

薛晨

电话: 010-81817979

QQ: 58409206

手机: 13801020197

MSN: xc-com@163.com

inRAx[®]



MVI69-MCM

CompactLogix Platform

Modbus Communication Module

User Manual

May 22, 2007


ProSoft[®]
TECHNOLOGY

薛晨

电话: 010-81817979 QQ: 58409206

手机: 13801020197

MSN: xc.com@163.com

Please Read This Notice

Successful application of this module requires a reasonable working knowledge of the Rockwell Automation CompactLogix or MicroLogix hardware, the MVI69-MCM Module and the application in which the combination is to be used. For this reason, it is important that those responsible for implementation satisfy themselves that the combination will meet the needs of the application without exposing personnel or equipment to unsafe or inappropriate working conditions.

This manual is provided to assist the user. Every attempt has been made to assure that the information provided is accurate and a true reflection of the product's installation requirements. In order to assure a complete understanding of the operation of the product, the user should read all applicable Rockwell Automation documentation on the operation of the Rockwell Automation hardware.

Under no conditions will ProSoft Technology be responsible or liable for indirect or consequential damages resulting from the use or application of the product.

Reproduction of the contents of this manual, in whole or in part, without written permission from ProSoft Technology is prohibited.

Information in this manual is subject to change without notice and does not represent a commitment on the part of ProSoft Technology Improvements and/or changes in this manual or the product may be made at any time. These changes will be made periodically to correct technical inaccuracies or typographical errors.

Your Feedback Please

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

ProSoft Technology

1675 Chester Avenue, Fourth Floor

Bakersfield, CA 93301

+1 (661) 716-5100

+1 (661) 716-5101 (Fax)

<http://www.prosoft-technology.com>

Copyright © ProSoft Technology, Inc. 2000 - 2007. All Rights Reserved.

MVI69-MCM User Manual

May 22, 2007

PSFT.MCM.MVI69.UM.07.05.22

ProSoft Technology®, ProLinX®, inRAX®, ProTalk® and RadioLinX® are Registered Trademarks of ProSoft Technology, Inc.

Contents

PLEASE READ THIS NOTICE.....	2
Your Feedback Please	2
GUIDE TO THE MVI69-MCM USER MANUAL.....	5
1 START HERE	7
1.1 System Requirements.....	7
1.2 Package Contents	8
1.3 Setting Jumpers	9
1.4 Installing the Module.....	10
1.5 Connect your PC to the Processor	13
1.6 Download the Sample Program to the Processor.....	14
1.6.1 Configuring RSLinx	15
1.7 Connect your PC to the Module.....	17
2 INSTALLING AND CONFIGURING THE MODULE.....	19
2.1 Installing and Configuring the Module with a CompactLogix Processor.....	21
2.1.1 After you Complete the Module Setup.....	26
2.2 Installing and Configuring the Module with a MicroLogix Processor.....	29
2.3 Module Data Object (MCMModuleDef)	32
2.3.1 Status Object (MCM1Status)	34
2.3.2 User Data Objects.....	34
2.3.3 Slave Polling Control and Status	35
2.3.4 MODBUS Message Data	35
2.4 Command List Overview.....	36
2.5 MODBUS Command Configuration	36
2.5.1 Floating Point Support	36
2.5.2 Commands Supported by the Module	42
2.5.3 Command Entry Formats.....	43
2.6 Uploading and Downloading the Configuration File.....	45
2.6.1 Required Hardware.....	46
2.6.2 Required Software	46
2.6.3 Transferring the Configuration File to Your PC.....	47
2.6.4 Transferring the Configuration File to the Module	50
3 LADDER LOGIC.....	53
4 DIAGNOSTICS AND TROUBLESHOOTING	55
4.1 Reading Status Data from the module	55
4.1.1 The Configuration/Debug Menu.....	55
4.1.2 Required Hardware	56
4.1.3 Required Software	57
4.1.4 Using the Configuration/Debug Port	57
4.1.5 Main Menu	58
4.1.6 Database View Menu.....	61
4.1.7 Backplane Menu	63
4.1.8 Protocol Serial MCM Menu	64

4.1.9	Master Command Error List Menu	65
4.1.10	Serial Port Menu	66
4.1.11	Data Analyzer	67
4.1.12	Data Analyzer Tips	70
4.2	LED Status Indicators.....	72
4.3	Clearing a Fault Condition	73
4.4	Troubleshooting.....	73
5	REFERENCE	75
5.1	Product Specifications	75
5.1.1	Features and Benefits	75
5.1.2	General Specifications	75
5.1.3	Hardware Specifications.....	76
5.1.4	Functional Specifications.....	76
5.2	Functional Overview	78
5.2.1	General Concepts.....	78
5.2.2	Normal Data Transfer	81
5.2.3	Special Blocks	86
5.2.4	Command Control Blocks.....	89
5.2.5	Data Flow between MVI69-MCM Module and CompactLogix or MicroLogix Processor.....	93
5.3	Cable Connections.....	97
5.3.1	RS-232 Configuration/Debug Port.....	97
5.3.2	RS-232	99
5.3.3	RS-422	101
5.3.4	RS-485	102
5.3.5	DB9 to RJ45 Adaptor (Cable 14).....	102
5.4	MCM Database Definition.....	103
5.5	Status Data Definition.....	103
5.6	Configuration Data Definition	105
5.6.1	Port 1 Setup.....	106
5.6.2	Port 2 Setup.....	108
5.6.3	Port 1 Commands.....	111
5.6.4	Port 2 Commands.....	111
5.6.5	Misc. Status	111
5.6.6	Command Control	113
SUPPORT, SERVICE & WARRANTY.....		115
Module Service and Repair		115
General Warranty Policy – Terms and Conditions		116
Limitation of Liability.....		117
RMA Procedures		117
INDEX.....		119

薛晨

Guide to the MVI69-MCM User Manual

Function		Section to Read	Details
Introduction (Must Do)	→	Start Here (page 7)	This Section introduces the customer to the module. Included are: package contents, system requirements, hardware installation, and basic configuration.
Verify Communication, Diagnostic and Troubleshooting	→	Verifying Communication (page 55) Diagnostics and Troubleshooting (page 55)	This section describes how to verify communications with the network. Diagnostic and Troubleshooting procedures.
Reference Product Specifications Functional Overview Glossary	→	Reference (page 74) Functional Overview (page 77) Product Specifications (page 75)	These sections contain general references associated with this product, Specifications, and the Functional Overview.
Support, Service, and Warranty Index	→	Support, Service and Warranty (page 114)	This section contains Support, Service and Warranty information. Index of chapters.

薛晨

薛晨

1 Start Here

In This Chapter

- System Requirements 7
- Package Contents 8
- Setting Jumpers 9
- Installing the Module 9
- Connect your PC to the Processor 13
- Download the Sample Program to the Processor 14
- Connect your PC to the Module 16

Installing the MVI69-MCM module requires a reasonable working knowledge of the Rockwell Automation hardware, the MVI69-MCM Module and the application in which they will be used.



Caution: It is important that those responsible for implementation can complete the application without exposing personnel, or equipment, to unsafe or inappropriate working conditions. Safety, quality and experience are key factors in a successful installation.

1.1 System Requirements

The MVI69-MCM 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-MCM module. The module requires 800mA of available power.
- Rockwell Automation RSLogix 5000 (CompactLogix) or RSLogix 500 programming software version 2.51 or higher.
- Rockwell Automation RSLinx communication software
- Pentium® II 450 MHz minimum. Pentium III 733 MHz (or better) recommended
- Supported operating systems:
 - Microsoft Windows XP Professional with Service Pack 1 or 2
 - Microsoft Windows 2000 Professional with Service Pack 1, 2, or 3
 - Microsoft Windows Server 2003
- 128 Mbytes of RAM minimum, 256 Mbytes of RAM

- 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 MCM protocol.

1.2 Package Contents

The following components are included with your MVI69-MCM module, and are all required for installation and configuration.

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

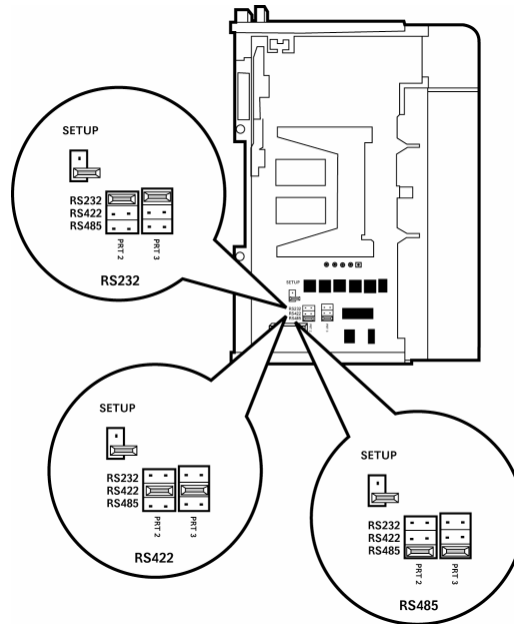
Qty.	Part Name	Part Number	Part Description
1	MVI69-MCM Module	MVI69-MCM	Modbus Communication Module
1	Cable	Cable #15, RS232 Null Modem	For RS232 Connection to the CFG Port
3	Cable	Cable #14, RJ45 to DB9 Male Adapter cable	For DB9 Connection to Module's Port
2	Adapter	1454-9F	Two Adapters, DB9 Female to Screw Terminal. For RS422 or RS485 Connections to Port 1 and 2 of the Module
1	ProSoft Solutions CD		Contains sample programs, utilities and documentation for the MVI69-MCM module.

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

薛晨

1.3 Setting Jumpers

When the module is manufactured, the port selection jumpers are set to RS-232. To use RS-422 or RS-485, you must set the jumpers to the correct position. The following diagram describes the jumper settings.



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.

薛晨

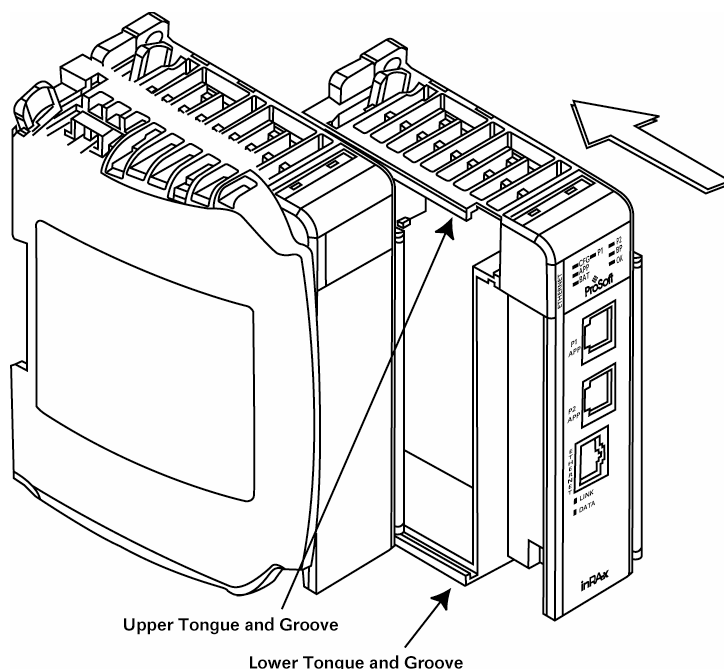
1.4 Installing the Module

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.

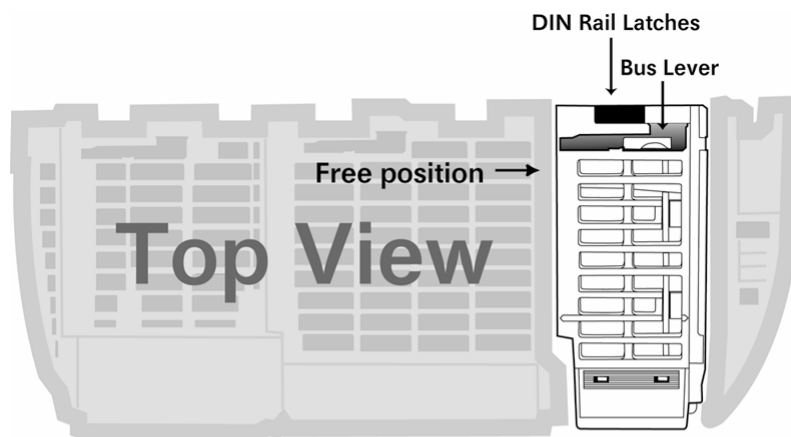
- 1 Align the module using the upper and lower tongue-and-groove slots with the adjacent module and slide forward in the direction of the arrow.



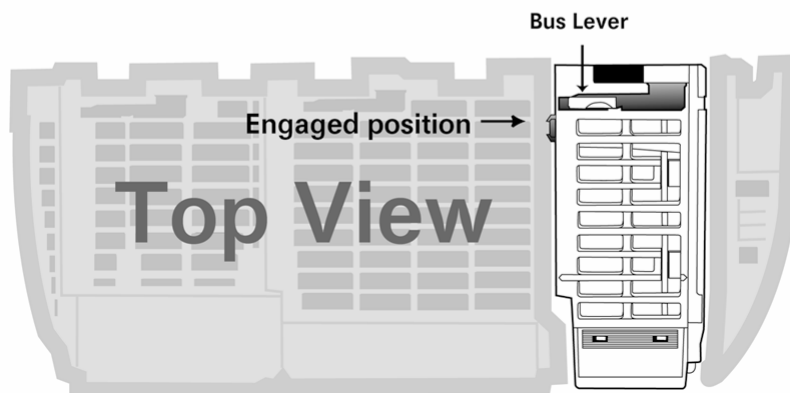
- 2 Move the module back along the tongue-and-groove slots until the bus connectors on the MVI69 module and the adjacent module line up with each other.

薛晨

- 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.



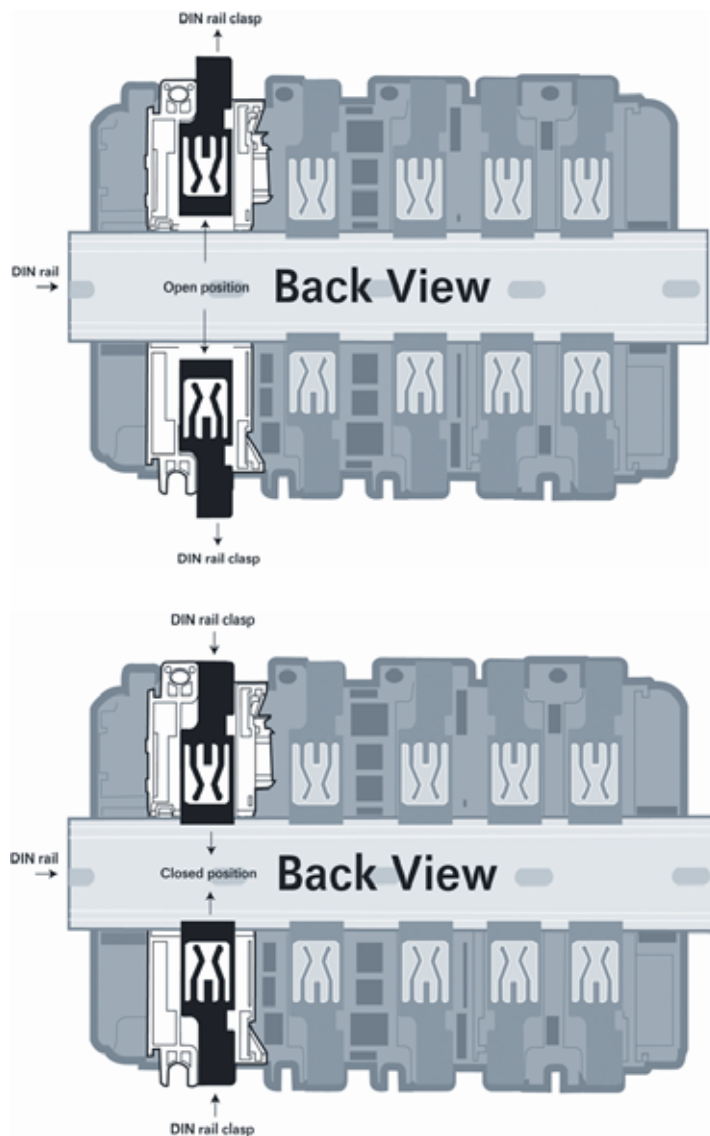
Move the Bus Lever to the left
until it clicks



- 4 Close all DIN rail latches.

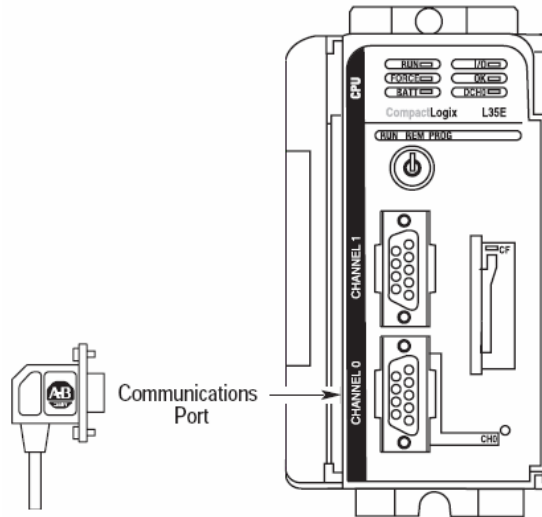
薛晨

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

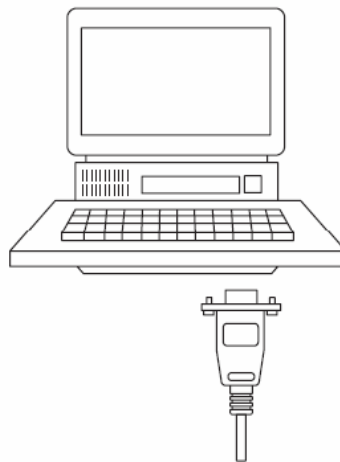


1.5 Connect your PC to the Processor

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



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



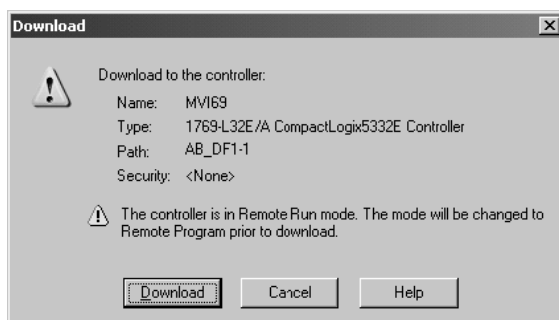
薛晨

1.6 Download the Sample Program to the Processor

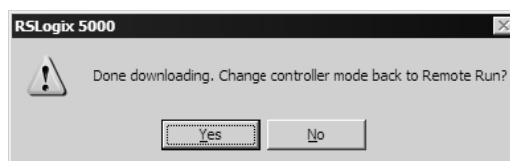
- ***To download the sample program from RSLogix 5000 to the CompactLogix processor:***

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

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



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

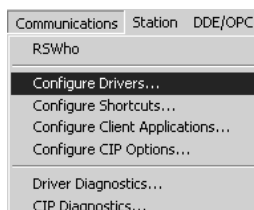


Note: If you receive an error message during these steps, refer to your RSLogix documentation to interpret and correct the error.

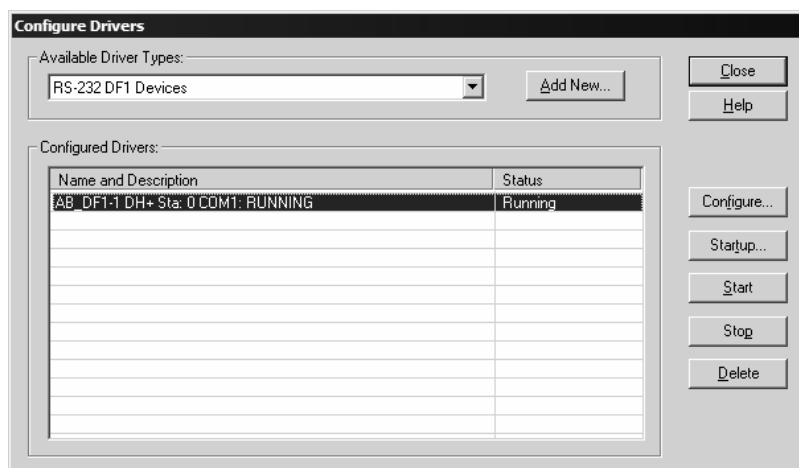
1.6.1 Configuring RSLinx

➤ **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.



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".

- Click to select the driver, and then click Configure. This action opens the Configure Allen-Bradley DF1 Communications Device dialog box.



The dialog box is titled "Configure Allen-Bradley DF1 Communications Device". It contains the following fields and controls:

- Device Name: AB_DF1-1
- Comm Port: COM1
- Device: Logix 5550 - Serial Port
- Baud Rate: 19200
- Station Number (Octal): 00
- Parity: None
- Error Checking: CRC
- Stop Bits: 1
- Protocol: Full Duplex
- Auto-Configure button
- ☐ Use Modem Dialer
- Configure Dialer button
- Ok, Cancel, Delete, and Help buttons at the bottom.

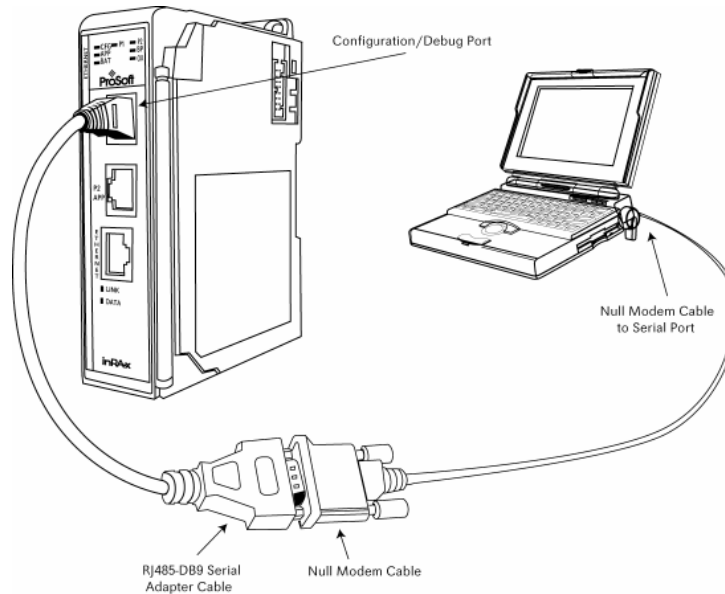
- Click the Auto-Configure button. RSLinx will attempt to configure your serial port to work with the selected driver.
- When you see the message "Auto Configuration Successful", click the OK button to dismiss the dialog box.

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.

1.7 Connect your PC to the Module

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

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



薛晨

2 Installing and Configuring the Module

In This Chapter

- Installing and Configuring the Module with a CompactLogix Processor 21
- Installing and Configuring the Module with a MicroLogix Processor 29
- Module Data Object (MCMModuleDef) 32
- Command List Overview 35
- MODBUS Command Configuration 36
- Uploading and Downloading the Configuration File 45

This chapter describes how to install and configure the module to work with your application. The configuration process consists of the following steps.

- 1 Use RSLogix to identify the module to the processor and add the module to a project.

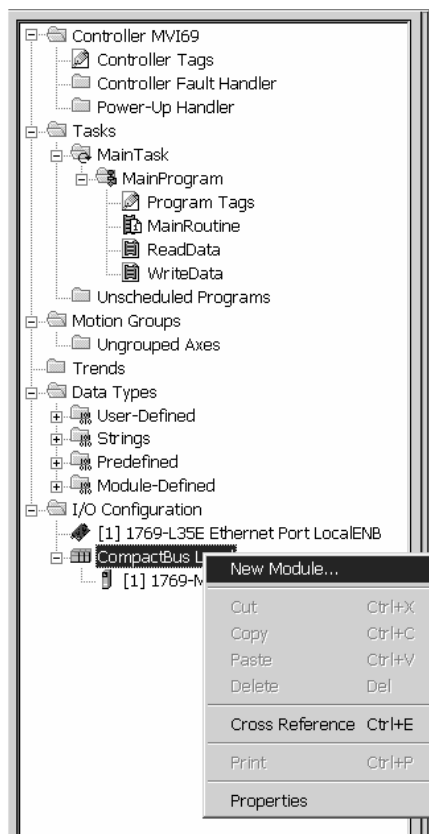
NOTE: The RSLogix software must be in "offline" mode to add the module to a project.

- 2 Modify the module's configuration files to meet the needs of your application, and copy the updated configuration to the module. Example configuration files are provided on the CD-ROM. Refer to the Modifying the Example Configuration File section, later in this chapter, for more information on the configuration files.
- 3 Modify the example ladder logic to meet the needs of your application, and copy the ladder logic to the processor. Example ladder logic files are provided on the CD-ROM.

Note: If you are installing this module in an existing application, you can copy the necessary elements from the example ladder logic into your application.

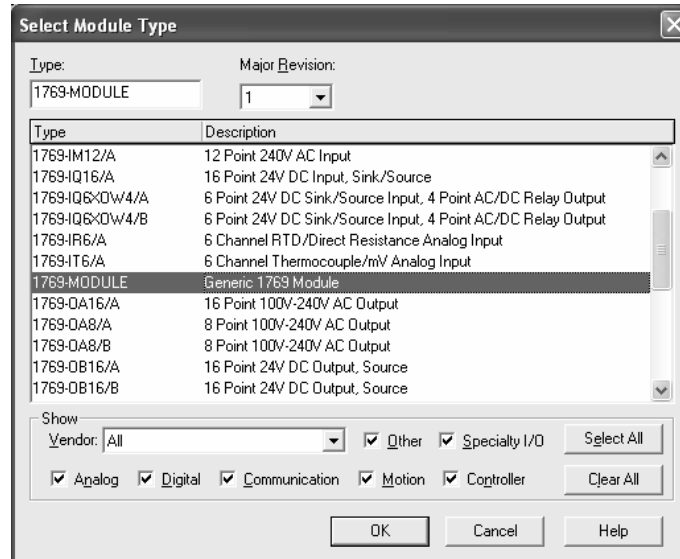
The rest of this chapter describes these steps in more detail.

The first step in installing and configuring the module is to define the module to the system. Select "I/O Configuration" in the Controller Organization List, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose "New Module".



薛晨

This action opens the Select Module Type dialog box.



Select "1769-Module (Generic 1769 Module)" from the list and click OK. This action opens the General page, allowing you to configure the block transfer size.

2.1 Installing and Configuring the Module with a CompactLogix Processor

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.

This chapter describes how to install and configure the module to work with your application. The configuration process consists of the following steps.

- 1 Use RSLogix to identify the module to the processor and add the module to a project.

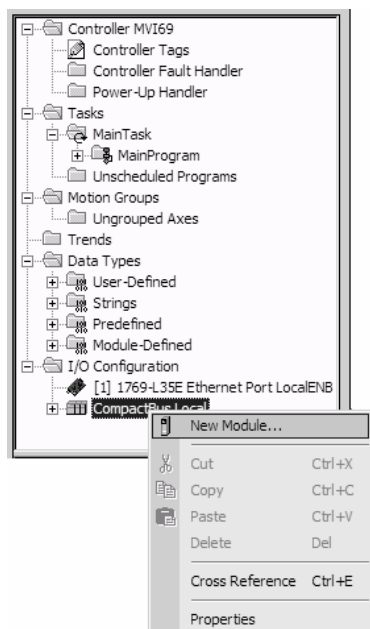
NOTE: The RSLogix software must be in "offline" mode to add the module to a project.

- 2 Modify the module's configuration files to meet the needs of your application, and copy the updated configuration to the module. Example configuration files are provided on the CD-ROM. Refer to the Modifying the Example Configuration File section, later in this chapter, for more information on the configuration files.
- 3 Modify the example ladder logic to meet the needs of your application, and copy the ladder logic to the processor. Example ladder logic files are provided on the CD-ROM.

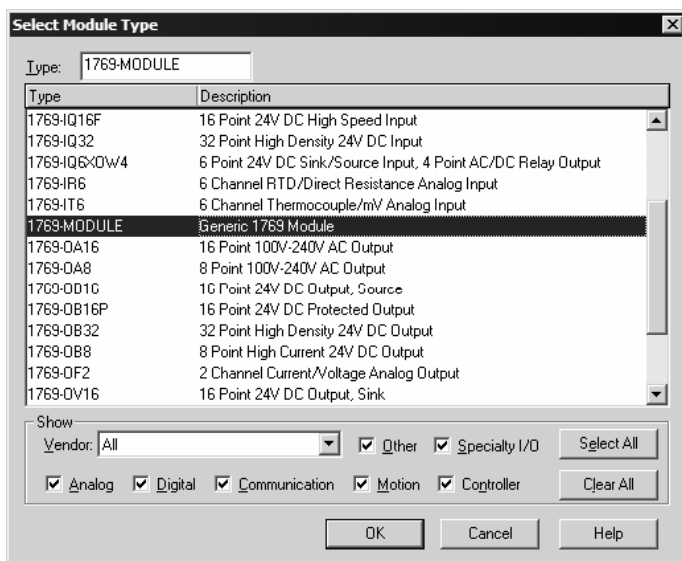
Note: If you are installing this module in an existing application, you can copy the necessary elements from the example ladder logic into your application.

The rest of this chapter describes these steps in more detail.

The first step in setting up the processor ladder file is to define the I/O type module to the system. 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 Select Module Type dialog box.



Select the 1769-Module (Generic 1769 Module) from the list and click OK.

You should configure the Connection Parameters according to the Block Transfer Size parameter in the configuration file as follows:

On the General page, fill in the values shown in the tables below, according to the Block Transfer Size parameter in the configuration file. You must select the **Comm Format** as **Data - INT**.

The configured Input Size and Output Size will depend on the block transfer size parameter defined in the configuration file. 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

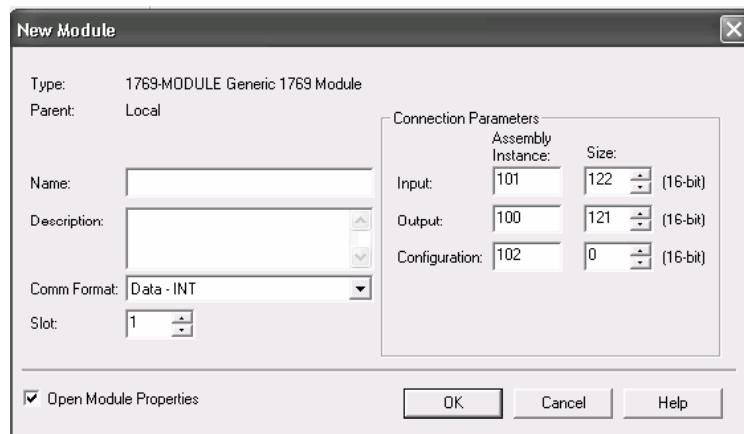
Field	Recommended Value
Type	1769-MODULE Generic 1769 Module
Parent	Local
Name	MVI69
Description	MVI69 Application Module
Comm Format	Data - INT
Slot	The slot number in the rack where the module is installed
Input Assembly Instance	101
Input Size	242
Output Assembly Instance	100
Output Size	241
Configuration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

Important: If you set the **Assembly Instance** and **Size** values incorrectly, the module will not communicate over the backplane of the CompactLogix or MicroLogix rack.

Click **Next** to continue.

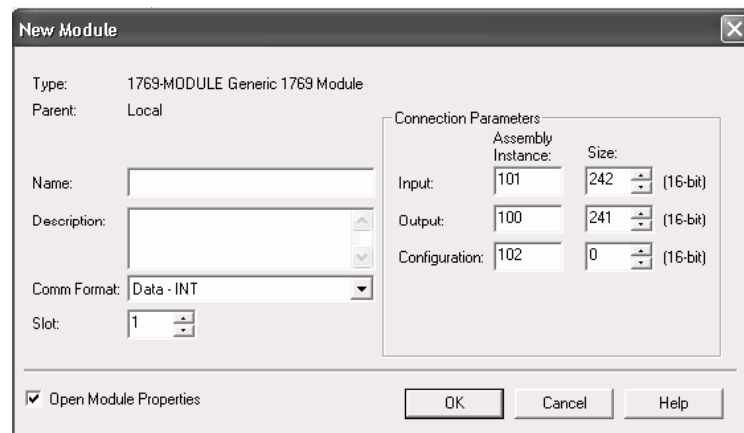
For Block Size 60 words

The screenshot shows the 'New Module' configuration window. The 'Type' is set to '1769-MODULE Generic 1769 Module' and the 'Parent' is 'Local'. The 'Name' field is empty. The 'Description' field is empty. The 'Comm Format' is set to 'Data - INT'. The 'Slot' is set to 1. The 'Connection Parameters' section contains three rows: 'Input' with 'Assembly Instance' 101 and 'Size' 62 (16-bit); 'Output' with 'Assembly Instance' 100 and 'Size' 61 (16-bit); and 'Configuration' with 'Assembly Instance' 102 and 'Size' 0 (16-bit). At the bottom, the 'Open Module Properties' checkbox is checked, and there are 'OK', 'Cancel', and 'Help' buttons.

For Block Size 120 words

The 'New Module' dialog box is shown with the following settings:

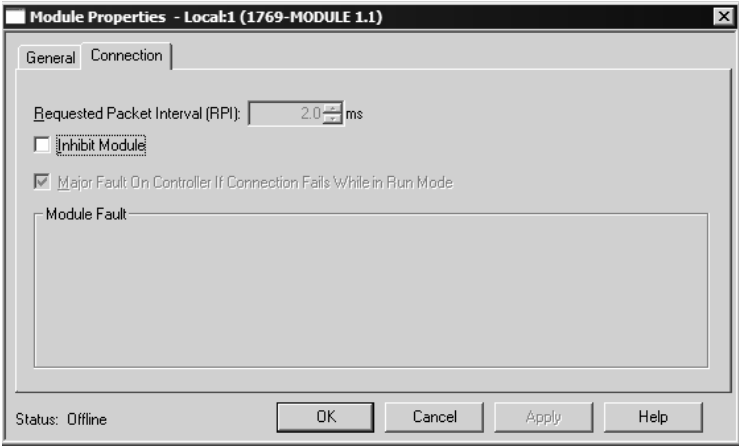
- Type: 1769-MODULE Generic 1769 Module
- Parent: Local
- Name: (empty text box)
- Description: (empty text box with up/down arrows)
- Comm Format: Data - INT (dropdown menu)
- Slot: 1 (spin box)
- Connection Parameters:
 - Input: Assembly Instance: 101, Size: 122 (16-bit)
 - Output: Assembly Instance: 100, Size: 121 (16-bit)
 - Configuration: Assembly Instance: 102, Size: 0 (16-bit)
- ☒ Open Module Properties
- Buttons: OK, Cancel, Help

For Block Size 240 words

The 'New Module' dialog box is shown with the following settings:

- Type: 1769-MODULE Generic 1769 Module
- Parent: Local
- Name: (empty text box)
- Description: (empty text box with up/down arrows)
- Comm Format: Data - INT (dropdown menu)
- Slot: 1 (spin box)
- Connection Parameters:
 - Input: Assembly Instance: 101, Size: 242 (16-bit)
 - Output: Assembly Instance: 100, Size: 241 (16-bit)
 - Configuration: Assembly Instance: 102, Size: 0 (16-bit)
- ☒ Open Module Properties
- Buttons: OK, Cancel, Help

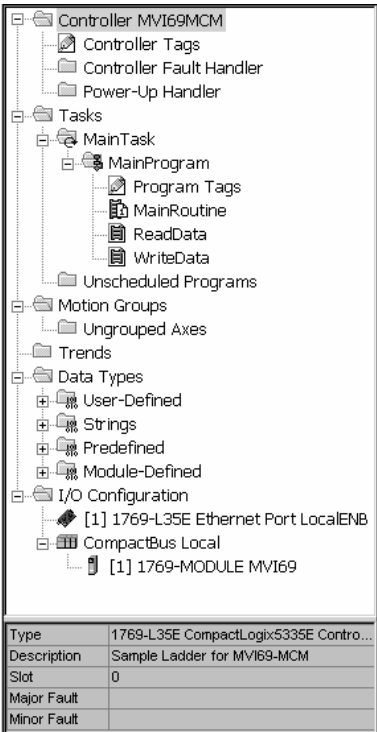
Fill in the dialog boxes as shown, adjusting the Name, Description and Slot options for your application. You must select the **Comm Format** as **Data - INT** in the dialog box. Failure to set the **Assembly Instance** and **Size** values correctly will result in a module that will not communicate over the backplane of the CompactLogix rack. Click Next to open the next dialog box.



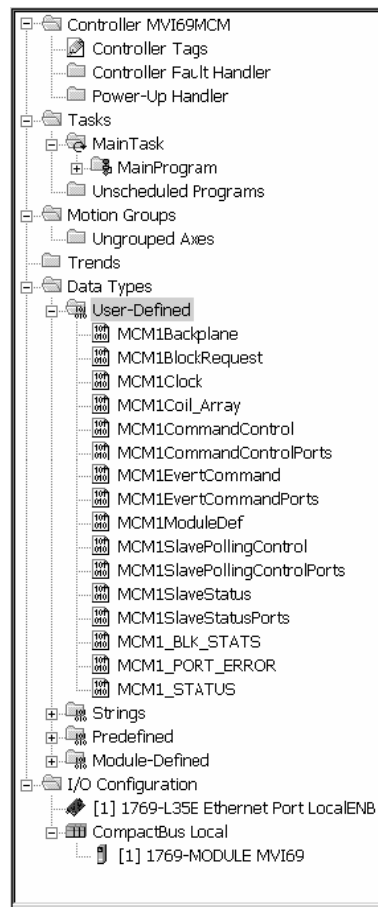
Select the Request Packet Interval value for scanning the I/O on the module. This value represents the minimum frequency the module will handle scheduled events. This value should not be set to less than 1 millisecond. Values between 1 and 10 milliseconds should work with most applications.

2.1.1 After you Complete the Module Setup

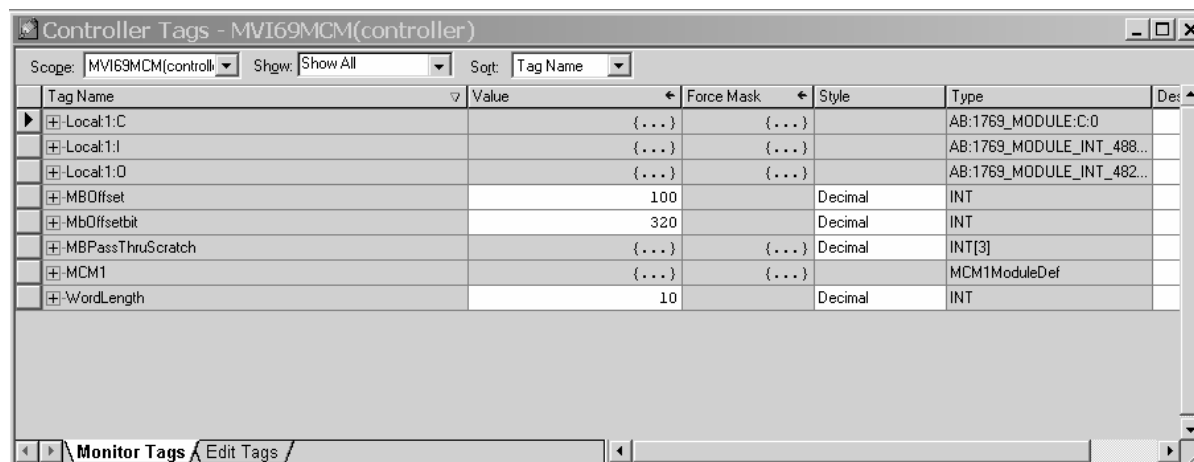
After you complete the module setup, the module will appear in the Controller Organization list. The data required for the module will be defined to the application, and objects will be allocated in the Controller Tags data area, as shown in the following illustration.



The next step is to define the User Defined Data Types to be used with the module. Copy these data types from the example ladder logic if you are not using the example. They will be defined if you are starting from the example ladder logic. The Controller Organization list should display the User Defined Data Types shown below:



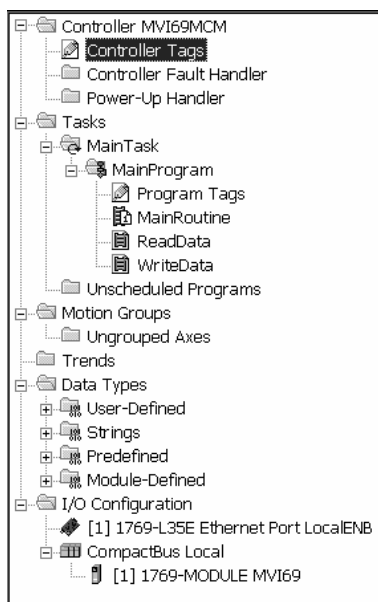
The next step in module setup is to define the data to be used to interface with the module and the ladder logic. Open the Controller Tags Edit Tags dialog box and enter the values shown in the following example. The MVI69-MCM module is defined in the example as MCM. You can set the tag name to any valid tag name you desire. If you are using the example ladder logic, this step has already been performed.



Tag Name	Value	Force Mask	Style	Type	Desc
Local:1:C	{...}	{...}		AB:1769_MODULE:C:0	
Local:1:I	{...}	{...}		AB:1769_MODULE_INT_488...	
Local:1:Q	{...}	{...}		AB:1769_MODULE_INT_482...	
MBOffset	100		Decimal	INT	
MbOffsetbit	320		Decimal	INT	
MBPassThruScratch	{...}	{...}	Decimal	INT[3]	
MCM1	{...}	{...}		MCM1ModuleDef	
WordLength	10		Decimal	INT	

At this point, take the time to fill in the configuration values in the MCM data table and adjust array sizes. Refer to the Module Data Object section of this document for information on configuring the module.

The last step is to add the ladder logic. If you are using the example ladder logic, adjust the ladder to fit your application. If you are not using the ladder example, copy the ladder logic shown in the Controller Organization list below to your application.



The module is now set up and ready to use with your application. Insert the module in the rack and attach the MODBUS serial communication cables. Download the new application to the controller and place the processor in run mode. If all the configuration parameters are set correctly and the module is attached to a MODBUS network, the module's Application LED (APP LED) should remain on and the backplane activity LED (BP ACT) should blink very rapidly. If you encounter errors, refer to **Diagnostics and Troubleshooting** (page 55) for information on how to connect to the module's Config/Debug port to use its troubleshooting features.

2.2 Installing and Configuring the Module with a MicroLogix Processor

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

This chapter describes how to install and configure the module to work with your application. The configuration process consists of the following steps.

- 1 Use RSLogix to identify the module to the processor and add the module to a project.

NOTE: The RSLogix software must be in "offline" mode to add the module to a project.

- 2 Modify the module's configuration files to meet the needs of your application, and copy the updated configuration to the module. Example configuration files are provided on the CD-ROM. Refer to the Modifying the Example Configuration File section, later in this chapter, for more information on the configuration files.
- 3 Modify the example ladder logic to meet the needs of your application, and copy the ladder logic to the processor. Example ladder logic files are provided on the CD-ROM.

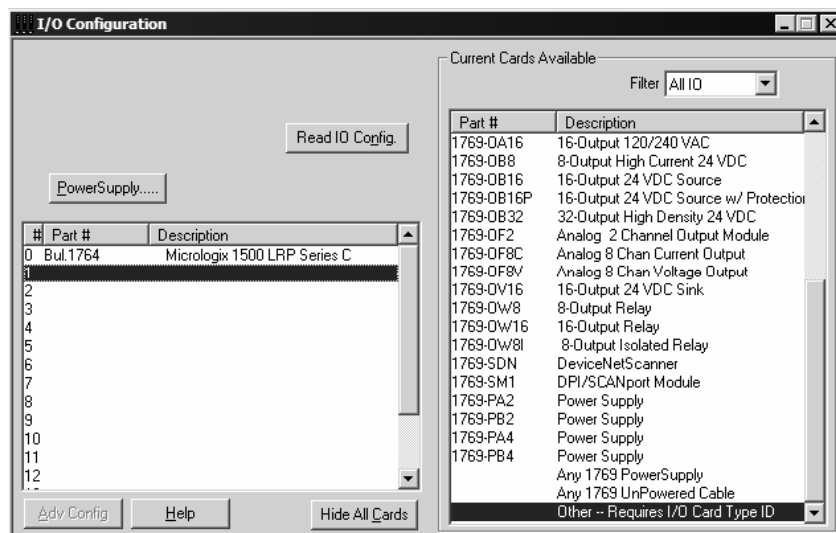
Note: If you are installing this module in an existing application, you can copy the necessary elements from the example ladder logic into your application.

The rest of this chapter describes these steps in more detail.

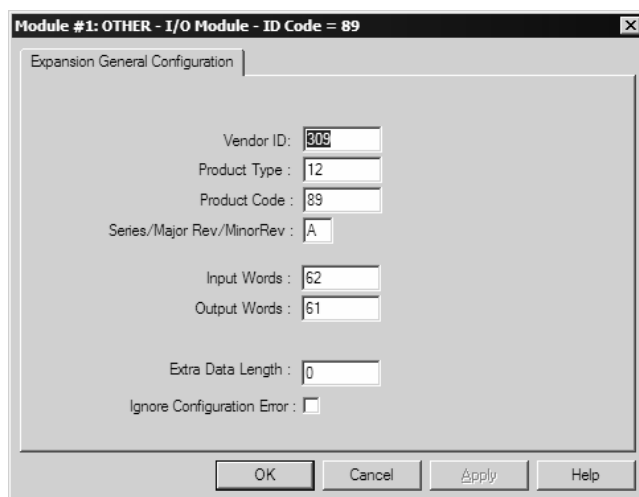
The first step in setting up the processor ladder file is to define the I/O type module to the system. Start RSLogix 500, and follow these steps:

- 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.



- 3 On 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 Module 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.



The input words and output words parameter 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 **Next** 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. Refer to the example Ladder Logic section in this manual.

Download the new application to the controller and place the processor in run mode. If you encounter errors, refer to **Diagnostics and Troubleshooting** (page 55) for information on how to connect to the module's Config/Debug port to use its troubleshooting features.

2.3 Module Data Object (MCMModuleDef)

All data related to the MVI69-MCM is stored in a user defined data type. An instance of the data type is required before the module can be used. This is done by declaring a variable of the data type in the Controller Tags Edit Tags dialog box. The structure of the object is displayed in the following figure.

Data Type: MCM1ModuleDef

Name:

Description:

Members:

Data Type Size: 4592

Name	Data Type	Style	Description
BlockTransferSize	INT	Decimal	
ReadData	INT[720]	Decimal	
WriteData	INT[720]	Decimal	
BP	MCM1Backplane		
ModuleStatus	MCM1_STATUS		
BlockRequest	MCM1BlockRequest		
ReadClock	MCM1Clock		
WriteClock	MCM1Clock		
CommandControl	MCM1CommandControlPorts		
EventCommand	MCM1EventCommandPorts		
SlavePollingControl	MCM1SlavePollingControlPorts		
SlaveStatus	MCM1SlaveStatusPorts		
MBCoil	MCM1Coil_Array		

Move Up Move Down OK Cancel Apply Help

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

This object reads and write data between the module and the processor. Values entered determine the ladder logic and data size required in the application. The ReadData and WriteData arrays must be sized to or larger than the count values entered. The ladder logic must process the number of blocks of data to be transferred. The number of blocks is computed as follows:

$$\text{BlockCnt} = \text{INT}(\text{RegCnt}/n) + \text{if}(\text{MOD}(\text{RegCnt}, n), 1, 0)$$

Where n is the block transfer size, and equals 60, 120 or 240.

If the register count is evenly divisible by n , the number of blocks is easy to compute and the ladder is much simpler to write. If the number is not evenly divisible by n , special handling of the last block of data must be developed, as it must transfer less than n words.

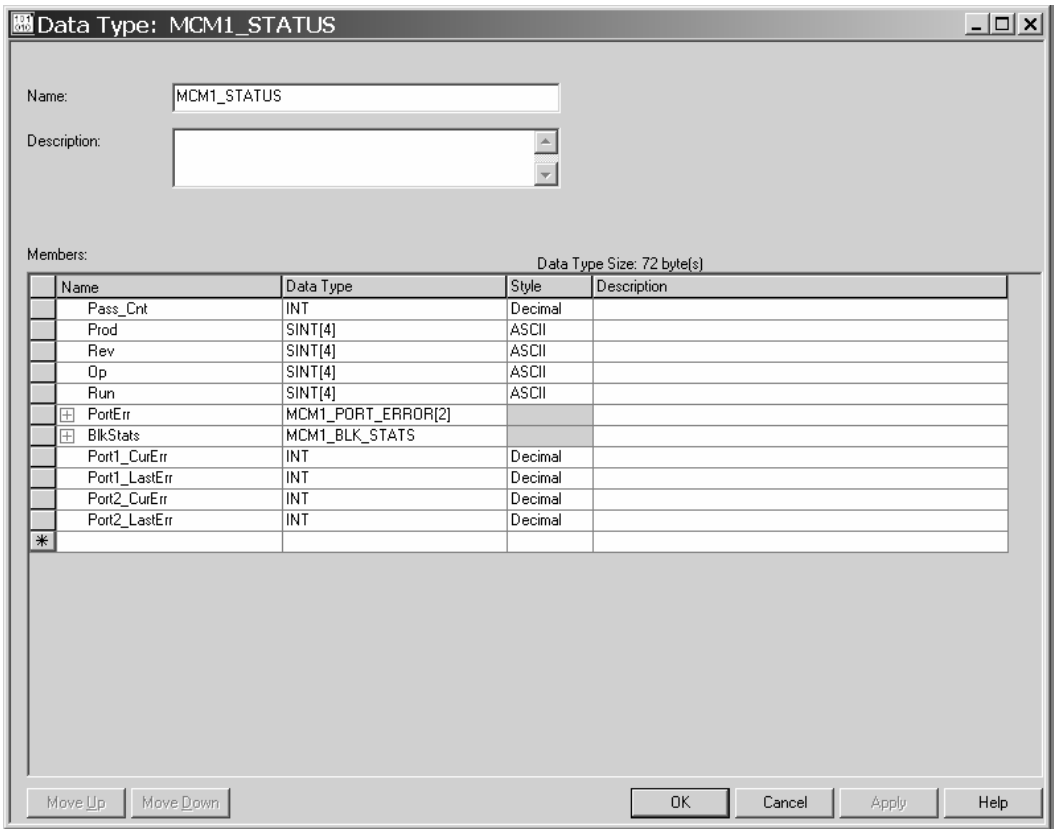
Important: It is recommended that the count values always be set to values evenly divisible by n.

The BPFail parameter determines if the module should continue communicating on the MODBUS network when the backplane transfer operation fails. A value of zero indicates that the module should continue communicating when the backplane is not operational. If the value is greater than zero, the backplane will be retried the entered number of times before a failure will be reported and communication will cease on the ports. When backplane communication is restored, the module will start communicating on the network. For example, if you enter a value of 10 for the parameter, the module will stop all MODBUS communications if 10 successive backplane errors are recognized. When a successful transfer is recognized, the module will resume communications on the network.

The ErrStatPtr parameter defines the location in the module's database where the error/status data will be stored. If the value is set to -1, the data will not be stored in the user data area. A value between 0 and 4939 will cause the module's program to store the data at the specified location.

2.3.1 *Status Object (MCM1Status)*

This object views the status of the module. The **MCM1Status** object shown below is updated each time a read block is received by the processor. Use this data to monitor the state of the module at a "real-time rate".



Refer to the Reference chapter for a complete listing of the data stored in this object.

2.3.2 *User Data Objects*

These objects hold data to be transferred between the processor and the MVI69-MCM module. The user data is the read and write data transferred between the processor and the module as "pages" of data up to 240 words long.

ReadData	INT[720]	Decimal	Data read from module
WriteData	INT[720]	Decimal	Data written to module

The read data (**ReadData**) is an array set to match the value entered in the **ReadRegCnt** parameter of the **MCMModule** object. For ease of use, this array should be dimensioned as an even increment of *n* words, where *n* = 60, 120 or 240 words. This data is paged up to *n* words at a time from the module to the processor. The ReadData task places the data received into the proper position

in the read data array. Use this data for status and control in the ladder logic of the processor.

The write data (**WriteData**) is an array set to match the value entered in the **WriteRegCnt** parameter of the **MCMModule** object. For ease of use, this array should be dimensioned as even increments of n words. This data is paged up to n words at a time from the processor to the module. The WriteData task places the write data into the output image for transfer to the module. This data is passed from the processor to the module for status and control information for use in other nodes on the network. If this array is > 480 registered, change the high LIM value in ReadData rung 1 and WriteData rung 21 of the ladder file.

2.3.3 *Slave Polling Control and Status*

Two arrays are allocated in the module's primary object to hold the polling status of each slave on the master ports. This status data can be used to determine which slaves are currently active on the port, are in communication error or have their polling suspended and disabled. Ladder logic in the processor can be written to monitor and control the status of each slave on a master port. The objects used are displayed in the following diagram:



Using special blocks, the processor can request the current data for the slaves. Through the use of other blocks, the processor can enable or disable the polling of selected slaves.

2.3.4 *MODBUS Message Data*

This new version of the module's program includes the pass-through mode. In this mode, write messages sent to a slave port are passed directly through to the processor. It is the responsibility of the ladder logic to process the message received using this feature. Two data objects are required for this mode: a variable to hold the length of the message and a buffer to hold the message. This information is passed from the module to the processor using a block identification code of 9996. Word two of this block contains the length of the message and the message starts at word 3. Other controller tags are required to store the controlled values contained in these messages. The MODBUS protocol supports controller of binary output (coils – functions 5 and 15) and registers (functions 6 and 16).

2.4 Command List Overview

In order to interface the MVI69-MCM module with MODBUS slave devices, you must construct a command list. The commands in the list specify the slave 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 master command list supports up to 100 commands.

The command list is processed from top (command #0) to bottom. A poll interval parameter is associated with each command to specify a minimum delay time in tenths of a second between the issuance of a command. If the user specifies a value of 10 for the parameter, the command will be executed no more frequently than every 1 second.

Write commands have a special feature, as they can be set to execute only if the data in the write command changes, which can improve network performance. If the register data values in the command have not changed since the command was last issued, the command will not be executed. To enable this feature, set the enable code for the command to a value of 2.

2.5 MODBUS Command Configuration

The ProSoft Technology MCM MODBUS Master and Slave communication drivers support several data read and write commands. When configuring a Master port, the decision on which command to use is made depending on the type of data being addressed, and the level of MODBUS support in the slave equipment. When configuring as a slave, it may be important to understand how the MODBUS commands function in order to determine how to structure the application data.

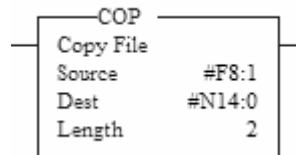
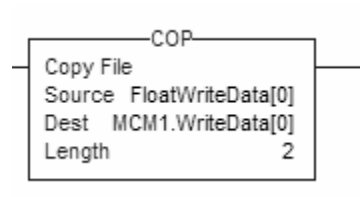
We have included an excerpt from the MODBUS Protocol Specification in the **Reference** (page 74) to assist in thoroughly understanding the functionality of the MODBUS protocol.

2.5.1 *Floating Point Support*

The movement of floating point data between the MCM module and other devices is easily accomplished as long as the device supports IEEE 754 Floating Point format. This IEEE format is a 32-bit single precision floating point format.

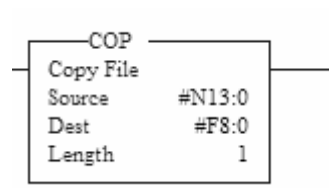
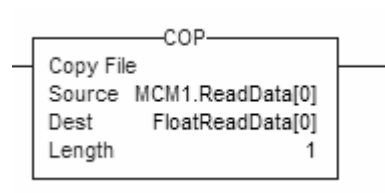
The programming necessary to move the floating point data is to take advantage of the COP command that exists in the Compact Logix and SLC processors. The COP command is unique in the CPX/SLC data movement commands in that it is an untyped function, meaning that no data conversion is done when moving data between file types (that is, it is an image copy not a value copy).

The structure of the COP command to move data from a Floating Point file into an integer file (something you would do to move floating point values to the module) is as follows:



This command will move one floating point value in two 16 bit integer images to the integer file. For multiple floating point values increase the count field by a factor of 2 per floating point value.

The structure of the COP command to move data from an Integer file to a Floating Point file (something you would do to receive floating point values from the module) is as follows:



This command will move two 16 bit integer registers containing one floating point value image to the floating point file. For multiple values increase the count field.

ENRON Floating Point Support

Many manufacturers have implemented special support in their drivers to support what is commonly called the Enron version of the MODBUS protocol. In this implementation, register addresses > 7000 are presumed to be floating point values. The significance to this is that the count field now becomes a 'number of values' field. In floating point format, each value represents two words.

Configuring the Floating Point Data Transfer

A common question when using the module as a Modbus Master is how floating point data is handled. This really depends on the slave device and how it addresses this application.

Just because your application is reading/writing floating point data, does not mean that you must configure the Float Flag, Float Start parameters within the module.

These parameters are only used to support what is typically referred to as Enron or Daniel Modbus, where one register address must have 32 bits, or one floating point value. Below is an example:

Example #1

Modbus Address	Data Type	Parameter
47101	32 bit REAL	TEMP Pump #1
47102	32 bit REAL	Pressure Pump #1
47103	32 bit REAL	TEMP Pump #2
47104	32 bit REAL	Pressure Pump #2

With the module configured as a master, you only need to enable these parameters to support a write to this type of addressing (Modbus FC 6 or 16).

If the slave device shows addressing as shown in Example #2, then you need not do anything with the Float Flag, Float Start parameters, as they use two Modbus addresses to represent one floating point value:

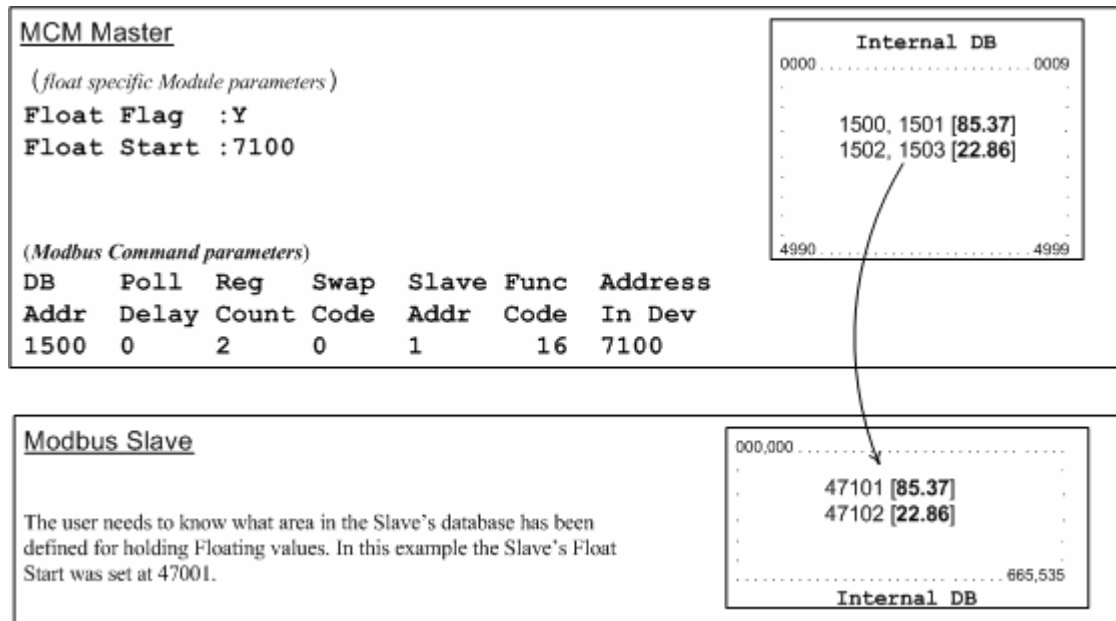
Example #2

Modbus Address	Data Type	Parameter
47101	32 bit REAL	TEMP Pump #1
47103	32 bit REAL	Pressure Pump #1
47105	32 bit REAL	TEMP Pump #2
47107	32 bit REAL	Pressure Pump #2

Because each 32 bit REAL value is represented by two Modbus Addresses (example 47101 and 47102 represent TEMP Pump #1), then you need not set the Float Flag, or Float Start for the module for Modbus FC 6 or 16 commands being written to the slave.

Below are specific examples:

Master is issuing Modbus command with FC 16 (with Float Flag: Yes) to transfer Float data to Slave.



(Float specific module parameters)

Float Flag: "Y" tells the Master to consider the data values that need to be sent to the Slave as floating point data where each data value is composed of 2 words (4 bytes or 32 bits).

Float Start: Tells the Master that if this address number is \leq the address number in "Addr in Dev" parameter to double the byte count quantity to be included in the Command FC6 or FC16 to be issued to the Slave. Otherwise the Master will ignore the "Float Flag: Y" and treat data as composed of 1 word, 2 bytes.

(Modbus Command parameters)

DB Addr – Tells the Master where in its data memory is the beginning of data to obtain and write out to the Slave (slave) device.

Reg Count – Tells the Master how many data points to send to the Slave. Two counts will mean two floating points with Float Flag: Y and the "Addr in Dev" \Rightarrow the "Float Start" Parameter.

Swap Code – Tells the Master how to orient the Byte and Word structure of the data value. This is device dependent. Check Command Entry formats Section.

Func Code – Tells the Master to write the float values to the Slave. FC16.

Addr in Dev – Tells the Master where in the Slave's database to locate the data.

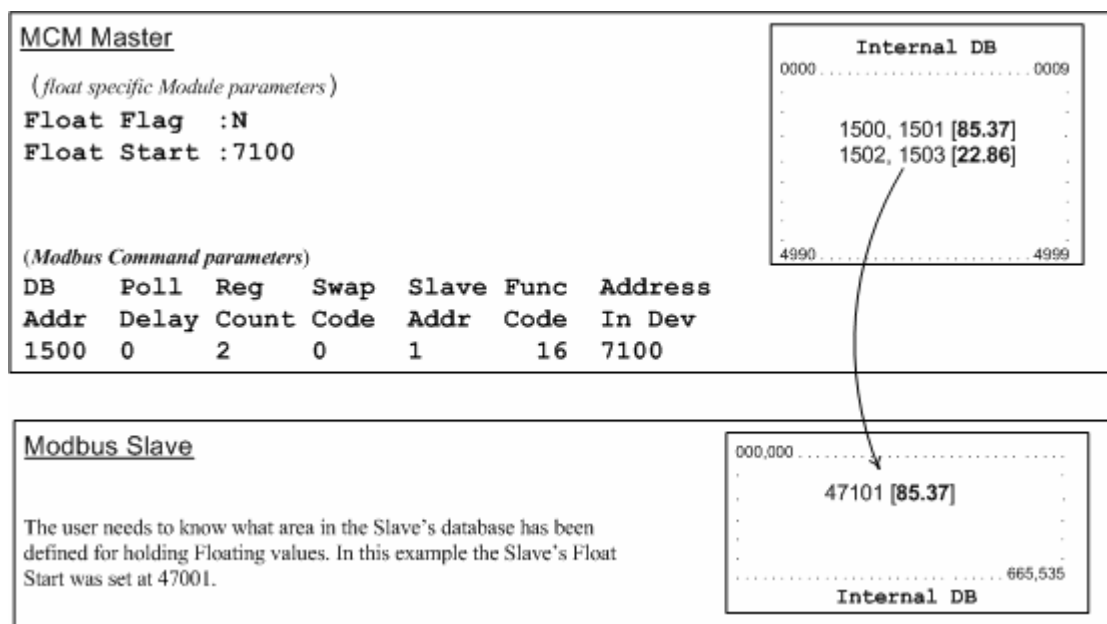
In the above example, the Master's Modbus command to transmit inside the Modbus packet will be as follows.

	Slave address	Function Code	Address in Device	Reg count	Byte Count	Data
DEC	01	16	7100	2	8	85.37 22.86
HEX	01	10	1B BC	00 02	08	BD 71 42 AA E1 48 41 B6

In conclusion

The Master's Modbus packet contains the data byte and data word counts that have been doubled from the amount specified by Reg Count due to the Float flag set to Y. Some Slaves look for the byte count in the data packet to know the length of the data to read from the wire. Other slaves know at which byte the data begins and read from the wire the remaining bytes in the packet as the data the Master is sending.

Master is issuing Modbus command with FC 16 (with Float Flag: No) to transfer Float data.



Float Flag: "N" tells the Master to ignore the floating values and treat each register data as a data point composed of 1 word, 2 bytes or 16 bits.

Float Start: Ignored.

DB Addr – same as when Float Flag: Y.

Reg Count – Tells the Master how many data points to send to the Slave.

Swap Code – same as when Float Flag: Y.

Func Code – same as when Float Flag: Y.

Addr in Dev – same as when Float Flag: Y as long as the Slave's Float Flag = Y.

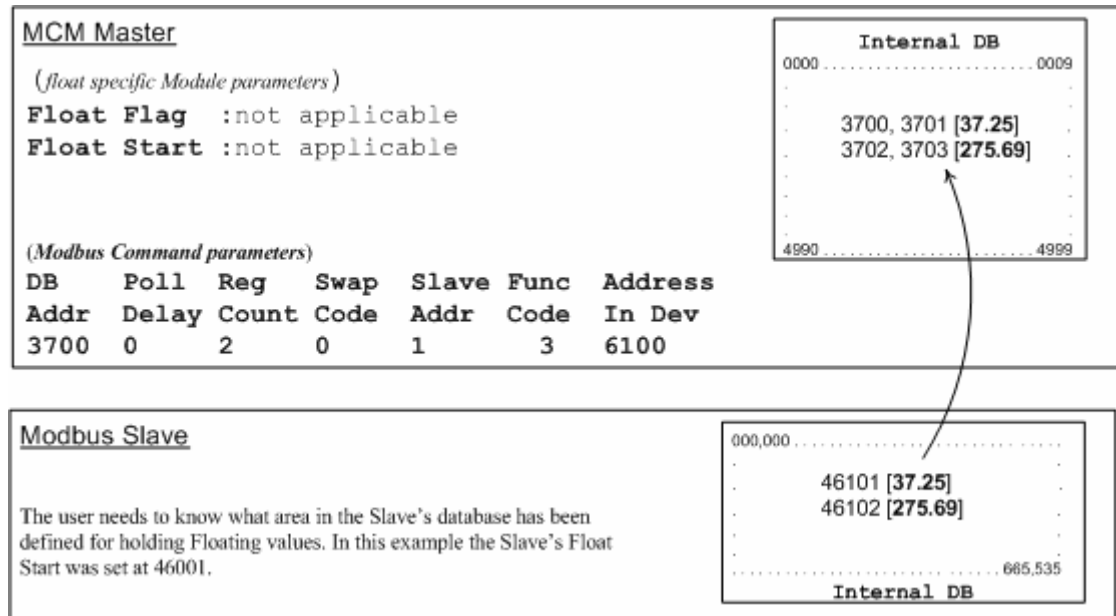
In the above example, the Master's Modbus command to transmit inside the Modbus packet will be as follows.

	Slave address	Function Code	Address in Device	Reg count	Byte Count	Data
DEC	01	16	7100	2	4	85.37
HEX	01	10	1B BC	00 02	04	BD 71 42 AA

In conclusion

The Master's Modbus packet contains the data byte and data word counts that have NOT been doubled from the amount specified by Reg Count due to the Float Flag set to N. The Slave looks for the byte count in the data packet to know the length of the data to read from the wire. Because of insufficient byte count, some slaves will read only half the data from the Master's transmission. Other slaves will read all 8 bytes in this example because they will know where in the packet the data starts and ignore the byte count parameter inside the Modbus packet.

Master is issuing Modbus command with FC 3 to transfer Float data from Slave.



Float Flag: Not applicable with Modbus Function Code 3.

Float Start: Not applicable with Modbus Function Code 3.

DB Addr – Tells the Master where in its data memory to store the data obtained from the Slave.

Reg Count – Tells the Master how many registers to request from the Slave.

Swap Code – same as above.

Func Code – Tells the Master to read the register values from the Slave. FC3.

Addr in Dev – Tells the Master where in the Slave's database to obtain the data.

In the above example, the Master's Modbus command to transmit inside the Modbus packet will be as follows.

	Slave address	Function Code	Address in Device	Reg count
DEC	01	3	6100	2
HEX	01	03	17 D4	00 02

In the above example the (Enron/Daniel supporting) Slave's Modbus command to transmit inside the Modbus packet will be as follows.

	Slave address	Function Code	Byte Count	Data
DEC	01	3	8	32.75 275.69
HEX	01	03	08	00 00 42 03 D8 52 43 89

In the above example the (a NON-Enron/Daniel supporting) Slave's Modbus command that will be transmitted inside the Modbus packet will be as follows.

	Slave address	Function Code	Byte Count	Data
DEC	01	3	4	32.75
HEX	01	03	04	00 00 42 03

2.5.2 Commands Supported by the Module

The format of each command in the list is dependent on the MODBUS Function Code being executed. The tables below list the functions supported by the module:

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

Each command list record has the same general format. The first part of the record contains the information relating to the communication module and the second part contains information required to interface to the MODBUS slave device.

2.5.3 Command Entry Formats

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

MODBUS Command Structure

Column #	1	2	3	4	5	6	7	8
Function Code	Enable Code	Internal Address	Poll Interval Time	Count	Swap Code	Slave Node	Function Code	Device MODBUS Address
fc1	Code	Bit	Seconds	Count	0	Address	1	Bit
fc2	Code	Bit	Seconds	Count	0	Address	2	Bit
fc3	Code	Register	Seconds	Count	Code	Address	3	Register
fc4	Code	Register	Seconds	Count	0	Address	4	Register
fc5	Code	Bit	Seconds	Count	0	Address	5	Bit
fc6	Code	Register	Seconds	Count	0	Address	6	Register
fc15	Code	Bit	Seconds	Count	0	Address	15	Bit
fc16	Code	Register	Seconds	Count	0	Address	16	Register

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

An example of a command list section of the CFG file is displayed below:

```
[MODBUS Port 1 Commands]
#           Internal      Poll      Reg  Swap      Node  MODBUS  MB Address
# Enable    Address  interval  Count  Code  Address  Func   in Device
START
           1           0           0     10       0         1         3           0
           1          10           0     10       0         1         4          10
END
```

Important: For bit commands, the addressable bit range can be 0 to 65535 (module memory registers 0 to 4095 * 16 bits per register). The structure of the module memory will not allow bit addressing of internal registers 4096 to 4999, as the bit number would be above 65535, the highest decimal number that can be represented with the 16 bit "Internal Address" field used by the module. Whenever Function codes (FCs) 1,2,5, or 15 are used by the module as a master, the Internal address must be specified as a bit address in the range of 0 to 65535.

The following table discusses each parameter. Each parameter is discussed below:

Command Parameter	Range	Description				
Enable	0,1,2	This field defines whether or not the command is to be executed and under what conditions.				
		<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>The command is disabled and will not be executed in the normal polling sequence.</td></tr></table>	Value	Description	0	The command is disabled and will not be executed in the normal polling sequence.
Value	Description					
0	The command is disabled and will not be executed in the normal polling sequence.					

Command Parameter	Range	Description										
		1 The command is executed each scan of the command list if the Poll Interval Time is set to zero. If the Poll Interval time is set, the command will be executed, when the interval timer expires.										
		2 The command will execute only if the internal data associated with the command changes. This value is valid only for write commands.										
Internal Address	0 to 4999 registers (0 to 65535 bits)	<p>This field specifies the internal database register to be associated with the command.</p> <ul style="list-style-type: none">If the command is a read function, the data read from the slave device will be placed starting at the register value entered in this field.If the command is a write function, the data written to the slave device will be sourced from the address specified.										
Poll Interval	0 to 65535	This parameter specifies the minimum interval to execute continuous commands (Enable code of 1). The parameter is entered in seconds. Therefore, if a value of 1 is entered for a command, the command will execute no more frequently than every 1 second.										
Count	Regs 1 to 125 Coils 1 to 2000	<p>This parameter specifies the number of registers or digital points to be associated with the command. Functions 5 and 6 ignore this field as they only apply to a single data point.</p> <p>For functions 1, 2 and 15, this parameter sets the number of digital points (inputs or coils) to be associated with the command.</p> <p>For functions 3, 4 and 16, this parameter sets the number of registers to be associated with the command.</p>										
Swap Code	0,1,2,3	<p>This parameter is used only for functions 3, 4, 6, and 16 to define if the data received (or sent) from the module is to be ordered differently than data received from the slave device. This parameter is helpful when dealing with floating-point or other multi-register values, as there is no standard method of storage of these data types in slave devices. This parameter can be set to order the register data received in an order useful by other applications. The following table defines the values and their associated operations:</p> <table><tr><th>Code</th><th>Description</th></tr><tr><td>0</td><td>None – No Change is made in the byte ordering</td></tr><tr><td>1</td><td>Words – The words are swapped</td></tr><tr><td>2</td><td>Words & Bytes – The words are swapped then the bytes in each word are swapped</td></tr><tr><td>3</td><td>Bytes – The bytes in each word are swapped</td></tr></table> <p>When swapping words, make sure you are using an even value in the Count Field. Odd values may generate unexpected results.</p>	Code	Description	0	None – No Change is made in the byte ordering	1	Words – The words are swapped	2	Words & Bytes – The words are swapped then the bytes in each word are swapped	3	Bytes – The bytes in each word are swapped
Code	Description											
0	None – No Change is made in the byte ordering											
1	Words – The words are swapped											
2	Words & Bytes – The words are swapped then the bytes in each word are swapped											
3	Bytes – The bytes in each word are swapped											
Slave Node	1 to 255 (0 is a broadcast)	<p>This parameter specifies the MODBUS slave node address on the network to be considered. Values of 1 to 255 are permitted. Most MODBUS devices only accept an address in the range of 1 to 247 so be careful. 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 this node address for read operations.</p>										

Command Parameter	Range	Description																		
Function Code	1,2,3,4,5,6, 15,16	<p>This parameter specifies the MODBUS function to be executed by the command. These function codes are defined in the MODBUS protocol. The following table defines the purpose of each function supported by the module. More information on the protocol is available from the Schneider Electric web site (www.modicon.com).</p> <table><tr><th>MODBUS Function Code</th><th>Description</th></tr><tr><td>1</td><td>Read Coil Status</td></tr><tr><td>2</td><td>Read Input Status</td></tr><tr><td>3</td><td>Read Holding Registers</td></tr><tr><td>4</td><td>Read Input Registers</td></tr><tr><td>5</td><td>Single Coil Write</td></tr><tr><td>6</td><td>Single Register Write</td></tr><tr><td>15</td><td>Multiple Coil Write</td></tr><tr><td>16</td><td>Multiple Register Write</td></tr></table>	MODBUS Function Code	Description	1	Read Coil Status	2	Read Input Status	3	Read Holding Registers	4	Read Input Registers	5	Single Coil Write	6	Single Register Write	15	Multiple Coil Write	16	Multiple Register Write
MODBUS Function Code	Description																			
1	Read Coil Status																			
2	Read Input Status																			
3	Read Holding Registers																			
4	Read Input Registers																			
5	Single Coil Write																			
6	Single Register Write																			
15	Multiple Coil Write																			
16	Multiple Register Write																			
Device Address		<p>This parameter specifies the starting MODBUS register or digital point address to be considered by the command in the MODBUS slave device. Refer to the documentation of each MODBUS slave device on the network for their register and digital point address assignments.</p> <p>The FC determines the addresses range and that this value will be the register or bit OFFSET into a given data range. For instance, if the command is to be a bit command (FC 1, 2, 5, or 15) to Read/Write a Coil 0X address 00001, then the value to enter here would be 0. For Coil address 00110, the value here would be 109. For register Read/Write commands (FC 3, 4, 6, or 16) in the 3X (FC4) or 4X (FC3), say 30001 or 40001, the value here would, again be 0. For 31101 or 41101, the value to enter for this parameter would be 1100.</p>																		

2.6 Uploading and Downloading the Configuration File

ProSoft modules are shipped with a pre-loaded configuration file. In order to edit this file, you must transfer the file from the module to your PC. After editing, you must transfer the file back to the module.

This section describes these procedures.

Important: The illustrations of configuration/debug menus in this section are intended as a general guide, and may not exactly match the configuration/debug menus in your own module. For specific information about the configuration/debug menus in your module, refer to The Configuration/Debug Menu.

2.6.1 *Required Hardware*

You can connect directly from your computer's serial port to the serial port on the module to view configuration information, perform maintenance, and send (upload) or receive (download) configuration files.

ProSoft Technology recommends the following minimum hardware to connect your computer to the module:

- 80486 based processor (Pentium preferred)
- 1 megabyte of memory
- At least one serial communications port available
- A null modem serial cable.

2.6.2 *Required Software*

In order to send and receive data over the serial port (COM port) on your computer to the module, you must use a communication program (terminal emulator).

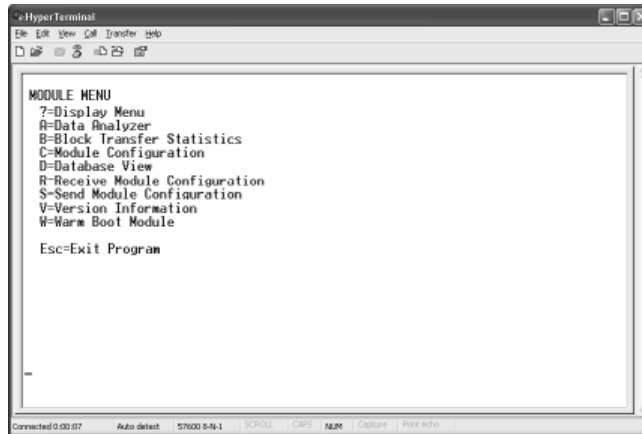
A simple communication program called HyperTerminal is pre-installed with recent versions of Microsoft Windows operating systems. If you are connecting from a machine running DOS, you must obtain and install a compatible communication program. The following table lists communication programs that have been tested by ProSoft Technology.

DOS	ProComm, as well as several other terminal emulation programs
Windows 3.1	Terminal
Windows 95/98	HyperTerminal
Windows NT/2000/XP	HyperTerminal

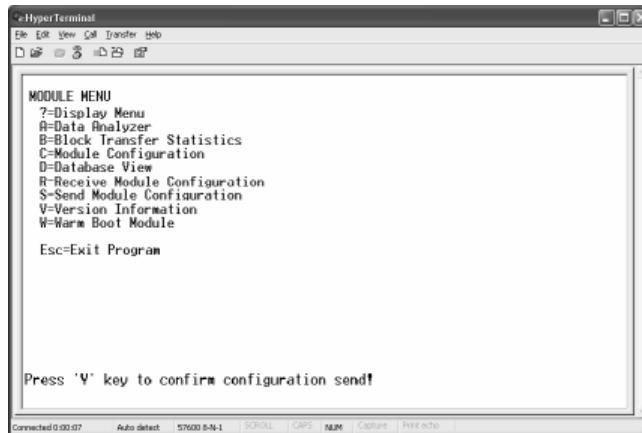
The module uses the Ymodem file transfer protocol to send (upload) and receive (download) configuration files from your module. If you use a communication program that is not on the list above, please be sure that it supports Ymodem file transfers.

2.6.3 *Transferring the Configuration File to Your PC*

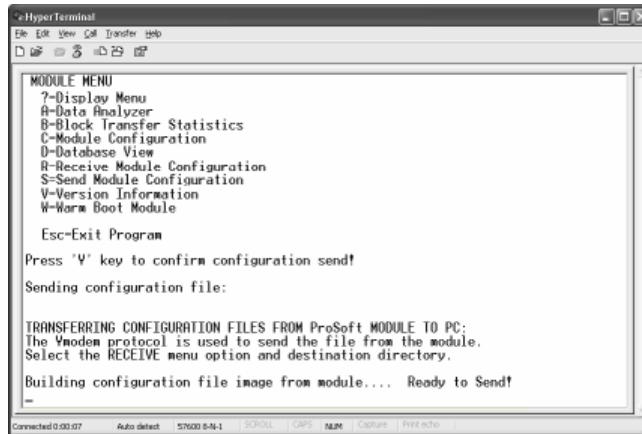
- 1 Connect your PC to the Configuration/Debug port of the module using a terminal program such as HyperTerminal. Press [?] to display the main menu.



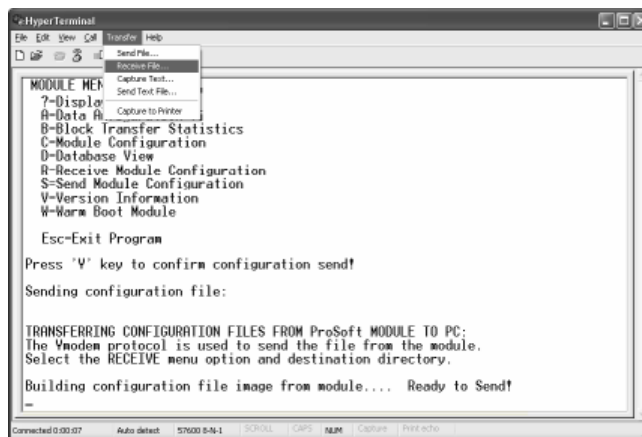
- 2 Press [S] (Send Module Configuration). The message "Press Y key to confirm configuration send!" is displayed at the bottom of the screen.



- 3 Press [Y]. The screen now indicates that the module is ready to send.



- 4 From the **Transfer** menu in HyperTerminal, select **Receive File**. This action opens the Receive File dialog box.



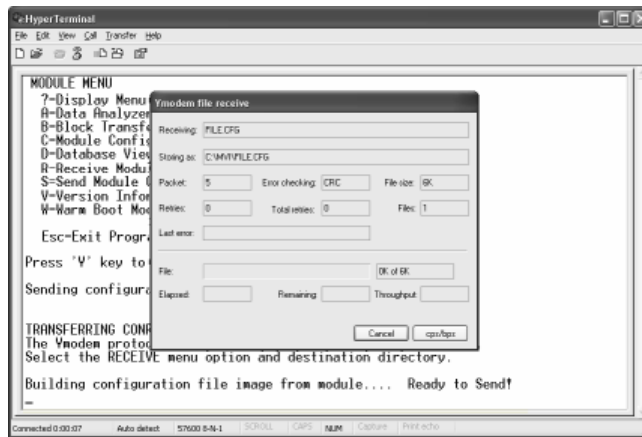
- 5 Use the Browse button to choose a folder on your computer to save the file, and then click Receive.



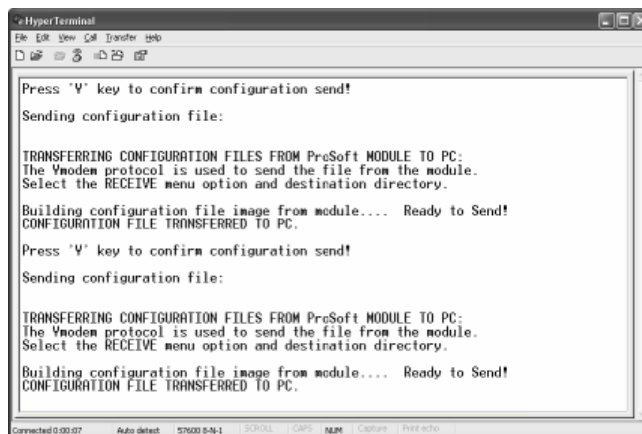
- **Note:** ProSoft Technology suggests that you upload the configuration file pre-loaded on your module. However, configuration files are also available on the ProSoft CD as well as the ProSoft Technology web site at <http://www.prosoft-technology.com>.

- 6 Select Ymodem as the receiving protocol.

- Click the Receive button. This action opens the Ymodem File Receive dialog box, showing the progress of your file transfer.



When the configuration file has been transferred to your PC, the dialog box will indicate that the transfer is complete.

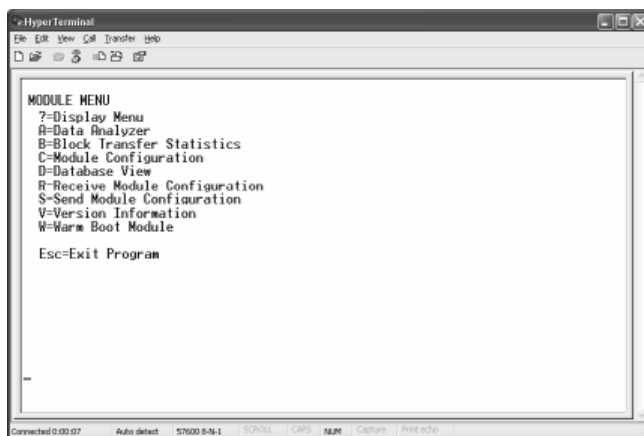


- The configuration file is now on your PC at the location you specified.
- You can now open and edit the file in a text editor such as Notepad. When you have finished editing the file, save it and close Notepad.

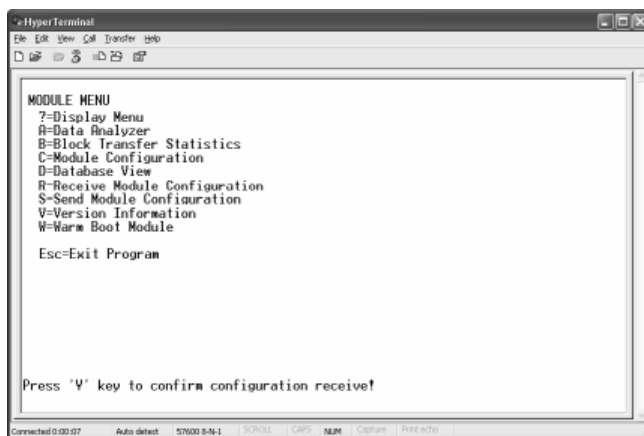
2.6.4 *Transferring the Configuration File to the Module*

Perform the following steps to transfer a configuration file from your PC to the module.

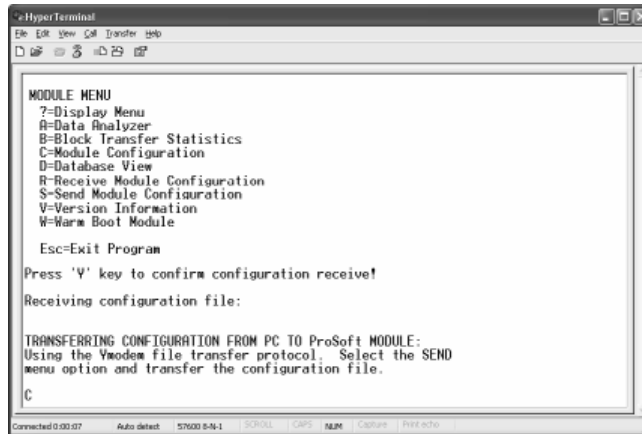
- 1 Connect your PC to the Configuration/Debug port of the module using a terminal program such as HyperTerminal. Press **[?]** to display the main menu.



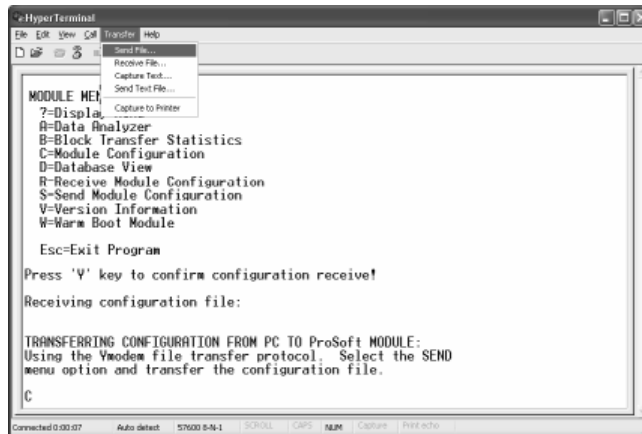
- 2 Press **[R]** (Receive Module Configuration). The message "Press Y key to confirm configuration receive!" is displayed at the bottom of the screen.



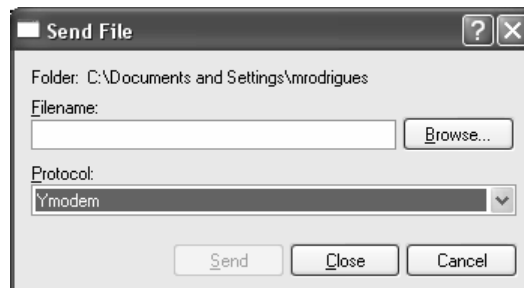
- 3 Press [Y]. The screen now indicates that the PC is ready to send.



- 4 From the **Transfer** menu in HyperTerminal, select **Send File**.



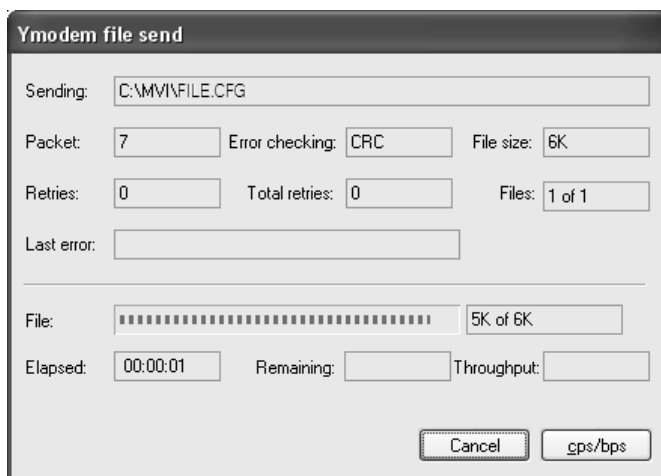
The Send File dialog appears.



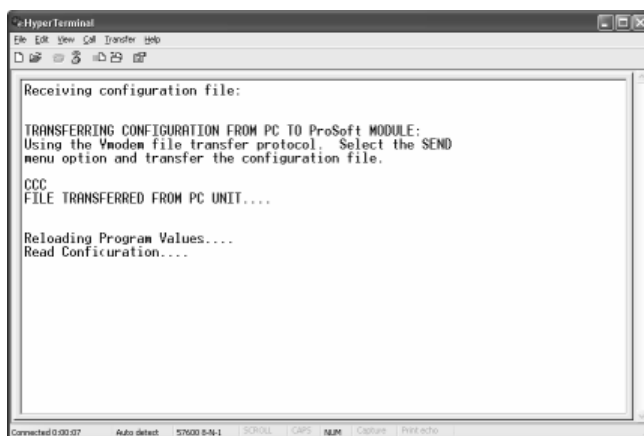
- 5 Use the Browse button to locate the configuration file your computer.

Note: This procedure assumes that you are uploading a newly edited configuration file from your PC to the module. However, configuration files are also available on the ProSoft CD as well as the ProSoft Technology web site at <http://www.prosoft-technology.com>.

- 6 Select **Ymodem** as the protocol.
- 7 Click the **Send** button. This action opens the Ymodem File Send dialog box.



When the file transfer is complete, the module's configuration/debug screen indicates that the module has reloaded program values, and displays information about the module.



- 8** Your module now contains the new configuration.

3 Ladder Logic

Ladder logic is required for application of the MVI69-MCM module. Tasks that must be handled by the ladder logic are module data transfer, special block handling and status data receipt. Additionally, a power-up handler may be needed to handle the initialization of the module's data and to clear any processor fault conditions.

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

4 Diagnostics and Troubleshooting

In This Chapter

- Reading Status Data from the module 55
- LED Status Indicators 72
- Clearing a Fault Condition..... 73
- Troubleshooting 73

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

- Status data values are transferred from the module to the processor.
- Data contained in the module can be viewed through the Configuration/Debug port attached to a terminal emulator.
- LED status indicators on the front of the module provide information on the module's status.

4.1 Reading Status Data from the module

The MVI69-MCM module returns a 29-word Status Data block that can be used to determine the module's operating status. This data is located in the module's database at registers 6670 to 6698 and at the location specified in the configuration. This data is transferred to the CompactLogix or MicroLogix processor continuously.

4.1.1 *The Configuration/Debug Menu*

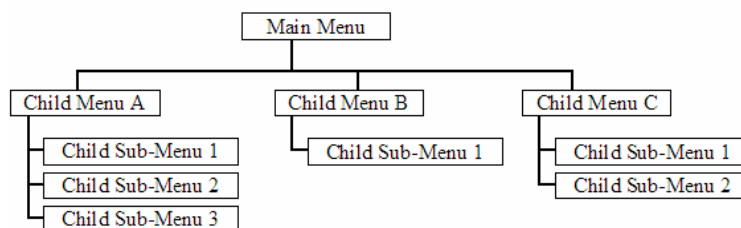
The Configuration and Debug menu for this module is arranged as a tree structure, with the Main Menu at the top of the tree, and one or more sub-menus for each menu command. The first menu you see when you connect to the module is the Main menu.

Because this is a text-based menu system, you enter commands by typing the command letter from your computer keyboard in the terminal application (for example, HyperTerminal). 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 sub-menus for this module contain commands to redisplay the menu or return to the previous menu. You can always return from a sub-menu 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 you the menus available for this module, and briefly discusses the commands available to you.

Keystrokes

The keyboard commands on these menus are almost always non-case sensitive. You can enter most commands in lower case or capital 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][/]**.

Also, take care to distinguish capital letter **[I]** from lower case letter **[i]** (L) and number **[1]**; likewise for capital letter **[O]** and number **[0]**. Although these characters look nearly the same on the screen, they perform different actions on the module.

4.1.2 Required Hardware

You can connect directly from your computer's serial port to the serial port on the module to view configuration information, perform maintenance, and send (upload) or receive (download) configuration files.

ProSoft Technology recommends the following minimum hardware to connect your computer to the module:

- 80486 based processor (Pentium preferred)
- 1 megabyte of memory
- At least one serial communications port available
- A null modem serial cable.

4.1.3 Required Software

In order to send and receive data over the serial port (COM port) on your computer to the module, you must use a communication program (terminal emulator).

A simple communication program called HyperTerminal is pre-installed with recent versions of Microsoft Windows operating systems. If you are connecting from a machine running DOS, you must obtain and install a compatible communication program. The following table lists communication programs that have been tested by ProSoft Technology.

DOS	ProComm, as well as several other terminal emulation programs
Windows 3.1	Terminal
Windows 95/98	HyperTerminal
Windows NT/2000/XP	HyperTerminal

The module uses the Ymodem file transfer protocol to send (upload) and receive (download) configuration files from your module. If you use a communication program that is not on the list above, please be sure that it supports Ymodem file transfers.

4.1.4 Using the Configuration/Debug Port

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

- 1 Connect your computer to the module's port using a null modem cable.
- 2 Start the communication program on your computer and configure the communication parameters with the following settings:

Baud Rate	57,600
Parity	None
Data Bits	8
Stop Bits	1
Software Handshaking	XON/XOFF

- 3 Open the connection. When you are connected, press the [?] key on your keyboard. If the system is set up properly, you will see a menu with the module name followed by a list of letters and the commands associated with them.

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

- 1 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.
- 2 Verify that RSLinx is not controlling the COM port. Refer to **Disabling the RSLinx Driver for the Com Port on the PC** (page 98).
- 3 Verify that your communication software is using the correct settings for baud rate, parity and handshaking.
- 4 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.

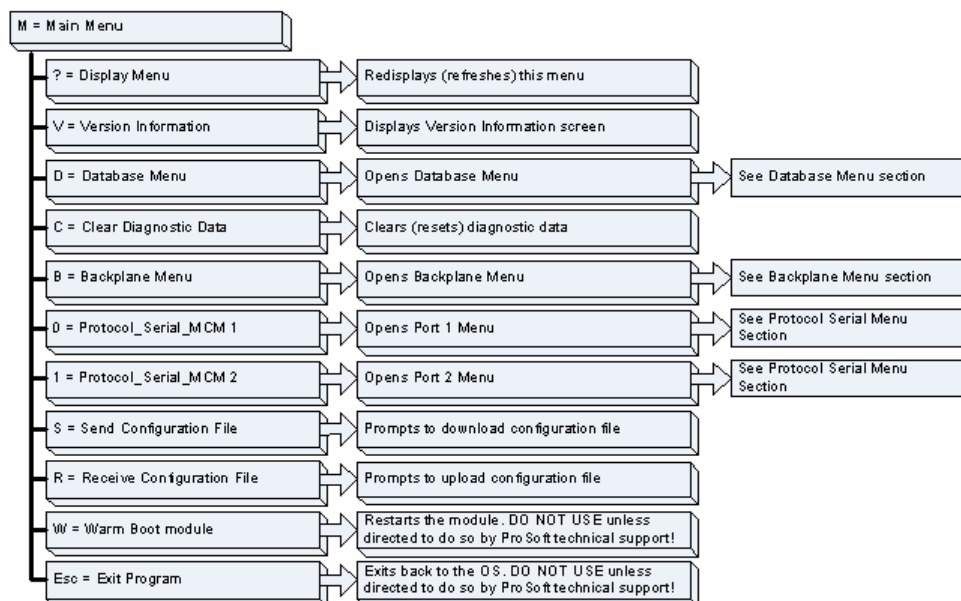
If you are still not able to establish a connection, you can contact ProSoft Technology Technical Support for further assistance.

4.1.5 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 on your terminal screen:

```
MVI69-MCM MENU
?=Display Menu
U=Version Information
D=Database Menu
C=Clear diagnostic data
B=Backplane Menu
0=Protocol_Serial_MCM 1
1=Protocol_Serial_MCM 2
S=Transfer Configuration from Unit to PC
R=Transfer Configuration from PC to Unit
W=Warm Boot Module
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.



Redisplaying the Menu

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

Viewing 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.

Opening the Database Menu

Press **[D]** to open the Database View menu. Use this menu command to view the current contents of the module's database.

Clearing Diagnostic Data

Press **[C]** to clear diagnostic data from the module's memory.

Opening the Backplane Menu

Press **[B]** from the Main Menu to view the Backplane Data Exchange List. Use this command to display the configuration and statistics of the backplane data transfer operations.

Tip: Repeat this command at one-second intervals to determine the number of blocks transferred each second.

Opening the Protocol Serial Menu

Press **[0]** or **[1]** to view the Protocol Serial Menu for ports 1 and 2, respectively.

Sending the Configuration File

Press **[S]** to upload (send) an updated configuration file to the module. For more information on receiving and sending configuration files, please see **Uploading and Downloading the Configuration File** (page 45).

Receiving the Configuration File

Press **[R]** to download (receive) the current configuration file from the module. For more information on receiving and sending configuration files, please see **Uploading and Downloading the Configuration File** (page 45).

Warm Booting the Module

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.

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 re-boot.

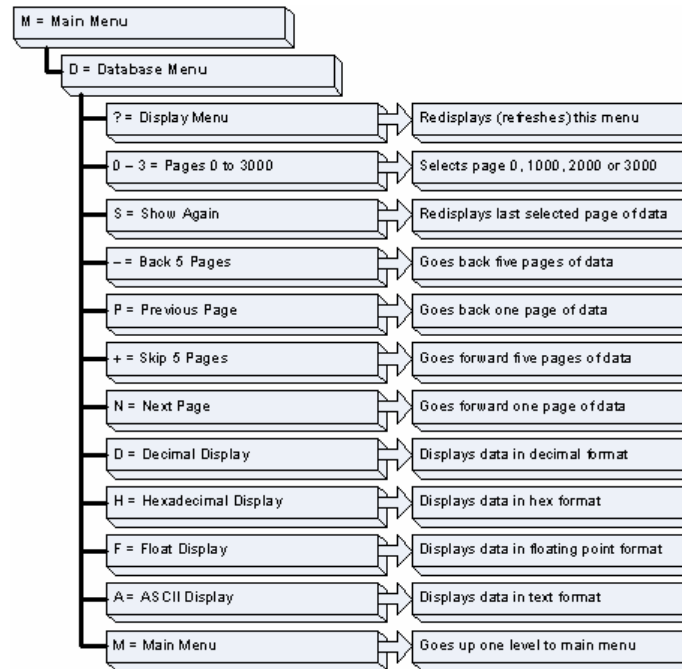
Exiting the Program

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.

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 ROM to configure the module.

4.1.6 Database View Menu

Press **[D]** from the Main Menu to open the Database View menu. Use this menu command to view the current contents of the module's database. Press **[?]** to view a list of commands available on this menu.



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.

Displaying the Current Page of Registers Again

DATABASE DISPLAY 0 TO 99 <DECIMAL>										
100	101	102	4	5	6	7	8	9	10	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	

This screen displays the current page of 100 registers in the database.

Moving Back Through 5 Pages of Registers

Press **[-]** from the Database View menu to skip back to the previous 500 registers of data.

Viewing the Previous 100 Registers of Data

Press **[P]** from the Database View menu to display the previous 100 registers of data.

Skipping 500 Registers of Data

Hold down **[Shift]** and press **[=]** to skip forward to the next 500 registers of data.

Viewing the Next 100 Registers of Data

Press **[N]** from the Database View menu to select and display the next 100 registers of data.

Viewing Data in Decimal Format

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

Viewing Data in Hexadecimal Format

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

Viewing Data in Floating Point Format

Press **[F]** from the Database View menu. Use this command 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

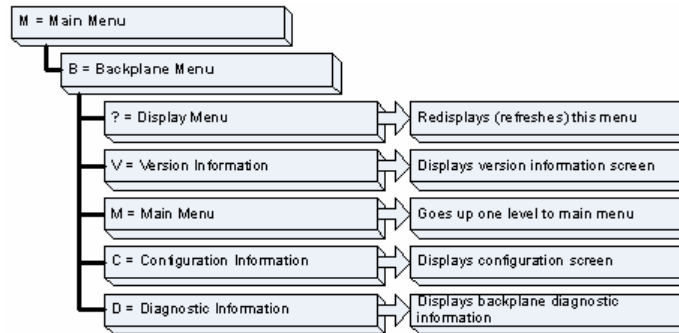
Press **[A]** to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Returning to the Main Menu

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

4.1.7 *Backplane Menu*

Press **[B]** from the Main Menu to view the Backplane Data Exchange List. Use this command to display the configuration and statistics of the backplane data transfer operations. Press **[?]** to view a list of commands available on this menu.



Redisplaying the Menu

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

Viewing 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.

Returning to the Main Menu

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

Viewing Configuration Information

Press **[C]** to view configuration information for the selected port, protocol, driver or device.

Viewing Backplane Diagnostic Information

Press **[D]** to view Backplane Diagnostic information.

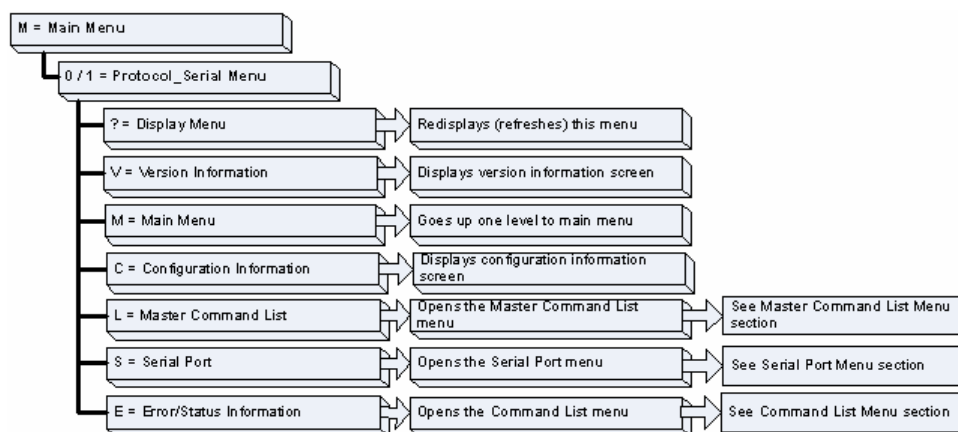
Use this command to display the configuration and statistics of the backplane data transfer operations between the module and the processor. The information

on this screen can help determine if there are communication problems between the processor and the module.

Tip: Repeat this command at one-second intervals to determine the number of blocks transferred each second

4.1.8 Protocol Serial MCM Menu

Press **[0]** or **[1]** to view protocol serial information for ports 1 and 2, respectively. Use this command to view a variety of error and status screens for the port. Press **[?]** to view a list of commands available on this menu.



Redisplaying the Menu

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

Viewing 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.

Returning to the Main Menu

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

Viewing Configuration Information

Press **[C]** to view configuration information for the selected port, protocol, driver or device.

Opening the Command List Menu

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

Opening the Serial Port Menu

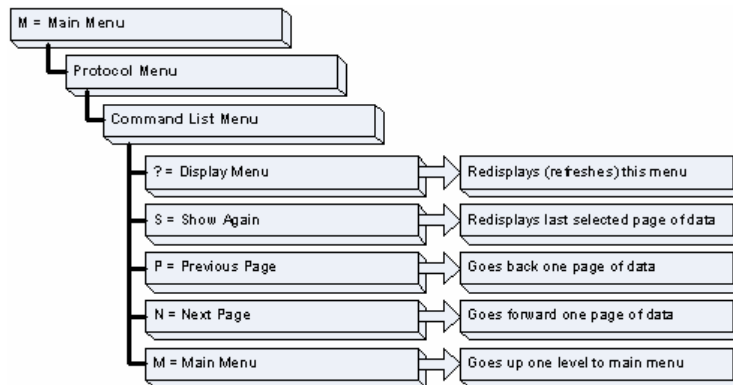
Press **[S]** to open the Serial Port menu. Use this command to view and change additional serial port driver settings.

Viewing Error and Status Data

Press **[E]** to display the error/status data for the module.

4.1.9 Master Command Error List Menu

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

Redisplaying the Current Page

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

Viewing the Previous 20 Commands

Press **[-]** to display data for the previous 20 commands.

Viewing the Previous Page of Commands

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

Viewing the Next 20 Commands

Press **[+]** to display data for the next 20 commands.

Viewing the Next Page of Commands

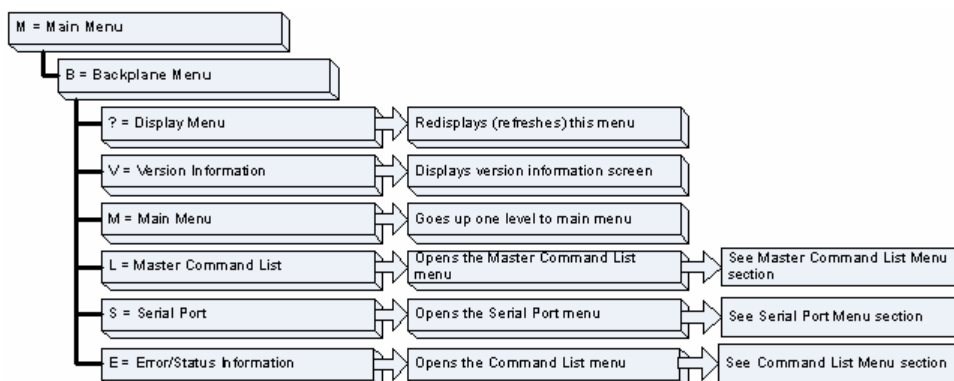
Press **[N]** to display the next page of commands.

Returning to the Main Menu

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

4.1.10 Serial Port Menu

Press **[S]** to open the Serial Port menu. Use this command to view and change additional serial port driver settings. Press **[?]** to view a list of commands available on this menu.



Redisplaying the Menu

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

Viewing 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.

Returning to the Main Menu

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

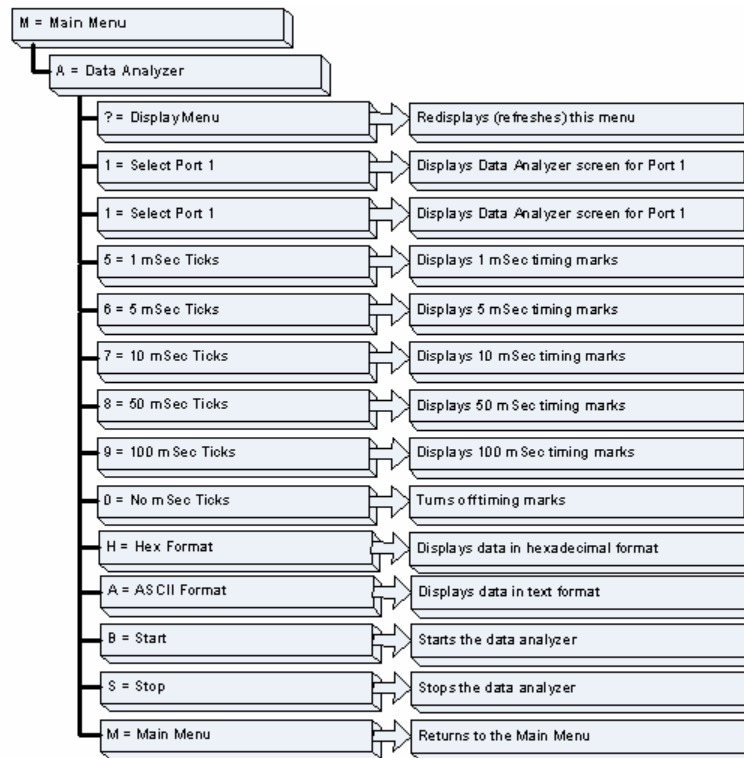
Opening the Data Analyzer Menu

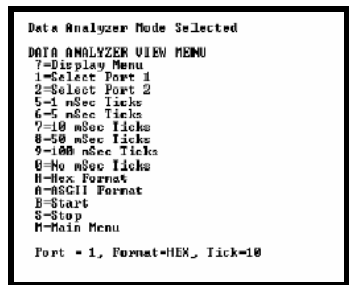
Press **[A]** to open the Data Analyzer Menu. Use this command to view all bytes of data transferred on each port. Both the transmitted and received data bytes are displayed. Refer to Data Analyzer for more information about this menu.

Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please be sure to press **[M]** to return to the main menu and disable the data analyzer. This action will allow the module to resume its normal operating mode.

4.1.11 Data Analyzer

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

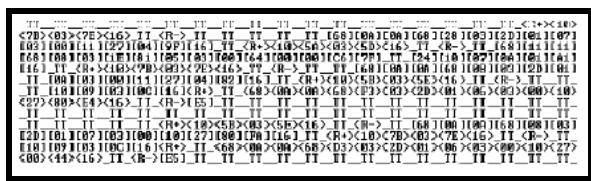




Important: When in analyzer mode, program execution will slow down. Only use this tool during a trouble-shooting session. Before disconnecting from the Config/Debug port, please be sure to press **[M]** to return to the main menu and disable the data analyzer. This action will allow the module to resume its normal operating mode.

Analyzing Data for Port 1

Press **[1]** to display I/O data for Port 1 in the Data Analyzer. The following illustration shows an example of the Data Analyzer output.



Analyzing Data for Port 2

Press **[2]** to display I/O data for Port 2 in the Data Analyzer.

Displaying Timing Marks in the Data Analyzer

You can display timing marks for a variety of intervals in the data analyzer screen. These timing marks can help you determine communication-timing characteristics.

Key	Interval
[5]	1 mSec ticks
[6]	5 mSec ticks
[7]	10 mSec ticks
[8]	50 mSec ticks
[9]	100 mSec ticks

Removing Timing Marks in the Data Analyzer

Press **[0]** to turn off timing marks in the Data Analyzer screen.

Returning to the Main Menu

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

4.1.12 Data Analyzer Tips

From the main menu, press **[A]** for the "Data Analyzer". You should see the following text appear on the screen:

Data Analyzer Mode Selected

After the "Data Analyzer" mode has been selected, press **[?]** to view the Data Analyzer menu. You will see the following menu:

```
DATA ANALYZER VIEW MENU
?=Display Menu
1=Select Port 1
2=Select Port 2
5=1 mSec Ticks
6=5 mSec Ticks
7=10 mSec Ticks
8=50 mSec Ticks
9=100 mSec Ticks
0=No mSec Ticks
H=Hex Format
A=ASCII Format
B=Start
S=Stop
M=Main Menu
```

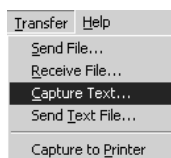
Port = 1, Format=HEX, Tick=10

From this menu, you can select the "Port", the "format", and the "ticks" that you can display the data in.

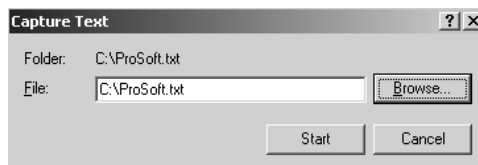
For most applications, HEX is the best format to view the data, and this does include ASCII based messages (because some characters will not display on HyperTerminal and by capturing the data in HEX, we can figure out what the corresponding ASCII characters are supposed to be).


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

After you have selected the Port, Format, and Tick, we are now ready to start a capture of this data. The easiest way to do so is to go up to the top of your HyperTerminal window, and do a Transfer -> Capture Text as shown below:



After selecting the above option, the following window will appear:



Next name the file, and select a directory to store the file in. In this example, we are creating a file ProSoft.txt and storing this file on our root C: drive. After you have done this, press the  button.

Now you have everything that shows up on the HyperTerminal screen being logged to a file called ProSoft.txt. This is the file that you will then be able to email to ProSoft Technical Support to assist with issues on the communications network.

To begin the display of the communications data, you will then want to press 'B' to tell the module to start printing the communications traffic out on the debug port of the module. After you have pressed 'B', you should see something like the following:

```
[03][00][04][00][05][00][06][00][07][00][08][00][09][FB][B7]_TT_TT_<R+><01><02>
<00><00><00><0A><F8><0D><R->_TT_TT_TT_[01][02][02][00][00][B9][B8]_TT_TT_<R+>
<01><03><00><00><00><0A><C5><CD><R->_TT_TT_[01][03][14][00][00][00][01][00]_TT
[02][00][03][00][04][00][05][00][06][00][07][00][08][00][09][CD][51]_TT_TT_<R+>
<01><01><00><00><00><A0><3C><72><R->_TT_TT_[01][01][14][00][00][01][00][02]_TT
[00][03][00][04][00][05][00][06][00][07][00][08][00][09][00][B7][52]_TT_TT_<R+>
<01><04><00><00><00><0A><70><0D><R->_TT_TT_[01][04][14][00][00][00][01][00]_TT
[02][00][03][00][04][00][05][00][06][00][07][00][08][00][09][FB][B7]_TT_TT_<R+>
<01><02><00><00><00><0A><F8><0D><R->_TT_TT_TT_[01][02][02][00][00][B9][B8]_TT
TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_TT_[01][03][14][00][00][00][01]
[00]_TT_TT_[02][00][03][00][04][00][05][00][06][00][07][00][08][09][CD][51]_TT
TT_<R+><01><01><00><00><00><A0><3C><72><R->_TT_TT_TT_[01][01][14][00][00][01]
[00][02]_TT_TT_[00][03][00][04][00][05][00][06][00][07][00][08][00][09][00][B7][52]
TT_TT_<R+><01><04><00><00><00><0A><70><0D><R->_TT_TT_[01][04][14][00][00][00]
[01][00]_TT_TT_[02][00][03][00][04][00][05][00][06][00][07][00][08][09][FB][B7]
TT_TT_<R+><01><02><00><00><00><0A><F8><0D><R->_TT_TT_[01][02][02][00][00][B9]
[B8]_TT_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_TT_[01][03][14][00][00]
[00][01][00]_TT_TT_[02][00][03][00][04][00][05][00][06][00][07][00][08][09][CD]
[51]_TT_TT_<R+><01><01><00><00><00><A0><3C><72><R->_TT_TT_TT_[01][01][14][00]
[00][01][00][02]_TT_TT_[00][03][00][04][00][05][00][06][00][07][00][08][00][09][00]
[B7][52]_TT_TT_<R+><01><04><00><00><00><0A><70><0D><R->_TT_TT_[01][04][14][00]
[00][00][01][00]_TT_TT_[02][00][03][00][04][00][05][00][06][00][07][00][08][09]
[FB][B7]_TT_TT_<R+><01><02><00><00><00><0A><F8><0D><R->_TT_TT_TT_[01][02][02]
[00][00][B9][B8]_TT_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_TT_
```

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

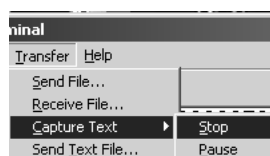
All characters shown in <> brackets are characters being sent out by the module.

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

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

After taking a minute or two of traffic capture, you will now want to stop the "Data Analyzer". To do so, press the 'S' key, and you will then see the scrolling of the data stop.

When you have captured the data you want to save, open the Transfer menu and choose Capture Text. On the secondary menu, choose Stop.



You have now captured, and saved the file to your PC. This file can now be used in analyzing the communications traffic on the line, and assist in determining communication errors.

4.2 LED Status Indicators

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

ProSoft Module	Color	Status	Indication
CFG	Green	On	Data is being transferred between the module and a remote terminal using the Configuration/Debug port.
		Off	No data is being transferred on the Configuration/Debug port.
P1	Green	On	Data is being transferred between the module and the MODBUS network on Port 1.
		Off	No data is being transferred on the port.
P2	Green	On	Data is being transferred between the module and the MODBUS network on Port 2.
		Off	No data is being transferred on the port.
APP	Amber	On	The MVI69-MCM is functioning normally.
		Off	The MVI69-MCM module program has recognized a communication error between the module and the processor.
BP ACT	Amber	On	The LED is on when the module is performing a write operation on the backplane.
		Off	The LED is off when the module is performing a read operation on the backplane. Under normal operation, the LED should blink rapidly on and off.
OK	Red/ Green	Off	The card is not receiving any power and is not securely plugged into the rack.
		Green	The module is operating normally.
		Red	The program has detected an error or is being configured. If the LED remains red for 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 the factory, as this is not a user serviceable item.

During module configuration, the OK LED will be red and the BP ACT LED will be on.

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

4.3 Clearing a Fault Condition

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

To clear the condition, follow these steps:

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

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

4.4 Troubleshooting

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

Processor Errors

Problem Description	Steps to take
Processor Fault	Verify that the module is plugged into the slot that has been configured for the module. Verify that the slot in the rack configuration has been set up correctly in the ladder logic.
Processor I/O LED flashes	This indicates a problem with backplane communications. Verify that all modules in the rack are configured in the ladder logic.

Module Errors

Problem Description	Steps to take
BP ACT LED remains off or blinks slowly	This indicates that backplane transfer operations are failing. Connect to the module's Configuration/Debug port to check this. To establish backplane communications, verify the following items: <ul style="list-style-type: none">▪ The processor is in Run mode.▪ The backplane driver is loaded in the module.▪ The module is configured for read and write block data transfer.▪ The ladder logic handles all read and write block situations.▪ The module is configured in the processor.
OK LED remains red	The program has halted or a critical error has occurred. Connect to the Configuration/Debug port to see if the module is running. If the program has halted, turn off power to the rack, remove the card from the rack and re-insert the card in the rack, and then restore power to the rack.

5 Reference

In This Chapter

➤ Product Specifications.....	75
➤ Functional Overview.....	77
➤ Cable Connections.....	97
➤ MCM Database Definition	103
➤ Status Data Definition	103
➤ Configuration Data Definition	105

5.1 Product Specifications

The MV69 Modbus Communication Module allows Rockwell Automation CompactLogix processors to interface easily with other Modbus protocol compatible devices.

Compatible devices include not only Modicon PLCs (which all support the Modbus protocol) but also a wide assortment of end devices. The module acts as an input/output module between the Modbus network and the CompactLogix backplane. The data transfer from the processor is asynchronous from the actions on the Modbus network. A 5000-word register space in the module exchanges data between the processor and the Modbus network.

5.1.1 Features and Benefits

The inRAx Modbus Master/Slave Communications module (MVI69-MCM) is designed to allow CompactLogix processors to interface easily with Modbus protocol-compatible devices and hosts.

Many host SCADA packages support the Modbus protocol, while devices commonly supporting the protocol include several PLCs, as well as many other third party devices in the marketplace. (For a partial list of devices that speak Modbus, please visit the ProSoft Tested section of the ProSoft web site).

5.1.2 General Specifications

- Single Slot – 1769 backplane compatible
- The module is recognized as an Input/Output module and has access to processor memory for data transfer between processor and module
- Ladder Logic is used for data transfer between module and processor. Sample ladder file included.
- Configuration data obtained from configuration text file downloaded to module. Sample configuration file included.
- Supports all CompactLogix processors: L20/L30/L31/L32/L35/L43 v16 only
- Supports MicroLogix 1500 Controller with LRP firmware version 6.0 or newer.

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
Operating Temp.	0 to 60°C (32 to 140°F)
Storage Temp.	40 to 85°C (40 to 185°F)
Relative Humidity	5 to 95% (non-condensing)
LED Indicators	Power and Module Status Application Status Serial Port Activity Serial Activity and Error Status
CFG Port (CFG)	RJ45 (DB-9F with supplied cable) RS-232 only No hardware handshaking
App Ports (P1,P2) (Serial modules)	RS-232, RS-485 or RS-422 (jumper selectable) RJ45 (DB-9F with supplied cable) RS-232 handshaking configurable 500V Optical isolation from backplane
Shipped with Unit	RJ45 to DB-9M cables for each port 6-foot RS-232 configuration Cable

5.1.4 Functional Specifications

Type	Specifications
Communication parameters (configurable)	Port 1: Baud Rate: 110 to 38.4K baud Port 2,3: Baud Rate: 110 to 115K baud Stop Bits: 1 or 2 Data Size: 5 to 8 bits Parity: None, Odd, Even RTS Timing delays: 0 to 65535 ms
Modbus Modes	RTU Mode (binary) with CRC-16 ASCII mode with LRC error checking
Floating Point Data	Floating point data movement supported, including configurable support for Enron implementation

Modbus Slave Protocol Specifications

The ports on the MVI69-MCM module can be individually configured to support the slave mode of the Modbus protocol. When in slave mode, the module can accept Modbus commands from a master to read/write data stored in the module's internal registers. This data is easily transferred to the CompactLogix processor's data registers.

Modbus Slave Driver

Node Address	1 to 247 software selectable
Status Data	Error codes, counters, and port status available per configured slave port starting at memory register 4400
Modbus Function Codes	1: Read Output Status 2: Read Input Status 3: Read Multiple Data Registers 4: Read Input Registers 5: Write Single Bit 6: Write Single Data Register 15: Write Multiple Bits 16: Write Multiple Data Register

Modbus Master Protocol Specifications

The ports on the MVI69-MCM module can be individually configured as Master ports. When configured in master mode, the MCM module is capable of reading and writing data to remote Modbus devices, enabling the CompactLogix platform to act as a SCADA sub-master, or a device data concentrator.

Modbus Master Driver

Command List	Up to 100 commands per master port, each fully configurable for function, slave address, register to/from addressing and word/bit count
Status Data	Error codes available on an individual command basis. In addition, a slave status list is maintained per active Modbus master port
Polling of Command List	Configurable polling of command list, including continuous and on change of data
Modbus Function Codes	1: Read Output Status 2: Read Input Status 3: Read Multiple Data Registers 4: Read Input Registers 5: Write Single Bit 6: Write Single Data Register 15: Write Multiple Bits 16: Write Multiple Data Register

5.2 Functional Overview

This section provides an overview of how the MVI69-MCM module transfers data using the MCM protocol. You should understand the important concepts in this chapter before you begin installing and configuring the module.

5.2.1 General Concepts

The following discussion explains several concepts that are important for understanding the operation of the MVI69-MCM module.

About the MODBUS 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 master and slave devices on a network. Later enhancements to the protocol allow communication over other types of networks.

MODBUS is a master/slave protocol. The master establishes a connection to the remote slave. When the connection is established, the master sends the MODBUS commands to the slave. The MVI69-MCM module works both as a master and as a slave.

The MVI69-MCM module acts as an input/output module between devices on a MODBUS network and the Rockwell Automation backplane. The module uses an internal database to pass data and commands between the processor and the master and slave devices on the MODBUS network.

Module Power Up

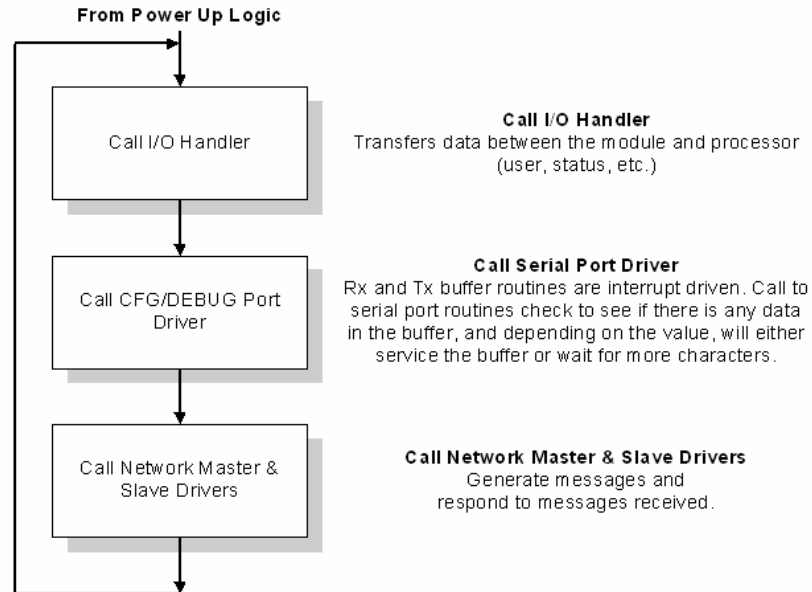
On power up the module begins performing the following logical functions:

- 1 Initialize hardware components
 - Initialize CompactLogix or MicroLogix backplane driver
 - Test and Clear all RAM
 - Initialize the serial communication ports
- 2 Module configuration
- 3 Initialize Module Register space
- 4 Enable Slave Driver on selected ports
- 5 Enable Master Driver on selected ports

After this initialization procedure is complete, the module will begin communicating with other nodes on the network, depending on the configuration.

Main Logic Loop

Upon completing the power up configuration process, the module enters an infinite loop that performs the following functions:



Backplane Data Transfer

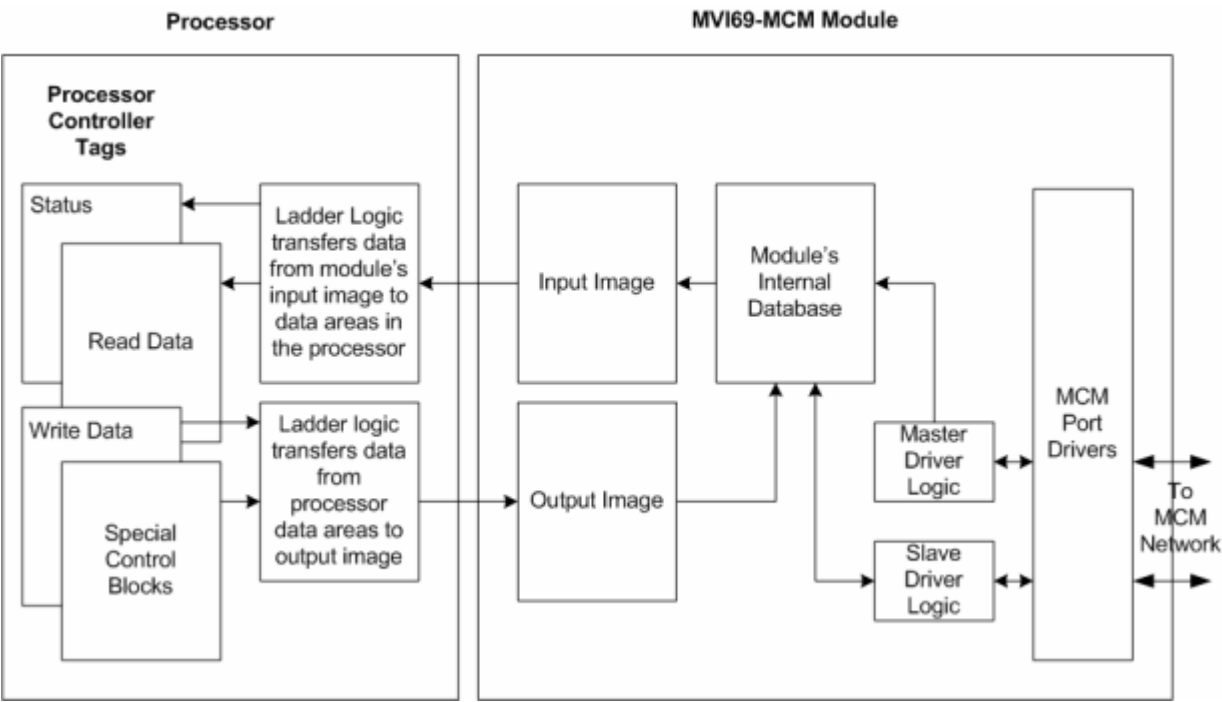
The MVI69-MCM module communicates directly over the CompactLogix or MicroLogix backplane. Data is paged between the module and the CompactLogix or MicroLogix processor across the backplane using the module's input and output images. The update frequency of the images is determined by the scheduled scan rate defined by the user for the module and the communication load on the module. Typical updates are in the range of 2 to 10 milliseconds.

The data is paged between the processor and the module using input and output image blocks. You can configure the size of the blocks using the Block Transfer Size parameter in the configuration file. You can configure blocks of 60, 120, or 240 words of data depending on the number of words allowed for your own application.

This bi-directional transference of data is accomplished by the module filling in data in the module's input image to send to the processor. Data in the input image is placed in the Controller Tags in the processor by the ladder logic. The input image for the module may be set to 62, 122, or 242 words depending on the block transfer size parameter set in the configuration file.

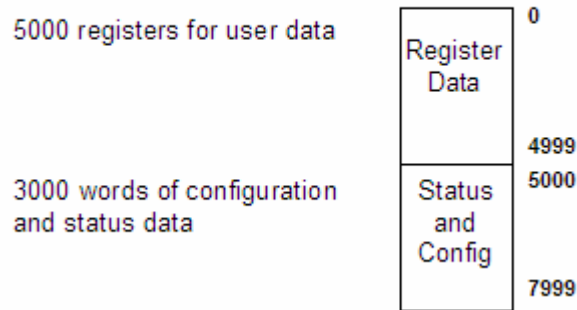
The processor inserts data to the module's output image to transfer to the module. The module's program extracts the data and places it in the module's internal database. The output image for the module may be set to 61, 121, or 241 words depending on the block transfer size parameter set in the configuration file.

The following illustration shows the data transfer method used to move data between the CompactLogix or MicroLogix processor, the MVI69-MCM module and the MODBUS network.



As shown in the diagram above, 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 CompactLogix or MicroLogix processor to interface the input and output image data with data defined in the Controller Tags. All data used by the module is stored in its internal database. The following illustration shows the layout of the database:

Module's Internal Database Structure



Data contained in this database is paged through the input and output images by coordination of the CompactLogix or MicroLogix ladder logic and the MVI69-MCM module's program. Up to 242 words of data can be transferred from the module to the processor at a time. Up to 241 words of data can be transferred from the processor to the module. The read and write block identification codes in each data block determine the function to be performed or the content of the data block. The block identification codes used by the module are listed below:

Block Range	Descriptions
-1	Status Block
0	Status Block
1 to 84	Read or write data
1000	Event Port 1
2000	Event Port 2
3000 to 3001	Port 1 slave polling control
3002 to 3006	Port 1 slave status
3100 to 3101	Port 2 slave polling control
3102 to 3106	Port 2 slave status
5000 to 5006	Port 1 command control
5100 to 5106	Port 2 command control
9958	Function Code 5 data formatted Pass-Thru Control Blocks
9956 and 9957	Function Code 6 and 16 (Floating-point) data formatted Pass-Thru Control Block
9959	Function Code 15 data formatted Pass-Thru Control Block
9998	Warm-boot control block
9999	Cold-boot control block

Each image has a defined structure depending on the data content and the function of the data transfer as defined in the following topics.

5.2.2 Normal Data Transfer

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

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:

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

where

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

The Read Block ID is an index value used to determine the location of where the data will be placed in the processor controller tag array of module read data. 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. Under normal, program operation, the module sequentially sends read blocks and requests write blocks. For example, if three read and two write blocks are used with the application, the sequence will be as follows:

R1W1→R2W2→R3W1→R1W2→R2W1→R3W2→R1W1→

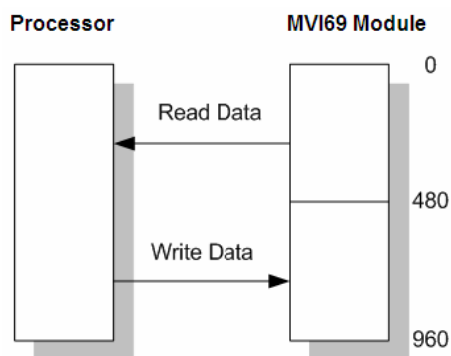
This sequence will continue until interrupted by other write block numbers sent by the controller or by a command request from a node on the MODBUS network or operator control through the module's Configuration/Debug port.

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
```

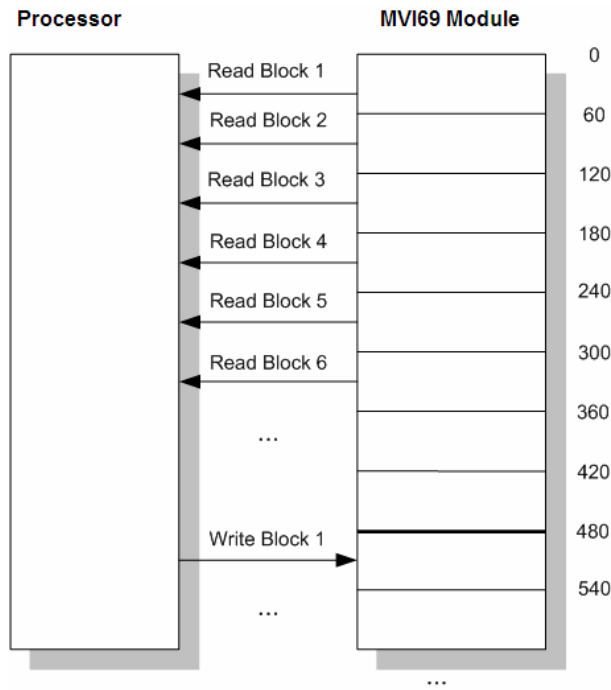
The backplane communication would be configured as follows:



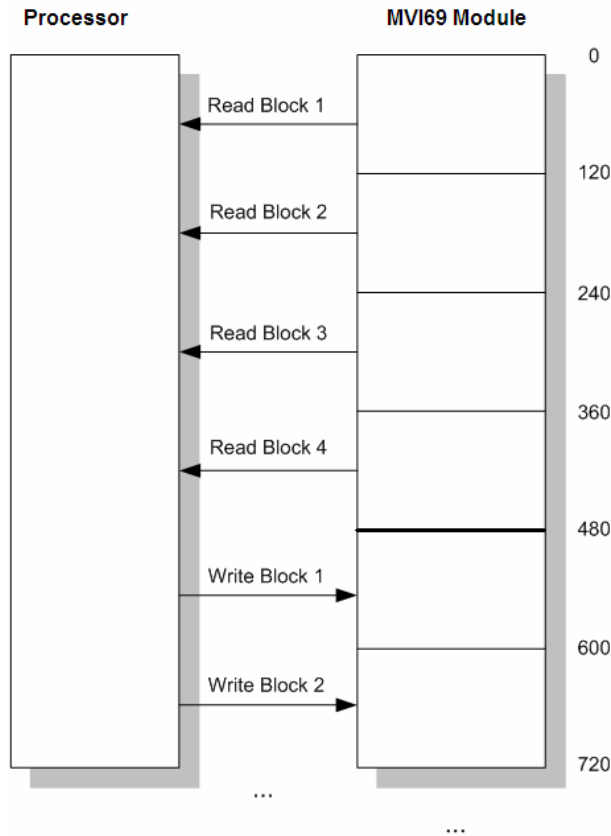
Database address 0 to 479 will be continuously transferred from the module to the processor. Database address 480 to 959 will continuously be transferred from the processor to the module.

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).

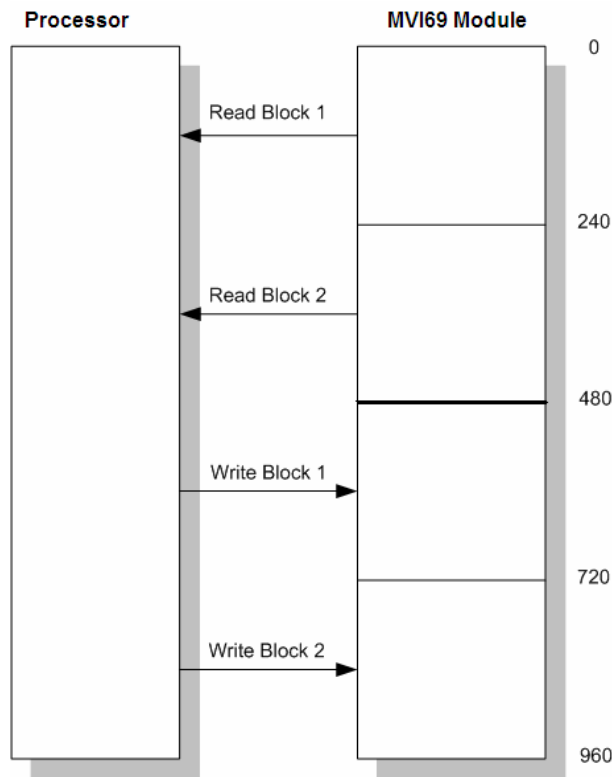
If Block Transfer Size = 60:



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:

Offset	Description	Length in words
0	Write Block ID	1
1 to n	Write Data	n

where $n = 60, 120, \text{ or } 240$ 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.

Initialize Output Data

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

Offset	Description	Length
0	4000 to 4083 for $n = 60$	1
1	4000 to 4083 for $n = 60$	1
2 to n	Spare	n

Where 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:

Offset	Description	Length
0	4000 to 4083	1
1 to n	Output Data	n

Status Data Block (Read Block ID = 0)

After the last Read Block is sent, the module builds an output image (ID = 0) to transfer the module's status information to the processor. This information can be used by the PLC program to determine the current status of the module. Ladder logic should be constructed to transfer the information in this block to a user data file. The structure of this block is shown in the following table:

Offset	Content	Description
0	Read Block ID	Block identification code -1 to indicate a status block.
1	Write Block ID	Block requested from the processor by the module.
2	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
3 - 4	Product Code	These two registers contain the product code of "MCM"
5 - 6	Product Version	These two registers contain the product version for the currently running software.
7 - 8	Operating System	These two registers contain the month and year values for the program operating system.
9 - 10	Run Number	These two registers contain the Run Number value for the currently running software.
11	Port 1 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
12	Port 1 Command List Response	This field contains the number of slave response messages received on the port.
13	Port 1 Command List Errors	This field contains the number of command errors processed on the port. These errors could be due to a bad response or command.
14	Port 1 Requests	This field contains the total number of messages sent out of the port.
15	Port 1 Responses	This field contains the total number of messages received on the port.
16	Port 1 Errors Sent	This field contains the total number of message errors sent out of the port.
17	Port 1 Errors Received	This field contains the total number of messages errors received on the port.
18	Port 2 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
19	Port 2 Command List Response	This field contains the number of slave response messages received on the port.

Offset	Content	Description
20	Port 2 Command List Errors	This field contains the number of command errors processed on the port. These errors could be due to a bad response or command.
21	Port 2 Requests	This field contains the total number of messages sent out the port.
22	Port 2 Responses	This field contains the total number of messages received on the port.
23	Port 2 Errors Sent	This field contains the total number of message errors sent out of the port.
24	Port 2 Errors Received	This field contains the total number of message errors received on the port.
25	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
26	Write Block Count	This field contains the total number of write blocks transferred from the processor to the module.
27	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
28	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
29	Command Block Count	This field contains the total number of command blocks received from the processor.
30	Error Block Count	This field contains the total number of block errors recognized by the module.
31	Port 1 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.
32	Port 1 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.
33	Port 2 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.
34	Port 2 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.

5.2.3 *Special Blocks*

Slave Status Blocks

Slave status blocks send status information of each slave device on a master port. Slaves attached to the master port can have one of the following states:

State	Description
0	The slave is inactive and not defined in the command list for the master port.
1	The slave is actively being polled or controlled by the master port and communications is successful.
2	The master port has failed to communicate with the slave device. Communications with the slave is suspended for a user defined period based on the scanning of the command list.

State	Description
3	Communications with the slave has been disabled by the ladder logic. No communication will occur with the slave until this state is cleared by the ladder logic.

Slaves are defined to the system when the module initializes the master command list. Each slave defined will be set to a state of one in this initial step. If the master port fails to communicate with a slave device (retry count expired on a command), the master will set the state of the slave to a value of 2 in the status table. This suspends communication with the slave device for a user specified scan count (**Error Delay Count** parameter in the configuration file). Each time a command in the list is scanned that has the address of a suspended slave, the delay counter value will be decremented. When the value reaches zero, the slave state will be set to one.

In order to read the slave status table, refer to the sample ladder logic. The ladder logic must send a special block to the module to request the data. Each port has a specific set of blocks to request the data as follows:

Block ID	Description
3002	Request status for slaves 0 to 59 for Port 1
3003	Request status for slaves 60 to 119 for Port 1
3004	Request status for slaves 120 to 179 for Port 1
3005	Request status for slaves 180 to 239 for Port 1
3006	Request status for slaves 240 to 255 for Port 1
3102	Request status for slaves 0 to 59 for Port 2
3103	Request status for slaves 60 to 119 for Port 2
3104	Request status for slaves 120 to 179 for Port 2
3105	Request status for slaves 180 to 239 for Port 2
3106	Request status for slaves 240 to 255 for Port 2

The format of these blocks is as shown below:

Write Block - Request Slave Status

Offset	Description	Length in words
0	3002 - 3006 or 3102 - 3106	1
1 to n	Spare	n

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

The module will recognize the request by receiving the special write block code and respond with a read block with the following format:

Read Block - Read Slave Status

Offset	Description	Length in words
0	3002 to 3006 or 3102 to 3106	1
1	Write Block ID	1
2 to 61	Slave Poll Status Data	60
62 to n	Spare (if present)	

The sample ladder logic shows how to override the value in the slave status table to disable slaves (state value of 3) by sending a special block of data from the processor to the slave. Port 1 slaves are disabled using block 3000, and Port 2 slaves are disabled using block 3100. Each block contains the slave node addresses to disable. The structure of the block is displayed below:

Write Block - Disable Slaves

Offset	Description	Length in words
0	3000 or 3100	1
1	Number of slaves in block	1
2 to 61	Slave indexes	60
62 to (n+1)	Spare	

n=120, or 240 (if configured)

The module will respond with a block with the same identification code received and indicate the number of slaves acted on with the block. The format of this response block is displayed below:

Read Block - Disable Slaves

Offset	Description	Length in words
0	3000 or 3100	1
1	Write Block ID	1
2	Number of slaves processed	1
3 to (n+1)	Spare	

n=60, 120, or 240 (if configured)

The sample ladder logic explains how to override the value in the slave status table to enable the slave (state value of 1) by sending a special block. Port 1 slaves are enabled using block 3001, and Port 2 slaves are enabled using block 3101. Each block contains the slave node addresses to enable. The format of the block is displayed below:

Write Block - Enable Slaves

Offset	Description	Length in words
0	3001 or 3101	1
1	Number of slaves in block	1
2	Slave indexes	1
3 to n	Spare	

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

The module will respond with a block with the same identification code received and indicate the number of slaves acted on with the block. The format of this response block is displayed below:

Read Block - Enable Slaves

Offset	Description	Length in words
0	3001 or 3101	1
1	Write Block ID	1

Offset	Description	Length in words
2	Number of slaves processed	1
3 to n	Spare	

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

Important: The slaves are enabled by default. Therefore, this block should only be used after Block 3000 or 3001 to re-enable the slaves.

5.2.4 Command Control Blocks

Command control blocks are special blocks used to control the module or request special data from the module. The current version of the software supports five command control blocks: event command control, command control, pass-through control blocks, warm boot and cold boot.

Event Command

Event command control blocks send MODBUS commands directly from the ladder logic to one of the master ports. The format for these blocks is displayed below:

Write Block - Event Command

Offset	Description	Length in words
0	1000 - 1255 or 2000 - 2255	1
1	Internal DB Address	1
2	Point Count	1
3	Swap Code	1
4	Function Code	1
5	Device Address	1
6 to n	Spare	

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

The block number defines the MODBUS port to be considered. Block 1000 commands are directed to Port 1, and block 2000 commands are directed to Port 2. The slave address is represented in the block number in the range of 0 to 255. The sum of these two values determines the block number. The parameters passed with the block construct the command.

- The **Internal DB Address** parameter specifies the module's database location to associate with the command.
- The **Point Count** parameter defines the number of registers for the command.
- The **Swap Code** changes the word or byte order.
- The Device Address parameter defines the MODBUS address on the target MODBUS device to consider.

- The **Function Code** parameter is one of those defined in the ProSoft MODBUS Command Set documentation.

The parameter fields in the block should be completed as required by the selected function code. Each command has its own set of parameters. When the block is received, the module will process it and place the command in the command queue. The module will respond to each event command block with a read block with the following format:

Read Block - Event Command

Offset	Description	Length in words
0	1000 - 1255 or 2000 - 2255	1
1	Write Block ID	1
2	0=Fail, 1=Success	1
3 to n	Spare	

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

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

Command Control

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

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

Write Block - Command Control

Offset	Description	Length in words
0	5001 to 5006 or 5101 to 5106	1
1	Command index	1
2	Command index	1
3	Command index	1
4	Command index	1
5	Command index	1
6	Command index	1
7 to n	Spare	

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

Blocks in the range of 5001 to 5006 are used for Port 1, and blocks in the range of 5101 to 5106 are used for Port 2. The last digit in the block code defines the number of commands to process in the block. For example, a block code of 5003 contains 3 command indexes that are to be used with Port 1. The Command index parameters in the block have a range of 0 to 99 and correspond to the master command list entries.

The module responds to a command control block with a block containing the number of commands added to the command queue for the port. The format of the block is displayed below:

Read Block - Command Control

Offset	Description	Length in words
0	5000 to 5006 or 5100 to 5106	1
1	Write Block ID	1
2	Number of commands added to command queue	1
3 to (n+1)	Spare	

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

Pass-Through Control Blocks

If one or more of the slave ports on the module are configured for the formatted pass-through mode, the module will pass blocks with identification codes of 9956, 9957, 9958 and 9959 to the processor for each received write command. Any MODBUS function 5, 6, 15 or 16 commands will be passed from the port to the processor using this block identification number. Ladder logic must handle the receipt of all MODBUS write functions to the processor and to respond as expected to commands issued by the remote MODBUS master device. The structure of the formatted pass-through control block is shown in the following tables:

Function 5

Offset	Description	Length in words
0	0	1
1	9958	1
2	Length	1
3	Data Address	1
4 to n	Data	

The ladder logic will be responsible for parsing and copying the received message and performing the proper control operation as expected by the master device. The processor must then respond to the pass-through control block with a write block with the following format.

Offset	Description	Length in words
0	9958	1

Offset	Description	Length in words
1 to n	Spare	

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

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

Function 6 and 16

Offset	Description	Length in words
0	0	1
1	9956/9957 (floating-point)	1
2	Length	1
3	Data Address	
4 to n	Data	

The ladder logic will be responsible for parsing and copying the received message and performing the proper control operation as expected by the master device. The processor must then respond to the pass-through control block with a write block with the following format.

Offset	Description	Length in words
0	9956/9957	1
1 to n	Spare	

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

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

Function 15

When the module receives a function code 15 when in pass-through mode, the module will write the data using block ID 9959 for multiple-bit data. First the bit mask clears the bits to be updated. This is accomplished by ANDing the inverted mask with the existing data. Next the new data ANDed with the mask is ORed with the existing data. This protects the other bits in the INT registers from being affected. This function can only be used if the Block Transfer Size parameter is set to 120 or 240 words.

Offset	Description	Length in words
0	0	1
1	9959	1
2	Write Block ID	1
3	Number of Words	1
4	Word Address	1
5 to 55	Data	50
56 to 105	Mask	50
106 to 180	Spare	15

The ladder logic will be responsible for parsing and copying the received message and performing the proper control operation as expected by the master

device. The processor must then respond to the pass-through control block with a write block with the following format.

Offset	Description	Length in words
0	9959	1
1 to n	Spare	

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

Warm Boot

This block is sent from the CompactLogix or MicroLogix processor to the module (output image) when the module is required to perform a warm-boot (software reset) operation. The structure of the control block is shown below:

Offset	Description	Length in words
0	9998	1
1 to n	Spare	247

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

Cold Boot

This block is sent from the CompactLogix or MicroLogix 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 structure of the control block is shown below:

Offset	Description	Length in words
0	9999	1
1 to n	Spare	247

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

5.2.5 Data Flow between MVI69-MCM Module and CompactLogix or MicroLogix Processor

The following topics describe the flow of data between the two pieces of hardware (CompactLogix or MicroLogix processor and MVI69-MCM module) and other nodes on the MODBUS network under the module's different operating modes. Each port on the module is configured to emulate a MODBUS master device or a MODBUS slave device. The operation of each port is dependent on this configuration. The sections below discuss the operation of each mode.

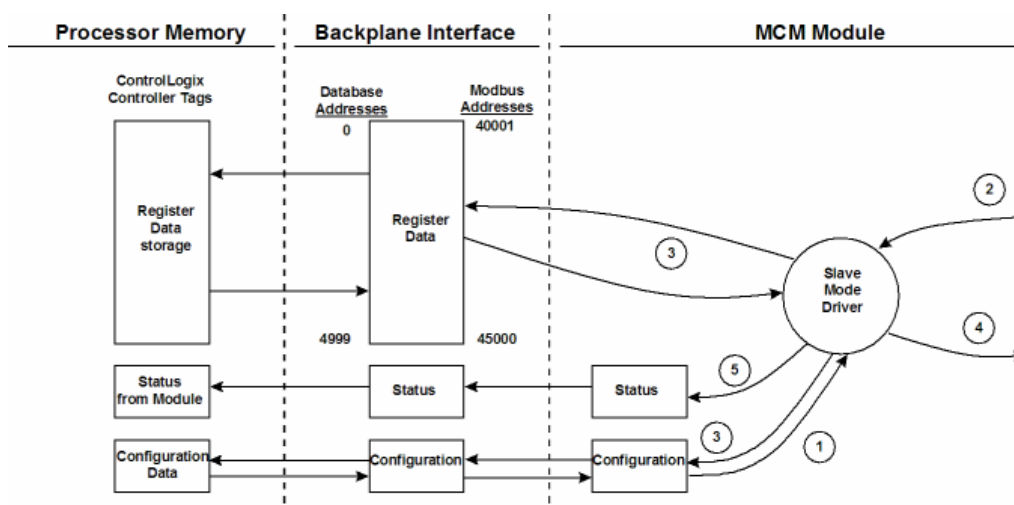
Slave Driver

The Slave Driver Mode allows the MVI69-MCM module to respond to data read and write commands issued by a master on the MODBUS network.

Step	Description
1	The MODBUS slave port driver receives the configuration information from the user defined .CFG file that is stored on the MVI69-MCM module. This information configures the serial port and define the slave node characteristics. Additionally, the configuration information contains data that can be used to offset data in the database to addresses requested in messages received from master units.
2	A Host device, such as a Modicon PLC or an MMI package, issues a read or write command to the module's node address. The port driver qualifies the message before accepting it into the module.
3	After the module accepts the command, the data is immediately transferred to or from the internal database in the module. If the command is a read command, the data is read out of the database and a response message is built. If the command is a write command, the data is written directly into the database and a response message is built.
4	After the data processing has been completed in Step 2, the response is issued to the originating master node.
5	Counters are available in the Status Block that permit the ladder logic program to determine the level of activity of the Slave Driver.

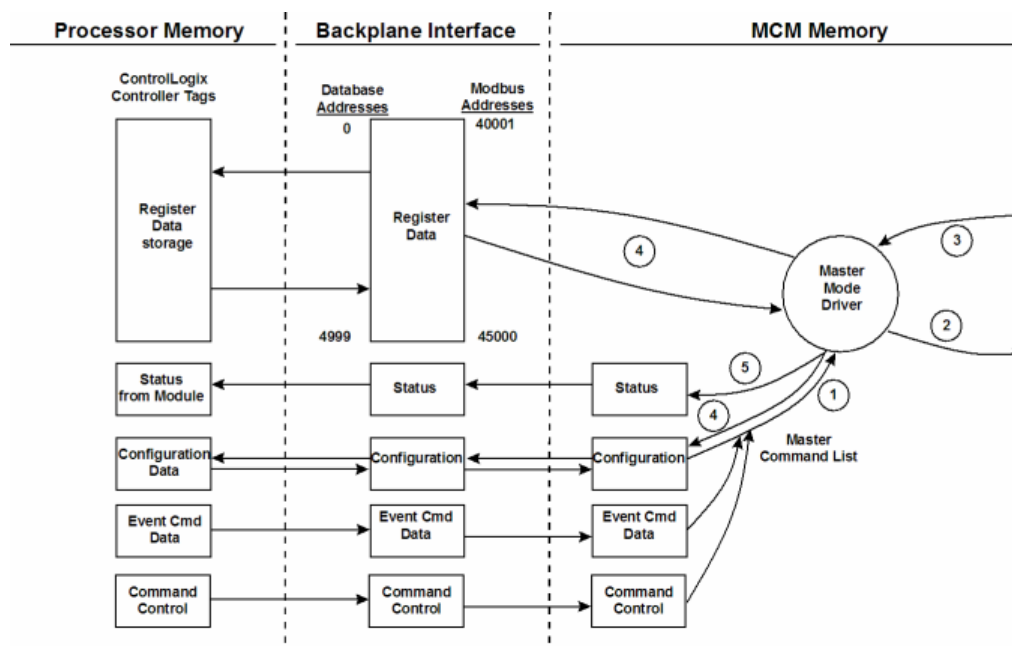
Review **Module Set Up** for a complete list of the parameters that must be defined for a slave port.

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



Master Driver Mode

In the Master mode, the MVI69-MCM module issues read or write commands to slave devices on the MODBUS network. These commands are user configured in the module via the Master Command List is received from the user defined configuration file that is stored on the MVI69-MCM module or can be issued directly from the Compact Logix processor (event command control). Command status is returned to the processor for each individual command in the command list status block. The location of this status block in the module's internal database is user defined. The following flow chart and associated table describe the flow of data into and out of the module.



Step	Description
1	The Master driver obtains configuration data from the user defined .CFG file that is stored locally on the MVI69-MCM module itself. The configuration data obtained includes port configuration, the number of commands, and the Master Command List that the MVI69-MCM module will issue, or commands can be issued directly from the Compact Logix processor (using event command control). These values are used by the Master driver to determine the type of commands to be issued to the other nodes on the MODBUS network.
2	After configuration, the Master driver begins transmitting read and/or write commands to the other nodes on the network. If writing data to another node, the data for the write command is obtained from the module's internal database to build the command.
3	Presuming successful processing by the node specified in the command, a response message is received into the Master driver for processing.
4	Data received from the node on the network is passed into the module's internal database, assuming a read command.
5	Status is returned to the CompactLogix or MicroLogix processor for each command in the Master Command List.

Refer to **Module Set Up** for a complete description of the parameters required to define the virtual MODBUS master port.

Important: You must take care when constructing each command in the list to ensure predictable operation of the module. If two commands write to the same internal database address of the module, the results will be invalid. All commands containing invalid data are ignored by the module.

Master Command List

In order to function in the Master Mode, you must define the module's Master Command List. This list contains up to 100 individual entries, with each entry containing the information required to construct a valid command. A valid command includes the following items:

- Command enable mode: (0) disabled, (1) continuous or (2) conditional
- Slave Node Address
- Command Type: Read or Write up to 125 words (2000 bits) per command
- Database Source and Destination Register Address: The addresses where data will be written or read.
- Count: The number of words to be transferred – 1 to 125 on FC 3, 4, or 16. Select the number of bits on FC 1, 2, 15.

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

Note: 125 words is the maximum count allowed by the MODBUS protocol. Some field devices may support less than the full 125 words. Check with your device manufacturer for the maximum count supported by your particular slave.

Transferring the Command Error List to the Processor

You can transfer the command error list to the processor from the module database. To place the table in the database, set the Command Error Pointer parameter to the database location desired.

To transfer this table to the processor, make sure that the Command Error table is in the database area covered by the Read Data.

Standard MODBUS Protocol Errors

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

Module Communication Error Codes

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

Command List Entry Errors

Code	Description
-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

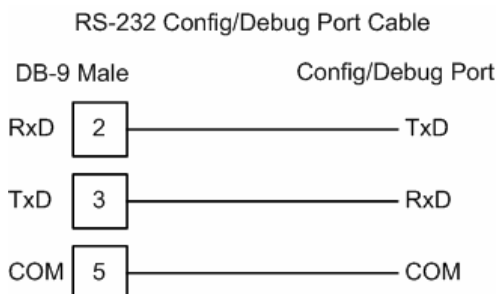
5.3 Cable Connections

The application ports on the MVI69-MCM module support RS-232, RS-422, and RS-485 interfaces. Please look at the module to ensure that the jumpers are set correctly to correspond with the type of interface you are using.

Note: When using RS-232 with radio modem applications, some radios or modems require hardware handshaking (control and monitoring of modem signal lines). Enable this in the configuration of the module by setting the UseCTS parameter to 1.

5.3.1 RS-232 Configuration/Debug Port

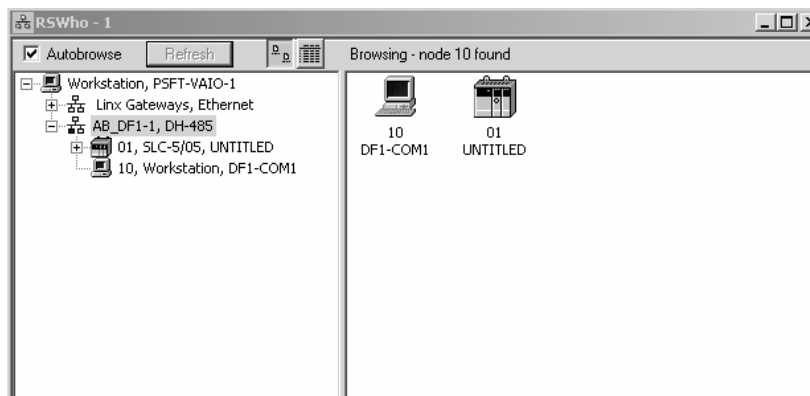
This port is physically an RJ45 connection. An RJ45 to DB-9 adapter cable is included with the module. This port permits a PC based terminal emulation program to view configuration and status data in the module and to control the module. The cable for communications on this port is shown in the following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



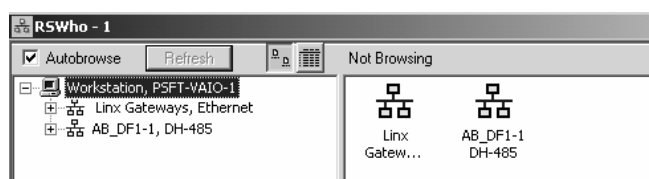
Disabling the RSLinx Driver for the Com Port on the PC

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 HyperTerminal or a similar terminal emulator, follow these steps to disable the RSLinx Driver.

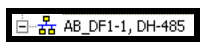
- 1 Open RSLinx and go to Communications>RSWho
- 2 Make sure that you are not actively browsing using the driver that you wish to stop. The following shows an actively browsed network:



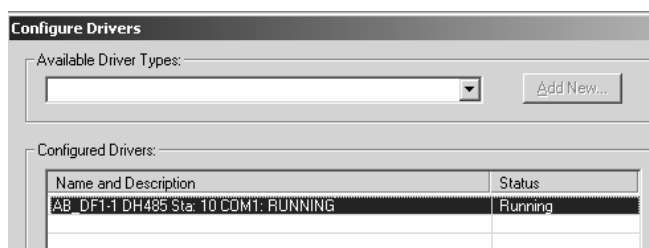
- 3 Notice how the DF1 driver is opened, and the driver is looking for node 1 (an SLC processor). If the network is being browsed, then you will not be able to stop this driver. To stop the driver your RSWho screen should look like this:



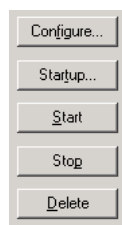
Branches are displayed or hidden by clicking on the  or the  icons.



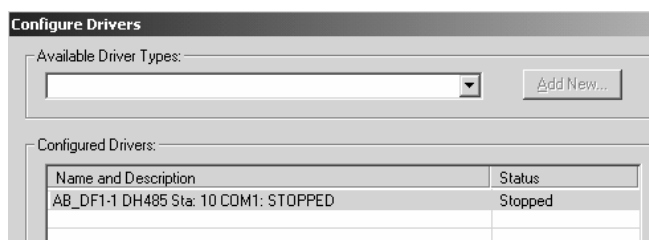
- 4 When you have verified that the driver is not being browsed, go to **Communications>Configure Drivers**
You may see something like this:



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



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

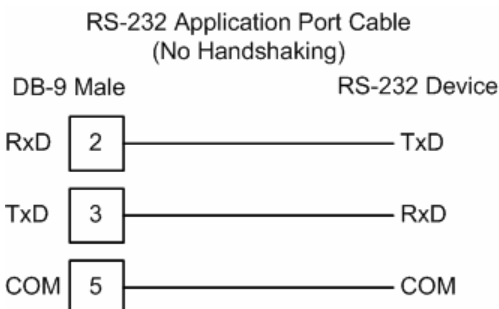


- 6 Upon seeing this, you may now use that com port to connect to the debug port of the module.

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.

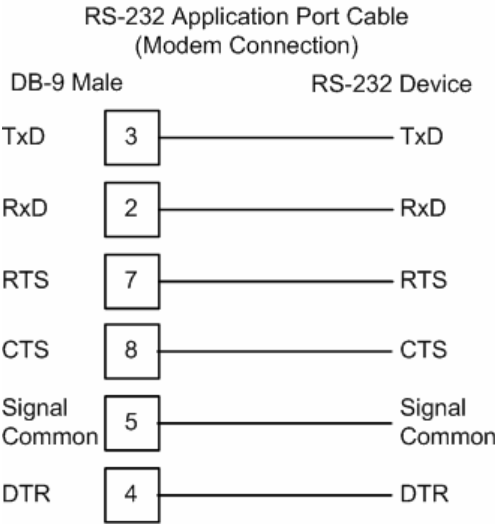
5.3.2 RS-232

When the RS-232 interface is selected, the use of hardware handshaking (control and monitoring of modem signal lines) is user definable. If no hardware handshaking will be used, the cable to connect to the port is as shown below:



RS-232 -- Modem Connection

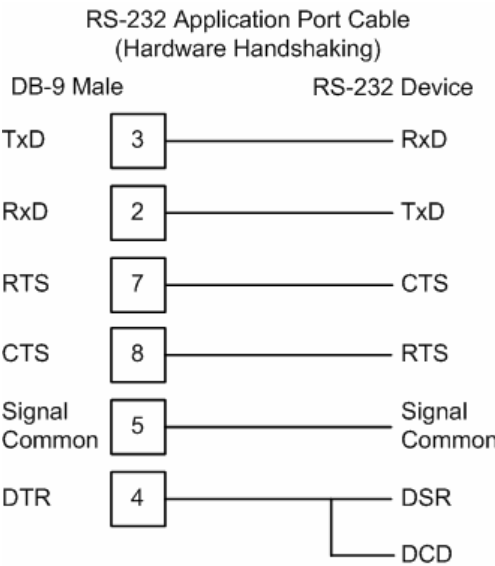
This type of connection is required between the module and a modem or other communication device.



The "Use CTS Line" parameter for the port configuration should be set to 'Y' for most modem applications.

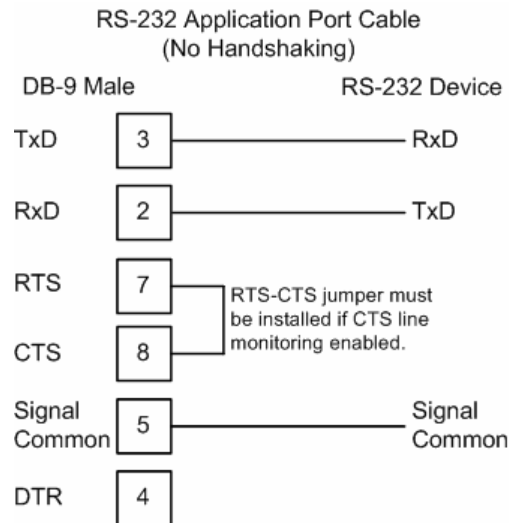
RS-232 -- Null Modem Connection (Hardware Handshaking)

This type of connection is used when the device connected to the module requires hardware handshaking (control and monitoring of modem signal lines).

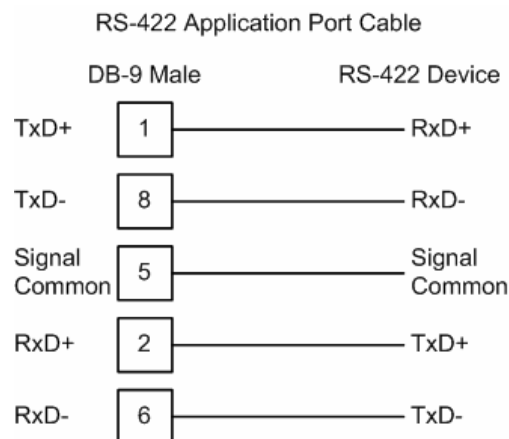


RS-232 -- Null Modem Connection (No Hardware Handshaking)

This type of connection can be used to connect the module to a computer or field device communication port.

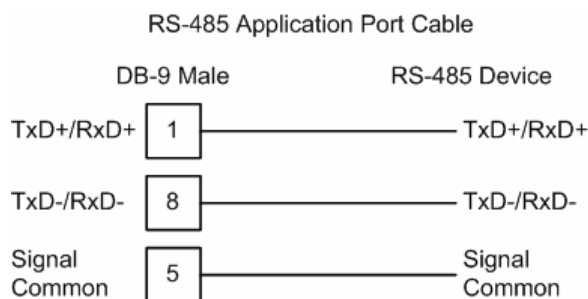


NOTE: If the port is configured with the "Use CTS Line" set to 'Y', then a jumper is required between the RTS and the CTS line on the module connection.

5.3.3**RS-422**

5.3.4 RS-485

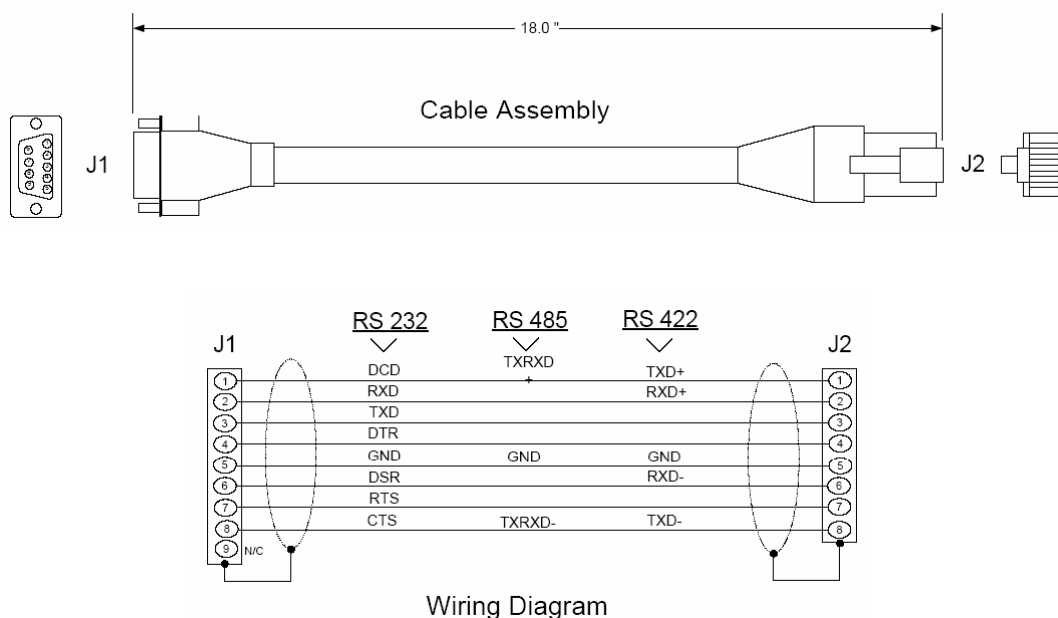
The RS-485 interface requires a single two or three wire cable. The Common connection is optional and dependent on the RS-485 network. The cable required for this interface is shown below:



RS-485 and RS-422 Tip

If communication in the RS-422/RS-485 mode does not work at first, despite all attempts, try switching termination polarities. Some manufacturers interpret +/- and A/B polarities differently.

5.3.5 DB9 to RJ45 Adaptor (Cable 14)



5.4 MCM Database Definition

This section contains a listing of the internal database of the MVI69-MCM module. This information can be used to interface other devices to the data contained in the module.

Register Range	Content	Size
0 - 4999	User Data	5000
5000 - 5009	Backplane Configuration	10
5010 - 5039	Port 1 Setup	30
5040 - 5069	Port 2 Setup	30
5070 - 5869	Port 1 Commands	800
5870 - 6669	Port 2 Commands	800
6670 - 6702	Misc. Status Data	32
6703 - 6749	Reserved	
6750 - 6759	Port 1 Status Data	10
6760 - 6769	Port 2 Status Data	10

The User Data area holds data collected from other nodes on the network (master read commands) or data received from the processor (write blocks). Additionally, this data area is used as a data source for the processor (read blocks) or other nodes on the network (write commands).

Detailed definition of the miscellaneous status data area can be found in **Misc. Status** (page 111).

Definition of the configuration data areas can be found in the data definition section of this document and in **Configuration Data Definition** (page 105).

5.5 Status Data Definition

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

Status Data Block Structure

Offset	Content	Description
6670	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
6671 to 6672	Product Code	These two registers contain the product code of "MCM"
6673 to 6674	Product Version	These two registers contain the product version for the current running software.
6675 to 6676	Operating System	These two registers contain the month and year values for the program operating system.
6677 to 6678	Run Number	These two registers contain the run number value for the currently running software.
6679	Port 1 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
6680	Port 1 Command List Response	This field contains the number of slave response messages received on the port.

Offset	Content	Description
6681	Port 1 Command List Errors	This field contains the number of command errors processed on the port. These errors could be due to a bad response or command.
6682	Port 1 Requests	This field contains the total number of messages sent out of the port.
6683	Port 1 Responses	This field contains the total number of messages received on the port.
6684	Port 1 Errors Sent	This field contains the total number of message errors sent out of the port.
6685	Port 1 Errors Received	This field contains the total number of message errors received on the port.
6686	Port 2 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
6687	Port 2 Command List Response	This field contains the number of slave response messages received on the port.
6688	Port 2 Command List Errors	This field contains the number of command errors processed on the port. These errors could be due to a bad response or command.
6689	Port 2 Requests	This field contains the total number of messages sent out the port.
6690	Port 2 Responses	This field contains the total number of messages received on the port.
6691	Port 2 Errors Sent	This field contains the total number of message errors sent out of the port.
6692	Port 2 Errors Received	This field contains the total number of message errors received on the port.
6693	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
6694	Write Block Count	This field contains the total number of write blocks transferred from the processor to the module.
6695	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
6696	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
6697	Command Block Count	This field contains the total number of command blocks received from the processor.
6698	Error Block Count	This field contains the total number of block errors recognized by the module.
6699	Port 1 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.
6700	Port 1 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.
6701	Port 2 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.
6702	Port 2 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.

5.6 Configuration Data Definition

This section contains listings of the MVI69-MCM module's database that are related to the module's configuration. This data is defined in the MVI69MCM.CFG file which initializes the module during bootstrap. When the module is configured as a slave, this data is available to the Modbus master and can be read at the offset addresses shown.

Additionally, this section contains the miscellaneous status data and command control database layout.

Group	Register	Content	Description
Backplane Setup	5000	Write Start Reg	This parameter specifies the starting register in the module where the data transferred from the processor will be placed. Valid range for this parameter is 0 to 4999.
	5001	Write Reg Count	This parameter specifies the number of registers to transfer from the processor to the module. Valid entry for this parameter is 0 to 5000.
	5002	Read Start Reg	This parameter specifies the starting register in the module where data will be transferred from the module to the processor. Valid range for this parameter is 0 to 4999.
	5003	Read Reg Count	This parameter specifies the number of registers to be transferred from the module to the processor. Valid entry for this parameter is 0 to 5000.
	5004	Backplane Fail	This parameter specifies the number of successive transfer errors that must occur before the communication ports are shut down. If the parameter is set to 0, the communication ports 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.
	5005	Error Status Pointer	This parameter specifies the register location in the module's database where module status data will be stored. If a value less than 0 is entered, the data will not be stored in the database. If the value specified is in the range of 0 to 4940, the data will be placed in the user data area.
	5006	Initialize Output Data	This parameter determines if the output data for the module should be initialized with values from the processor. If the parameter is set to No, the output data will be initialized to 0. If the parameter is set to Yes, 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.
	5007	BT Size	This parameter defines the size of the block transfer data area for the application. Valid values are 60, 120 and 240.
	5008	Spare	
	5009	Spare	

5.6.1 Port 1 Setup

Register	Content	Description
5010	Enable	This parameter defines if this MODBUS port will be used. If the parameter is set to 0, the port is disabled. A value of 1 enables the port.
5011	Type	This parameter specifies if the port will emulate a MODBUS master device (0), a MODBUS slave device without pass-through (1), or a MODBUS slave device with unformatted pass-through (2), or a MODBUS slave device with formatted pass-through and data swapping (3).
5012	Float Flag	This flag specifies if the floating-point data access functionality is to be implemented. If the float flag is set to 1, MODBUS functions 3, 6, and 16 will interpret floating-point values for registers as specified by the two following parameters.
5013	Float Start	This parameter defines the first register of floating-point data. All requests with register values greater than or equal to this value will be considered floating-point data requests. This parameter is only used if the Float Flag is enabled.
5014	Float Offset	This parameter defines the start register for floating-point data in the internal database. This parameter is only used if the Float Flag is enabled.
5015	Protocol	This parameter specifies the MODBUS protocol to be used on the port. Valid protocols are: 0 = MODBUS RTU and 1 = MODBUS ASCII.
5016	Baud Rate	This is the baud rate to be used on the port. Enter the baud rate as a value. For example, to select 19K baud, enter 19200. Valid entries are 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 576, and 115.
5017	Parity	This is the parity code to be used for the port. Values are None, Odd, Even.
5018	Data Bits	This parameter sets the number of data bits for each word used by the protocol. Valid entries for this field are 5 through 8.
5019	Stop Bits	This parameter sets the number of stop bits to be used with each data value sent. Valid entries are 1 and 2.
5020	RTS On	This parameter sets the number of milliseconds to delay after RTS is asserted before the data will be transmitted. Valid values are in the range of 0 to 65535 milliseconds.
5021	RTS Off	This parameter sets the number of milliseconds to delay after the last byte of data is sent before the RTS modem signal will be set low. Valid values are in the range of 0 to 65535.
5022	Minimum Response Time	This parameter specifies the minimum number of milliseconds to delay before responding to a request message. This pre-send delay is applied before the RTS on time. This may be required when communicating with slow devices.

Register	Content	Description
5023	Use CTS Line	This parameter specifies if the CTS modem control line is to be used. If the parameter is set to 0, the CTS line will not be monitored. If the parameter is set to 1, the CTS line will be monitored and must be high before the module will send data. This parameter is normally only required when half-duplex modems are used for communication (2-wire).
5024	Slave ID	This parameter defines the virtual MODBUS slave address for the internal database. All requests received by the port with this address are processed by the module. Verify that each device has a unique address on a network. Valid range for this parameter is 1 to 255 (247 on some networks).
5025	Bit in Offset	This parameter specifies the offset address in the internal MODBUS database that is to be used with network requests for MODBUS Function 2 commands. For example, if the value is set to 150, an address request of 0 will return the value at register 150 in the database.
5026	Word in Offset	This parameter specifies the offset address in the internal MODBUS database that is to be used with network request for MODBUS function 4 commands. For example, if the value is set to 150, an address request of 0 will return the value at register 150 in the database.
5027	Out in Offset	This parameter specifies the offset address in the internal MODBUS database that is to be used with network requests for MODBUS function 1, 5, or 15 commands. For example, if the value is set to 100, an address request of 0 will correspond to register 100 in the database.
5028	Holding Reg Offset	This parameter specifies the offset address in the internal MODBUS database that is to be used with network requests for MODBUS function 3, 6, or 16 commands. For example, if a value of 50 is entered, a request for address 0 will correspond to the register 50 in the database.
5029	Command Count	This parameter specifies the number of commands to be processed by the MODBUS master port.
5030	Minimum Command Delay	This parameter specifies the number of milliseconds to wait between issuing each command. This delay value is not applied to retries.
5031	Command Error Pointer	This parameter sets the address in the internal MODBUS database where the command error will be placed. If the value is set to -1, the data will not be transferred to the database. The valid range of values for this parameter is -1 to 4999.

Register	Content	Description
5032	Response Timeout	This parameter represents the message response timeout period in 1-millisecond increments. This is the time that a port configured as a master will wait before re-transmitting a command if no response is received from the addressed slave. The value is set depending upon the communication network used and the expected response time of the slowest device on the network.
5033	Retry Count	This parameter specifies the number of times a command will be retried if it fails. If the master port does not receive a response after the last retry, the slave devices communication will be suspended on the port for Error Delay Counter scans.
5034	Error Delay Counter	This parameter specifies the number of polls to skip on the slave before trying to re-establish communications. After the slave fails to respond, the master will skip commands to be sent to the slave the number of times entered in this parameter.
5035	Spare	
5036	Spare	
5037	Spare	
5038	Spare	
5039	Spare	

5.6.2 Port 2 Setup

Register	Content	Description
5040	Enable	This parameter defines if this MODBUS port will be used. If the parameter is set to 0, the port is disabled. A value of 1 enables the port.
5041	Type	This parameter specifies if the port will emulate a MODBUS master device (0), a MODBUS slave device without pass-through (1), or a MODBUS slave device with unformatted pass-through (2), or a MODBUS slave device with formatted pass-through and data swapping (3).
5042	Float Flag	This flag specifies if the floating-point data access functionality is to be implemented. If the float flag is set to 1, MODBUS functions 3, 6, and 16 will interpret floating-point values for registers as specified by the two following parameters.
5043	Float Start	This parameter defines the first register of floating-point data. All requests with register values greater than or equal to this value will be considered floating-point data requests. This parameter is only used if the Float Flag is enabled.
5044	Float Offset	This parameter defines the start register for floating-point data in the internal database. This parameter is only used if the Float Flag is enabled.
5045	Protocol	This parameter specifies the MODBUS protocol to be used on the port. Valid protocols are: 0 = MODBUS RTU and 1 = MODBUS ASCII.

Register	Content	Description
5046	Baud Rate	This is the baud rate to be used on the port. Enter the baud rate as a value. For example, to select 19K baud, enter 19200. Valid entries are 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 384 (for 38400bps), 576 (for 57600bps), and 115 (for 115,200bps).
5047	Parity	This is the parity code to be used for the port. Values are None, Odd, Even.
5048	Data Bits	This parameter sets the number of data bits for each word used by the protocol. Valid entries for this field are 5 through 8.
5049	Stop Bits	This parameter sets the number of stop bits to be used with each data value sent. Valid entries are 1 and 2.
5050	RTS On	This parameter sets the number of milliseconds to delay after RTS is asserted before the data will be transmitted. Valid values are in the range of 0 to 65535 milliseconds.
5051	RTS Off	This parameter sets the number of milliseconds to delay after the last byte of data is sent before the RTS modem signal will be set low. Valid values are in the range of 0 to 65535.
5052	Minimum Response Time	This parameter specifies the minimum number of milliseconds to delay before responding to a request message. This pre-send delay is applied before the RTS on time. This may be required when communicating with slow devices.
5053	Use CTS Line	This parameter specifies if the CTS modem control line is to be used. If the parameter is set to 0, the CTS line will not be monitored. If the parameter is set to 1, the CTS line will be monitored and must be high before the module will send data. This parameter is normally only required when half-duplex modems are used for communication (2-wire).
5054	Slave ID	This parameter defines the virtual MODBUS slave address for the internal database. All requests received by the port with this address are processed by the module. Verify that each device has a unique address on a network. Valid range for this parameter is 1 to 255 (247 on some networks).
5055	Bit in Offset	This parameter specifies the offset address in the internal MODBUS database that is to be used with network requests for MODBUS Function 2 commands. For example, if the value is set to 150, an address request of 0 will return the value at register 150 in the database.
5056	Word in Offset	This parameter specifies the offset address in the internal MODBUS database that is to be used with network request for MODBUS function 4 commands. For example, if the value is set to 150, an address request of 0 will return the value at register 150 in the database.

Register	Content	Description
5057	Out in Offset	This parameter specifies the offset address in the internal MODBUS database that is to be used with network requests for MODBUS function 1, 5, or 15 commands. For example, if the value is set to 100, an address request of 0 will correspond to register 100 in the database.
5058	Holding Reg Offset	This parameter specifies the offset address in the internal MODBUS database that is to be used with network requests for MODBUS function 3, 6, or 16 commands. For example, if a value of 50 is entered, a request for address 0 will correspond to the register 50 in the database.
5059	Command Count	This parameter specifies the number of commands to be processed by the MODBUS master port.
5060	Minimum Command Delay	This parameter specifies the number of milliseconds to wait between issuing each command. This delay value is not applied to retries.
5061	Command Error Pointer	This parameter sets the address in the internal MODBUS database where the command error will be placed. If the value is set to -1, the data will not be transferred to the database. The valid range of values for this parameter is -1 to 4999.
5062	Response Timeout	This parameter represents the message response timeout period in 1-millisecond increments. This is the time that a port configured as a master will wait before re-transmitting a command if no response is received from the addressed slave. The value is set depending upon the communication network used and the expected response time of the slowest device on the network.
5063	Retry Count	This parameter specifies the number of times a command will be retried if it fails. If the master port does not receive a response after the last retry, the slave devices communication will be suspended on the port for Error Delay Counter scans.
5064	Error Delay Counter	This parameter specifies the number of polls to skip on the slave before trying to re-establish communications. After the slave fails to respond, the master will skip commands to be sent to the slave the number of times entered in this parameter.
5065	Spare	
5066	Spare	
5067	Spare	
5068	Spare	
5069	Spare	

5.6.3 Port 1 Commands

Register	Content	Description
5070 - 5777	Command #1	This set of registers contains the parameters for the first command in the master command list. The structure of this data area is as described in the data object section of the documentation.
5078 - 5085	Command #2	Command #2 data set
-	-	-
5852 - 5859	Command #100	Command #100 data set

5.6.4 Port 2 Commands

Register	Content	Description
5870 - 5877	Command #1	This set of registers contains the parameters for the first command in the master command list. The structure of this data area is as described in the data object section of the documentation.
5878 - 5885	Command #2	Command #2 data set
-	-	-
6662 - 6669	Command #100	Command #100 data set

5.6.5 Misc. Status

Register	Content	Description
6670	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
6671 - 6672	Product Code	These two registers contain the product code of "MCM".
6673 - 6674	Product Version	These two registers contain the product version for the current running software.
6675 - 6676	Operating System	These two registers contain the month and year values for the program operating system.
6677 - 6678	Run Number	These two registers contain the run number value for the currently running software.
6679	Port 1 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
6680	Port 1 Command List Response	This field contains the number of slave response messages received on the port.
6681	Port 1 Command List Errors	This field contains the number of command errors processed on the port. These errors could be due to a bad response or command.
6682	Port 1 Requests	This field contains the total number of messages sent out of the port.
6683	Port 1 Responses	This field contains the total number of messages received on the port.
6684	Port 1 Errors Sent	This field contains the total number of message errors sent out of the port.

Register	Content	Description
6685	Port 1 Errors Received	This field contains the total number of message errors received on the port.
6686	Port 2 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
6687	Port 2 Command List Response	This field contains the number of slave response messages received on the port.
6688	Port 2 Command List Errors	This field contains the number of command errors processed on the port. These errors could be due to a bad response or command.
6689	Port 2 Requests	This field contains the total number of messages sent out the port.
6690	Port 2 Responses	This field contains the total number of messages received on the port.
6691	Port 2 Errors Sent	This field contains the total number of message errors sent out the port.
6692	Port 2 Errors Received	This field contains the total number of message errors received on the port.
6693	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
6694	Write Block Count	This field contains the total number of write blocks transferred from the module to the processor.
6695	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
6696	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
6697	Command Block Count	This field contains the total number of command blocks received from the processor.
6698	Error Block Count	This field contains the total number of block errors recognized by the module.
6699	Port 1 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.
6700	Port 1 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with the error.
6701	Port 2 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.
6702	Port 2 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.
6703	Spare	
-	-	-
6749	Spare	
6750	Port 1 Use Guard Band	Use packet gap timeout for messages (Yes or No). Use only in multi-drop applications.

Register	Content	Description
6751	Port 1 Guard Band Time	A value of 0 uses the default baud rate or you can set a timeout value in milliseconds.
6752	Port 1 Fcn 99 Offset	Internal DB offset to Function 99 counter.
6753	Spare	
-		
6759	Spare	
6760	Port 2 Use Guard Band	Use packet gap timeout for messages (Yes or No). Use only in multi-drop applications.
6761	Port 2 Guard Band Time	A value of 0 uses the default baud rate or you can set a timeout value in milliseconds.
6762	Port 2 Fcn 99 Offset	Internal DB offset to Function 99 counter.
6763	Spare	
-	-	-
6799	Spare	

5.6.6 *Command Control*

Register	Content	Description
6800	Command Code	Enter one of the valid control command codes in this register to control the module (9997, 9998, or 9999).
6801	Command Data	Not Used
-	-	-
6999	Command Data	Not Used

Support, Service & Warranty

ProSoft Technology, Inc. survives on its ability to provide meaningful support to its customers. Should any questions or problems arise, please feel free to contact us at:

Internet	Web Site: http://www.prosoft-technology.com/support
	E-mail address: support@prosoft-technology.com
Phone	+1 (661) 716-5100
	+1 (661) 716-5101 (Fax)
Postal Mail	ProSoft Technology, Inc.
	1675 Chester Avenue, Fourth Floor
	Bakersfield, CA 93301

Before calling for support, please prepare yourself for the call. In order to provide the best and quickest support possible, we will most likely ask for the following information:

- 1 Product Version Number
- 2 System architecture
- 3 Module configuration and contents of configuration file, if the module requires one.
- 4 Module Operation
 - Configuration/Debug status information
 - LED patterns
- 5 Information about the processor and user data files as viewed through the processor configuration software and LED patterns on the processor
- 6 Details about the serial devices interfaced

An after-hours answering system allows pager access to one of our qualified technical and/or application support engineers at any time to answer the questions that are important to you.

Module Service and Repair

The MVI69-MCM device is an electronic product, designed and manufactured to function under somewhat adverse conditions. As with any product, through age, misapplication, or any one of many possible problems the device may require repair.

When purchased from ProSoft Technology, Inc., the device has a 1 year parts and labor warranty (3 years for RadioLinx) according to the limits specified in the warranty. Replacement and/or returns should be directed to the distributor from whom the product was purchased. If you must return the device for repair, obtain an RMA (Returned Material Authorization) number from ProSoft Technology, Inc. Please call the factory for this number, and print the number prominently on the outside of the shipping carton used to return the device.

General Warranty Policy – Terms and Conditions

ProSoft Technology, Inc. (hereinafter referred to as ProSoft) warrants that the Product shall conform to and perform in accordance with published technical specifications and the accompanying written materials, and shall be free of defects in materials and workmanship, for the period of time herein indicated, such warranty period commencing upon receipt of the Product. Limited warranty service may be obtained by delivering the Product to ProSoft in accordance with our product return procedures and providing proof of purchase and receipt date. Customer agrees to insure the Product or assume the risk of loss or damage in transit, to prepay shipping charges to ProSoft, and to use the original shipping container or equivalent. Contact ProSoft Customer Service for more information.

This warranty is limited to the repair and/or replacement, at ProSoft's election, of defective or non-conforming Product, and ProSoft shall not be responsible for the failure of the Product to perform specified functions, or any other non-conformance caused by or attributable to: (a) any misuse, misapplication, accidental damage, abnormal or unusually heavy use, neglect, abuse, alteration (b) failure of Customer to adhere to ProSoft's specifications or instructions, (c) any associated or complementary equipment, software, or user-created programming including, but not limited to, programs developed with any IEC1131-3 programming languages, 'C' for example, and not furnished by ProSoft, (d) improper installation, unauthorized repair or modification (e) improper testing, or causes external to the product such as, but not limited to, excessive heat or humidity, power failure, power surges or natural disaster, compatibility with other hardware and software products introduced after the time of purchase, or products or accessories not manufactured by ProSoft; all of which components, software and products are provided as-is. In no event will ProSoft be held liable for any direct or indirect, incidental consequential damage, loss of data, or other malady arising from the purchase or use of ProSoft products.

ProSoft's software or electronic products are designed and manufactured to function under adverse environmental conditions as described in the hardware specifications for this product. As with any product, however, through age, misapplication, or any one of many possible problems, the device may require repair.

ProSoft warrants its products to be free from defects in material and workmanship and shall conform to and perform in accordance with published technical specifications and the accompanying written materials for up to one year (12 months) from the date of original purchase (3 years for RadioLinx products) from ProSoft. If you need to return the device for repair, obtain an RMA (Returned Material Authorization) number from ProSoft Technology, Inc. in accordance with the RMA instructions below. Please call the factory for this number, and print the number prominently on the outside of the shipping carton used to return the device.

If the product is received within the warranty period ProSoft will repair or replace the defective product at our option and cost.

Warranty Procedure: Upon return of the hardware product ProSoft will, at its option, repair or replace the product at no additional charge, freight prepaid, except as set forth below. Repair parts and replacement product will be furnished on an exchange basis and will be either reconditioned or new. All replaced product and parts become the property of ProSoft. If ProSoft determines that the Product is not under warranty, it will, at the Customer's option, repair the Product using then current ProSoft standard rates for parts and labor, and return the product freight collect.

Limitation of Liability

EXCEPT AS EXPRESSLY PROVIDED HEREIN, PROSOFT MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, WITH RESPECT TO ANY EQUIPMENT, PARTS OR SERVICES PROVIDED PURSUANT TO THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER PROSOFT OR ITS DEALER SHALL BE LIABLE FOR ANY OTHER DAMAGES, INCLUDING BUT NOT LIMITED TO DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION IN CONTRACT OR TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), SUCH AS, BUT NOT LIMITED TO, LOSS OF ANTICIPATED PROFITS OR BENEFITS RESULTING FROM, OR ARISING OUT OF, OR IN CONNECTION WITH THE USE OR FURNISHING OF EQUIPMENT, PARTS OR SERVICES HEREUNDER OR THE PERFORMANCE, USE OR INABILITY TO USE THE SAME, EVEN IF ProSoft OR ITS DEALER'S TOTAL LIABILITY EXCEED THE PRICE PAID FOR THE PRODUCT.

Where directed by State Law, some of the above exclusions or limitations may not be applicable in some states. This warranty provides specific legal rights; other rights that vary from state to state may also exist. This warranty shall not be applicable to the extent that any provisions of this warranty are prohibited by any Federal, State or Municipal Law that cannot be preempted. Contact ProSoft Customer Service at +1 (661) 716-5100 for more information.

RMA Procedures

In the event that repairs are required for any reason, contact ProSoft Technical Support at +1 661.716.5100. A Technical Support Engineer will ask you to perform several tests in an attempt to diagnose the problem. Simply calling and asking for a RMA without following our diagnostic instructions or suggestions will lead to the return request being denied. If, after these tests are completed, the module is found to be defective, we will provide the necessary RMA number with instructions on returning the module for repair.

Index

A

About the MODBUS Protocol • 78
After you Complete the Module Setup • 26
Analyzing Data for Port 1 • 68
Analyzing Data for Port 2 • 68

B

Backplane Data Transfer • 79
Backplane Menu • 63

C

Cable Connections • 97
Clearing a Fault Condition • 73
Clearing Diagnostic Data • 59
Cold Boot • 93
Command Control • 90, 113
Command Control Blocks • 89
Command Entry Formats • 43
Command List Entry Errors • 97
Command List Overview • 36
Commands Supported by the Module • 42
Configuration Data Definition • 103, 105
Configuring RSLinx • 15
Configuring the Floating Point Data Transfer • 38
Connect your PC to the Module • 17
Connect your PC to the Processor • 13

D

Data Analyzer • 67
Data Analyzer Tips • 70
Data Flow between MVI69-MCM Module and CompactLogix or MicroLogix Processor • 93
Database View Menu • 61
DB9 to RJ45 Adaptor (Cable 14) • 102
Diagnostics and Troubleshooting • 5, 29, 31, 55
Disabling the RSLinx Driver for the Com Port on the PC • 57, 97
Displaying the Current Page of Registers Again • 61
Displaying Timing Marks in the Data Analyzer • 68
Download the Sample Program to the Processor • 14

E

ENRON Floating Point Support • 37
Event Command • 89
Exiting the Program • 60

F

Features and Benefits • 75
Floating Point Support • 36
Function 15 • 92
Function 5 • 91
Function 6 and 16 • 92
Functional Overview • 5, 78
Functional Specifications • 76

G

General Concepts • 78
General Specifications • 75
Guide to the MVI69-MCM User Manual • 5

H

Hardware Specifications • 76

I

If Block Transfer Size = 120: • 83
If Block Transfer Size = 240: • 84
If Block Transfer Size = 60: • 83
Initialize Output Data • 84
Installing and Configuring the Module • 19
Installing and Configuring the Module with a CompactLogix Processor • 21
Installing and Configuring the Module with a MicroLogix Processor • 29
Installing the Module • 10

K

Keystrokes • 56

L

Ladder Logic • 53
LED Status Indicators • 72

M

Main Logic Loop • 79
Main Menu • 58
Master Command Error List Menu • 65
Master Command List • 96
Master Driver Mode • 94
MCM Database Definition • 103
Misc. Status • 103, 111
MODBUS Command Configuration • 36

MODBUS Message Data • 35
Module Communication Error Codes • 96
Module Data Object (MCMModuleDef) • 32
Module Power Up • 78
Moving Back Through 5 Pages of Registers • 62

N

Navigation • 56
Normal Data Transfer • 81

O

Opening the Backplane Menu • 59
Opening the Command List Menu • 65
Opening the Data Analyzer Menu • 67
Opening the Database Menu • 59
Opening the Protocol Serial Menu • 59
Opening the Serial Port Menu • 65

P

Package Contents • 8
Pass-Through Control Blocks • 91
Please Read This Notice • 2
Port 1 Commands • 111
Port 1 Setup • 106
Port 2 Commands • 111
Port 2 Setup • 108
Product Specifications • 5, 75
Protocol Serial MCM Menu • 64

R

Read Block • 81
Read Block - Command Control • 91
Read Block - Disable Slaves • 88
Read Block - Enable Slaves • 88
Read Block - Event Command • 90
Read Block - Read Slave Status • 87
Reading Status Data from the module • 5, 55
Receiving the Configuration File • 59
Redisplaying the Current Page • 65
Redisplaying the Menu • 59, 63, 64, 66
Reference • 5, 36, 75
Removing Timing Marks in the Data Analyzer • 68
Required Hardware • 46, 56
Required Software • 46, 57
Returning to the Main Menu • 62, 63, 64, 66, 70
RS-232 • 99
RS-232 -- Modem Connection • 100
RS-232 -- Null Modem Connection (Hardware Handshaking) • 100
RS-232 -- Null Modem Connection (No Hardware Handshaking) • 101
RS-232 Configuration/Debug Port • 97

RS-422 • 101
RS-485 • 102
RS-485 and RS-422 Tip • 102

S

Sending the Configuration File • 59
Serial Port Menu • 66
Setting Jumpers • 9
Skipping 500 Registers of Data • 62
Slave Driver • 93
Slave Polling Control and Status • 35
Slave Status Blocks • 86
Special Blocks • 86
Standard MODBUS Protocol Errors • 96
Start Here • 5, 7
Starting the Data Analyzer • 69
Status Data Block (Read Block ID = 0) • 85
Status Data Definition • 103
Status Object (MCM1Status) • 34
Stopping the Data Analyzer • 69
Support, Service & Warranty • 5, 115
System Requirements • 7

T

The Configuration/Debug Menu • 55
Transferring the Command Error List to the Processor • 96
Transferring the Configuration File to the Module • 50
Transferring the Configuration File to Your PC • 47
Troubleshooting • 73

U

Uploading and Downloading the Configuration File • 45, 59
User Data Objects • 34
Using the Configuration/Debug Port • 57

V

Viewing Backplane Diagnostic Information • 63
Viewing Configuration Information • 63, 65
Viewing Data in ASCII (Text) Format • 62, 69
Viewing Data in Decimal Format • 62
Viewing Data in Floating Point Format • 62
Viewing Data in Hexadecimal Format • 62, 69
Viewing Error and Status Data • 65
Viewing Register Pages • 61
Viewing the Next 100 Registers of Data • 62
Viewing the Next 20 Commands • 65
Viewing the Next Page of Commands • 66
Viewing the Previous 100 Registers of Data • 62

Viewing the Previous 20 Commands • 65
Viewing the Previous Page of Commands •
65
Viewing Version Information • 59, 63, 64, 66

W

Warm Boot • 93
Warm Booting the Module • 60
Write Block • 84
Write Block - Command Control • 90
Write Block - Disable Slaves • 88
Write Block - Enable Slaves • 88
Write Block - Event Command • 89
Write Block - Request Slave Status • 87

Y

Your Feedback Please • 2