



Where Automation Connects.



inRax[®] **MVI56E-MNETCR**

ControlLogix Platform

Modbus TCP/IP Multi Client
Enhanced Communications Module
for Remote Chassis

June 14, 2011

USER MANUAL

Your Feedback Please

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

How to Contact Us

ProSoft Technology

5201 Truxtun Ave., 3rd Floor

Bakersfield, CA 93309

+1 (661) 716-5100

+1 (661) 716-5101 (Fax)

www.prosoft-technology.com

support@prosoft-technology.com

Copyright © 2011 ProSoft Technology, Inc., all rights reserved.

MVI56E-MNETCR User Manual

June 14, 2011

ProSoft Technology[®], ProLinx[®], inRAx[®], ProTalk[®], and RadioLinx[®] are Registered Trademarks of ProSoft Technology, Inc. All other brand or product names are or may be trademarks of, and are used to identify products and services of, their respective owners.

ProSoft Technology[®] Product Documentation

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

Battery Life Advisory

Note: Modules manufactured after April 1st, 2011 do not contain a battery. For modules manufactured before that date the following applies:

The module uses a rechargeable Lithium Vanadium Pentoxide battery to back up 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 24 hours.

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.

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

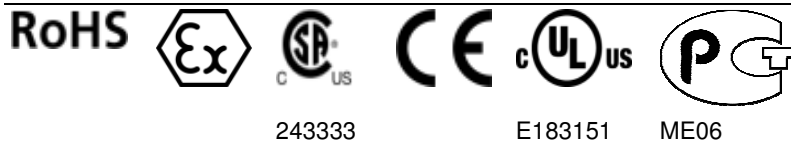
- A** Warning - Explosion Hazard - When in hazardous locations, turn off power before replacing or wiring modules.
- B** Warning - Explosion Hazard - Do not disconnect equipment unless power has been switched off or the area is known to be non-hazardous.
- C** These products are intended to be mounted 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 Vdc; 3 mA @ 24 Vdc
- Operating Temperature: 0 °C to 60 °C (32 °F to 140 °F)
- Storage Temperature: -40 °C to 85 °C (-40 °F to 185 °F)
- Shock: 30 g operational; 50 g non-operational; Vibration: 5 g from 10 Hz to 150 Hz
- Relative Humidity 5% to 95% (without condensation)
- All phase conductor sizes must be at least 1.3 mm (squared) and all earth ground conductors must be at least 4mm (squared).

Markings

Agency	Applicable Standards
RoHS	
ATEX	EN60079-0 July 2006 EN60079-15 October 2005
CSA	IEC61010
CE	EMC-EN61326-1:2006 EN61000-6-4:2007
CSA CB Safety	CA/10533/CSA IEC 61010-1 Ed. 2 CB 243333-2056722 (2090408)
cULus	
GOST-R	Test 2.4



Class I Division 2 Groups A, B, C, and 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.

Contents

Your Feedback Please	2
How to Contact Us	2
ProSoft Technology® Product Documentation	2
Battery Life Advisory	3
Important Safety Information - MVI56E Modules	3
1 Start Here	9
1.1 What's New?	10
1.2 System Requirements	11
1.3 Package Contents	12
1.4 Setting Jumpers	13
1.5 Installing the Module in the Rack	14
1.6 Installing the Configuration Tools	16
1.6.1 Installing ProSoft Configuration Builder	16
1.7 Connecting Your PC to the Module	17
1.8 Setting Temporary IP Address	18
1.8.1 Using CIPconnect to Connect to the Module	22
1.9 Connecting to the Module's Web Page	24
1.10 Uploading the Add-On Instruction from the Module	25
1.11 Creating a New RSLogix 5000 Project	26
1.11.1 Create the Remote Network	27
1.11.2 Import Add-On Instruction	33
1.11.3 Connecting Your PC to the ControlLogix Processor	43
1.11.4 Downloading the Sample Program to the Processor	43
2 Configuring the MVI56E-MNETCR Module	45
2.1 Using ProSoft Configuration Builder Software	46
2.1.1 Setting Up the Project	47
2.1.2 Renaming PCB Objects	49
2.1.3 Module	50
2.1.4 MNET Client x	52
2.1.5 MNET Client x Commands	54
2.1.6 Static ARP Table	60
2.1.7 Ethernet Configuration	61
2.2 Downloading the Project to the Module	62
2.3 Using CIPconnect® to Connect to the Module	63
2.3.1 Example 1: Local Rack Application	65
2.3.2 Example 2: Remote Rack Application	68
3 Ladder Logic	71
3.1 Module Data Object (MNETCRMODULEDEF)	72
3.1.1 User Data Object (MNETCRDATA)	73
3.1.2 Command Control Data Object (MNETCCONTROL)	74
3.1.3 Status Object (MNETCRSTATUS)	75
3.1.4 Backplane Control Object (MNETCRUTIL)	77

4	Diagnostics and Troubleshooting	79
4.1	LED Status Indicators	80
4.1.1	Scrolling LED Status Indicators	80
4.1.2	Ethernet LED Indicators	81
4.1.3	Non-Scrolling LED Status Indicators	81
4.1.4	Troubleshooting	82
4.1.5	Clearing a Fault Condition	83
4.2	Using the Diagnostics Menu in ProSoft Configuration Builder	84
4.2.1	The Diagnostics Menu	87
4.2.2	Monitoring Module Information	87
4.2.3	Monitoring Backplane Information	88
4.2.4	Monitoring MNET Client Information.....	89
4.2.5	Monitoring Database Information.....	90
4.3	Reading Status Data from the Module	91
5	Reference	93
5.1	Product Specifications	94
5.1.1	General Specifications	94
5.1.2	Functional Specifications	95
5.1.3	Hardware Specifications	96
5.2	Functional Overview	97
5.2.1	About the MODBUS/TCP Protocol	97
5.2.2	Backplane Data Transfer	98
5.2.3	Special Function Blocks.....	102
5.3	Data Flow between MVI56E-MNETCR Module, Processor, and Network.....	111
5.3.1	Client Driver	112
5.3.2	Client Command List	113
5.3.3	Client Command Errors	113
5.4	Ethernet Cable Specifications.....	116
5.4.1	Ethernet Cable Configuration	116
5.4.2	Ethernet Performance.....	117
5.5	Modbus Protocol Specification	118
5.5.1	Read Coil Status (Function Code 01)	118
5.5.2	Read Input Status (Function Code 02)	120
5.5.3	Read Holding Registers (Function Code 03).....	121
5.5.4	Read Input Registers (Function Code 04)	122
5.5.5	Force Single Coil (Function Code 05)	123
5.5.6	Preset Single Register (Function Code 06)	124
5.5.7	Diagnostics (Function Code 08)	125
5.5.8	Force Multiple Coils (Function Code 15)	127
5.5.9	Preset Multiple Registers (Function Code 16).....	128
5.5.10	Modbus Exception Responses	129
5.6	Using the Optional Add-On Instruction Rung Import	132
5.6.1	Before You Begin.....	132
5.6.2	Overview	132
5.6.3	Installing the Rung Import with Utility Add-On Instruction	133
5.6.4	Reading the Ethernet Settings from the Module.....	138
5.6.5	Writing the Ethernet Settings to the Module	139
5.6.6	Reading the Clock Value from the Module	140
5.6.7	Writing the Clock Value to the Module	141
5.7	Adding the Module to an Existing Project.....	142

5.8	Using the Sample Program - RSLogix 5000 Version 15 and earlier.....	145
-----	---	-----

6	Support, Service & Warranty	147
----------	--	------------

	Contacting Technical Support.....	147
6.1	Return Material Authorization (RMA) Policies and Conditions.....	149
6.1.1	Returning Any Product	149
6.1.2	Returning Units Under Warranty	150
6.1.3	Returning Units Out of Warranty	150
6.2	LIMITED WARRANTY.....	151
6.2.1	What Is Covered By This Warranty.....	151
6.2.2	What Is Not Covered By This Warranty	152
6.2.3	Disclaimer Regarding High Risk Activities	152
6.2.4	Intellectual Property Indemnity.....	153
6.2.5	Disclaimer of all Other Warranties	153
6.2.6	Limitation of Remedies **	154
6.2.7	Time Limit for Bringing Suit	154
6.2.8	No Other Warranties	154
6.2.9	Allocation of Risks.....	154
6.2.10	Controlling Law and Severability.....	155

Index	157
--------------	------------

1 Start Here

In This Chapter

❖ What's New?	10
❖ System Requirements	11
❖ Package Contents	12
❖ Setting Jumpers	13
❖ Installing the Module in the Rack.....	14
❖ Installing the Configuration Tools	16
❖ Connecting Your PC to the Module	17
❖ Setting Temporary IP Address	18
❖ Connecting to the Module's Web Page	24
❖ Uploading the Add-On Instruction from the Module.....	25
❖ Creating 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-MNETCR module's application port(s)

1.1 What's New?

MVI56E products are **backward compatible** with existing MVI56 products, ladder logic, and module configuration files already in use. Easily swap and upgrade products while benefiting from an array of new features designed to improve interoperability and enhance ease of use.

- **Web Server:** The built-in web server and web page allow access to manuals and other tools previously provided only on a product CD-ROM or from the ProSoft Technology® web site.
- **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 MVI56E modules on the network and to temporarily change an IP address to connect with a 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 Module:** 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 standard English messages for status and alarm data, and for processor and network communication status.

1.2 System Requirements

The MVI56E-MNETCR 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-MNETCR module. The module requires 800 mA 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-MNETCR module, and are all required for installation and configuration.

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

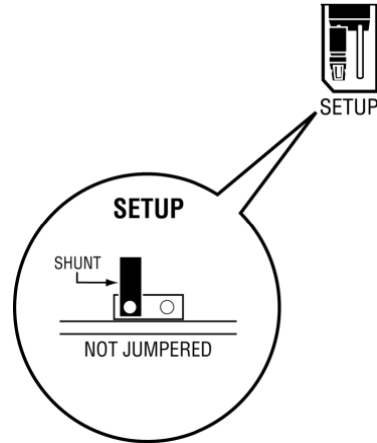
Qty.	Part Name	Part Number	Part Description
1	MVI56E-MNETCR Module	MVI56E-MNETCR	Modbus TCP/IP Multi Client Enhanced Communications Module for Remote Chassis
1	Cable	RL-CBL025	5-foot Ethernet Straight-Through Cable (Gray)
1	ProSoft Solutions CD	CD-013	Contains configuration tools for the MVI56E-MNETCR module
1	Insert		MVI56E-MNETCR 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-MNETCR jumper configuration.



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

1.5 Installing the Module in the Rack

If you have not already installed and configured your ControlLogix processor and power supply, please do so before installing the MVI56E-MNETCR module. Refer to your Rockwell Automation product documentation for installation instructions.

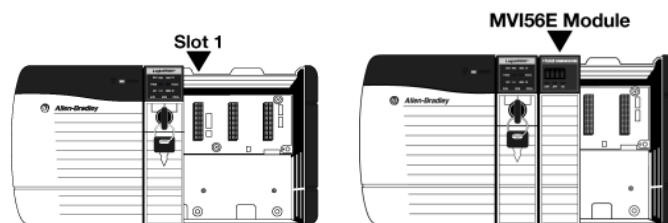
Warning: You must follow all safety instructions when installing this or any other electronic devices. Failure to follow safety procedures could result in damage to hardware or data, or even serious injury or death to personnel. Refer to the documentation for each device you plan to connect to verify 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-MNETCR 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. This can cause unintended machine motion or loss of process control. Electrical arcs may also cause an explosion when they happen 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.

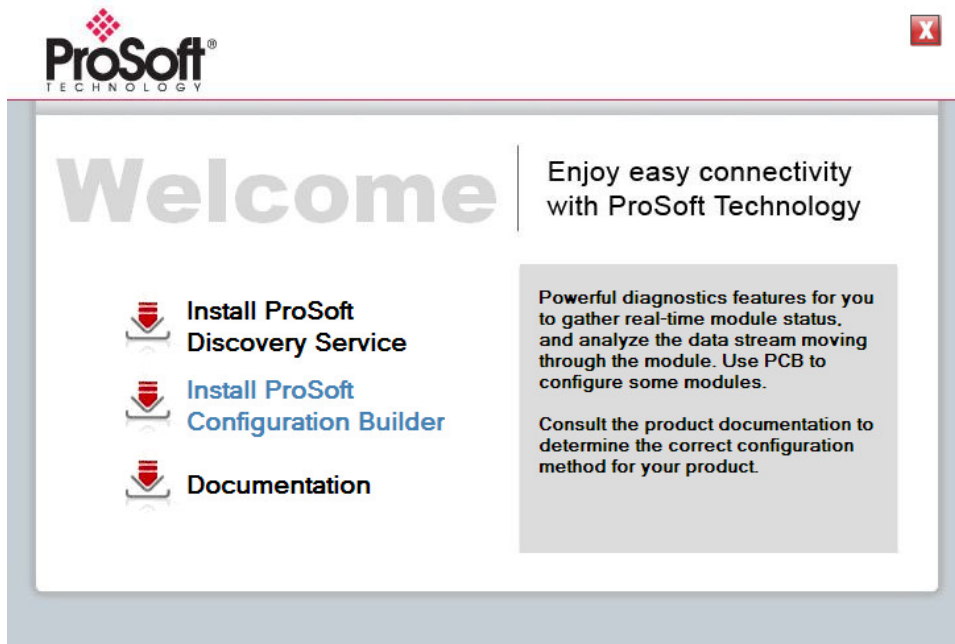
Note: If you insert the module improperly, the system may stop working or may behave unpredictably.

1.6 Installing the Configuration Tools

1.6.1 Installing ProSoft Configuration Builder

To install ProSoft Configuration Builder from the CD-ROM

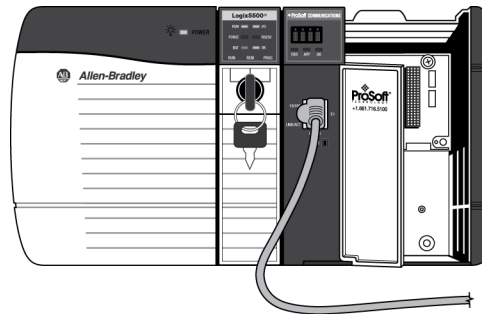
- 1 Insert the *ProSoft Solutions* CD-ROM into the CD drive of your PC. Wait for the startup screen to appear.



- 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 Connecting Your PC to the Module

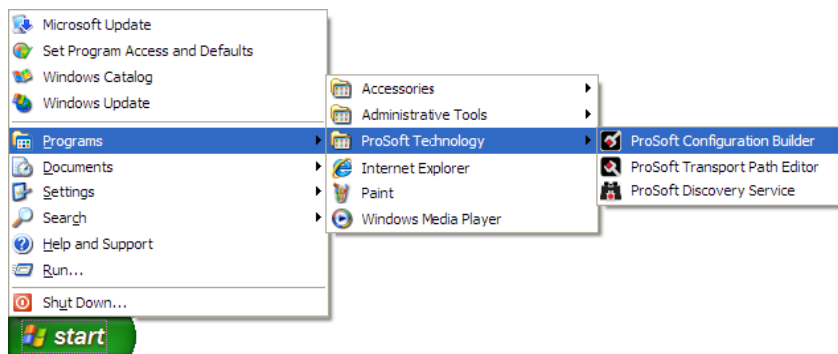
With the module securely mounted, connect one end of the Ethernet cable to the *Config (E1)* Port, and the other end to an Ethernet hub or switch accessible from the same network as your PC. You can also connect directly from the Ethernet Port on your PC to the *Config (E1)* Port on the module by using an Ethernet crossover cable (not included).



1.8 Setting Temporary IP Address

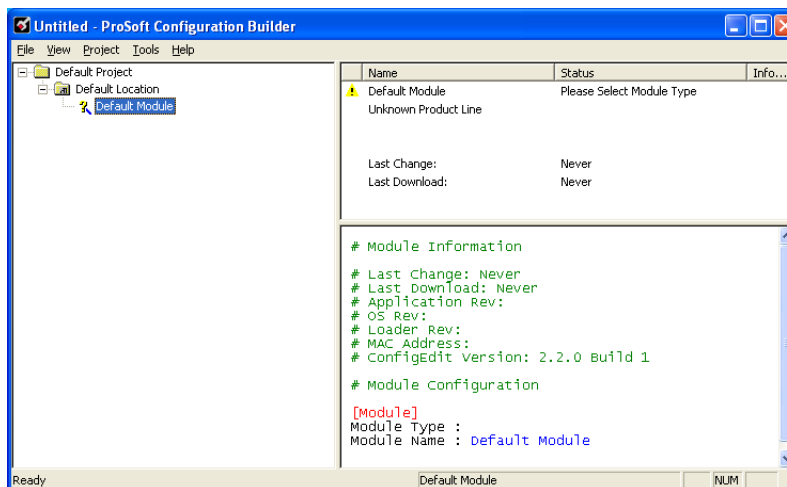
Important: *ProSoft Configuration Builder* locates MVI56E-MNETCR modules through UDP broadcast messages. These messages may be blocked by routers or layer 3 switches. In that case, *ProSoft Discovery Service* will be unable to locate the modules. To use *ProSoft Configuration Builder*, arrange the Ethernet 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 routing of the UDP broadcast messages.

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

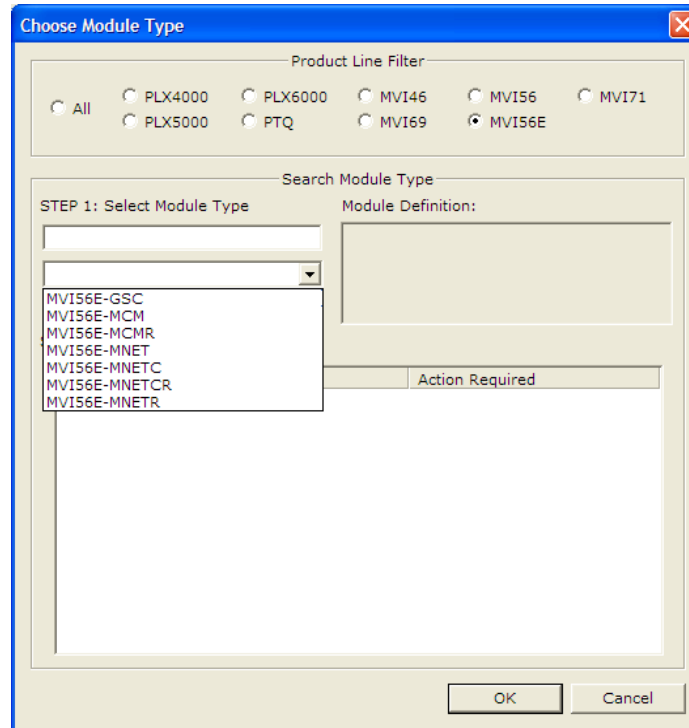


- 2 Click to start **PROSOFT CONFIGURATION BUILDER**.

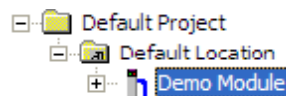
If you have used other Windows configuration tools before, you will find the screen layout familiar. *PCB's* window consists of a tree view on the left, and an information pane and a configuration pane on the right side of the window. When you first start *PCB*, the tree view consists of folders for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



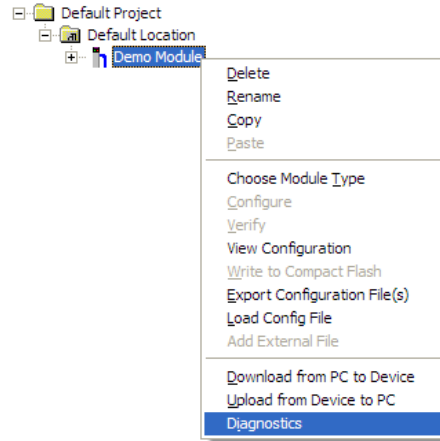
- 3 Use the mouse to select **DEFAULT MODULE** in the tree view, and then click the right mouse button to open a shortcut menu.
- 4 On the shortcut menu, select **CHOOSE MODULE TYPE**. This action opens the *Choose Module Type* dialog box.



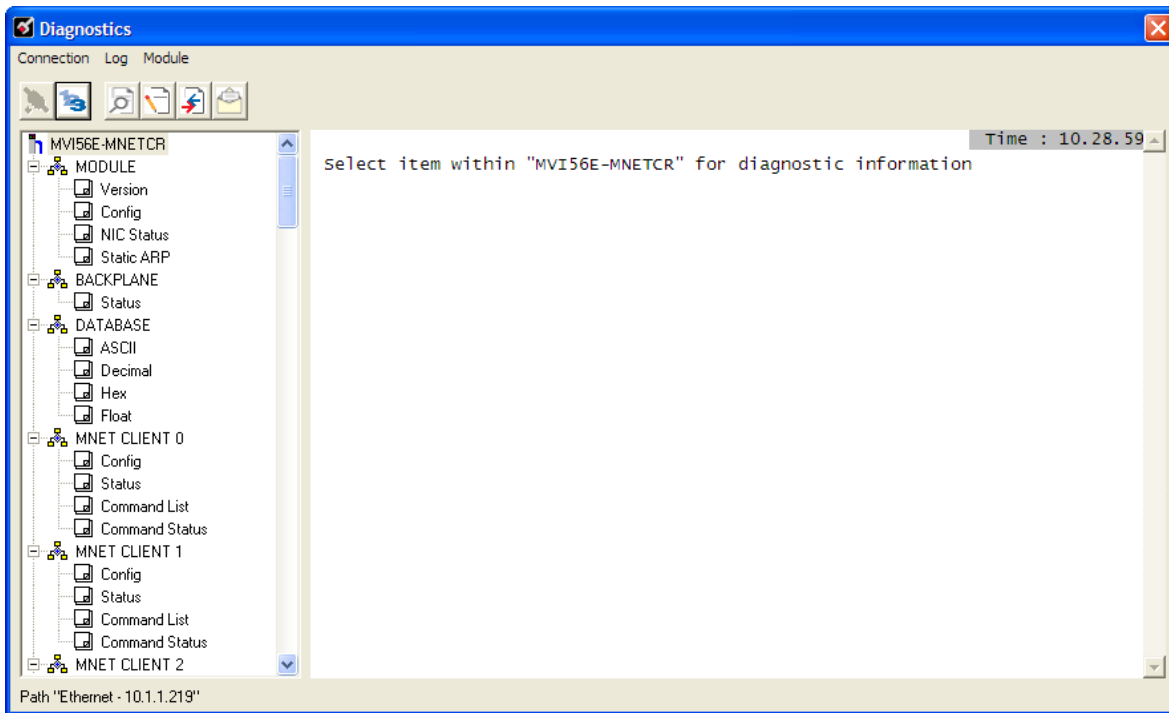
- 5 In the *Product Line Filter* area of the dialog box, select **MVI56E**. In the **SELECT MODULE TYPE** dropdown list, select **MVI56E-MNETCR**, and then click **OK** to save your settings and return to the *ProSoft Configuration Builder* window.
- 6 Right-click the module icon.



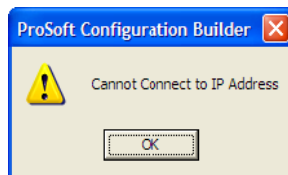
7 On the shortcut menu, choose **DIAGNOSTICS**.



This action opens the *Diagnostics* dialog box.



If there is no response from the module,

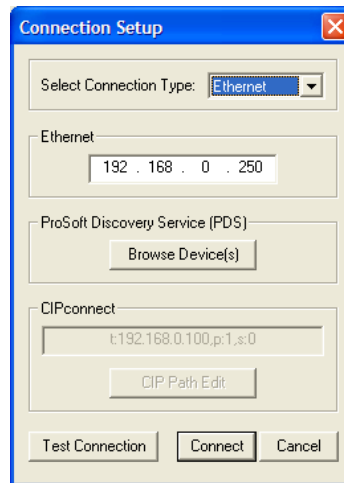


- 1 Click the **SET UP CONNECTION** button to browse for the module's IP address.

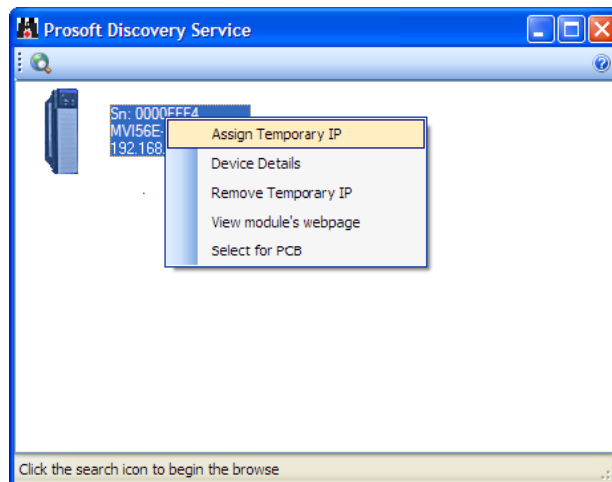


Click to set up connection

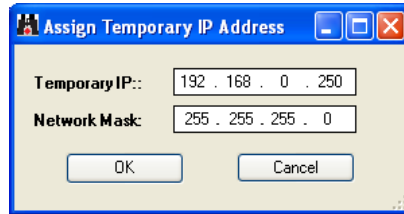
- 2 On the *Connection Setup* dialog box, click the **TEST CONNECTION** button to verify if the module is accessible with the current settings.



- 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**.

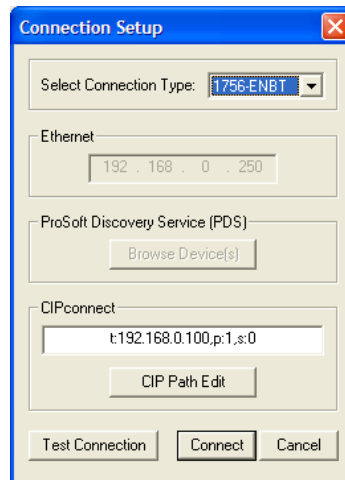
1.8.1 Using CIPconnect to Connect to the Module

You can use CIPconnect® to connect a PC to the MVI56E-MNETCR module over Ethernet using Rockwell Automation's 1756-ENBT EtherNet/IP® module. This allows you to configure the MVI56E-MNETCR 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:

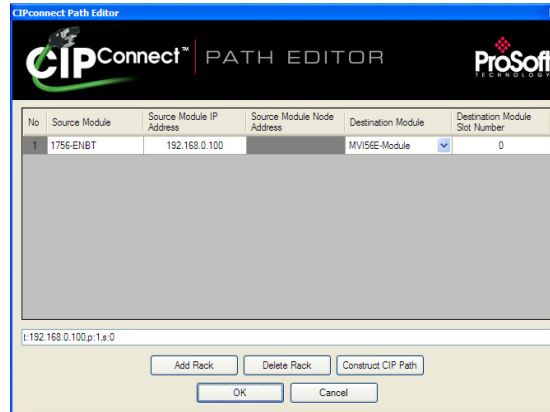
- The IP addresses and slot numbers of any 1756-ENBT modules in the path
- The ControlNet node numbers and slot numbers of any 1756-CNBx ControlNet Bridge modules in the path
- The slot number of the MVI56E-MNETCR in the destination ControlLogix chassis (the last ENBT/CNBx and chassis in the path).

To use CIPconnect, follow these steps.

- 1 In the *Select 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-MNETCR 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-MNETCR 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-MNETCR is available in a remote rack (accessible through ControlNet or Ethernet/IP), include all racks (by using the **ADD RACK** button).

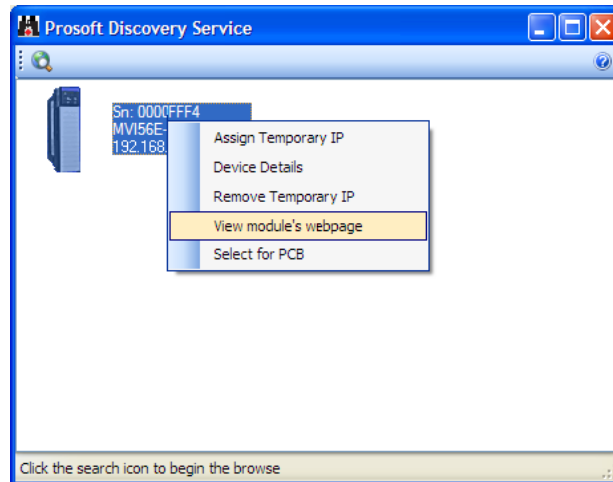
Parameter	Description
Source Module	Source module type. This field is automatically selected depending on the destination module of the last rack (1756-CNB or 1756-ENBT).
Source Module IP Address	IP address of the source module (only applicable for 1756-ENBT)
Source Module Node Address	Node address of 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-MNETCR 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 Connecting 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.

All manuals and configuration tools are available from the module's web page

The screenshot shows the web page for the 'Modbus TCP/IP Client for Remote ControlLogix MVI56E-MNETCR'. The page features the ProSoft Technology logo at the top left. A navigation menu on the left includes sections for 'DOWNLOADS' (Safety Info, Setup Guide, User Manuals, Add-On Software, Datasheet), 'FUNCTIONS' (Firmware Upgrade, Set Date & Time), and 'Technical Support' and 'Homepage'. The main content area displays the following device details:

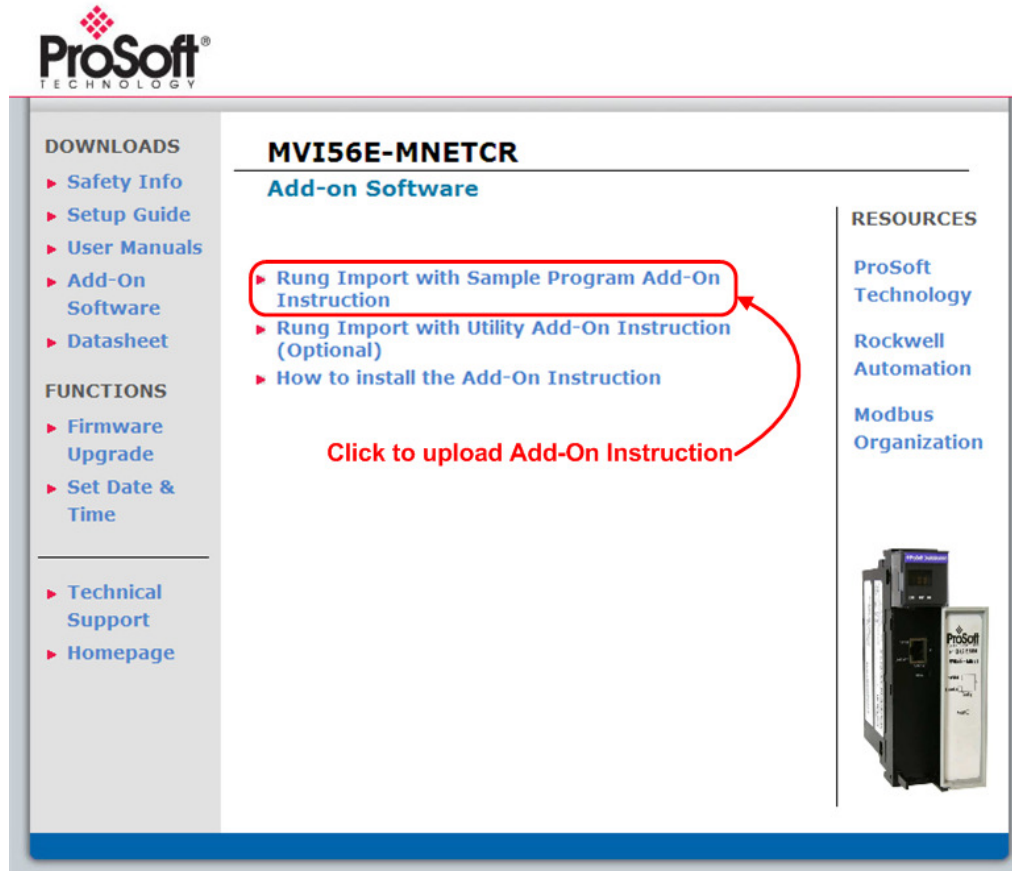
Module Name	MVI56E-MNETCR
Ethernet Address (MAC)	00:0D:8D:00:3A:98
IP Address	10.1.3.195
Product Revision	2.04.006 2.6.25 #20
Firmware Version Date	11/12/09 - 01
Serial Number	000003E9
Status	Running
Uptime	01:06:45

On the right side, there is a 'RESOURCES' section with links to 'ProSoft Technology', 'Rockwell Automation', and 'Modbus Organization'. A small image of the module is shown at the bottom right of the page.

Important: The temporary IP address is only valid until the next time the module is initialized. Please refer to Setting Temporary IP Address (page 18) in the MVI56E-MNETCR User Manual for information on how to set the module's permanent IP address.

1.10 Uploading the Add-On Instruction from the Module

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



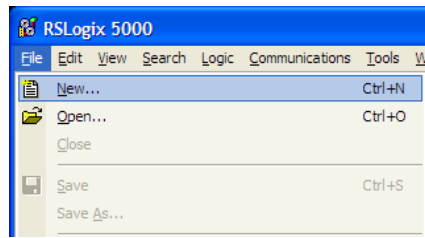
Two Add-On Instructions are provided:

- The *Rung Import with Sample Program Add-On Instruction*:
MVI56EMNETCR_AddOn_Rung_v1_3.L5X
Includes the user-defined data types, data objects and ladder logic required to configure the MVI56E-MNETCR module.
- The *Rung Import with Utility Add-On Instruction (Optional)*:
MVI56EMNETCR_Optional_Rung_v1_0.L5X

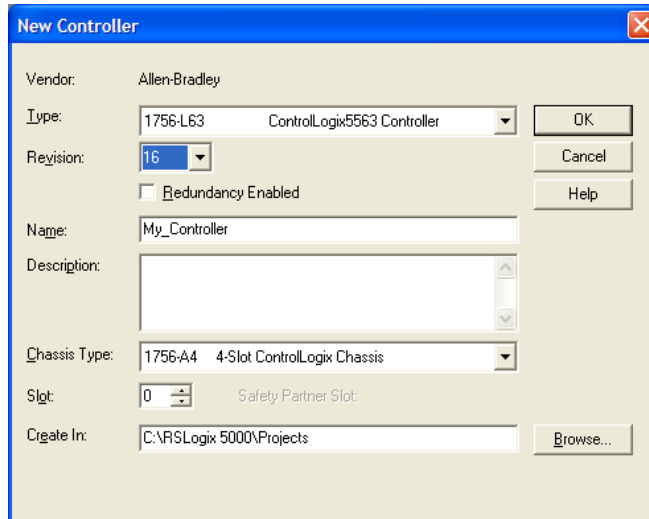
If your processor uses an earlier version of RSLogix 5000, see Using the Sample Program (page 145).

1.11 Creating a New RSLogix 5000 Project

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



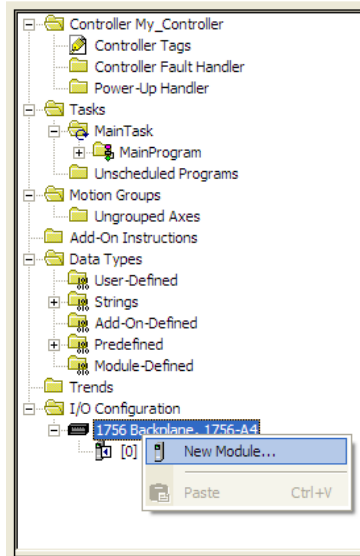
- 2 Select your ControlLogix controller model.
- 3 Select **REVISION 16**.
- 4 Enter a name for your controller, such as *My_Controller*.
- 5 Select your ControlLogix chassis type.
- 6 Select **SLOT 0** for the controller.



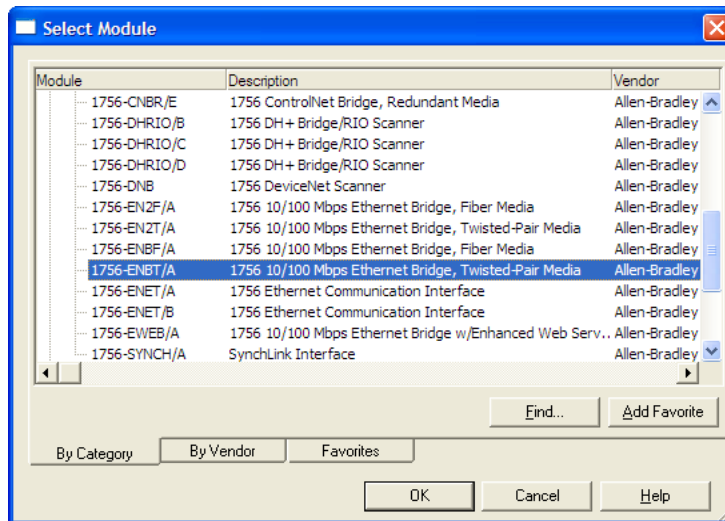
Note: If you are installing the MVI56E-MNETCR module in a remote rack, follow these next few steps. If you are installing the module in a local rack, follow the steps in Create the Module - Local Rack (page 31).

1.11.1 Create the Remote Network

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

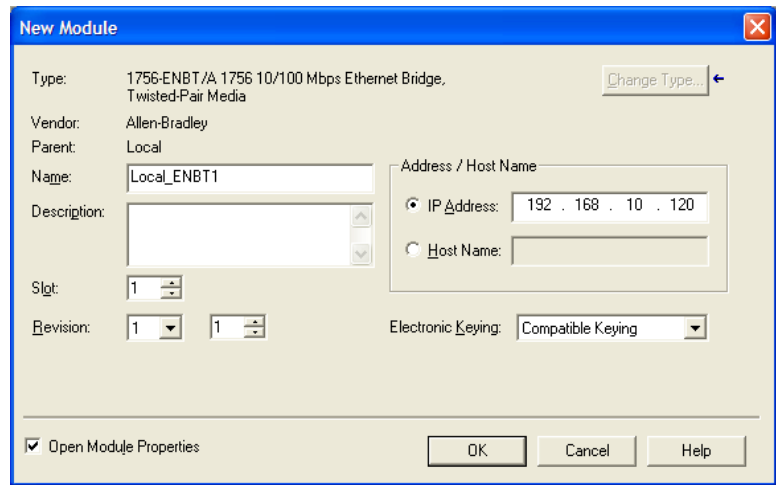


- 2 Expand the **COMMUNICATIONS** module selections and then select the Ethernet Bridge module that matches your hardware. This example uses a 1756-ENBT/A module.

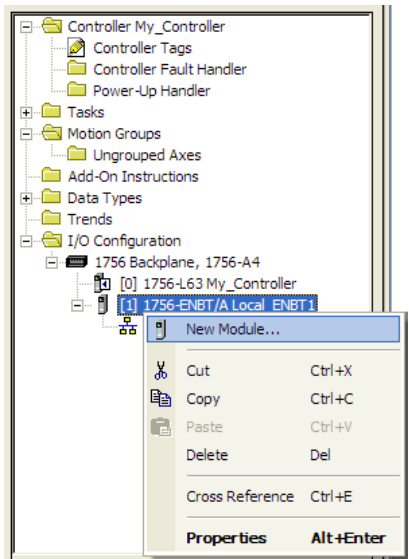


Note: If you are prompted to "Select Major Revision", choose the lower of the available revision numbers.

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



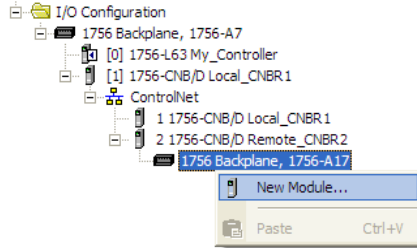
- 4 Click **OK**.
- 5 Next, select the **1756-ENBT** module that you just created in the Controller Organization pane and click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.



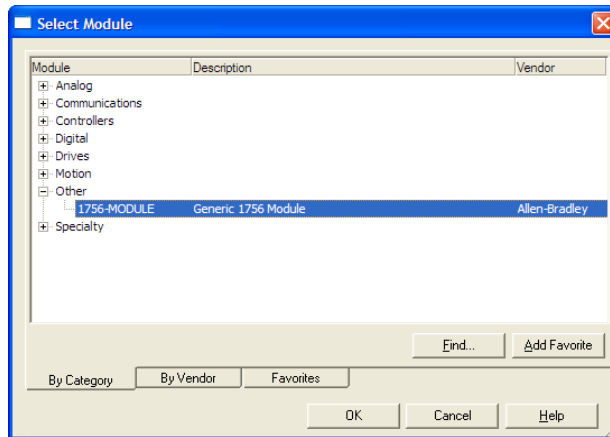
- 6 Repeat steps 2 and 3 to add the second EtherNet/IP module to the remote rack.

Create the Module - Remote Rack

- 1 In the **CONTROLLER ORGANIZATION** window, select the remote **1756 BACKPLANE** node, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.



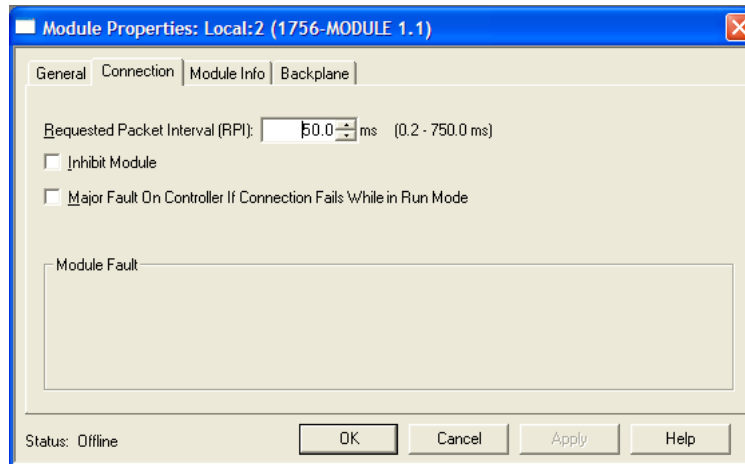
- 2 This action opens the **SELECT MODULE** dialog box. Expand the **OTHER** node, and then select **1756-MODULE (GENERIC 1756 MODULE)**



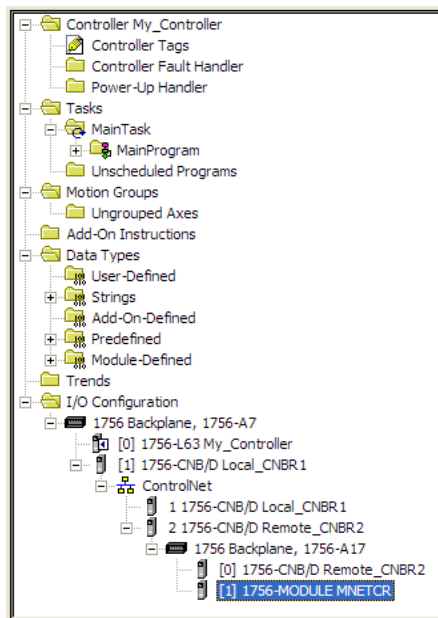
- 3 Set the Module Properties values as follows:

Parameter	Value
Name	Enter a module identification string. The recommended value is MNETCR.
Description	Enter a description for the module. Example: Modbus TCP/IP Multi Client Enhanced Communications Module for Remote Chassis.
Comm Format	Select DATA-INT (Very Important)
Slot	Enter the slot number in the rack where the MVI56E-MNETCR module will be installed.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

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



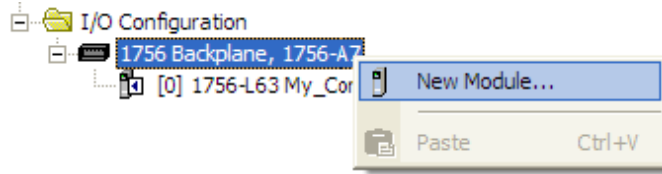
The **MVI56E-MNETCR** module is now visible in the **I/O CONFIGURATION** pane.



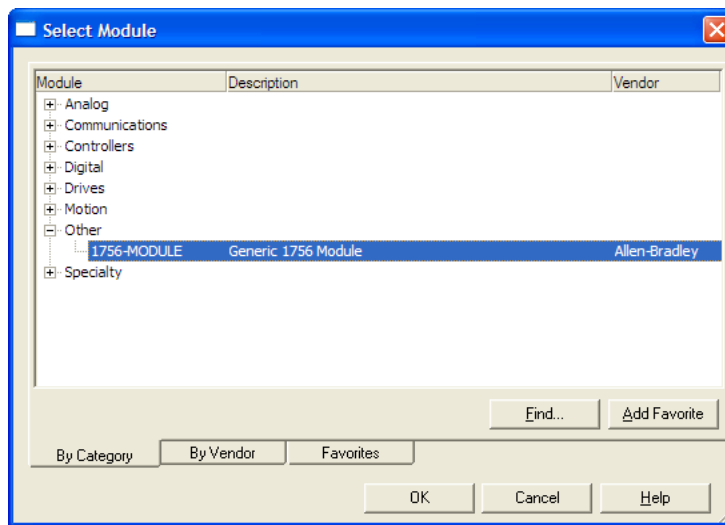
Note: If you are installing the MVI56E-MNETCR module in a local rack, follow these next few steps. If you are installing the module in a remote rack, follow the steps in Create the Module - Remote Rack (page 27).

Create the Module - Local Rack

- 1 In the **CONTROLLER ORGANIZATION** window, expand the I/O configuration node.
- 2 Select the **1756 BACKPLANE** node, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE...**



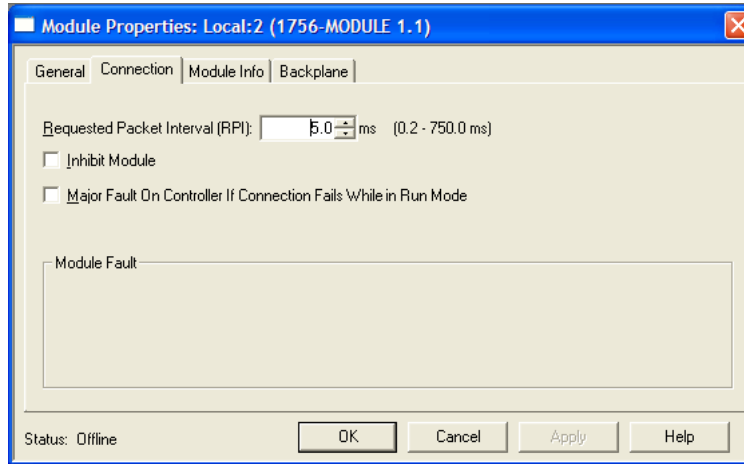
- 3 This action opens the **SELECT MODULE** dialog box.



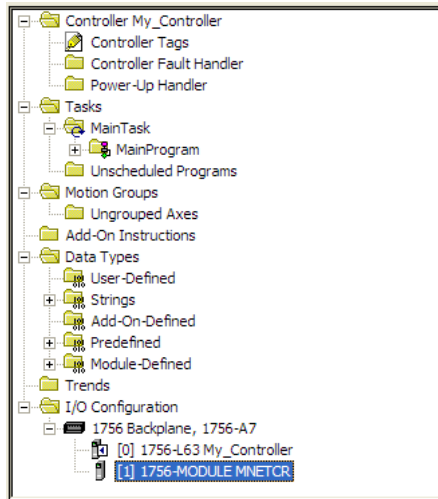
- 4 Select the **1756-MODULE (GENERIC 1756 MODULE)** from the list and click **OK**.
- 5 Set the Module Properties values as follows:

Parameter	Value
Name	Enter a module identification string. The recommended value is MNETCR.
Description	Enter a description for the module. Example: Modbus TCP/IP Multi Client Enhanced Communications Module for Remote Chassis.
Comm Format	Select DATA-INT (Very Important)
Slot	Enter the slot number in the rack where the MVI56E-MNETCR module is to be installed.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

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

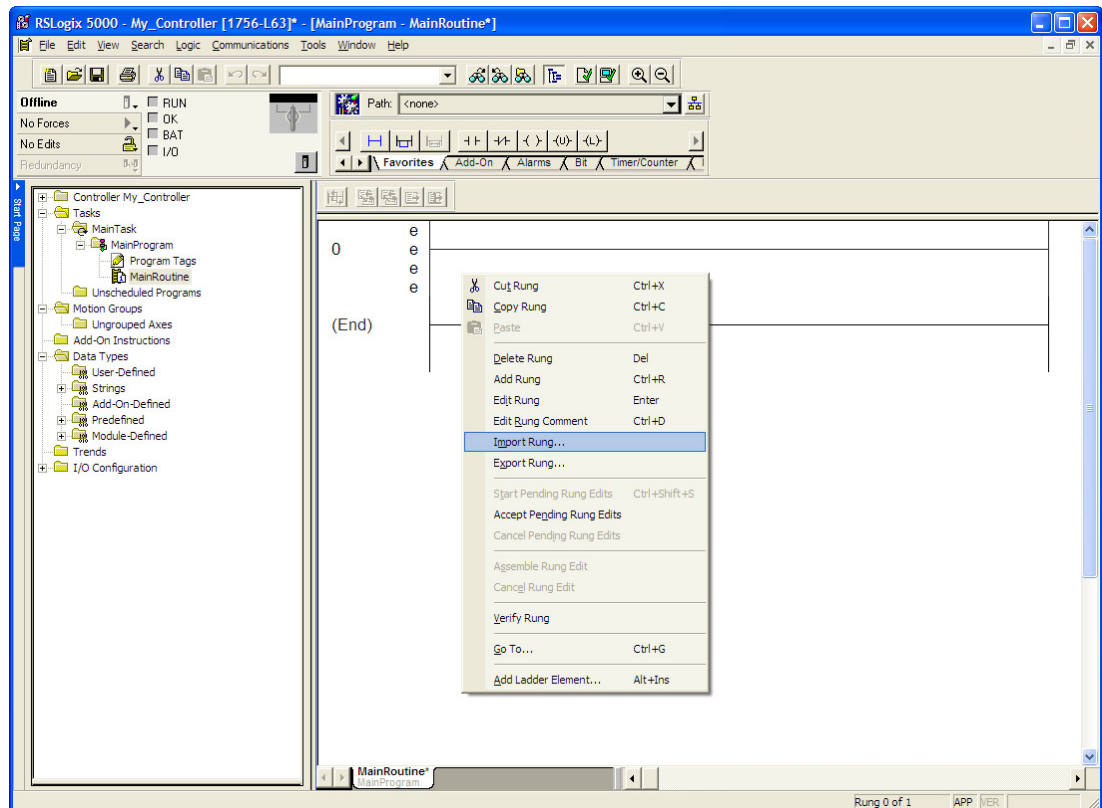


The **MVI56E-MNETCR** module is now visible in the **I/O CONFIGURATION** pane.

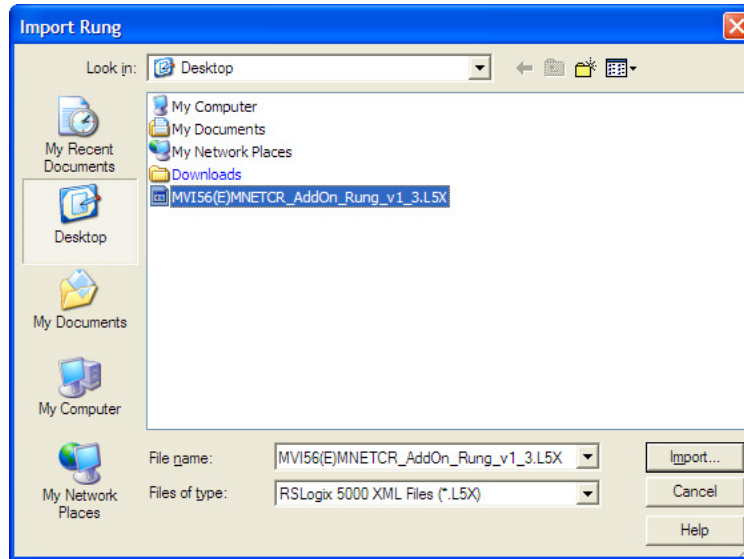


1.11.2 Import Add-On Instruction

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

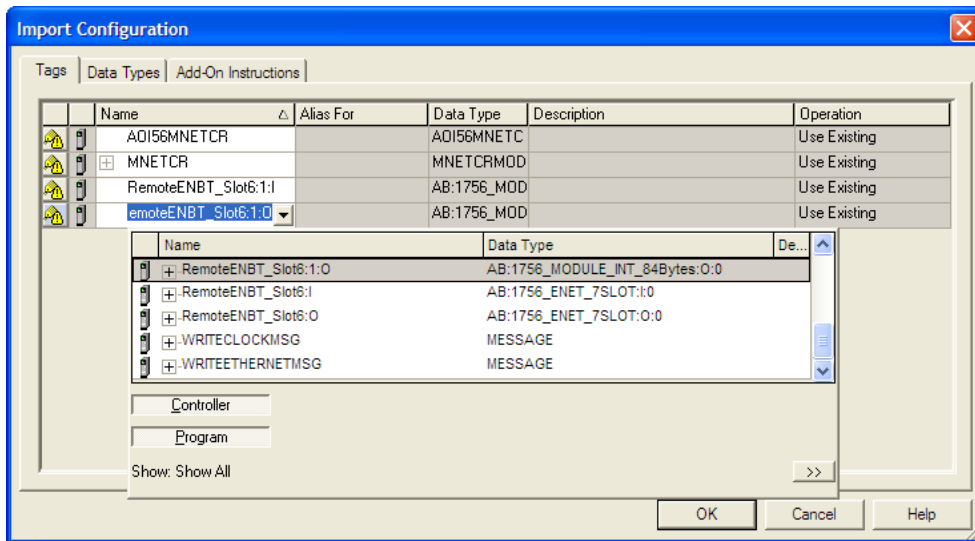
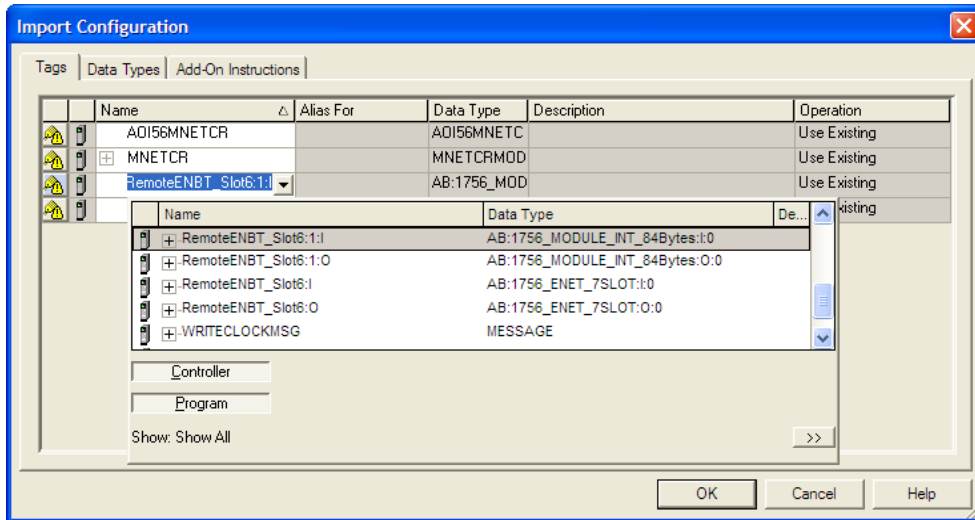


- 4 Navigate to the location on your PC where you saved (page 25) the Add-On Instruction (for example, "My Documents" or "Desktop"). Select the **MVI56EMNETCR_ADDON_RUNG_v1_3.L5X** file

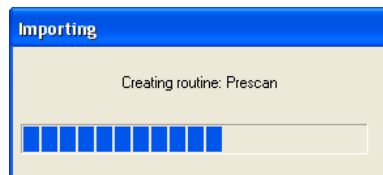


This action opens the **IMPORT CONFIGURATION** dialog box, showing the controller tags that will be created.

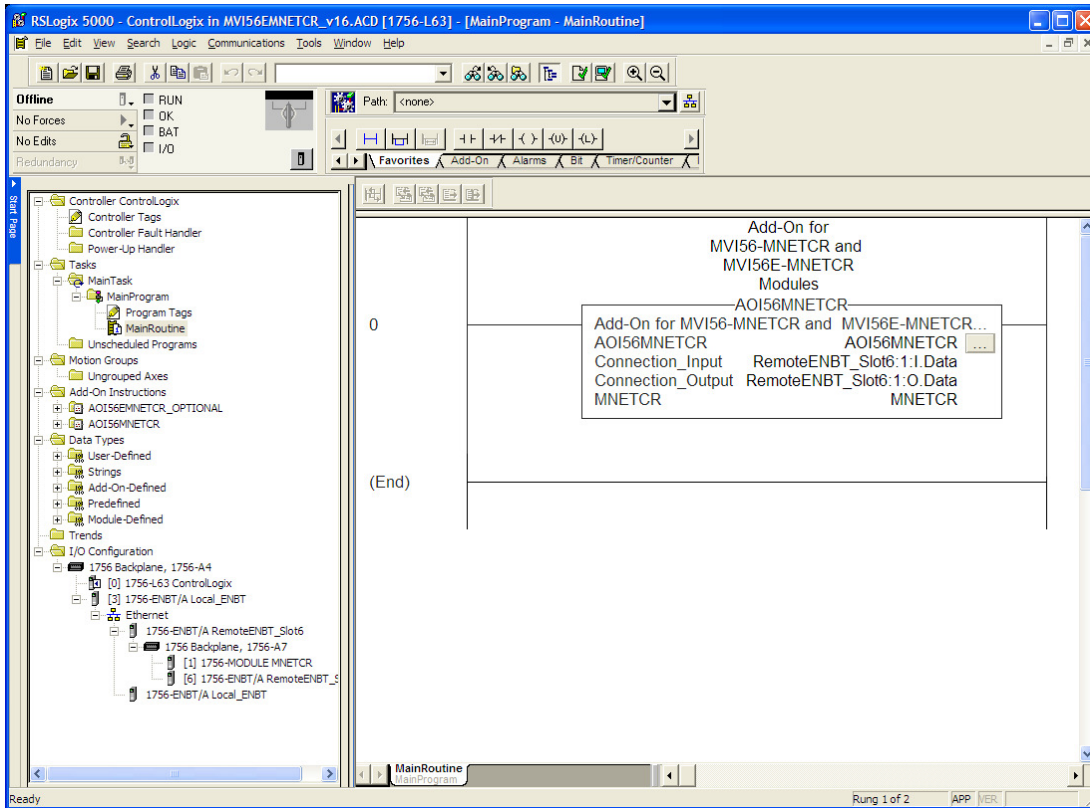
- If you are installing the module in a Remote Rack, open the dropdown menus for the Input and Output tags, and select the MNETCR module in the remote rack.



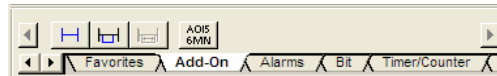
- 5 Click **OK** to confirm the import. RSLogix will indicate that the import is in progress:



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



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

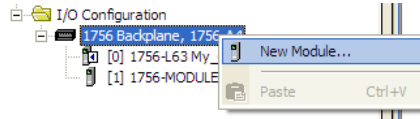


- 6 Save the application and then download the sample ladder logic into the processor.

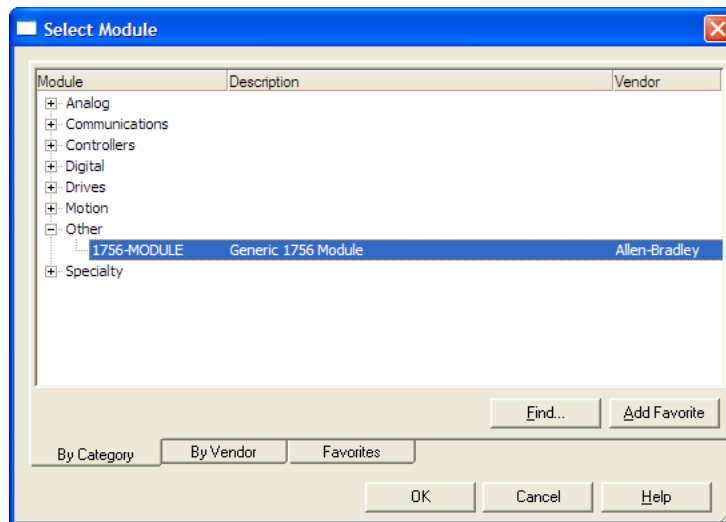
Adding Multiple Modules (Optional)

Important: If your application requires more than one MVI56E-MNETCR 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**.



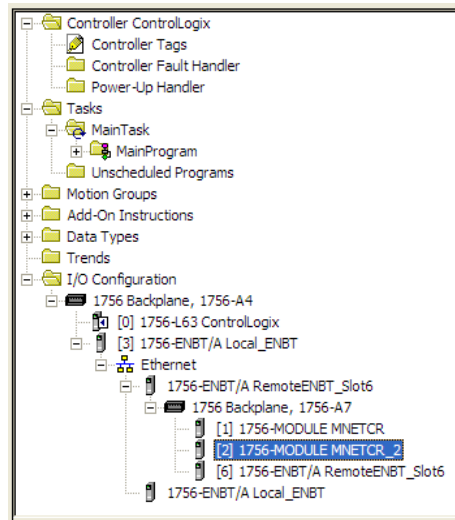
- 2 Select **1756-MODULE**



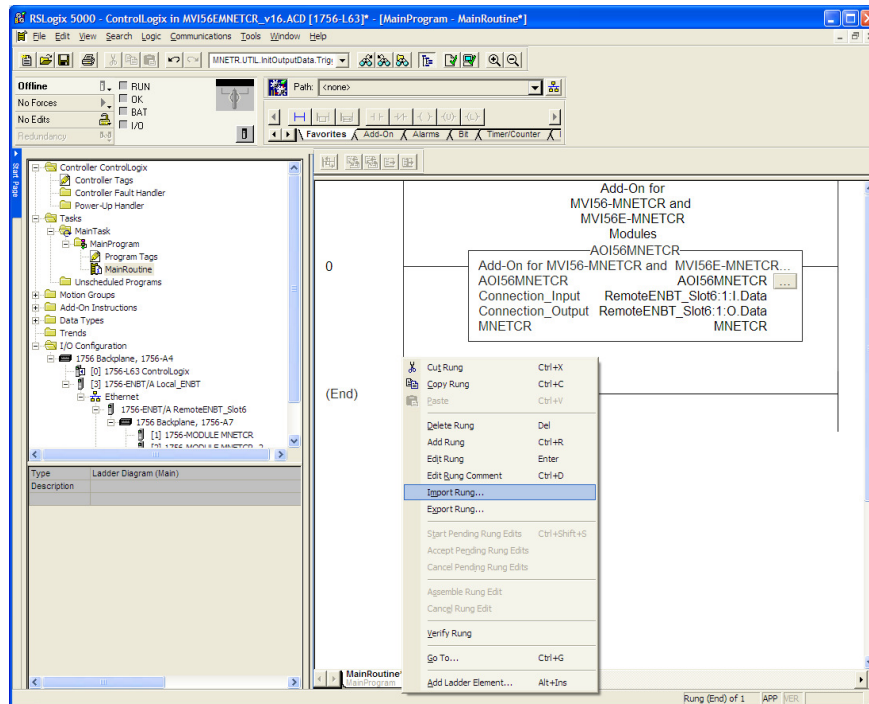
- 3 Fill the module properties as follows:

Parameter	Value
Name	Enter a module identification string. Example: MNETCR_2.
Description	Enter a description for the module. Example: Modbus TCP/IP Multi Client Enhanced Communications Module for Remote Chassis
Comm Format	Select DATA-INT .
Slot	Enter the slot number in the rack where the MVI56E-MNETCR module is located.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

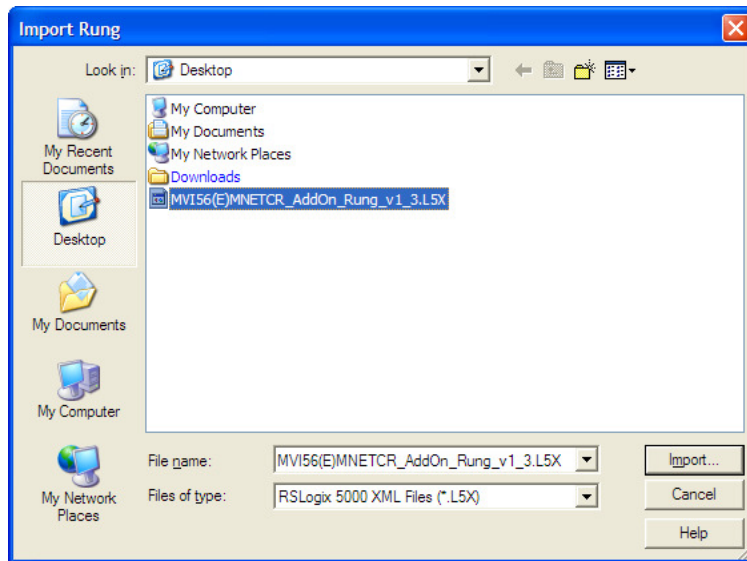
4 Click **OK** to confirm. The new module is now visible:



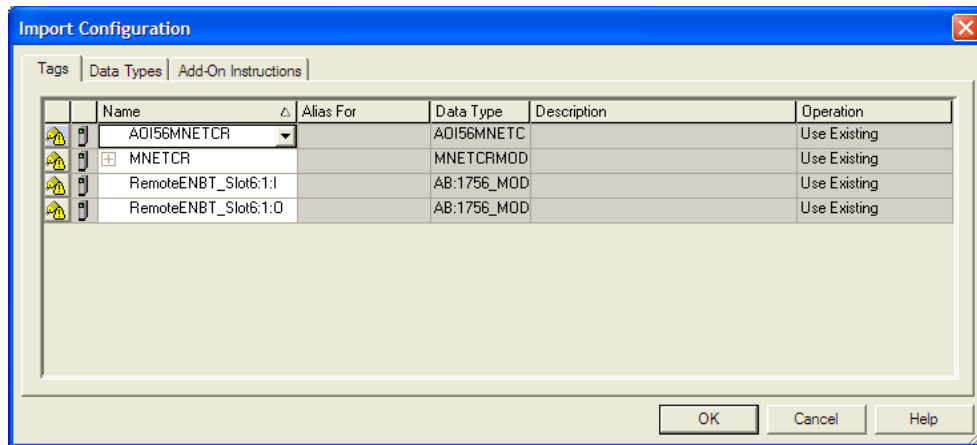
- 5 Expand the **TASKS** folder, and then expand the **MAINTASK** folder.
- 6 In the **MAINPROGRAM** folder, double-click to open the **MAINROUTINE** ladder.
- 7 Select an empty rung in the routine, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT RUNG...**



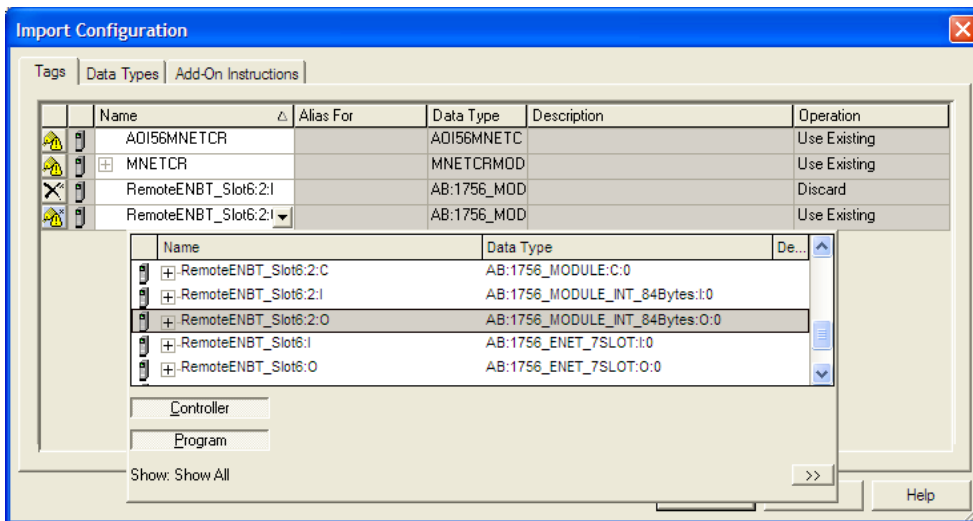
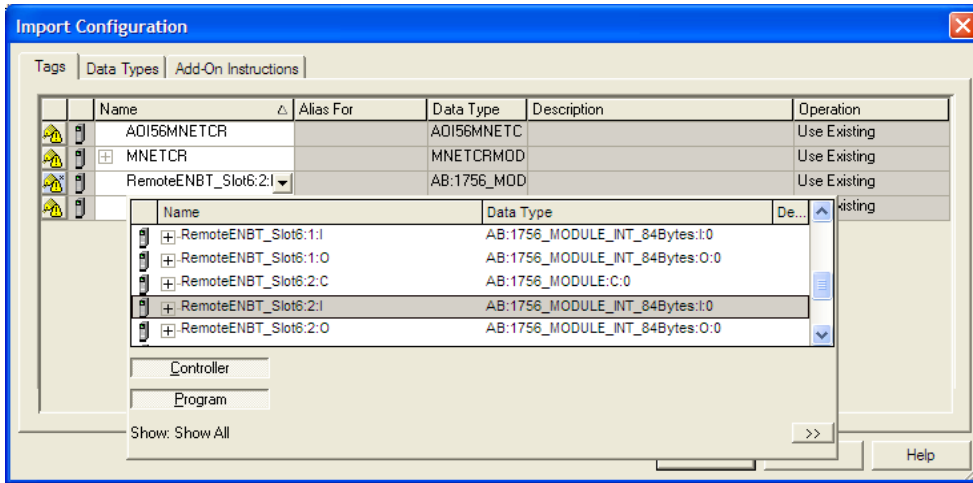
- 8 Select the **MVI56EMNETCR_ADDON_RUNG_v1_3.L5X** file, and then click **IMPORT**.



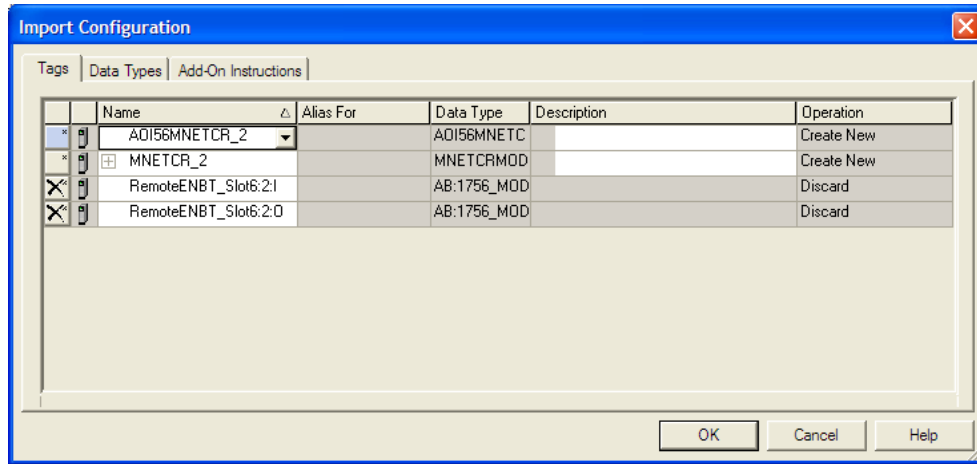
- 9 This action opens the **IMPORT CONFIGURATION** window, which shows the tags that will be imported.



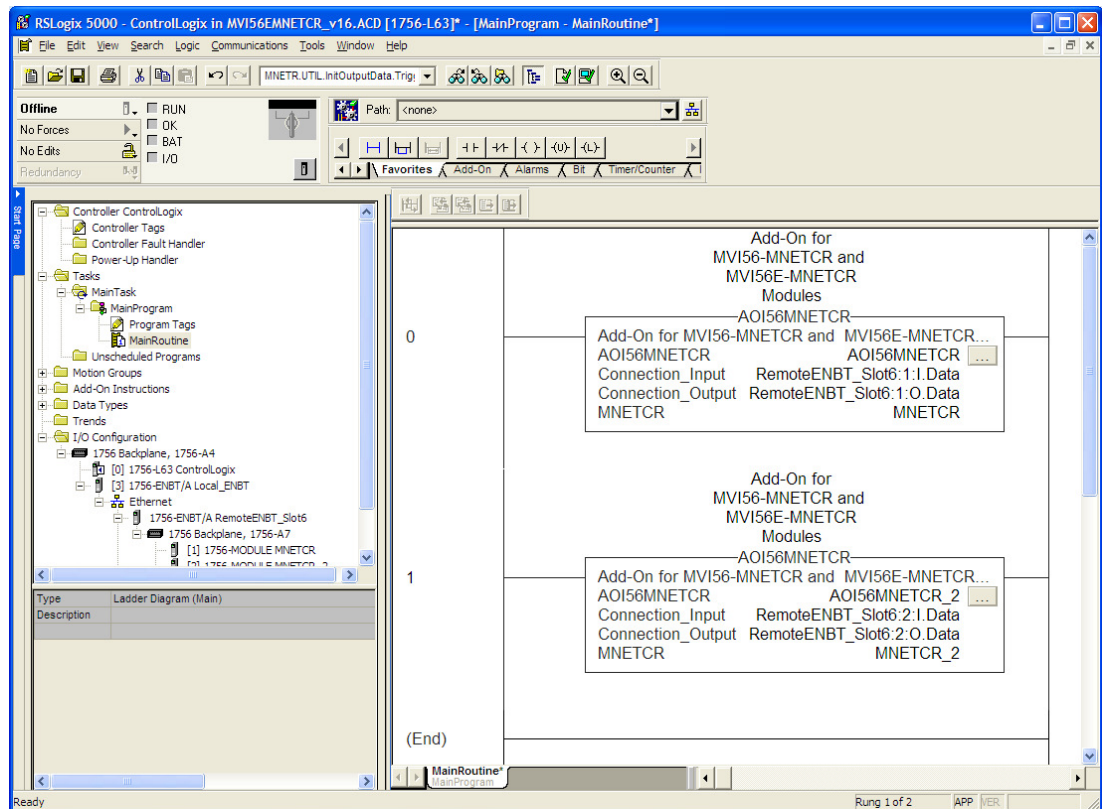
- Associate the I/O connection variables to the correct module. The default values are RemoteENBT_Slot6:1:I and RemoteENBT_Slot6:1:O so these require change.



11 Change the default tags **MNETCR** and **AOI56MNETCR** to avoid conflict with existing tags. In this procedure, you will append the string "_2" as shown in the following illustration.



12 Click **OK** to confirm.



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

Adjusting the Input and Output Array Sizes (Optional)

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

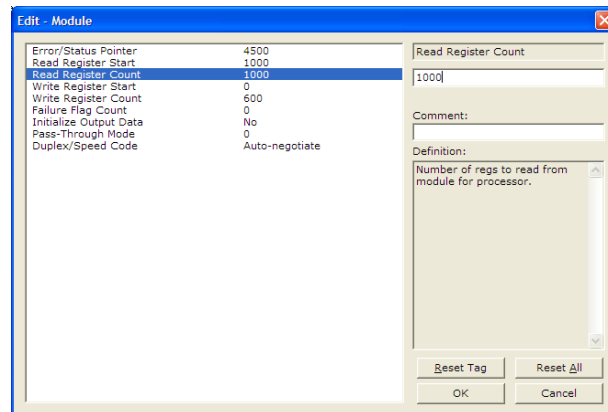
- Read Data
- Write Data

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

The MVI56E-MNETCR Add-On Instruction rung is configured for 600 registers of Read Data and 600 registers of Write Data, which is sufficient for most applications. However, you can configure the sizes of these data areas to meet the needs of your application.

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

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



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

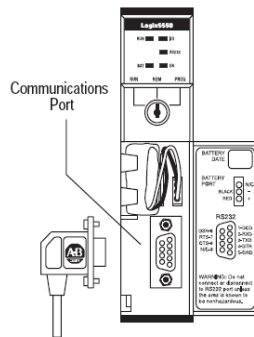
- 3 Save and download the sample configuration to the module.

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

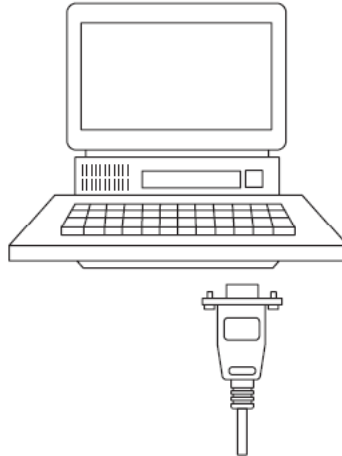
1.11.3 Connecting Your PC to the ControlLogix Processor

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

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



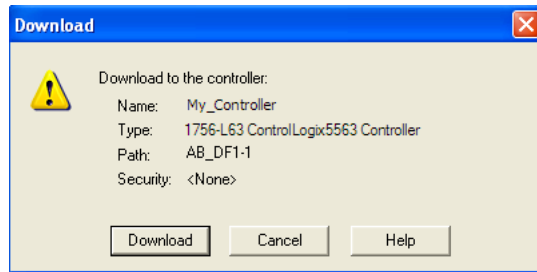
- 2 Connect the straight connector end of the cable to the serial port on your computer.



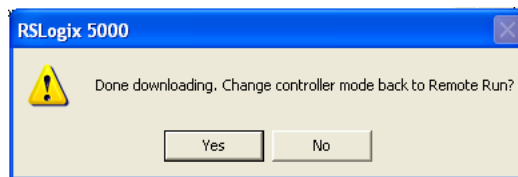
1.11.4 Downloading the Sample Program to the Processor

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

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



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



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

2 Configuring the MVI56E-MNETCR Module

In This Chapter

- ❖ Using ProSoft Configuration Builder Software..... 46
- ❖ Downloading the Project to the Module..... 62
- ❖ Using CIPconnect® to Connect to the Module..... 63

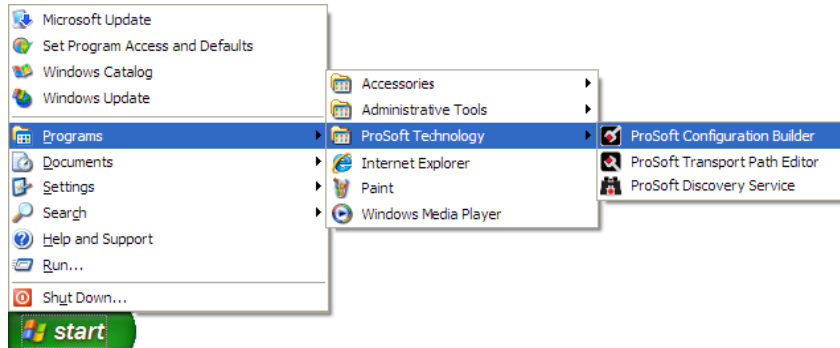
2.1 Using ProSoft Configuration Builder Software

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.

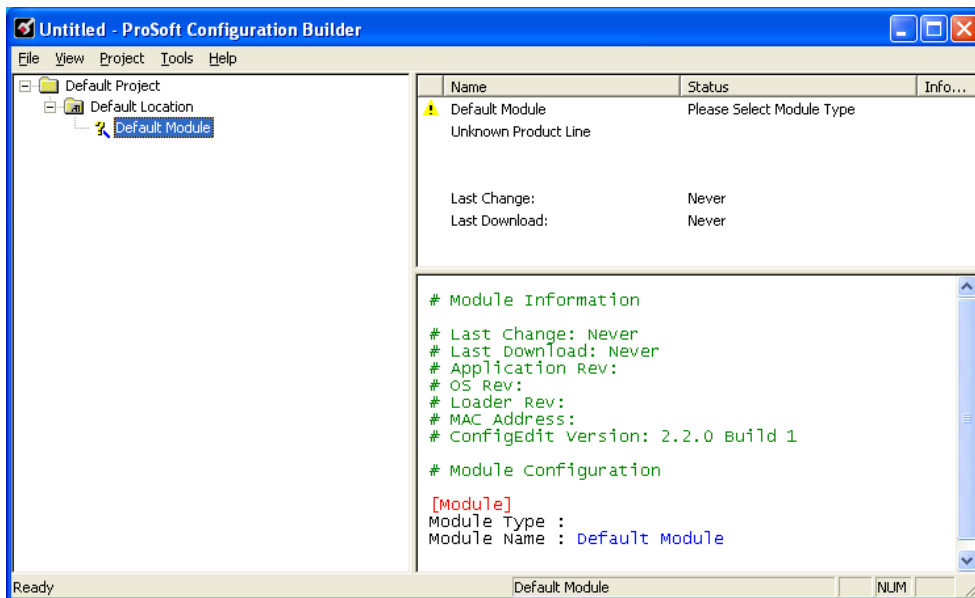
Note: During startup and initialization, the MVI56E-MNETCR module receives its protocol and backplane configuration information from the installed Personality Module (Compact Flash). Use *ProSoft Configuration Builder* to configure module settings and to download changes to the Personality Module.

2.1.1 Setting Up the Project

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



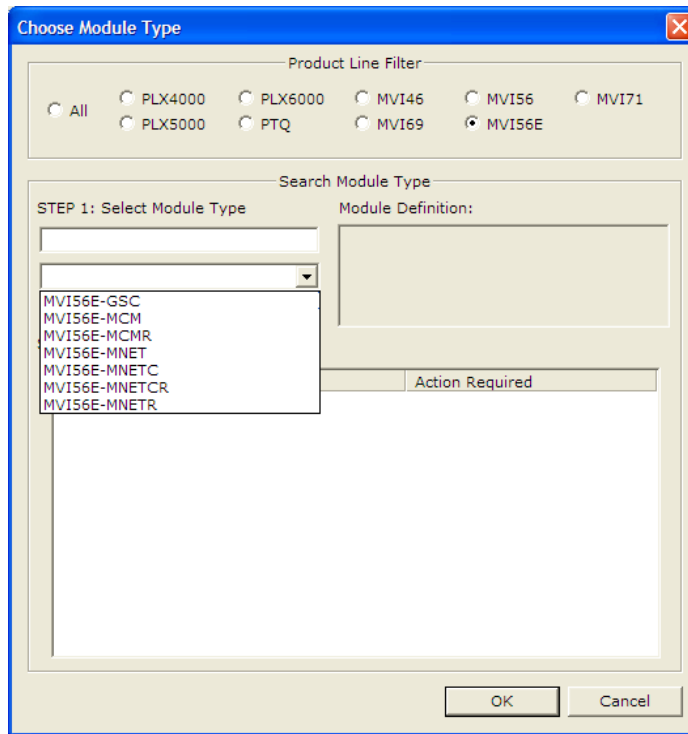
If you have used other Windows configuration tools before, you will find the screen layout familiar. *PCB's* window consists of a tree view on the left, and an information pane and a configuration pane on the right side of the window. When you first start *PCB*, the tree view consists of folders for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



Your first task is to add the MVI56E-MNETCR 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.

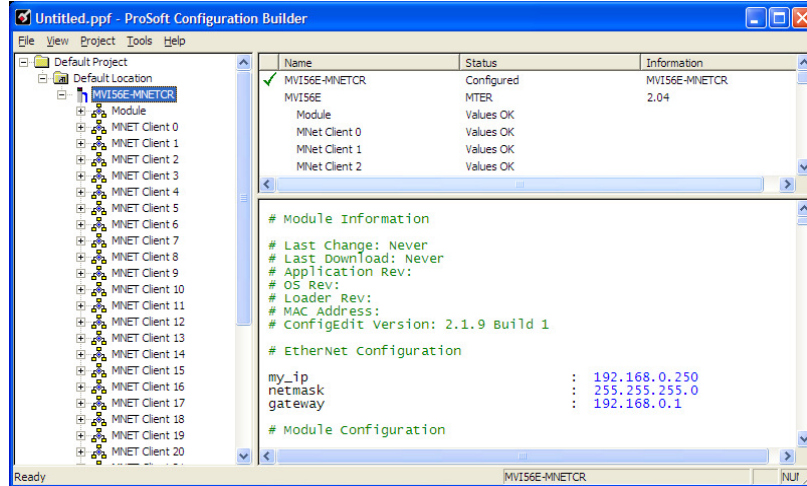
- 2 On the shortcut menu, select **CHOOSE MODULE TYPE**. This action opens the *Choose Module Type* dialog box.



- 3 In the *Product Line Filter* area of the dialog box, select **MVI56E**. In the **SELECT MODULE TYPE** dropdown list, select **MVI56E-MNETCR**, and then click **OK** to save your settings and return to the *ProSoft Configuration Builder* window.

2.1.2 Renaming PCB Objects

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



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

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

Configuring Module Parameters

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

Printing a Configuration File

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

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

Error/Status Pointer

-1 to 4955

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

Read Register Start

0 to 4999

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

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

Read Register Count

0 to 5000

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

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

Write Register Start

0 to 4999

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

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

Write Register Count

0 to 5000

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

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

Failure Flag Count

If this value is greater than zero the protocol communication will be interrupted once the backplane failure is detected, or communication with the processor fails. A value of zero will disable this feature.

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.

Duplex/Speed Code

0, 1, 2, 3 or 4

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

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

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

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

Minimum Command Delay

0 to 65535 milliseconds

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

Response Timeout

0 to 65535 milliseconds

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

Retry Count

0 to 10

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

Float Flag

YES or NO

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

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

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

Float Start

F0 to 65535

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

Float Offset

0 to 9999

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

ARP Timeout

1 to 60

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

Command Error Delay

0 to 300

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

MBAP Port Override

YES or NO

If this parameter is set to **YES**, all messages generated by the Client driver will be MBAP format messages to all Service Port values.

If this parameter is set to **No** (default value), or is omitted from the configuration file, all messages sent to Service Port 502 will be MBAP format messages, and all other Service Ports values will use the encapsulated Modbus message format (MNET).

Each Client is configured independently in the configuration file.

This parameter applies to firmware version 1.05 and above. For downward compatibility, you may omit this parameter from the Client's configuration.

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. Each command list record has the same format. The first part of the record contains the information relating to the MVI56E-MNETCR 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 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 16 commands.

The command list is processed from top (command #1) to bottom. A poll interval parameter is associated with each command to specify a minimum delay time in tenths of a second between the 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.

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
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 (Write) Multiple Coils
16	Preset (Write) Multiple Registers

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

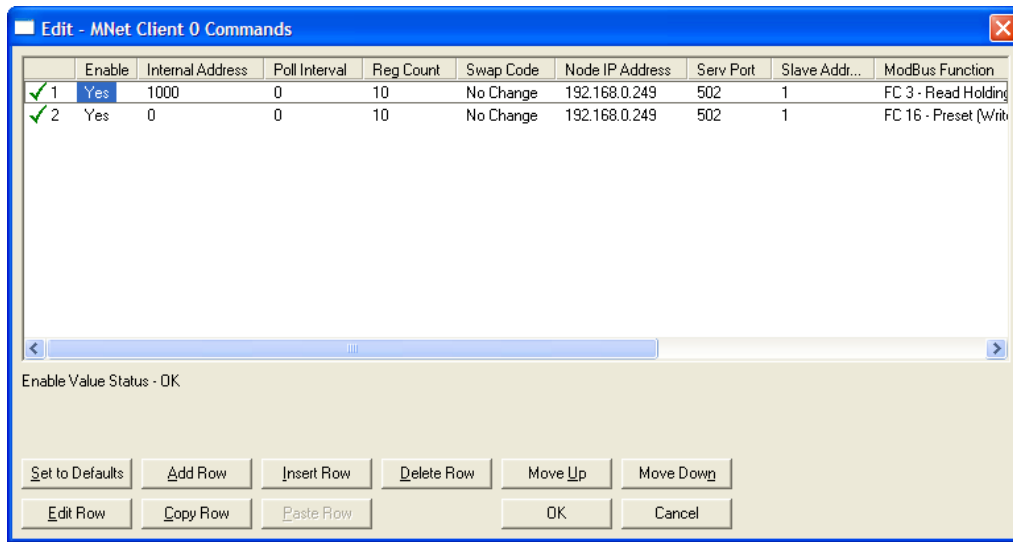
Command Entry Formats

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

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

The first part of the record is the module information, which relates to the MVI56E module and the second part contains information required to interface to the server device.

Command list example:



Enable

NO (0) or YES (1)

This field defines whether or not the command is to be executed.

Value	Description
No (0)	The command is disabled and will not be executed in the normal polling sequence.
Yes (1)	The command is executed each scan of the command list if the Poll Interval Time is set to zero. If the Poll Interval time is set, the command will be executed when the interval timer expires.

Internal Address

0 to 4999 (for word-level addressing)

or

0 to 65535 (for bit-level addressing)

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

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

Poll Interval

0 to 65535

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

Reg Count

Regs: **1 to 125**

Coils: **1 to 800**

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

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

Swap Code

NONE

SWAP WORDS

SWAP WORDS & BYTES

SWAP BYTES

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

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

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

Node IP Address

xxx.xxx.xxx.xxx

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

Service Port

502 or other supported ports on server

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

Slave Address

0 - Broadcast to all nodes

1 to **255**

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

Note: Use the *Node IP Address* parameter (page 58) to address commands to a remote Modbus TCP/IP device.

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

Modbus Function

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

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

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

MB Address in Device

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

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

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

Address calculation examples:

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

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

Comment

0 to 35 alphanumeric characters

2.1.6 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-MNETCR 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.

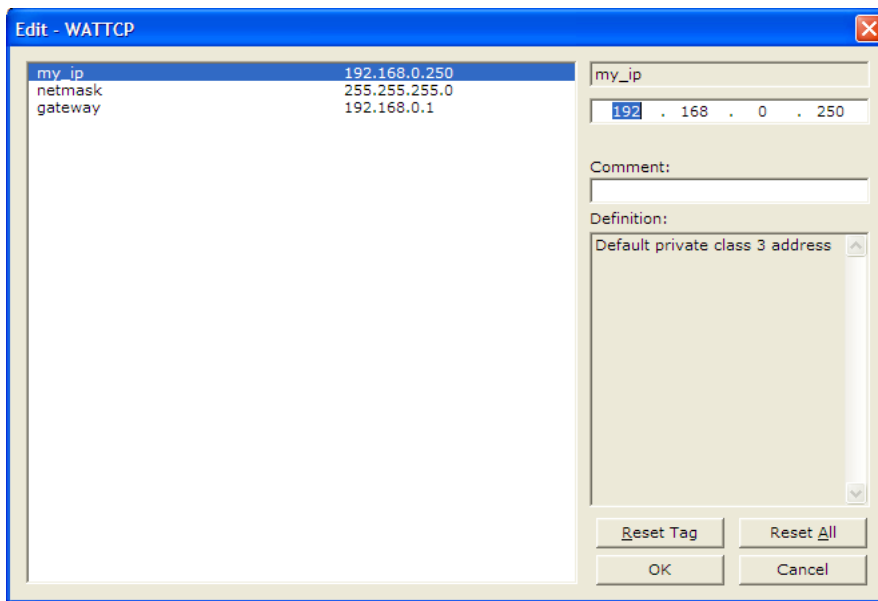
2.1.7 Ethernet Configuration

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.



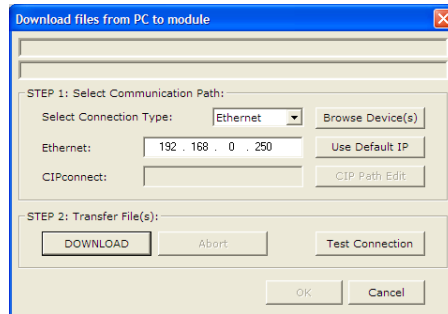
- 3 Edit the values for *my_ip*, *netmask* (subnet mask) and *gateway* (default gateway).
- 4 When you are finished editing, click **OK** to save your changes and return to the *ProSoft Configuration Builder* window.

2.2 Downloading 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-MNETCR 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.

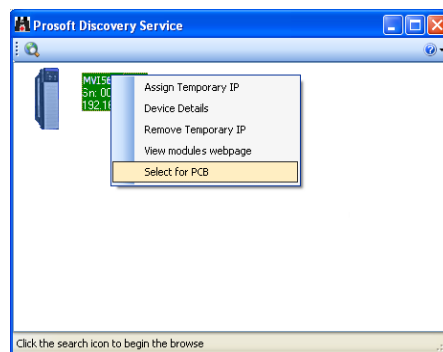


Click **TEST CONNECTION** to verify that the temporary IP address is correct.

- 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.

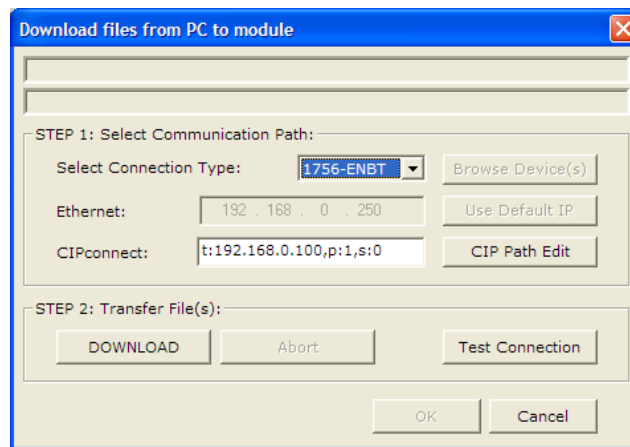
2.3 Using CIPconnect® to Connect to the Module

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

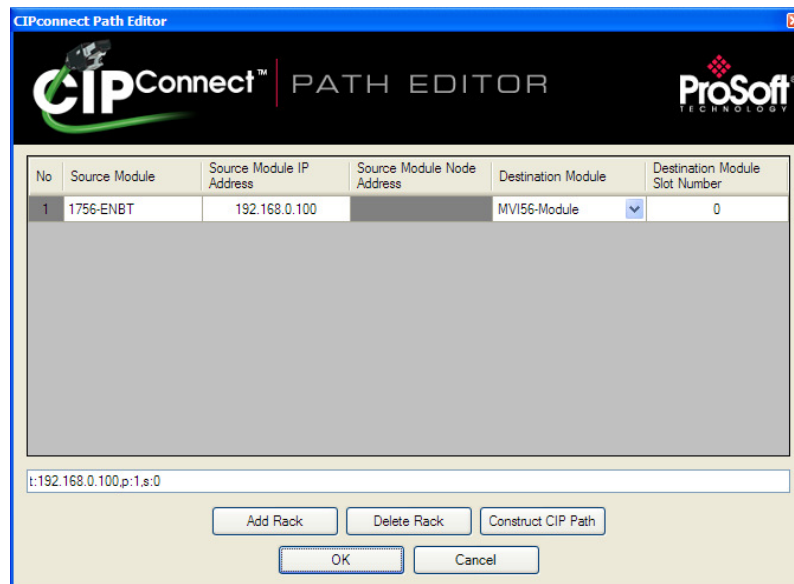
- The IP addresses and slot numbers of any 1756-ENBT modules in the path
- The slot number of the MVI56E-MNETCR in the destination ControlLogix chassis (the last ENBTx and chassis in the path).

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-MNETCR 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-MNETCR 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-MNETCR 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.

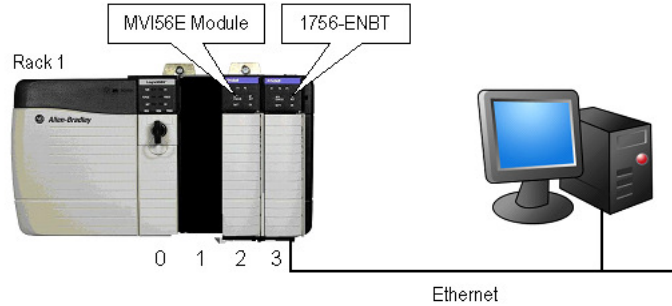
To use the CIPconnect Path Editor, follow these steps.

- 1** Configure the path between the 1756-ENBT connected to your PC and the MVI56E-MNETCR 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

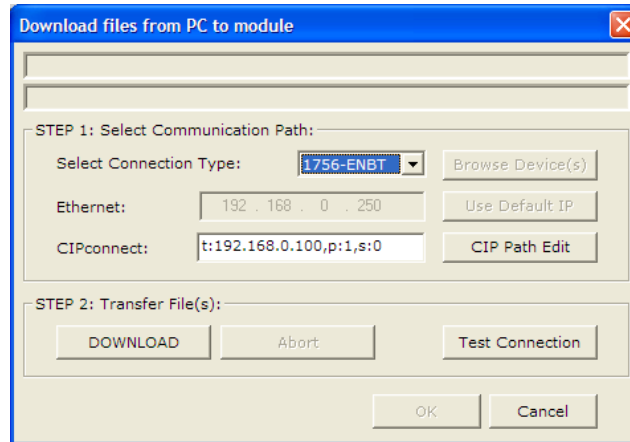
For this example, the MVI56E-MNETCR module is located in the same rack as the 1756-ENBT that is connected to the PC.



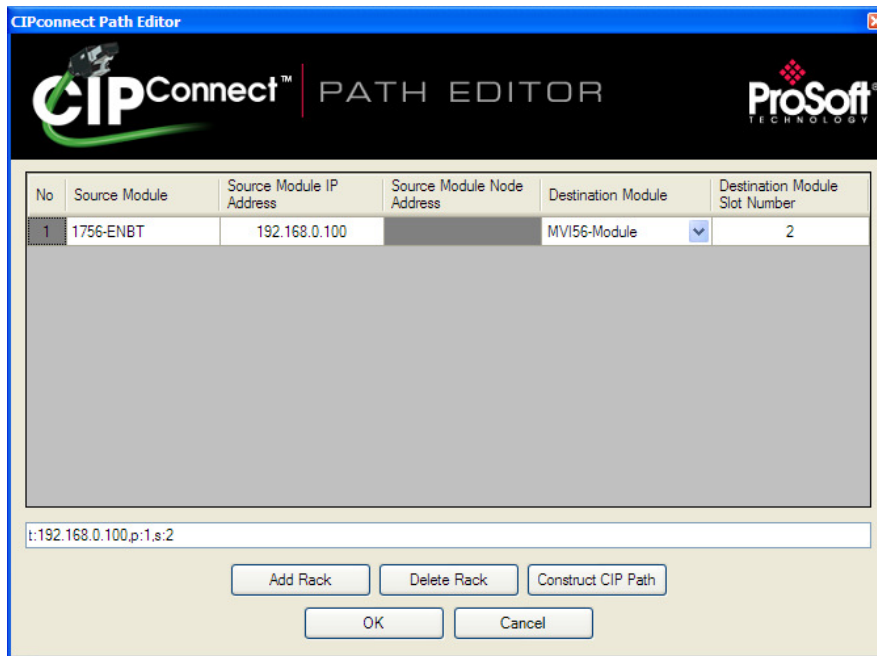
Rack 1

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

- 1 In the *Download* dialog box, click **CIP PATH EDIT**.

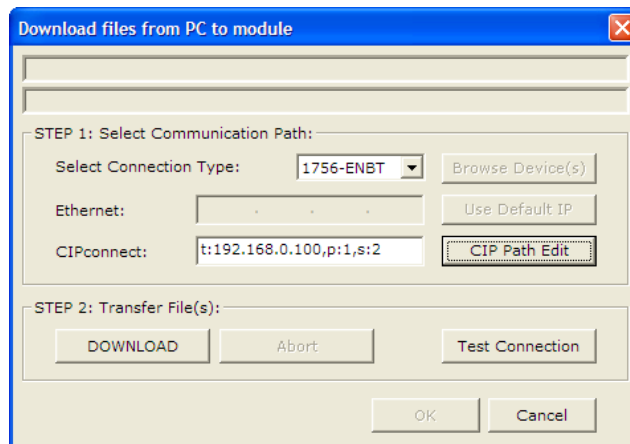


- 2 Configure the path as shown in the following illustration, and click **CONSTRUCT CIP PATH** to build the path in text format.

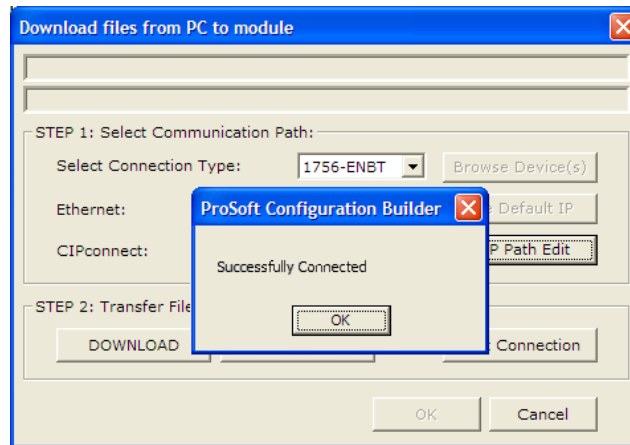


Click **OK** to close the *CIPconnect Path Editor* and return to the *Download* dialog box.

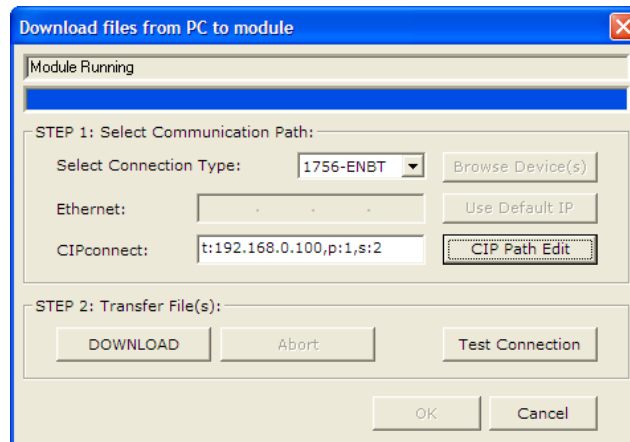
- 3 Check the new path in the *Download* dialog box.



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

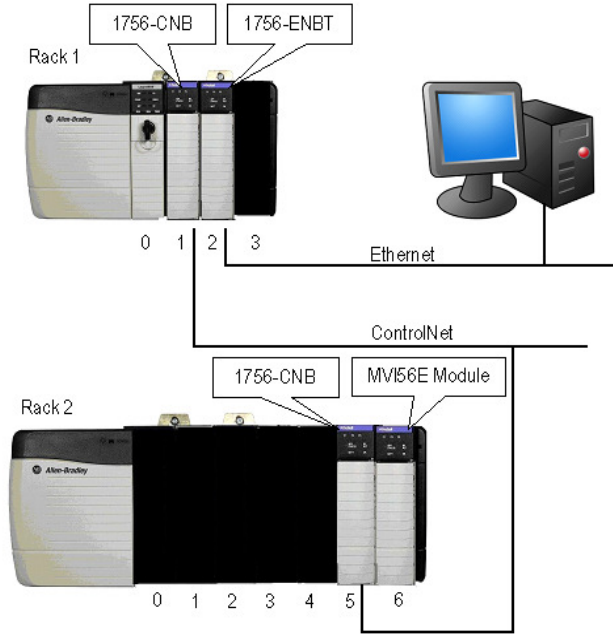


- 5 Click **OK** to close the Test Connection pop-up and then click **DOWNLOAD** to download the configuration files to the module through the path.



2.3.2 Example 2: Remote Rack Application

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



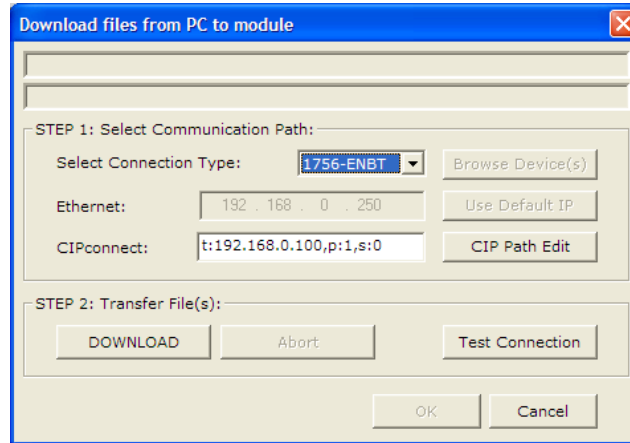
Rack 1

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

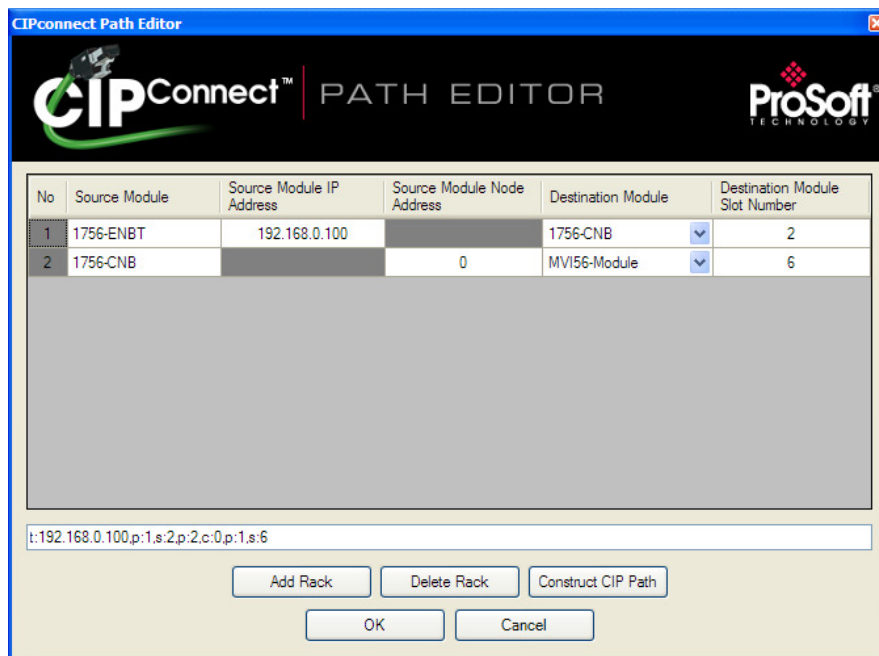
Rack 2

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

- 1 In the *Download* dialog box, click **CIP PATH EDIT**.

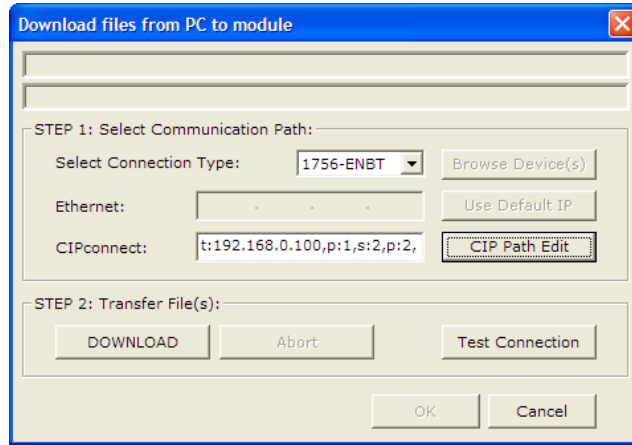


- 2 Configure the path as shown in the following illustration and click **CONSTRUCT CIP PATH** to build the path in text format.

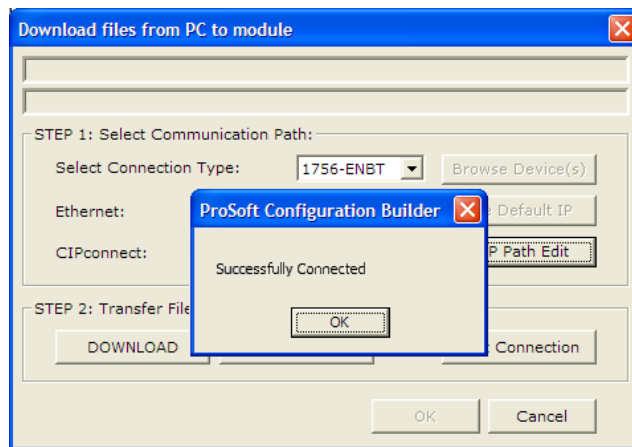


Click **OK** to close the *CIPconnect Path Editor* and return to the *Download* dialog box.

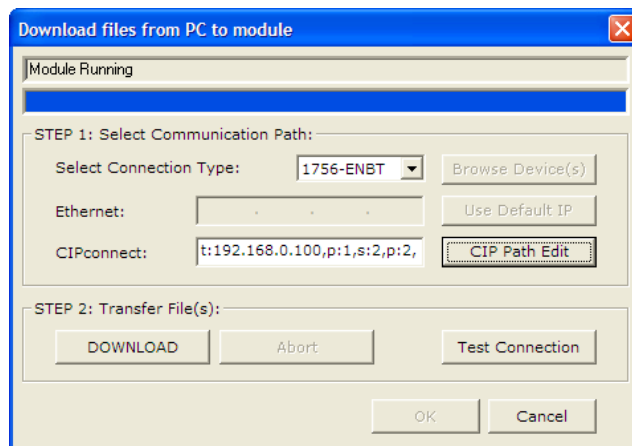
- 3 Check the new path in the *Download* dialog box.



- 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

- ❖ Module Data Object (MNETCRMODULEDEF)..... 72

Ladder logic is required for managing communication between the MVI56E-MNETCR module and the processor. The ladder logic handles tasks such as:

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

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

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

3.1 Module Data Object (MNETCRMODULEDEF)

All data related to the MVI56E-MNETCR 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 this object.

Name	Data Type	Description
DATA	MNETCRDATA (page 73)	Data read from module
CONTROL	MNETCRCONTROL (page 74)	MNETCR Module control
STATUS	MNETCRSTATUS (page 75)	Client ,Server Status and blocks status
UTIL	MNETCRUTIL (page 77)	Block statistics

This object contains objects that define the configuration, user data, status and command control data related to the module.

3.1.1 User Data Object (MNETCRDATA)

These objects hold data to be transferred between the processor and the MVI56E-MNETCR module. The user data is the read and write data transferred between the processor and the module as "pages" of data up to 200 words long.

Name	Data Type	Description
ReadData	INT[600]	Data Read from the Module. Set array equal to the size set in the Configuration file.
WriteData	INT[600]	Data Write to the Module. Set array equal to the size set in the Configuration file.

ReadData is a controller tag array that should be sized to match the value entered in the **READ REGISTER COUNT** parameter of the .CFG file or in ProSoft Configuration Builder (PCB) (page 50). For ease of use, both this array and the **READ REGISTER COUNT** should be dimensioned in even increments of 40 words. This data is paged up to 40 words at a time from the module to the processor. The *ReadData* ladder logic task places the data received into the proper position in the *MNETCR.DATA.ReadData* array. The data received in the *ReadData* array will contain status and control information sent by other nodes on the network for use by processor logic. The *ReadData* array may also be used to hold certain module status and error data which can then be monitored by processor logic.

WriteData is a controller tag array that should be sized to match the value entered in the **WRITE REGISTER COUNT** parameter of the .CFG file or in PCB (page 51). For ease of use, both this array and the **WRITE REGISTER COUNT** parameter should be dimensioned in even increments of 40 words. This data is paged up to 40 words at a time from the processor to the module. The *WriteData* ladder logic task gets the data from the proper position in the *MNETCR.DATA.WriteData* and puts that data into the output image for transfer to the module. Once the data has been stored in the module's database, it is available for use as status and control information by other nodes on the network.

3.1.2 Command Control Data Object (MNETCCONTROL)

Values used by program for data transfer operation between the module and the processor.

Name	Data Type	Description
BootTimer	TIMER	Timer used to clear both cold and warm boot requests
WarmBoot	BOOL	Hardware reset of the Module
ColdBoot	BOOL	Configuration data reset in the Module
EventCmdTrigger	BOOL	Initiates Event Command.
EventCmdPending	BOOL	Allows Event Command.
ClientID	INT	Client ID to poll a server with the Event Command.
EventCmd	MNETCREVENTCMD (page 74)	Holds Event Command configuration
CmdControl	MNETCRCMDCONTROL (page 75)	Holds Command Control status
CmdControlTrigger	BOOL	Initiates Command Control.
CmdControlPending	BOOL	Halts rung until Module is ready
IPAddress	MNETCRIPADDRESS (page 75)	IP address statistics including triggers
WriteCmdBits	INT[30]	Selects individual clients to activate its commands.

Event Command Data Object (MNETCREVENTCMD)

The Event object is not placed in the module definition object and is only required when event commands are utilized in the application. This structure holds the information required for an event command. An array of these objects should be defined and hold the event command set to be employed in the application. The following illustration shows the structure of the 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
SlvAddrNode	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
MBFunction	INT	Modbus function code for message
Address	INT	Address to interface with in device

Command Control Data Object (MNETCRCMDCONTROL)

MVI56E-MNETCR Command Control

Name	Data Type	Description
ClientIDreq	INT	Requested Client ID.
CmdQty	INT	Command Quantity for the Client.
CmdIndex	INT[16]	Command Index per client. (0 to 15)

IP Address Object (MNETCRIPADDRESS)

This object holds IP address status with set IP & get IP triggers.

Name	Data Type	Description
IPgetTrigger	BOOL	Triggers get IP address.
IPreceived	INT[4]	Displays received IP address.
IPsetTrigger	BOOL	Triggers set IP address
IPrequested	INT[4]	Displays requested IP address

3.1.3 Status Object (MNETCRSTATUS)

This object contains the status of the module. The MNETCRSTATUS object 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".

Name	Data Type	Description
PassCnt	INT	Program cycle counter
BlockStats	MNETCRBLOCKSTATS (page 76)	Block Statistics
CmdBits	INT[30]	Commands bits array to be used for 30 clients
ClientStatsTrigger	BOOL	Get Client Status
ClientIDReq	INT	Client ID requested.
ClientStatus	MNETCRCLIENTSTATS[30] (page 76)	Client Status requests
ClientIDRec	INT	Client ID received.
CmdErrorList	INT[16]	Command Error List
ClientStatsPending	BOOL	Allows Get Client Status

Block Statistics Object (MNETCRBLOCKSTATS)

This object contains a structure that includes the status information for the data transfer operations between the processor and the module. 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

Client Statistics Object (MNETCRCLIENTSTATS)

This block also contains a structure that includes Client statistics. 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-MNETCR Status Data Definition (page 105, page 112) for a complete listing of the data stored in the status object.

3.1.4 Backplane Control Object (MNETCRUTIL)

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

Name	Data Type	Description
LastRead	INT	Index of last read block
LastWrite	INT	Index of last write block
BlockIndex	INT	Computed block offset for data table
ReadDataSizeGet	INT	Gets ReadData Array Length.
WriteDataSizeGet	INT	Gets WriteData Array Length.
ReadDataBlkCount	INT	Holds the value of the Block Counts of the Read Data Array.
WriteDataBlkCount	INT	Holds the value of the Block Counts of the Write Data Array.
RBTSremainder	INT	Holds remainder calculation value from the read array.
WBTSremainder	INT	Holds remainder calculation value from the write array.
IPgetPending	BOOL	Allows setting module IP address
IPsetPending	BOOL	Allows getting module IP address

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.

4 Diagnostics and Troubleshooting

In This Chapter

- ❖ LED Status Indicators..... 80
- ❖ Using the Diagnostics Menu in ProSoft Configuration Builder..... 84
- ❖ Reading Status Data from the Module 91

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

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

4.1 LED Status Indicators

4.1.1 Scrolling 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 <ul style="list-style-type: none"> ▪ 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 Add-On Instruction. After power up and every reconfiguration, the module will display the configuration of the application port(s). The information consists of: Client <ul style="list-style-type: none"> ▪ C0 C2 C3 C4 C29

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 (page 79) for further information about the error.

4.1.2 Ethernet LED Indicators

The Ethernet LEDs indicate the module's Ethernet port status as follows:

LED	State	Description
Data	OFF	Ethernet connected at 10Mbps duplex speed
	AMBER Solid	Ethernet connected at 100Mbps duplex speed
Link	OFF	No physical network connection is detected. No Ethernet communication is possible. Check wiring and cables.
	GREEN Solid or Blinking	Physical network connection detected. This LED must be ON solid for Ethernet communication to be possible.

4.1.3 Non-Scrolling LED Status Indicators

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

LED Label	Color	Status	Indication
APP	Red or Green	OFF	The module is not receiving adequate power or is not securely plugged into the rack. May also be OFF during configuration download.
		GREEN	The MVI56E-MNETCR is working normally.
		RED	The most common cause is that the module has detected a communication error during operation of an application port. The following conditions may also cause a RED LED: <ul style="list-style-type: none"> ▪ The firmware is initializing during startup ▪ The firmware detects an on-board hardware problem during startup ▪ Failure of application port hardware during startup ▪ The module is shutting down ▪ The module is rebooting due to a ColdBoot or WarmBoot request from the ladder logic or Debug Menu
OK	Red or Green	OFF	The module 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.
ERR			Not Used

4.1.4 Troubleshooting

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

Processor Errors

Problem description	Steps to take
Processor fault	Verify that the module is plugged into the slot that has been configured for the module in the I/O Configuration of RSLogix. Verify that the slot location in the rack has been configured correctly in the ladder logic.
Processor I/O LED flashes	This indicates a problem with backplane communications. A problem could exist between the processor and any installed I/O module, not just the MVI56E-MNETCR. Verify that all modules in the rack are correctly configured in the ladder logic.

Module Errors

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

4.1.5 Clearing a Fault Condition

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

To clear the condition, follow these steps:

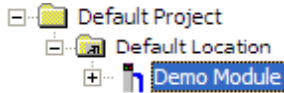
- 1** Turn off power to the rack.
- 2** Remove the card from the rack.
- 3** Verify that all jumpers are set correctly.
- 4** If the module requires a Compact Flash card, verify that the card is installed correctly.
- 5** Re-insert the card in the rack and turn the power back on.
- 6** Verify correct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

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

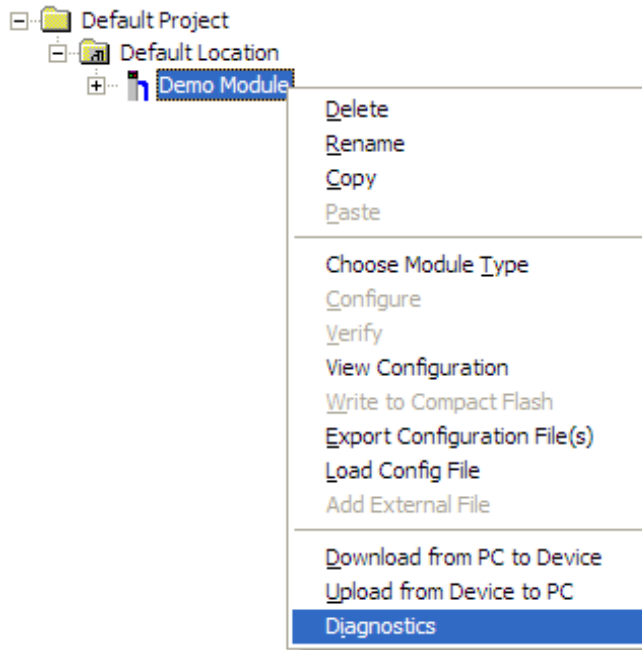
4.2 Using the Diagnostics Menu in ProSoft Configuration Builder

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

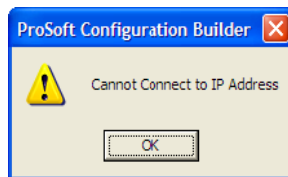
- 1 In *ProSoft Configuration Builder*, select the module, and then click the right mouse button to open a shortcut menu.



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



This action opens the *Diagnostics* dialog box.
If there is no response from the module,

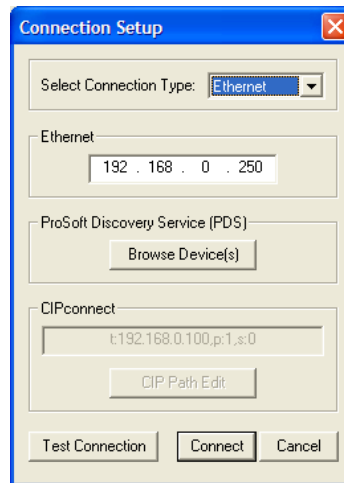


- 1 Click the **SET UP CONNECTION** button to browse for the module's IP address.

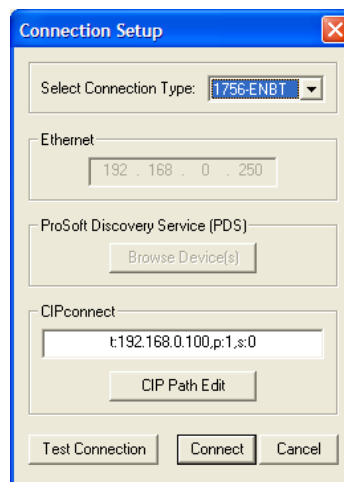


Click to set up connection

- 2 In the *Connection Setup* dialog box, click the **TEST CONNECTION** button to verify if the module is accessible with the current settings.

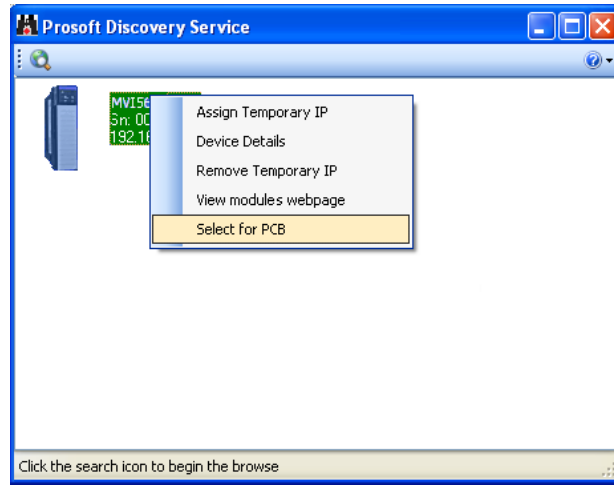


You can also use CIPconnect[®] to connect to the module through a 1756-ENBT card.



Refer to Using CIPconnect to Connect to the Module (page 63, page 22) for information on how to construct a CIP path.

- 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**.



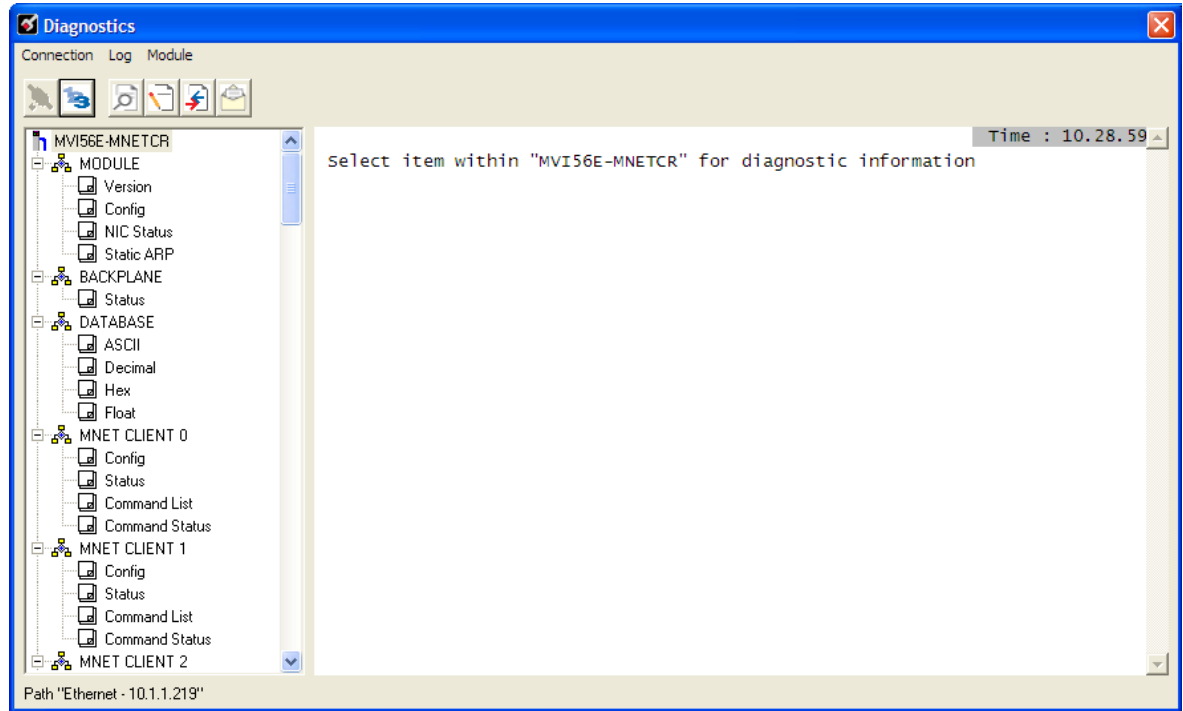
Close *ProSoft Discovery Service*, and click the **CONNECT** button again.

- 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.

4.2.1 The Diagnostics Menu

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



4.2.2 Monitoring Module Information

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

Version

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

```
MVI56E-MNETCR > MODULE > Version :  
PRODUCT NAME CODE           :MTER  
SOFTWARE REVISION LEVEL     :2.04  
OPERATING SYSTEM REVISION   :1109  
RUN NUMBER                   :1201  
PROGRAM SCAN COUNTER        :43337  
IP ADDRESS                   :10.1.3.195  
ETHERNET ADDRESS (MAC)      :00:0d:8d:00:3a:98  
BACKPLANE DRIVER VERSION    :2.06  
BACKPLANE API VERSION       :1.01  
MODULE NAME                  :MVI56E-MNETCR  
VENDOR ID                    :309  
DEVICE TYPE                  :12  
PRODUCT CODE                 :5012  
SERIAL NUMBER                :000003E9  
REVISION                     :2.04  
SLOT                         :1
```

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

Config

Use the *Configuration* menu to view backplane configuration settings for the MVI56E-MNETCR module.

The information on this menu corresponds with the configuration information in the *Module* settings in *ProSoft Configuration Builder*.

NIC Status

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

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

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 60) defines a list of static IP addresses that the module will use when an ARP is required.

4.2.3 Monitoring Backplane Information

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

Backplane Status

Use the *Status* menu 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-MNETCR Status object.

4.2.4 Monitoring MNET Client Information

Use the *MNET CLIENT* menu to view the configuration and status information for the MNET Client(s).

Config

Use the *Configuration* menu to view configuration settings for MNET Client x. The information on this menu corresponds with the configuration information in the *MNET Client x* settings in *ProSoft Configuration Builder*.

Status

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

Command List

Use the *Command List* menu to view the command list settings for MNET Client x. The information on this menu corresponds with the settings in the *MNET Client x Commands* settings in *ProSoft Configuration Builder*.

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

Command Status

Use the *Command Status* menu to view MNET Client x Command status.

A zero indicates no error.

A non-zero value indicates an error. Refer to Client Command Errors (page 113) for an explanation of each value.

4.2.5 Monitoring Database Information

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

You can view data in the following formats:

ASCII

```

DATABASE DISPLAY 0 to 99 (ASCII) :
' 0 M C E R 2 . 0 1 0 1 0 9 2 1 0 1
j 0 1 0 0 0 0 0 0 0 0 0 0 0 0
i 0 $ 0 0 0 0 0 0 0 0 0 0 0 0
    
```

Decimal

```

DATABASE DISPLAY 0 to 99 (DECIMAL) : [Refresh Counter: 24]
      0 5520 17229 21061 11826 12592 13360 14640 12594 12592
      892 0 3566 3567 0 0 0 0 892 0
      3566 3567 0 0 0 0 28075 28074 28074 0
      0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0
    
```

Float

```

DATABASE DISPLAY 0 to 49 (FLOAT) : [Refresh Counter: 8]
-1.42363105E+028 2.11809419E+011 2.56376298E-009 1.68041093E-004 2.56393351E-009
1.25976732E-042 1.71398323E-030 0.00000000E+000 0.00000000E+000 1.25976732E-042
1.71398323E-030 0.00000000E+000 0.00000000E+000 1.08282789E+034 4.30548953E-041
0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000
0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000
0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000
0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000
0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000
0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000
0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000
0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000 0.00000000E+000
    
```

Hexadecimal

```

DATABASE DISPLAY 0 to 99 (HEXADECIMAL) :
0000 8164 434D 5245 2E32 3130 3430 3930 3132 3130
038A 0000 0E26 0E27 0000 0000 0000 0000 038A 0000
0E26 0E27 0000 0000 0000 0000 81EE 81ED 81ED 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    
```

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

4.3 Reading Status Data from the Module

The MVI56E-MNETCR module maintains a Status Data Table that can be used to determine the module's operating status. This data is located in the module's database at the location specified by the Error Status Pointer configuration parameter (page 50). This data is transferred to the processor continuously. For a complete listing of the status data object, refer to Status Data Definition (page 105, page 112).

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 3999)
- Version Information
- Control over the module (warm boot, cold boot, transfer configuration)
- Facility to upload and download the module's configuration file

5 Reference

In This Chapter

- ❖ Product Specifications 94
- ❖ Functional Overview 97
- ❖ Data Flow between MVI56E-MNETCR Module, Processor, and Network 111
- ❖ Ethernet Cable Specifications 116
- ❖ Modbus Protocol Specification 118
- ❖ Using the Optional Add-On Instruction Rung Import 132
- ❖ Adding the Module to an Existing Project 142
- ❖ Using the Sample Program - RSLogix 5000 Version 15 and earlier.... 145

5.1 Product Specifications

The MVI56E Modbus TCP/IP Multi Client Enhanced Communications Module for Remote Chassis allows Rockwell Automation® ControlLogix® Programmable Automation Controllers (PACs) to interface easily with multiple Modbus TCP/IP server-compatible instruments and devices. The multi-Client module improves performance when controlling multiple servers on a Modbus TCP/IP network, by supporting up to 30 Clients.

MVI56E enhancements include configuration and management through the module's Ethernet port, CIPconnect® technology for bridging through ControlNet™ and EtherNet/IP™ networks, and a web server to access on-board documentation, Add-On Instructions, and sample program files.

This module uses a small I/O data image for transfer of data between the module and the ControlLogix processor, making it ideal for ControlNet or Ethernet applications with the module in a remote rack.

5.1.1 General Specifications

- Backward compatible with previous MVI56-MNETC versions
- Single-slot 1756 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
- ProSoft Configuration Builder (PCB) software supported, a Windows-based graphical user interface providing simple product and network configuration
- Sample ladder logic and Add-On Instructions (AOI) are used for data transfer between module and processor
- 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 for easy disaster recovery, allowing quick in-the-field product replacement by transferring the CF card

Modbus TCP/IP Client (Master)

The MVI56E-MNETCR is a Client-only module that will operate on a local or remote rack. This module was created to improve performance when controlling multiple servers on a Modbus TCP/IP network. The module supports up to 30 Clients with up to 16 commands for each Client.

- Actively reads data from and writes data to Modbus TCP/IP devices, using MBAP or Encapsulated Modbus message formats
- Transmits Modbus Function Codes 1, 2, 3, 4, 5, 6, 7, 15, and 16
- Offers 30 Client connections with up to 16 commands each to talk to multiple servers
- ControlLogix processor can be programmed to use special functions to control the activity on the Client by actively selecting commands to execute from the command list (Command Control) or by issuing commands directly from the ladder logic (Event Commands)

5.1.2 Functional Specifications

- Modbus data types overlap in the module's memory database, so the same data can be conveniently read or written as bit-level or register-level data.
- Configurable floating point data movement is supported, including support for Enron or Daniel[®] floating point formats
- Special functions (Command Control, Event Commands, status, etc.) are supported by message transfer (unscheduled) using the MSG instruction
- Configurable parameters for the Client including a minimum response delay of 0 to 65535 ms and floating point support
- Supports up to 30 Clients with up to 16 commands for each Client
- Error codes, counters, and module status available from module memory through the Clients, or through the ladder logic and controller tags in RSLogix 5000

5.1.3 Hardware Specifications

Specification	Description
Backplane Current Load	800 mA @ 5 Vdc 3 mA @ 24 Vdc
Operating Temperature	0°C to 60°C (32°F to 140°F)
Storage Temperature	-40°C to 85°C (-40°F to 185°F)
Shock	30 g operational 50 g non-operational Vibration: 5 g from 10 to 150 Hz
Relative Humidity	5% to 95% (without condensation)
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

5.2.1 About the MODBUS/TCP 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.

MODBUS/TCP 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 commands to the server. The MVI56E-MNETCR module simulates up to 30 clients.

Aside from the benefits of Ethernet versus serial communications (including performance, distance, and flexibility) for industrial networks, the MODBUS/TCP protocol allows for remote administration and control of devices over an Internet connection. It is important to note that not all Internet protocols are implemented in the module, for example, HTTP and SMTP protocols are not available. Nevertheless, the efficiency, scalability, and low cost of a MODBUS/TCP network make this an ideal solution for industrial applications.

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

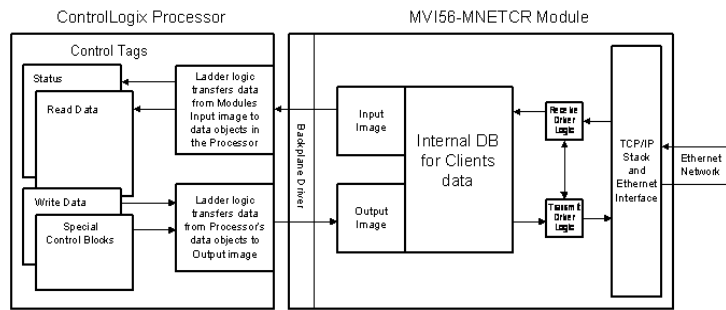
5.2.2 Backplane Data Transfer

The MVI56E-MNETCR 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.

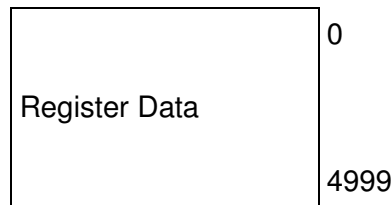
The following illustration shows the data transfer method used to move data between the ControlLogix processor, the MVI56E-MNETCR module and the Modbus TCP/IP Network.



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

Module's Internal Database Structure

5000 registers for user data



Data contained in this database is paged through the input and output images by coordination of the ControlLogix ladder logic and the MVI56E-MNETCR 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	Null block
0	Null block
1 to 125	Read or write data
1000 to 1124	Output Initialization Blocks
2000 to 2029	Event Command Block (page 103)
3000 to 3029	Client status request/response blocks (page 105)
5001 to 5016	Command Control (page 106, page 102)
9990	Set new module IP address (page 107)
9991	Get module IP address
9998	Warm-boot control block (page 109)
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), and Command control blocks (2000, 5000 to 5016, 9998 and 9999).

Normal Data Transfers

Normal data transfer includes the paging of the user data and status data found in the module’s internal database in registers 0 to 4999. These data are transferred through read (input image) and write (output image) blocks. The following topics describe the function and structure of each block.

Block Response from Module to Processor

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

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

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

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

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

R1W1→R2W2→R3W1→R1W2→R2W1→R3W2→R1W1→

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

Block Request from Processor to Module

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

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

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

Status Read Data Block IDs 0 and -1

This block is automatically copied from the module into the MNETCR.STATUS array when the block ID is 0 or -1, 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; Read, Write, Parse, Command, Event and Error Counts	6
8 to 37	Client 0 to Client 29 Command Execution Control Bits	2
38 to 40	Reserved	17
41	Read Block ID (-1 or 0)	1

Status Data Block Structure

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

Offset	Content	Description
1	Pass Count	This value is incremented each time a complete program cycle occurs in the module.
2	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
3	Write Block Count	This field contains the total number of write blocks transferred from the processor to the module.
4	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
5	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
6	Command Block Count	This field contains the total number of command blocks received from the processor.
7	Error Block Count	This field contains the total number of block errors recognized by the module.
8 to 37	Enable/Disable Command Bits	Each bit in each word is used to enable or disable individual commands in each client

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

5.2.3 Special Function Blocks

Special function blocks are special optional blocks used to request special tasks from the module. The MVI56E-MNETCR supports the following special function blocks:

- Initialize Output Data
- Event Command
- Client Status request
- Client Status response
- Command Control
- Set new module IP address
- Get module IP address
- Warm-boot
- Cold-boot

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

Event Command Blocks (2000 to 2029)

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 to 2029	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.

Word	Description
5	This word contains the TCP service port the message will be interfaced. For example, to interface with a MBAP device, the word should contain a value of 502. To interface with a MNET device, a value of 2000 should be utilized. Any value from 0 to 65535 is permitted. A value of 502 will cause a MBAP formatted message to be generated. All other values will generate an encapsulated Modbus message.
6	This word contains the Modbus node address for the message. This field should have a value from 0 to 41.
7	This word contains the internal Modbus address in the module to use with the command. This word can contain a value from 0 to 4999.
8	This word contains the count parameter that determines the number of digital points or registers to associate with the command.
9	The parameter specifies the swap type for the data. This function is only valid for function codes 3 and 4.
10	This word contains the Modbus function code for the command.
11	This word contains the Modbus address in the slave device to be associated with the command.
12 to 41	Spare

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

Offset	Description	Length
0	Write Block ID	1
1	0=Fail, 1=Success	1
2 to 40	Spare	39
41	2000 to 2029	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 (16 commands for each queue).

Client Status Request Blocks (3000 to 3029)

Offset	Description	Length
0	3000 to 3029 (last digits indicate which client to consider)	1
1 to 41	Spare	40

Client Status Response

Offset	Description	Length
0	Write Block ID	1
1	3000 to 3029 number requested	1
2 to 11	Client status data	10
12 to 27	Command error list data for client	16
28 to 40	Reserved	13
41	3000 to 3029	1

Client Status Data

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

Command Control Blocks (5001 to 5016)

Command control blocks place commands in the command list into the command queue. The client has a command queue of up to 16 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. 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 5016	1
1	Client to Utilize	1
2	Command index	1
3	Command index	1
4	Command index	1
5	Command index	1
6	Command index	1
7	Command index	1
8	Command index	1
9	Command index	1
10	Command index	1
11	Command index	1
12	Command index	1
13	Command index	1
14	Command index	1
15	Command index	1
16	Command index	1
17	Command index	1
18 to 41	Spare	24

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 15 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 5016	1

Reset Module Status Block (9971)

This block allows the processor to reset all status values available from the module to the processor or through the PCB diagnostics menu. This block is triggered through the following data type and controller tag elements:

Data Type: MNETCRCONTROL

Name: MNETCRCONTROL

Description: Values used by program for data transfer operation between the module and the processor.

Members: Data Type Size: 164 byte(s)

Name	Data Type	Style	Description
BootTimer	TIMER		Timer used to clear both cold and warm bo
WarmBoot	BOOL	Decimal	Partial reset of the Module.
ColdBoot	BOOL	Decimal	Full reset of the Module.
ResetStatus	BOOL	Decimal	Resets module status
EventCmdTrigger	BOOL	Decimal	Initiates Event Command.
EventCmdPending	BOOL	Decimal	Allows Event Command.
ClientID	INT	Decimal	Client ID to poll a server with the Event Con
EventCmd	MNETCREVENTCMD		Holds Event Command configuration
CmdControl	MNETCRCMDCONTROL		Holds Command Control status
CmdControlTrigger	BOOL	Decimal	Initiates Command Control.
CmdControlPending	BOOL	Decimal	Allows CmdControlTriger.
IPAddress	MNETCRIPADDRESS		IP address statistics including triggers
WriteCmdBits	INT[30]	Decimal	Selects individual clients to activate its o

-	MNETCR	{...}	M
+	MNETCR.DATA	{...}	M
-	MNETCR.CONTROL	{...}	M
+	MNETCR.CONTROL.BootTimer	{...}	T
-	MNETCR.CONTROL.WarmBoot	0	Decimal B
-	MNETCR.CONTROL.ColdBoot	0	Decimal B
-	MNETCR.CONTROL.ResetStatus	0	Decimal B

Set Module IP Address Block (9990)

IP Set Request (Write Block)

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

IP Set Response (Read Block)

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

Get Module IP Address Block (9991)

IP Get Request (Write Block)

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

IP Get Response (Read Block)

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

Warm Boot Block (9998)

This block is equivalent to performing a software reset, and causes the module to exit the program, reload the configuration file, and then restart the program. The Warm Boot control block also initializes the application port(s) and status data, and resets all internal registers to zero.

Note: In some cases, the read section of the module database (transferred from module to processor) must keep its values after a reboot. To repopulate the module's registers with the last values the module sent to the processor, set the **INITIALIZE OUTPUT DATA** parameter in the module configuration to **YES**.

The following table describes the format of the control block.

Offset	Description	Length
0	9998	1
1 to 41	Spare	41

The module does not send a response block for this command.

Cold Boot Block (9999)

This block is equivalent to performing a hardware reset, and causes the module to restart in the same way as if the power was cycled. The Cold Boot control block also reloads the module's backplane and application port drivers, restarts the program, and resets all internal registers to zero.

Note: In some cases, the read section of the module database (transferred from module to processor) must keep its values after a reboot. To repopulate the module's registers with the last values the module sent to the processor, set the **INITIALIZE OUTPUT DATA** parameter in the module configuration to **YES**.

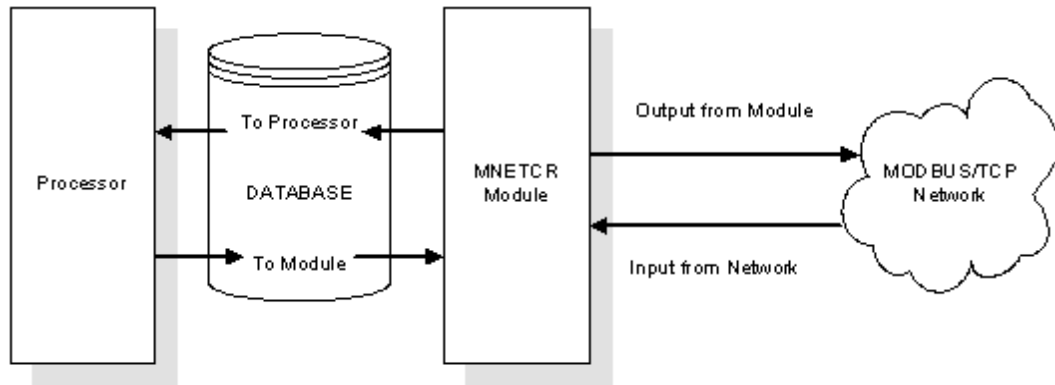
The following table describes the format of the control block.

Offset	Description	Length
0	9999	1
1 to 41	Spare	41

The module does not send a response block for this command.

5.3 Data Flow between MVI56E-MNETCR Module, Processor, and Network

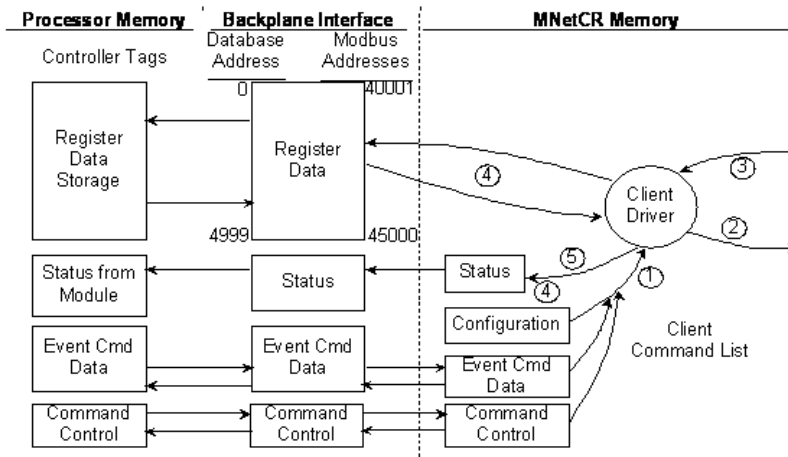
The following topics describe the flow of data between the two pieces of hardware (processor and MVI56E-MNETCR module) and other nodes on the Modbus TCP/IP network. The module contains up to 30 clients, which can generate either MBAP (Modbus API for network communications) or MNET requests dependent on the service port selected in the command.



The following topics discuss the operation of the client drivers.

5.3.1 Client Driver

In the client driver, the MVI56E-MNETCR module issues read or write commands to servers on the Modbus TCP/IP network using 30 simulated clients. These commands are user configured in the module via the Client Command List for each client received from the module's configuration file (MNET.CFG) or issued directly from the 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 associated table describe the flow of data into and out of the module.



Step	Description
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.

5.3.2 Client Command List

In order for the Client to function, the module's Client Command List must be defined in the *MNET Client x Commands* section of the configuration. This list contains up to 16 individual entries, with each entry containing the information required to construct a valid command. This includes the following:

- Command enable mode: (0) disabled or (1) continuous
- 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

5.3.3 Client Command Errors

You can use the configuration's *MNET Client x Command Error Pointer*, which is separately configured for each Client, to set the database offset register where all command error codes will be stored. This means that the first register refers to command 1 and so on.

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

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

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

Standard Modbus Exception Code Errors

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

Module Communication Error Codes

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

MNET Client Specific Errors

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

Command List Entry Errors

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

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

5.4 Ethernet Cable Specifications

The recommended cable is Category 5 or better. A Category 5 cable has four twisted pairs of wires, 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 116) for a diagram of how to configure Ethernet cable.

5.4.1 Ethernet Cable Configuration

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

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

5.4.2 Ethernet Performance

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

- Accessing the web interface (refreshing the page, downloading files, and so on) may affect MNETCR performance
- High Ethernet traffic may impact MNETCR performance (consider CIPconnect (page 63, page 22) for these applications and disconnect the module Ethernet port from the network).

5.5 Modbus Protocol Specification

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

5.5.1 Read Coil Status (Function Code 01)

Query

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

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

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

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

Response

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

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

Adr	Func	Byte Count	Data Coil Status 20 to 27	Data Coil Status 28 to 35	Data Coil Status 36 to 43	Data Coil Status 44 to 51	Data Coil Status 52 to 56	Error Check Field
11	01	05	CD	6B	B2	0E	1B	CRC

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

5.5.2 Read Input Status (Function Code 02)

Query

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

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

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

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

Response

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

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

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

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

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

5.5.3 Read Holding Registers (Function Code 03)

Query

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

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

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

Response

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

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

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

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

5.5.4 Read Input Registers (Function Code 04)

Query

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

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

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

Response

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

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

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

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

5.5.5 Force Single Coil (Function Code 05)

Query

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

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

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

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

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

Response

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

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

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

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

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

5.5.6 Preset Single Register (Function Code 06)

Query

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

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

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

Response

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

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

5.5.7 Diagnostics (Function Code 08)

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

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

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

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

Sub-function codes supported

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

00 Return Query Data

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

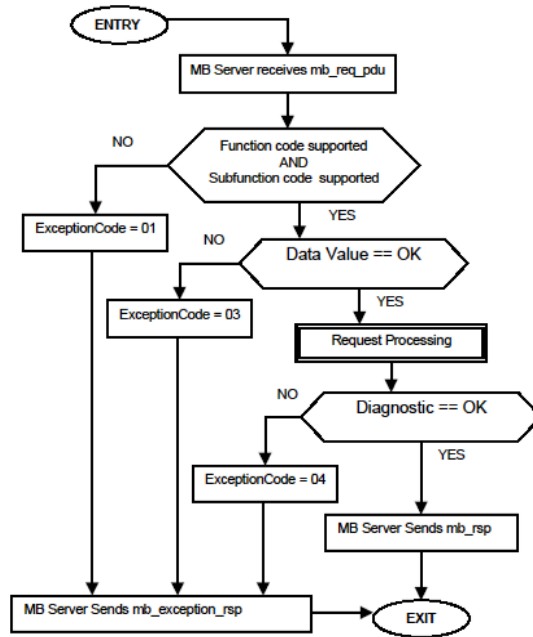
Sub-function	Data Field (Request)	Data Field (Response)
00 00	Any	Echo Request Data

Example and state diagram

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

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

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



5.5.8 Force Multiple Coils (Function Code 15)

Query

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

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

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

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

Response

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

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

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

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

5.5.9 Preset Multiple Registers (Function Code 16)

Query

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

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

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

Response

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

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

5.5.10 Modbus Exception Responses

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

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

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

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

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

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

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

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

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

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

Modbus Exception Codes

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

Code	Name	Meaning
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 Optional Add-On Instruction Rung Import

5.6.1 Before You Begin

- Make sure that you have installed RSLogix 5000 version 16 (or later)
- Download the Optional Add-On file *MVI56EMNETCR_Optional_Rung_v1_0.L5X* from the module's web page and copy it to a folder in your PC

5.6.2 Overview

The Optional Add-On Instruction Rung Import contains optional logic for MVI56E-MNETCR 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.

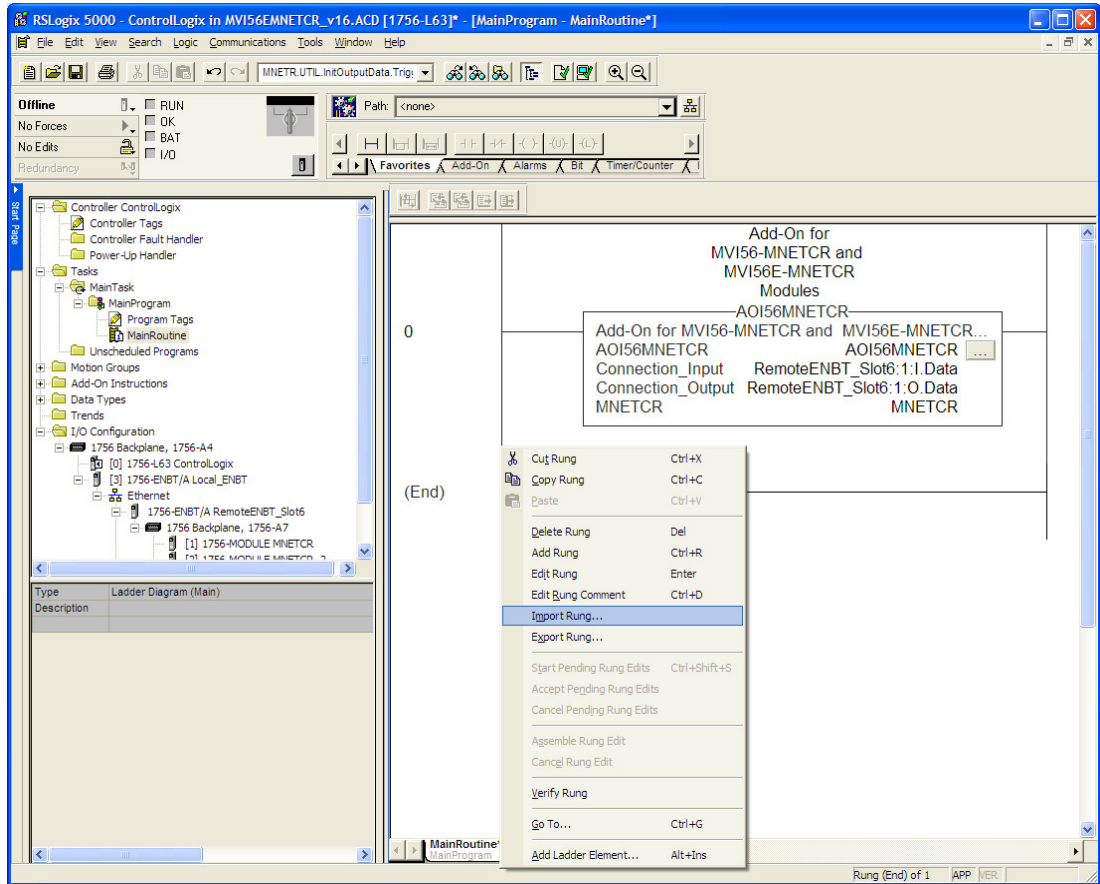
Note: This is an optional feature. You can perform the same task through PCB (ProSoft Configuration Builder). Even if your PC is in a different network group you can still access the module through PCB by setting a temporary IP address.

- **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.

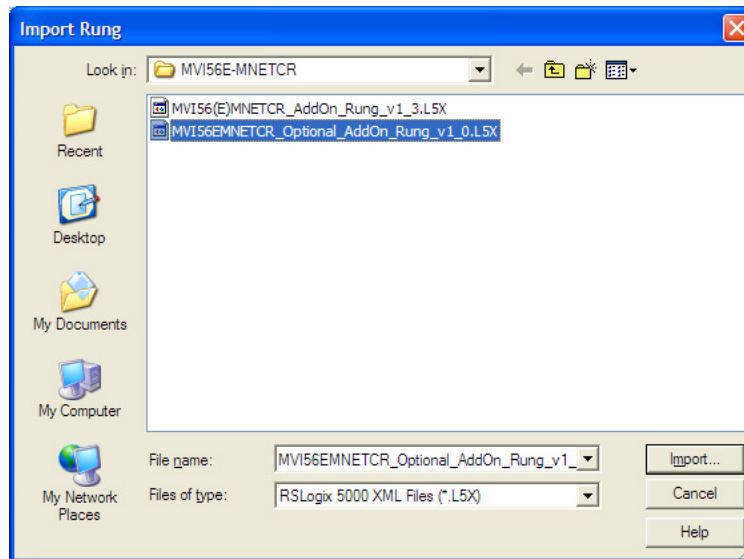
Important: The Optional Add-On Instruction only supports the two features listed above. You must use the sample ladder logic for all other features including backplane transfer of Modbus TCP/IP data.

5.6.3 Installing the Rung Import with Utility Add-On Instruction

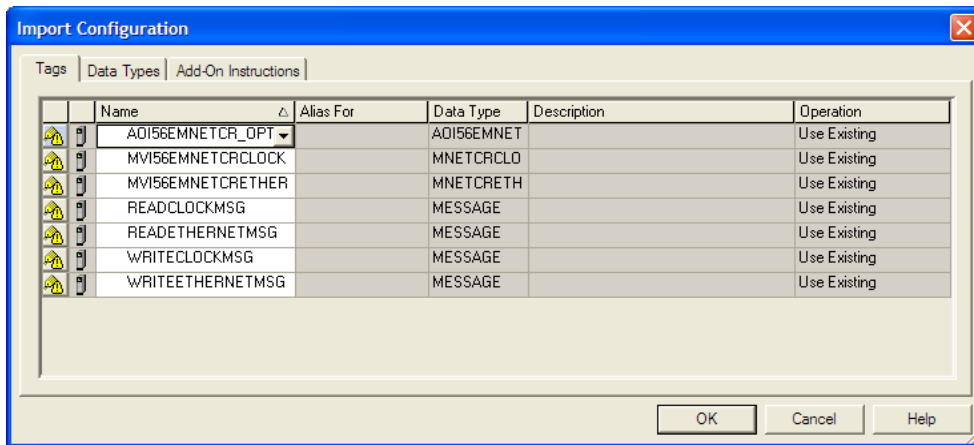
- 1 Right-click on an empty rung in the main routine of your existing ladder logic and choose **IMPORT RUNG...**



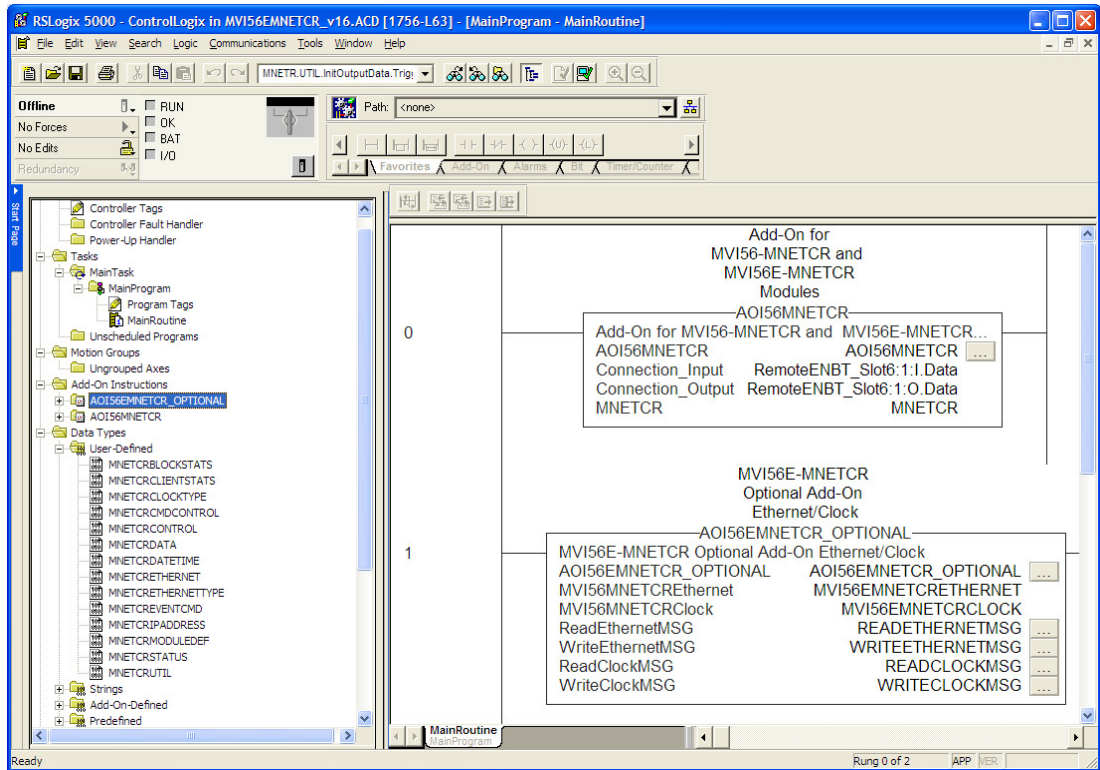
- 2 Navigate to the folder where you saved MVI56EMNETCR_Optional_AddOn_Rung_<version #>.L5X and select the file.



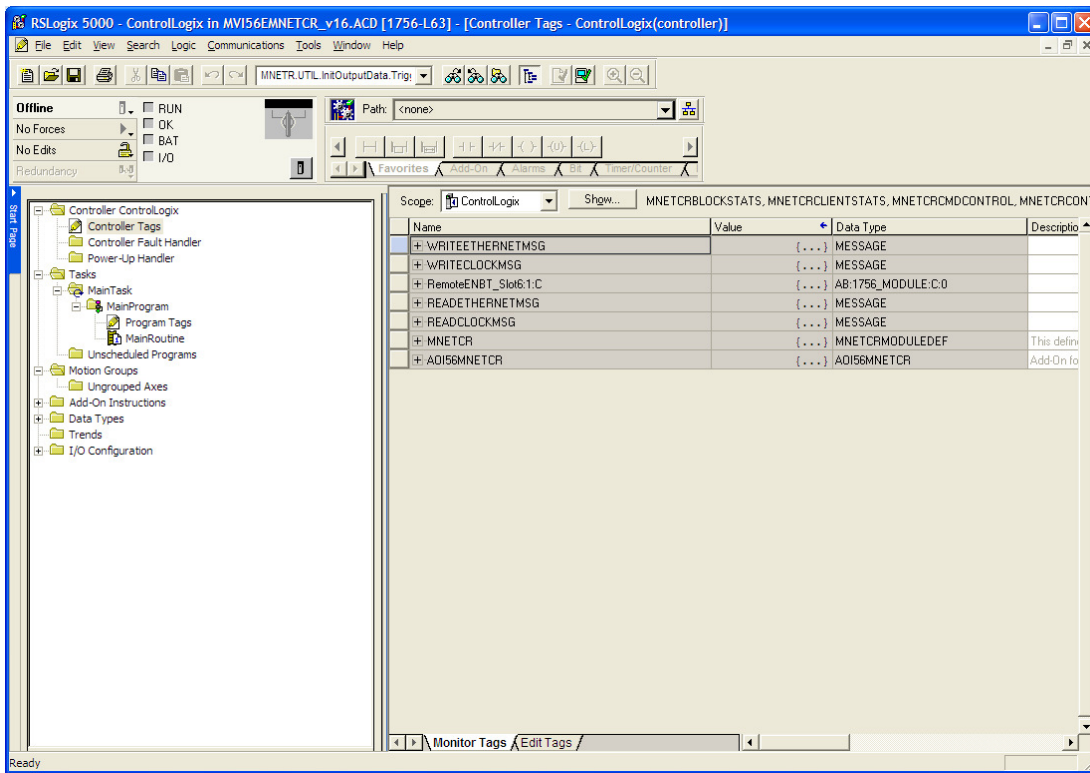
In the **IMPORT CONFIGURATION** window, click **OK**.



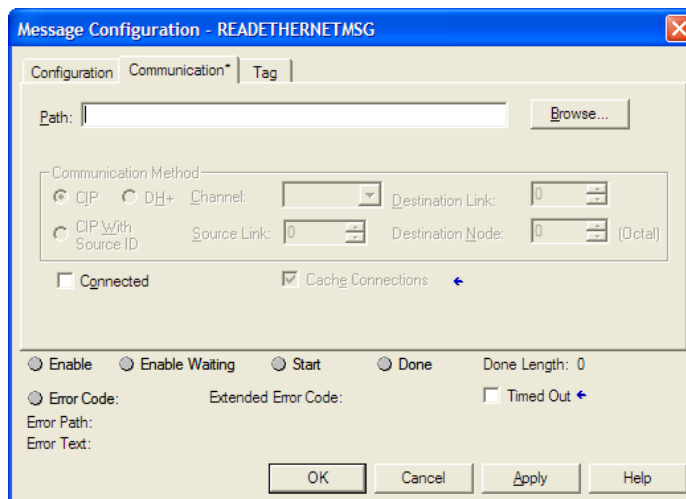
The Add-On Instruction will be now visible in the ladder logic. Observe that the procedure has also imported data types and controller tags associated to the Add-On Instruction.



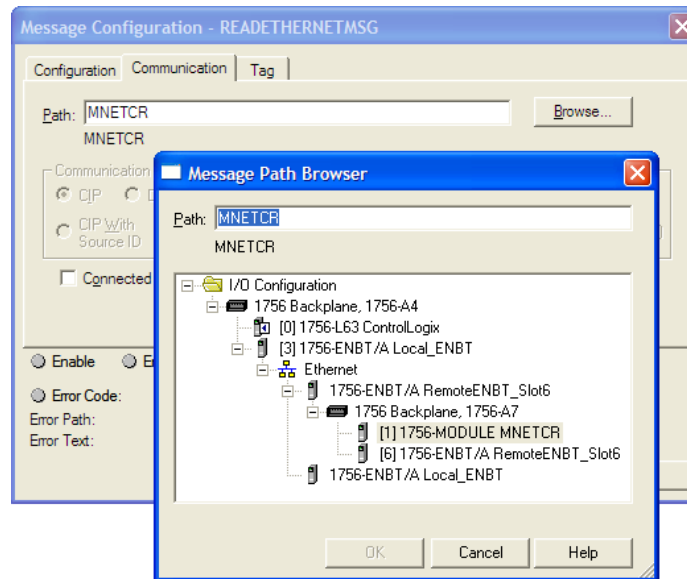
You will notice that new tags have been imported: four **MESSAGE** tags, **MVI56MNETCCLOCK** and **MVI56MNETCETHERNET** tags.



- 3 In the Add-On Instruction, click the [...] button next to each **MSG** tag to open the **MESSAGE CONFIGURATION TAG**.
- 4 Click the **COMMUNICATION** tab and click the **BROWSE** button as follows.



5 Select the module to configure the message path.



5.6.4 Reading the Ethernet Settings from the Module

Expand the **MVI56EMNETCRETHERNET** controller tag and move a value of 1 to **MVI56EMNETCRETHERNET.READ**.

[-] MVI56EMNETCRETHERNET	{...}
MVI56EMNETCRETHERNET.Read	1
MVI56EMNETCRETHERNET.Write	0
[-] MVI56EMNETCRETHERNET.Config	{...}
[-] MVI56EMNETCRETHERNET.Config.IP	{...}
+ MVI56EMNETCRETHERNET.Config.IP[0]	0
+ MVI56EMNETCRETHERNET.Config.IP[1]	0
+ MVI56EMNETCRETHERNET.Config.IP[2]	0
+ MVI56EMNETCRETHERNET.Config.IP[3]	0
[-] MVI56EMNETCRETHERNET.Config.Netmask	{...}
+ MVI56EMNETCRETHERNET.Config.Netmask[0]	0
+ MVI56EMNETCRETHERNET.Config.Netmask[1]	0
+ MVI56EMNETCRETHERNET.Config.Netmask[2]	0
+ MVI56EMNETCRETHERNET.Config.Netmask[3]	0
[-] MVI56EMNETCRETHERNET.Config.Gateway	{...}
+ MVI56EMNETCRETHERNET.Config.Gateway[0]	0
+ MVI56EMNETCRETHERNET.Config.Gateway[1]	0
+ MVI56EMNETCRETHERNET.Config.Gateway[2]	0
+ MVI56EMNETCRETHERNET.Config.Gateway[3]	0

The bit will be automatically reset and the current Ethernet settings will be copied to **MVI56EMNETCRETHERNET** controller tag as follows.

[-] MVI56EMNETCRETHERNET	{...}
MVI56EMNETCRETHERNET.Read	0
MVI56EMNETCRETHERNET.Write	0
[-] MVI56EMNETCRETHERNET.Config	{...}
[-] MVI56EMNETCRETHERNET.Config.IP	{...}
+ MVI56EMNETCRETHERNET.Config.IP[0]	105
+ MVI56EMNETCRETHERNET.Config.IP[1]	102
+ MVI56EMNETCRETHERNET.Config.IP[2]	0
+ MVI56EMNETCRETHERNET.Config.IP[3]	12
[-] MVI56EMNETCRETHERNET.Config.Netmask	{...}
+ MVI56EMNETCRETHERNET.Config.Netmask[0]	255
+ MVI56EMNETCRETHERNET.Config.Netmask[1]	255
+ MVI56EMNETCRETHERNET.Config.Netmask[2]	255
+ MVI56EMNETCRETHERNET.Config.Netmask[3]	0
[-] MVI56EMNETCRETHERNET.Config.Gateway	{...}
+ MVI56EMNETCRETHERNET.Config.Gateway[0]	192
+ MVI56EMNETCRETHERNET.Config.Gateway[1]	168
+ MVI56EMNETCRETHERNET.Config.Gateway[2]	0
+ MVI56EMNETCRETHERNET.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

5.6.5 Writing the Ethernet Settings to the Module

Expand the **MVI56EMNETCRETHERNET** controller tag.

Set the new Ethernet configuration in **MVI56EMNETCRETHERNET.CONFIG**

Move a value of 1 to **MVI56EMNETCRETHERNET.WRITE**

[-] MVI56EMNETCRETHERNET	{...}
[-] MVI56EMNETCRETHERNET.Read	0
[-] MVI56EMNETCRETHERNET.Write	1
[-] MVI56EMNETCRETHERNET.Config	{...}
[-] MVI56EMNETCRETHERNET.Config.IP	{...}
+ MVI56EMNETCRETHERNET.Config.IP[0]	0
+ MVI56EMNETCRETHERNET.Config.IP[1]	0
+ MVI56EMNETCRETHERNET.Config.IP[2]	0
+ MVI56EMNETCRETHERNET.Config.IP[3]	0
[-] MVI56EMNETCRETHERNET.Config.Netmask	{...}
+ MVI56EMNETCRETHERNET.Config.Netmask[0]	0
+ MVI56EMNETCRETHERNET.Config.Netmask[1]	0
+ MVI56EMNETCRETHERNET.Config.Netmask[2]	0
+ MVI56EMNETCRETHERNET.Config.Netmask[3]	0
[-] MVI56EMNETCRETHERNET.Config.Gateway	{...}
+ MVI56EMNETCRETHERNET.Config.Gateway[0]	0
+ MVI56EMNETCRETHERNET.Config.Gateway[1]	0
+ MVI56EMNETCRETHERNET.Config.Gateway[2]	0
+ MVI56EMNETCRETHERNET.Config.Gateway[3]	0

After the message is executed, the **MVI56EMNETCRETHERNET.WRITE** bit resets to 0.

[-] MVI56EMNETCRETHERNET	{...}
[-] MVI56EMNETCRETHERNET.Read	0
[-] MVI56EMNETCRETHERNET.Write	0
[-] MVI56EMNETCRETHERNET.Config	{...}
[-] MVI56EMNETCRETHERNET.Config.IP	{...}
+ MVI56EMNETCRETHERNET.Config.IP[0]	105
+ MVI56EMNETCRETHERNET.Config.IP[1]	102
+ MVI56EMNETCRETHERNET.Config.IP[2]	0
+ MVI56EMNETCRETHERNET.Config.IP[3]	12
[-] MVI56EMNETCRETHERNET.Config.Netmask	{...}
+ MVI56EMNETCRETHERNET.Config.Netmask[0]	255
+ MVI56EMNETCRETHERNET.Config.Netmask[1]	255
+ MVI56EMNETCRETHERNET.Config.Netmask[2]	255
+ MVI56EMNETCRETHERNET.Config.Netmask[3]	0
[-] MVI56EMNETCRETHERNET.Config.Gateway	{...}
+ MVI56EMNETCRETHERNET.Config.Gateway[0]	192
+ MVI56EMNETCRETHERNET.Config.Gateway[1]	168
+ MVI56EMNETCRETHERNET.Config.Gateway[2]	0
+ MVI56EMNETCRETHERNET.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

5.6.6 Reading the Clock Value from the Module

Expand the **MVI56EMNETCRLOCK** controller tag and move a value of 1 to **MVI56EMNETCRLOCK.READ**

[-] MVI56EMNETCRLOCK	{...}
[-] MVI56EMNETCRLOCK.Read	1
[-] MVI56EMNETCRLOCK.Write	0
[-] MVI56EMNETCRLOCK.Config	{...}
[+] MVI56EMNETCRLOCK.Config.Year	0
[+] MVI56EMNETCRLOCK.Config.Month	0
[+] MVI56EMNETCRLOCK.Config.Day	0
[+] MVI56EMNETCRLOCK.Config.Hour	0
[+] MVI56EMNETCRLOCK.Config.Minute	0
[+] MVI56EMNETCRLOCK.Config.Seconds	0

The bit will be automatically reset and the current clock value will be copied to **MVI56EMNETCRLOCK.CONFIG** controller tag as follows.

[-] MVI56EMNETCRLOCK	{...}
[-] MVI56EMNETCRLOCK.Read	0
[-] MVI56EMNETCRLOCK.Write	0
[-] MVI56EMNETCRLOCK.Config	{...}
[+] MVI56EMNETCRLOCK.Config.Year	2009
[+] MVI56EMNETCRLOCK.Config.Month	11
[+] MVI56EMNETCRLOCK.Config.Day	18
[+] MVI56EMNETCRLOCK.Config.Hour	10
[+] MVI56EMNETCRLOCK.Config.Minute	21
[+] MVI56EMNETCRLOCK.Config.Seconds	45

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 **MVI56EMNETCRLOCK** controller tag.
 Set the new Clock value in **MVI56EMNETCRLOCK.CONFIG**
 Move a value of 1 to **MVI56EMNETCRLOCK.WRITE**

[-] MVI56EMNETCRLOCK	{...}
[-] MVI56EMNETCRLOCK.Read	0
[-] MVI56EMNETCRLOCK.Write	1
[-] MVI56EMNETCRLOCK.Config	{...}
[+] MVI56EMNETCRLOCK.Config.Year	2009
[+] MVI56EMNETCRLOCK.Config.Month	11
[+] MVI56EMNETCRLOCK.Config.Day	18
[+] MVI56EMNETCRLOCK.Config.Hour	10
[+] MVI56EMNETCRLOCK.Config.Minute	21
[+] MVI56EMNETCRLOCK.Config.Seconds	45

The bit will be automatically reset to 0.

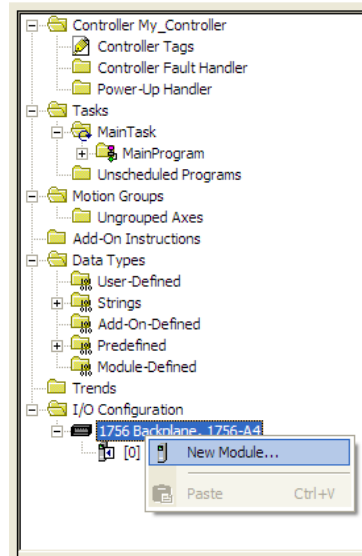
[-] MVI56EMNETCRLOCK	{...}
[-] MVI56EMNETCRLOCK.Read	0
[-] MVI56EMNETCRLOCK.Write	0
[-] MVI56EMNETCRLOCK.Config	{...}
[+] MVI56EMNETCRLOCK.Config.Year	2009
[+] MVI56EMNETCRLOCK.Config.Month	11
[+] MVI56EMNETCRLOCK.Config.Day	18
[+] MVI56EMNETCRLOCK.Config.Hour	10
[+] MVI56EMNETCRLOCK.Config.Minute	21
[+] MVI56EMNETCRLOCK.Config.Seconds	45

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

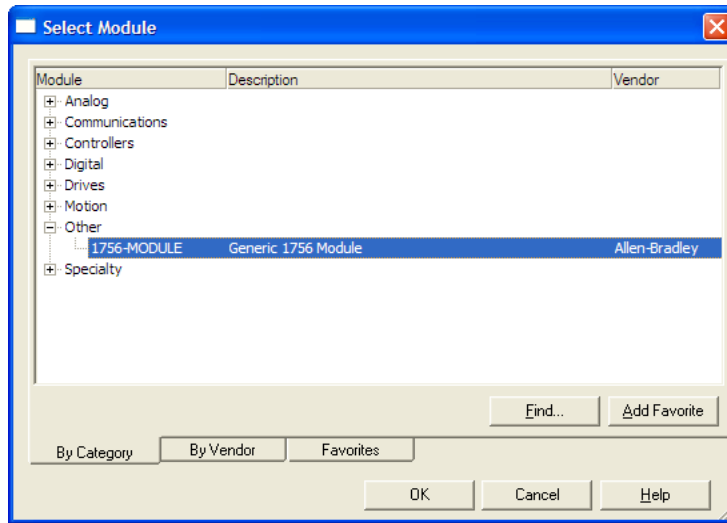
[-] WRITELOCKMSG	{...}
[+] WRITELOCKMSG.Flags	16#0200
[-] WRITELOCKMSG.EW	0
[-] WRITELOCKMSG.ER	0
[-] WRITELOCKMSG.DN	0
[-] WRITELOCKMSG.ST	0
[-] WRITELOCKMSG.EN	0
[-] WRITELOCKMSG.TO	0
[-] WRITELOCKMSG.EN_CC	1
[+] WRITELOCKMSG.ERR	16#0000
[+] WRITELOCKMSG.EXERR	16#0000_0000
[+] WRITELOCKMSG.ERR_SRC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

5.7 Adding the Module to an Existing Project

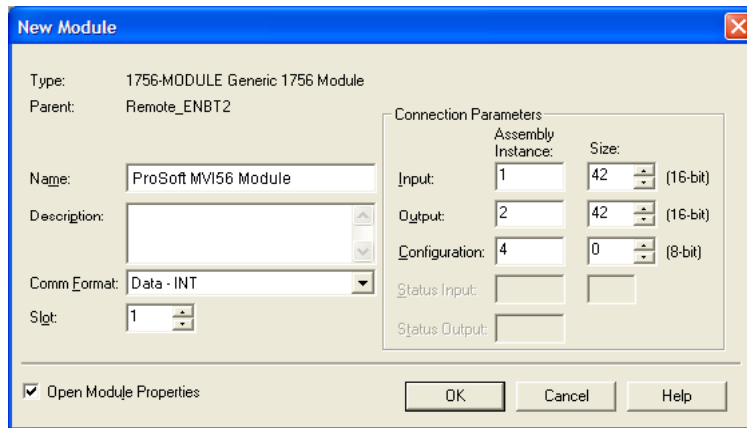
- 1 **Add the MVI56E-MNETCR module to the project.** Select the **I/O CONFIGURATION** folder in the **CONTROLLER ORGANIZATION** window, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW MODULE**.



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



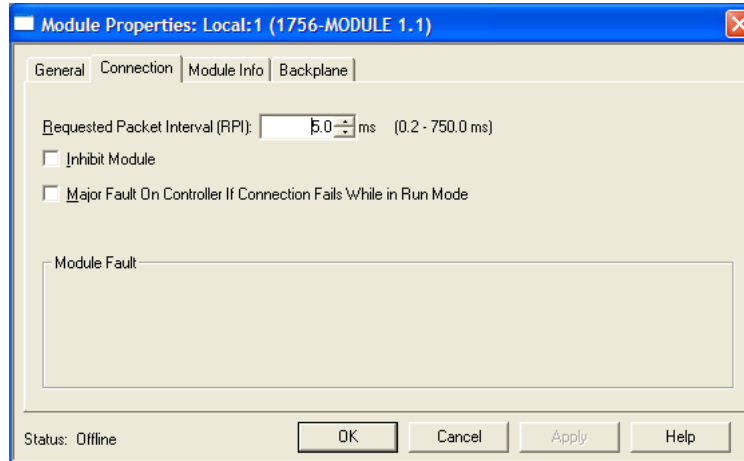
Select the **1756-MODULE** (Generic 1756 Module) from the list and click **OK**. This action opens the **NEW MODULE** dialog box.



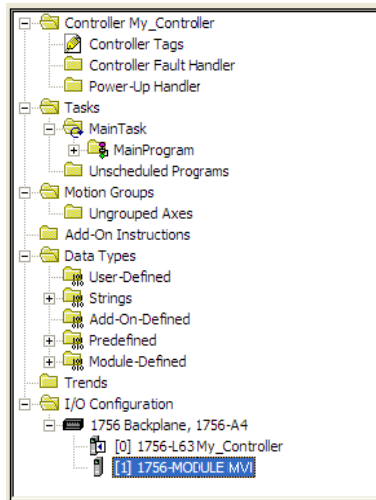
Parameter	Value
Name	Enter a module identification string. The recommended value is MNETCR.
Description	Enter a description for the module. Example: Modbus TCP/IP Multi Client Enhanced Communications Module for Remote Chassis.
Comm Format	Select DATA-INT (Very Important)
Slot	Enter the slot number in the rack where the MVI56E-MNETCR module will be installed.
Input Assembly Instance	1
Input Size	42
Output Assembly Instance	2
Output Size	42
Configuration Assembly Instance	4
Configuration Size	0

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

- 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.



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



- Copy the Controller Tags from the sample program.
- Copy the User Defined Data Types from the sample program.
- Copy the Ladder Rungs from the sample program.
- Save and Download (page 43) the new application to the controller and place the processor in run mode.

5.8 Using the Sample Program - RSLogix 5000 Version 15 and earlier

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

6 Support, Service & Warranty

In This Chapter

- ❖ Contacting Technical Support 147
- ❖ Return Material Authorization (RMA) Policies and Conditions..... 149
- ❖ LIMITED WARRANTY..... 151

Contacting Technical Support

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

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

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

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

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

Internet	Web Site: www.prosoft-technology.com/support E-mail address: support@prosoft-technology.com
Asia Pacific (location in Malaysia)	Tel: +603.7724.2080, E-mail: asiapc@prosoft-technology.com Languages spoken include: Chinese, English
Asia Pacific (location in China)	Tel: +86.21.5187.7337 x888, E-mail: asiapc@prosoft-technology.com Languages spoken include: Chinese, English
Europe (location in Toulouse, France)	Tel: +33 (0) 5.34.36.87.20, E-mail: support.EMEA@prosoft-technology.com Languages spoken include: French, English
Europe (location in Dubai, UAE)	Tel: +971-4-214-6911, E-mail: mea@prosoft-technology.com Languages spoken include: English, Hindi
North America (location in California)	Tel: +1.661.716.5100, E-mail: support@prosoft-technology.com Languages spoken include: English, Spanish
Latin America (Oficina Regional)	Tel: +1-281-2989109, E-Mail: latinam@prosoft-technology.com Languages spoken include: Spanish, English
Latin America (location in Puebla, Mexico)	Tel: +52-222-3-99-6565, E-mail: soporte@prosoft-technology.com Languages spoken include: Spanish
Brasil (location in Sao Paulo)	Tel: +55-11-5083-3776, E-mail: brasil@prosoft-technology.com Languages spoken include: Portuguese, English

6.1 Return Material Authorization (RMA) Policies and Conditions

The following Return Material Authorization (RMA) Policies and Conditions (collectively, "RMA Policies") apply to any returned product. These RMA Policies are subject to change by ProSoft Technology, Inc., without notice. For warranty information, see Limited Warranty (page 151). In the event of any inconsistency between the RMA Policies and the Warranty, the Warranty shall govern.

6.1.1 Returning Any Product

- a) In order to return a Product for repair, exchange, or otherwise, the Customer must obtain a Return Material Authorization (RMA) number from ProSoft Technology and comply with ProSoft Technology shipping instructions.
- b) In the event that the Customer experiences a problem with the Product for any reason, Customer should contact ProSoft Technical Support at one of the telephone numbers listed above (page 147). A Technical Support Engineer will request that you perform several tests in an attempt to isolate the problem. If after completing these tests, the Product is found to be the source of the problem, we will issue an RMA.
- c) All returned Products must be shipped freight prepaid, in the original shipping container or equivalent, to the location specified by ProSoft Technology, and be accompanied by proof of purchase and receipt date. The RMA number is to be prominently marked on the outside of the shipping box. Customer agrees to insure the Product or assume the risk of loss or damage in transit. Products shipped to ProSoft Technology using a shipment method other than that specified by ProSoft Technology, or shipped without an RMA number will be returned to the Customer, freight collect. Contact ProSoft Technical Support for further information.
- d) A 10% restocking fee applies to all warranty credit returns, whereby a Customer has an application change, ordered too many, does not need, etc. Returns for credit require that all accessory parts included in the original box (i.e.; antennas, cables) be returned. Failure to return these items will result in a deduction from the total credit due for each missing item.

6.1.2 Returning Units Under Warranty

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

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

6.1.3 Returning Units Out of Warranty

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

The following is a list of non-repairable units:

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

6.2 LIMITED WARRANTY

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

6.2.1 What Is Covered By This Warranty

- a) *Warranty On New Products:* ProSoft warrants, to the original purchaser, that the Product that is the subject of the sale will (1) conform to and perform in accordance with published specifications prepared, approved and issued by ProSoft, and (2) will be free from defects in material or workmanship; provided these warranties only cover Product that is sold as new. This Warranty expires three (3) years from the date of shipment for Product purchased **on or after** January 1st, 2008, or one (1) year from the date of shipment for Product purchased **before** January 1st, 2008 (the "Warranty Period"). If the Customer discovers within the Warranty Period a failure of the Product to conform to specifications, or a defect in material or workmanship of the Product, the Customer must promptly notify ProSoft by fax, email or telephone. In no event may that notification be received by ProSoft later than 39 months from date of original shipment. Within a reasonable time after notification, ProSoft will correct any failure of the Product to conform to specifications or any defect in material or workmanship of the Product, with either new or remanufactured replacement parts. ProSoft reserves the right, and at its sole discretion, may replace unrepairable units with new or remanufactured equipment. All replacement units will be covered under warranty for the 3 year period commencing from the date of original equipment purchase, not the date of shipment of the replacement unit. Such repair, including both parts and labor, will be performed at ProSoft's expense. All warranty service will be performed at service centers designated by ProSoft.
- b) *Warranty On Services:* Materials and labor performed by ProSoft to repair a verified malfunction or defect are warranted in the terms specified above for new Product, provided said warranty will be for the period remaining on the original new equipment warranty or, if the original warranty is no longer in effect, for a period of 90 days from the date of repair.

6.2.2 What Is Not Covered By This Warranty

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

6.2.3 Disclaimer Regarding High Risk Activities

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

6.2.4 Intellectual Property Indemnity

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

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

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

6.2.5 Disclaimer of all Other Warranties

The Warranty set forth in What Is Covered By This Warranty (page 151) are in lieu of all other warranties, express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

6.2.6 Limitation of Remedies **

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

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

6.2.7 Time Limit for Bringing Suit

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

6.2.8 No Other Warranties

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

6.2.9 Allocation of Risks

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

6.2.10 Controlling Law and Severability

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

Index

O

00 Return Query Data • 125

A

About the MODBUS/TCP Protocol • 97
Adding Multiple Modules (Optional) • 37
Adding the Module to an Existing Project • 142
Adjusting the Input and Output Array Sizes (Optional) • 42
Allocation of Risks • 154
ARP Timeout • 53

B

Backplane Control Object (MNETCRUTIL) • 72, 77
Backplane Data Transfer • 98
Backplane Status • 88
Battery Life Advisory • 3
Before You Begin • 132
Block 9990
 Set Module IP Address • 99, 108
Block 9991
 Get Module IP Address • 109
Block 9998
 Warm Boot • 99, 109
Block 9999
 Cold Boot • 110
Block Request from Processor to Module • 100
Block Response from Module to Processor • 100
Block Statistics Object (MNETCRBLOCKSTATS) • 75, 76

C

Clearing a Fault Condition • 83
Client Command Errors • 89, 113
Client Command List • 113
Client Driver • 76, 91, 112
Client Statistics Object (MNETCRCLIENTSTATS) • 75, 76
Client Status Data • 76, 91, 105
Client Status Request Blocks (3000 to 3029) • 99, 105
Client Status Response • 105
Command Control Blocks (5001 to 5016) • 99, 106
Command Control Data Object (MNETCCONTROL) • 72, 74
Command Control Data Object (MNETCRCMDCONTROL) • 74, 75
Command Entry Formats • 55
Command Error Delay • 54
Command Error Pointer • 52
Command List • 89
Command List Entry Errors • 115

Command List Overview • 54
Command Status • 89
Commands Supported by the Module • 55
Comment • 59
Config • 88, 89
Configuring Module Parameters • 49
Configuring the MVI56E-MNETCR Module • 45
Connecting to the Module's Web Page • 24
Connecting Your PC to the ControlLogix Processor • 43
Connecting Your PC to the Module • 17
Contacting Technical Support • 147, 149
Controlling Law and Severability • 155
Create the Module - Local Rack • 26, 31
Create the Module - Remote Rack • 29
Create the Remote Network • 27, 30
Creating a New RSLogix 5000 Project • 26

D

Data Flow between MVI56E-MNETCR Module, Processor, and Network • 111
Diagnostics (Function Code 08) • 125
Diagnostics and Troubleshooting • 79, 80
Disclaimer of all Other Warranties • 153
Disclaimer Regarding High Risk Activities • 152
Downloading the Project to the Module • 62
Downloading the Sample Program to the Processor • 43, 144
Duplex/Speed Code • 52

E

Enable • 56
Error/Status Pointer • 50, 52, 91
Ethernet Cable Configuration • 116
Ethernet Cable Specifications • 116
Ethernet Configuration • 61
Ethernet LED Indicators • 81
Ethernet Performance • 117
Event Command Blocks (2000 to 2029) • 99, 103
Event Command Data Object (MNETCREVENTCMD) • 74
Example 1
 Local Rack Application • 65
Example 2
 Remote Rack Application • 68
Example and state diagram • 125

F

Failure Flag Count • 51
Float Flag • 53
Float Offset • 53
Float Start • 53
Force Multiple Coils (Function Code 15) • 127
Force Single Coil (Function Code 05) • 123
Functional Overview • 97
Functional Specifications • 95

G

General Specifications • 94

H

Hardware MAC Address • 60
Hardware Specifications • 96
How to Contact Us • 2

I

Import Add-On Instruction • 33
Important Safety Information - MVI56E Modules • 3
Initialize Output Data • 51, 102
Installing ProSoft Configuration Builder • 16
Installing the Configuration Tools • 16
Installing the Module in the Rack • 14
Installing the Rung Import with Utility Add-On Instruction • 133
Intellectual Property Indemnity • 153
Internal Address • 56
IP Address • 60
IP Address Object (MNETCRIPADDRESS) • 74, 75

L

Ladder Logic • 71
LED Status Indicators • 80
Limitation of Remedies ** • 154
LIMITED WARRANTY • 149, 151

M

MB Address in Device • 59
MBAP Port Override • 54
Minimum Command Delay • 52
MNET Client Specific Errors • 114
MNET Client x • 52
MNET Client x Commands • 54
Modbus Exception Codes • 130
Modbus Exception Responses • 129
Modbus Function • 58
Modbus Protocol Specification • 118
Modbus TCP/IP Client (Master) • 95
Module • 50
Module Communication Error Codes • 114
Module Data Object (MNETCRMODULEDEF) • 72
Monitoring Backplane Information • 88
Monitoring Database Information • 90
Monitoring MNET Client Information • 89
Monitoring Module Information • 87

N

NIC Status • 88
No Other Warranties • 154
Node IP Address • 58
Non-Scrolling LED Status Indicators • 81
Normal Data Transfers • 100

O

Overview • 132

P

Package Contents • 12
Pinouts • 96, 116
Poll Interval • 57
Preset Multiple Registers (Function Code 16) • 128
Preset Single Register (Function Code 06) • 124
Printing a Configuration File • 49
Product Specifications • 94
ProSoft Technology® Product Documentation • 2

R

Read Coil Status (Function Code 01) • 118
Read Holding Registers (Function Code 03) • 121
Read Input Registers (Function Code 04) • 122
Read Input Status (Function Code 02) • 120
Read Register Count • 50, 73
Read Register Start • 50
Reading Status Data from the Module • 91
Reading the Clock Value from the Module • 140
Reading the Ethernet Settings from the Module • 138
Reference • 93
Reg Count • 57
Renaming PCB Objects • 49
Response Timeout • 52
Retry Count • 53
Return Material Authorization (RMA) Policies and Conditions • 149
Returning Any Product • 149
Returning Units Out of Warranty • 150
Returning Units Under Warranty • 150

S

Scrolling LED Status Indicators • 80
Service Port • 58
Setting Jumpers • 13
Setting Temporary IP Address • 18, 24
Setting Up the Project • 47
Slave Address • 58
Special Function Blocks • 99, 102
Standard Modbus Exception Code Errors • 114
Start Here • 9
Static ARP • 88
Static ARP Table • 60, 88
Status • 89
Status Data Block Structure • 101
Status Object (MNETCRSTATUS) • 72, 75
Status Read Data Block IDs 0 and -1 • 101
Sub-function codes supported • 125
Support, Service & Warranty • 147
Swap Code • 57
System Requirements • 11

T

The Diagnostics Menu • 87
Time Limit for Bringing Suit • 154
Troubleshooting • 82

U

- Uploading the Add-On Instruction from the Module • 25, 34
- User Data Object (MNETCRDATA) • 72, 73
- Using CIPconnect to Connect to the Module • 22, 85, 117
- Using CIPconnect® to Connect to the Module • 63, 85, 117
- Using ProSoft Configuration Builder Software • 46
- Using the Diagnostics Menu in ProSoft Configuration Builder • 84
- Using the Optional Add-On Instruction Rung Import • 132
- Using the Sample Program - RSLogix 5000 Version 15 and earlier • 25, 145

V

- Version • 87

W

- What Is Covered By This Warranty • 151, 153
- What Is Not Covered By This Warranty • 152
- What's New? • 10
- Write Register Count • 51, 73
- Write Register Start • 51
- Writing the Clock Value to the Module • 141
- Writing the Ethernet Settings to the Module • 139

Y

- Your Feedback Please • 2