitoring Backplane Information 90
- Monitoring Database Information 91
- Monitoring MNET Client Information 92
- Monitoring MNET Server Information 93
- Data Analyzer 93
- LED Status Indicators 97

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

- Status data values are transferred from the module to the processor.
- Data contained in the module can be viewed in ProSoft Configuration Builder through the Ethernet port.
- LED status indicators on the front of the module provide information on the module's status.

4.1 Reading Status Data from the Module

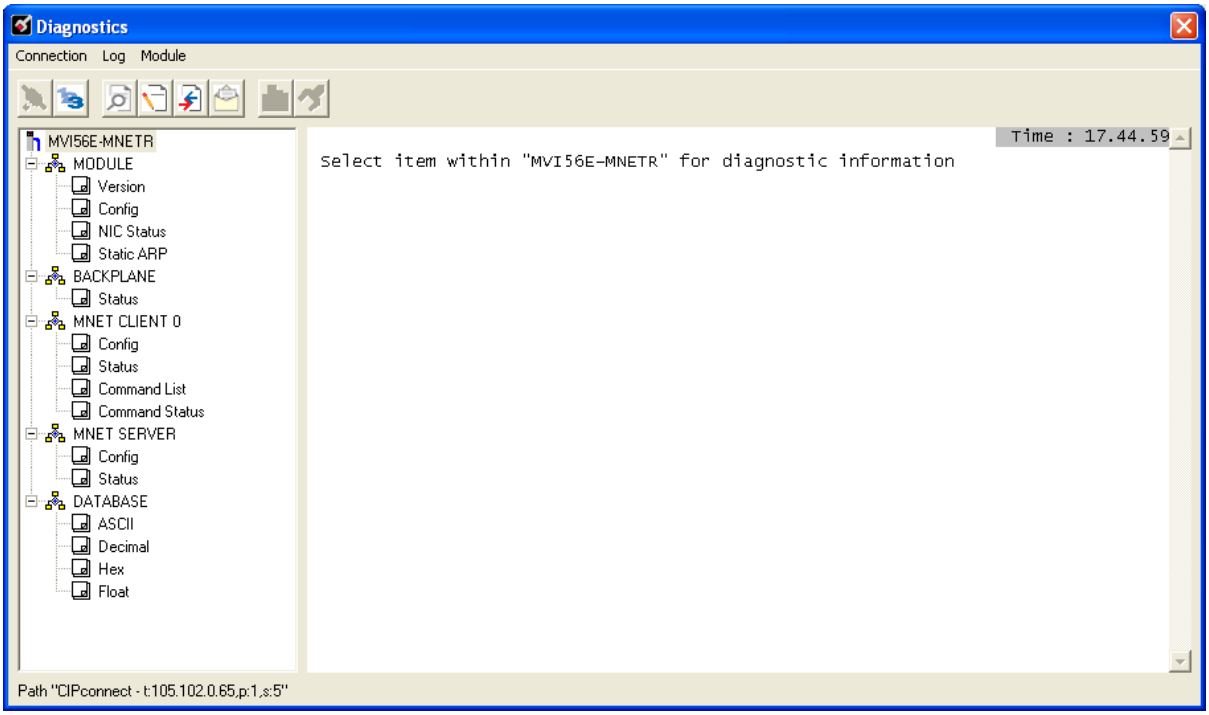
The MVI56E-MNETR module returns a 47-word Status Data block (page 119) that can be used to determine the module's operating status. This data is located in the module's database at a user set location. This data is transferred to the ControlLogix processor continuously.

The Configuration/Debug port provides the following functionality:

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

4.2 The Diagnostics Menu

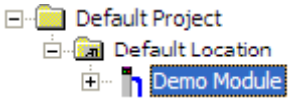
The Diagnostics menu for this module is arranged as a tree structure, with the Main Menu at the top of the tree, and one or more sub-menus for each menu command. The first menu you see when you connect to the module is the Main menu.



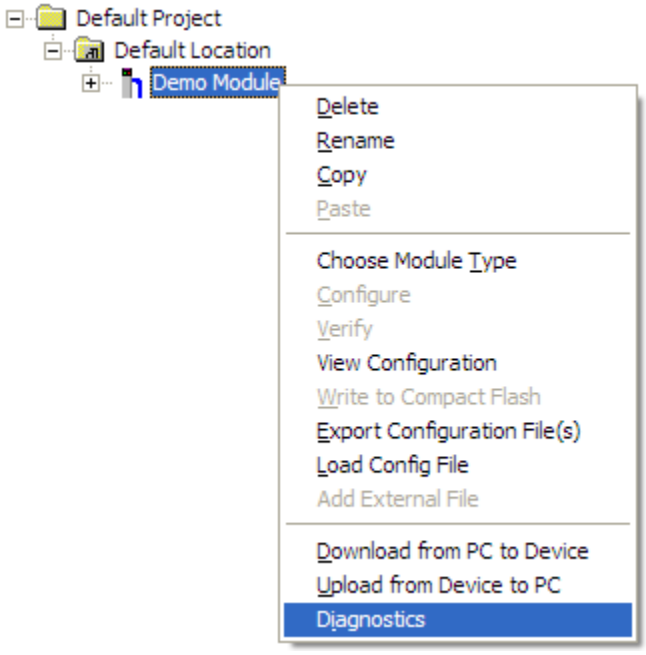
4.2.1 Using the Diagnostics Menu in ProSoft Configuration Builder

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

1 Start PCB program with the application file to be tested. Right click over the module icon.

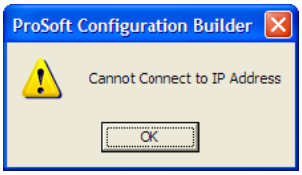


2 On the shortcut menu, choose **DIAGNOSTICS.**



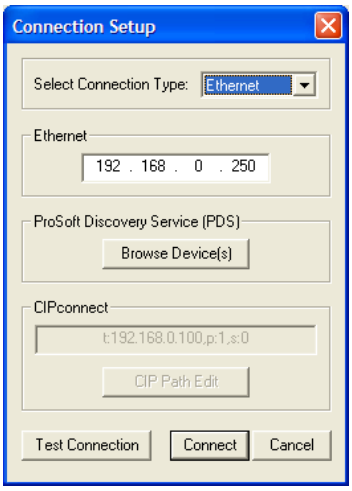
This action opens the **DIAGNOSTICS** dialog box.

If there is no response from the module,

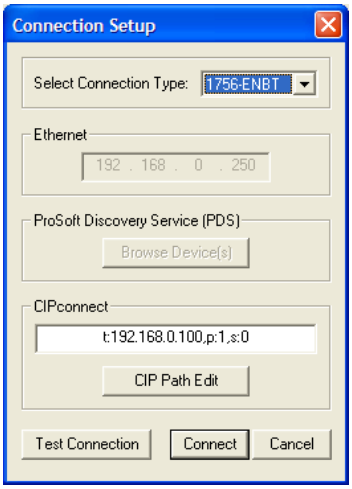


1 Click the **SET UP CONNECTION** button to browse for the module's IP address.

2 On the **CONNECTION SETUP** dialog box, click the **TEST CONNECTION** button to verify if the module is accessible with the current settings

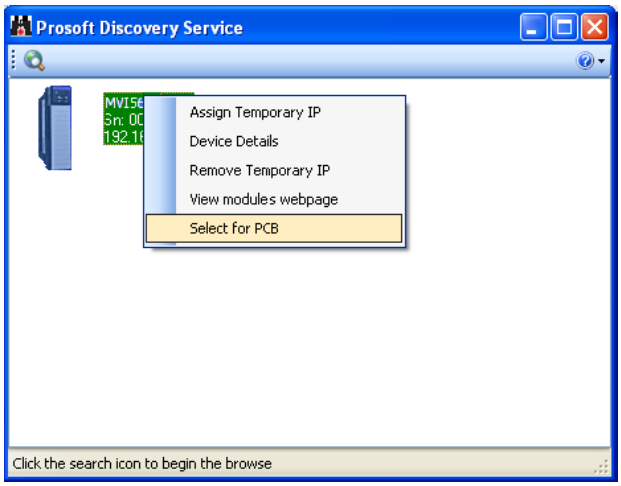


You can also use CIPconnect to connect to the module through a 1756-ENBT card.



Refer to Using CIPconnect to Connect to the Module for information on how to construct a CIP path (page 68).

3 If PCB is still unable to connect to the module, click the **BROWSE DEVICE(S)** button to open the **PROSOFT DISCOVERY SERVICE**. Select the module, then right click and choose **SELECT FOR PCB**.



Close ProSoft Discovery Service, and click the **CONNECT** button again.

4 If all of these troubleshooting steps fail, verify that the Ethernet cable is connected properly between your computer and the module, either through a hub or switch (using the grey cable) or directly between your computer and the module (using the red cable).

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

4.3 Monitoring Module Information

Use the **MODULE** menu to view configuration and hardware information for the MVI56E-MNETR module's backplane and Ethernet application port.

4.3.1 Version

Use the **VERSION** menu to view module hardware and firmware information.

```
MVI56E-MNETR > MODULE > Version :

Product Code                    :MNER
Revision                        :2.01
OpSys                           :0409
Run Number                      :2301
IP Address                      :105.102.0.25
MAC Address                     :00:0d:8d:00:3a:99
Program Scan Cnt                :51423
BP Driver Version               :2.06
BP API Version                  :1.01
Module Name                     :MVI56E-MNETR
Vendor ID                       :309
Device Type                     :12
Product Code                    :5010
Serial Number                   :000003EA
BP Revision                     :2.01
Slot                            :1
```

The values on this menu correspond with the contents of the module's Miscellaneous Status registers.

4.3.2 Config

Use the **CONFIGURATION** menu to view backplane configuration settings for the MVI56E-MNETR module.

The information on this menu corresponds with the configuration information in the **[MODULE]** settings in ProSoft Configuration Builder (page 53).

4.3.3 NIC Status

Use the **NIC STATUS** (Network Interface Card) menu to view configuration and status information for the MVI56E-MNETR module's Ethernet application port.

The information on this menu is useful for troubleshooting Ethernet network connectivity problems.

4.3.4 Static ARP

Use the **STATIC ARP** menu to view the list of IP and MAC addresses that are configured not to receive ARP (Address Resolution Protocol) messages from the module.

The Static ARP Table (page 65) defines a list of static IP addresses that the module will use when an ARP is required.

4.4 Monitoring Backplane Information

Use the **BACKPLANE** menu to view the backplane status information for the MVI56E-MNETR module.

4.4.1 Backplane Status

Click **STATUS** to view current backplane status, including

- Number of retries
- Backplane Status
- Fail Count
- Number of words read
- Number of words written
- Number of words parsed
- Error count
- Event count
- Command count

During normal operation, the Read, Write, and Parsing values should increment continuously, while the error value should not increment.

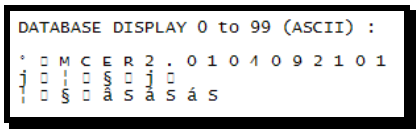
The status values on this menu correspond with the members of the MVI56E-MNETR Status object (page 119).

4.5 Monitoring Database Information

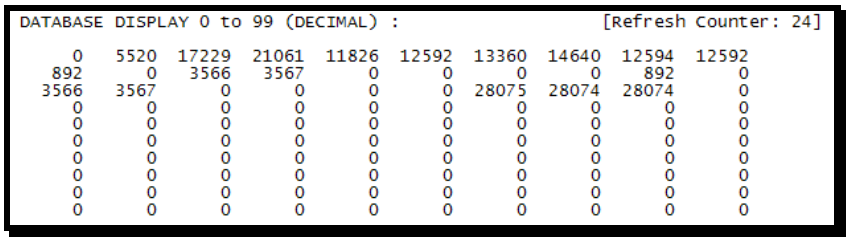
Use the **DATABASE** menu to view the contents of the MVI56E-MNETR module's internal database. The data locations on this menu corresponds with the MVI56E-MNETR Database Definition

You can view data in the following formats:

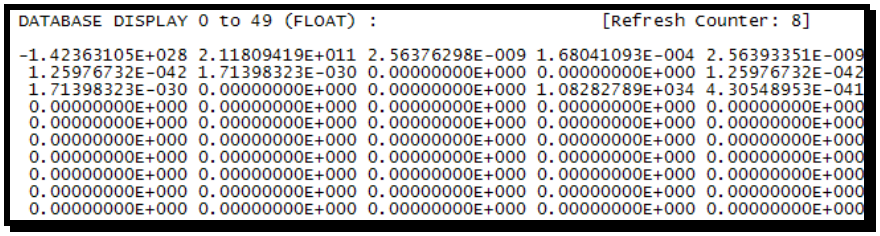
ASCII



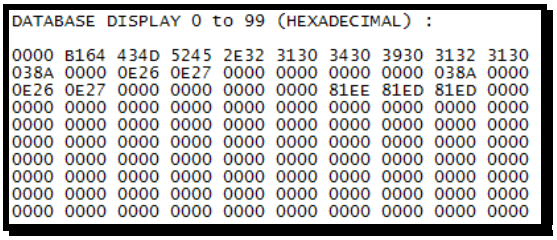
Decimal



Float



Hexadecimal



Use the scroll bar on the right edge of the window to view each page (100 words) of data.

4.6 Monitoring MNET Client Information

Use the **MNET CLIENT INFORMATION** menu to view the configuration and status information for the MNET Client.

4.6.1 Command List

Use the **COMMAND LIST** menu to view the command list settings for MNET Client 0. The information on this menu corresponds with the settings in the [MNET Client 0 Commands] settings in ProSoft Configuration Builder (page 57).

Use the scroll bar on the right edge of the window to view each MNET Client command.

4.6.2 Command Status

Use the **COMMAND STATUS** menu to view MNET Client 0 Command status.

A zero indicates no error.

A non-zero value indicates an error. Refer to Error Codes for an explanation of each value.

4.6.3 Config

Use the **CONFIGURATION** menu to view configuration settings for MNET Client 0.

The information on this menu corresponds with the configuration information in the [MNET Client 0] settings in ProSoft Configuration Builder (page 55).

4.6.4 Status

Use the **STATUS** menu to view status for MNET Client 0. During normal operation, the number of requests and responses should increment, while the number of errors should not change.

4.7 Monitoring MNET Server Information

Use the **MNET SERVER INFORMATION** menu to view the configuration and status information for the MNET Server.

4.7.1 Config

Use the **CONFIGURATION** menu to view configuration settings for MNET Servers connected to the MNET Client.

The information on this menu corresponds with the configuration information in the [MNET Servers] settings in ProSoft Configuration Builder (page 63).

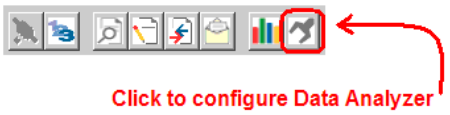
4.7.2 Status

Use the **STATUS** menu to view the status of each MNET Server connected to the MNET Client 0. During normal operation, the number of requests and responses should increment, while the number of errors should not change.

4.8 Data Analyzer

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

4.8.1 Configuring the Data Analyzer



Select Timing Interval

Time Ticks help you visualize how much data is transmitted on the port for a specified interval. Select the interval to display, or choose No Ticks to turn off timing marks.

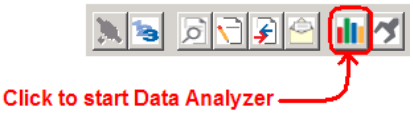
Select the Communication Port to Analyze

You can view incoming and outgoing data for one application port at a time. Choose the application port to analyze.

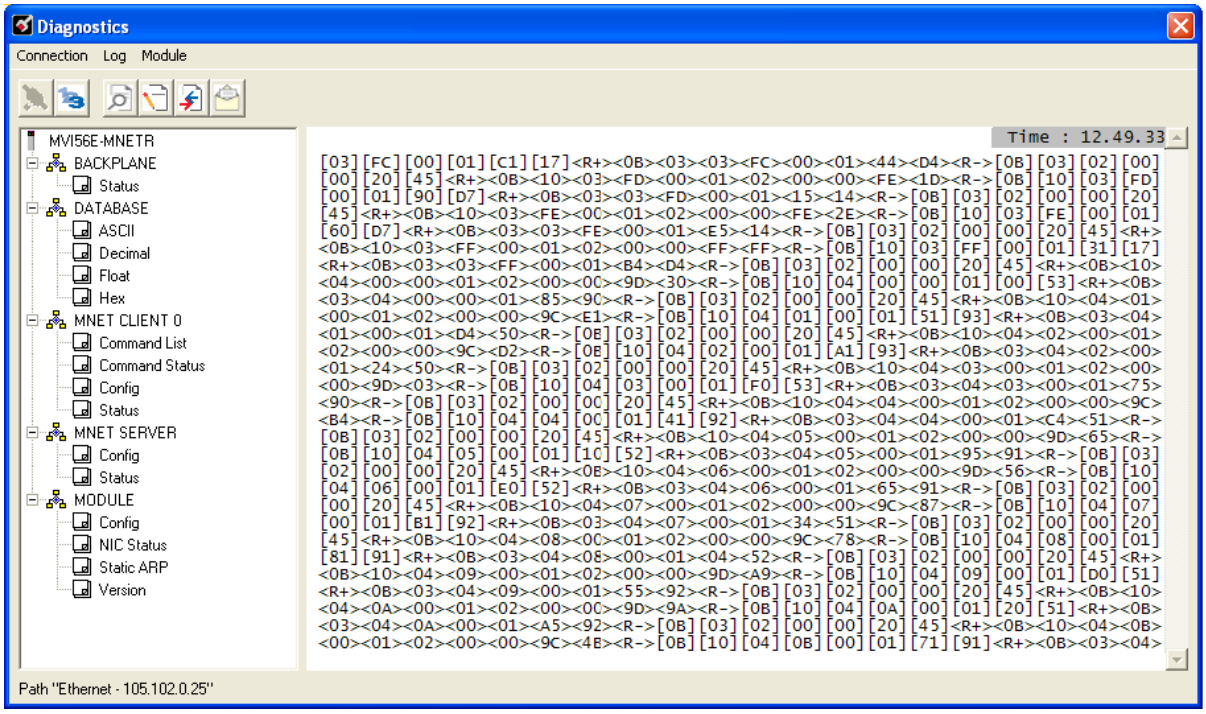
Select the Data Format

You can view incoming and outgoing data in Hexadecimal (HEX) or Alphanumeric (ASCII) format.

4.8.2 Starting the Data Analyzer



The following illustration shows an example of the Data Analyzer output.



The Data Analyzer can display the following special characters.

Character	Definition
[ ]	Data enclosed in these characters represent data received on the port.
< >	Data enclosed in these characters represent data transmitted on the port.
<R+>	These characters are inserted when the RTS line is driven high on the port.
<R->	These characters are inserted when the RTS line is dropped low on the port.
<CS>	These characters are displayed when the CTS line is recognized high.
_TT_	These characters are displayed when the "Time Tick" is set to any value other than "No Ticks".

4.8.3 Stopping the Data Analyzer

Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please stop the data analyzer. This action will allow the module to resume its normal high speed operating mode.

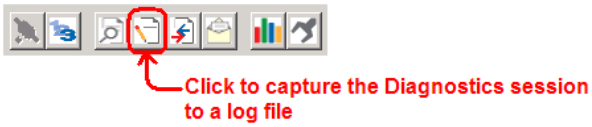
4.8.4 Data Analyzer Tips

For most applications, HEX is the best format to view the data, and this does include ASCII based messages (because some characters will not display in the Diagnostics window, and by capturing the data in HEX, we can figure out what the corresponding ASCII characters are supposed to be).

The Tick value is a timing mark. The module will print a _TT for every xx milliseconds of no data on the line. Usually 10milliseconds is the best value to start with.

To save a capture file of your Diagnostics session

1. After you have selected the Port, Format, and Tick, we are now ready to start a capture of this data.

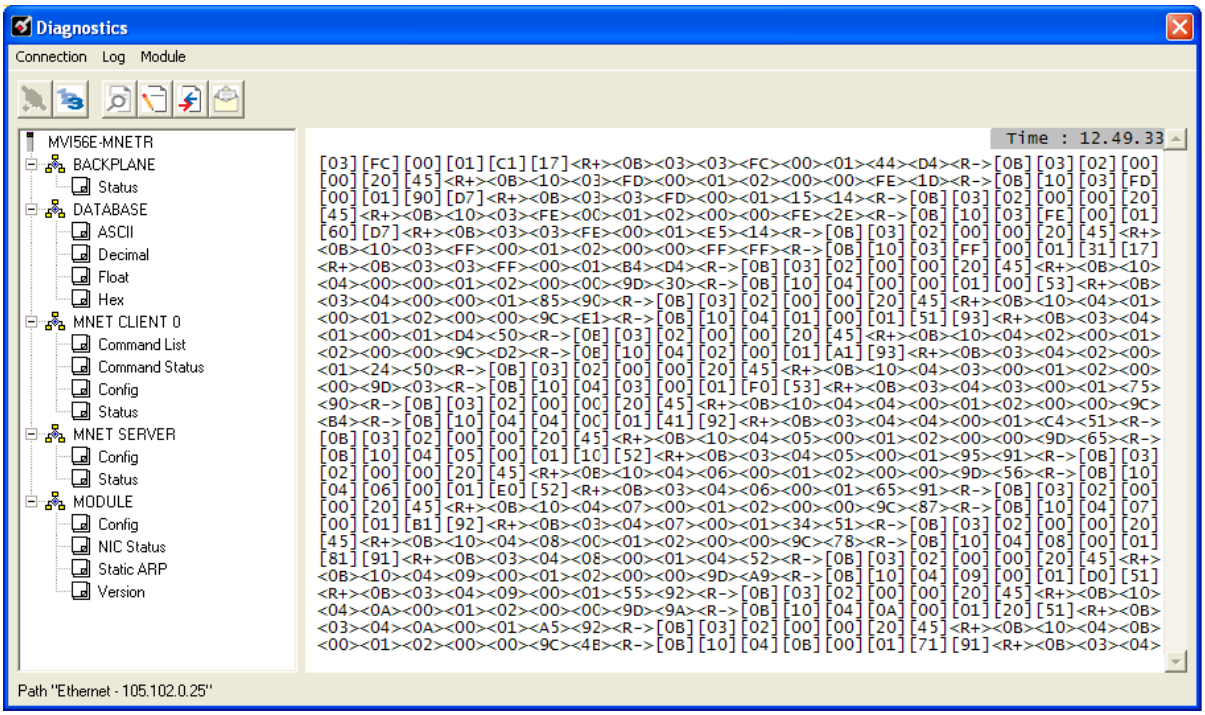


2. When you have captured the data you want to save, click again to stop capturing data.

You have now captured, and saved the file to your PC. This file can now be used in analyzing the communications traffic on the line, and assist in determining communication errors. The log file name is PCB-Log.txt, located in the root directory of your hard drive (normally Drive C).

Now you have everything that shows up on the Diagnostics screen being logged to a file called PCB-Log.txt. You can email this file to ProSoft Technical Support for help with issues on the communications network.

To begin the display of the communications data, start the Data Analyzer. When the Data Analyzer is running, you should see something like this.



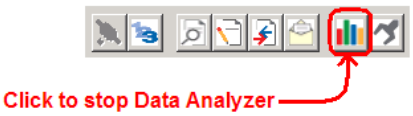
The <R+> means that the module is transitioning the communications line to a transmit state.

All characters shown in <> brackets are characters being sent out by the module.

The <R-> shows when the module is done transmitting data, and is now ready to receive information back.

And finally, all characters shown in the [ ] brackets is information being received from another device by the module.

After taking a minute or two of traffic capture, stop the Data Analyzer.



4.9 LED Status Indicators

The scrolling LED display indicates the module's operating status as follows:

Initialization Messages

Code	Message
Boot / DDOK	Module is initializing
Ladd	Module is waiting for required module configuration data from ladder logic to configure the application port(s)
Waiting for Processor Connection	Module did not connect to processor during initialization ▪ Sample ladder logic or AOI is not loaded on processor ▪ Module is located in a different slot than the one configured in the ladder logic/AOI ▪ Processor is not in RUN or REM RUN mode
Last config: <date>	Indicates the last date when the module changed its IP address. You can update the module date and time through the module's web page, or with the optional MVI56E Utility Add-On Instruction.
C0 (Client): CmdCnt: X MinDly : X CmdOffs: X RespTmout : X Retries : X ErrOffs : X ARPTmout : X ErrDelay : X FltFlag : X FltSt : X FltOffs : X SVR (server) : BIOffs: X WIOffs : X OutOffs : X HoldOffs : X FltFlag : X FltSt : X FltSt : X CommTmout : X	After power up and every reconfiguration, the module will display the configuration the application port(s). The information consists of: **Client** ▪ CmdCnt : number of commands configured for the client ▪ MinDly : Minimum Command Delay parameter ▪ CmdOffs : Command Error Pointer parameter ▪ RespTmout : Response Timeout parameter ▪ Retries : Retry Count parameter ▪ ErrOffs : Error/Status Offset parameter ▪ ARPTmout : ARP Timeout parameter ▪ ErrDelay: Command Error Delay parameter ▪ FltFlag: Float Flag parameter ▪ Flt St : Float Start parameter ▪ FltOffs : Float Offset parameter **Server** ▪ BIOffs: Bit Input Offset parameter ▪ WIOffs : Word Input Offset parameter ▪ OutOffs : Output offset parameter ▪ HoldOffs : Holding Register offset parameter ▪ FltFlag: Float Flag parameter ▪ Flt St : Float Start parameter ▪ FltOffs : Float Offset parameter

Operation Messages

After the initialization step, the following message pattern will be repeated.

```
<Backplane Status> <IP Address> <Backplane Status> <Port Status>
```

Code	Message
<Backplane Status>	OK: Module is communicating with processor ERR: Module is unable to communicate with processor. For this scenario, the <Port Status> message above is replaced with "Processor faulted or is in program mode".
<IP Address>	Module IP address
<C0>	OK: Port is communicating without error Communication Errors: port is having communication errors. Refer to PCB diagnostics for further information about the error.

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

Module	Color	Status	Indication
APP	Amber	On	The MVI56E-MNETR is working normally.
		Off	The MVI56E-MNETR module program has recognized a communication error on its Application Port(s).
OK	Red/ Green	Off	The card is not receiving adequate power, or is not securely plugged into the rack.
		Green	The module is operating normally.
		Red	The module has detected an internal error, or is being initialized. If the LED remains red for over 10 seconds, the module is not working. Remove it from the rack and re-insert it to restart its internal 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.

4.9.1 Configuration Error Word

If a configuration error is found for the Client, the Client configuration error word will have a value other than zero. The configuration error word bits have the following definitions:

Bit	Description	Value
0		0x0001
1		0x0002
2		0x0004
3		0x0008
4	Invalid retry count parameter	0x0010
5	The float flag parameter is not valid.	0x0020
6	The float start parameter is not valid.	0x0040
7	The float offset parameter is not valid.	0x0080

Bit	Description	Value
8	The ARP Timeout is not in range (ARP Timeout parameter 0 or greater than 60000 milliseconds) and will default to 5000 milliseconds.	0x0100
9	The Command Error Delay is > 300 and will default to 300.	0x0200
10		0x0400
11		0x0800
12		0x1000
13		0x2000
14		0x4000
15		0x8000

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

4.9.2 Ethernet LED Indicators

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

4.9.3 Clearing a Fault Condition

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

To clear the condition, follow these steps:

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

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 Support.

4.9.4 Troubleshooting

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

Processor Errors

Problem Description	Steps to take
Processor Fault	Verify that the module is plugged into the slot that has been configured for the module. 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. Verify that all modules in the rack are configured in the ladder logic.

Module Errors

Problem Description	Steps to take
BP ACT LED remains off or blinks slowly	This indicates that backplane transfer operations are failing. Connect to the module's Configuration/Debug port to check this.
MVI56E modules with scrolling LED display: *<Backplane Status>* condition reads ERR	To establish backplane communications, verify the following items: ▪ The processor is in Run mode. ▪ The backplane driver is loaded in the module. ▪ The module is configured for read and write block data transfer. ▪ The ladder logic handles all read and write block situations. ▪ The module is configured in the processor.
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 the card in the rack, and then restore power to the rack.

5 Reference

In This Chapter

- Product Specifications 101
- Functional Overview 103
- Ethernet Cable Specifications 118
- MVI56E-MNETR Status Data Definition 119
- Modbus Protocol Specification 121
- Using the Rung Import with Utility Add-On Instruction 132
- Adding the Module to an Existing Project 141
- Using the Sample Program - RSLogix Version 15 and earlier 144

5.1 Product Specifications

The MVI56E-MNETR Modbus TCP/IP Client/Server Communication Module allows Rockwell Automation® ControlLogix® processors to interface easily with other Modbus TCP/IP compatible devices.

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

5.1.1 General Specifications

- Backward-compatible with previous MVI56-MNETR versions
- Single-Slot ControlLogix® backplane compatible
- 10/100 Mbps auto crossover detection Ethernet configuration and application port
- User definable module data memory mapping of up to 5000, 16-bit registers
- CIPconnect®-enabled network configuration and diagnostics monitoring using ControlLogix 1756-ENxT and 1756-CNB modules and EtherNet/IP™ pass-through communication
- Reduced I/O image size designed specifically to optimize remote rack implementations
- 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 (AOIs) are used for data transfer between module and processor

- Internal web server provides access to product documentation, module status, diagnostics, and firmware updates
- 4-character alpha-numeric scrolling LED display of status and diagnostics data in plain English – no cryptic error or alarm codes to decipher
- ProSoft Discovery Service (PDS) software used to locate the module on the network and assign temporary IP address
- Personality Module - a non-volatile, industrial-grade Compact Flash (CF)card used to store network and module configuration, allowing quick in-the-field product replacement by transferring the CF card

5.1.2 Functional Specifications

- The MVI56E-MNETR transfers data in small I/O blocks than the MVI56E-MNET, which makes it ideal for installations in remote racks or where bandwidth is limited.
- Works well with redundant ControlLogix Programmable Automation Controllers (PACs) using ControlNet.
- Module appears to the ControlLogix processor as an input/output (I/O) module
- 40-word scheduled I/O image blocks used for data transfers require significantly less bandwidth than the MVI56E-MNET
- Retrieving module status and executing special functions (command control, event commands, etc.) are supported in ladder logic by special block transfer codes

5.1.3 Hardware Specifications

Specification	Description
Backplane Current Load	800 mA @ 5 V DC 3 mA @ 24V DC
Operating Temperature	0° to 60°C (32° to 140°F)
Storage Temperature	-40° to 85°C (-40° to 185°F)
Shock	30g Operational 50g non-operational Vibration: 5 g from 10 to 150 Hz
Relative Humidity	5% to 95% (non-condensing)
LED Indicators	Battery Status (ERR) Application Status (APP) Module Status (OK)
4-Character, Scrolling, Alpha-Numeric LED Display	Shows Module, Version, IP, Application Port Setting, Port Status, and Error Information
Debug/Configuration/Application Ethernet port (E1)	
Ethernet Port	10/100 Base-T, RJ45 Connector, for CAT5 cable Link and Activity LED indicators Auto-crossover cable detection
Shipped with Unit	5-foot Ethernet straight-through cable

5.2 Functional Overview

This section describes how the MVI56E-MNETR module transfers data between itself and the processor, and how it implements the Modbus TCP/IP protocol.

5.2.1 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 MVI56E-MNETR module works both as a client and as a server.

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

The MVI56E-MNETR module acts as an input/output module between devices on a MODBUS TCP/IP network and the Rockwell Automation backplane. The module uses an internal database to pass data and commands between the processor and the client and server devices on the MODBUS TCP/IP network.

5.2.2 Module Power Up

On power up the module begins performing the following logical functions:

1. Initialize hardware components
   - Initialize ControlLogix backplane driver
   - Test and Clear all RAM
   - Read configuration for module from MNET.CFG file on Compact Flash Disk
2. Initialize Module Register space
3. Enable Server Drivers
4. Enable Client Driver

When the module has received the configuration, the module will begin communicating with other nodes on the network, depending on the configuration.

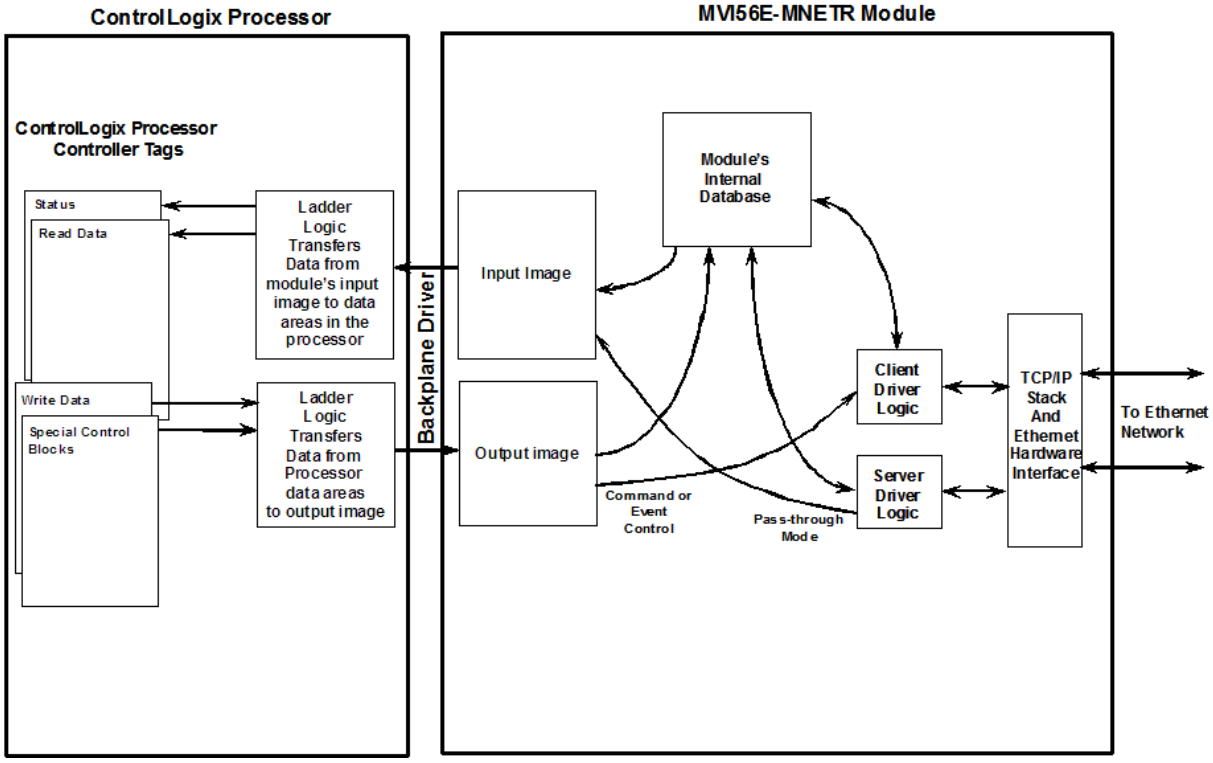
5.2.3 Backplane Data Transfer

The MVI56E-MNETR module communicates directly over the ControlLogix backplane. Data is paged between the module and the ControlLogix processor across the backplane using the module's input and output images. The update frequency of the images is determined by the scheduled scan rate defined by the user for the module and the communication load on the module. Typical updates are in the range of 1 to 10 milliseconds.

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

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

The following illustration shows the data transfer method used to move data between the ControlLogix processor, the MVI56E-MNETR module and the Modbus TCP/IP Network.



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

Module's Internal Database Structure

Data contained in this database is paged through the input and output images by coordination of the ControlLogix ladder logic and the MVI56E-MNETR module's program. Up to 40 words of data can be transferred from the module to the processor at a time. Up to 40 words of data can be transferred from the processor to the module. Each image has a defined structure depending on the data content and the function of the data transfer. The module uses the following block numbers:

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

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

Normal Data Transfer

Normal data transfer includes the paging of the user data found in the module's internal database in registers 0 to 4999 and the status data. These data are transferred through read (input image) and write (output image) blocks.

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

Status Read Data Block

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

Offset	Description	Length
0	Write Block ID	1
1	Program Scan Counter	1
2 to 7	Block Transfer Status	6
8 to 9	Reserved Server Status	2
10 to 11	MNET Server Status	2
12 to 13	MBAP Server Status	2
14 to 23	MNET Client Status	10
24 to 40	Reserved	17
41	Read Block ID (-1 or 0)	1

MVI56E-MNETR Client Status

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

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.

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

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

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

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

R1W1→R2W2→R3W1→R1W2→R2W1→R3W2→R1W1→

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

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.

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

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

Initialize Output Data

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

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

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

The following table describes the structure of the response block.

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

Command Control Blocks

Command Control Blocks are special optional blocks used to request special tasks from the module. The current version of the software supports four command control blocks; event command control, command control, warm boot, and cold boot.

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

Event Command

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

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

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

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

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

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

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

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

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

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

Command Control

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

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

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

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

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

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

Warm Boot

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

Offset	Description	Length
0	9998	1
1 to 41	Spare	41

Cold Boot

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

Offset	Description	Length
0	9999	1
1 to 41	Spare	41

Pass-Through Control Blocks

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

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

Unformatted

Unformatted Pass-Through Command (Read Block)

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

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

Unformatted Pass-Through Command (Write Block)

Offset	Description	Length
0	9996	1
1 to 41	Spare	41

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

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

Formatted

Formatted Pass-Through Command Blocks (Read Block)

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

Formatted Pass-Through Command Blocks (Read Block)

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

Formatted Pass-Through Command Blocks (Read Block)

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

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

Formatted Pass-Through Response (Write Block)

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

Formatted Pass-Through Response (Write Block)

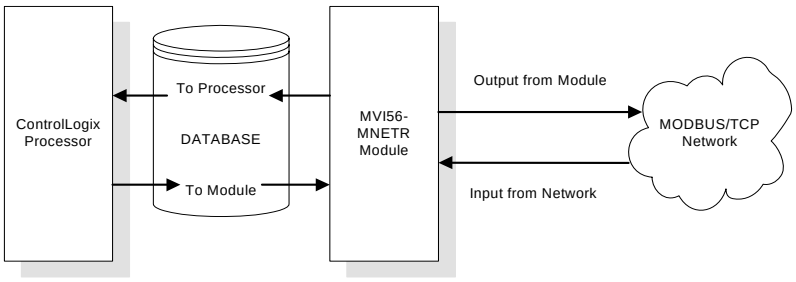
Offset	Description	Length
0	9959	1
1 to 41	Spare data area	41

Formatted Pass-Through Response (Write Block)

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

5.2.4 Data Flow between MVI56E-MNETR Module and ControlLogix Processor

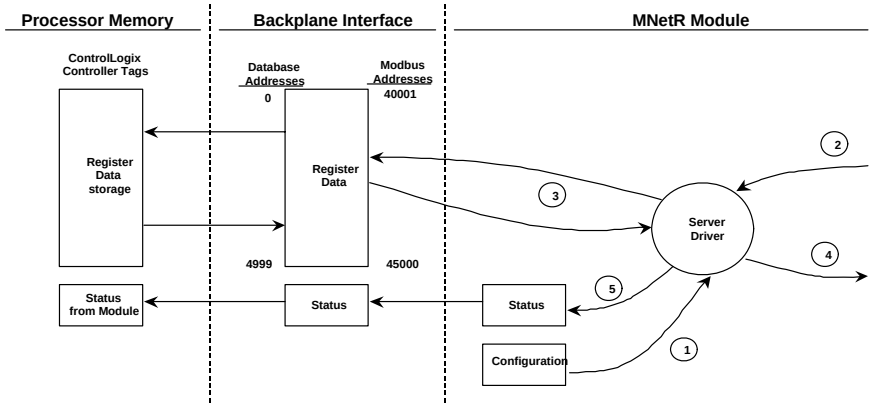
The following topics describe the flow of data between the two pieces of hardware (ControlLogix processor and MVI56E-MNETR module) and other nodes on the Modbus TCP/IP network under the module's different operating modes. The module contains both servers and a client. The servers accept TCP/IP connections on service ports 502 (MBAP) (10 servers) and 2000 (MNET) (10 servers). The client can generate either MBAP or MNET requests dependent on the service port selected in the command.



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

Server Driver

The Server Driver allows the MVI56E-MNETR module to respond to data read and write commands issued by clients on the Modbus TCP/IP network. The following illustration and associated table describe the flow of data into and out of the module.



1 The server driver receives the configuration information from the configuration file on the Compact Flash Disk, and the module initializes the servers.

2 A 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 qualifies the message before accepting it into the module.

3 When the module accepts the command, the data is immediately transferred to or from the internal database in the module. If the command is a read command, the data is read out of the database and a response message is built. If the command is a write command, the data is written directly into the database and a response message is built. If the pass-through feature is utilized, the write message is transferred directly to the processor and is not written to the module's database.

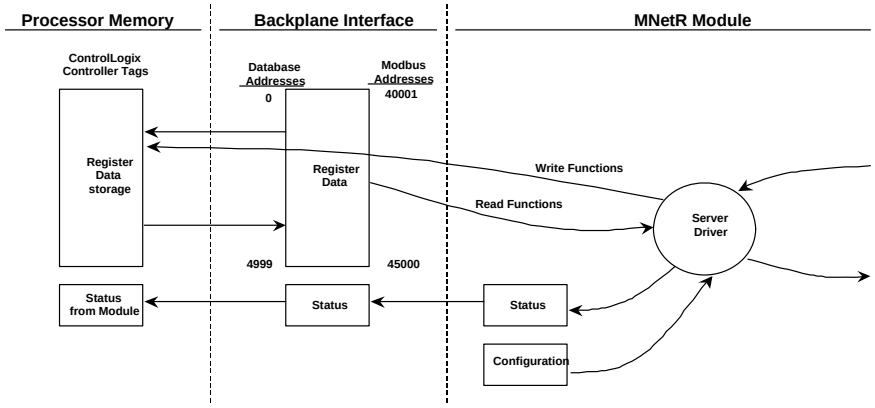
4 When the data processing has been completed in Step 3, the response is issued to the originating master node.

5 Counters are available in the Status Block that permit the ladder logic program to determine the level of activity of the Server Driver.

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

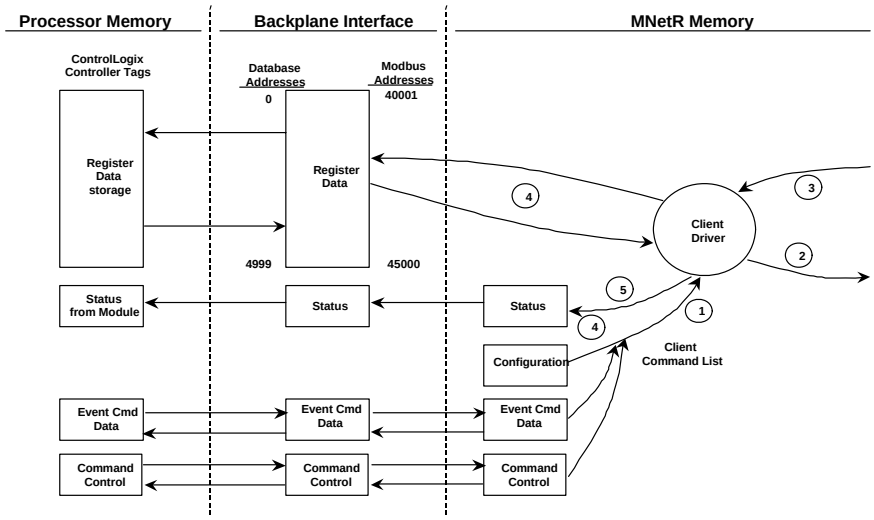
An exception to this normal mode is when the pass-through mode is implemented. In this mode, all write requests will be passed directly to the processor and will not be placed in the database. This permits direct, remote control of the processor without the intermediate database. This mode is especially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

following illustration describes the data flow for a slave port with pass-through enabled:



Client Driver

In the client driver, the MVI56E-MNETR module issues read or write commands to servers on the Modbus TCP/IP network. These commands are user configured in the module via the Client Command List received from the module's configuration file (MNET.CFG) or issued directly from the ControlLogix processor (event command control). 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 flow chart and table describe the flow of data into and out of the module.



1 The client driver obtains configuration data from the MNET.CFG file when the module restarts. The configuration data obtained 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 the other nodes on the Modbus TCP/IP network.

2 When configured, the client driver begins transmitting read and/or write commands to the other nodes on the network. If writing data to another node, the data for the write command is obtained from the module's internal database to build the command.
3 Presuming successful processing by the node specified in the command, a response message is received into the client driver for processing.
4 Data received from the node on the network is passed into the module's internal database, assuming a read command.
5 Status data is returned to the processor for the client and a Command List error table can be established in the module's internal database.

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 0 Client Command Error Pointer in the MNET.CFG file 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 Protocol Errors

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

Module Communication Error Codes

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

MNET Client Specific Errors

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

Command List Entry Errors

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

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

5.3 Ethernet Cable Specifications

The recommended cable is Category 5 or better. A Category 5 cable has four twisted pairs of wire, which are color-coded and cannot be swapped. The module uses only two of the four pairs.

The Ethernet port on the module is Auto-Sensing. You can use either a standard Ethernet straight-through cable or a crossover cable when connecting the module to an Ethernet hub, a 10/100 Base-T Ethernet switch, or directly to a PC. The module will detect the cable type and use the appropriate pins to send and receive Ethernet signals.

Ethernet cabling is like U.S. telephone cables, except that it has eight conductors. Some hubs have one input that can accept either a straight-through or crossover cable, depending on a switch position. In this case, you must ensure that the switch position and cable type agree.

Refer to Ethernet cable configuration (page 118) for a diagram of how to configure Ethernet cable.

5.3.1 Ethernet Cable Configuration

Note: The standard connector view shown is color-coded for a straight-through cable.

Crossover cable	
RJ-45 PIN	RJ-45 PIN
1 Rx+	3 Tx+
2 Rx-	6 Tx-
3 Tx+	1 Rx+
6 Tx-	2 Rx-

Straight- through cable	
RJ-45 PIN	RJ-45 PIN
1 Rx+	1 Tx+
2 Rx-	2 Tx-
3 Tx+	3 Rx+
6 Tx-	6 Rx-

5.3.2 Ethernet Performance - MVI56E Modules

Ethernet performance on the MVI56E-MNETR module can affect the operation of the MNETR application ports in the following ways.

- Accessing the web interface (refresh the page, download files, etc…) may impact MNETR performance
- High Ethernet traffic may impact MNETR performance (consider CIPconnect (page 68) for these applications and disconnect the module Ethernet port from the network.

5.4 MVI56E-MNETR Status Data Definition

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

Offset	Content	Description
0	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
1	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
2	Write Block Count	This field contains the total number of write blocks transferred from the processor to the module.
3	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
4	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
5	Command Block Count	This field contains the total number of command blocks received from the processor.
6	Error Block Count	This field contains the total number of block errors recognized by the module.
7	Reserved	Not Used
8	Reserved	Not Used
9	Reserved	Not Used
10	Reserved	Not Used
11	Reserved	Not Used
12	Reserved	Not Used
13	Reserved	Not Used
14	Reserved	Not Used
15	Reserved	Not Used
16	Reserved	Not Used
17	Reserved	Not Used
18	Reserved	Not Used
19	Reserved	Not Used
20	MNET Request Count	This counter increments each time a MNET (port 2000) request is received.
21	MNET Response Count	This counter is incremented each time a MNET (port 2000) response message is sent.
22	Reserved	Not Used
23	Reserved	Not Used
24	Reserved	Not Used
25	Reserved	Not Used
26	Reserved	Not Used
27	Reserved	Not Used
28	Reserved	Not Used
29	Reserved	Not Used

Offset	Content	Description
30	MBAP Request Count	This counter increments each time a MBAP (port 502) request is received.
31	MBAP Response Count	This counter is incremented each time a MBAP (port 502) response message is sent.
32	Reserved	Not Used
33	Reserved	Not Used
34	Reserved	Not Used
36	Reserved	Not Used
36	Reserved	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	Client Request Count	This value is incremented each time a request message is issued.
243	Client Response Count	This value is incremented each time a response message is received.
244	Client Error Sent Count	This value is incremented each time an error is sent from the client.
245	Client Error Received Count	This value is incremented each time an error is received from a remote unit.
246	Client Cfg Error Word	This word contains a bit map that defines configuration errors in the configuration file for the client.
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.

5.5 Modbus Protocol Specification

The following pages give additional reference information regarding the Modbus protocol commands supported by the MVI56E-MNETR.

5.5.1 Commands Supported by the Module

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

The following table lists the functions supported by the module.

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

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

5.5.2 Read Coil Status (Function Code 01)

Query

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

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

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

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

Response

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

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

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

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

5.5.3 Read Input Status (Function Code 02)

Query

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

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

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

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

Response

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

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

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

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

Due to the quantity of input statuses requested, the last data field which is shown as 35 HEX = 0011 0101 (binary) contains the status of only 6 inputs (10213 to 102180) instead of 8 inputs. The two left-most bits are provided as zeros to fill the 8-bit format.

5.5.4 Read Holding Registers (Function Code 03)

Query

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

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

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

Response

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

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

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

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

5.5.5 Read Input Registers (Function Code 04)

Query

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

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

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

Response

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

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

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

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

5.5.6 Force Single Coil (Function Code 05)

Query

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

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

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

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

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

Response

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

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

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

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

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

5.5.7 Preset Single Register (Function Code 06)

Query

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

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

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

Response

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

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

5.5.8 Diagnostics (Function Code 08)

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

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

In general, issuing a diagnostic function to a remote device does not affect the running of the user program in the remote device. User logic, like discrete and registers, is not accessed by the diagnostics. Certain functions can optionally reset error counters in the remote device.

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

The following diagnostic functions are dedicated to serial line devices.

The normal response to the Return Query Data request is to loopback the same data. The function code and sub-function codes are also echoed.

Request

Function code	1 Byte	0x08
Sub-function	2 Bytes	
Data	N x 2 Bytes	

Response

Function code	1 Byte	0x08
Sub-function	2 Bytes	
Data	N x 2 Bytes	

Error

Error code	1 Byte	0x88
Exception code	1 Byte	01 or 03 or 04

Sub-function codes supported

Only Sub-function 00 is supported by the MVI56E-MNETR module.

00 Return Query Data

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

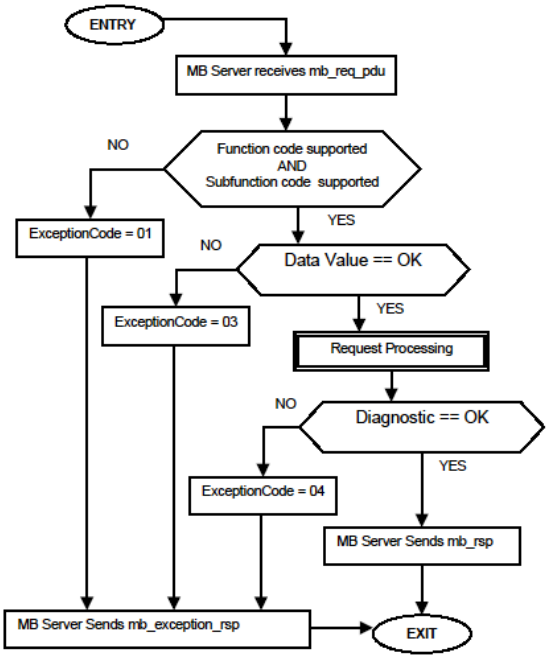
Sub-function	Data Field (Request)	Data Field (Response)
00 00	Any	Echo Request Data

Example and state diagram

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

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

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



5.5.9 Force Multiple Coils (Function Code 15)

Query

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

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

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

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

Response

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

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

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

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

5.5.10 Preset Multiple Registers (Function Code 16)

Query

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

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

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

Response

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

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

5.5.11 Modbus Exception Responses

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

- If the server device receives the request without a communication error, and can handle the query normally, it returns a normal response.
- If the server does not receive the request due to a communication error, no response is returned. The Client program will eventually process a timeout condition for the request.

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

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

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

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

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

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

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

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

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

Modbus Exception Codes

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

5.6 Using the Rung Import with Utility Add-On Instruction

5.6.1 Before you Begin

- Make sure that you have installed RSLogix 5000 version 16 (or later)
- Download the Add-On file *MVI56(E)MNETR_Optional_AddOn_Rung_<version #>.L5X* from the module's web page and copy it to a folder in your PC

5.6.2 Overview

The Rung Import with Utility Add-On Instruction contains optional logic for MVI56E-MNETR applications to perform the following tasks.

- Read/Write Ethernet Configuration

  Allows the processor to read or write the module IP address, netmask and gateway values.

Note: This is an optional feature. You can perform the same task through PCB (ProSoft Configuration Builder). Even if your PC is in a different network group you can still access the module through PCB by setting a temporary IP address.

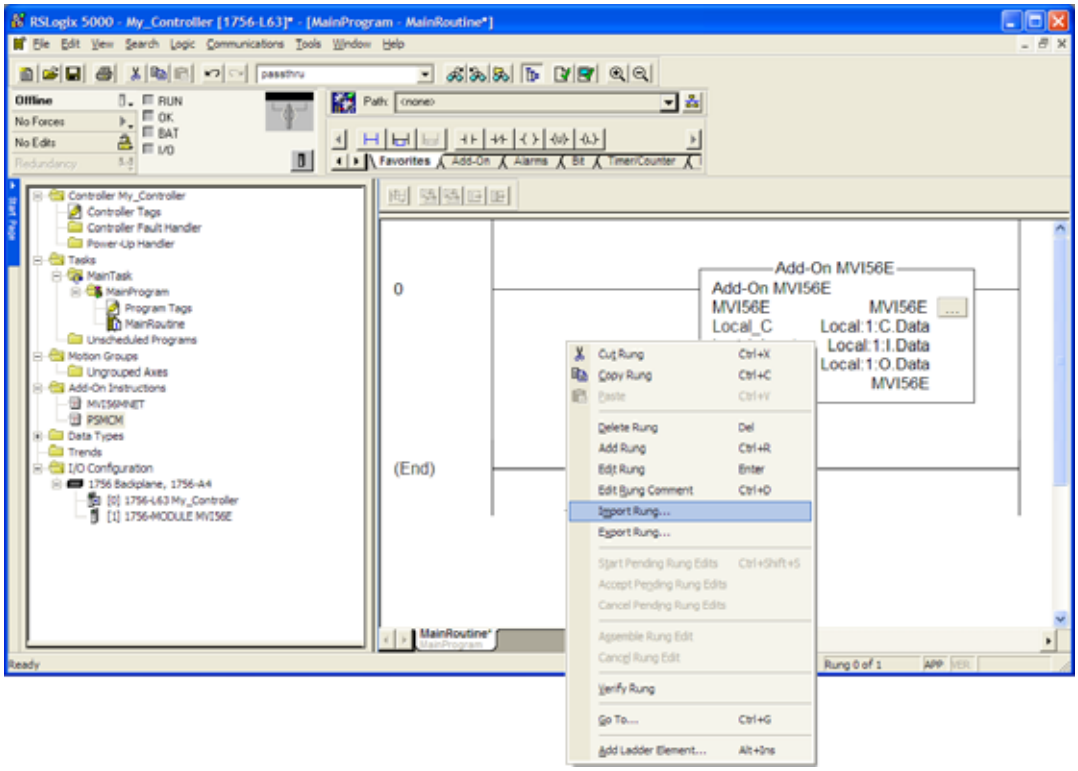
- Read/Write Module Clock Value

  Allows the processor to read and write the module clock settings. The module clock stores the last time that the Ethernet configuration was changed. The date and time of the last Ethernet configuration change is displayed in the scrolling LED during module power up.

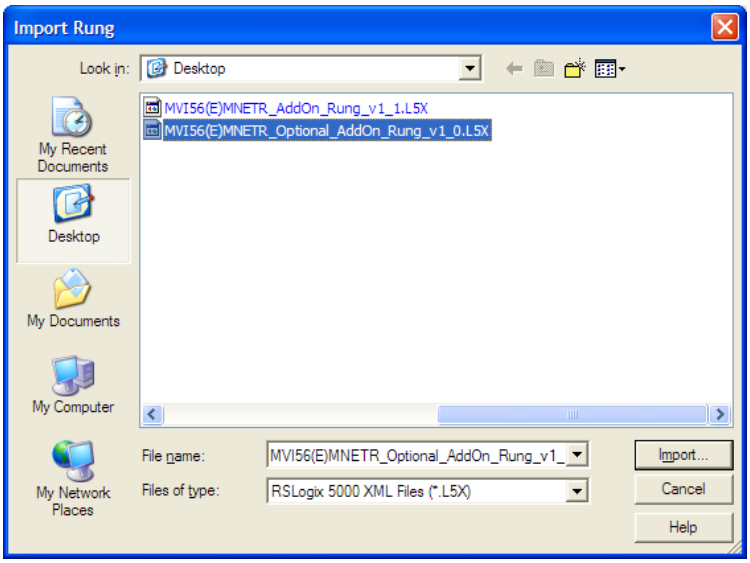
Important: The Rung Import with Utility Add-On Instruction only supports the two features listed above. You must use the sample ladder logic for all other features including backplane transfer of Modbus data.

5.6.3 Installing the Rung Import with Utility Add-On Instruction

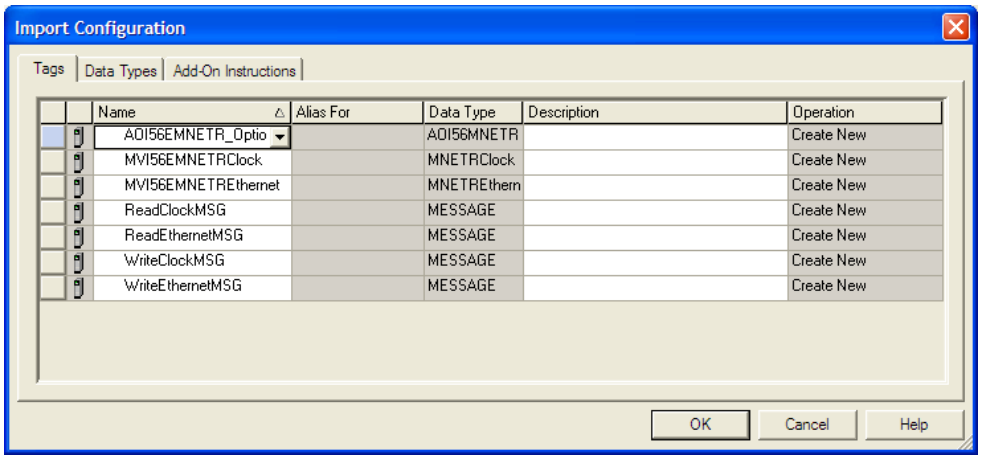
1 Right-click on an empty rung in the main routine of your existing ladder logic and choose **IMPORT RUNG…**



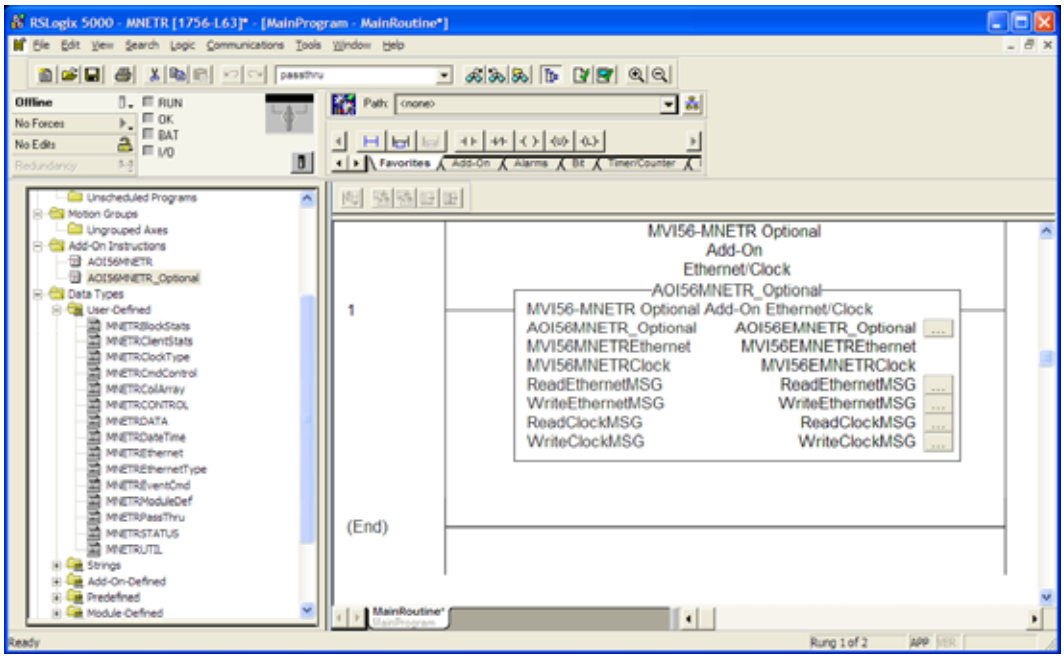
2 Navigate to the folder where you saved MVI56(E)MNETR_Optional_AddOn_Rung_<version #>.L5X and select the file.



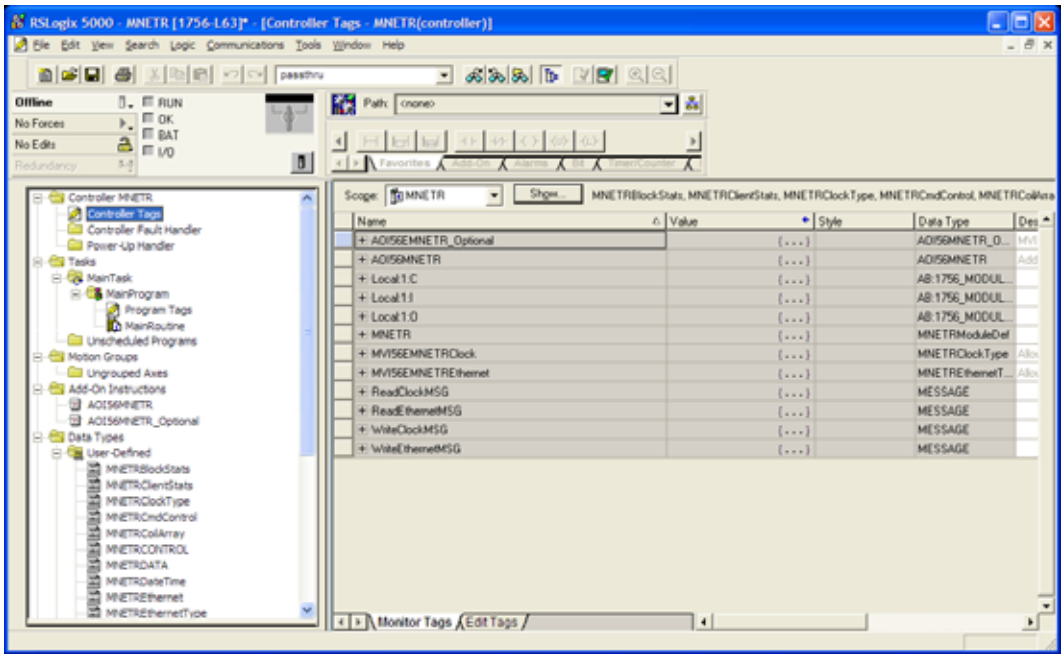
3 In the **IMPORT CONFIGURATION** window, click **OK.**



The Add-On Instruction will be now visible in the ladder logic. Observe that the procedure has also imported data types and controller tags associated to the Add-On Instruction.

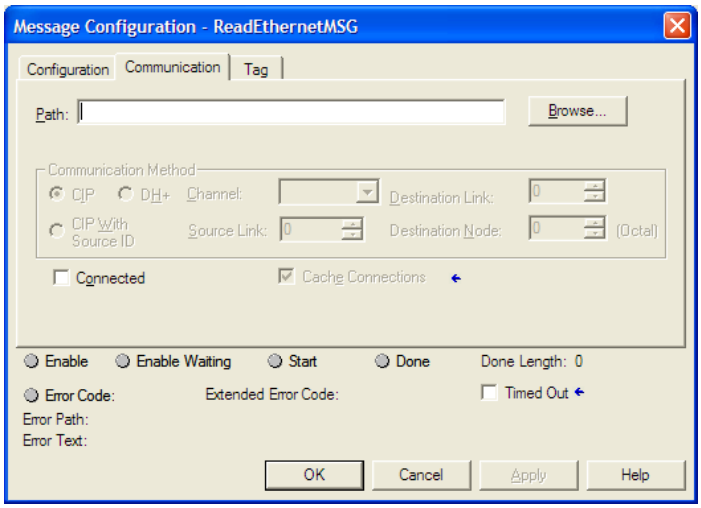


You will notice that new tags have been imported: four **MESSAGE** tags, **MVI56MNETRCLOCK** and **MVI56MNETRETHERNET** tags.

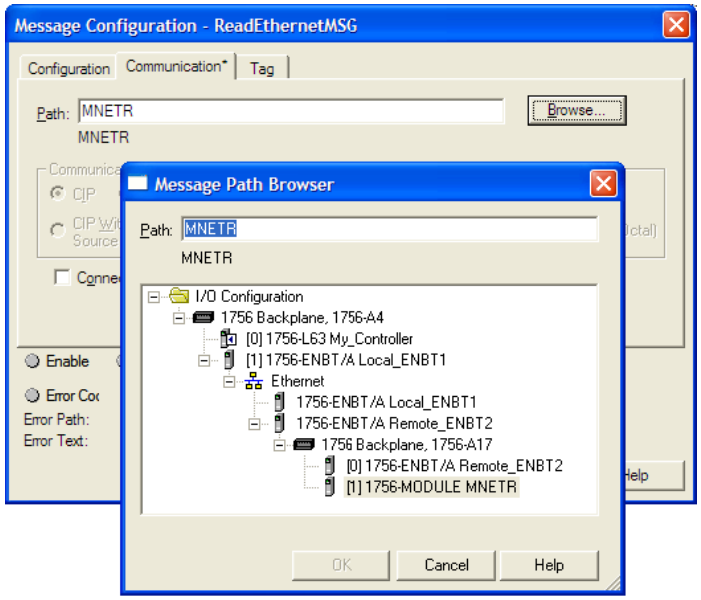


4 In the Add-On Instruction, click the **[...]** button next to each **MSG** tag to open the **MESSAGE CONFIGURATION TAG**.

5 Click the **COMMUNICATION** tab and click the **BROWSE** button as follows.



6 Select the module to configure the message path.



5.6.4 Reading the Ethernet Settings from the Module

Expand the **MVI56MNETRETHERNET** controller tag and move a value of 1 to **MVI56MNETRETHERNET.READ.**

MVI56EMNETREthernet	{...}
MVI56EMNETREthernet.Read	1
MVI56EMNETREthernet.Write	0
MVI56EMNETREthernet.Config	{...}
MVI56EMNETREthernet.Config.IP	{...}
MVI56EMNETREthernet.Config.IP[0]	0
MVI56EMNETREthernet.Config.IP[1]	0
MVI56EMNETREthernet.Config.IP[2]	0
MVI56EMNETREthernet.Config.IP[3]	0
MVI56EMNETREthernet.Config.Netmask	{...}
MVI56EMNETREthernet.Config.Netmask[0]	0
MVI56EMNETREthernet.Config.Netmask[1]	0
MVI56EMNETREthernet.Config.Netmask[2]	0
MVI56EMNETREthernet.Config.Netmask[3]	0
MVI56EMNETREthernet.Config.Gateway	{...}
MVI56EMNETREthernet.Config.Gateway[0]	0
MVI56EMNETREthernet.Config.Gateway[1]	0
MVI56EMNETREthernet.Config.Gateway[2]	0
MVI56EMNETREthernet.Config.Gateway[3]	0

The bit will be automatically reset and the current Ethernet settings will be copied to **MVI56MNETRETHERNET** controller tag as follows.

MVI56EMNETREthernet	{...}
MVI56EMNETREthernet.Read	0
MVI56EMNETREthernet.Write	0
MVI56EMNETREthernet.Config	{...}
MVI56EMNETREthernet.Config.IP	{...}
MVI56EMNETREthernet.Config.IP[0]	105
MVI56EMNETREthernet.Config.IP[1]	102
MVI56EMNETREthernet.Config.IP[2]	0
MVI56EMNETREthernet.Config.IP[3]	32
MVI56EMNETREthernet.Config.Netmask	{...}
MVI56EMNETREthernet.Config.Netmask[0]	255
MVI56EMNETREthernet.Config.Netmask[1]	255
MVI56EMNETREthernet.Config.Netmask[2]	255
MVI56EMNETREthernet.Config.Netmask[3]	0
MVI56EMNETREthernet.Config.Gateway	{...}
MVI56EMNETREthernet.Config.Gateway[0]	192
MVI56EMNETREthernet.Config.Gateway[1]	168
MVI56EMNETREthernet.Config.Gateway[2]	0
MVI56EMNETREthernet.Config.Gateway[3]	1

To check the status of the message, refer to the **READETHERNETMSG** tag.

ReadEthernetMSG	{...}
ReadEthernetMSG.Flags	16#0200
ReadEthernetMSG.EW	0
ReadEthernetMSG.ER	0
ReadEthernetMSG.DN	0
ReadEthernetMSG.ST	0
ReadEthernetMSG.EN	0
ReadEthernetMSG.TO	0
ReadEthernetMSG.EN_CC	1
ReadEthernetMSG.ERR	16#0000
ReadEthernetMSG.EXERR	16#0000_0000
ReadEthernetMSG.ERR_SRC	0
ReadEthernetMSG.DN_LEN	0
ReadEthernetMSG.REQ_LEN	0

5.6.5 Writing the Ethernet Settings to the Module

Expand the **MVI56EMNETRETHERNET** controller tag.

Set the new Ethernet configuration in **MVI56EMNETRETHERNET.CONFIG**

Move a value of 1 to **MVI56MNETRETHERNET.WRITE**

MVI56EMNETREthernet	{...}
MVI56EMNETREthernet.Read	0
MVI56EMNETREthernet.Write	1
MVI56EMNETREthernet.Config	{...}
MVI56EMNETREthernet.Config.IP	{...}
MVI56EMNETREthernet.Config.IP[0]	105
MVI56EMNETREthernet.Config.IP[1]	102
MVI56EMNETREthernet.Config.IP[2]	0
MVI56EMNETREthernet.Config.IP[3]	32
MVI56EMNETREthernet.Config.Netmask	{...}
MVI56EMNETREthernet.Config.Netmask[0]	255
MVI56EMNETREthernet.Config.Netmask[1]	255
MVI56EMNETREthernet.Config.Netmask[2]	255
MVI56EMNETREthernet.Config.Netmask[3]	0
MVI56EMNETREthernet.Config.Gateway	{...}
MVI56EMNETREthernet.Config.Gateway[0]	192
MVI56EMNETREthernet.Config.Gateway[1]	168
MVI56EMNETREthernet.Config.Gateway[2]	0
MVI56EMNETREthernet.Config.Gateway[3]	1

After the message is executed, the **MVI56MNETRETHERNET.WRITE** bit resets to 0.

MVI56EMNETREthernet	{...}
MVI56EMNETREthernet.Read	0
MVI56EMNETREthernet.Write	0
MVI56EMNETREthernet.Config	{...}
MVI56EMNETREthernet.Config.IP	{...}
MVI56EMNETREthernet.Config.IP[0]	105
MVI56EMNETREthernet.Config.IP[1]	102
MVI56EMNETREthernet.Config.IP[2]	0
MVI56EMNETREthernet.Config.IP[3]	32
MVI56EMNETREthernet.Config.Netmask	{...}
MVI56EMNETREthernet.Config.Netmask[0]	255
MVI56EMNETREthernet.Config.Netmask[1]	255
MVI56EMNETREthernet.Config.Netmask[2]	255
MVI56EMNETREthernet.Config.Netmask[3]	0
MVI56EMNETREthernet.Config.Gateway	{...}
MVI56EMNETREthernet.Config.Gateway[0]	192
MVI56EMNETREthernet.Config.Gateway[1]	168
MVI56EMNETREthernet.Config.Gateway[2]	0
MVI56EMNETREthernet.Config.Gateway[3]	1

To check the status of the message, refer to the **WRITEETHERNETMSG** tag.

WriteEthernetMSG	{...}
WriteEthernetMSG.Flags	16#0200
WriteEthernetMSG.EW	0
WriteEthernetMSG.ER	0
WriteEthernetMSG.DN	0
WriteEthernetMSG.ST	0
WriteEthernetMSG.EN	0
WriteEthernetMSG.TO	0
WriteEthernetMSG.EN_CC	1
WriteEthernetMSG.ERR	16#0000
WriteEthernetMSG.EXERR	16#0000_0000
WriteEthernetMSG.ERR_SRC	0
WriteEthernetMSG.DN_LEN	0
WriteEthernetMSG.REQ_LEN	24

5.6.6 Reading the Clock Value from the Module

Expand the **MVI56MNETRCLOCK** controller tag and move a value of 1 to **MVI56MNETRCLOCK.READ**

MVI56EMNETRClock	{...}
MVI56EMNETRClock.Read	1
MVI56EMNETRClock.Write	0
MVI56EMNETRClock.Config	{...}
MVI56EMNETRClock.Config.Year	0
MVI56EMNETRClock.Config.Month	0
MVI56EMNETRClock.Config.Day	0
MVI56EMNETRClock.Config.Hour	0
MVI56EMNETRClock.Config.Minute	0
MVI56EMNETRClock.Config.Seconds	0

The bit will be automatically reset and the current clock value will be copied to **MVI56MNETRCLOCK.CONFIG** controller tag as follows.

MVI56EMNETRClock	{...}
MVI56EMNETRClock.Read	0
MVI56EMNETRClock.Write	0
MVI56EMNETRClock.Config	{...}
MVI56EMNETRClock.Config.Year	2009
MVI56EMNETRClock.Config.Month	5
MVI56EMNETRClock.Config.Day	4
MVI56EMNETRClock.Config.Hour	15
MVI56EMNETRClock.Config.Minute	38
MVI56EMNETRClock.Config.Seconds	9

To check the status of the message, refer to the **READCLOCKMSG** tag.

ReadClockMSG	{...}
ReadClockMSG.Flags	16#0200
ReadClockMSG.EW	0
ReadClockMSG.ER	0
ReadClockMSG.DN	0
ReadClockMSG.ST	0
ReadClockMSG.EN	0
ReadClockMSG.TO	0
ReadClockMSG.EN_CC	1
ReadClockMSG.ERR	16#0000
ReadClockMSG.EXERR	16#0000_0000
ReadClockMSG.ERR_SRC	0
ReadClockMSG.DN_LEN	0
ReadClockMSG.REQ_LEN	0

5.6.7 Writing the Clock Value to the Module

Expand the **MVI56MNETRCLOCK** controller tag.

Set the new Clock value in **MVI56MNETRCLOCK.CONFIG**

Move a value of 1 to **MVI56MNETRCLOCK.WRITE**

MVI56EMNETRClock	{...}
MVI56EMNETRClock.Read	0
MVI56EMNETRClock.Write	1
MVI56EMNETRClock.Config	{...}
MVI56EMNETRClock.Config.Year	2009
MVI56EMNETRClock.Config.Month	5
MVI56EMNETRClock.Config.Day	4
MVI56EMNETRClock.Config.Hour	15
MVI56EMNETRClock.Config.Minute	38
MVI56EMNETRClock.Config.Seconds	9

The bit will be automatically reset to 0.

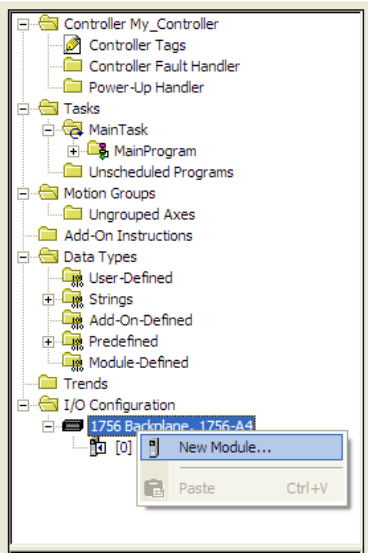
MVI56EMNETRClock	{...}
MVI56EMNETRClock.Read	0
MVI56EMNETRClock.Write	0
MVI56EMNETRClock.Config	{...}
MVI56EMNETRClock.Config.Year	2009
MVI56EMNETRClock.Config.Month	5
MVI56EMNETRClock.Config.Day	4
MVI56EMNETRClock.Config.Hour	15
MVI56EMNETRClock.Config.Minute	38
MVI56EMNETRClock.Config.Seconds	9

To check the status of the message, refer to the **WRITECLOCKMSG** tag.

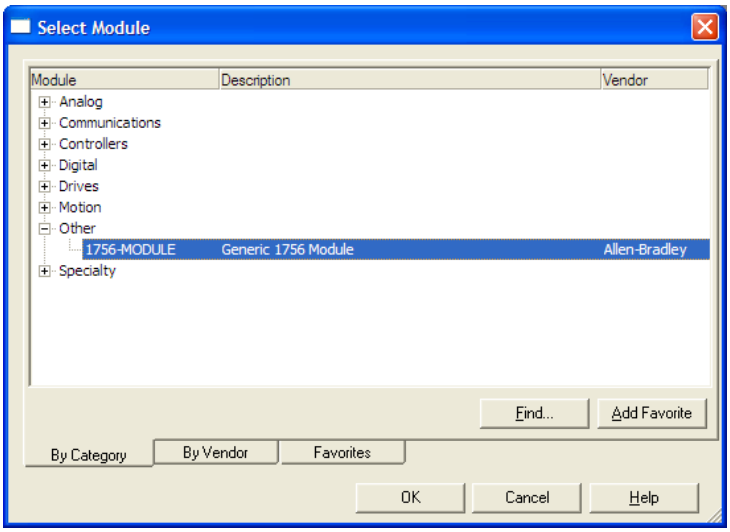
WriteClockMSG	{...}
WriteClockMSG.Flags	16#0200
WriteClockMSG.EW	0
WriteClockMSG.ER	0
WriteClockMSG.DN	0
WriteClockMSG.ST	0
WriteClockMSG.EN	0
WriteClockMSG.TO	0
WriteClockMSG.EN_CC	1
WriteClockMSG.ERR	16#0000
WriteClockMSG.EXERR	16#0000_0000
WriteClockMSG.ERR_SRC	0
WriteClockMSG.DN_LEN	0
WriteClockMSG.REQ_LEN	24

5.7 Adding the Module to an Existing Project

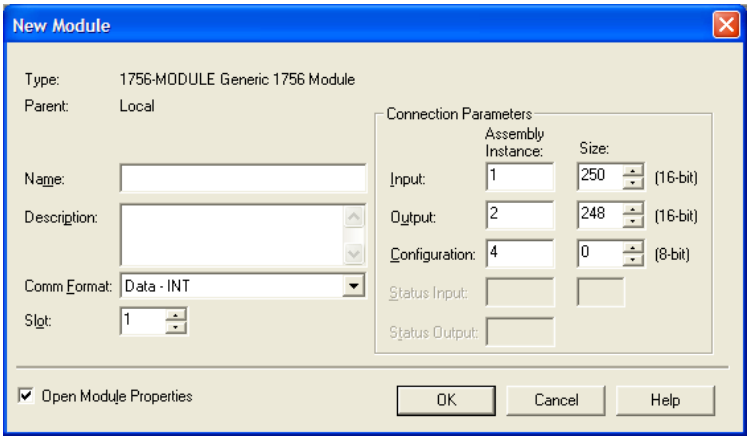
1 **Add the** MVI56E-MNETR **module to the project.** Select the the **I/O CONFIGURATION** folder in the **CONTROLLER ORGANIZATION** window, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE.**



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

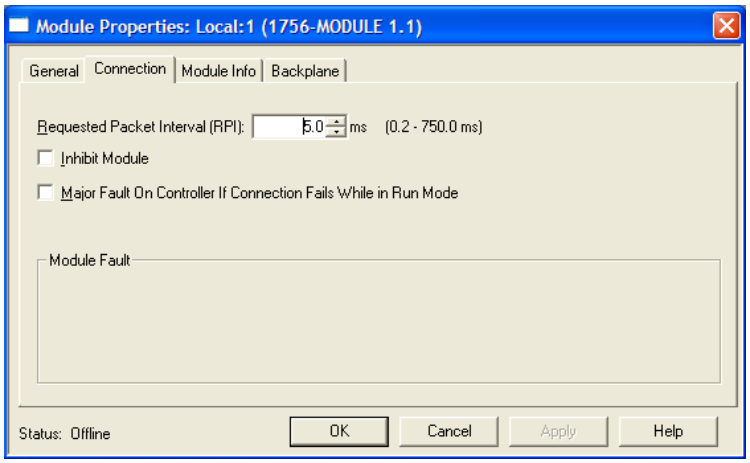


Select the **1756-MODULE** (Generic 1756 Module) from the list and click **OK.** This action opens the **NEW MODULE** dialog box.

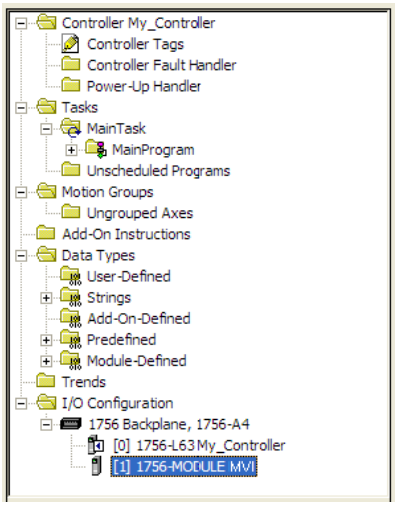


Enter the Name, Description and Slot options for your application. You must select the **COMM FORMAT AS DATA - INT** in the dialog box, otherwise the module will not communicate over the backplane of the ControlLogix rack. Click OK to continue.

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



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



4 Copy the Controller Tags from the sample program.
5 Copy the User Defined Data Types from the sample program.
6 Copy the Ladder Rungs from the sample program.
7 Save and Download (page 48) the new application to the controller and place the processor in run mode.

5.8 Using the Sample Program - RSLogix Version 15 and earlier

The sample program included with your MVI56E-MNETR Module contains predefined controller tags, configuration information, data types, and ladder logic that allow the module to communicate between the ControlLogix processor and a network of Modbus devices. For most applications, the sample program will work without modification.

6 Support, Service & Warranty

In This Chapter

- How to Contact Us: Technical Support 145
- Return Material Authorization (RMA) Policies and Conditions 146
- LIMITED WARRANTY 147

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

1 Product Version Number
2 System architecture
3 Network details

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

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

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

6.1 How to Contact Us: Technical Support

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

Asia Pacific

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

Europe (location in Toulouse, France)

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

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

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

Brasil (location in Sao Paulo)

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

6.2 Return Material Authorization (RMA) Policies and Conditions

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

6.2.1 All Product Returns:

a) In order to return a Product for repair, exchange or otherwise, the Customer must obtain a Returned Material Authorization (RMA) number from ProSoft and comply with ProSoft shipping instructions.
b) In the event that the Customer experiences a problem with the Product for any reason, Customer should contact ProSoft Technical Support at one of the telephone numbers listed above (page 145). A Technical Support Engineer will request that you perform several tests in an attempt to isolate the problem. If after completing these tests, the Product is found to be the source of the problem, we will issue an RMA.
c) All returned Products must be shipped freight prepaid, in the original shipping container or equivalent, to the location specified by ProSoft, and be accompanied by proof of purchase and receipt date. The RMA number is to be prominently marked on the outside of the shipping box. Customer agrees to insure the Product or assume the risk of loss or damage in transit. Products shipped to ProSoft using a shipment method other than that specified by ProSoft or shipped without an RMA number will be returned to the Customer, freight collect. Contact ProSoft Technical Support for further information.
d) A 10% restocking fee applies to all warranty credit returns whereby a Customer has an application change, ordered too many, does not need, and so on.

6.2.2 Procedures for Return of Units Under Warranty:

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

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

6.2.3 Procedures for Return of Units Out of Warranty:

a) Customer sends unit in for evaluation
b) If no defect is found, Customer will be charged the equivalent of $100 USD, plus freight charges, duties and taxes as applicable. A new purchase order will be required.

c) If unit is repaired, charge to Customer will be 30% of current list price (USD) plus freight charges, duties and taxes as applicable. A new purchase order will be required or authorization to use the purchase order submitted for evaluation fee.

The following is a list of non-repairable units:

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

6.3 LIMITED WARRANTY

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

6.3.1 What Is Covered By This Warranty

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

b) *Warranty On Services*: Materials and labor performed by ProSoft to repair a verified malfunction or defect are warranteed in the terms specified above for new Product, provided said warranty will be for the period remaining on the original new equipment warranty or, if the original warranty is no longer in effect, for a period of 90 days from the date of repair.

6.3.2 What Is Not Covered By This Warranty

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

6.3.3 Disclaimer Regarding High Risk Activities

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

6.3.4 Intellectual Property Indemnity

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

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

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

6.3.5 Disclaimer of all Other Warranties

The Warranty set forth in What Is Covered By This Warranty (page 147) are in lieu of all other warranties, express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

6.3.6 Limitation of Remedies **

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

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

6.3.7 Time Limit for Bringing Suit

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

6.3.8 No Other Warranties

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

6.3.9 Allocation of Risks

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

6.3.10 Controlling Law and Severability

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

Index

[

[MNET Client x Commands] • 57, 92
[MNET Client x] • 55, 92
[MNET Servers] • 63, 93
[Module] • 53, 90
[Static ARP Table] • 65, 90

0

00 Return Query Data • 127

A

About the MODBUS TCP/IP Protocol • 103
Adding Multiple Modules (Optional) • 40
Adding the Module to an Existing Project • 141
All Product Returns: • 146
Allocation of Risks • 150
ARP Timeout • 56

B

Backplane Data Transfer • 104
Backplane Status • 90
Battery Life Advisory • 3
Before you Begin • 132
Bit Input Offset • 64

C

CIPconnect • 21
Clearing a Fault Condition • 99
Client Command Errors • 116
Client Command List • 116
Client Driver • 115
Cold Boot • 111
Command Control • 110
Command Control Blocks • 108
Command Entry Formats • 58
Command Error Delay • 56
Command Error Pointer • 55
Command List • 92
Command List Entry Errors • 117
Command List Overview • 57
Command Status • 92
Commands Supported by the Module • 121
Comment • 62
Config • 90, 92, 93
Configuration Error Word • 98
Configuring Module Parameters • 52
Configuring the Data Analyzer • 93
Configuring the MVI56E-MNETR Module • 49
Connect to the Module's Web Page • 23
Connect your PC to the ControlLogix Processor • 39
Connect your PC to the Module • 15
Controlling Law and Severability • 150
Create a new RSLogix 5000 project • 26
Create the Module - Local Rack • 26, 31
Create the Module - Remote Rack • 29
Create the Remote Network • 27, 31

D

Data Analyzer • 93
Data Analyzer Tips • 95
Data Flow between MVI56E-MNETR Module and ControlLogix Processor • 113
Diagnostics (Function Code 08) • 126
Diagnostics and Troubleshooting • 7, 85
Disclaimer of all Other Warranties • 149
Disclaimer Regarding High Risk Activities • 148
Download the Project to the Module • 67
Download the Sample Program to the Processor • 48, 143
Duplex/Speed Code • 54

E

Enable • 59
Error/Status Pointer • 53, 55
Ethernet Cable Configuration • 118
Ethernet Cable Specifications • 118
Ethernet Configuration • 66
Ethernet LED Indicators • 99
Ethernet Performance - MVI56E Modules • 118
Event Command • 108
Example 1
    Local Rack Application • 71
Example 2
    Remote Rack Application • 74
Example and state diagram • 127

F

Failure Flag Count • 54
Float Flag • 55, 63
Float Offset • 56, 64
Float Start • 56, 64
Force Multiple Coils (Function Code 15) • 128
Force Single Coil (Function Code 05) • 125
Formatted • 112
Functional Overview • 7, 103
Functional Specifications • 102

G

General Specifications • 101
Guide to the MVI56E-MNETR User Manual • 7

H

Hardware MAC Address • 65
Hardware Specifications • 102
Holding Register Offset • 64
How to Contact Us
    Technical Support • 145, 146

I

Import Add-On Instruction • 34
Important Safety Information - MVI56E Modules • 2
Initialize Output Data • 54, 107
Install ProSoft Configuration Builder • 14
Install the Configuration Tools • 14
Install the Module in the Rack • 12
Installing the Rung Import with Utility Add-On Instruction • 133
Intellectual Property Indemnity • 149
Internal Address • 59
IP Address • 65

L

Ladder Logic • 79
LED Status Indicators • 7, 97
Limitation of Remedies ** • 150
LIMITED WARRANTY • 147

M

MB Address in Device • 62
Minimum Command Delay • 55
MNET Client Specific Errors • 117
MNETRBlockStats • 81
MNETRClientStats • 81
MNETRCmdControl • 82
MNETRCoilArray • 84
MNETRCONTROL • 81
MNETRDATA • 80
MNETREventCmd • 82
MNETRModuleDef • 79
MNETRPassThru • 83
MNETRSTATUS • 80
MNETRUTIL • 84
Modbus Exception Codes • 131
Modbus Exception Responses • 129
Modbus Function • 61
Modbus Message Data • 84
Modbus Protocol Specification • 57, 121
Module Communication Error Codes • 117
Module Power Up • 103
Monitoring Backplane Information • 90
Monitoring Database Information • 91
Monitoring MNET Client Information • 92
Monitoring MNET Server Information • 93
Monitoring Module Information • 89
MVI56E-MNETR Client Status • 106
MVI56E-MNETR Status Data Definition • 85, 90, 119

N

NIC Status • 90
No Other Warranties • 150
Node IP Address • 61
Normal Data Transfer • 106

O

Output Offset • 64
Overview • 132

P

Package Contents • 11
Pass-Through Control Blocks • 84, 111
Pass-Through Mode • 54
Pinouts • 118
Poll Interval • 60
Preset Multiple Registers (Function Code 16) • 129
Preset Single Register (Function Code 06) • 126
Printing a Configuration File • 53
Procedures for Return of Units Out of Warranty: • 146
Procedures for Return of Units Under Warranty: • 146
Product Specifications • 7, 101
ProSoft Technology® Product Documentation • 3

R

Read Block • 106
Read Coil Status (Function Code 01) • 121
Read Holding Registers (Function Code 03) • 123
Read Input Registers (Function Code 04) • 124
Read Input Status (Function Code 02) • 122
Read Register Count • 53
Read Register Start • 53
Reading Status Data from the Module • 85
Reading the Clock Value from the Module • 139
Reading the Ethernet Settings from the Module • 137
Reference • 7, 101
Reg Count • 60
Response Timeout • 55
Retry Count • 55
Return Material Authorization (RMA) Policies and Conditions • 146

S

Server Driver • 114
Service Port • 61
Set Module Parameters • 52
Set Temporary IP Address • 15
Set Up the Project • 49
Setting Jumpers • 12
Slave Address • 61
Standard Modbus Protocol Errors • 117
Start Here • 7, 9
Starting the Data Analyzer • 94
Static ARP • 90
Status • 92, 93
Status Read Data Block • 106
Stopping the Data Analyzer • 95
Sub-function codes supported • 127
Support, Service & Warranty • 7, 145
Swap Code • 60
System Requirements • 10

T

The Diagnostics Menu • 86
Time Limit for Bringing Suit • 150
Troubleshooting • 100

U

Unformatted • 111
Upload the Add-On Instruction from the Module • 25, 35
Using CIPconnect® to Connect to the Module • 68, 88, 118
Using ProSoft Configuration Builder Software • 49
Using the Diagnostics Menu in ProSoft Configuration Builder • 86
Using the Rung Import with Utility Add-On Instruction • 132
Using the Sample Program - RSLogix Version 15 and earlier • 144

V

Version • 89

W

Warm Boot • 110
What Is Covered By This Warranty • 147, 149
What Is Not Covered By This Warranty • 148
What's Different? • 10
Word Input Offset • 65
Write Block • 107
Write Register Count • 53
Write Register Start • 53
Writing the Clock Value to the Module • 140
Writing the Ethernet Settings to the Module • 138

Y

Your Feedback Please • 3