Where Automation Connects.

MVI69-MNETC

CompactLogix or MicroLogix Platform

Modbus TCP/IP Multi-Client Communication Module

November 11, 2010

USER MANUAL

Your Feedback Please

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

How to Contact Us

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

MVI69-MNETC User Manual
November 11, 2010

ProSoft Technology® Product Documentation

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

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

North America: +1.661.716.5100

Asia Pacific: +603.7724.2080

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

Latin America: +1.281.298.9109

Important Installation Instructions

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

A WARNING - EXPLOSION HAZARD - SUBSTITUTION OF COMPONENTS MAY IMPAIR SUITABILITY FOR CLASS I, DIV. 2;

B WARNING - EXPLOSION HAZARD - WHEN IN HAZARDOUS LOCATIONS, TURN OFF POWER BEFORE REPLACING OR WIRING MODULES

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

D THIS DEVICE SHALL BE POWERED BY CLASS 2 OUTPUTS ONLY.

MVI (Multi Vendor Interface) Modules

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

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

Warnings

North America Warnings

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 non-hazardous.

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

ATEX Warnings and Conditions of Safe Usage

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

A Warning - Explosion Hazard - When in hazardous locations, turn off power before replacing or wiring modules.

B Warning - Explosion Hazard - Do not disconnect equipment unless power has been switched off or the area is known to be non-hazardous.

C These products are intended to be mounted in an IP54 enclosure. The devices shall provide external means to prevent the rated voltage being exceeded by transient disturbances of more than 40%. This device must be used only with ATEX certified backplanes.

D DO NOT OPEN WHEN ENERGIZED.

Warning: This module is not hot-swappable! Always remove power from the rack before inserting or removing this module, or damage may result to the module, the processor, or other connected devices.

Battery Life Advisory

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

Note: The battery is not user replaceable.

Markings

Electrical Ratings

- Backplane Current Load: 800 mA @ 5.1 Vdc
- Power Supply Distance Rating: 2
- Operating Temperature: 0°C to 60°C (32°F to 140°F)
- Storage Temperature: -40°C to 85°C (-40°F to 185°F)
- Relative Humidity: 5% to 95% (without condensation)
- All phase conductor sizes must be at least 1.3 mm(squared) and all earth ground conductors must be at least 4mm(squared).

Label Markings

<cULus>

E193122

Class I Div 2

Groups A,B,C,D T6

-30°C <= Ta <= 60°C

<ATEX>

II 3 G

Ex nA IIC T6 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.

Agency Approvals and Certifications

Agency	Applicable Standard(s)
ATEX	EN 60079-0:2006, EN 60079-15:2005
DNV	DET NORSKE VERITAS Test 2.4
CE	EN61000-6-4:2007
CB Safety	CA/10533/CSA, IEC 61010-1 Ed. 2, CB 243333-2056722 (2090408)
GOST-R	EN 61010
CSA	61010
cULus	UL508, UL1604, CSA 22.2 No 142 & 213

ME06

E193122

Contents

Guide to the MVI69-MNETC User Manual

Function		Section to Read	Details
Introduction (Must Do)	→	Start Here (page 13)	This section introduces the customer to the module. Included are: package contents, system requirements, hardware installation, and basic configuration.
Diagnostic and Troubleshooting	→	Diagnostics and Troubleshooting (page 81)	This section describes Diagnostic and Troubleshooting procedures.
Reference Product Specifications Functional Overview	→	Reference (page 99) Product Specifications (page 99) Functional Overview (page 102)	These sections contain general references associated with this product, Specifications, and the Functional Overview.
Support, Service, and Warranty Index	→	Support, Service and Warranty (page 141) Index	This section contains Support, Service and Warranty information. Index of chapters.

1 Start Here

In This Chapter

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 CompactLogix or MicroLogix devices to a power source and to the MVI69-MNETC module’s application port(s)

1.1 System Requirements

The MVI69-MNETC module requires the following minimum hardware and software components:

- *Rockwell Automation CompactLogix* or *MicroLogix* processor, with compatible power supply and one free slot in the rack, for the MVI69-MNETC module. The module requires 800 mA of available power.

Important: The MVI69-MNETC module has a power supply distance rating of 2 (L43 and L45 installations on first 2 slots of 1769 bus).
Important: For 1769-L23x processors, please make note of the following limitations.

- 1769-L23-QBFC1B = 800 mA at 5 Vdc (One MVI69-MNETC will use all 800 mA of available power. No other modules can be used with an MVI69 module connected to this processor.)
- 1769-L23E-QB1B = 1000 mA at 5 Vdc (One MVI69-MNETC will use 800 mA of available power. One other module can be used on this rack provided it consumes less than 200 mA at 5 Vdc.)
- 1769-L23E-QBFC1B = 450 mA at 5 Vdc (No MVI69 module can be used with this processor.)

- *Rockwell Automation RSLogix 5000* (*CompactLogix*) or *RSLogix 500* (*MicroLogix*) programming software
- *Rockwell Automation RSLinx* communication software
- *Pentium*® *II* 450 MHz minimum. *Pentium III* 733 MHz (or better) recommended
- Supported operating systems:
  - *Microsoft Windows XP Professional* with *Service Pack* 1 or 2
  - *Microsoft Windows 2000 Professional* with *Service Pack* 1, 2, or 3
  - *Microsoft Windows Server 2003*
- 128 Mbytes of RAM minimum, 256 Mbytes of RAM recommended
- 100 Mbytes of free hard disk space (or more based on application requirements)
- 256-color VGA graphics adapter, 800 x 600 minimum resolution (True Color 1024 × 768 recommended)
- CD-ROM drive
- HyperTerminal or other terminal emulator program capable of file transfers using Zmodem protocol.

NOTE: MVI69/PS69 modules will not work with CompactLogix L4x processors using RSLogix 5000 v17. All other processor combinations and RSLogix versions will work correctly.

1.2 Package Contents

The following components are included with your MVI69-MNETC 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	MVI69-MNETC Module	MVI69-MNETC	Modbus TCP/IP Multi Client Communication Module
1	Cable	Cable #15 - RS232 Null Modem	For RS232 between a Personal Computer (PC) and the CFG port of the module
1	Cable	Cable #14 - RJ45 to DB9 Male Adapter	For connecting the module's port to Cable #15 for RS-232 connections
1	inRAx Solutions CD		Contains sample programs, utilities and documentation for the MVI69-MNETC module.

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

1.3 Installing ProSoft Configuration Builder Software

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

Installing ProSoft Configuration Builder from the ProSoft website

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

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

Installing ProSoft Configuration Builder from the Product CD-ROM

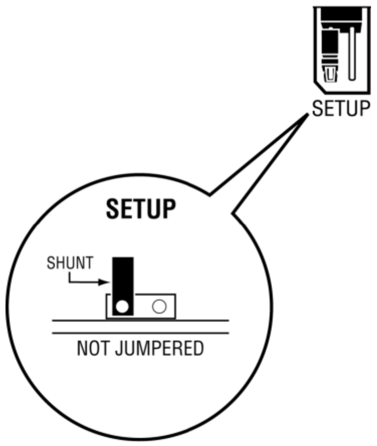
1 Insert the *ProSoft Solutions Product CD-ROM* into the CD-ROM drive of your PC. Wait for the startup screen to appear.
2 On the startup screen, click **PRODUCT DOCUMENTATION.** This action opens a *Windows Explorer* file tree window.
3 Click to open the **UTILITIES** folder. This folder contains all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

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

1.4 Setting Jumpers

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

The following illustration shows the MVI69-MNETC 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.

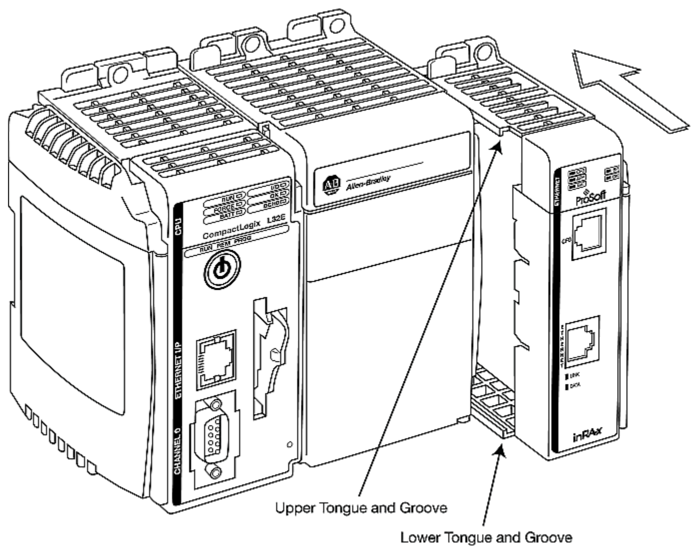
1.5 Installing the Module in the Rack

This section describes how to install the module into a CompactLogix or MicroLogix rack.

Before you attempt to install the module, make sure that the bus lever of the adjacent module is in the unlocked (fully right) position.

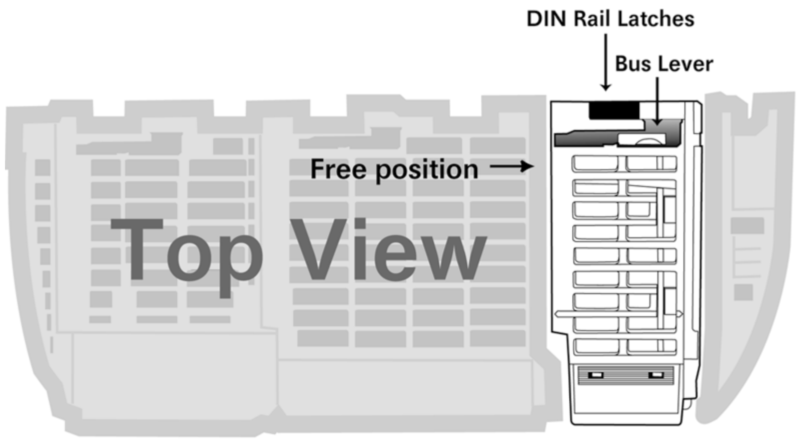
Warning: This module is not hot-swappable! Always remove power from the rack before inserting or removing this module, or damage may result to the module, the processor, or other connected devices.

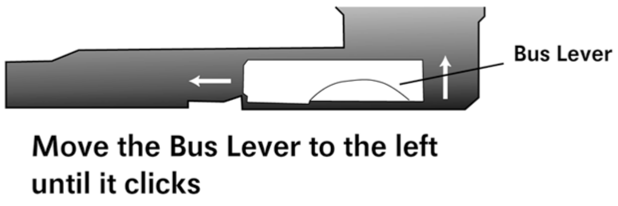
1 Align the module using the upper and lower tongue-and-groove slots with the adjacent module and slide forward in the direction of the arrow.

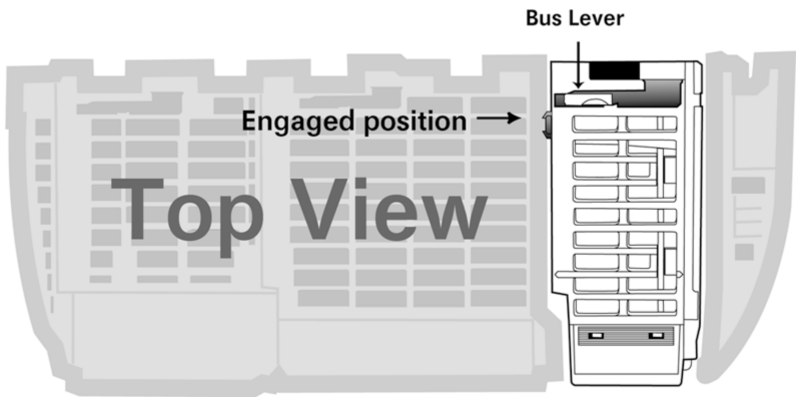


2 Move the module back along the tongue-and-groove slots until the bus connectors on the MVI69 module and the adjacent module line up with each other.

3 Push the module's bus lever back slightly to clear the positioning tab and move it firmly to the left until it clicks. Ensure that it is locked firmly in place.

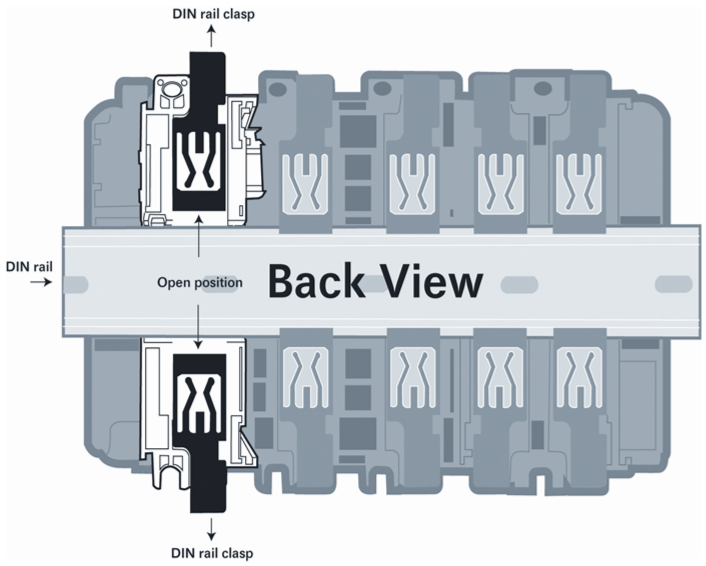


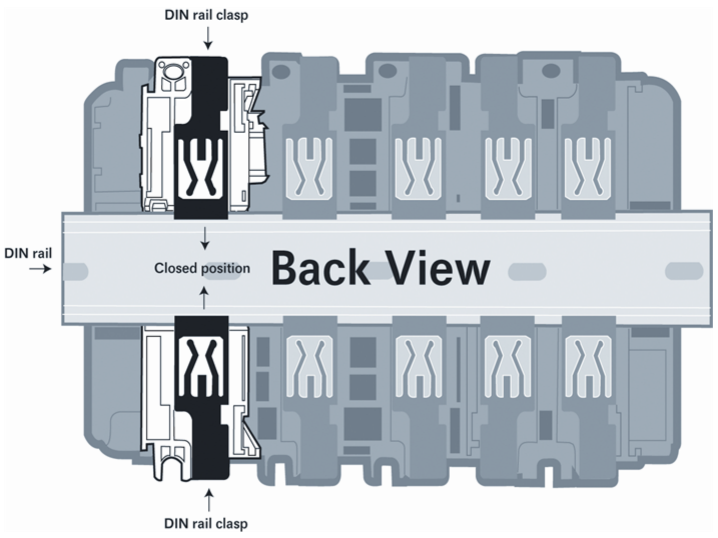




4 Close all DIN-rail latches.

5 Press the DIN-rail mounting area of the controller against the DIN-rail. The latches will momentarily open and lock into place.





2 Configuring the MVI69-MNETC Module

In This Chapter

The MVI69-MNETC module works on both the CompactLogix and the MicroLogix platform. The ladder logic you should import for the module depends on what type of processor you have, and what RSLogix software you are running.

If you have ...	... use this ProSoft ladder import.	File ext.	Instructions
CompactLogix processor with RSLogix 5000 Version 16 or higher	Add-On Instruction rung MVI69MNETC_AddOn_Rung_v1_3 .L5X	.L5X	MVI69-MNETC Add-On Instruction Rung Import Procedure (page 22)
CompactLogix processor with RSLogix 5000 Version 15 or earlier	Sample Ladder program. Use the version number that matches the version number of your RSLogix 5000 software.	.ACD	Adding the Module to an Existing CompactLogix Project (page 75)
MicroLogix processor (uses RSLogix 500)	Sample Ladder program for MicroLogix	.RSS	Adding the Module to an Existing MicroLogix Project (page 79)

All of these ladder files can be downloaded from www.prosoft-technology.com or copied from the ProSoft Solutions CD-ROM.

2.1 MVI69-MNETC Add-On Instruction Rung Import Procedure

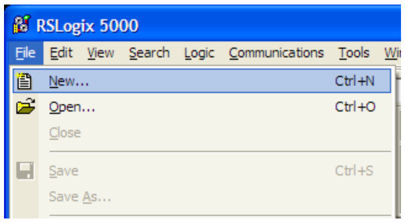
Note: This section only applies if you are using a CompactLogix processor running RSLogix 5000 version 16 or higher. If you are configuring the MVI69-MNETC module with an earlier version of RSLogix 5000, please refer to Adding the Module to an Existing CompactLogix Project (page 75). If you are using a MicroLogix processor, please see Adding the Module to an Existing MicroLogix Project (page 79).

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

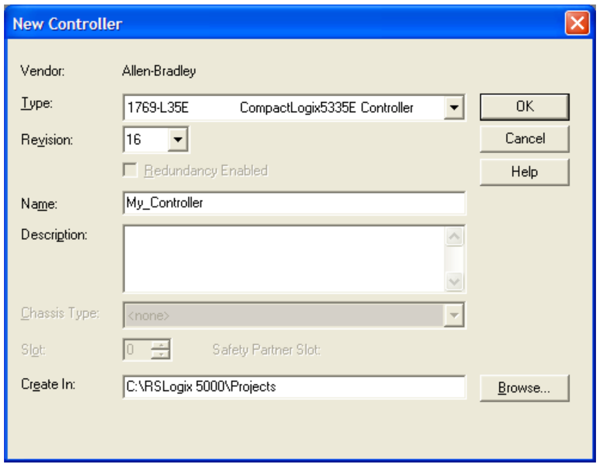
File Name	Description
MVI69MNETC_AddOn_Rung_v1_3.L5X	L5X file contains the Add-On instruction, the user defined data types, data objects and ladder logic required to set up the MVI69-MNETC module

2.1.1 Creating a New RSLogix 5000 Project

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



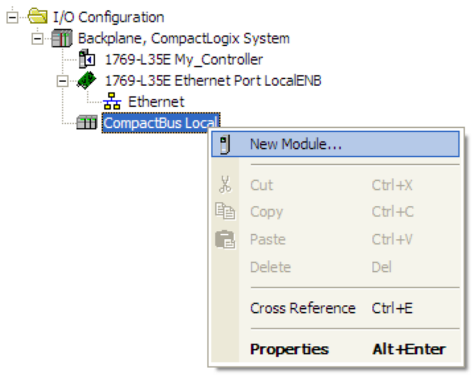
2 Select **REVISION 16**.


New Controller

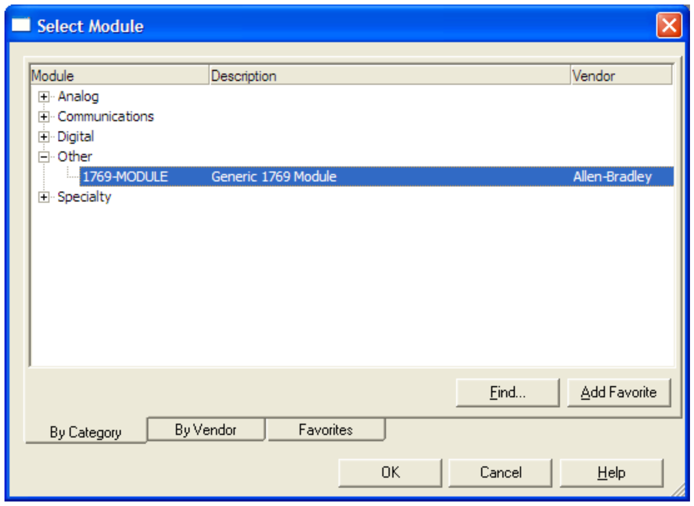
Vendor: Allen-Bradley
Type: 1769-L35E CompactLogix5335E Controller
Revision: 16
Redundancy Enabled
Name: My_Controller
Description:
Chassis Type: <none>
Slot: 0 Safety Partner Slot:
Create In: C:\RSLogix 5000\Projects

2.1.2 Creating the Module

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



2 Select **1769-MODULE**.



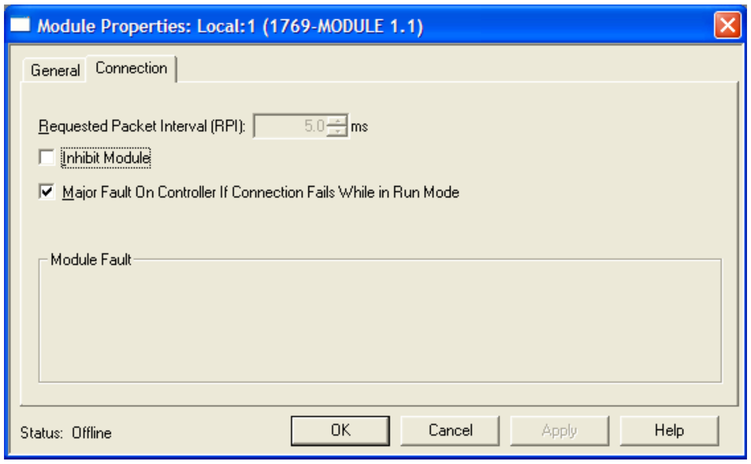
3 Set the Module Properties values as follows:

Parameter	**Value**
Name	Enter a module identification string. Example: **MNETC**
Description	Enter a description for the module. Example: **MODBUS TCP/IP MULTI CLIENT COMMUNICATION MODULE**
Comm Format	Select **DATA-INT**
Slot	Enter the slot number in the rack where the MVI69-MNETC module will be installed.
Input Assembly Instance	**101**
Input Size	**62 / 122 / 242**
Output Assembly Instance	**100**
Output Size	**61 / 121 / 241**
Configuration Assembly Instance	**102**
Configuration Size	**0**

The following illustration shows an example where the module was configured for a block transfer size of 60 words (input block size = 62 words, output block size = 61 words):
The following options are available:

Block Transfer Size	Input Block Size	Output Block Size
60	62	61
120	122	121
240	242	241

4 On the *Connection* tab, check or un-check, as desired, the *Major Fault* option.

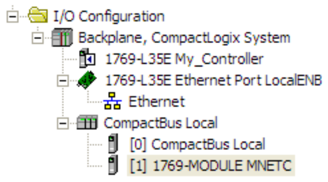


Module Properties: Local:1 (1769-MODULE 1.1)

Requested Packet Interval (RPI): 5.0 ms
☐ Inhibit Module
☑ Major Fault On Controller If Connection Fails While in Run Mode

Status: Offline

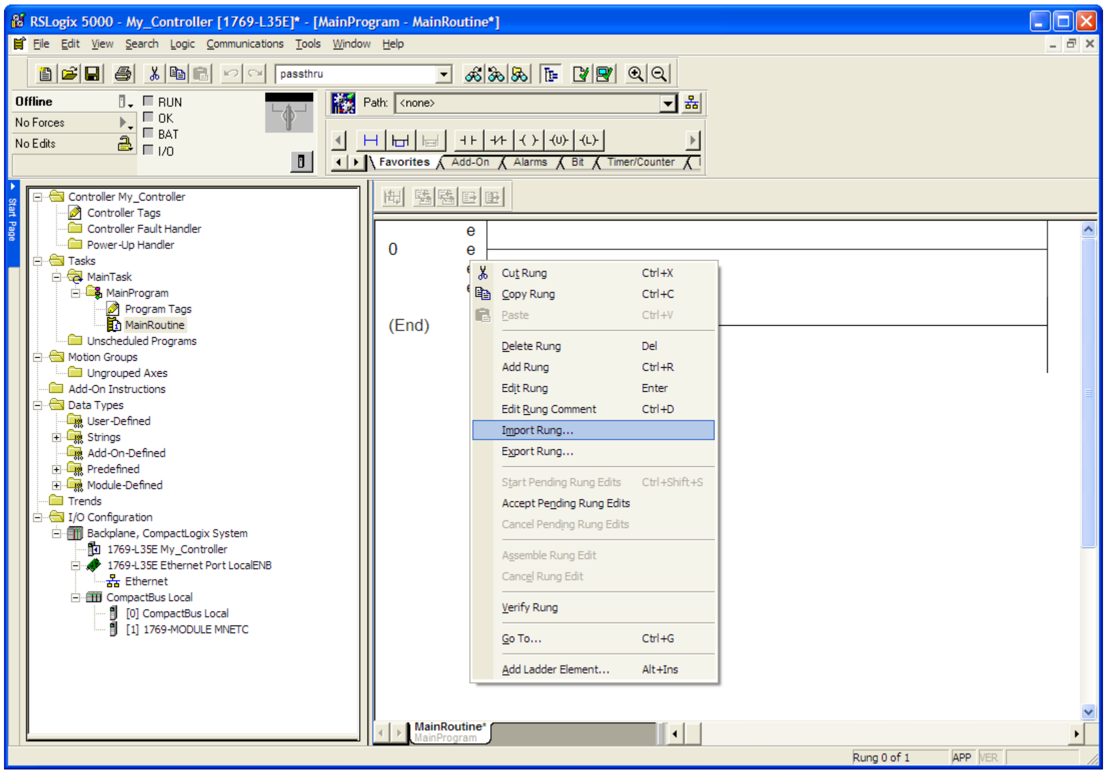
Now the MVI69-MNETC module will be visible at the I/O Configuration section.



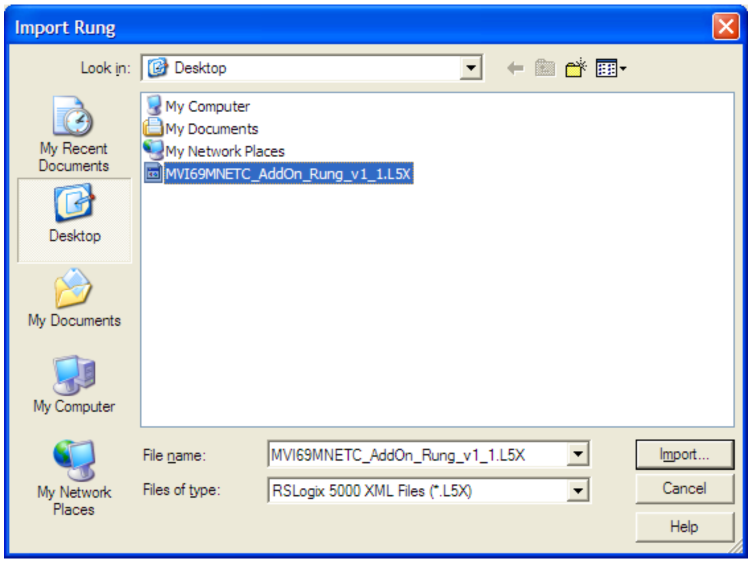
```
I/O Configuration
  Backplane, CompactLogix System
    1769-L35E My_Controller
    1769-L35E Ethernet Port LocalENB
      Ethernet
    CompactBus Local
      [0] CompactBus Local
      [1] 1769-MODULE MNETC
```

2.1.3 Importing the Ladder Rung

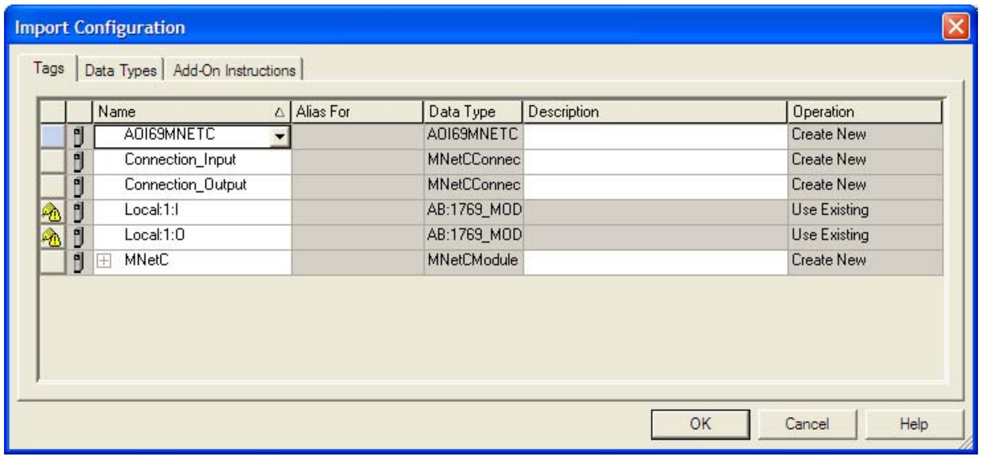
1 Expand the **TASKS** folder, and then expand the **MAIN TASK** folder.
2 Double-click to open the **MAINROUTINE** ladder.
3 Select an empty rung, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT RUNG**.



4 Select the **MVI69MNETC_ADDON_RUNG_V1_3.L5X** file.

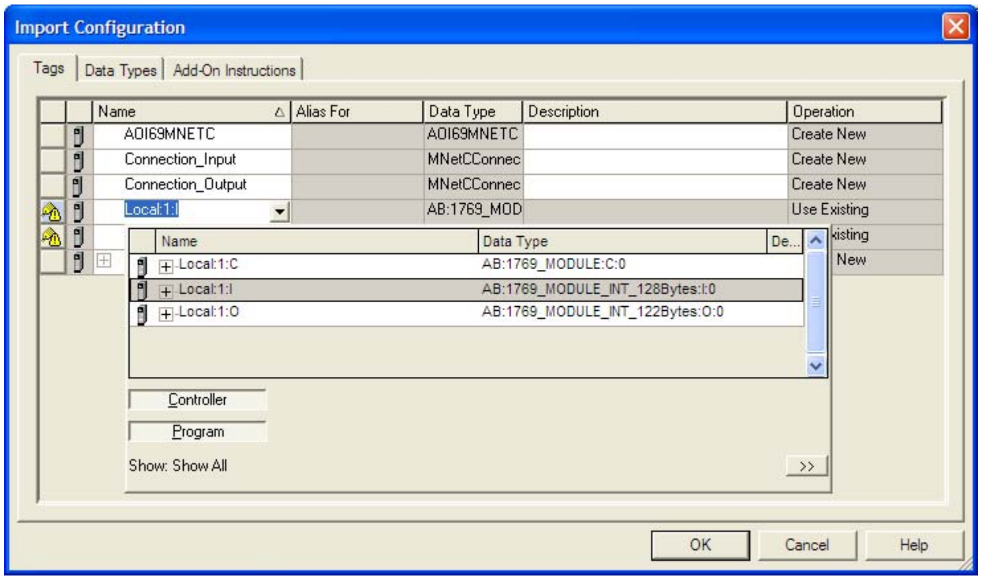


This action opens the *Import Configuration* dialog box, showing the controller tags to be created during the import procedure.



Name	Alias For	Data Type	Description	Operation
AOI69MNETC		AOI69MNETC		Create New
Connection_Input		MNetCConnec		Create New
Connection_Output		MNetCConnec		Create New
Local:1:I		AB:1769_MOD		Use Existing
Local:1:O		AB:1769_MOD		Use Existing
MNetC		MNetCModule		Create New

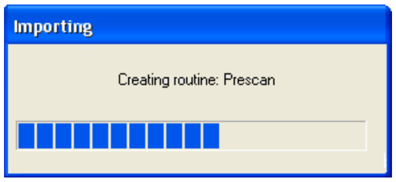
5 If you are using the module in a different slot (or remote rack), select the correct connection input and output variables associated to the module. If your module is located in slot 1 of the local rack, this step is not required.



Name	Data Type	De...
Local:1:C	AB:1769_MODULE:C:0	
Local:1:I	AB:1769_MODULE_INT_128Bytes:I:0	
Local:1:O	AB:1769_MODULE_INT_122Bytes:O:0	

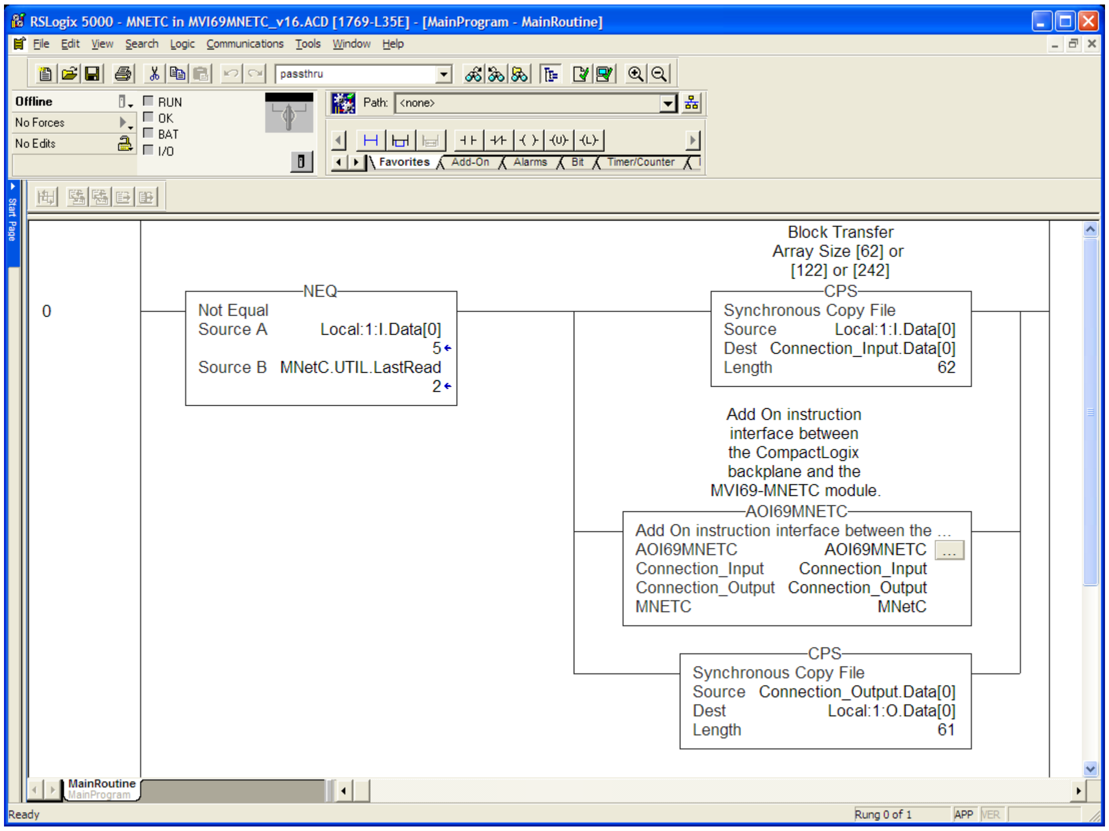
Show: Show All

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



Creating routine: Prescan

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



0

NEQ
Not Equal
Source A Local:1:I.Data[0]
5
Source B MNetC.UTIL.LastRead
2

Block Transfer
Array Size [62] or
[122] or [242]

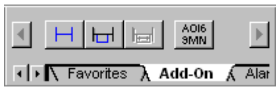
CPS
Synchronous Copy File
Source Local:1:I.Data[0]
Dest Connection_Input.Data[0]
Length 62

Add On instruction
interface between
the CompactLogix
backplane and the
MVI69-MNETC module.

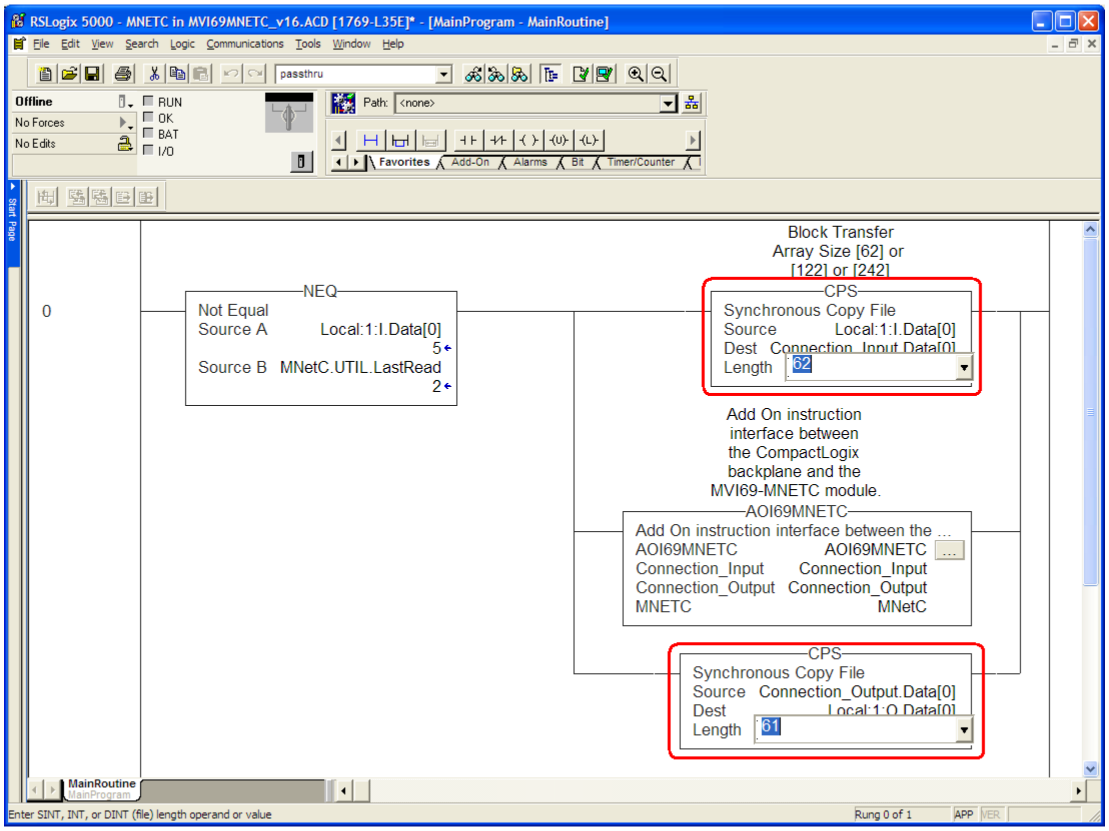
AOI69MNETC
Add On instruction interface between the ...
AOI69MNETC AOI69MNETC
Connection_Input Connection_Input
Connection_Output Connection_Output
MNETC MNetC

CPS
Synchronous Copy File
Source Connection_Output.Data[0]
Dest Local:1:O.Data[0]
Length 61

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



7 The imported rung will contain the Add-On instruction with two CPS instructions as follows below. The CPS instructions are set by default for a length of 62/61 words.



0

NEQ
Not Equal
Source A Local:1:I.Data[0]
5
Source B MNetC.UTIL.LastRead
2

Block Transfer
Array Size [62] or
[122] or [242]

CPS
Synchronous Copy File
Source Local:1:I.Data[0]
Dest Connection_Input.Data[0]
Length 62

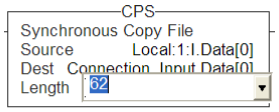
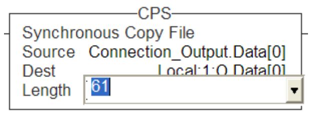
Add On instruction
interface between
the CompactLogix
backplane and the
MVI69-MNETC module.

AOI69MNETC
Add On instruction interface between the ...
AOI69MNETC AOI69MNETC
Connection_Input Connection_Input
Connection_Output Connection_Output
MNETC MNetC

CPS
Synchronous Copy File
Source Connection_Output.Data[0]
Dest Local:1:O.Data[0]
Length 61

Enter SINT, INT, or DINT (file) length operand or value
Rung 0 of 1
APP VER

Edit the above CPS instructions Length field values according to the following table.

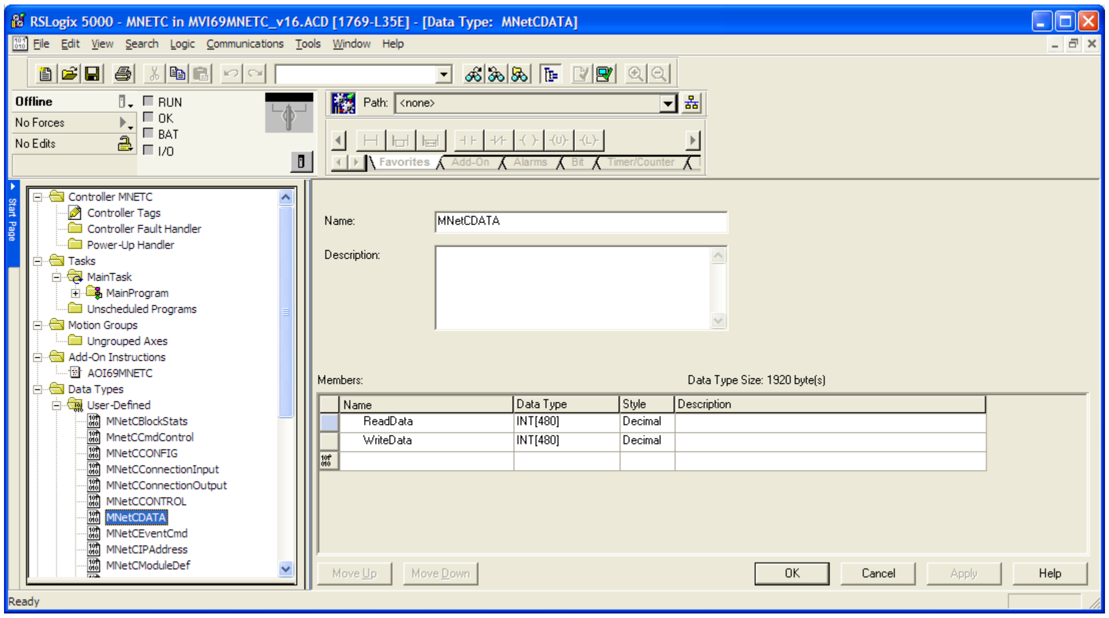
"Block Transfer Size Parameter" – 60/120/240 options)		**Ladder Routine window:**	
Connection Parameters:		**CPS instructions Length field values:**	
Input Size:	Output Size:	 CPS Synchronous Copy File Source Local:1:I.Data[0] Dest Connection_Input.Data[0] Length 62	 CPS Synchronous Copy File Source Connection_Output.Data[0] Dest Local:1:O.Data[0] Length 61
62	61	62	61
122	121	122	121
242	241	242	241

2.1.4 Setting the Read and Write Data Array Sizes

The imported rung contains the *MNetC.DATA* controller tag arrays *ReadData* and *WriteData*, set to the factory default array sizes of 480 elements each. These controller tags will contain:

- Read data - data copied from the module to the processor
- Write data - data copied from the processor to the module

If you have changed the *Read Register Count* and *Write Register Count* values in the *Module* section of the module's configuration, you must adjust these array sizes to match those values.



RSLogix 5000 - MNETC in MVI69MNETC_v16.ACD [1769-L35E] - [Data Type: MNetCDATA]

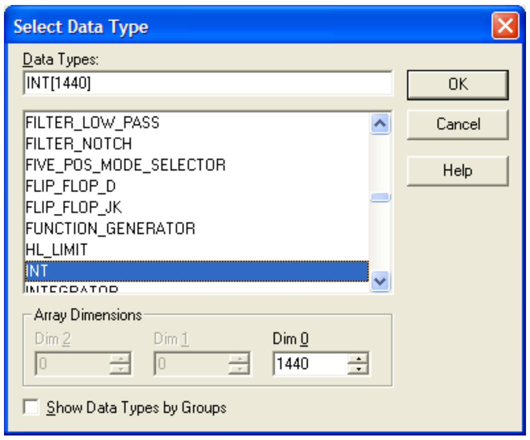
Name: MNetCDATA

Description:

Members: Data Type Size: 1920 byte(s)

Name	Data Type	Style	Description
ReadData	INT[480]	Decimal	
WriteData	INT[480]	Decimal	

Example: If the *Read Register Count* parameter in the *Module* section of the configuration is set to 1440, set the *ReadData* tag array size to INT[1440].

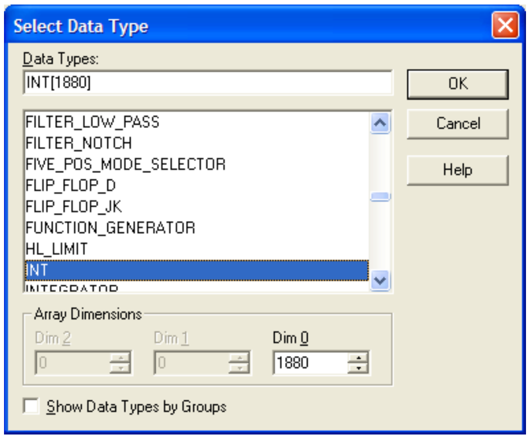


Select Data Type

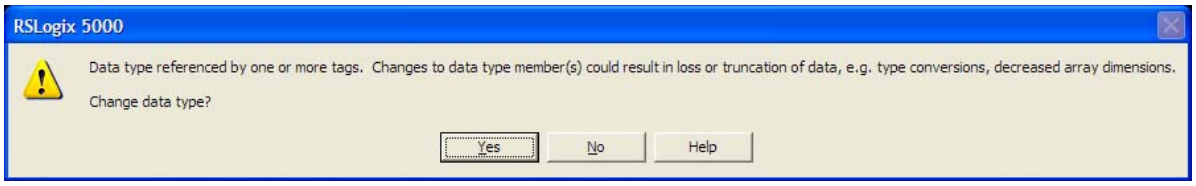
Data Types: INT[1440]

Array Dimensions: Dim 2: 0, Dim 1: 0, Dim 0: 1440

Example: If the *Write Register Count* parameter in the *Module* section of the configuration is set to 1880, set the *WriteData* tag array size to INT[1880].



You will be prompted to confirm the changes. Click **YES** to continue.

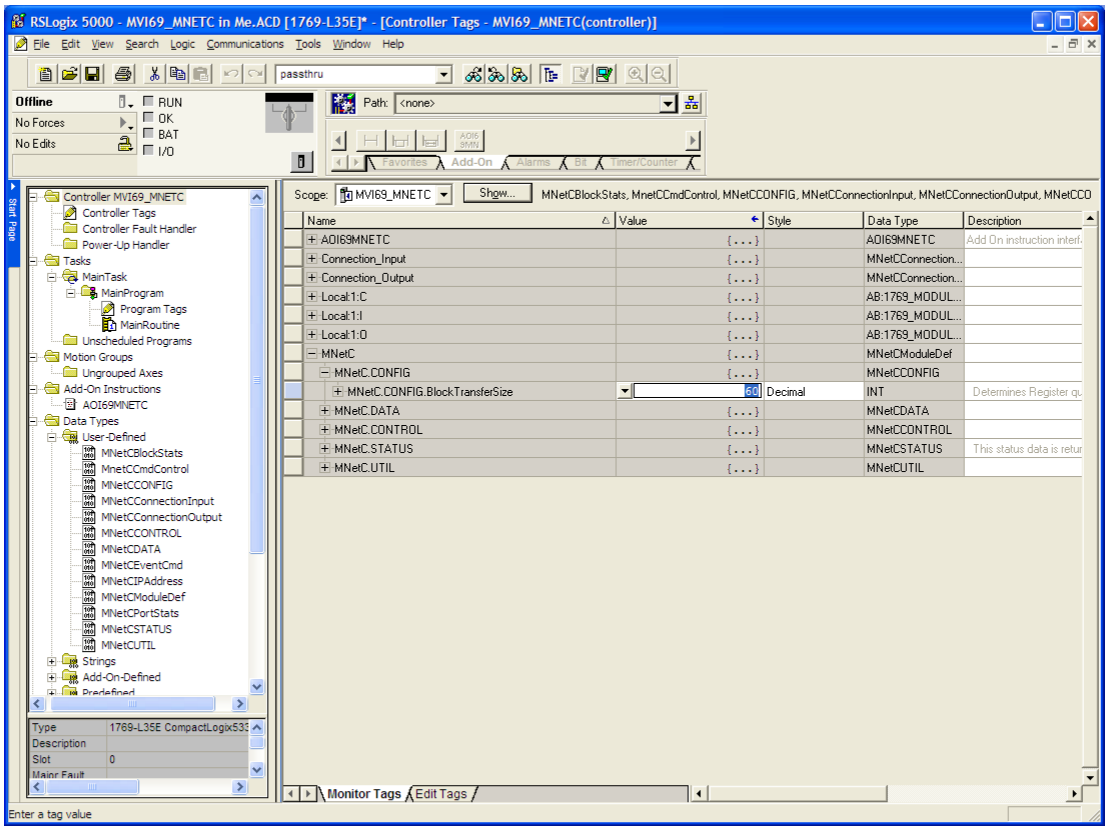

RSLogix 5000

Data type referenced by one or more tags. Changes to data type member(s) could result in loss or truncation of data, e.g. type conversions, decreased array dimensions.

Change data type?

2.1.5 Setting the Block Transfer Size

The *MNetC.CONFIG.BlockTransferSize* controller tag is set to 60 in the Add-On Instruction. If you have configured a different block transfer size in the module's configuration file, you must change this value to match.



RSLogix 5000 - MVI69_MNETC in Me.ACD [1769-L35E]* - [Controller Tags - MVI69_MNETC(controller)]

Scope: MVI69_MNETC Show... MNetCBlockStats, MnetCCmdControl, MNetCCONFIG, MNetCConnectionInput, MNetCConnectionOutput, MNetCCO

Name	Value	Style	Data Type	Description
+ AOI69MNETC	{...}		AOI69MNETC	Add On instruction interf
+ Connection_Input	{...}		MNetCConnection...	
+ Connection_Output	{...}		MNetCConnection...	
+ Local:1:C	{...}		AB:1769_MODUL...	
+ Local:1:I	{...}		AB:1769_MODUL...	
+ Local:1:O	{...}		AB:1769_MODUL...	
− MNetC	{...}		MNetCModuleDef	
− MNetC.CONFIG	{...}		MNetCCONFIG	
+ MNetC.CONFIG.BlockTransferSize	60	Decimal	INT	Determines Register qu
+ MNetC.DATA	{...}		MNetCDATA	
+ MNetC.CONTROL	{...}		MNetCCONTROL	
+ MNetC.STATUS	{...}		MNetCSTATUS	This status data is retur
+ MNetC.UTIL	{...}		MNetCUTIL	

Enter a tag value

Edit the tag values according to the following table.

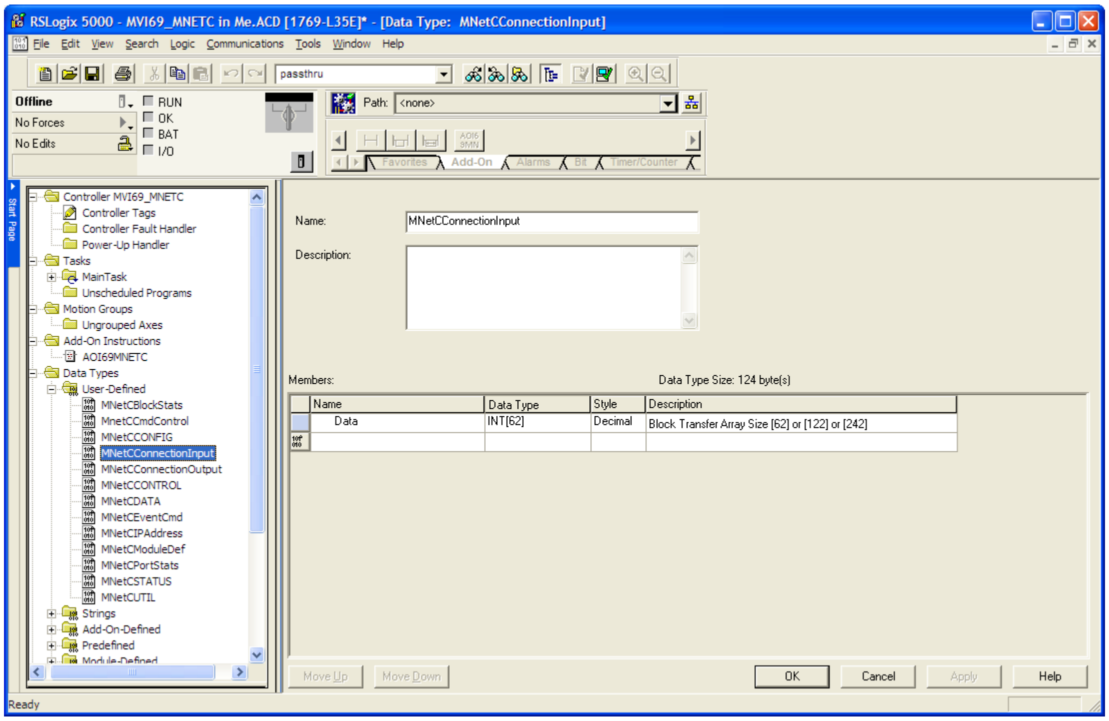
Module Properties dialog box:		Controller Organizer's Controller Tags folder:
Connection Parameters:		**MNetC.CONFIG.BlockTransferSize tag value:**
Input Size:	**Output Size:**	
62	61	60
122	121	120
242	241	240

2.1.6 Setting the Connection Input and Output Sizes

You must also configure the following data types to match the block transfer size value used for your project:

- *MNetCConnectionInput* – Data type used for the Connection Input pin in the Add-On instruction.
- *MNetCConnectionOutput* – Data type used for the Connection Output pin in the Add-On instruction.

Open the user-defined data type definition *MNetCConnectionInput* as follows.



Name: MNetCConnectionInput

Description:

Members:

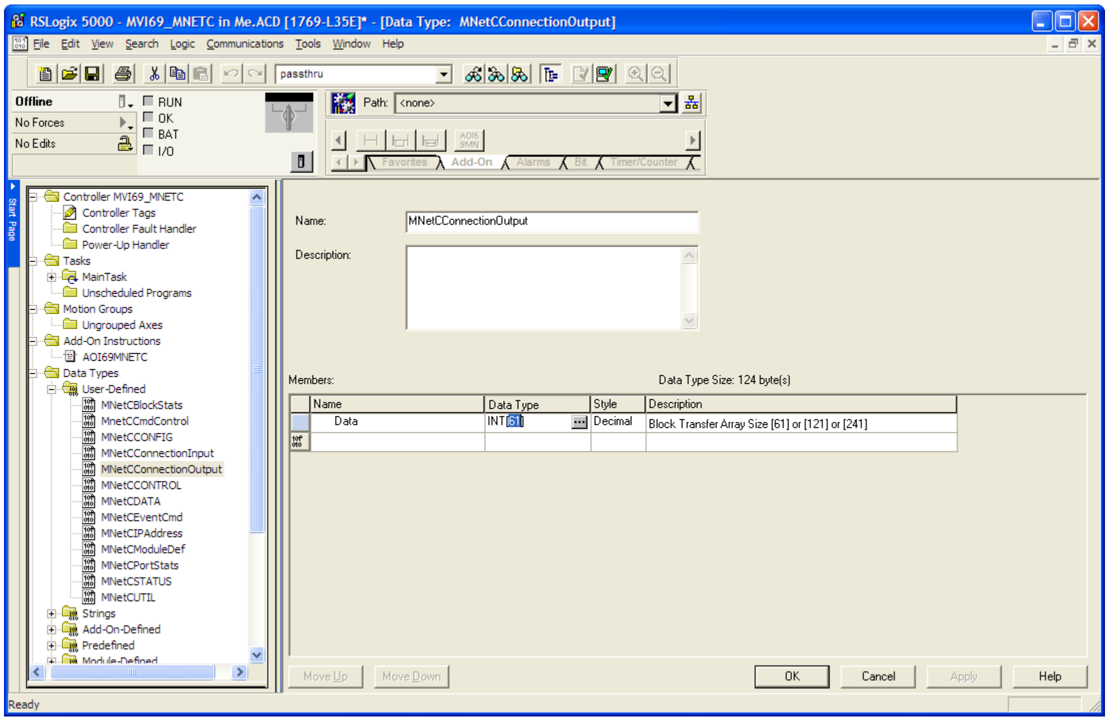
Data Type Size: 124 byte(s)

Name	Data Type	Style	Description
Data	INT[62]	Decimal	Block Transfer Array Size [62] or [122] or [242]

Edit the tag values according to the following table.

Module Properties dialog box:	Controller Organizer's Controller Tags folder:
Connection Parameters:	**MNetCConnectionInput.Data tag value:**
Input Size:	
62	62
122	122
242	242

Open the user-defined data type definition *MNetCConnectionOutput* as follows:



Name: MNetCConnectionOutput

Description:

Members: Data Type Size: 124 byte(s)

Name	Data Type	Style	Description
Data	INT[61]	Decimal	Block Transfer Array Size [61] or [121] or [241]

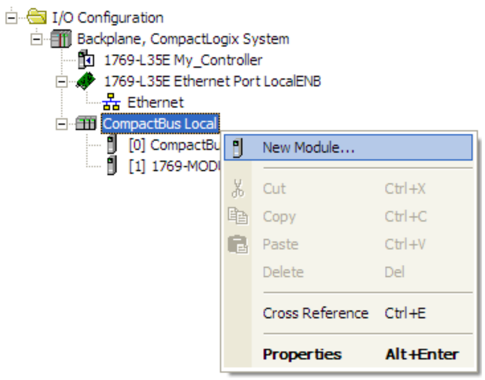
Edit the tag values according to the following table.

Module Properties dialog box:	**Controller Organizer's Controller Tags folder:**
Connection Parameters:	**MNetCConnectionOutput.Data tag value:**
Output Size:	
61	61
121	121
241	241

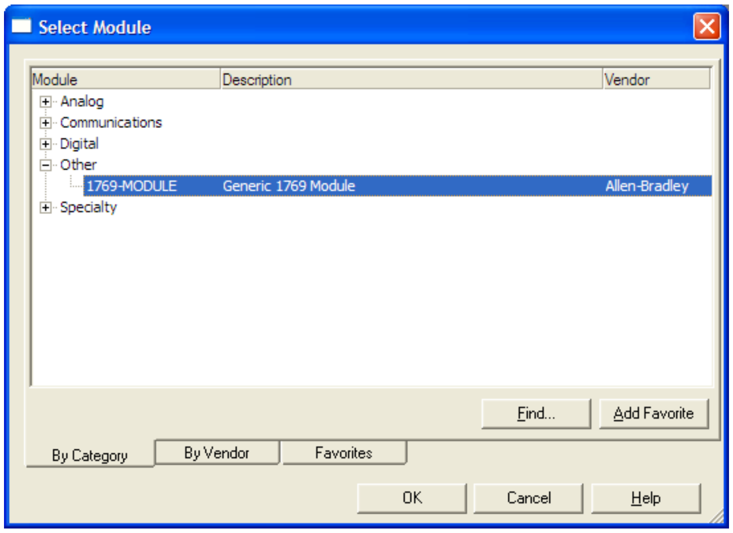
2.1.7 Adding Multiple Modules (Optional)

Important: If your application requires more than one MVI69-MNETC module in the same project, follow the steps below and make certain that both modules are assigned identical Block Transfer Sizes.

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



2 Select **1769-MODULE**.

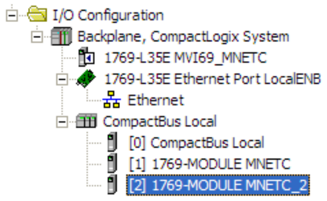


Module	Description	Vendor
1769-MODULE	Generic 1769 Module	Allen-Bradley

3 Fill the module properties as follows.

Parameter	**Value**
Name	Enter a module identification string. Example: **MNETC_2**
Description	Enter a description for the module. Example: **MODBUS TCP/IP MULTI CLIENT COMMUNICATION MODULE**
Comm Format	Select **DATA-INT**
Slot	Enter the slot number in the rack where the MVI69-MNETC module will be installed.
Input Assembly Instance	**101**
Input Size	**62 / 122 / 242**
Output Assembly Instance	**100**
Output Size	**61 / 121 / 241**
Configuration Assembly Instance	**102**
Configuration Size	**0**

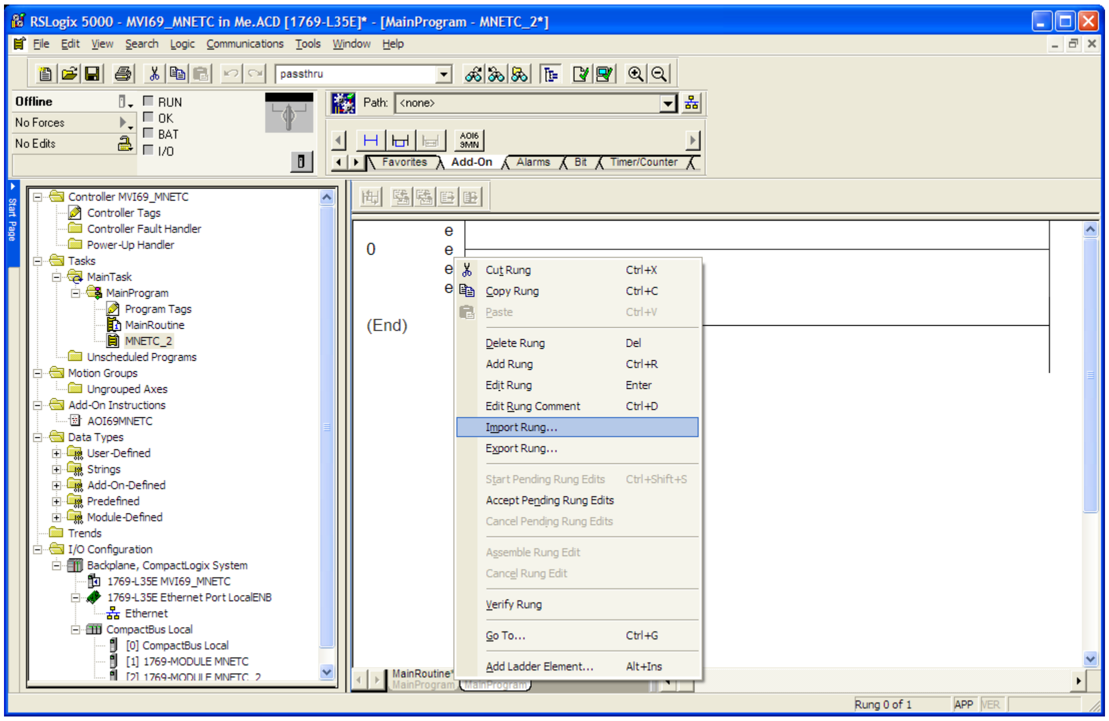
4 Click **OK** to confirm. The new module is now visible:



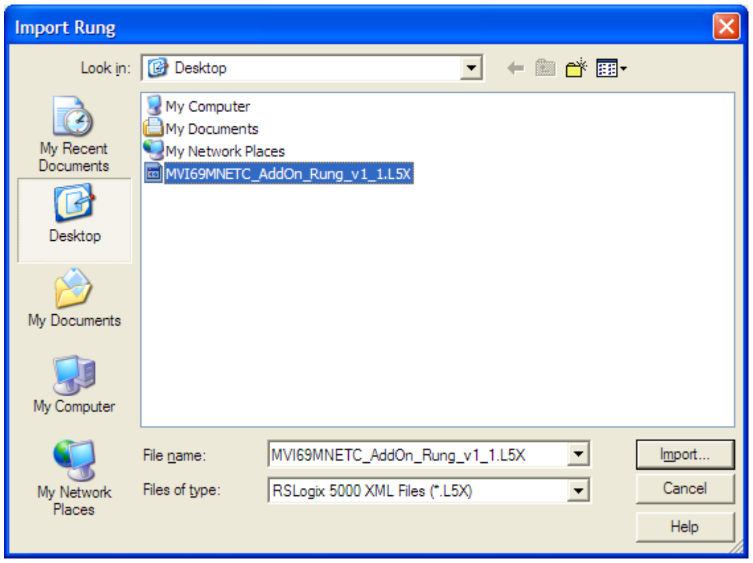
- I/O Configuration
  - Backplane, CompactLogix System
    - 1769-L35E MVI69_MNETC
    - 1769-L35E Ethernet Port LocalENB
      - Ethernet
    - CompactBus Local
      - [0] CompactBus Local
      - [1] 1769-MODULE MNETC
      - [2] 1769-MODULE MNETC_2

5 Expand the **TASKS** folder, and then expand the **MAINTASK** folder.
6 On the **MAINPROGRAM** folder, click the right mouse button to open a shortcut menu. On the shortcut menu, choose **NEW ROUTINE**.
7 In the **NEW ROUTINE** dialog box, enter the name and description of your routine, and then click **OK.**
8 Select an empty rung in the new routine, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose **IMPORT RUNG**.

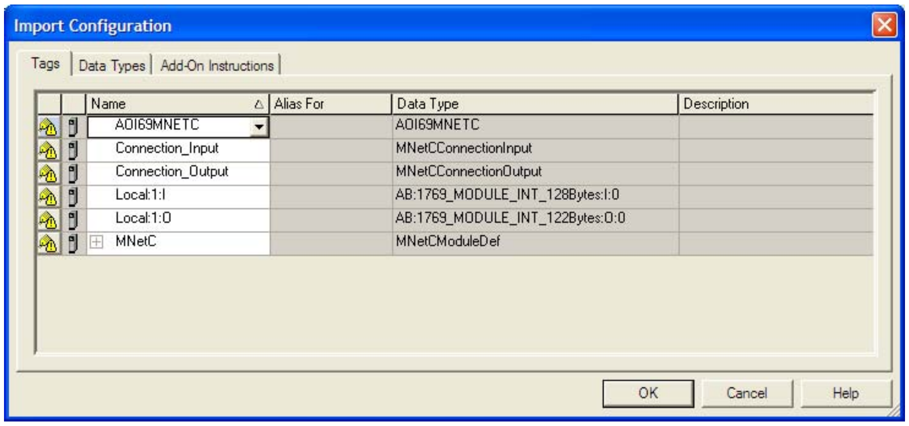
Note: It is not necessary to create a completely new routine. It is possible to add the MVI69-MNETC_2 module in the previously created routine. If you do create a new routine, you must create a rung with a jump instruction in the previous routine to the new routine.



9 Select the file **MVI69MNETC_ADDON_RUNG_V1_3.L5X**.

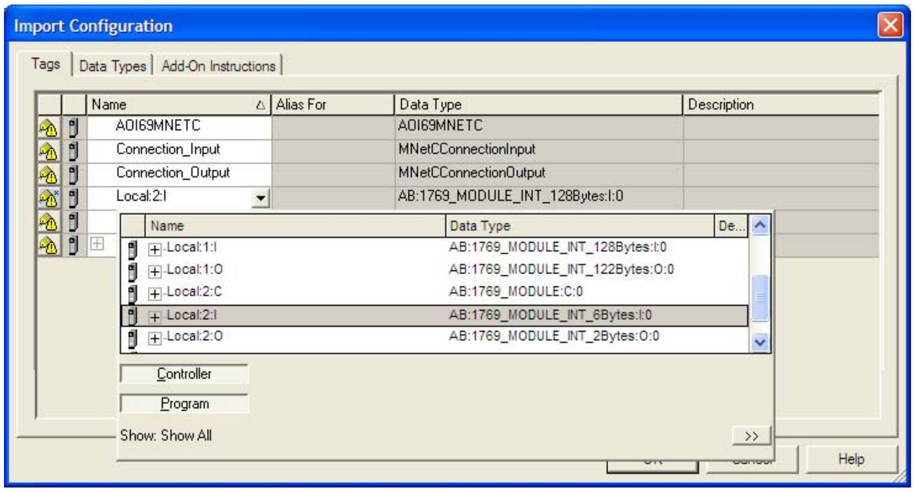


10 The *Import Configuration* dialog box shows the tags to be imported:



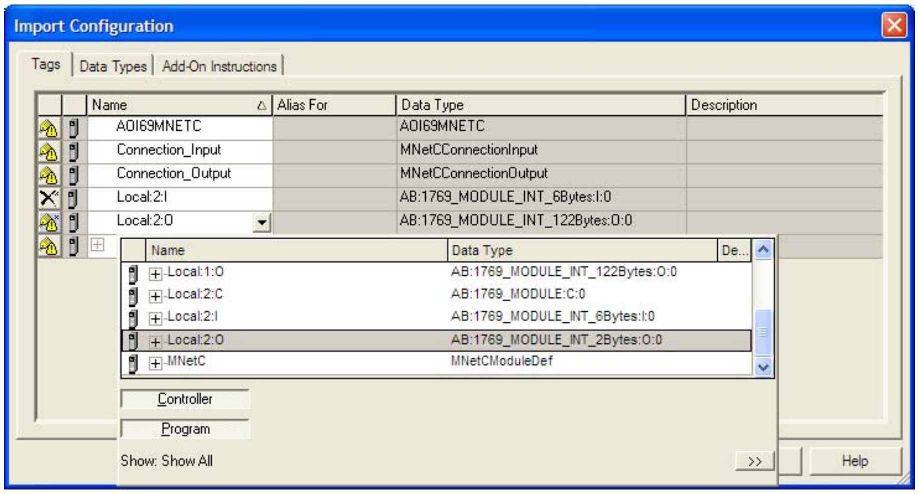
Name	Alias For	Data Type	Description
AOI69MNETC		AOI69MNETC	
Connection_Input		MNetCConnectionInput	
Connection_Output		MNetCConnectionOutput	
Local:1:I		AB:1769_MODULE_INT_128Bytes:I:0	
Local:1:O		AB:1769_MODULE_INT_122Bytes:O:0	
MNetC		MNetCModuleDef	

11 Associate the I/O connection variables to the correct module. The default values are Local:1:I and Local:1:O. You must reassign these to the new module's location.



Name	Alias For	Data Type	Description
AOI69MNETC		AOI69MNETC	
Connection_Input		MNetCConnectionInput	
Connection_Output		MNetCConnectionOutput	
Local:2:I		AB:1769_MODULE_INT_128Bytes:I:0	

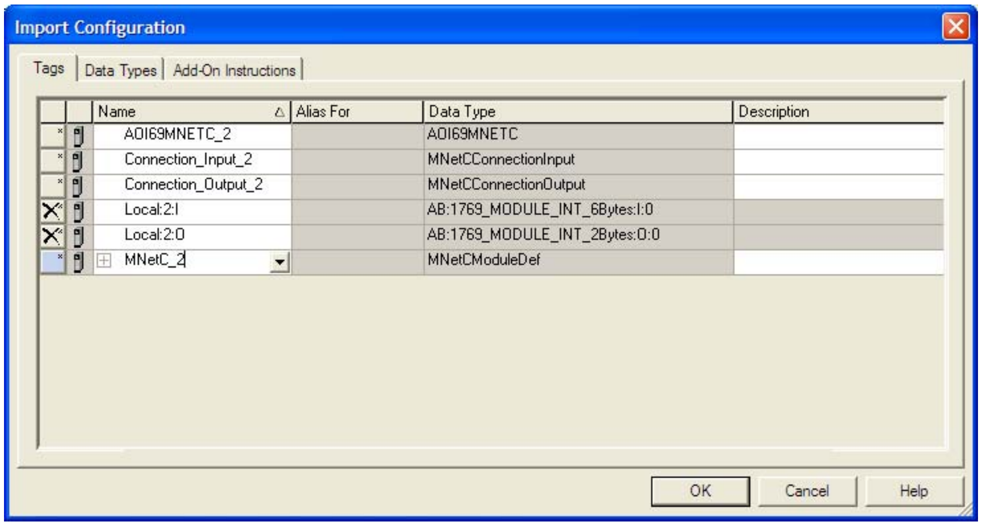
Name	Data Type	De...
Local:1:I	AB:1769_MODULE_INT_128Bytes:I:0	
Local:1:O	AB:1769_MODULE_INT_122Bytes:O:0	
Local:2:C	AB:1769_MODULE:C:0	
Local:2:I	AB:1769_MODULE_INT_6Bytes:I:0	
Local:2:O	AB:1769_MODULE_INT_2Bytes:O:0	



Name	Alias For	Data Type	Description
AOI69MNETC		AOI69MNETC	
Connection_Input		MNetCConnectionInput	
Connection_Output		MNetCConnectionOutput	
Local:2:I		AB:1769_MODULE_INT_6Bytes:I:0	
Local:2:O		AB:1769_MODULE_INT_122Bytes:O:0	

Name	Data Type	De...
Local:1:O	AB:1769_MODULE_INT_122Bytes:O:0	
Local:2:C	AB:1769_MODULE:C:0	
Local:2:I	AB:1769_MODULE_INT_6Bytes:I:0	
Local:2:O	AB:1769_MODULE_INT_2Bytes:O:0	
MNetC	MNetCModuleDef	

12 Change the default tags **MNETC** and **AOI69MNETC** to avoid conflict with existing tags. This example procedure will append the string "_2" as follows:

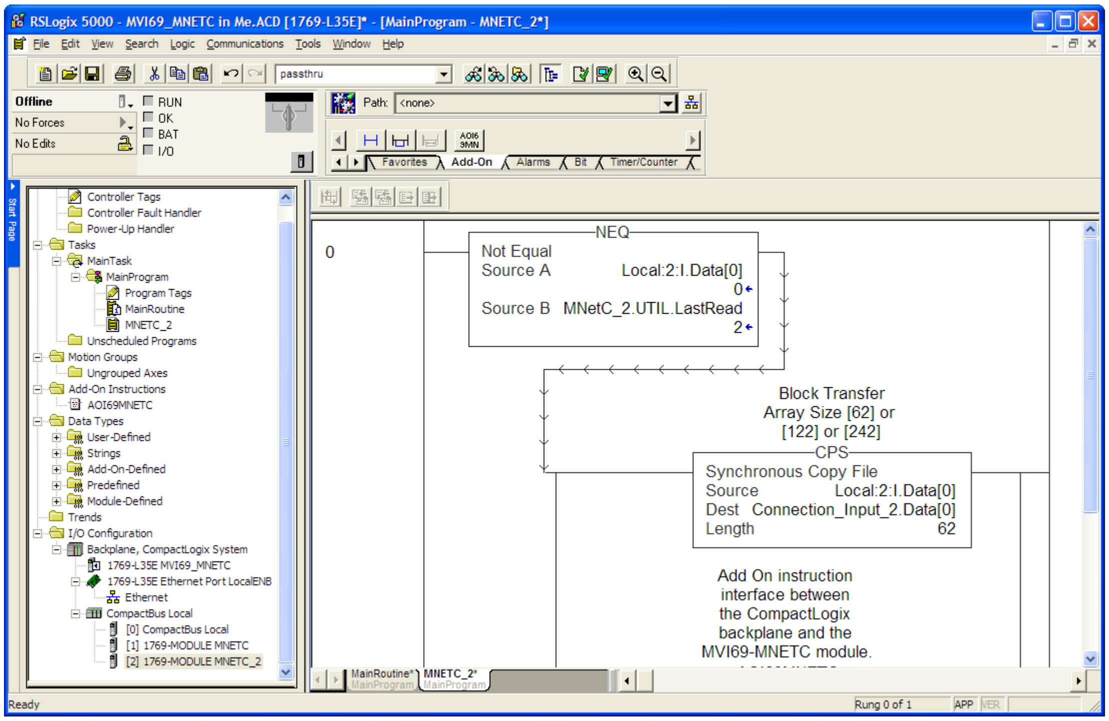


Import Configuration

Tags | Data Types | Add-On Instructions

Name	Alias For	Data Type	Description
AOI69MNETC_2		AOI69MNETC	
Connection_Input_2		MNetCConnectionInput	
Connection_Output_2		MNetCConnectionOutput	
Local:2:I		AB:1769_MODULE_INT_6Bytes:I:0	
Local:2:O		AB:1769_MODULE_INT_2Bytes:O:0	
MNetC_2		MNetCModuleDef	

13 You will be prompted to confirm your change. Click **OK** to continue.



RSLogix 5000 - MVI69_MNETC in Me.ACD [1769-L35E]* - [MainProgram - MNETC_2*]

0

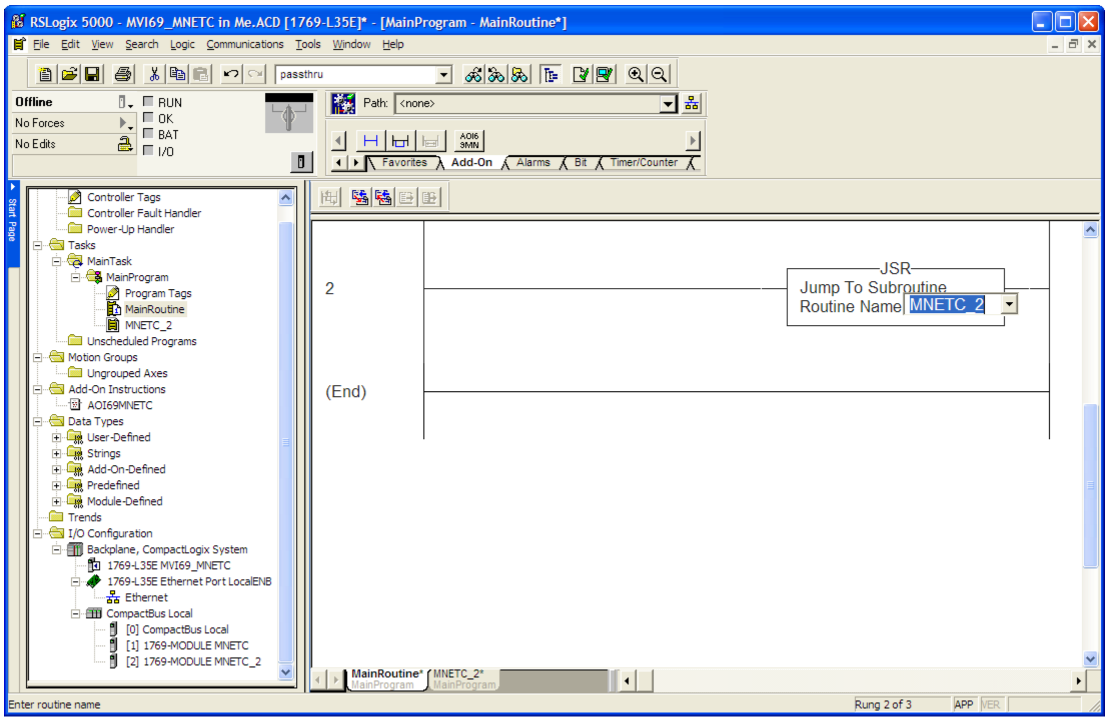
NEQ
Not Equal
Source A Local:2:I.Data[0]
0
Source B MNetC_2.UTIL.LastRead
2

Block Transfer
Array Size [62] or
[122] or [242]

CPS
Synchronous Copy File
Source Local:2:I.Data[0]
Dest Connection_Input_2.Data[0]
Length 62

Add On instruction
interface between
the CompactLogix
backplane and the
MVI69-MNETC module.

14 Because the second module's logic was created in a new routine in this example, you must enter a rung in the Main routine with a **JSR** instruction to the new routine to enable the PLC logic to communicate with both modules.

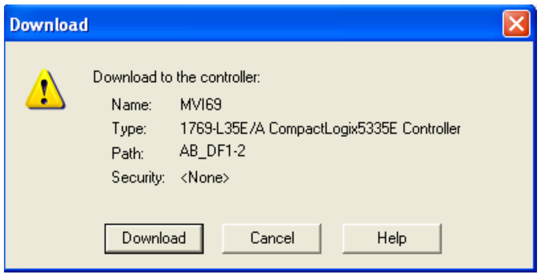


The setup procedure is now complete. Save the project and download the application to your CompactLogix processor.

2.2 Downloading the Sample Program to the Processor

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

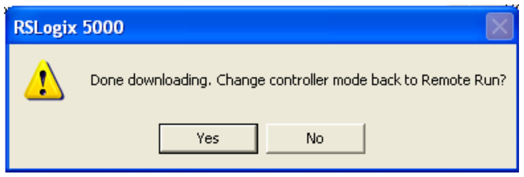
1 If you are not already online to the processor, open the **COMMUNICATIONS** menu, and then choose **DOWNLOAD**. RSLogix will establish communication with the processor.
2 When communication is established, RSLogix will open a confirmation dialog box. Click the **DOWNLOAD** button to transfer the sample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


Download

Download to the controller:

Name: MVI69
Type: 1769-L35E/A CompactLogix5335E Controller
Path: AB_DF1-2
Security: <None>

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


RSLogix 5000

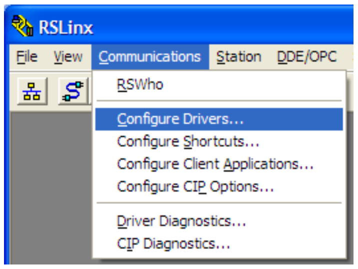
Done downloading. Change controller mode back to Remote Run?

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

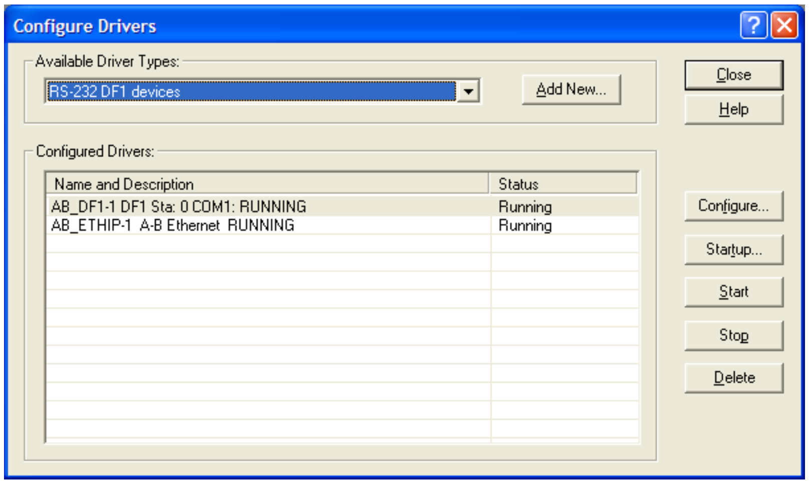
2.2.1 Configuring the RSLinx Driver for the PC COM Port

If RSLogix is unable to establish communication with the processor, follow these steps.

1 Open *RSLinx*.
2 Open the **COMMUNICATIONS** menu, and choose **CONFIGURE DRIVERS**.



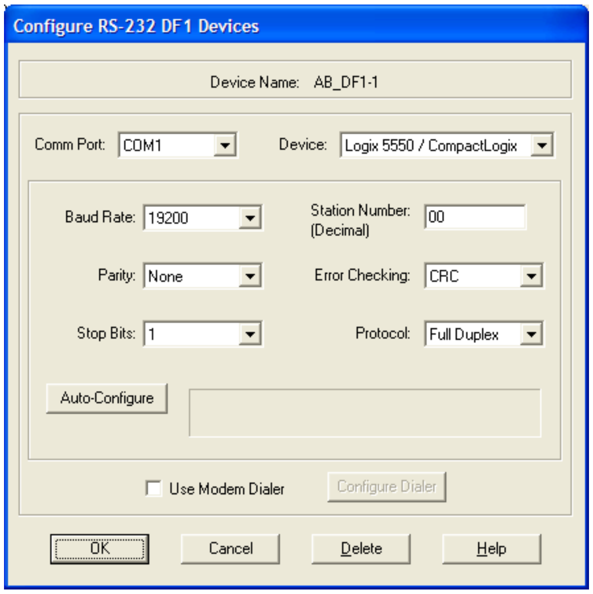
This action opens the *Configure Drivers* dialog box.



Name and Description	Status
AB_DF1-1 DF1 Sta: 0 COM1: RUNNING	Running
AB_ETHIP-1 A-B Ethernet RUNNING	Running

Note: If the list of configured drivers is blank, you must first choose and configure a driver from the Available Driver Types list. The recommended driver type to choose for serial communication with the processor is *RS-232 DF1 Devices*.

3 Click to select the driver, and then click **CONFIGURE**. This action opens the *Configure RS-232 DF1 Devices* dialog box.



Configure RS-232 DF1 Devices

Device Name: AB_DF1-1

Comm Port: COM1
Device: Logix 5550 / CompactLogix

Baud Rate: 19200
Station Number: (Decimal) 00
Parity: None
Error Checking: CRC
Stop Bits: 1
Protocol: Full Duplex

4 Click the **AUTO-CONFIGURE** button. RSLinx will attempt to configure your serial port to work with the selected driver.

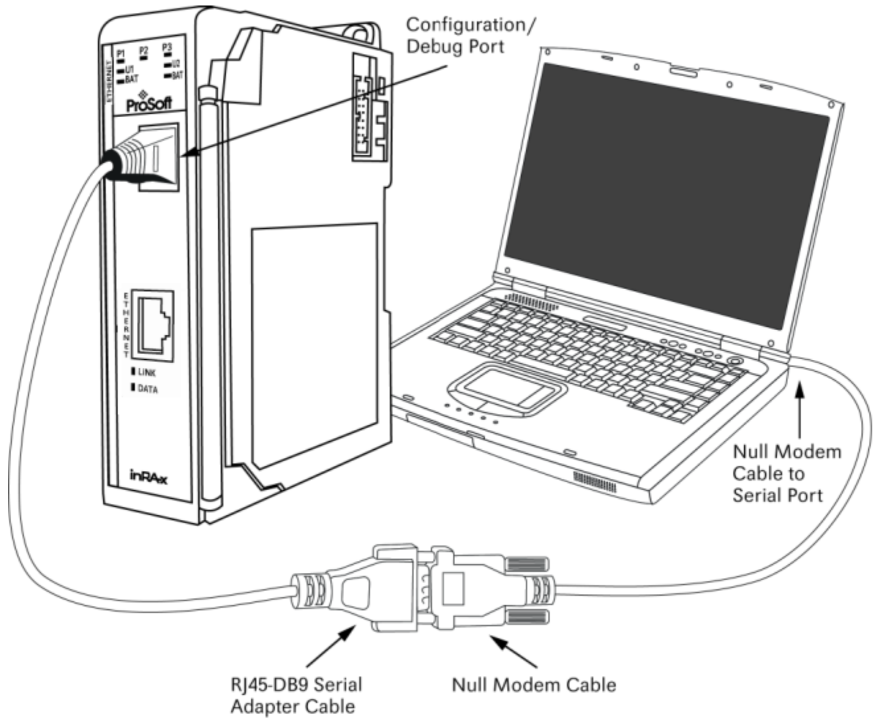
5 When you see the message *Auto Configuration Successful*, click the **OK** button to dismiss the dialog box.

Note: If the auto-configuration procedure fails, verify that the cables are connected correctly between the processor and the serial port on your computer, and then try again. If you are still unable to auto-configure the port, refer to your RSLinx documentation for further troubleshooting steps.

2.3 Connecting Your PC to the Module

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

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

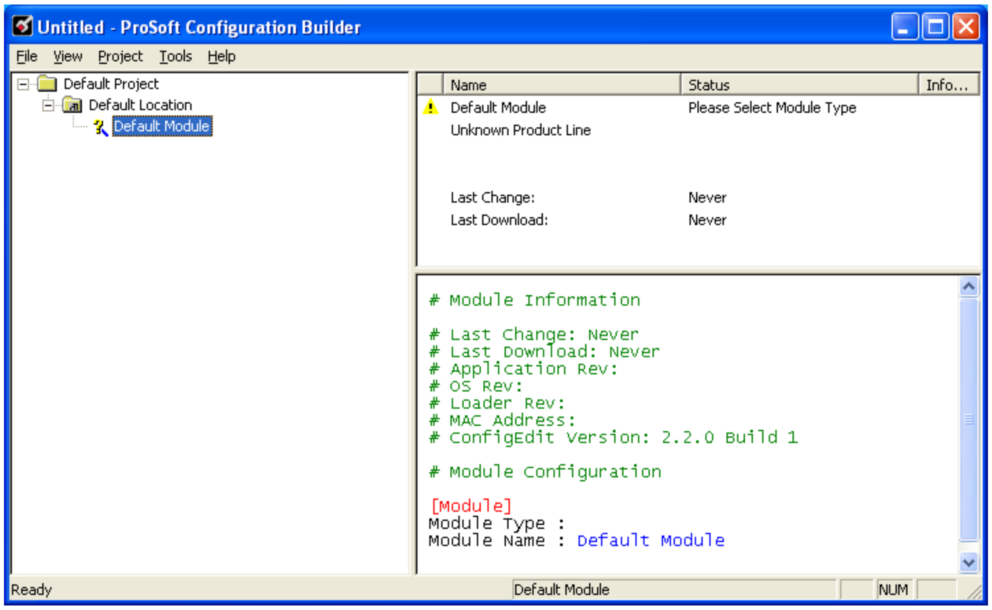


2.4 Using ProSoft Configuration Builder

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

2.4.1 Setting Up the Project

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


Untitled - ProSoft Configuration Builder
File View Project Tools Help
Default Project
Default Location
Default Module
Ready
Default Module
NUM

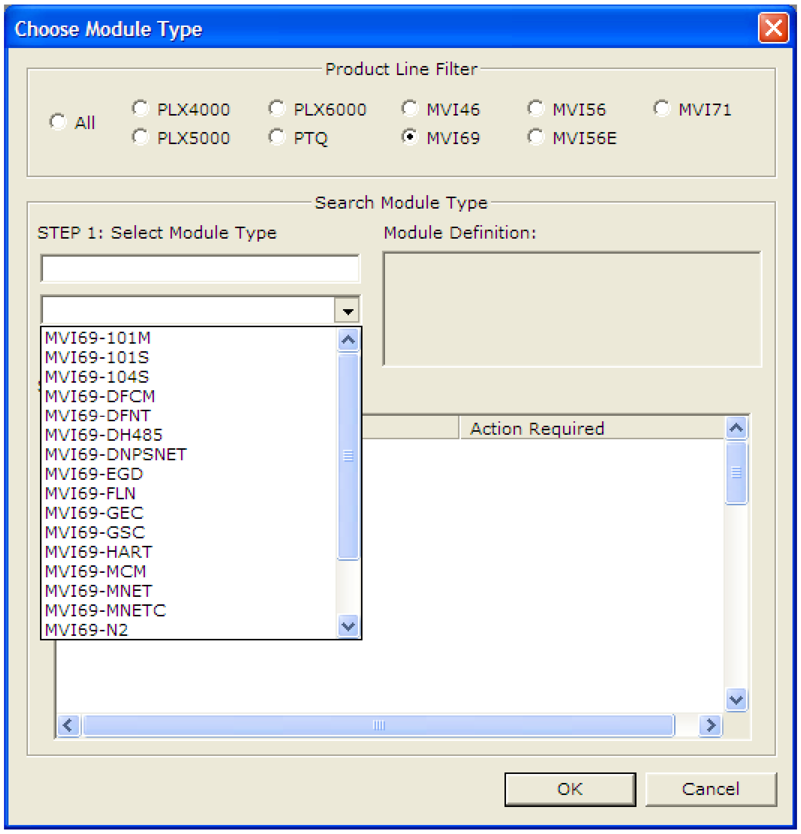
Name	Status	Info...
Default Module	Please Select Module Type	
Unknown Product Line		
Last Change:	Never	
Last Download:	Never	

```
# Module Information

# Last Change: Never
# Last Download: Never
# Application Rev:
# OS Rev:
# Loader Rev:
# MAC Address:
# ConfigEdit Version: 2.2.0 Build 1

# Module Configuration

[Module]
Module Type :
Module Name : Default Module
```

Your first task is to add the MVI69-MNETC 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.

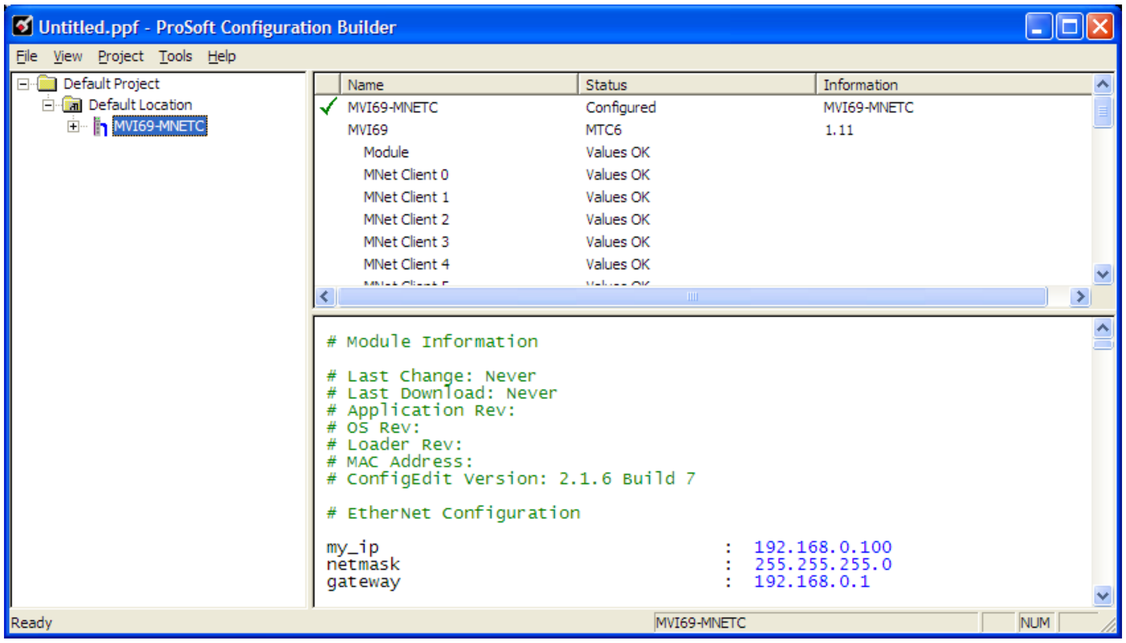


Product Line Filter: [✓ MVI69]

3 In the *Product Line Filter* area of the dialog box, select **MVI69**. In the *Select Module Type* dropdown list, select **MVI69-MNETC**, and then click **OK** to save your settings and return to the *ProSoft Configuration Builder* window.

2.4.2 Setting Module Parameters

Notice that the contents of the information pane and the configuration pane changed when you added the MVI69-MNETC module to the project.


Untitled.ppf - ProSoft Configuration Builder
File View Project Tools Help
Default Project
Default Location
MVI69-MNETC
Ready
MVI69-MNETC
NUM

Name	Status	Information
✓ MVI69-MNETC	Configured	MVI69-MNETC
MVI69	MTC6	1.11
Module	Values OK	
MNet Client 0	Values OK	
MNet Client 1	Values OK	
MNet Client 2	Values OK	
MNet Client 3	Values OK	
MNet Client 4	Values OK	

```
# Module Information

# Last Change: Never
# Last Download: Never
# Application Rev:
# OS Rev:
# Loader Rev:
# MAC Address:
# ConfigEdit Version: 2.1.6 Build 7

# EtherNet Configuration

my_ip                              :  192.168.0.100
netmask                            :  255.255.255.0
gateway                            :  192.168.0.1
```

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

__Renaming an Object__

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

__Configuring Module Parameters__

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

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.**

2.5 Module Configuration

2.5.1 Module

This section of the configuration describes the database setup and module-level parameters.

<u>Backplane Error/Status Pointer</u>

1 to **4955**

This parameter sets the address in the internal database where the backplane error/status data will be placed. If you want the error/status data to be moved to the processor and placed into the *ReadData* array, the value entered should be a module memory address in the Read Data area. If the value is set to **-1**, the error/status data will not be stored in the module's internal database and will not be transferred to the processor's *ReadData* array.

Enabling the *Error/Status Pointer* is optional. The error/status data is routinely returned as part of the input image, which is continually being transferred from the module to the processor. For more information, see Normal Data Transfer Blocks (page 104).

<u>Read Register Start</u>

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.

<u>Read Register Count</u>

0 to **5000**

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

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

Write Register Start

0 to **4999**

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

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

Write Register Count

0 to **5000**

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

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

Failure Flag Count

0 through **65535**

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

Block Transfer Size

60, **120** or **240**

This read-only parameter specifies the number of words in each block transferred between the module and processor.

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.

2.5.2 MNET Client x

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

Client Error/Status Pointer

-1 to **4990**

This parameter sets the address in the internal database where the Client error/status data will be placed. If you want the error/status data to be moved to the processor and placed into the *ReadData* array, the value entered should be a module memory address in the Read Data area. If the value is set to **-1**, the error/status data will not be stored in the module's internal database and will not be transferred to the processor's *ReadData* array.

Enabling the *Error/Status Pointer* is optional. Alternatively, the error/status data for a specific Client can be requested by the processor and returned in a special Client Status block. For more information, see Client Status Blocks (page 113).

Command Error Pointer

-1 to **4999**

This parameter sets the address in the internal database where the Command Error List data will be placed. If you want the Command Error List data to be moved to the processor and placed into the *ReadData* array, 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.

Enabling the *Command Error Pointer* is optional. Alternatively, the Command Error List data for a specific Client can be requested by the processor and returned in a special Client Status block. For more information, see Client Status Blocks (page 113).

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 on the type of communication network used, and the expected response time of the slowest device on the network.

Retry Count

0 to **10**

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

Float Flag

YES or **NO**

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

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

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

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

Float Start

0 to **65535**

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

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

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

Float Offset

0 to **9999**

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

ARP Timeout

1 to **60**

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

Command Error Delay

0 to **300**

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

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.

2.5.3 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 MVI69-MNETC 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 issuances of a command. If the user specifies a value of **10** for the parameter, the command will be executed no more frequently than every 1 second.

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.

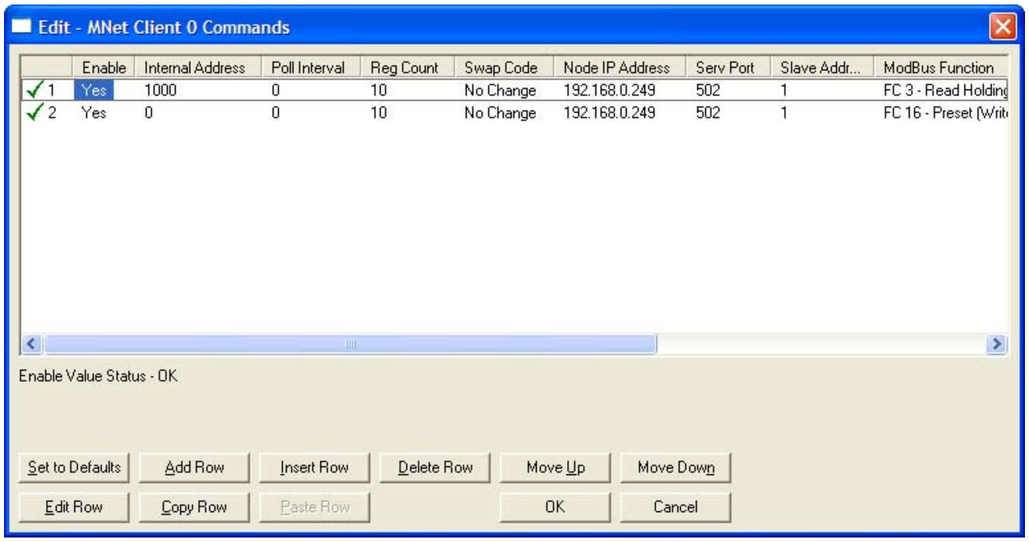
Command Entry Formats

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 MVI69 module, and the second part contains information required to interface to the server device.

Command list example:



	Enable	Internal Address	Poll Interval	Reg Count	Swap Code	Node IP Address	Serv Port	Slave Addr...	ModBus Function
✓ 1	Yes	1000	0	10	No Change	192.168.0.249	502	1	FC 3 - Read Holding
✓ 2	Yes	0	0	10	No Change	192.168.0.249	502	1	FC 16 - Preset (Writ

Enable Value Status - OK

Enable

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

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 (0)	The command is executed each scan of the command list if the Poll Interval Time is set to zero (0). If the Poll Interval time is set, the command will be executed when the interval timer expires.
Conditional (2)	The command executes only if the internal data associated with the command changes.

Important: The commands must also be enabled in the ladder logic in order for them to be executed. The *MNETC.CONTROL.CmdControl.WriteCmdBits[x]* controller tag array holds 16-command bit arrays for each Client. If a bit for a specific command is set to zero (0) in the *WriteCmdBits[x]* controller tag, the command will not be executed, regardless of its enabled or disabled state in the configuration. For more information, see Command Control Blocks (page 114).

Internal Address

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

or

0 to **4999** (for word-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-level or register-level address.

Poll Interval

0 to **65535**

This parameter specifies the minimum interval between issuances of a command during continuous command execution (*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 port numbers supported on a 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.

2.5.4 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 MVI69-MNETC 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.

2.5.5 Ethernet Configuration

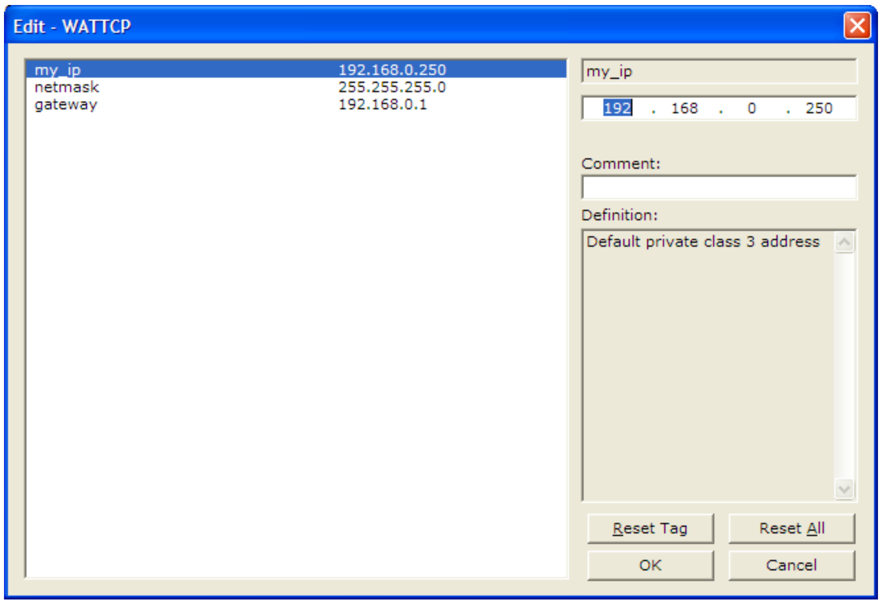
Use this procedure to configure the Ethernet settings for your module. You must assign an IP address, subnet mask and gateway address. After you complete this step, you can connect to the module with an Ethernet cable.

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.



Edit - WATTCP

my_ip	192.168.0.250
netmask	255.255.255.0
gateway	192.168.0.1

my_ip

192 . 168 . 0 . 250

Comment:

Definition:

Default private class 3 address

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

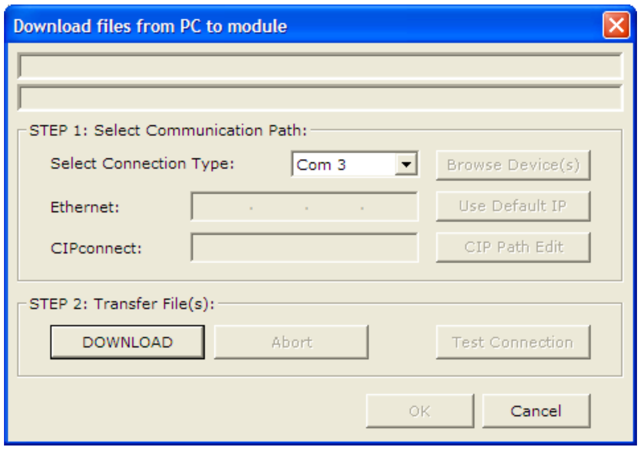
4 When you are finished editing, click **OK** to save your changes and return to the *ProSoft Configuration Builder* window.

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

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

1 In the tree view in *ProSoft Configuration Builder*, click once to select the module.

2 Open the *Project* menu, and then choose **MODULE/DOWNLOAD**. The program will scan your PC for a valid com port (this may take a few seconds). When *PCB* has found a valid COM port, the *Download* dialog box will open.


Download files from PC to module

STEP 1: Select Communication Path:

Select Connection Type: Com 3

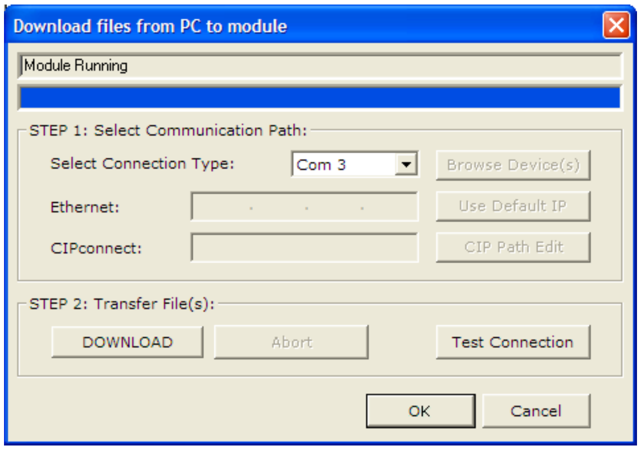
Ethernet:

CIPconnect:

STEP 2: Transfer File(s):

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

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


Download files from PC to module

Module Running

STEP 1: Select Communication Path:

Select Connection Type: Com 3

Ethernet:

CIPconnect:

STEP 2: Transfer File(s):

3 Ladder Logic

In This Chapter

Ladder logic is required for managing communication between the MVI69-MNETC 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 commented to provide information on the purpose and function of the user-defined data types and controller tags. For most applications, the Import Rung with Add-On Instruction will work without modification.

3.1 Controller Tags

Data related to the MVI69-MNETC is stored in the ladder logic in variables called controller tags. Individual controller tags can be grouped into collections of controller tags called controller tag structures. A controller tag structure can contain any combination of:

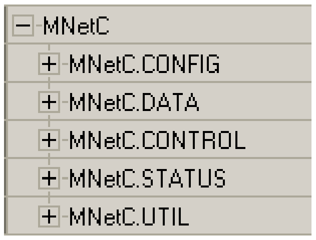
- Individual controller tags
- Controller tag arrays
- Lower-level controller tag structures

The controller tags for the module are pre-programmed into the Add-On Instruction Import Rung ladder logic. You can find them in the *Controller Tags* subfolder, located in the *Controller* folder in the *Controller Organizer* pane of the main RSLogix 5000 window.

This controller tag structure is arranged as a tree structure. Individual controller tags are found at the lowest level of the tree structure. Each individual controller tag is defined to hold data of a specific type, such as integer or floating-point data. Controller tag structures are declared with user-defined data types, which are collections of data types.

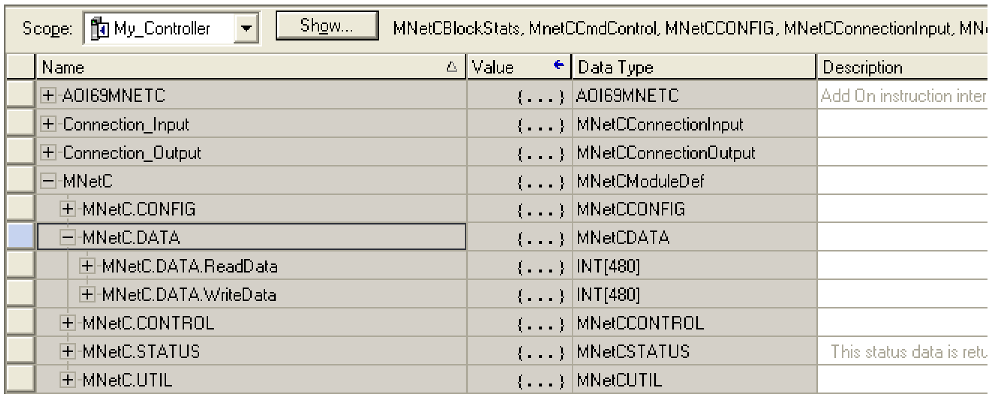
3.1.1 MVI69-MNETC Controller Tags

The main controller tag structure, *MNetC*, is broken down into five lower-level controller tag structures.



The five lower-level controller tag structures contain other controller tags and controller tag structures. Click the **[+]** sign next to any controller tag structure to expand it and view the next level in the structure.

For example, if you expand the *MNetC.DATA* controller tag structure, you will see that it contains two controller tag arrays, *MNetC.DATA.ReadData* and *MNetC.DATA.WriteData*, which are 480-element integer arrays by default.



Scope: My_Controller Show... MNetCBlockStats, MnetCCmdControl, MNetCCONFIG, MNetCConnectionInput, MN

Name	Value	Data Type	Description
+ AOI69MNETC	{...}	AOI69MNETC	Add On instruction inter
+ Connection_Input	{...}	MNetCConnectionInput	
+ Connection_Output	{...}	MNetCConnectionOutput	
− MNetC	{...}	MNetCModuleDef	
+ MNetC.CONFIG	{...}	MNetCCONFIG	
− MNetC.DATA	{...}	MNetCDATA	
+ MNetC.DATA.ReadData	{...}	INT[480]	
+ MNetC.DATA.WriteData	{...}	INT[480]	
+ MNetC.CONTROL	{...}	MNetCCONTROL	
+ MNetC.STATUS	{...}	MNetCSTATUS	This status data is retu
+ MNetC.UTIL	{...}	MNetCUTIL	

Notice that the *Data Type* column displays the data types used to declare each controller tag, controller tag array or controller tag structure. Individual controller tags are declared with basic data types, such as INT and BOOL. Controller tag arrays are declared with arrays of basic data types. Controller tag structures are declared with user-defined data types (UDTs).

3.2 User-Defined Data Types (UDTs)

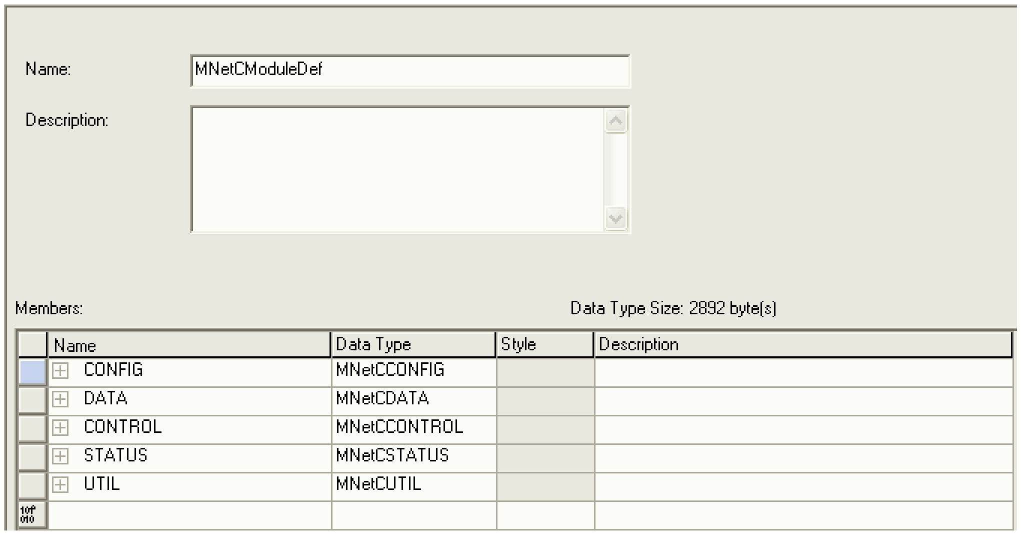
User-defined data types (UDTs) allow users to organize collections of data types into groupings. These groupings, or data type structures, can then be used to declare the data types for controller tag structures. Another advantage of defining a UDT is that it may be re-used in other controller tag structures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

The Add-On Instruction Import Rung ladder logic for the module has pre-defined UDTs. You can find them in the *User-Defined* subfolder, located in the *Data Types* folder in the *Controller Organizer* pane of the main RSLogix window. Like the controller tags, the UDTs are organized in a multiple-level tree structure.

3.2.1 MVI69-MNETC User-Defined Data Types

Fourteen different UDTs are defined for the MVI69-MNETC Add-On Instruction.

The main UDT, *MNetCModuleDef*, contains all the data types for the module and was used to create the main controller tag structure, *MNetC*. There are five UDTs one level below *MNetCModuleDef*. These lower-level UDTs were used to create the *MNetC.CONFIG*, *MNetC.DATA*, *MNetC.CONTROL*, *MNetC.STATUS*, and *MNetC.UTIL* controller tags structures.



Name: MNetCModuleDef

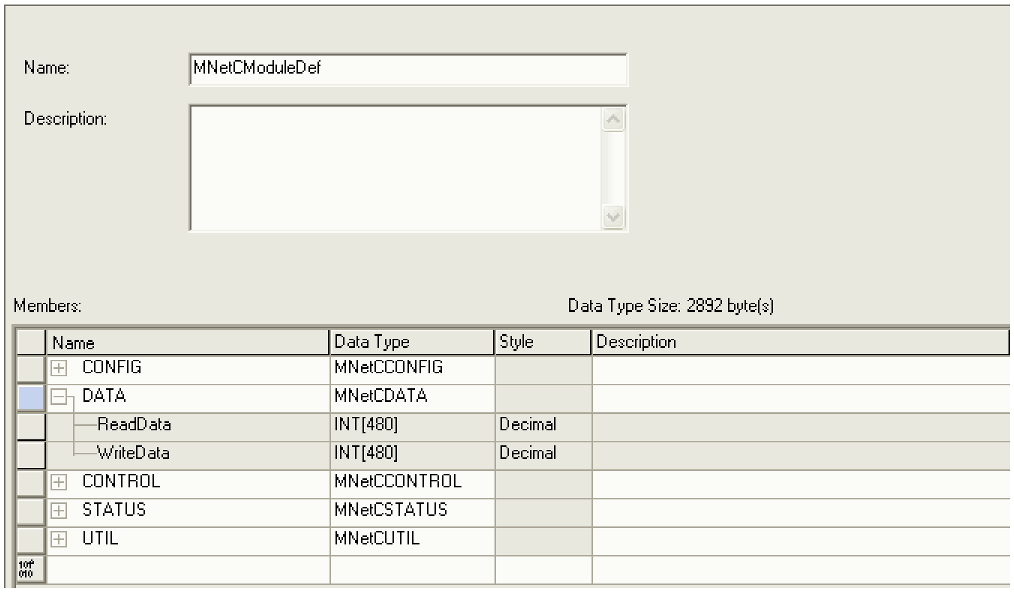
Description:

Members: Data Type Size: 2892 byte(s)

Name	Data Type	Style	Description
⊞ CONFIG	MNetCCONFIG		
⊞ DATA	MNetCDATA		
⊞ CONTROL	MNetCCONTROL		
⊞ STATUS	MNetCSTATUS		
⊞ UTIL	MNetCUTIL		

Click the **[+]** signs to expand the UDT structures and view lower-level UDTs.

For example, if you expand *MNetCDATA*, you will see that it contains two UDTs, *ReadData* and *WriteData*. Both of these are 480-element integer arrays by default.



Name: MNetCModuleDef

Description:

Members: Data Type Size: 2892 byte(s)

Name	Data Type	Style	Description
⊞ CONFIG	MNetCCONFIG		
⊟ DATA	MNetCDATA		
ReadData	INT[480]	Decimal	
WriteData	INT[480]	Decimal	
⊞ CONTROL	MNetCCONTROL		
⊞ STATUS	MNetCSTATUS		
⊞ UTIL	MNetCUTIL		

Notice that these UDTs are the data types declared for the *MNetC.DATA.ReadData* and *MNetC.DATA.WriteData* controller tag arrays.

3.3 Using Controller Tags

You can use controller tags to:

- View read and write data that is being transferred between the module and the processor.
- View status data for the module.
- Set up and trigger special functions.
- Initiate module restarts (Warm Boot or Cold Boot).

3.4 Controller Tag Overview

Controller Tag	Description
MNetC.CONFIG	Sets the block transfer size
MNetC.DATA	MNET input and output data transferred between the processor and the module
MNetC.CONTROL	Governs the data movement between the PLC rack and the module
MNetC.STATUS	Status information
MNetC.UTIL	Block statistics and generic tags used for internal ladder processing (DO NOT MODIFY)

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

3.4.1 MNetC.CONFIG

Controller Tag	Data Type	Description
BlockTransferSize	INT	Must match the configuration parameter *Block Transfer Size* (60, 120, 240). This value must be set in the ladder logic by the user.

3.4.2 MNetC.DATA

The controller tags in *MNetC.DATA* hold user data to be transferred between the processor and the MVI69-MNETC module. This read and write data is transferred between the processor and the module as "pages," or blocks, of data. The size of the blocks depends on the *Block Transfer Size* parameter (60, 120 or 240 words), set in the configuration and the ladder logic. For example, if the *Block Transfer Size* is set to 60 words, then the user data in the Read and Write areas is transferred between the module and the processor 60 words at a time, in 60-word blocks.

The data types for the *MNetC.DATA.ReadData and MNetCDATA.WriteData* controller tag arrays are integer arrays containing a variable number of elements.

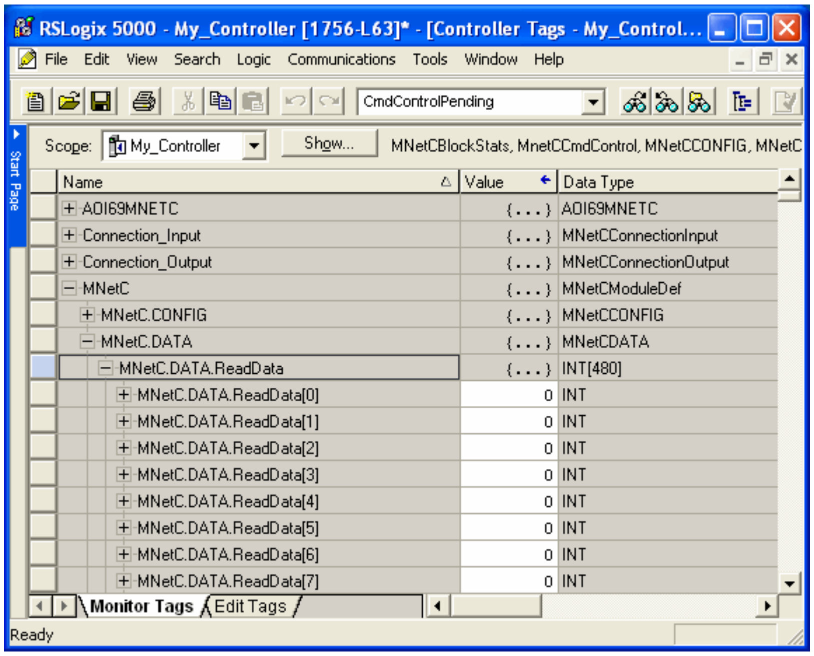
Controller Tag	Data Type	Description
ReadData	INT[x]	Data read from module. Array size must be set to equal the *Read Register Count* in the configuration.
WriteData	INT[x]	Data to write to module. Array size must be set to equal the *Write Register Count* in the configuration.

MNETCR.DATA.ReadData

The size of the *ReadData* controller tag array should be set by the user to match the value entered in the *Read Register Count* (page 48) parameter of the configuration. For ease of use, the size of the module's Read Data area and the processor's *ReadData* array should both be dimensioned as a multiple of the *Block Transfer Size*.

Example

If the *Block Transfer Size* is set to 120 words, then a Read area size of 480 would allow all the Read data to be transferred over to the processor in exactly 4 Read blocks containing 120 words each. The ladder logic places the received data into the proper position in the *ReadData* array, which should also be dimensioned to 480 elements.


RSLogix 5000 - My_Controller [1756-L63]* - [Controller Tags - My_Control...
File Edit View Search Logic Communications Tools Window Help
CmdControlPending
Scope: My_Controller
Show...
MNetCBlockStats, MnetCCmdControl, MNetCCONFIG, MNetC
Monitor Tags
Edit Tags
Ready

Name	Value	Data Type
+ AOI69MNETC	{...}	AOI69MNETC
+ Connection_Input	{...}	MNetCConnectionInput
+ Connection_Output	{...}	MNetCConnectionOutput
− MNetC	{...}	MNetCModuleDef
+ MNetC.CONFIG	{...}	MNetCCONFIG
− MNetC.DATA	{...}	MNetCDATA
− MNetC.DATA.ReadData	{...}	INT[480]
+ MNetC.DATA.ReadData[0]	0	INT
+ MNetC.DATA.ReadData[1]	0	INT
+ MNetC.DATA.ReadData[2]	0	INT
+ MNetC.DATA.ReadData[3]	0	INT
+ MNetC.DATA.ReadData[4]	0	INT
+ MNetC.DATA.ReadData[5]	0	INT
+ MNetC.DATA.ReadData[6]	0	INT
+ MNetC.DATA.ReadData[7]	0	INT

The *ReadData* array is related to the contents of the Read Data area of the module's internal database. To view the actual registers in the module's internal database, access the database display from *ProSoft Configuration Builder's Diagnostics* menu. For more information, see the section on *PCB* Diagnostics (page 86).

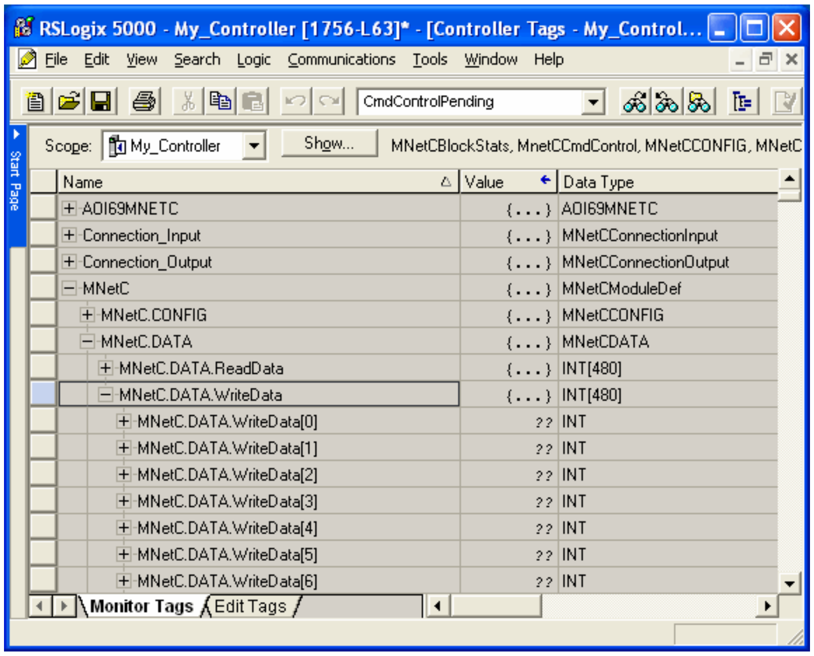
```
DATABASE DISPLAY 0 TO 99 (DECIMAL)

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

MNETCR.DATA.WriteData

The size of the *WriteData* controller tag array should be set by the user to match the value entered in the *Write Register Count* (page 49) parameter of the configuration. For ease of use, the size of the module's Write Data area and the processor's *WriteData* array should both be dimensioned as a multiple of the *Block Transfer Size*.

Example

If the *Block Transfer Size* is set to 120 words, then a Write area size of 480 would allow all the Write data to be transferred over to the processor in exactly 4 Write blocks containing 120 words each. The ladder logic places the received data into the proper position in the *WriteData* array, which should also be dimensioned to 480 elements.


RSLogix 5000 - My_Controller [1756-L63]* - [Controller Tags - My_Control...
File Edit View Search Logic Communications Tools Window Help
CmdControlPending
Start Page
Scope: My_Controller
Show...
MNetCBlockStats, MnetCCmdControl, MNetCCONFIG, MNetC
Monitor Tags
Edit Tags

Name	Value	Data Type
+ AOI69MNETC	{...}	AOI69MNETC
+ Connection_Input	{...}	MNetCConnectionInput
+ Connection_Output	{...}	MNetCConnectionOutput
− MNetC	{...}	MNetCModuleDef
+ MNetC.CONFIG	{...}	MNetCCONFIG
− MNetC.DATA	{...}	MNetCDATA
+ MNetC.DATA.ReadData	{...}	INT[480]
− MNetC.DATA.WriteData	{...}	INT[480]
+ MNetC.DATA.WriteData[0]	??	INT
+ MNetC.DATA.WriteData[1]	??	INT
+ MNetC.DATA.WriteData[2]	??	INT
+ MNetC.DATA.WriteData[3]	??	INT
+ MNetC.DATA.WriteData[4]	??	INT
+ MNetC.DATA.WriteData[5]	??	INT
+ MNetC.DATA.WriteData[6]	??	INT

The *WriteData* array is related to the contents of the Write Data area of the module's internal database. To view the actual registers in the module's internal database, access the database display from *ProSoft Configuration Builder's Diagnostics* menu. For more information, see the section on *PCB* Diagnostics (page 86).

```
DATABASE DISPLAY 1000 TO 1099 (DECIMAL)

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

3.4.3 MNetC.CONTROL

This controller tag structure is used to request special tasks from the module. For more information, see Special Function Blocks (page 109).

Controller Tag	Data Type	Description
BootTimer	TIMER	Timer used to clear both cold and warm boot requests
ColdBoot	BOOL	Hardware reset of the module
WarmBoot	BOOL	Configuration data reset in the module.
EventCmd	MNetCEventCmd[1]	Data structure to use for Event Commands
EventCmdTrigger	BOOL	Initiates Event Command block request
EventCmdPending	BOOL	Halts rung until module is ready
CmdControl	MnetCCmdControl	Data structure to use for Command Control
CmdControlPending	BOOL	Halts rung until module is ready
CmdControlTrigger	BOOL	Initiates Command Control block request
IPAdressControl	MNetCIPAddress	Data structure to set and get IP addresses
CmdBitPending	BOOL	Not used
CmdBitTrigger	BOOL	Not used

3.4.4 MNetC.STATUS

This controller tag structure contains module and Client status data. For a more complete description of the *MNetC.STATUS* controller tag structure, refer to the Status Data Definition (page 94).

Controller Tag	Data Type	Description
PassCnt	INT	Program cycle counter
Block	MNetCBlockStats	Block transfer statistics
MnetServer	MNetCPortStats	Not used
MBAPServer	MNetCPortStats	Not used
StatReqClientID	INT	Specifies Client (**0**-**29**) to get status data for when requesting Client Status block
CmdBits	INT[30]	Displays enabled or disabled status of all 16 commands in the *Client x Command List* for each Client.
ClientStats	MNetCPortStats[30]	Client status data
ClientStatsPending	BOOL	Halts rung until module is ready
ClientStatsTrigger	BOOL	Initiates Client Status block request
CmdErrorList	INT[16]	Command Error List

3.4.5 MNetC.UTIL

This controller tag structure stores the variables required for the data transfer between the processor and the MVI69-MNETC module.

Name	**Data Type**	**Description**
ReadDataSizeGet	INT	Gets *ReadData* array size
WriteDataSizeGet	INT	Gets *WriteData* array size
ReadDataBlkCount	INT	Holds the value of the block counts of the *ReadData* array
WriteDataBlkCount	INT	Holds the value of the block counts of the *WriteData* array
RBTSremainder	INT	The remainder of the 60, 120 or 240 block to transfer
WBTSremainder	INT	The remainder of the 60, 120 or 240 block to transfer
LastRead	INT	Index of last Read block
LastWrite	INT	Index of last Write block
BlockIndex	INT	Computed block offset for data table
StatusIndex	INT	Computed block offset for status data
InitOutputData	MNetCInitOutData	Used to bring the module into a known state after a restart operation

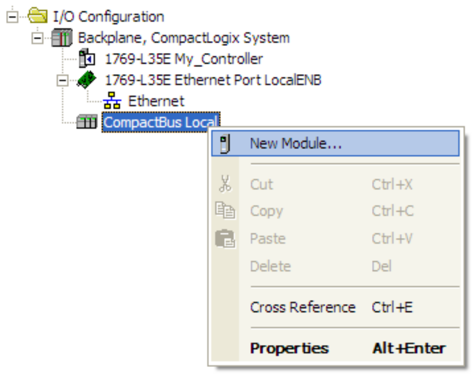
The *LastRead* tag stores the latest Read Block ID received from the module. The *LastWrite* tag stores the latest Write Block ID to be sent to the module. The *BlockIndex* tag is an intermediate variable used during the block calculation.

3.5 Adding the Module to an Existing CompactLogix Project

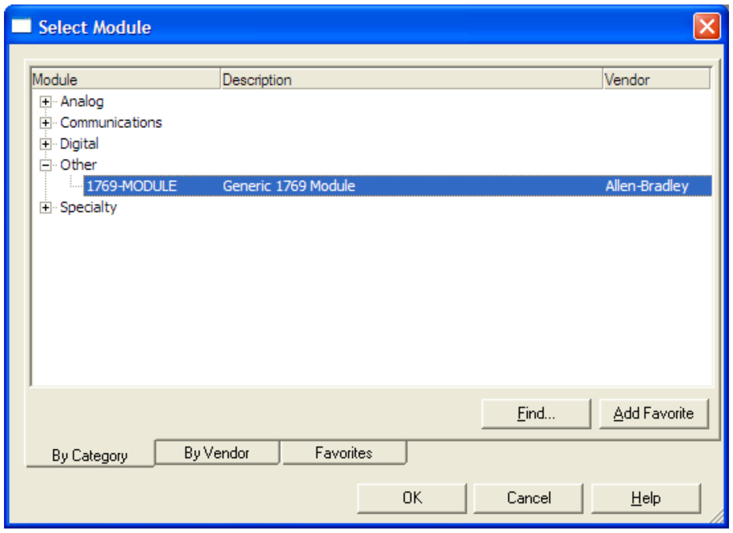
Important: The following steps describe how to install and configure the MVI69-MNETC module with RSLogix 5000 version 15 or older. If you are using RSLogix 5000 version 16, please refer to MVI69-MNETC Add-On Instruction Rung Import Procedure (page 22).
Important: The MVI69-MNETC module has a power supply distance rating of 2 (L43 and L45 installations on first 2 slots of 1769 bus).

If you are installing and configuring the module with a CompactLogix processor, follow these steps. If you are using a MicroLogix processor, refer to the next section.

1 Add the MVI69-MNETC module to the project. Right-click the mouse button on the **I/O CONFIGURATION** option in the *Controller Organization* window to display a pop-up menu. Select the **NEW MODULE** option from the **I/O CONFIGURATION** menu.

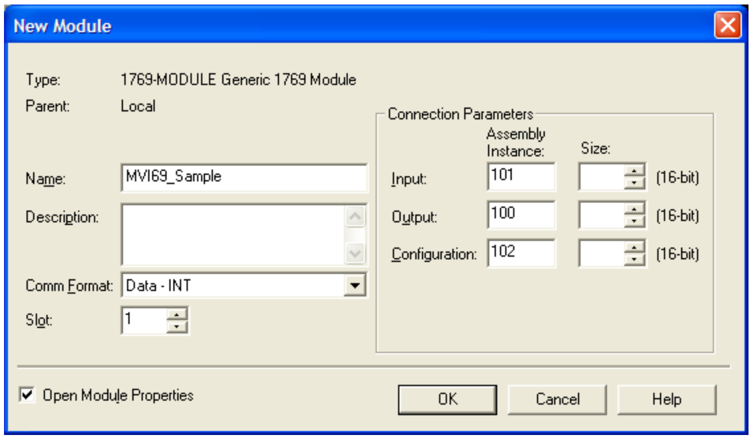


This action opens the following dialog box:



Module	Description	Vendor
Analog		
Communications		
Digital		
Other		
1769-MODULE	Generic 1769 Module	Allen-Bradley
Specialty		

2 Select the **1769-MODULE (GENERIC 1769 MODULE)** from the list and click **OK**.



Type: 1769-MODULE Generic 1769 Module
Parent: Local
Name: MVI69_Sample
Description:
Comm Format: Data - INT
Slot: 1
Input: 101
Output: 100
Configuration: 102

3 Enter the Name, Description and Slot options for your application, using the values in the illustration above. You must select the *Comm Format* as **DATA - INT** in the dialog box, otherwise the module will not communicate over the backplane of the CompactLogix rack.

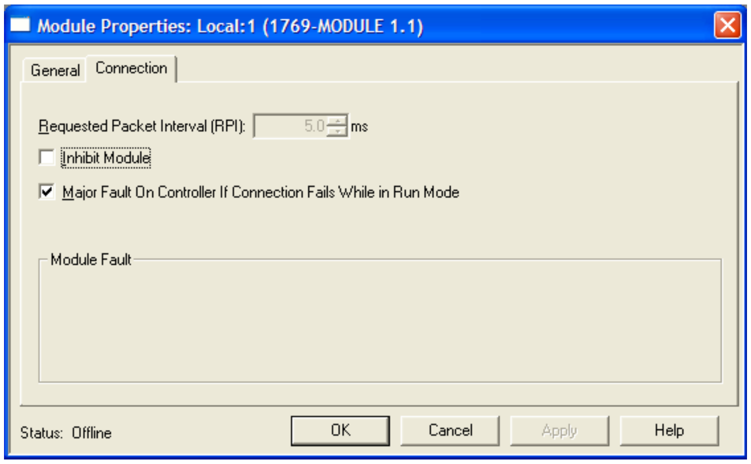
4 Configure the *Connection Parameters* to match to the *Block Transfer Size* parameter in the configuration. Use the values in the table corresponding with the block transfer size you configured.

Block Transfer Size = 60	
Field	**Recommended Value**
Type	1769-MODULE Generic 1769 Module
Parent	Local
Name	MVI69
Description	MVI69 Application Module
Comm Format	**DATA - INT**
Slot	The slot number in the rack where the module is installed
Input Assembly Instance	**101**
Input Size	**62**
Output Assembly Instance	**100**
Output Size	**61**
Configuration Assembly Instance	**102**
Configuration Size	**0**

Block Transfer Size = 120	
Field	**Recommended Value**
Type	1769-MODULE Generic 1769 Module
Parent	Local
Name	MVI69
Description	MVI69 Application Module
Comm Format	**DATA - INT**
Slot	The slot number in the rack where the module is installed
Input Assembly Instance	**101**
Input Size	**122**
Output Assembly Instance	**100**
Output Size	**121**
Configuration Assembly Instance	**102**
Configuration Size	**0**

Block Transfer Size = 240	
Field	**Recommended Value**
Type	1769-MODULE Generic 1769 Module
Parent	Local
Name	MVI69
Description	MVI69 Application Module
Comm Format	**DATA - INT**
Slot	The slot number in the rack where the module is installed
Input Assembly Instance	**101**
Input Size	**242**
Output Assembly Instance	**100**
Output Size	**241**
Configuration Assembly Instance	**102**
Configuration Size	**0**

5 Click **NEXT** to continue.

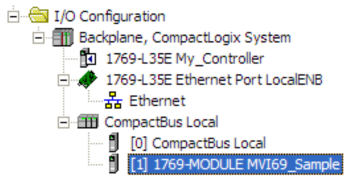

Module Properties: Local:1 (1769-MODULE 1.1)

Requested Packet Interval (RPI): 5.0 ms
☐ Inhibit Module
☑ Major Fault On Controller If Connection Fails While in Run Mode

Status: Offline

6 Select the *Request Packet Interval* value for scanning the I/O on the module. This value represents the minimum frequency the module will handle scheduled events. This value should not be set to less than **1** millisecond. Values between **1** and **10** milliseconds should work with most applications.

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



I/O Configuration
Backplane, CompactLogix System
1769-L35E My_Controller
1769-L35E Ethernet Port LocalENB
Ethernet
CompactBus Local
[0] CompactBus Local
[1] 1769-MODULE MVI69_Sample

8 Copy the user-defined data types (page 68) from the sample program.

9 Copy the controller tags (page 66) from the sample program.

10 Copy the Ladder Rungs from the sample program.

11 Save and download (page 40) the new application to the controller and place the processor in run mode.

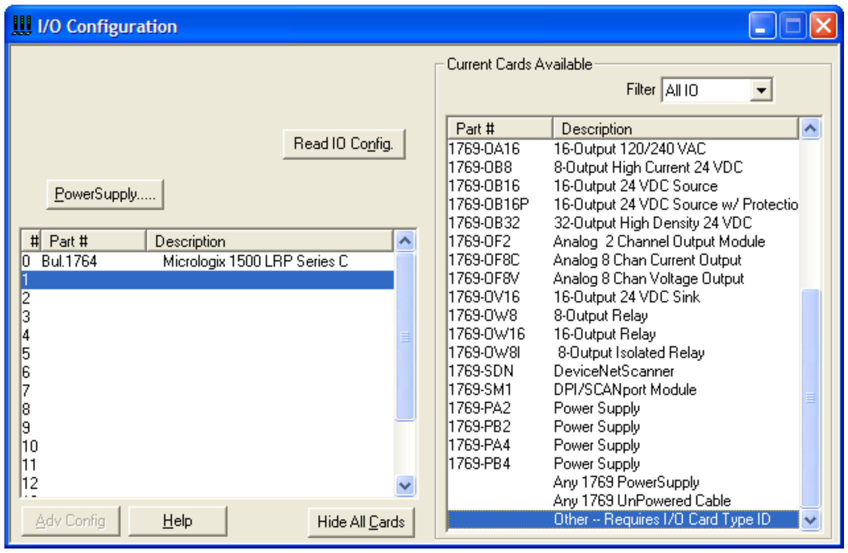
3.6 Adding the Module to an Existing MicroLogix Project

If you are installing and configuring the module with a MicroLogix controller, follow these steps. If you are using a CompactLogix controller, refer to the previous section.

The first step in setting up the processor ladder file is to define the I/O type module to the system. Start RSLogix 500, and follow these steps:

1 In RSLogix, open your existing application, or start a new application, depending on your requirements.

2 Double-click the **I/O CONFIGURATION** icon located in the *Controller* folder in the project tree. This action opens the *I/O Configuration* dialog box.

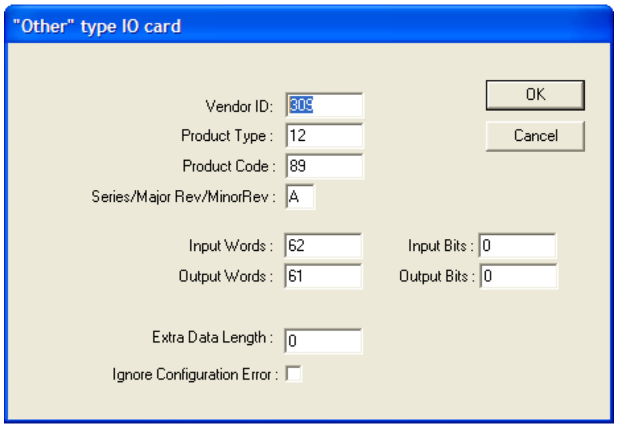


#	Part #	Description
0	Bul.1764	Micrologix 1500 LRP Series C
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

Part #	Description
1769-OA16	16-Output 120/240 VAC
1769-OB8	8-Output High Current 24 VDC
1769-OB16	16-Output 24 VDC Source
1769-OB16P	16-Output 24 VDC Source w/ Protectio
1769-OB32	32-Output High Density 24 VDC
1769-OF2	Analog 2 Channel Output Module
1769-OF8C	Analog 8 Chan Current Output
1769-OF8V	Analog 8 Chan Voltage Output
1769-OV16	16-Output 24 VDC Sink
1769-OW8	8-Output Relay
1769-OW16	16-Output Relay
1769-OW8I	8-Output Isolated Relay
1769-SDN	DeviceNetScanner
1769-SM1	DPI/SCANport Module
1769-PA2	Power Supply
1769-PB2	Power Supply
1769-PA4	Power Supply
1769-PB4	Power Supply
	Any 1769 PowerSupply
	Any 1769 UnPowered Cable
	Other -- Requires I/O Card Type ID

3 In the *I/O Configuration* dialog box, select "**OTHER - REQUIRES I/O CARD TYPE ID**" at the bottom of the list in the right pane, and then double-click to open the *"Other" type IO card* dialog box.

4 Enter the values shown in the following illustration to define the module correctly for the MicroLogix processor, and then click **OK** to save your configuration.



Vendor ID: 309
Product Type : 12
Product Code : 89
Series/Major Rev/MinorRev : A
Input Words : 62
Output Words : 61
Input Bits : 0
Output Bits : 0
Extra Data Length : 0
Ignore Configuration Error : ☐

The *Input Words* and *Output Words* parameters will depend on the *Block Transfer Size* parameter you specify in the configuration file. Use the values from the following table.

Block Transfer Size	**Input Words**	**Output Words**
60	62	61
120	122	121
240	242	241

5 Click **OK** to continue.
6 After completing the module setup, the *I/O Configuration* dialog box will display the module's presence.

The last step is to add the ladder logic. If you are using the example ladder logic, adjust the ladder to fit your application.

Download the new application to the controller and place the processor in RUN mode. If you encounter errors, refer to Diagnostics and Troubleshooting (page 81) for information on how to connect to the module's Config/Debug port to use its troubleshooting features.

4 Diagnostics and Troubleshooting

In This Chapter

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

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

4.1 LED Status Indicators

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

LED	Color	Status	Indication
CFG	Green	ON	Data is being transferred between the module and a remote terminal using the Configuration/Debug port.
		OFF	No data is being transferred on the Configuration/Debug port.
EP	Green	ON	Indicator not used
		OFF	Indicator not used
APP	Amber	OFF	The MVI69-MNETC is working normally.
		ON	The MVI69-MNETC module program has recognized a communication error with a server on the network.
BP	Amber	ON	The LED is ON when the module is performing a write operation on the backplane.
		OFF	The LED is off when the module is performing a read operation on the backplane. Under normal operation, the LED should blink rapidly ON and OFF.
OK	Red/ Green	OFF	The card is not receiving any power and is not securely plugged into the rack.
		GREEN	The module is operating normally.
		RED	The program has detected an error or is being configured. If the LED remains RED for over 10 seconds, the program has probably halted. Remove the card from the rack and re-insert the card to restart the module's program.
BAT	Red	OFF	The battery voltage is OK and functioning.
		ON	The battery voltage is low or battery is not present. Allow battery to charge by keeping module plugged into rack for 24 hours. If BAT LED still does not go OFF, contact ProSoft Technology, as this is not a user serviceable item.

If the APP, BP ACT and OK LEDs blink at a rate of every one-second, call ProSoft Technology support. There is a serious problem with the module, and it will have to be sent back to ProSoft.

4.1.1 Ethernet LED Indicators

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

4.1.2 Clearing a Fault Condition

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

To clear the condition, follow these steps:

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

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

4.1.3 Troubleshooting

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

Processor Errors

Problem description	**Steps to take**
Processor fault	Verify that the module is plugged into the slot that has been configured for the module in the I/O Configuration of RSLogix. Verify that the slot location in the rack has been configured correctly in the ladder logic.
Processor I/O LED flashes	This indicates a problem with backplane communications. A problem could exist between the processor and any installed I/O module, not just the MVI69-MNETC. Verify that all modules in the rack are correctly configured in the ladder logic.

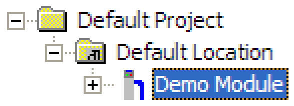
Module Errors

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: ▪ The processor is in RUN or REM RUN mode. ▪ The backplane driver is loaded in the module. ▪ The module is configured for read and write data block transfer. ▪ The ladder logic handles all read and write block situations. ▪ The module is properly configured in the processor I/O configuration and ladder logic.
OK LED remains RED	The program has halted or a critical error has occurred. Connect to the Configuration/Debug port to see if the module is running. If the program has halted, turn off power to the rack, remove the card from the rack and re-insert it, and then restore power to the rack.

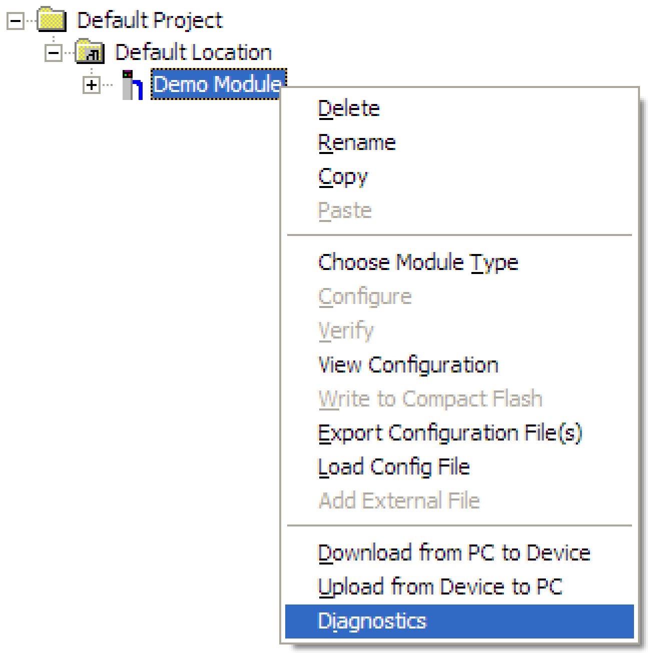
4.2 Using the Diagnostic Window in ProSoft Configuration Builder

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

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

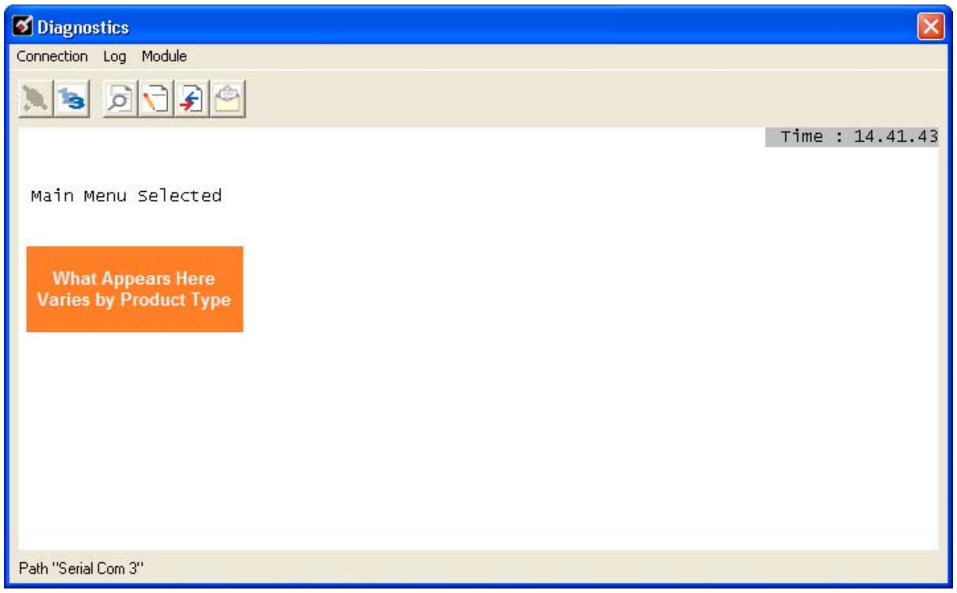


2 On the shortcut menu, choose **DIAGNOSTICS.**



This action opens the *Diagnostics* dialog box.

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

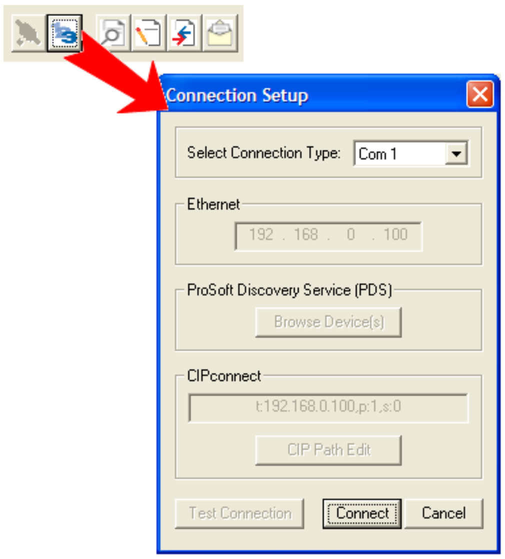


```
Time : 14.41.43

Main Menu Selected
```

Path "Serial Com 3"

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

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



Select Connection Type: Com 1

192 . 168 . 0 . 100

t:192.168.0.100,p:1,s:0

2 Verify that the null modem cable is connected properly between your computer's serial port and the module. A regular serial cable will not work.

3 On computers with more than one serial port, verify that your communication program is connected to the same port that is connected to the module.

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

4.2.1 Using ProSoft Configuration Builder (PCB) for Diagnostics

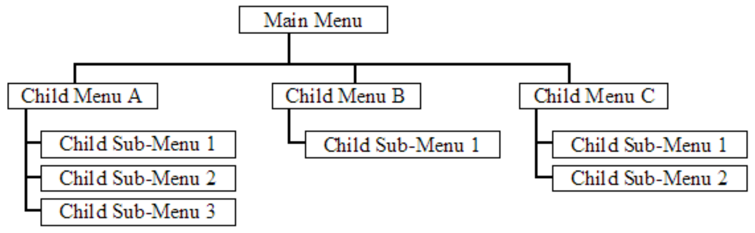
The *Configuration and Debug* menu for this module is arranged as a tree structure, with the *Main* menu at the top of the tree, and one or more submenus for each menu command. The first menu you see when you connect to the module is the *Main* menu.

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

Navigation

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

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



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

Keystrokes

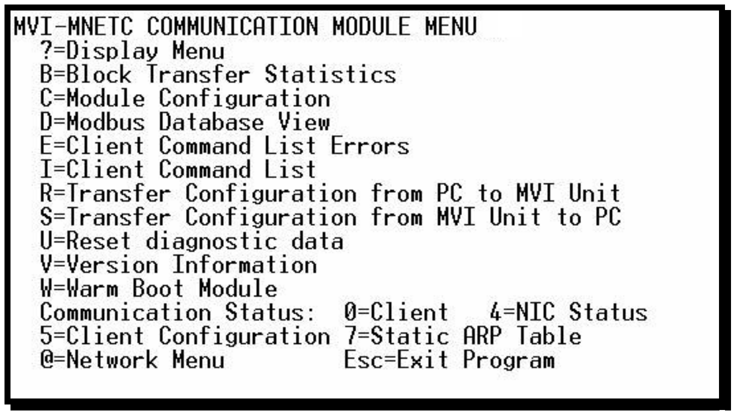
The keyboard commands on these menus are usually not case sensitive. You can enter most commands in lowercase or uppercase letters.

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

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

4.2.2 Main Menu

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



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

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

Viewing Block Transfer Statistics

Press **[B]** from the *Main* menu to view the *Block Transfer Statistics* screen.

Use this command to display the configuration and statistics of the backplane data transfer operations between the module and the processor. The information on this screen can help determine if there are communication problems between the processor and the module.

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

Viewing Module Configuration

Press **[C]** to view the *Module Configuration* screen.

Use this command to display the current configuration and statistics for the module.

Opening the Database View Menu

Press **[D]** to open the *Database View* menu.

Use this menu command to view the current contents of the module's database. For more information about this submenu, see Database View Menu (page 90).

Opening the Command Error List Menu

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

Opening the Command List Menu

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

Receiving the Configuration File

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

Sending the Configuration File

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

Resetting Diagnostic Data

Press **[U]** to reset the status counters for the Client and/or server(s) in the module.

Viewing Version Information

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

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

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

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

Warm Booting the Module

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

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

Viewing Client Status

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

Viewing NIC Status

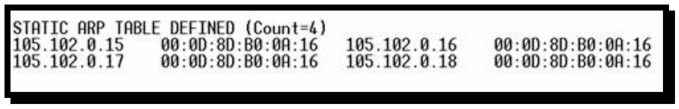
Press **[4]** to view NIC status. Use this command to view the communication status for the Network Interface Card.

Viewing Client Configuration

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

Viewing the Static ARP Table

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



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

Opening the Network Menu

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

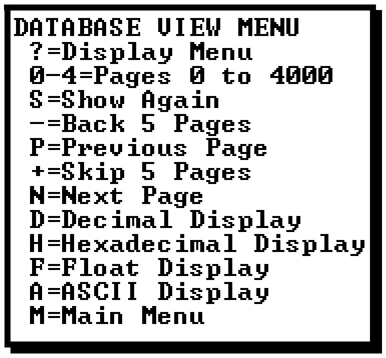
The *Network* menu allows you to send, receive and view the WATTCP.CFG file that contains the IP, gateway and other network specification information. For more information about this submenu, see Network Menu (page 92).

Exiting the Program

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

4.2.3 Modbus Database View Menu

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


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

All data contained in the module's database is available for viewing using the commands. Refer to the Modbus Protocol Specification (page 127) for information on the structure of Modbus messages. Each option available on the menu is discussed in the following topics.

Viewing Register Pages

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

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

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

Redisplaying the Current Page

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

Moving Back Through 5 Pages of Registers

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

Viewing the Previous Page of Registers

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

Moving Forward Through 5 Pages of Registers

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

Viewing the Next Page of Registers

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

Viewing Data in Decimal Format

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

Viewing Data in Hexadecimal Format

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

Viewing Data in Floating-Point Format

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

Viewing Data in ASCII (Text) Format

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

Returning to the Main Menu

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

4.2.4 Network Menu

From the *Main* menu press **[@]** to display the *Network* menu. The *Network* menu allows you to send, receive, and view the WATTCP.CFG file that contains the IP and module addresses, and other network information.

Transferring WATTCP.CFG to the Module

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

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

Transferring WATTCP.CFG to the PC

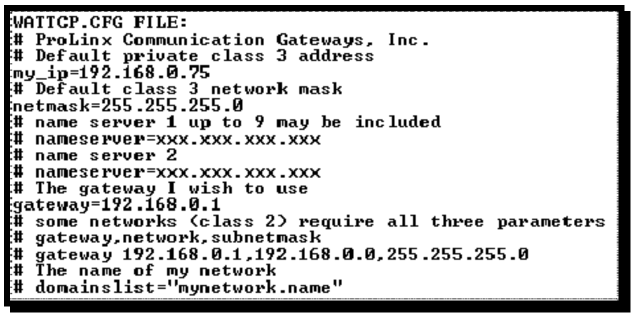
Press **[S]** to transfer the WATTCP.CFG file from the module to your PC.

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

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

Viewing the WATTCP.CFG File on the module

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



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

Returning to the Main Menu

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

4.3 Reading Status Data from the Module

Module status information is useful for troubleshooting and can be accessed in several different ways.

In the ladder logic's *MNETC.STATUS* controller tag structure.

The MVI69-MNETC module returns status data in the input image that can be used to determine the module's operating status. This data is transferred from the module to the processor continuously as part of the normal data transfer block sequence (page 104). You can view this data in the *MNETC.STATUS* controller tag structure in the ladder logic.

Client status data can also be requested and returned in a special Client Status block (page 113), outside of the normal data transfer block sequence. The status data contained in the Client Status block is different from the status data in the normal data transfer blocks. It can also be viewed in the *MNETC.STATUS* controller tag structure.

For more information about status data contained in *MNETC.STATUS*, see the Status Data Definition (page 94).

In *ProSoft Configuration Builder's Diagnostics* screens.

For more information, see the section on PCB Diagnostics (page 86).

In database locations specified by *Error/Status Pointers* (optional).

If optional *Error/Status Pointer*s are enabled, the status data can also be found in the Read Data area of the module's database at the locations specified by the pointer configuration parameters. For more information, see Backplane Error/Status Pointer (page 48), Client Error/Status Pointer (page 50) and Command Error Pointer (page 51).

Via the Configuration/Debug port.

Use a terminal program such as HyperTerminal. The Configuration/Debug port provides the following functionality:

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

4.3.1 Status Data Definition

This section contains a description of the controller tags in the *MNetC.STATUS* controller tag structure, which contains module and Client status data.

- The first seven and the eleventh controller tags contain status data routinely transferred from the module to the processor in the normal data transfer block sequence (page 104).
- The remaining controller tags are used to request and receive Client status data via the Client Status block functionality (page 113).

Note: In order to access up-to-date status data from these remaining controller tags, you must ensure that a Client Status block was recently received from the module. Client Status blocks are not routinely sent from the module; they are returned on a once-per-request basis as a response to a Client Status block request from the processor.

Controller Tag	Data Type	Description
PassCnt	INT	This value is incremented each time a complete program cycle occurs in the module.
Block.Read	INT	Total number of read blocks transferred from the module to the processor
Block.Write	INT	Total number of write blocks transferred from the processor to the module
Block.Parse	INT	Total number of blocks successfully parsed that were received from the processor
Block.Event	INT	Total number of command event blocks received from the processor
Block.Cmd	INT	Total number of command blocks received from the processor
Block.Err	INT	This field contains the total number of block errors recognized by the module.
MnetServer	MNetCPortStats	Not used
MBAPServer	MNetCPortStats	Not used
StatReqClientID	INT	Specifies Client (**0**-**29**) to get status data for when requesting Client Status block
CmdBits[x]	INT	Displays enabled or disabled status of all 16 commands in the *Client x Command List* for each Client. If the bit is set, the command will execute. If the bit is clear, the command will be disabled.
ClientStats[x].CmdReq	INT	Total number of command list requests sent from Client
ClientStats[x].CmdResp	INT	Total number of command list responses received by Client
ClientStats[x].CmdErr	INT	This value is incremented each time an error message is received from a remote unit or a local error is generated for a command.
ClientStats[x].Requests	INT	Not used
ClientStats[x].Responses	INT	Not used
ClientStats[x].ErrSent	INT	Not used
ClientStats[x].ErrRec	INT	Not used

Controller Tag	Data Type	Description
ClientStats[x].CfgErrWord	INT	Configuration Error Word - This word contains a bitmap that indicates general module configuration errors.
ClientStats[x].CurErr	INT	Most recent error code recorded for the Client
ClientStats[x].LastErr	INT	Previous most recent error code recorded for the Client
ClientStatsPending	BOOL	Temporary variable used to prevent a new Client Status block request from being sent to the module until the previously sent Client Status block request has been completely processed and a response block has been returned.
ClientStatsTrigger	BOOL	Initiates request for Client Status block from module when set to **1**
CmdErrorList[x]	INT	Command error code for each command (0-15) on the specified Client's command list

4.3.2 Configuration Error Word

The *Configuration Error Word* contains Client configuration error indications, in a bit-mapped format. Specific bits in the module's *Configuration Error Word* are turned on (set to 1) to indicate various configuration errors. The *Configuration Error Word* appears in the *MNETC.STATUS.ClientStats[x]* controller tag array.

Bits set to 1 in the *Configuration Error Word* indicate the following errors.

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

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

4.3.3 Client Command Errors

There are several different ways to view Client Command Errors.

- In the *MNETC.STATS.CmdErrorList* controller tag array
- On the Client status data screens in the *ProSoft Configuration Builder Diagnostics*
- At a module database location specified by the configuration's *MNET Client x Command Error Pointer*, if the *Command Error Pointer* is enabled. This means that the first register refers to command 1 and so on.

Word 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 commands in the Client Command Last are polled and executed, an error value is maintained in the module for each command. This error list can be transferred to the processor.

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 (same as -36)
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 (same as -11)
-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 Reference

In This Chapter

5.1 Product Specifications

The MVI69 Modbus TCP/IP Client Communication Module allows Rockwell Automation® CompactLogix or MicroLogix® processors to interface easily with other Modbus compatible devices.

Compatible devices include Modicon PACs, as well as a wide variety of instruments and devices. The module acts as an input/output module between the Modbus network and the Rockwell Automation backplane. The data transfer from the processor is asynchronous from the actions on the Modbus Client controlled network. A 5000-word register space in the module exchanges data between the processor and the Modbus network.

5.1.1 General Specifications

- Single-slot, 1769 backplane-compatible
- 10/100 MB Ethernet port
- Module I/O data memory mapping supports up to 5000 registers and is user definable
- ProSoft Configuration Builder (PCB) software supported, a Windows-based graphical user interface providing simple product and network configuration
- Sample Ladder Logic and Add-On Instructions (AOI) are used for data transfer between module and processor and module configuration
- Personality Module (non-volatile CF card) used to store module and network configuration allowing for in the field quick product replacement
- Supports CompactLogix processors with 1769 I/O bus capability and at least 800 mA of 5 Vdc backplane current available
- Also supports MicroLogix 1500 LRP

5.1.2 *Functional Specifications*

The MVI69-MNETC will operate on a local or expansion rack. This module was created to improve the performance when servers are not needed on a Modbus TCP/IP network. The module supports up to 30 Clients with up to 16 commands for each Client, making it easy to enable/disable the commands from the ladder logic. The Client command control word contains one bit for each command.

- 10/100 MB Ethernet Application port
- Supports Enron version of Modbus protocol for floating-point data transactions
- PCB includes a powerful Modbus network analyzer
- Special functions (Event Commands, Command Control, 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
- All data mapping begins at Modbus register 40001.
- Error codes, network error counters, and port status data available in user data memory

Client Specifications

A port configured as a virtual Modbus Client device on the MVI69-MNETC module actively issues Modbus commands to other nodes on the Modbus network. Additionally, the Client ports have an optimized polling characteristic that polls servers with communication problems less frequently. The CompactLogix or MicroLogix processor can be programmed to control the activity on the port by actively selecting commands from the command list to execute or issuing commands directly from the ladder logic.

5.1.3 Hardware Specifications

Specification	Description
Dimensions	Standard 1769 single-slot module
Current Load	800 mA max @ 5 Vdc Power supply distance rating of 2 (L43 and L45 installations on first 2 slots of 1769 bus)
Operating Temp.	0°C to 60°C (32°F to 140°F)
Storage Temp.	-40°C to 85°C (-40°F to 185°F)
Relative Humidity	5% to 95% (without condensation)
LED Indicators	Power and Module Status Application Status CFG Port Activity Ethernet Port Activity Error Status
CFG Port (CFG)	RJ45 (DB-9M with supplied cable) RS-232 only No hardware handshaking
App Port (Ethernet modules)	10/100 Base-T Ethernet compatible interface Electrical Isolation 1500 Vrms at 50 Hz to 60 Hz for 60 s, applied as specified in section 5.3.2 of IEC 60950: 1991 Ethernet Broadcast Storm Resiliency = less than or equal to 5000 [ARP] frames-per-second and less than or equal to 5 minutes duration
Shipped with Unit	RJ45 to DB-9M cables for each port 6-foot RS-232 configuration cable

5.2 Functional Overview

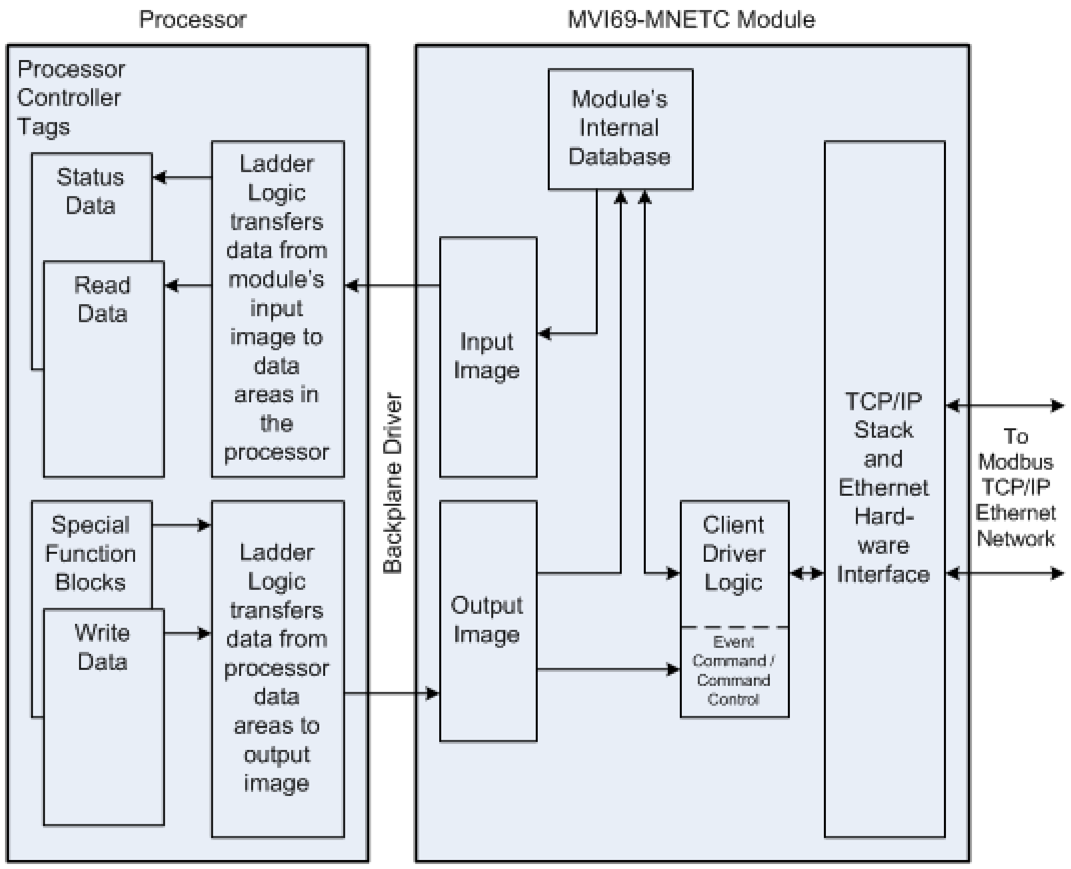
5.2.1 Backplane Data Transfer

The MVI69-MNETC module communicates directly over the processor’s backplane. Data is paged between the module and the processor across the backplane using the module's input and output images. The update frequency of the images is determined by the scheduled scan rate defined by the user for the module and the communication load on the module. Typical update times range from 2 to 10 milliseconds.

This bi-directional transfer of data is accomplished by the module putting data in the input image to send to the processor. Data in the input image is placed in the processor's controller tags by ladder logic. The input image for the module may be configured to 62, 122, or 242 words with the *Block Transfer Size* parameter.

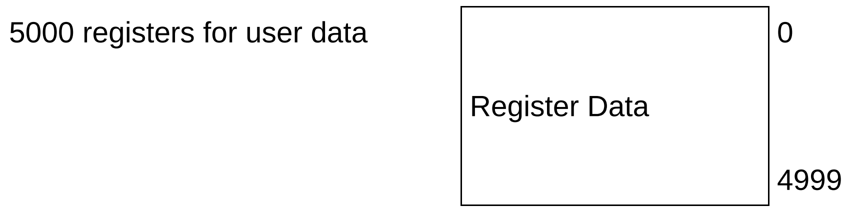
Processor logic inserts data to the output image to be transferred to the module. The module's firmware program extracts the data and places it in the module's internal database. The output image for the module may configured to 61, 121, or 241 words with the *Block Transfer Size* parameter.

The following illustration shows the data transfer method used to move data between the processor, the MVI69-MNETC module and the Modbus TCP/IP network.


Processor
MVI69-MNETC Module
Processor Controller Tags
Status Data
Read Data
Ladder Logic transfers data from module's input image to data areas in the processor
Special Function Blocks
Write Data
Ladder Logic transfers data from processor data areas to output image
Backplane Driver
Module's Internal Database
Input Image
Output Image
Client Driver Logic
Event Command / Command Control
TCP/IP Stack and Ethernet Hard-ware Interface
To Modbus TCP/IP Ethernet 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 processor to interface the input and output image data with data contained in the controller tags. All data used by the module is stored in its internal database. This database is defined as a virtual Modbus data table with addresses from 0 (40001 Modbus) to 4999 (45000 Modbus).

Module's Internal Database Structure



Data contained in this database is transferred in blocks, or pages, using the input and output images. Processor ladder logic and the MVI69-MNETC module's program work together to coordinate these block transfers. The block structure of each block type depends on the data content and the data transfer function to be performed by the block. The module uses the following block identification numbers.

Block ID Range	Descriptions
-1 or 0	Status block
1 to 84	Read or Write blocks
1000 to 1083	Initialize Output Data blocks
2000 to 2029	Event Command blocks
3000 to 3029	Client Status blocks
5001 to 5016	Command Control blocks
9990	Set New Module IP Address block
9991	Get Module IP Address block
9998	Warm Boot block
9999	Cold Boot block

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

Normal data transfer blocks

- Read, Write and Status blocks (-1 to 84)

Special function blocks

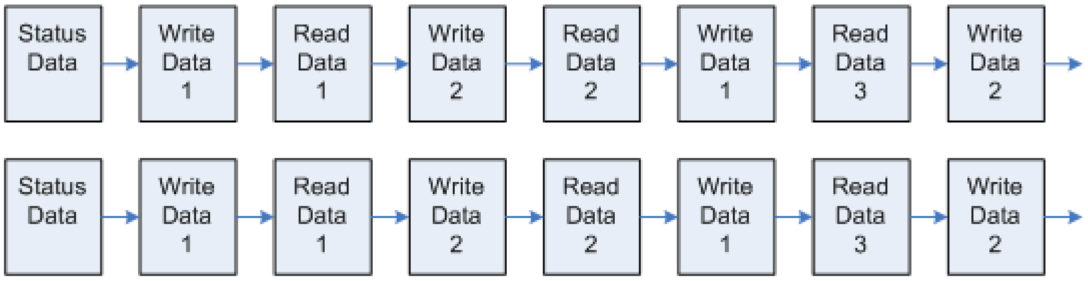
- Initialize Output Data blocks (1000 to 1083)
- Event Command blocks (2000 to 2029)
- Client Status blocks (3000 to 3029)
- Command Control blocks (5001 to 5016)
- IP Address blocks (9990 and 9991)
- Warm Boot and Cold Boot blocks (9998 and 9999)

Normal Data Transfer Blocks

Normal data transfer includes the paging of user data between the processor's data areas and the module's internal database (registers 0 to 4999), as well as the paging of status data. These data are transferred through Read (input image), Write (output image) and Status blocks. The amount of data paged in each block depends on the *Block Transfer Size* set by the user in the *MNETC.CONFIG.BlockTransferSize* controller tag (possible values are 60, 120, or 240).

During normal program operation, the module sequentially sends Read and Status Data blocks and receives Write blocks. The Status block is first in the sequence, followed by alternating Write and Read blocks.

As an example, assume that an application's Read Data area consists of 180 words and its Write Data area consists of 120 words, and that the user has set the *Block Transfer Size* to 60. Since the Read and Write data is paged 60 words at a time, the module will use 3 Read blocks and 2 Write blocks to transfer the data. The Read, Write and Status blocks will be sequenced as follows.



This sequence will continue until interrupted by other special function blocks sent by the processor, by a command request from a node on the Modbus network, or by operator control through the module's Configuration/Debug port.

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

Read Block

These blocks of data transfer information from the module to the processor. The structure of the input image used to transfer this data is shown below:

Read Block from Module to Processor

Word Offset	Description	Length (words)
0	Read Block ID	1
1	Write Block ID	1
2 to (*n* + 1)	Read Data	*n*

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

The Read Block ID is an index value used to determine where the data from module memory will be placed in the *ReadData[x]* controller tag array of the processor. The number of data words per transfer depends on the configured *Block Transfer Size* parameter in the configuration file (possible values are 60, 120, or 240).

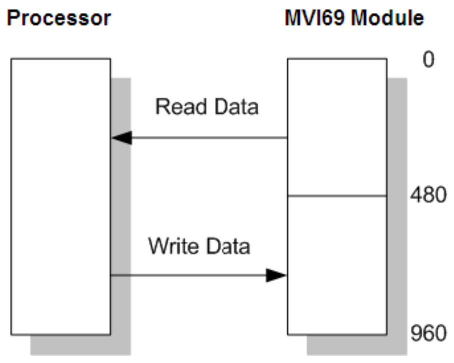
The Write Block ID associated with the block requests data from the processor.

The following example shows a typical backplane communication application.

If the backplane parameters are configured as follows:

```
Read Register Start:         0
Read Register Count:         480
Write Register Start:  480
Write Register Count:  480
```

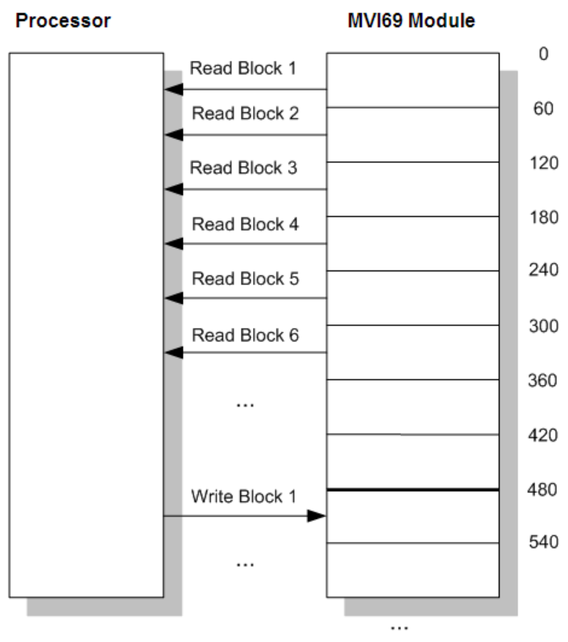
The backplane communication would be configured as follows:



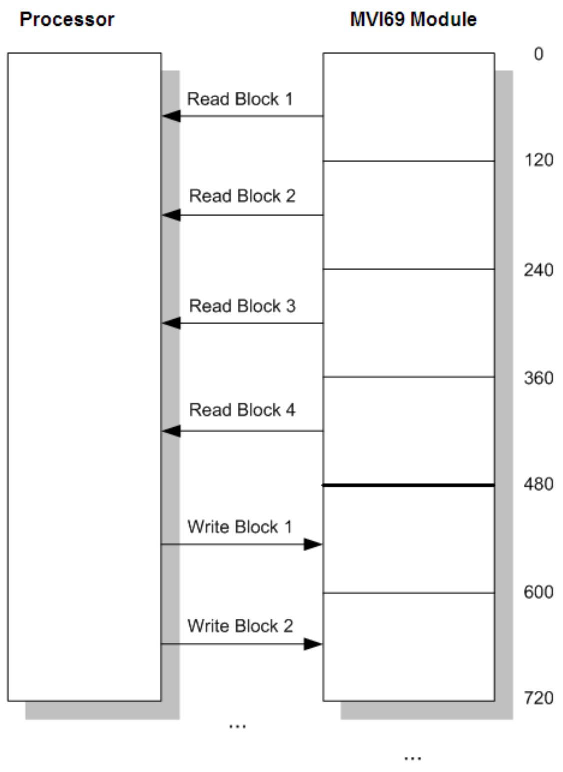
Database address 0 to 479 will be continuously transferred from the module to the processor. Database address 480 to 959 will continuously be transferred from the processor to the module.

The *Block Transfer Size* parameter basically configures how the Read Data and Write Data areas are broken down into data blocks (60, 120, or 240).

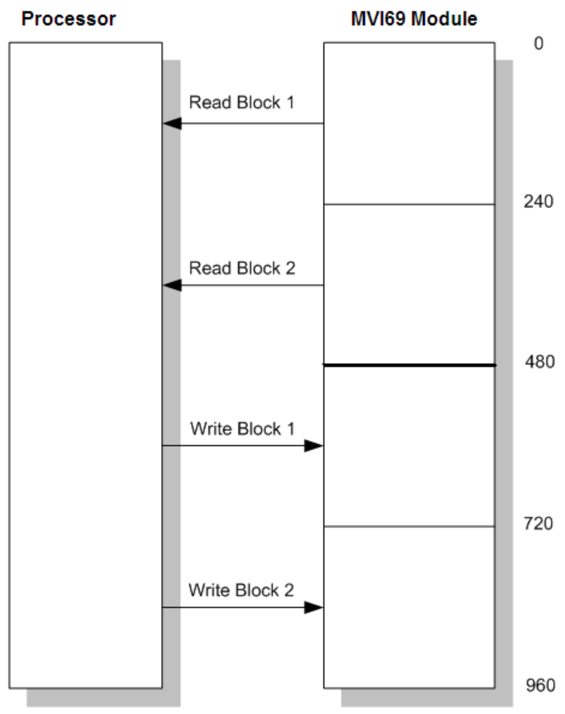
If Block Transfer Size = 60



If Block Transfer Size = 120



If Block Transfer Size = 240



Write Block

These blocks of data transfer information from the processor to the module. The structure of the output image used to transfer this data is shown below.

Write Block from Processor to Module

Word Offset	Description	Length (words)
0	Write Block ID	1
1 to *n*	Write Data	*n*

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

The Write Block ID is an index value used to determine the location in the module's database where the data will be placed. The number of data words per transfer depends on the configured *Block Transfer Size* parameter in the configuration file (possible values are 60, 120, or 240).

Status Block (Read Block ID = 0 or -1)

The Status block is automatically copied from the module into the *MNETC.STATUS* controller tag structure when the Read Block ID is 0 or -1, and contains module and Client status information. The third column in the table below contains the controller tags corresponding to each word in the block, if applicable. For a more complete description of the block contents, see the Status Data Definition (page 94).

Status Block from Module to Processor

Word Offset	Content	MnetC.STATUS Controller Tag
0	Read Block ID	
1	Write Block ID	
Block Statistics		
2	Program Scan Count	PassCnt
3	Read Block Count	Block.Read
4	Write Block Count	Block.Write
5	Parse Block Count	Block.Parse
6	Command Event Block Count	Block.Event
7	Command Block Count	Block.Cmd
8	Error Block Count	Block.Err
MNET Client		
9 to 38	Client 0 to Client 29 Command Execution Control Bits	CmdBits[x]
38 to (*n* + 1)	Spare	

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Additional Client status data is transferred in the Client Status blocks (page 113). The contents of these blocks are also described in the Status Data Definition.

Special Function Blocks

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

Initialize Output Data Blocks (1000 to 1083)

When the module performs a restart operation, it will request blocks of output data from the processor to initialize the module's output data. 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.

Block Request from Module to Processor

Word Offset	Description	Length
0	1000 to 1083 for n = 60, depending on configured memory write size.	1
1	1000 to 1083 for n = 60, depending on configured memory write size.	1
2 to (*n* + 1)	Spare	*n*

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

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 format of the returned write block is shown in the following table.

Block Response from Processor to Module

Word Offset	Description	Length
0	1000 to 1083, depending on configured memory write size.	1
1 to *n*	Output Data	*n*

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Event Command Blocks (2000-2029)

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

During routine operation, the module continuously cycles through the user-defined *MNET Client x Command List* (page 54) for each Client, examining commands in the order they are listed, and sending enabled commands on the network. However, the module also has a special command priority queue, which is an internal buffer that holds commands from special function blocks until they can be sent on the network.

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

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

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

Block Request from Processor to Module

Word Offset	Description
0	*Block ID* - This word contains the block **2000** to **2029** identification code to indicate that this block contains a command to execute by the Client driver. The last two digits indicate which Client is to execute the command.
1 to 4	*IP Address* -These words contain the IP address for the server the message is intended. Each digit (**0** to **255**) of the IP address is placed in one of the four registers. For example, to reach IP address 192.168.0.100, enter the following values in words 1 to 4 → 192, 168, 0 and 100. The module will construct the normal dotted IP address from the values entered. The values entered will be anded with the mask 0x00ff to insure the values are in the range of 0 to 255.
5	*Service Port* - 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	*Slave Address* - This word contains the Modbus node address for the message.
7	*Internal DB Address* - 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	*Point Count* - This word contains the count parameter that determines the number of digital points or registers to associate with the command.
9	*Swap Code* - The parameter specifies the swap type for the data. This function is only valid for function codes 3 and 4.

Word Offset	Description
10	*Modbus Function Code* - This word contains the Modbus function code for the command.
11	*Device Database Address* - This word contains the Modbus address in the slave device to be associated with the command.
12 to *n*	Spare

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

The module will use the parameters passed in this block to construct the command. The module then places the command in the command priority queue (if the queue is not already full; maximum capacity is 16 commands), and returns a response block to tell the ladder logic whether or not the command has been successfully added to the queue.

Block Response from Module to Processor

Word Offset	Description	Length
0	2000 to 2029	1
1	Write Block ID	1
2	0=Fail, 1=Success	1
3 to (*n* + 1)	Spare	*n* - 1

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Word 3 of the block can be used by the ladder logic to determine if the command was successfully added to the command priority queue. The command will fail if the queue for the Client is already full at the time when the Event Command block is received by the module.

Controller Tags

The elements of the *MNetC.CONTROL* controller tag structure contain all the values needed to build one Modbus TCP/IP command, have it sent to a specific Client on the module, and control the processing of the returned response block.

Controller Tag	**Data Type**	**Description**
EventCmd[0].IP0	INT	Enter the first digit of the destination server's IP address.
EventCmd[0].IP1	INT	Enter the second digit of the destination server's IP address.
EventCmd[0].IP2	INT	Enter the third digit of the destination server's IP address.
EventCmd[0].IP3	INT	Enter the fourth digit of the destination server's IP address.
EventCmd[0].ServPort	INT	Enter the TCP Service Port number (**0**-**65535**). Enter **502** for a MBAP message or **2000** for a MNET message.
EventCmd[0].Node	INT	Enter the Modbus slave node address. Enter **0** if not needed.
EventCmd[0].DBAddress	INT	Enter the module internal database address to associate with the command.
EventCmd[0].Count	INT	Enter the number of words or bits to be transferred by the Client.
EventCmd[0].Swap	INT	Enter the swap code for the data. This function is only valid for function codes 3 and 4.
EventCmd[0].Function	INT	Enter the Modbus function code for the command.
EventCmd[0].Address	INT	Enter the database address for the server.
EventCmd[0].EventCmd ClientID	INT	Enter the Client to execute the command (**0** to **29**).
EventCmdTrigger	BOOL	When all other values have been entered, set this bit to one (**1**) to trigger the execution of the Event Command.
EventCmdPending	BOOL	Temporary variable used to prevent a new Event Command block from being sent to the module until the previously sent Event Command block has been completely processed and a response block has been returned.

Client Status Blocks (3000 to 3029)

Client status data for a specific Client can be requested and returned in a special Client Status block. The status data contained in the Client Status block is different from the status data contained in the normal data transfer blocks.

Block Request from Processor to Module

Word Offset	Description	Length
0	3000 to 3029 (last digits indicate which Client to consider)	1
1 to *n*	Spare	*n*

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Block Response from Module to Processor

Word Offset	Description	Length
0	3000 to 3029	1
1	Write Block ID	1
2 to 11	Client status data	10
12 to 27	Command error list data for Client	16
28 to (*n* + 1)	Reserved	*n* - 26

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Client Status Data

Word Offset	Client Status	MNetC.STATUS Controller Tag
2	Total number of command list requests	ClientStats[x].CmdReq
3	Total number of command list responses	ClientStats[x].CmdResp
4	Total number of command list errors	ClientStats[x].CmdErr
5	Not used	ClientStats[x].Requests
6	Not used	ClientStats[x].Responses
7	Not used	ClientStats[x].ErrSent
8	Not used	ClientStats[x].ErrRec
9	Configuration Error Word	ClientStats[x].CfgErrWord
10	Current error	ClientStats[x].CurErr
11	Last error	ClientStats[x].LastErr

Status information transferred in the Client Status block can be viewed in the *MNetC.STATUS* controller tag structure in the ladder logic. For more information, see the Status Data Definition (page 94).

Controller Tags

To issue a Client Status block request, enter the appropriate values in the following members of the *MNetC.STATUS* controller tag structure in the ladder logic.

Controller Tag	Data Type	Description
StatReqClientID	INT	Enter the Client (**0**-**29**) to request status data for.
ClientStatsTrigger	BOOL	Set the value of this tag to **1** to trigger the Client Status block request.

Command Control Blocks (5001 to 5016)

Note: Command Control is not needed for normal Modbus command list polling operations and is needed only occasionally for special circumstances.

During routine operation, the module continuously cycles through the user-defined *MNET Client x Command List* (page 54) for each Client, examining commands in the order they are listed, and sending enabled commands on the network. However, the module also has a special command priority queue, which is an internal buffer that holds commands from special function blocks until they can be sent on the network.

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

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

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

Commands in the *MNET Client x Command List* may be either enabled for routine polling or disabled and excluded from routine polling. A disabled command has its bit in the *MNetC.CONTROL.WriteCmdBits[x]* controller tag array set to zero (**0**) and is skipped during routine polling. An enabled command has its bit in the *WriteCmdBits[x]* controller tag set to one (**1**) and is sent during routine polling. However, Command Control allows any command in the predefined *MNET Client x Command List* to be added to the command priority queue, whether it is enabled for routine polling or not.

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

A single Command Control block request can place up to 16 commands from the *MNET Client x Command List* into the command priority queue.

Block Request from Processor to Module

Word Offset	Description	Length
0	Command Control block identification code of **5001** to **5016**. The rightmost digit indicates the number of commands (**1** to **16**) to add to the command priority queue.	1
1	Client index (**0** to **29**)	1
2	This word contains the Command Index for the first command to be entered into the queue.	1
3	Command Index 2	1
4	Command Index 3	1
5	Command Index 4	1
6	Command Index 5	1
7	Command Index 6	1
8	Command Index 7	1
9	Command Index 8	1
10	Command Index 9	1
11	Command Index 10	1
12	Command Index 11	1
13	Command Index 12	1
14	Command Index 13	1
15	Command Index 14	1
16	Command Index 15	1
17	Command Index 16	1
18 to *n*	Spare	*n* - 17

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

The last digit in the block identification code indicates the number of commands to process. For example, a block identification code of **5003** indicates that three commands are to be placed in the queue. In this case, the first three of the 16 available Command Indexes will be used to determine exactly which three commands will be added to the queue, and to set their order of execution.

Values to enter for the 16 Command Indexes range from **0** to **15** and correspond to the *MNET Client x Command List* entries, which are numbered from 1 to 16. To determine the Command Index value, subtract one (**1**) from the row number of the command in the *MNET Client x Command List*, as seen in the *Command Editor* window of *ProSoft Configuration Builder (PCB)*.

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

Block Response from Module to Processor

Word Offset	Description
0	This word contains the block 5001 to 5016 requested by the processor.
1	This word contains the next write block identification code.
2	This word contains the number of commands in the block placed in the command queue.
3 to (*n* + 1)	Spare

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Controller Tags

The *MNetC.CONTROL* controller tag structure holds all the values needed to create one Command Control block, have it sent to the module, and control the processing of the returned response block.

Controller Tag	Data Type	Description
CmdControl.CmdCntReqClientID	INT	Client (**0**-**29**) to execute command
CmdControl.CmdCntReqCMDqty	INT	Enter a decimal value representing the quantity of commands to be requested in the Command Control block (**1** to **16**).
CmdControl.CmdCntReqCMDindex	INT[16]	Enter the **ROW NUMBER** of the command in the *MNET Client x Command List* in *Prosoft Configuration Builder* minus **1**. This is a 16-element array. Each element holds one Command Index.
CmdControl.WriteCmdBits	INT(30)	Enter a **1** (enable) or a **0** (disable) to select which commands on the configuration's *Client x Command List* will be executed during routine polling. There is one 16-bit word for each of the 30 Clients. Each of the 16 bits corresponds to one of the 16 commands available to each Client. The state of these *WriteCmdBits* overrides whatever value may be assigned to the *Enable* parameter in the configuration. **Note:** This parameter only affects routine polling. It has no effect on Command Control blocks.
CmdControl.CmdBitsTrigger	BOOL	Set this tag to **1** to activate newly entered Write Command Bits. **Note:** Do not use this parameter to execute a Command Control block. Use the *CmdControlTrigger* to execute Command Control blocks.
CmdControlPending	BOOL	Temporary variable used to prevent a new Command Control block from being sent to the module until the previously sent Command Control block has been completely processed and a response block has been returned.
CmdControlTrigger	BOOL	Set this tag to **1** to trigger the execution of a Command Control block after all the other parameters have been entered.
CmdBitPending	BOOL	Not used.
CmdBitTrigger	BOOL	Not used.

Setting Write Command Bits

Each command in a Client's Command List is controlled through the ladder logic. The bits set in the *MNetC.CONTROL.CmdControl.WriteCmdBits[x]* controller tag override the *Enable* parameter in the configuration. If a bit for a specific command is set to zero (**0**) in the *WriteCmdBits[x]* controller tag, the command will not be executed, regardless of whether it is enabled or disabled in the configuration.

Each Client is assigned one 16-bit word in the *WriteCmdBits[x]* controller tag array, with one bit for each of the 16 commands in its Command List. When a bit corresponding to a command is set to **1**, the command is executed during routine polling according to the parameters set for it in the *MNET Client x Command List*. If the bit is set to **0**, the command is skipped during routine polling.

Note: If you change the bits in the *MNetC.CONTROL.CmdControl.WriteCmdBits[x]* controller tag, you must set the *MNetC.CONTROL.CmdControl.CmdBitsTrigger* tag to 1 to activate them.

Set New Module IP Address Block (9990)

Block Request from Processor to Module

Word 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 *n*	Reserved	*n* - 4

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Block Response from Module to Processor

Word Offset	Description	Length
0	0	1
1	Write Block ID	1
2	First digit of dotted IP address	1
3	Second digit of dotted IP address	1
4	Third digit of dotted IP address	1
5	Last digit of dotted IP address	1
6 to (*n* + 1)	Spare data area	*n* - 4

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Get Module IP Address Block (9991)

Block Request from Processor to Module

Word Offset	Description	Length
0	9991	1
1 to *n*	Spare data area	*n*

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Block Response from Module to Processor

Word Offset	Description	Length
0	0	1
1	Write Block ID	1
2	First digit of dotted IP address	1
3	Second digit of dotted IP address	1
4	Third digit of dotted IP address	1
5	Last digit of dotted IP address	1
6 to (*n* + 1)	Spare data area	*n* - 4

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Warm Boot Block (9998)

This block is sent from the CompactLogix or MicroLogix to the module when the module is required to perform a Warm Boot (software reset) operation. The block is commonly sent to the module any time configuration data modifications are made in the controller tags data area. This causes the module to read the new configuration information and to restart. The following table describes the format of the Warm Boot block.

Block Request from Processor to Module

Word Offset	**Description**	**Length**
0	9998	1
1 to *n*	Not used	*n*

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

Cold Boot Block (9999)

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

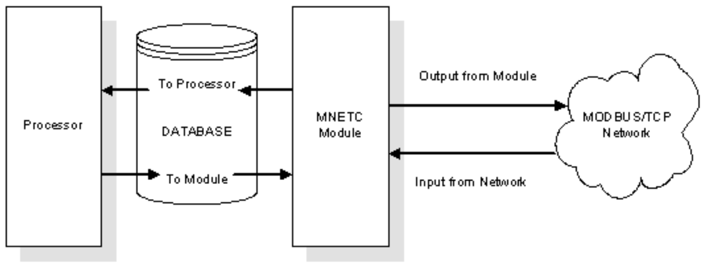
Block Request from Processor to Module

Word Offset	**Description**	**Length**
0	9999	1
1 to *n*	Spare	*n*

n = 60, 120, or 240 depending on the Block Transfer Size parameter (refer to the configuration file).

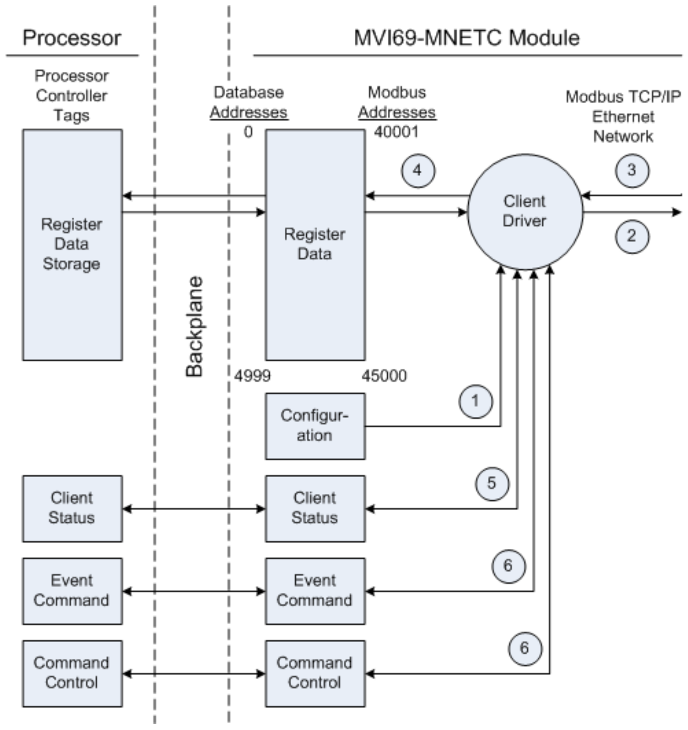
5.2.2 Data Flow between MVI69-MNETC Module and Processor

The following topics describe the flow of data between the two pieces of hardware (processor and MVI69-MNETC 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.



Client Driver

In the Client driver, the MVI69-MNETC module issues read or write commands to servers on the Modbus TCP/IP network using up to 30 simulated Clients. The commands originate either from the module's user-configured *Client x Command List* for each Client, or directly from the processor as Event Commands. The commands from the *Client x Command List* are executed either via routine polling or as a result of special Command Control block requests from the processor. Client status data is returned to the processor in special Client Status blocks. The following flowchart describes the flow of data into and out of the module.



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

Client Command List

In order for the Client to function, the module's Client Command List must be defined 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

For information on troubleshooting commands, see Client Command Errors (page 97).

5.3 Cable Connections

The MVI69-MNETC module has the following functional communication connections installed:

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

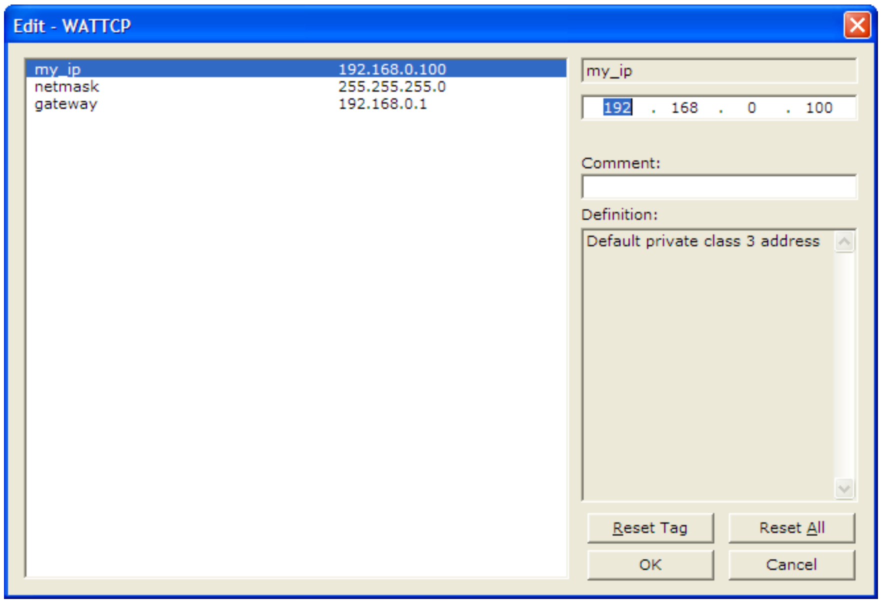
5.3.1 Ethernet Connection

The MVI69-MNETC module has an RJ45 port located on the front of the module, labeled *Ethernet*, for use with the TCP/IP network. The module is connected to the Ethernet network using an Ethernet cable between the module's Ethernet port and an Ethernet switch or hub.

Note: Depending on hardware configuration, you may see more than one RJ45 port on the module. The Ethernet port is labeled *Ethernet*.
Warning: The MVI69-MNETC module is NOT compatible with Power Over Ethernet (IEEE802.3af / IEEE802.3at) networks. Do NOT connect the module to Ethernet devices, hubs, switches or networks that supply AC or DC power over the Ethernet cable. Failure to observe this precaution may result in damage to hardware, or injury to personnel.
Important: The module requires a static (fixed) IP address that is not shared with any other device on the Ethernet network. Obtain a list of suitable IP addresses from your network administrator BEFORE configuring the Ethernet port on this module.

Ethernet Port Configuration - wattcp.cfg

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



my_ip	192.168.0.100
netmask	255.255.255.0
gateway	192.168.0.1

my_ip

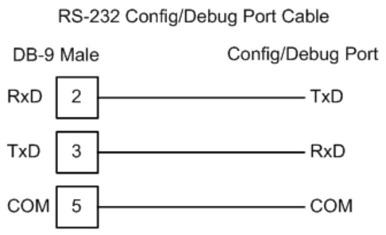
192 . 168 . 0 . 100

Default private class 3 address

You may also view the network configuration using a PC serial port connection and an ASCII terminal program (like Windows HyperTerminal) by selecting **[@]** (Network Menu) and **[V]** (View) options when connected to the Debug port. For more information on serial port access, see the chapter on Diagnostics and Troubleshooting (page 81).

5.3.2 RS-232 Configuration/Debug Port

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

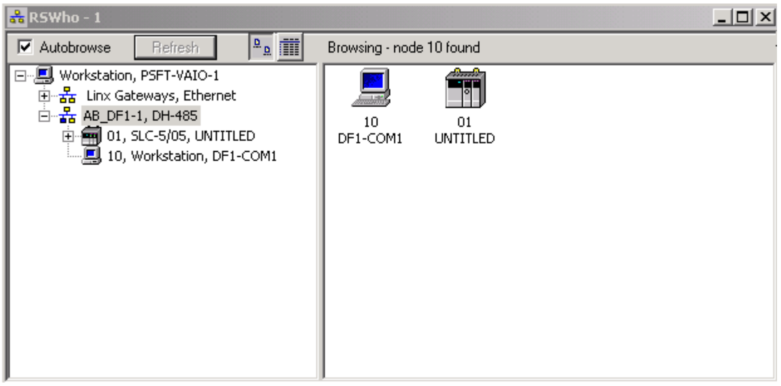


Disabling the RSLinx Driver for the Com Port on the PC

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

1 Open *RSLinx* and go to **COMMUNICATIONS>RSWHO**.

2 Make sure that you are not actively browsing using the driver that you wish to stop. The following shows an actively browsed network:



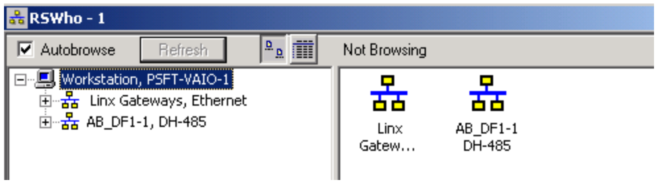
Browsing - node 10 found

Workstation, PSFT-VAIO-1
Linx Gateways, Ethernet
AB_DF1-1, DH-485
01, SLC-5/05, UNTITLED
10, Workstation, DF1-COM1

10
DF1-COM1

01
UNTITLED

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



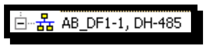
Not Browsing

Workstation, PSFT-VAIO-1
Linx Gateways, Ethernet
AB_DF1-1, DH-485

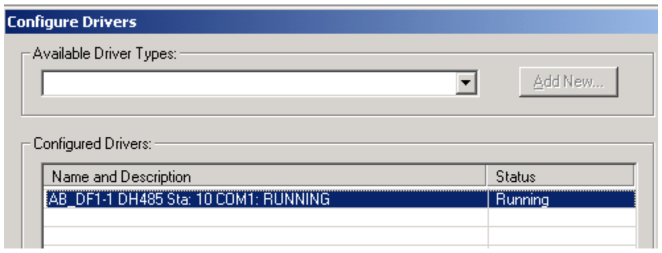
Linx
Gatew...

AB_DF1-1
DH-485

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

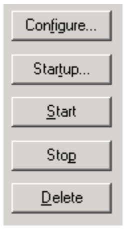


4 When you have verified that the driver is not being browsed, go to **COMMUNICATIONS>CONFIGURE DRIVERS**.
You may see something like this:

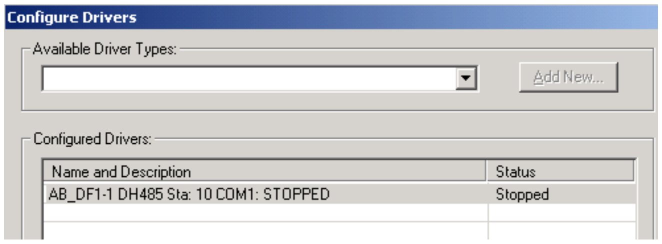


Name and Description	Status
AB_DF1-1 DH485 Sta: 10 COM1: RUNNING	Running

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



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

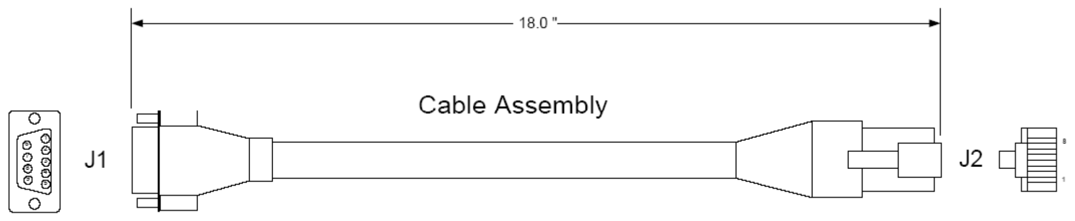


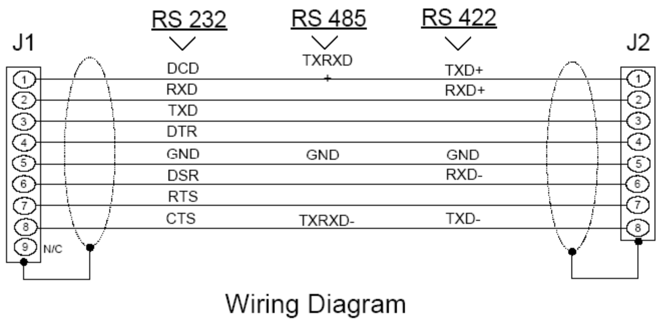
Name and Description	Status
AB_DF1-1 DH485 Sta: 10 COM1: STOPPED	Stopped

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

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

5.3.3 DB9 to RJ45 Adaptor (Cable 14)





5.4 Modbus Protocol Specification

The following pages give additional reference information regarding the Modbus protocol commands supported by the MVI69-MNETC.

5.4.1 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 MVI69-MNETC 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 MVI69-MNETC 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.

5.4.2 Read Coil Status (Function Code 01)

Query

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

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

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

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

Response

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

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

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

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

5.4.3 Read Input Status (Function Code 02)

Query

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

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

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

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

Response

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

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

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

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

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

5.4.4 Read Holding Registers (Function Code 03)

Query

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

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

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

Response

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

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

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

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

5.4.5 Read Input Registers (Function Code 04)

Query

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

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

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

Response

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

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

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

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

5.4.6 Force Single Coil (Function Code 05)

Query

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

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

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

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

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

Response

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

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

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

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

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

5.4.7 Preset Single Register (Function Code 06)

Query

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

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

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

Response

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

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

5.4.8 Diagnostics (Function Code 08)

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

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

In general, issuing a diagnostic function to a remote device does not affect the running of the user program in the remote device. 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 MVI69-MNETC 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.

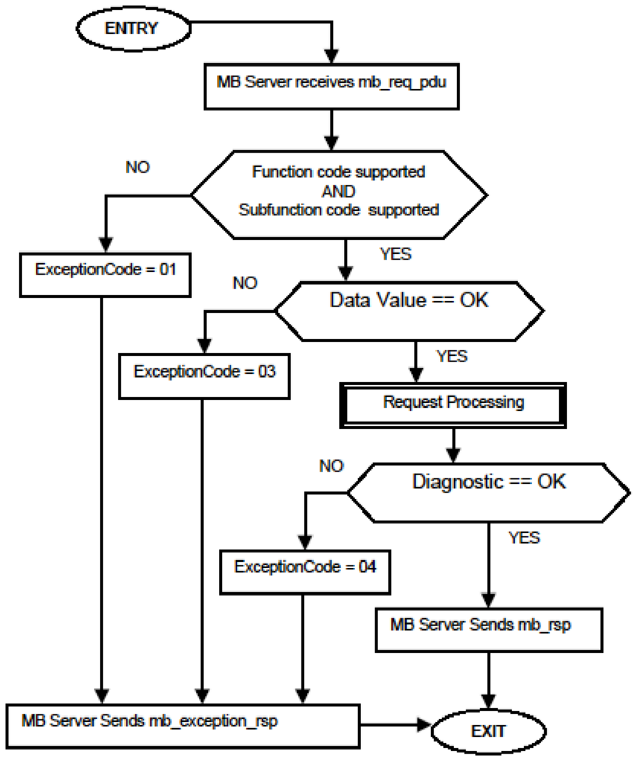
Sub-function	**Data Field (Request)**	**Data Field (Response)**
00 00	Any	Echo Request Data

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.



5.4.9 Force Multiple Coils (Function Code 15)

Query

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

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

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

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

Response

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

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

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

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

5.4.10 Preset Multiple Registers (Function Code 16)

Query

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

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

5.4.11 Modbus Exception Responses

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

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

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

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

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

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

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

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

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

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

Modbus Exception Codes

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

6 Support, Service & Warranty

In This Chapter

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. Detailed contact information for all our worldwide locations is available on the following page.*

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 145). 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 141). 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 warranteed in the terms specified above for new Product, provided said warranty will be for the period remaining on the original new equipment warranty or, if the original warranty is no longer in effect, for a period of 90 days from the date of repair.

6.2.2 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 of communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly or indirectly to death, personal injury or severe physical or environmental damage (collectively, "high risk activities"). ProSoft specifically disclaims any express or implied warranty of fitness for high risk activities.

6.2.4 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 145) 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