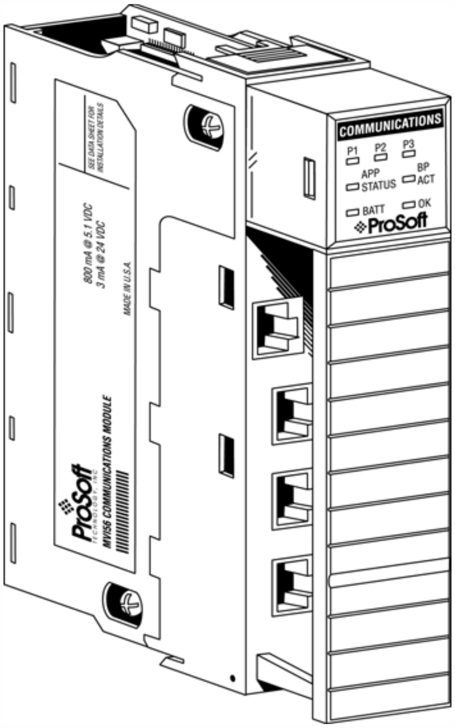
inRAx



MVI56-MCM

ControlLogix Platform

Modbus Communication Module

User Manual

January 28, 2005

ProSoft
TECHNOLOGY

Please Read This Notice

Successful application of this module requires a reasonable working knowledge of the Allen Bradley ControlLogix Platform Modbus Communication Module hardware 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 Allen Bradley documentation on the operation of the Allen Bradley hardware.

Under no conditions will ProSoft Technology, Inc. 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, Inc. is prohibited.

Information in this manual is subject to change without notice and does not represent a commitment on the part of ProSoft Technology, Inc. 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, Inc.
1675 Chester Avenue, Second Floor
Bakersfield, CA 93301
(661) 716-5100
(661) 716-5101 (Fax)
http://www.prosoft-technology.com

MVI56-MCM User Manual
January 28, 2005

Contents

1 Introduction

In This Chapter

The MVI56-MCM ("Modbus Communication Module") product allows Allen-Bradley ControlLogix I/O compatible processors to easily interface with other Modbus protocol compatible devices. Compatible devices include not only Modicon PLC's (which all support the Modbus protocol) but also a wide assortment of end devices.

The MVI56-MCM module acts as a gateway between the Modbus network and the Allen-Bradley backplane. The data transfer from the ControlLogix processor is asynchronous from the actions on the Modbus network. A 5000-word register space in the module is used to exchange data between the processor and the Modbus network.

1.1 General Concepts

The following discussion covers several concepts that are key to understanding the operation of the MVI56-MCM module.

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

1 Initialize hardware components

2 Initialize ControlLogix backplane driver

  o Test and Clear all RAM

  o Initialize the serial communication ports

  o Wait for Module Configuration from ControlLogix processor

3 Initialize Module Register space

4 Enable Slave Driver on selected ports

5 Enable Master Driver on selected ports

Once the module has received the Module Configuration Block from the processor, the module will begin communicating with other nodes on the network, depending on the configuration.

1.2 Setting Up the Module

Once your module is installed, you can begin the process of modifying your module configuration and ladder logic. Before beginning this process, you should understand the architecture, which is presented in the next section. The remaining sections explain how to make modifications to the existing .cfg file and sample ladder logic.

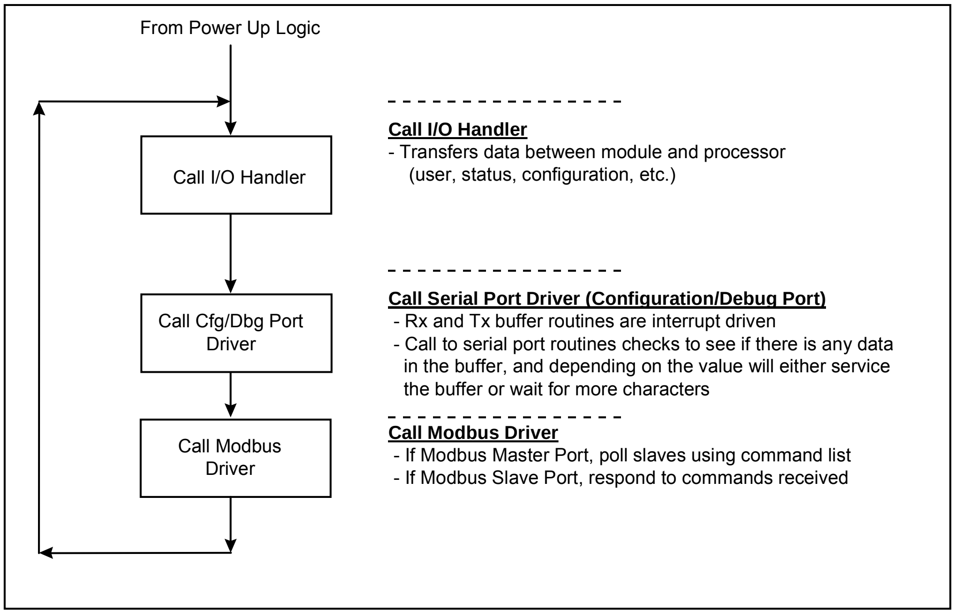
2 Understanding the Architecture

In This Chapter

This section gives the reader a functional overview of the MVI56-MCM module. Details associated with the ladder logic and the memory-map are not covered in this section (refer to ***Modifying the Module Configuration*** (on page 29)) A thorough understanding of the information contained in this document is required for successful implementation of the module in a user application. If you already understand the content of this section, refer to the ***Modifying the Module Configuration*** section to get the module up and running. If you are not familiar with the data transfer and Modbus protocol operations, read this document before setting up the module.

2.1 Main Logic Loop

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


From Power Up Logic
Call I/O Handler
Call Cfg/Dbg Port Driver
Call Modbus Driver
Call I/O Handler
- Transfers data between module and processor (user, status, configuration, etc.)
Call Serial Port Driver (Configuration/Debug Port)
- Rx and Tx buffer routines are interrupt driven
- Call to serial port routines checks to see if there is any data in the buffer, and depending on the value will either service the buffer or wait for more characters
Call Modbus Driver
- If Modbus Master Port, poll slaves using command list
- If Modbus Slave Port, respond to commands received

2.2 ControlLogix Processor Not in Run

Anytime the module detects that the processor has gone out of the Run mode (for example, Fault or PGM), the Modbus ports can be shut down as prescribed in the user configuration. When the processor is returned to a running state, the module will resume communications on the network.

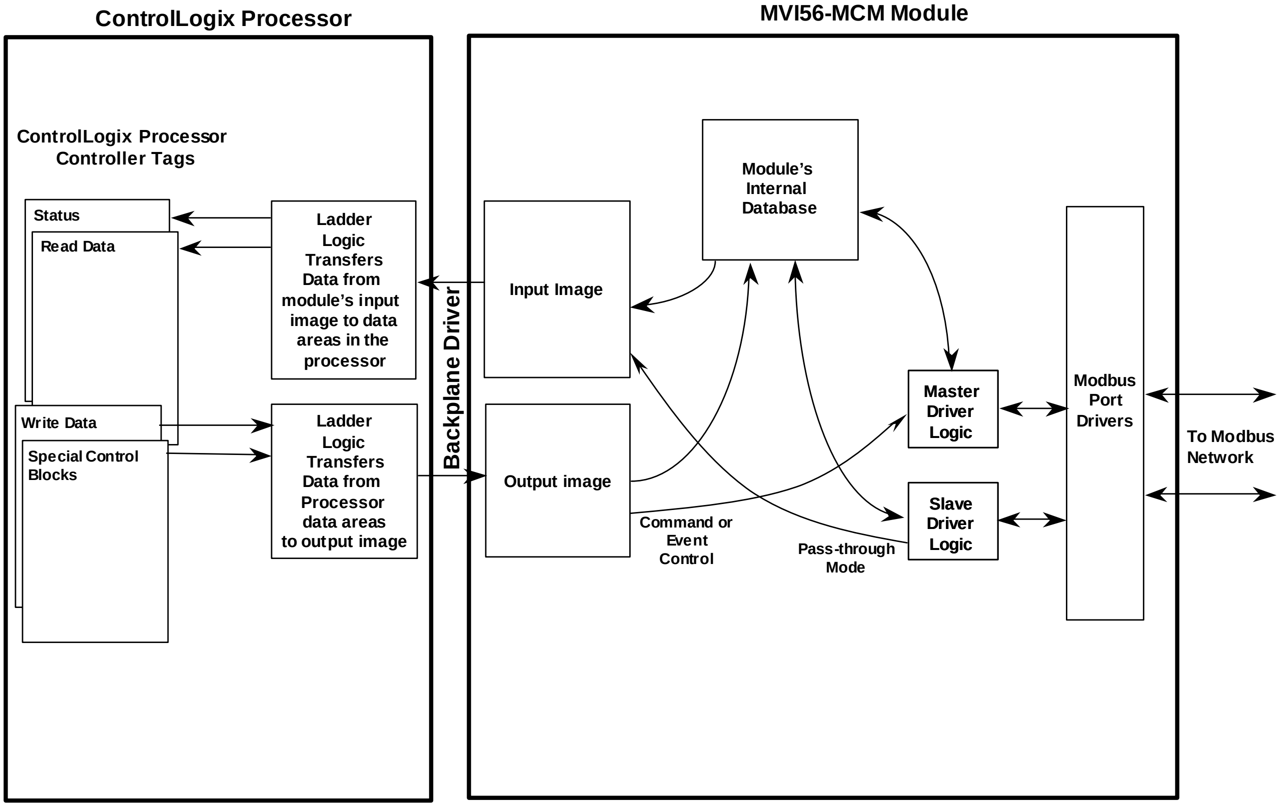
2.3 Backplane Data Transfer

The MVI56-MCM module is unique in the way that the ControlLogix backplane is utilized. Data is paged between the module and the ControlLogix processor across the backplane using the module's input and output images. The update frequency of the images is determined by the scheduled scan rate defined by the user for the module and the communication load on the module. Typical updates are in the range of 2 to 10 milliseconds.

This bi-directional transference of data is accomplished by the module filling in data in the module's input image to send to the processor. Data in the input image is placed in the Controller Tags in the processor by the ladder logic. The input image for the module is set to 250 words. This large data area permits fast throughput of data between the module and the processor.

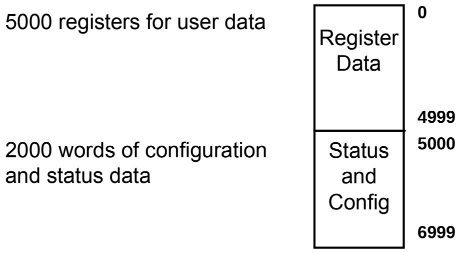
The processor inserts data to the module's output image to be transferred to the module. The module's program extracts the data and places it in the module's internal database. The output image for the module is set to 248 words. This large data area permits fast throughput of data from the processor to the module.

The following diagram shows the data transfer method used to move data between the ControlLogix processor, the MVI56-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 ControlLogix processor to interface the input and output image data with data defined in the Controller Tags. All data used by the module is stored in its internal database. This database is defined as a virtual Modbus data table with addresses from 0 (40001 Modbus) to 6999 (47000 Modbus). The diagram below displays the layout of the database:

Module's Internal Database Structure


5000 registers for user data
Register Data
0
4999
2000 words of configuration and status data
Status and Config
5000
6999

Data contained in this database is paged through the input and output images by coordination of the ControlLogix ladder logic and the MVI56-MCM module's program. Up to 248 words of data can be transferred from the module to the processor at once. Up to 247 words of data can be transferred from the processor to the module. Each image has a defined structure depending on the data content and the function of the data transfer as defined below.

2.4 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. Refer to the **Module Set Up** section for a description of the data objects used with the blocks and the ladder logic required. The structure and function of each block is discussed below.

2.4.1 Read Block

These blocks of data are used to transfer information from the module to the ControlLogix processor. The structure of the input image used to transfer this data is shown in the following table:

Offset	Description	Length
0	Reserved	1
1	Write Block ID	1
2 – 201	Read Data	200
202	Program Scan Counter	1

Offset	Description	Length
203 – 204	Product Code	2
205 – 206	Product Version	2
207 – 208	Operating System	2
209 – 210	Run Number	2
211 – 217	Port 1 Error Status	7
218 – 224	Port 2 Error Status	7
225 – 230	Data Transfer Status	6
231	Port 1 Current Error/Index	1
232	Port 1 Last Error/Index	1
233	Port 2 Current Error/Index	1
234	Port 2 Last Error/Index	1
235 – 248	Spare	14
249	Read Block ID	1

The Read Block ID is an index value used to determine the location of where the data will be placed in the ControlLogix processor controller tag array of module read data. Each transfer can move up to 200 words (block offsets 2 to 201) of data. In addition to moving user data, the block also contains status data for the module. This last set of data is transferred with each new block of data and is used for high-speed data movement.

The Write Block ID associated with the block is used to request data from the ControlLogix 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.

2.4.2 Write Block

These blocks of data are used to transfer information from the ControlLogix processor to the module. The structure of the output image used to transfer this data is shown in the following table:

Offset	Description	Length
0	Write Block ID	1
1 – 200	Write Data	200
201 – 247	Spare	47

The Write Block ID is an index value used to determine the location in the module's database where the data will be placed. Each transfer can move up to 200 words (block offsets 1 to 200) of data.

2.5 Configuration Data Transfer

When the module performs a restart operation, it will request configuration information from the ControlLogix processor. This data is transferred to the module in specially formatted write blocks (output image). The module will poll for each block by setting the required write block number in a read block (input image). Refer to the **Module Set Up** section for a description of the data objects used with the blocks and the ladder logic required. The format of the blocks for configuration is given in the following sections.

2.5.1 Module Configuration Data

This block is used to send general configuration information from the processor to the module. The data is transferred in a block with an identification code of 9000. The structure of the block is displayed in the following table:

Offset	Description	Length
0	9000	1
1 – 6	Backplane Setup	6
7 – 31	Port 1 Configuration	25
32 – 56	Port 2 Configuration	25
57 – 59	Port 1 Aux. Configuration	3
60 – 62	Port 2 Aux. Configuration	3
63 – 247	Spare	185

The read block used to request the configuration has the following structure:

Offset	Description	Length
0	Reserved	1
1	9000	1
2	Module Configuration Errors	1
3	Port 1 Configuration Errors	1
4	Port 2 Configuration Errors	1
5 – 248	Spare	244
249	-2 or –3	1

If there are any errors in the configuration, the bit associated with the error will be set in one of the three configuration error words. The error must be corrected before the module starts its normal mode of operation.

2.6 Master Command Data List

Each port on the module can be configured as a Modbus master device containing its own list of one hundred commands. The commands are read from the processor using the following Write Block ID's: Modbus Port 1 -- 6000 to 6003 and Modbus Port 2 -- 6100 to 6103. The module will sequentially poll for each block from the processor. Ladder logic must be written to handle each and every one of the data transfers. The structure of each block is shown in the following table.

Offset	Description	Length
0	6000 to 6003 and 6100 to 6103	1
1 – 8	Command Definition	8
9 – 16	Command Definition	8
17 – 24	Command Definition	8
25 – 32	Command Definition	8
33 – 40	Command Definition	8
41 – 48	Command Definition	8
49 – 56	Command Definition	8
57 – 64	Command Definition	8
65 – 72	Command Definition	8
73 – 80	Command Definition	8
81 – 88	Command Definition	8
89 – 96	Command Definition	8
97 – 104	Command Definition	8
105 – 112	Command Definition	8
113 – 120	Command Definition	8
121 – 128	Command Definition	8
129 – 136	Command Definition	8
137 – 144	Command Definition	8
145 – 152	Command Definition	8
153 – 160	Command Definition	8
161 – 168	Command Definition	8
169 – 176	Command Definition	8
177 – 184	Command Definition	8
185 – 192	Command Definition	8
193 – 200	Command Definition	8

2.7 Slave Status Blocks

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

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.
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 (**ErrorDelayCntr** value in the **MCMPort** object for each port). 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. This will enable polling of the slave.

BLOCK ID	DESCRIPTION
3002	Request for first 128 slave status values for Modbus Port 1
3003	Request for last 128 slave status values for Modbus Port 1
3102	Request for first 128 slave status values for Modbus Port 2
3103	Request for last 128 slave status values for Modbus Port 2

The format of these blocks is as shown in the following table:

Offset	Description	Length
0	3002 – 3003 or 3102 – 3103	1
1 – 247	Spare	246

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

Offset	Description	Length
0	Reserved	1
1	Write Block ID	1
2 – 129	Slave Poll Status Data	128
130 – 248	Spare	119
249	3002 – 3003 or 3102 – 3103	1

Ladder logic can be written to override the value in the slave status table. It can disable (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 in the following table:

Offset	Description	Length
0	3000 or 3100	1
1	Number of Slaves in Block	1

Offset	Description	Length
2 – 201	Slave indexes	200
202 – 247	Spare	46

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 in the following table:

Offset	Description	Length
0	Reserved	1
1	Write Block ID	1
2	Number of slaves processed	1
3 – 248	Spare	246
249	3000 or 3100	1

Ladder logic can be written 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 in the following table:

Offset	Description	Length
0	3001 or 3101	1
1	Number of Slaves in Block	1
2 – 201	Slave indexes	200
202 – 247	Spare	46

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 in the following table:

Offset	Description	Length
0	Reserved	1
1	Write Block ID	1
2	Number of slaves processed	1
3 – 248	Spare	246
249	3001 or 3101	1

2.8 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, write configuration, warm boot and cold boot.

2.8.1 Event Command

Event command control blocks are used to send Modbus commands directly from the ladder logic to one of the master ports. The format for these blocks is displayed in the following table:

Offset	Description	Length
0	1000 – 1255 or 2000 – 2255	1
1	Internal DB Address	1
2	Point Count	1
3	Swap Code	1
4	Modbus Function Code	1
5	Device Database Address	1
6 – 247	Spare	242

The block number defines the Modbus port to be considered and the slave node to be accessed. Blocks in the 1000 range are directed to Modbus Port 1, and blocks in the 2000 range are directed to Modbus 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 other parameters passed with the block are used to 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 points or registers for the command. The **Swap Code** is used with Modbus function 3 requests to change the word or byte order. The **Modbus Function Code** has one of the following values 1, 2, 3, 4, 5, 6, 15 or 16. The **Device Database Address** is the Modbus register or point in the remote slave device to be associated with the command. When the command receives the block, it will process it and place it in the command queue. The module will respond to each event command block with a read block with the following format:

Offset	Description	Length
0	Reserved	1
1	Write Block ID	1
2	0 = Fail, 1 = Success	1
3 – 248	Spare	246
249	1000 – 1255 or 2000 - 2255	1

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

2.8.2 Command Control

Command control blocks are used to 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 format of the block is displayed in the following table:

Offset	Description	Length
0	5001 – 5006 or 5101 – 5106	1
1	Command index (MCM.P1.CMD [*command index value*])	1
2	Command index (MCM.P1.CMD [*command index value*])	1
3	Command index (MCM.P1.CMD [*command index value*])	1
4	Command index (MCM.P1.CMD [*command index value*])	1
5	Command index (MCM.P1.CMD [*command index value*])	1
6	Command index (MCM.P1.CMD [*command index value*])	1
7 – 247	Spare	241

Blocks in the range of 5001 to 5006 are used for Modbus Port 1, and blocks in the range of 5101 to 5106 are used for Modbus 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 Modbus 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 in the following table:

Offset	Description	Length
0	Reserved	1
1	Write Block ID	1
2	Number of commands added to command queue	1
3 – 248	Spare	246
249	5000 – 5006 or 5100 - 5106	1

2.8.3 Write Configuration

This block is sent from the ControlLogix processor to the module to force the module to write its current configuration back to the processor. This function is used when the module's configuration has been altered remotely using database write operations. The write block contains a value of -9000 in the first word. The module will respond with blocks containing the module configuration data. Ladder logic must be written to handle the receipt of these blocks. The blocks transferred from the module are as follows:

Block -9000, General Configuration Data:

Offset	Description	Length
0	Reserved	1
1	-9000	1
2 – 7	Backplane Setup	6
8 – 32	Port 1 Configuration	25
33 – 57	Port 2 Configuration	25
58 – 60	Port 1 Aux. Configuration	3
61 – 63	Port 2 Aux. Configuration	3
64 – 248	Spare	185
249	-9000	1

Blocks -6000 to -6003 and -6100 to 6103, Master Command List Data for ports 1 and 2, respectively:

Offset	Description	Length
0	Reserved	1
1	-6000 to –6003 and –6100 to –6103	1
2 – 9	Command Definition	8
10 –17	Command Definition	8
18 – 25	Command Definition	8
26 – 33	Command Definition	8
34 – 41	Command Definition	8
42 – 49	Command Definition	8
50 – 57	Command Definition	8
58 – 65	Command Definition	8
66 – 73	Command Definition	8
74 – 81	Command Definition	8
82 – 89	Command Definition	8
90 – 97	Command Definition	8
98 – 105	Command Definition	8
106 – 113	Command Definition	8
114 – 121	Command Definition	8
122 – 129	Command Definition	8
130 – 137	Command Definition	8
138 – 145	Command Definition	8
146 – 153	Command Definition	8
154 – 161	Command Definition	8
162 – 169	Command Definition	8
170 – 177	Command Definition	8

Offset	Description	Length
178 – 185	Command Definition	8
186 – 193	Command Definition	8
194 – 201	Command Definition	8
202 – 248	Spare	47
249	-6000 to –6003 and –6100 to –6103	1

Each of these blocks must be handled by the ladder logic for proper module operation.

2.8.4 Warm Boot

This block is sent from the ControlLogix processor to the module (output image) when the module is required to perform a warm-boot (software reset) operation. This block is commonly sent to the module any time configuration data modifications are made in the controller tags data area. This will force the module to read the new configuration information and to restart. The structure of the control block is shown in the following table:

Offset	Description	Length
0	9998	1
1 – 247	Spare	247

2.8.5 Cold Boot

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

Offset	Description	Length
0	9999	1
1 – 247	Spare	247

2.9 Pass-Through Control Blocks

2.9.1 Unformatted Pass-Through Control Blocks

If one or more of the slave ports on the module are configured for the unformatted pass-through mode of operation, the module will pass blocks with identification codes of 9996 to the processor for each received write command. Any Modbus function 5, 6, 15 and 16 commands will be passed from the port to the processor using this block identification number. Ladder logic must be written to 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 unformatted pass-through control block is shown in the following table:

Offset	Description	Length
0	0	1
1	9996	1
2	Number of bytes in Modbus message	1
3 – 248	Modbus message received	246
249	9996	1

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

Offset	Description	Length
0	9996	1
1 – 247	Spare	247

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

2.9.2 Formatted Pass-Through Control Blocks

If one or more of the slave ports on the module are configured for the formatted pass-through mode of operation, the module will pass blocks with identification codes of 9996 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 be written to 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
0	0	1
1	9958	1
2	1	1
3	Bit Address	1
4	Data	1
5 – 248	Modbus message received	244
249	9958	1

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

Offset	Description	Length
0	9958	1
1 – 247	Spare	247

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

Function 6 and 16

Offset	Description	Length
0	0	1
1	9956/9957 (Floating-point)	1
2	Number of data words	1
3	Data Address	1
4 – 248	Data	244
249	9956/9957	1

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

Offset	Description	Length
0	9956/9957	1
1 – 247	Spare	247

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

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 is used to clear 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.

Offset	Description	Length
0	0	1
1	9959	1
2	Number of Words	1
3	Word Address	1
4 – 53	Data	50
54 – 103	Mask	50
104 – 248	Spare	145
249	9959	1

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

Offset	Description	Length
0	9959	1
1 – 247	Spare	247

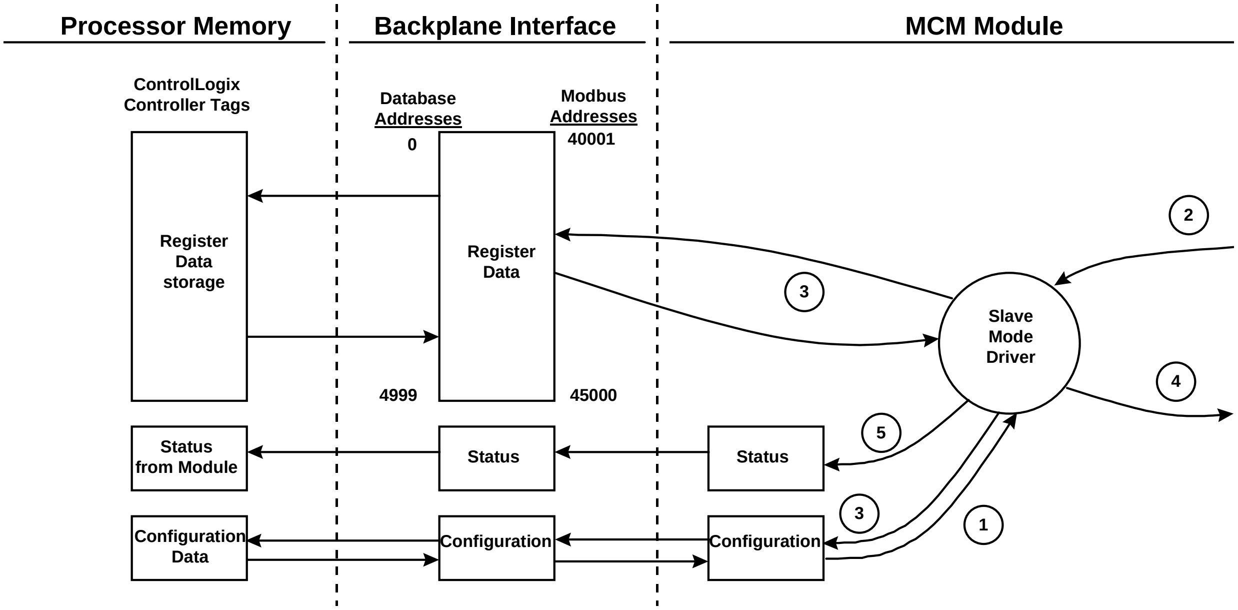
This will inform the module that the command has been processed and can be cleared from the pass-through queue.

2.10 Data Flow Between MVI56-MCM Module and ControlLogix Processor

The following discussion details the flow of data between the two pieces of hardware (ControlLogix processor and MVI56-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.

2.10.1 Slave Driver

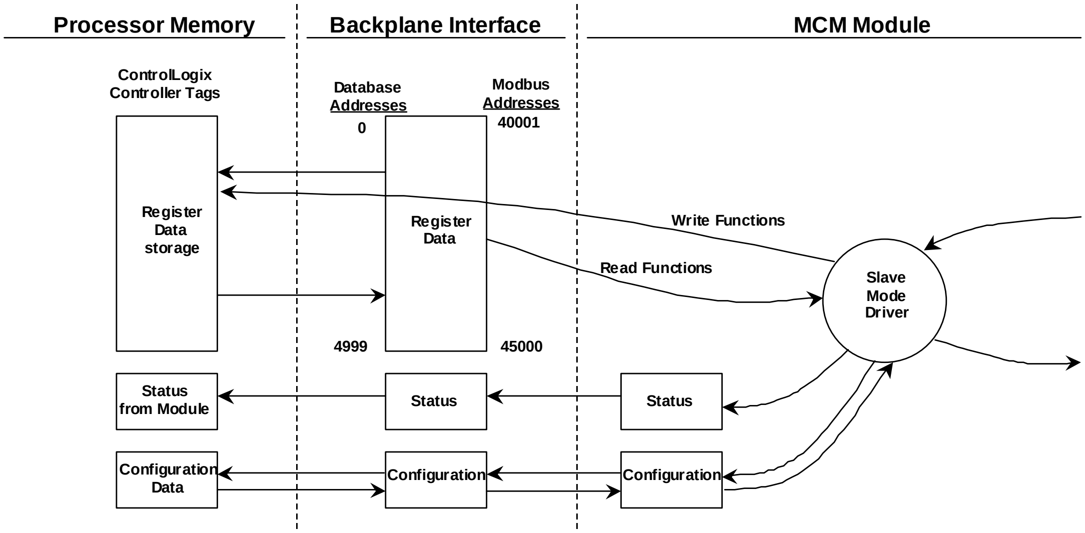
The Slave Driver Mode allows the MVI56-MCM module to respond to data read and write commands issued by a master on the Modbus network. The following flow chart and associated table detail the flow of data into and out of the module.



Step	Description
1	The Modbus slave port driver receives the configuration information from the ControlLogix processor. This information is used to configure 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	Once 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	Once 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.

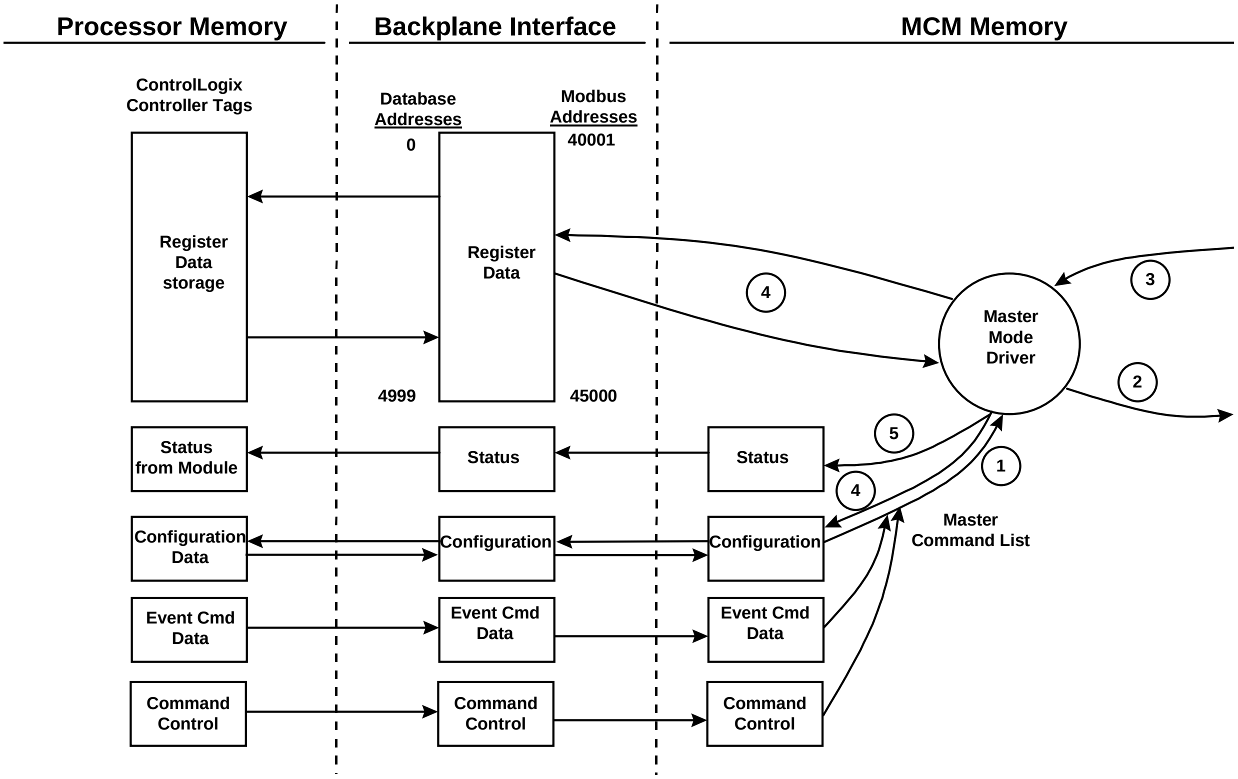
Review the **Module Set Up** section for a complete list of the parameters that must be defined for a slave port.

An exception to this normal mode of operation 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 of operation. The following diagram illustrates the dataflow for a slave port with pass-through enabled:



2.10.2 Master Driver Mode

In the Master Mode of operation, the MVI56-MCM module is responsible for issuing read or write commands to slave devices on the Modbus network. These commands are user configured in the module via the Master Command List received from the ControlLogix processor or issued directly from the ControlLogix processor (event command control). Command status is returned to the processor for each individual command in the command list status block. The location of this status block in the module's internal database is user defined. The following flow chart and associated table detail the flow of data into and out of the module.



Step	Description
1	The Master driver obtains configuration data from the ControlLogix processor. The configuration data obtained includes the number of commands and the Master Command List. 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 (See the MVI56-MCM Module Set Up Guide).
2	Once configured, 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.

Step	Description
5	Status is returned to the ControlLogix processor for each command in the Master Command List.

Refer to the **Module Set Up** section for a complete description of the parameters required to define the virtual Modbus master port. Refer to the **MCM Driver** documentation for a complete discussion of the structure and content of each command. Care must be taken in constructing each command in the list for predictable operation of the module. If two commands write to the same internal database address of the module, the results will not be as desired. All commands containing invalid data are ignored by the module.

Master Command List

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

- Command enable mode ((0) disabled, (1) continuous 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 – Determines where data will be placed and/or obtained
- Count – Select 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.

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

Code	Description
-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

3 Modifying the Module Configuration

In This Chapter

In order for the MVI56-MCM module to function, a minimum amount of configuration data must be transferred to the module. The table below provides an overview of the different types of configuration data that the module requires, depending on the operating modes to be supported.

Module Register Address	**Functional Modes Affected**	**Name**	**Description**
5000-5009	Data Transfer	General Module Configuration	This section of the configuration data contains the module configuration data that defines the data transfer between the module and the ControlLogix processor.
5010-5039 and 5040-5069	Master and Slave	Port Configuration	These sections are used to define the characteristics of each of the Modbus serial communication ports on the module. These parameters must be set correctly for proper module operation.
5070-5869 and 5870-6669	Master	Master Command List	If the module's Master Mode functionality is to be supported on a port, the Master Command List must be set up.
6750–6770	Master and Slave	Aux. Port Configuration	These sections are used to define the characteristics of each of the Modbus serial communication ports on the module. These parameters must be set correctly for proper module operation.

Refer to the Setting Up the Module section for a description of the configuration of the module. The MVI56-MCM module must be configured at least once when the card is first powered, and any time thereafter when the parameters must be changed.

3.1 Power Up

On power up, the module enters into a logical loop waiting to receive configuration data from the processor. Upon receipt, the module will begin execution of the command list if it is present.

3.2 Changing Parameters During Operation

A copy of the module's configuration data is mapped in the module's database as displayed in the table above. These values are initialized when the module first receives its configuration from the ControlLogix processor. Any node on the network can change this data. A master port on the module may poll a slave for the data or a slave port could receive the data from a remote master unit. The module will not use this data until it is commanded. Ladder logic can be written to issue a Write Configuration command block (9997) to the module. A remote device can set a value of 9997 at address 6800 in the module to download the configuration to the processor. Alternatively, the configuration/debug port on the module can be used to issue the command directly to the module. All three of these methods will force the module to download the configuration to the ControlLogix processor. Ladder logic must exist in the processor to accept the blocks sent by the module. If everything is configured correctly, the module can receive its configuration from a remote device.

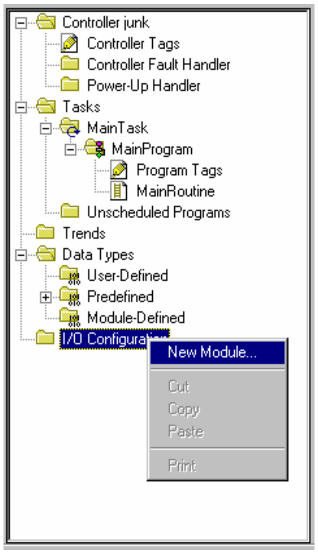
3.3 Setting Up the Module

Set up of the MVI56-MCM module only requires software configuration using the RSLogix 5000 program. The easiest method to implement the module is to start with the example provided with the module (MVI56_MCM_EX1.ACD).

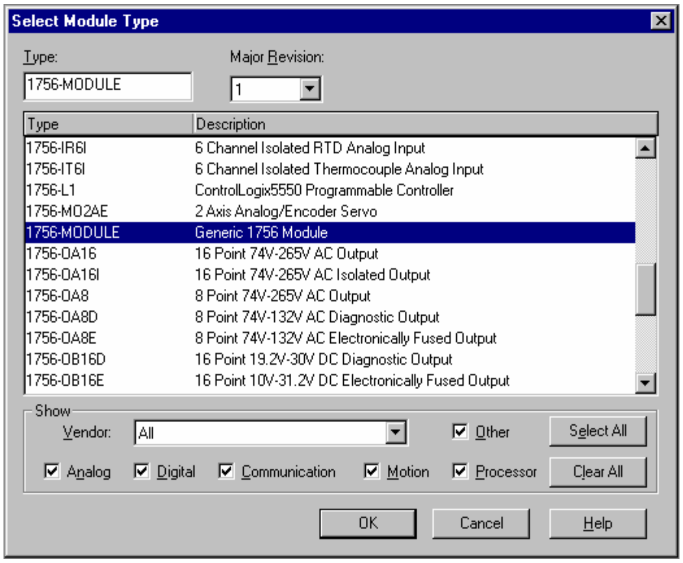
If you are installing this module in an existing application, you can simply copy the elements required from the example ladder logic to your application.

Note: The module can only be added to a project using the software in offline mode.

The first step in setting up the module is to define the 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 causes the program to display the following dialog box:

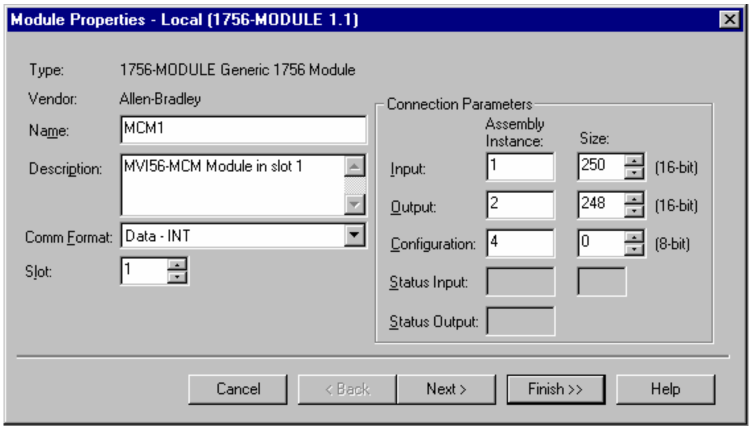

Select Module Type

Type: 1756-MODULE
Major Revision: 1

Type	Description
1756-IR6I	6 Channel Isolated RTD Analog Input
1756-IT6I	6 Channel Isolated Thermocouple Analog Input
1756-L1	ControlLogix5550 Programmable Controller
1756-M02AE	2 Axis Analog/Encoder Servo
1756-MODULE	Generic 1756 Module
1756-OA16	16 Point 74V-265V AC Output
1756-OA16I	16 Point 74V-265V AC Isolated Output
1756-OA8	8 Point 74V-265V AC Output
1756-OA8D	8 Point 74V-132V AC Diagnostic Output
1756-OA8E	8 Point 74V-132V AC Electronically Fused Output
1756-OB16D	16 Point 19.2V-30V DC Diagnostic Output
1756-OB16E	16 Point 10V-31.2V DC Electronically Fused Output

Show
Vendor: All
☑ Other
☑ Analog ☑ Digital ☑ Communication ☑ Motion ☑ Processor

Select the 1756-Module (Generic 1756 Module) from the list and select the OK button. The following dialog box is displayed:



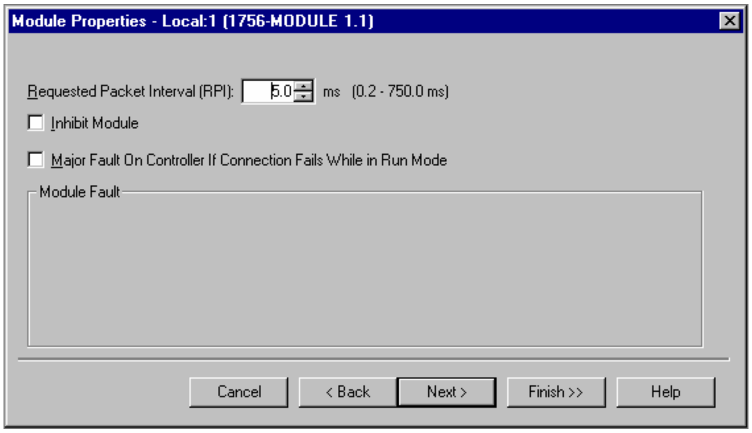
Module Properties - Local (1756-MODULE 1.1)

Type: 1756-MODULE Generic 1756 Module
Vendor: Allen-Bradley
Name: MCM1
Description: MVI56-MCM Module in slot 1
Comm Format: Data - INT
Slot: 1

Connection Parameters

	Assembly Instance:	Size:	
Input:	1	250	(16-bit)
Output:	2	248	(16-bit)
Configuration:	4	0	(8-bit)
Status Input:			
Status Output:			

Fill in the dialog boxes as shown adjusting the Name, Description and Slot options for your application. Be certain to 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 ControlLogix rack. Select the Next command button to display the following dialog box.

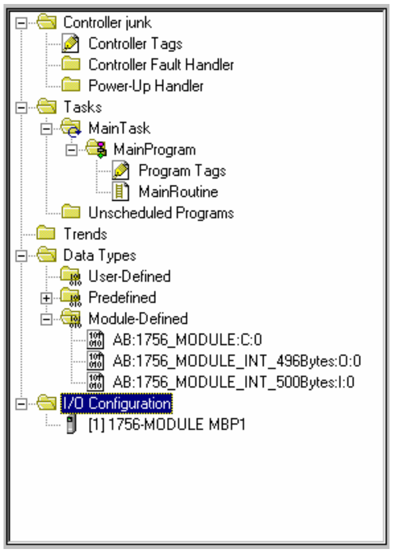


Module Properties - Local:1 (1756-MODULE 1.1)

Requested Packet Interval (RPI): 5.0 ms (0.2 - 750.0 ms)
□ Inhibit Module
□ Major Fault On Controller If Connection Fails While in Run Mode

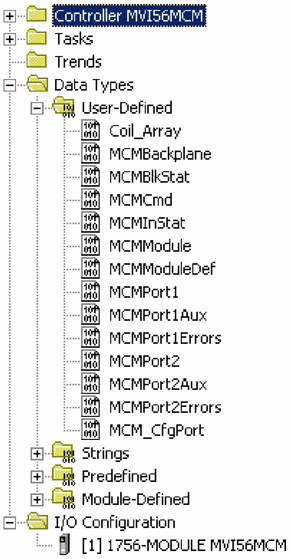
Select the Request Packet Interval value for scanning the I/O on the module. This value represents the minimum frequency the module will handle scheduled events. This value should not be set to less than 1 millisecond. Values between 1 and 10 milliseconds should work with most applications.

After completing the module setup, the Controller Organization window displays the module's presence. The data required for the module is then defined to the application, and objects are allocated in the Controller Tags data area. An example of the Controller Organization window is shown in the following example:



```
Controller junk
    Controller Tags
    Controller Fault Handler
    Power-Up Handler
Tasks
    MainTask
        MainProgram
            Program Tags
            MainRoutine
    Unscheduled Programs
Trends
Data Types
    User-Defined
    Predefined
    Module-Defined
        AB:1756_MODULE:C:0
        AB:1756_MODULE_INT_496Bytes:O:0
        AB:1756_MODULE_INT_500Bytes:I:0
I/O Configuration
    [1] 1756-MODULE MBP1
```

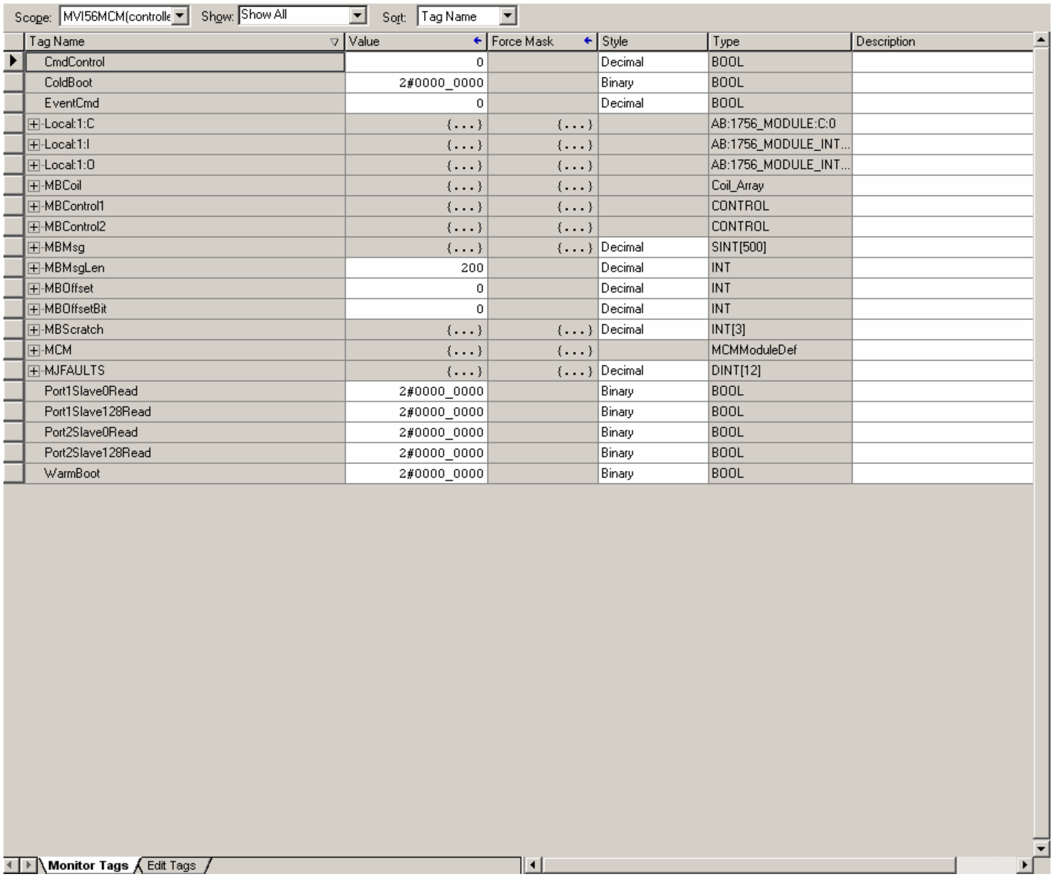
The next step in the module's setup is to define the User Defined Data Types to be used with the module. Copy these data types from the example ladder logic if you are not using the example. They are defined if you are starting from the example ladder logic. The Controller Organization window should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



```
Controller MVI56MCM
Tasks
Trends
Data Types
    User-Defined
        Coil_Array
        MCMBackplane
        MCMBlkStat
        MCMCmd
        MCMInStat
        MCMModule
        MCMModuleDef
        MCMPort1
        MCMPort1Aux
        MCMPort1Errors
        MCMPort2
        MCMPort2Aux
        MCMPort2Errors
        MCM_CfgPort
    Strings
    Predefined
    Module-Defined
I/O Configuration
    [1] 1756-MODULE MVI56MCM
```

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 MVI56-MCM module is defined in the example as MCM1. 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.

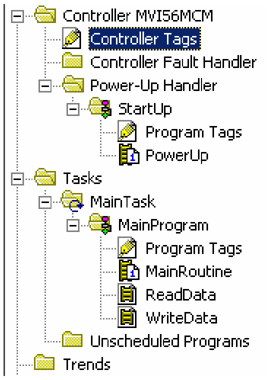


Scope: MVI56MCM(controlle Show: Show All Sort: Tag Name

Tag Name	Value	Force Mask	Style	Type	Description
CmdControl	0		Decimal	BOOL	
ColdBoot	2#0000_0000		Binary	BOOL	
EventCmd	0		Decimal	BOOL	
+Local:1:C	{...}	{...}		AB:1756_MODULE:C:0	
+Local:1:I	{...}	{...}		AB:1756_MODULE_INT...	
+Local:1:O	{...}	{...}		AB:1756_MODULE_INT...	
+MBCoil	{...}	{...}		Coil_Array	
+MBControl1	{...}	{...}		CONTROL	
+MBControl2	{...}	{...}		CONTROL	
+MBMsg	{...}	{...}	Decimal	SINT[500]	
+MBMsgLen	200		Decimal	INT	
+MBOffset	0		Decimal	INT	
+MBOffsetBit	0		Decimal	INT	
+MBScratch	{...}	{...}	Decimal	INT[3]	
+MCM	{...}	{...}		MCMModuleDef	
+MJFAULTS	{...}	{...}	Decimal	DINT[12]	
Port1Slave0Read	2#0000_0000		Binary	BOOL	
Port1Slave128Read	2#0000_0000		Binary	BOOL	
Port2Slave0Read	2#0000_0000		Binary	BOOL	
Port2Slave128Read	2#0000_0000		Binary	BOOL	
WarmBoot	2#0000_0000		Binary	BOOL	

At this point, take the time to fill in the configuration values in the MCM1 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 in the module setup 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 window below to your application.

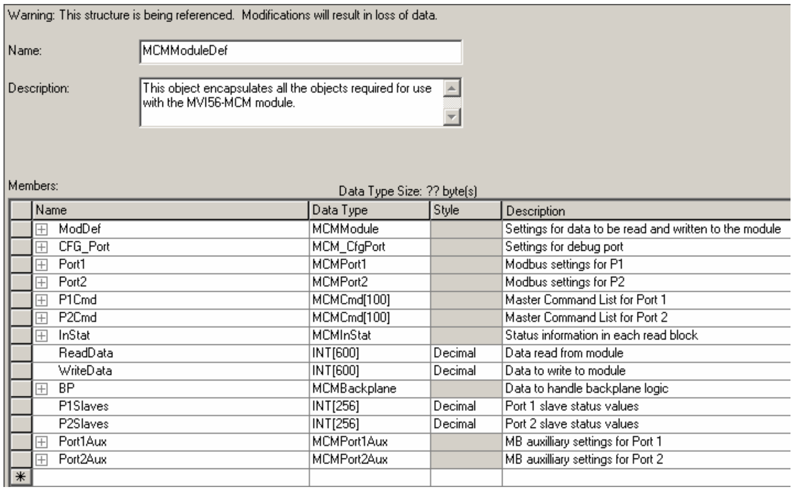


The module is now set up and ready to be used with your application.

Download the new application to the processor 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 off and the backplane activity LED (BP ACT) should blink very rapidly. Refer to the **Diagnostics and Troubleshooting** section of this manual if you encounter errors. Attach a computer or terminal to Debug/Configuration port on the module and check the status of the module using the resident debugger in the module.

3.4 Module Data Object (MCMModuleDef)

All data related to the MVI56-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 simply 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.



Warning: This structure is being referenced. Modifications will result in loss of data.

Name: MCMModuleDef

Description: This object encapsulates all the objects required for use with the MVI56-MCM module.

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

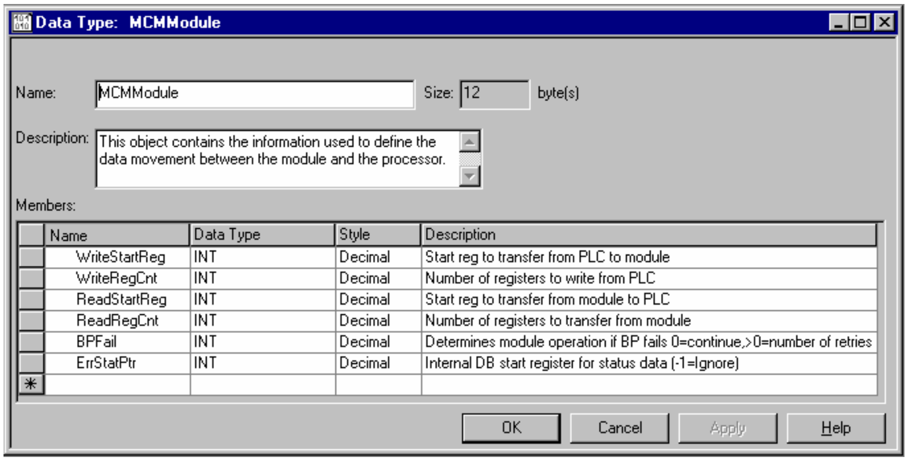
Name	Data Type	Style	Description
ModDef	MCMModule		Settings for data to be read and written to the module
CFG_Port	MCM_CfgPort		Settings for debug port
Port1	MCMPort1		Modbus settings for P1
Port2	MCMPort2		Modbus settings for P2
P1Cmd	MCMCmd[100]		Master Command List for Port 1
P2Cmd	MCMCmd[100]		Master Command List for Port 2
InStat	MCMInStat		Status information in each read block
ReadData	INT[600]	Decimal	Data read from module
WriteData	INT[600]	Decimal	Data to write to module
BP	MCMBackplane		Data to handle backplane logic
P1Slaves	INT[256]	Decimal	Port 1 slave status values
P2Slaves	INT[256]	Decimal	Port 2 slave status values
Port1Aux	MCMPort1Aux		MB auxilliary settings for Port 1
Port2Aux	MCMPort2Aux		MB auxilliary settings for Port 2

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

3.4.1 Configuration Objects

Configuration of the module is performed by simply filling in the values in the module object defined in the Controller Tags Edit Tags dialog. Each parameter required by the module has a defined location in the object. The following tables and discussions describe the parameters set in the dialog box. You can view these tables by opening the data type under the User Defined Data Type option in the Controller Organization window.

Data Transfer Parameters (MCMModule)



Name: MCMModule

Size: 12 byte(s)

Description: This object contains the information used to define the data movement between the module and the processor.

Members:

Name	Data Type	Style	Description
WriteStartReg	INT	Decimal	Start reg to transfer from PLC to module
WriteRegCnt	INT	Decimal	Number of registers to write from PLC
ReadStartReg	INT	Decimal	Start reg to transfer from module to PLC
ReadRegCnt	INT	Decimal	Number of registers to transfer from module
BPFail	INT	Decimal	Determines module operation if BP fails 0=continue,>0=number of retries
ErrStatPtr	INT	Decimal	Internal DB start register for status data (-1=Ignore)

This object is used to define the parameters for data movement 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 be written to process the number of blocks of data to be transferred. The number of blocks is computed as follows:

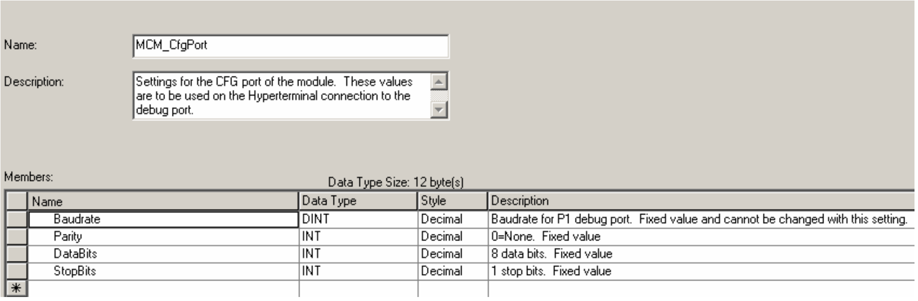
BlockCnt = INT(RegCnt/200) + if(MOD(RegCnt,200), 1,0)

If the register count is evenly divisible by 200, the number of blocks is easy to compute and the ladder is much simpler to write. If the number is not evenly divisible by 200, special handling of the last block of data must developed, as it must transfer less than 200 words. **It is recommended that the count values always be set to values evenly divisible by 200.**

The BPFail parameter is used to determine 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 is used to define 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.

Modbus Port Parameters (MCMPort)

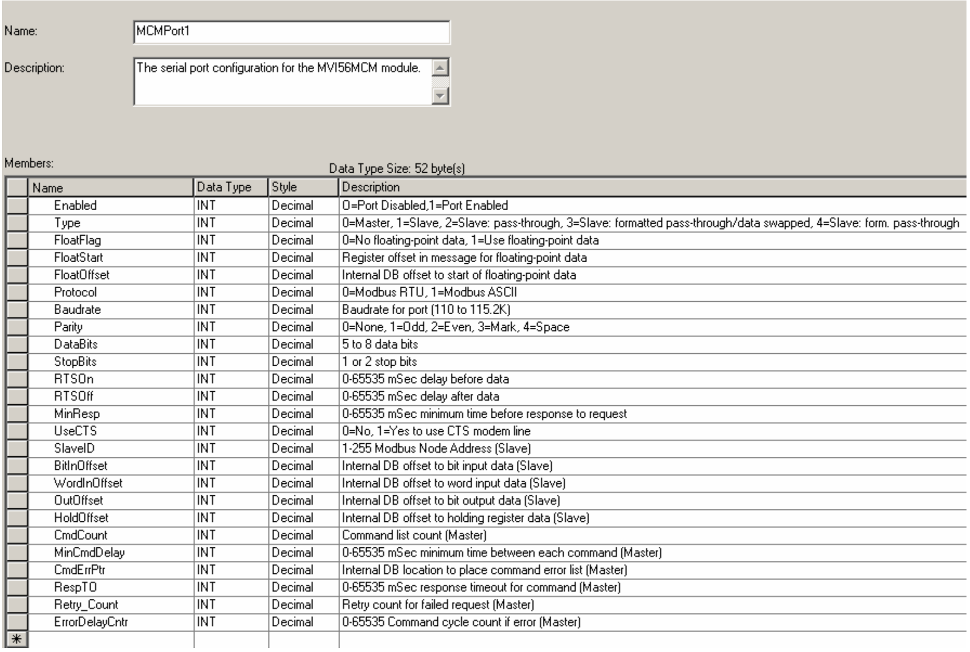


Name: MCM_CfgPort

Description: Settings for the CFG port of the module. These values are to be used on the Hyperterminal connection to the debug port.

Members: Data Type Size: 12 byte(s)

Name	Data Type	Style	Description
Baudrate	DINT	Decimal	Baudrate for P1 debug port. Fixed value and cannot be changed with this setting.
Parity	INT	Decimal	0=None. Fixed value
DataBits	INT	Decimal	8 data bits. Fixed value
StopBits	INT	Decimal	1 stop bits. Fixed value



Name: MCMPort1

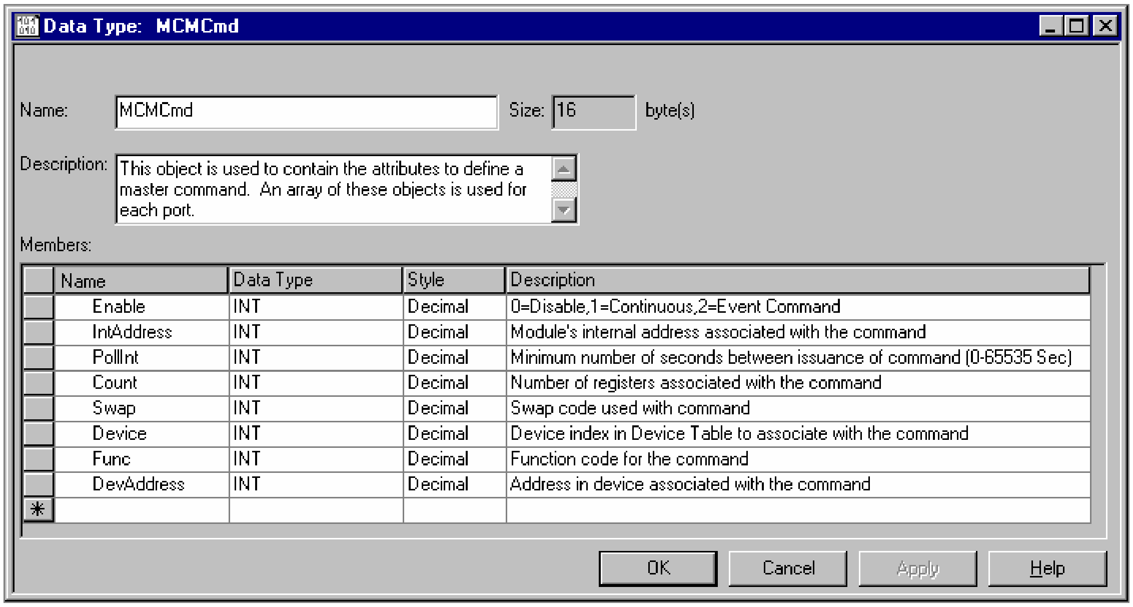
Description: The serial port configuration for the MVI56MCM module.

Members: Data Type Size: 52 byte(s)

Name	Data Type	Style	Description
Enabled	INT	Decimal	0=Port Disabled,1=Port Enabled
Type	INT	Decimal	0=Master, 1=Slave, 2=Slave: pass-through, 3=Slave: formatted pass-through/data swapped, 4=Slave: form. pass-through
FloatFlag	INT	Decimal	0=No floating-point data, 1=Use floating-point data
FloatStart	INT	Decimal	Register offset in message for floating-point data
FloatOffset	INT	Decimal	Internal DB offset to start of floating-point data
Protocol	INT	Decimal	0=Modbus RTU, 1=Modbus ASCII
Baudrate	INT	Decimal	Baudrate for port (110 to 115.2K)
Parity	INT	Decimal	0=None, 1=Odd, 2=Even, 3=Mark, 4=Space
DataBits	INT	Decimal	5 to 8 data bits
StopBits	INT	Decimal	1 or 2 stop bits
RTSOn	INT	Decimal	0-65535 mSec delay before data
RTSOff	INT	Decimal	0-65535 mSec delay after data
MinResp	INT	Decimal	0-65535 mSec minimum time before response to request
UseCTS	INT	Decimal	0=No, 1=Yes to use CTS modem line
SlaveID	INT	Decimal	1-255 Modbus Node Address (Slave)
BitInOffset	INT	Decimal	Internal DB offset to bit input data (Slave)
WordInOffset	INT	Decimal	Internal DB offset to word input data (Slave)
OutOffset	INT	Decimal	Internal DB offset to bit output data (Slave)
HoldOffset	INT	Decimal	Internal DB offset to holding register data (Slave)
CmdCount	INT	Decimal	Command list count (Master)
MinCmdDelay	INT	Decimal	0-65535 mSec minimum time between each command (Master)
CmdErrPtr	INT	Decimal	Internal DB location to place command error list (Master)
RespTO	INT	Decimal	0-65535 mSec response timeout for command (Master)
Retry_Count	INT	Decimal	Retry count for failed request (Master)
ErrorDelayCntr	INT	Decimal	0-65535 Command cycle count if error (Master)

This object is used to define the parameters for the operation of each of the Modbus ports on the module. Refer to Appendix C for the definition of each parameter.

Modbus Master Commands (MCMCmd)



Name: MCMCmd

Size: 16 byte(s)

Description: This object is used to contain the attributes to define a master command. An array of these objects is used for each port.

Members:

Name	Data Type	Style	Description
Enable	INT	Decimal	0=Disable,1=Continuous,2=Event Command
IntAddress	INT	Decimal	Module's internal address associated with the command
PollInt	INT	Decimal	Minimum number of seconds between issuance of command (0-65535 Sec)
Count	INT	Decimal	Number of registers associated with the command
Swap	INT	Decimal	Swap code used with command
Device	INT	Decimal	Device index in Device Table to associate with the command
Func	INT	Decimal	Function code for the command
DevAddress	INT	Decimal	Address in device associated with the command

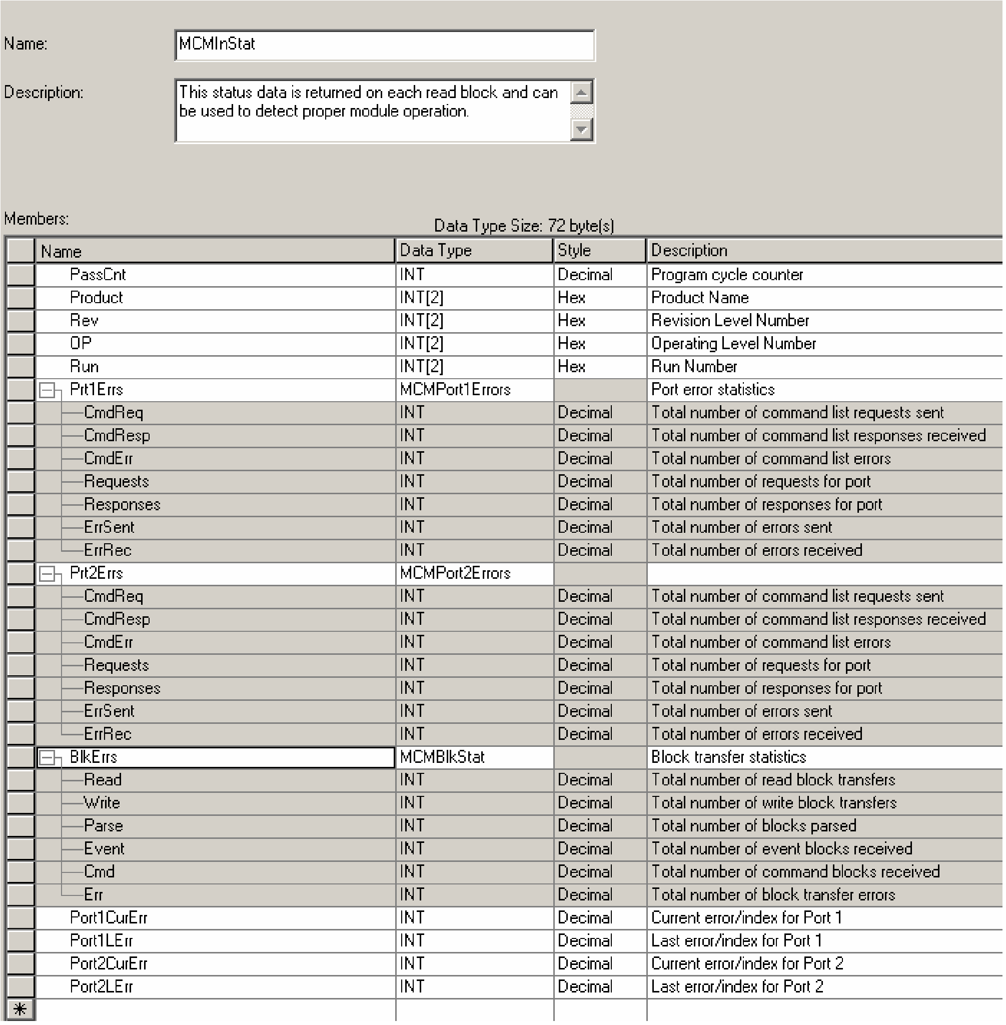
This object is used to define the parameters for each command in the master command list. The **MCMModuleDef** object contains an array of these objects that define the complete list for each port. The definition of each parameter required for each command is given below:

Parameter	**Description**
Enable	This parameter is used to define if the command will be executed or will be disregarded. The following values are valid: 0=Disables the command and it will not execute. 1=The command will be considered for execution each scan of the command list and will be controlled by the PollInt parameter. And 2=The command will only execute if the data associated with the command has changed since the command was last issued. This option is only available for write commands.
IntAddress	This parameter specifies the starting internal register address to be associated with the command. Valid entry for this parameter is 0 to 4999 registers or 0 to 65535 bits when addressing bit-level command.
PollInt	This parameter defines the minimum number of seconds to wait between the execution of continuous commands (Enable=1). This poll interval command can be used to lighten the communications load on a busy network. Valid entry for this parameter is 0 to 65535.
Count	This parameter defines the number of registers to be considered by the command. Valid entry for this parameter is 1 to 125 words or 2000 bits.
Swap	This parameter is used to specify if the data used in the command must be altered when a Modbus function code 3 is used to read data from a node on the network. Values that can be assigned are as follows: 0=no swapping of data, 1=swap word values, 2=swap word and byte values and 3=swap byte values. This option is used when interfacing the module with ASCII and floating-point data on other devices.

Parameter	Description
Device	This parameter is used to assign the Modbus slave node address for the module to reach with the command on the Modbus network. This parameter can be assigned values from 0 to 255. Most Modbus networks limit the upper value to 247.
Func	This parameter specifies the Modbus function to be performed by the command. Valid entries are 1, 2, 3, 4, 5, 6, 15 and 16.
DevAddress	This parameter defines the starting address in the device being considered by the command. Values entered in this field are dependent on the node's database definition. Refer to the specific manufacture's database definition for the device to determine the location of the data to be interfaced.

3.4.2 Status Object (MCMInStat)

This object is used to view the status of the module. The **MCMInStat** object shown below is updated each time a read block is received by the processor. This data can be used to monitor the state of the module at a "real-time rate".



Name: MCMInStat

Description: This status data is returned on each read block and can be used to detect proper module operation.

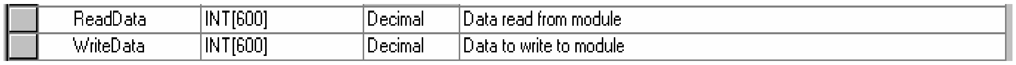
Members: Data Type Size: 72 byte(s)

Name	Data Type	Style	Description
PassCnt	INT	Decimal	Program cycle counter
Product	INT[2]	Hex	Product Name
Rev	INT[2]	Hex	Revision Level Number
OP	INT[2]	Hex	Operating Level Number
Run	INT[2]	Hex	Run Number
⊟ Prt1Errs	MCMPort1Errors		Port error statistics
—CmdReq	INT	Decimal	Total number of command list requests sent
—CmdResp	INT	Decimal	Total number of command list responses received
—CmdErr	INT	Decimal	Total number of command list errors
—Requests	INT	Decimal	Total number of requests for port
—Responses	INT	Decimal	Total number of responses for port
—ErrSent	INT	Decimal	Total number of errors sent
—ErrRec	INT	Decimal	Total number of errors received
⊟ Prt2Errs	MCMPort2Errors		
—CmdReq	INT	Decimal	Total number of command list requests sent
—CmdResp	INT	Decimal	Total number of command list responses received
—CmdErr	INT	Decimal	Total number of command list errors
—Requests	INT	Decimal	Total number of requests for port
—Responses	INT	Decimal	Total number of responses for port
—ErrSent	INT	Decimal	Total number of errors sent
—ErrRec	INT	Decimal	Total number of errors received
⊟ BlkErrs	MCMBlkStat		Block transfer statistics
—Read	INT	Decimal	Total number of read block transfers
—Write	INT	Decimal	Total number of write block transfers
—Parse	INT	Decimal	Total number of blocks parsed
—Event	INT	Decimal	Total number of event blocks received
—Cmd	INT	Decimal	Total number of command blocks received
—Err	INT	Decimal	Total number of block transfer errors
Port1CurErr	INT	Decimal	Current error/index for Port 1
Port1LErr	INT	Decimal	Last error/index for Port 1
Port2CurErr	INT	Decimal	Current error/index for Port 2
Port2LErr	INT	Decimal	Last error/index for Port 2

Refer to Appendix B for a complete listing of the data stored in this object.

3.5 User Data Objects

These objects are used to hold data to be transferred between the processor and the MVI56-MCM module. The user data is the read and write data transferred between the processor and the module as "pages" of data up to 200 words long.



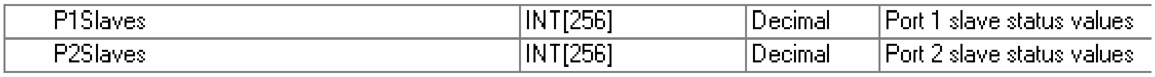
ReadData	INT[600]	Decimal	Data read from module
WriteData	INT[600]	Decimal	Data to write to module

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 200 words. This data is paged up to 200 words at a time from the module to the processor. The ReadData task is responsible for placing the data received into the proper position in the read data array. This data can be used 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 200 words. This data is paged up to 200 words at a time from the processor to the module. The WriteData task is responsible for placing 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 > 600 registers, change the High LIM value in ReadData rung 2 and WriteData rung 10 of the ladder file.

3.6 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:



P1Slaves	INT[256]	Decimal	Port 1 slave status values
P2Slaves	INT[256]	Decimal	Port 2 slave status values

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.

3.7 Modbus Message Data

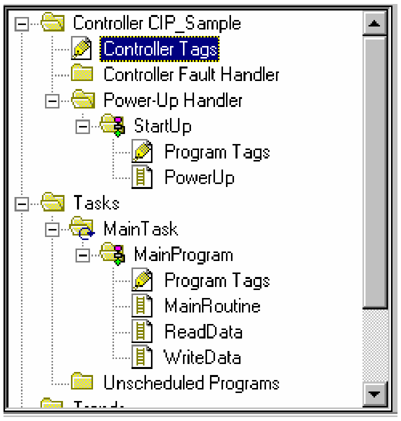
This new version of the module's program includes the pass-through mode of operation. In this mode, write messages sent to a slave port are passed directly through to the processor. It is the responsibility of the ladder logic to process the message received using this feature. Two data objects are required for this mode of operation: 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).

4 Modifying the Sample Ladder Logic

In This Chapter

Ladder logic is required for application of the MVI56-MCM module. Tasks that must be handled by the ladder logic are module configuration, data transfer, special block handling and status data receipt. This section discusses each aspect of the ladder logic as required by the module. Additionally, a power-up handler should be written to handle the initialization of the module's data and to clear any processor fault conditions.

The Controller Organization window for the example ladder logic for the MVI56-MCM module is shown in the following example.

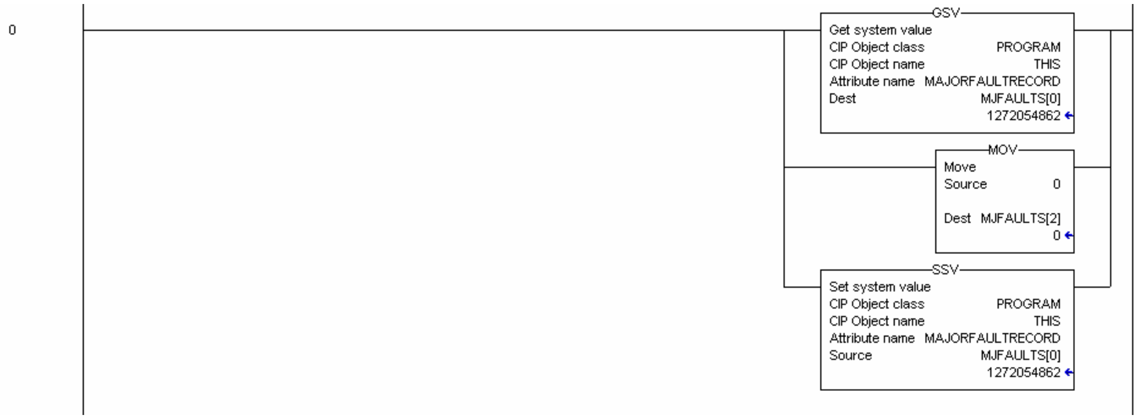


```
Controller CIP_Sample
    Controller Tags
    Controller Fault Handler
    Power-Up Handler
        StartUp
            Program Tags
            PowerUp
Tasks
    MainTask
        MainProgram
            Program Tags
            MainRoutine
            ReadData
            WriteData
    Unscheduled Programs
```

4.1 PowerUp

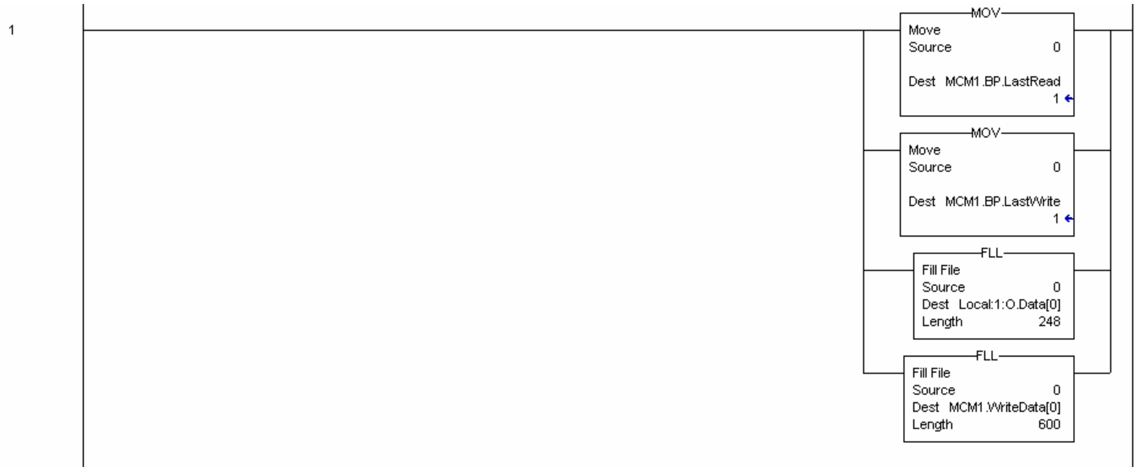
The PowerUp ladder logic is used to initialize the data objects used by the MVI56-MCM module and to recover controller faults on initial power-up of the processor. The ladder logic required to perform these tasks is shown in the following paragraphs:

This rung is used to recover from a processor fault condition due to power-loss and restart when the processor is in run mode. You may also have to handle other fault conditions. Additionally, a fault handler can be written for the processor to handle other faults. The MJFAULTS object must be defined in the Controller Tags before it can be used in this logic:



```
0
GSV
Get system value
CIP Object class          PROGRAM
CIP Object name              THIS
Attribute name  MAJORFAULTRECORD
Dest                  MJFAULTS[0]
                       1272054862

MOV
Move
Source          0
Dest  MJFAULTS[2]
                0

SSV
Set system value
CIP Object class          PROGRAM
CIP Object name              THIS
Attribute name  MAJORFAULTRECORD
Source                MJFAULTS[0]
                       1272054862
```

This rung is used to initialize the last read and write values, the output image for the MVI56-MCM module and the write data area to zero. The last read (**MCM.BP.LastRead**) and write (**MCM.BP.LastWrite**) values are used in the data transfer logic. The output image for the MVI56-MCM module (**Local:1:O.Data[ ]**) is used to transfer data from the processor to the module. The write data area (**MCM.WriteData[ ]**) is used to store the processor data to be written to the module using the output image.

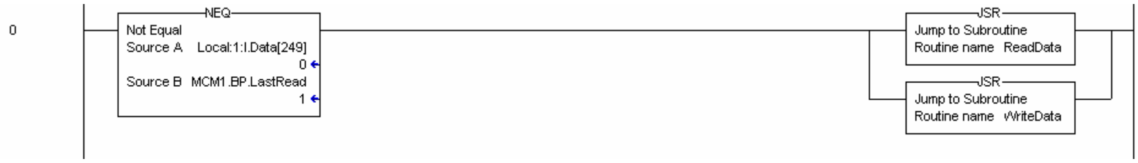


```
1
MOV
Move
Source                 0
Dest  MCM1.BP.LastRead
                       1

MOV
Move
Source                 0
Dest  MCM1.BP.LastWrite
                       1

FLL
Fill File
Source                 0
Dest   Local:1:O.Data[0]
Length               248

FLL
Fill File
Source                 0
Dest  MCM1.WriteData[0]
Length               600
```

4.2 MainRoutine

The MainRoutine is used to recognize the presence of new read data from the module for the processor. The module will cycle through its list of read blocks to transfer data from the module to the processor. Whenever new data is available, the module will set the value for the block in the module's input image

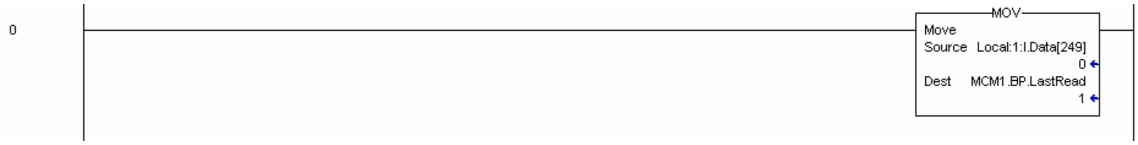
(**Local:1:I:Data[249]**). The ladder logic must constantly scan this input word for a new value. When a new value is present, the ladder logic should perform the ReadData and WriteData tasks in that order.



0
NEQ
Not Equal
Source A Local:1:I.Data[249]
0
Source B MCM1.BP.LastRead
1
JSR
Jump to Subroutine
Routine name ReadData
JSR
Jump to Subroutine
Routine name WriteData

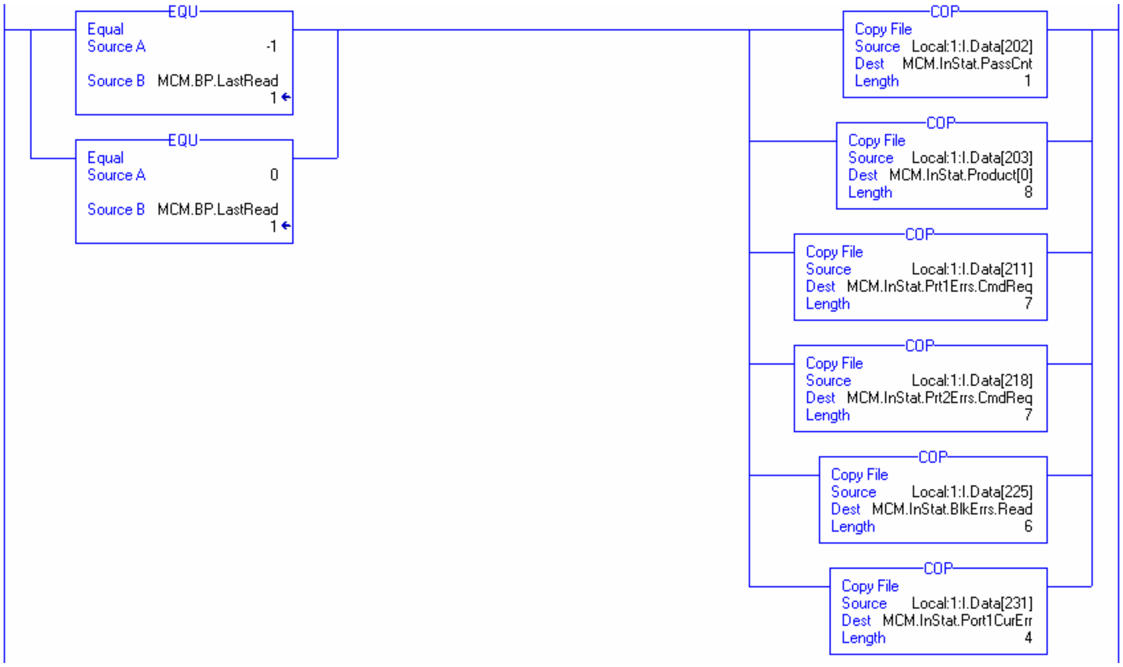
4.3 ReadData

The ReadData task is responsible for handling all new data received from the module and placing it in the proper location in the processor. Data is transferred from the module to the processor using the module's input image (**Local:1:I:Data[ ]**). The first rung of the task sets the last read block number (**MCM1.BP.LastRead**) to the current block number sent from the module (**Local:1:I:Data[249]**).



0
MOV
Move
Source Local:1:I.Data[249]
0
Dest MCM1.BP.LastRead
1

If the module is configured for zero blocks, it will send blocks with identification codes of zero and –1. No user data will be included in these blocks. They will only contain the status data. The rung shown below displays logic to handle these blocks.



EQU
Equal
Source A -1
Source B MCM.BP.LastRead
1

EQU
Equal
Source A 0
Source B MCM.BP.LastRead
1

COP
Copy File
Source Local:1:I.Data[202]
Dest MCM.InStat.PassCnt
Length 1

COP
Copy File
Source Local:1:I.Data[203]
Dest MCM.InStat.Product[0]
Length 8

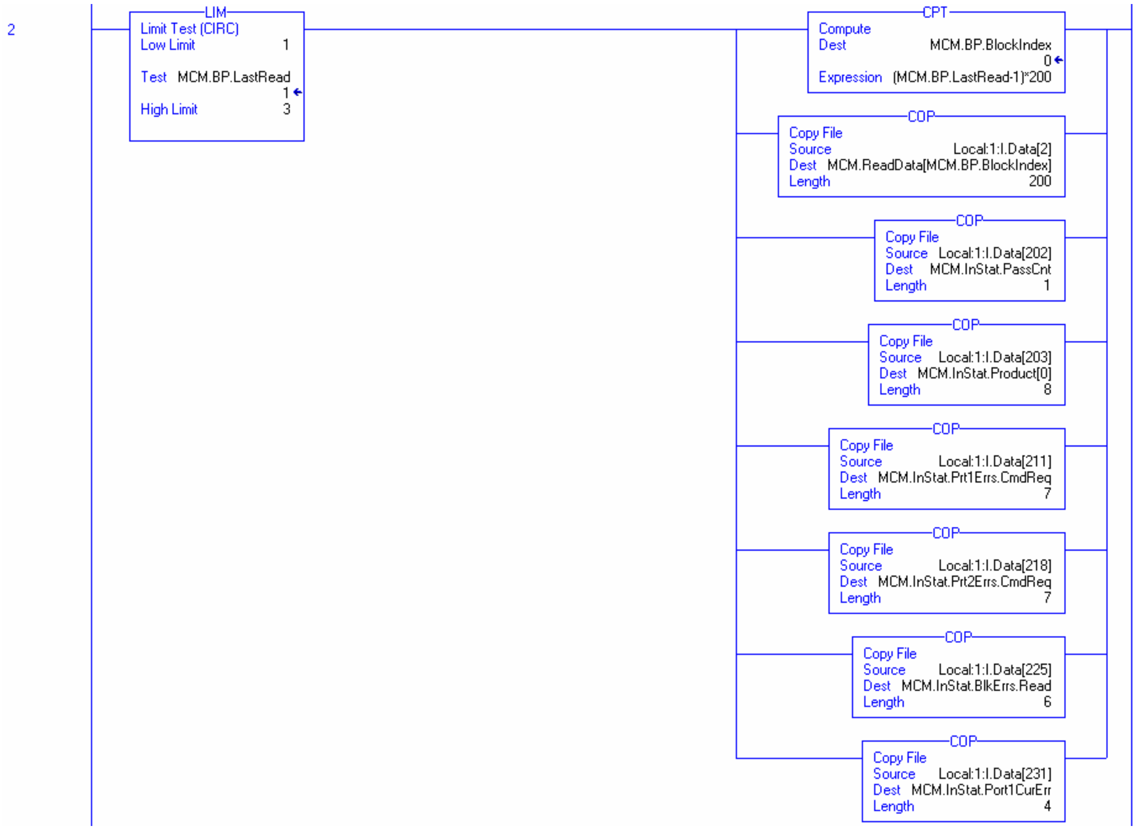
COP
Copy File
Source Local:1:I.Data[211]
Dest MCM.InStat.Prt1Errs.CmdReq
Length 7

COP
Copy File
Source Local:1:I.Data[218]
Dest MCM.InStat.Prt2Errs.CmdReq
Length 7

COP
Copy File
Source Local:1:I.Data[225]
Dest MCM.InStat.BlkErrs.Read
Length 6

COP
Copy File
Source Local:1:I.Data[231]
Dest MCM.InStat.Port1CurErr
Length 4

The next rung of the ladder logic determines if the new data received in the input image is user data. If user data is present, the ladder logic will place the data in the correct location in the processor's read data area (**MCM.ReadData[ ]**). Up to 200 data words can be transferred in each block transfer. In addition to the user data, the block also contains important status data. This data should be copied to the correct data area in the module (**MCM.InStat**). This status data can be used to determine the "health" of the MVI56-MCM module.



2

LIM
Limit Test (CIRC)
Low Limit 1
Test MCM.BP.LastRead
1
High Limit 3

CPT
Compute
Dest MCM.BP.BlockIndex
0
Expression (MCM.BP.LastRead-1)*200

COP
Copy File
Source Local:1:I.Data[2]
Dest MCM.ReadData[MCM.BP.BlockIndex]
Length 200

COP
Copy File
Source Local:1:I.Data[202]
Dest MCM.InStat.PassCnt
Length 1

COP
Copy File
Source Local:1:I.Data[203]
Dest MCM.InStat.Product[0]
Length 8

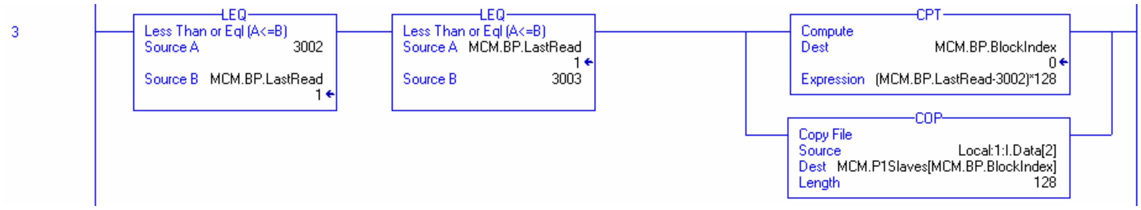
COP
Copy File
Source Local:1:I.Data[211]
Dest MCM.InStat.Prt1Errs.CmdReq
Length 7

COP
Copy File
Source Local:1:I.Data[218]
Dest MCM.InStat.Prt2Errs.CmdReq
Length 7

COP
Copy File
Source Local:1:I.Data[225]
Dest MCM.InStat.BlkErrs.Read
Length 6

COP
Copy File
Source Local:1:I.Data[231]
Dest MCM.InStat.Port1CurErr
Length 4

The next two rungs of ladder logic are used to handle the receipt of the slave node status data. These blocks are requested by the processor in the WriteData task and sent from the module to the processor. The two rungs below display the logic required to process these blocks.



3

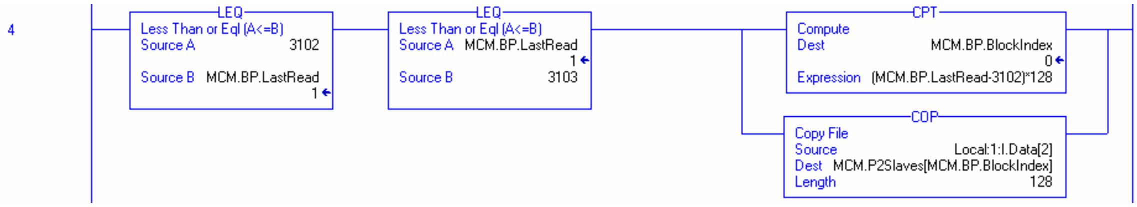
LEQ
Less Than or Eql (A<=B)
Source A 3002
Source B MCM.BP.LastRead
1

LEQ
Less Than or Eql (A<=B)
Source A MCM.BP.LastRead
1
Source B 3003

CPT
Compute
Dest MCM.BP.BlockIndex
0
Expression (MCM.BP.LastRead-3002)*128

COP
Copy File
Source Local:1:I.Data[2]
Dest MCM.P1Slaves[MCM.BP.BlockIndex]
Length 128

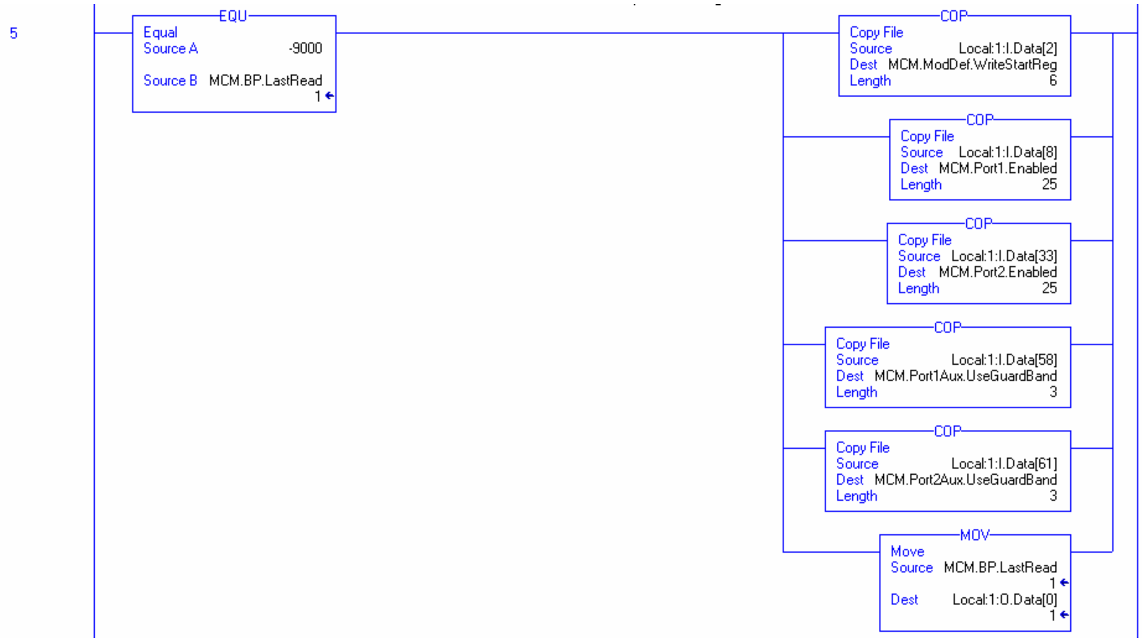
This rung is used to handle slaves attached to the Modbus Port 1. Two blocks of 128 slaves each are processed by the rung and the data is stored in the proper array location.



```
4
LEQ
Less Than or Eql (A<=B)
Source A              3102
Source B   MCM.BP.LastRead
                         1

LEQ
Less Than or Eql (A<=B)
Source A   MCM.BP.LastRead
                         1
Source B              3103

CPT
Compute
Dest                  MCM.BP.BlockIndex
                                      0
Expression   (MCM.BP.LastRead-3102)*128

COP
Copy File
Source                     Local:1:I.Data[2]
Dest   MCM.P2Slaves[MCM.BP.BlockIndex]
Length                                   128
```

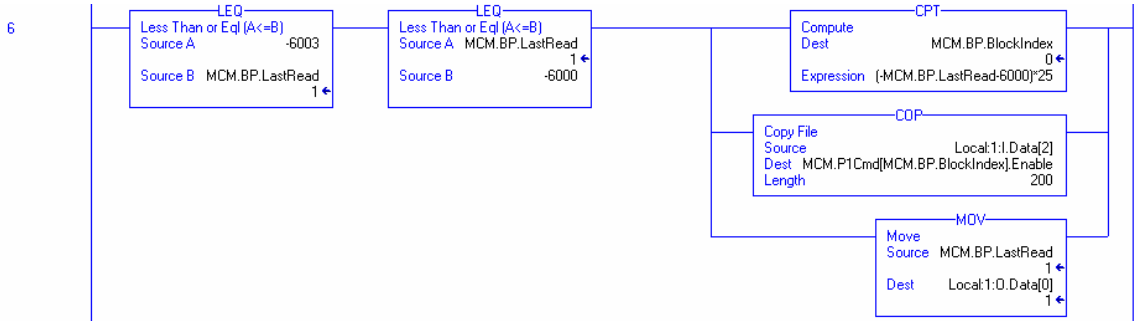
This rung is used to handle slaves attached to the Modbus Port 2.

If the processor is to receive the module's configuration from a remote source through the module's database, it must be programmed to handle special blocks. The configuration information is transferred from the module to the processor through –9000, -6000 to –6003 and –6100 to –6103 blocks. Ladder logic to handle this function is displayed in the following example.



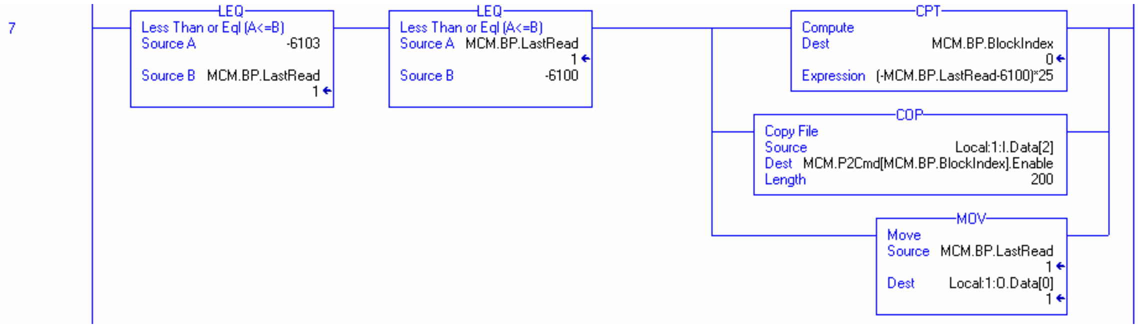
```
5
EQU
Equal
Source A             -9000
Source B   MCM.BP.LastRead
                         1

COP
Copy File
Source             Local:1:I.Data[2]
Dest   MCM.ModDef.WriteStartReg
Length                             6

COP
Copy File
Source   Local:1:I.Data[8]
Dest   MCM.Port1.Enabled
Length                  25

COP
Copy File
Source  Local:1:I.Data[33]
Dest   MCM.Port2.Enabled
Length                  25

COP
Copy File
Source            Local:1:I.Data[58]
Dest   MCM.Port1Aux.UseGuardBand
Length                             3

COP
Copy File
Source            Local:1:I.Data[61]
Dest   MCM.Port2Aux.UseGuardBand
Length                             3

MOV
Move
Source   MCM.BP.LastRead
                       1
Dest      Local:1:O.Data[0]
                       1
```

This rung is used to process the receipt of general configuration information for the module.



```
6
LEQ
Less Than or Eql (A<=B)
Source A              -6003
Source B  MCM.BP.LastRead
                          1

LEQ
Less Than or Eql (A<=B)
Source A  MCM.BP.LastRead
                          1
Source B              -6000

CPT
Compute
Dest            MCM.BP.BlockIndex
                                0
Expression  (-MCM.BP.LastRead-6000)*25

COP
Copy File
Source                  Local:1:I.Data[2]
Dest   MCM.P1Cmd[MCM.BP.BlockIndex].Enable
Length                               200

MOV
Move
Source  MCM.BP.LastRead
                      1
Dest    Local:1:O.Data[0]
                      1
```

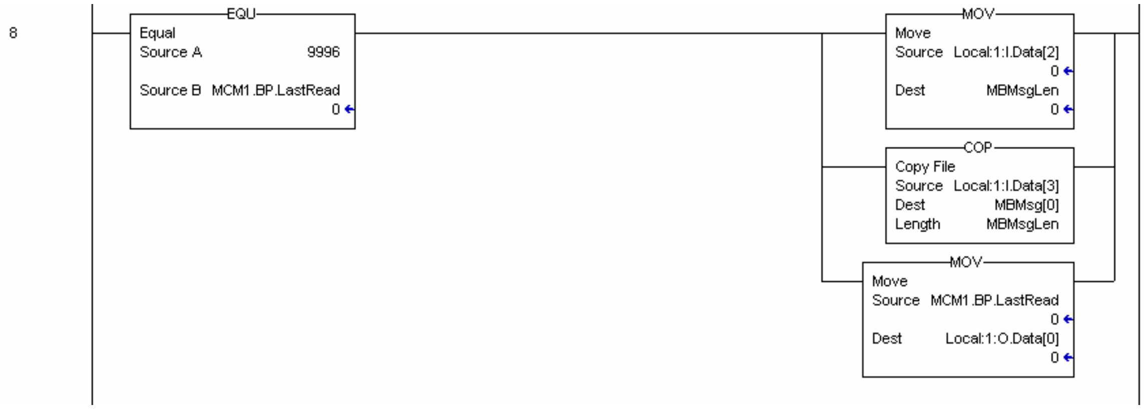
This rung is used to handle the receipt of master command list data for the Modbus Port 1.



```
7
LEQ
Less Than or Eql (A<=B)
Source A              -6103
Source B  MCM.BP.LastRead
                          1

LEQ
Less Than or Eql (A<=B)
Source A  MCM.BP.LastRead
                          1
Source B              -6100

CPT
Compute
Dest            MCM.BP.BlockIndex
                                0
Expression  (-MCM.BP.LastRead-6100)*25

COP
Copy File
Source                  Local:1:I.Data[2]
Dest   MCM.P2Cmd[MCM.BP.BlockIndex].Enable
Length                               200

MOV
Move
Source  MCM.BP.LastRead
                      1
Dest    Local:1:O.Data[0]
                      1
```

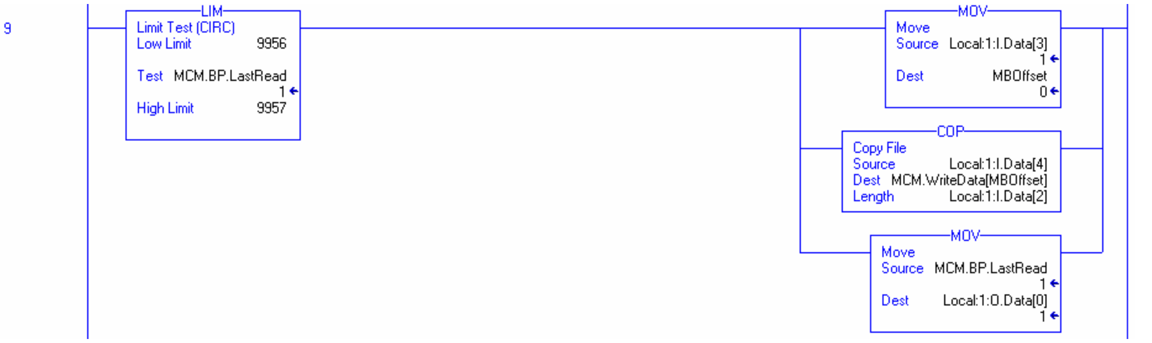
This rung is used to handle the receipt of master command list data for the Modbus Port 2. Other blocks may have to be handled in future upgrades of the product.

The following rung is used to copy a message passed to the processor directly from the remote host through the module (pass-through mode).



```
8
EQU
Equal
Source A               9996
Source B  MCM1.BP.LastRead
                          0

MOV
Move
Source  Local:1:I.Data[2]
                        0
Dest            MBMsgLen
                        0

COP
Copy File
Source  Local:1:I.Data[3]
Dest            MBMsg[0]
Length          MBMsgLen

MOV
Move
Source  MCM1.BP.LastRead
                       0
Dest   Local:1:O.Data[0]
                       0
```

This rung is used to handle Function Code 6 and 16 requests when the module is being used in formatted Pass-Thru mode.



9

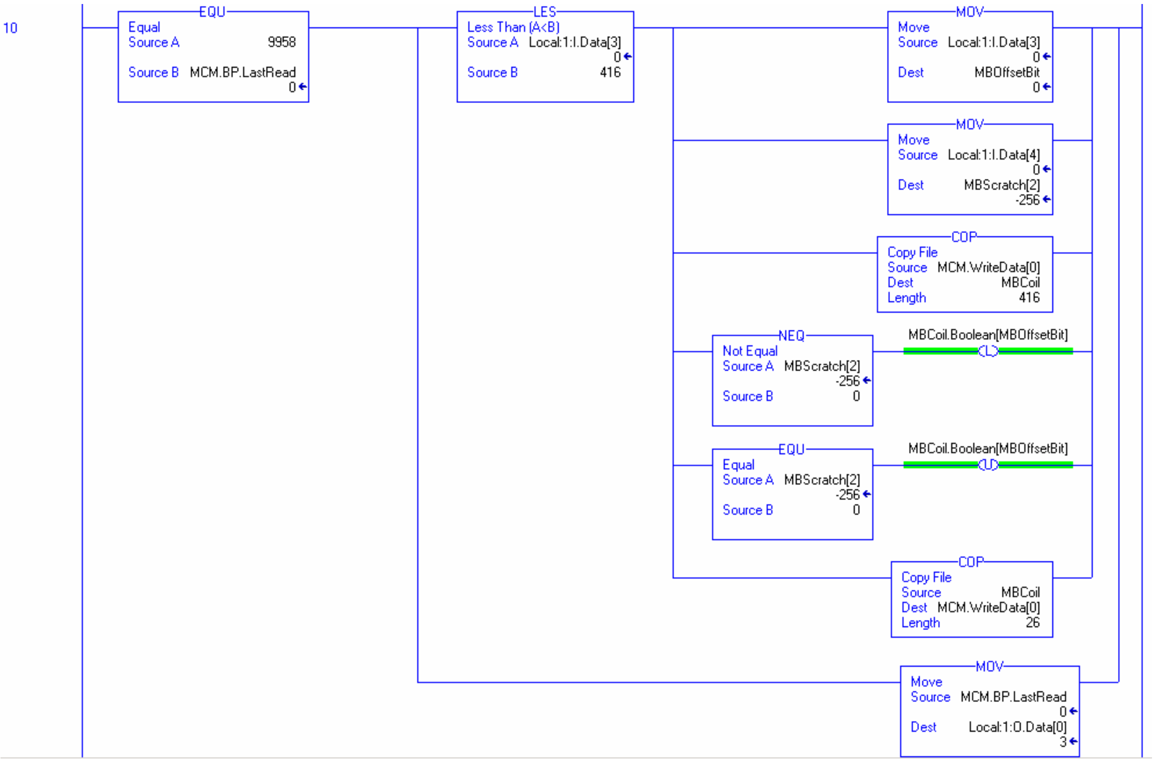
LIM
Limit Test (CIRC)
Low Limit 9956
Test MCM.BP.LastRead
1
High Limit 9957

MOV
Move
Source Local:1:I.Data[3]
1
Dest MBOffset
0

COP
Copy File
Source Local:1:I.Data[4]
Dest MCM.WriteData[MBOffset]
Length Local:1:I.Data[2]

MOV
Move
Source MCM.BP.LastRead
1
Dest Local:1:O.Data[0]
1

This rung is used to handle Function Code 5 requests when the module is being used as a slave in formatted Pass-Thru mode:



10

EQU
Equal
Source A 9958
Source B MCM.BP.LastRead
0

LES
Less Than (A<B)
Source A Local:1:I.Data[3]
0
Source B 416

MOV
Move
Source Local:1:I.Data[3]
0
Dest MBOffsetBit
0

MOV
Move
Source Local:1:I.Data[4]
0
Dest MBScratch[2]
-256

COP
Copy File
Source MCM.WriteData[0]
Dest MBCoil
Length 416

NEQ
Not Equal
Source A MBScratch[2]
-256
Source B 0

MBCoil.Boolean[MBOffsetBit]
L

EQU
Equal
Source A MBScratch[2]
-256
Source B 0

MBCoil.Boolean[MBOffsetBit]
U

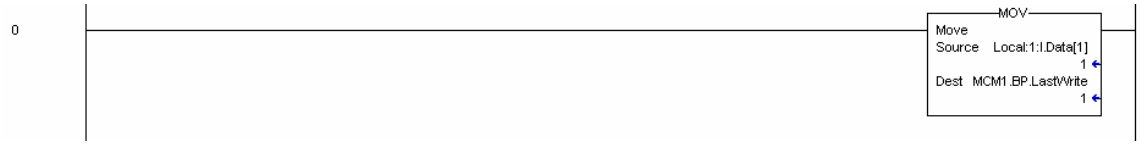
COP
Copy File
Source MBCoil
Dest MCM.WriteData[0]
Length 26

MOV
Move
Source MCM.BP.LastRead
0
Dest Local:1:O.Data[0]
3

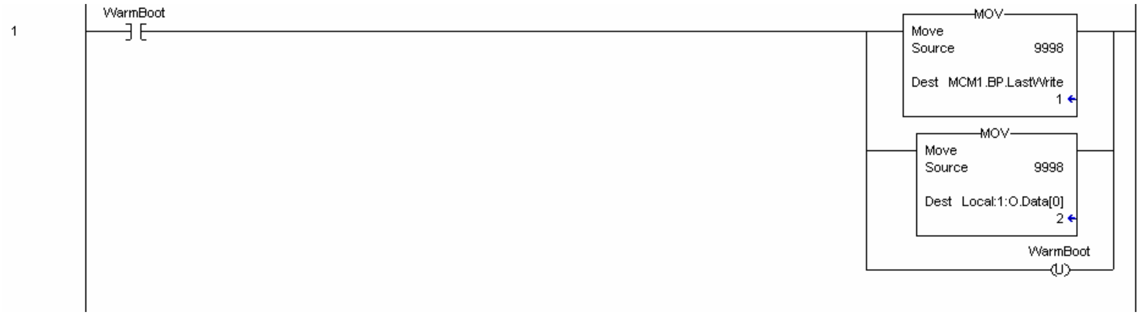
Note: Rung 11 (not shown) is used for the pass-thru logic for function code 15. This rung should NOT be altered.

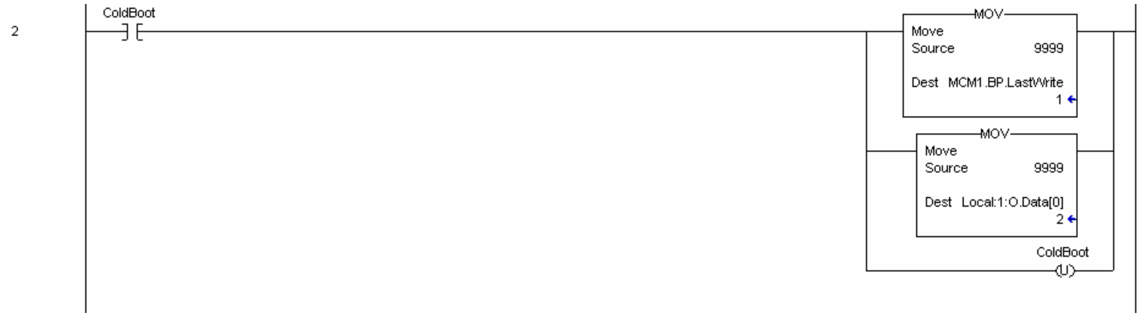
4.4 WriteData

The WriteData task is responsible for sending data from the processor to the MVI56-MCM module. Data is transferred from the processor to the module using the module's output image (**Local:1:O:Data[ ]**). The first rung is used to store the currently requested data set in the module's **MCM.BP.LastWrite** data object. This object is used in all subsequent ladder logic in case the input word (**Local:1:I:Data[1]**) changes during processing.

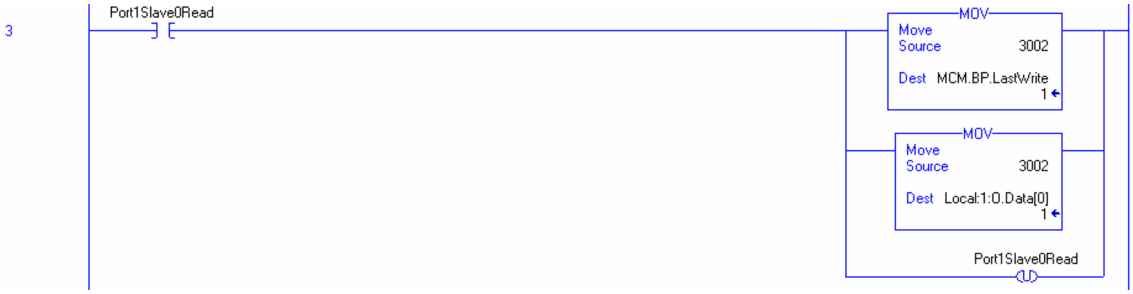


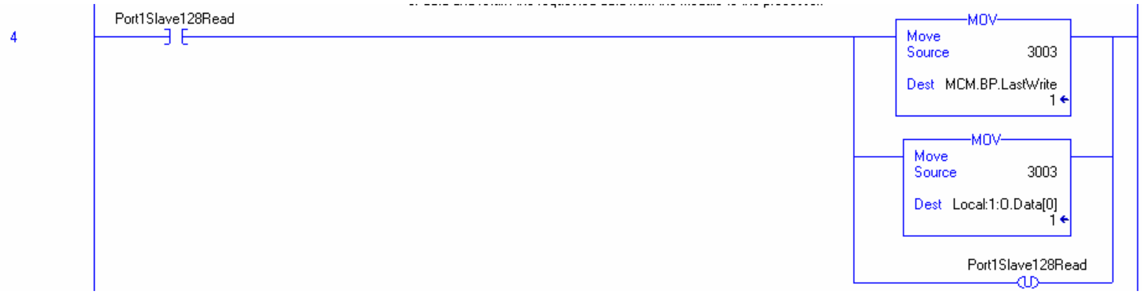
The next two rungs are used to handle processor control of the module using the warm- and cold- boot control block numbers. When the processor requires the module to perform one of these operations, it simply copies the block number into the output image of the module and the module will perform the operation. Be certain to set the required block number in the last write object to prevent further processing in the WriteData task. Examples of each control block are provided in the following rungs.



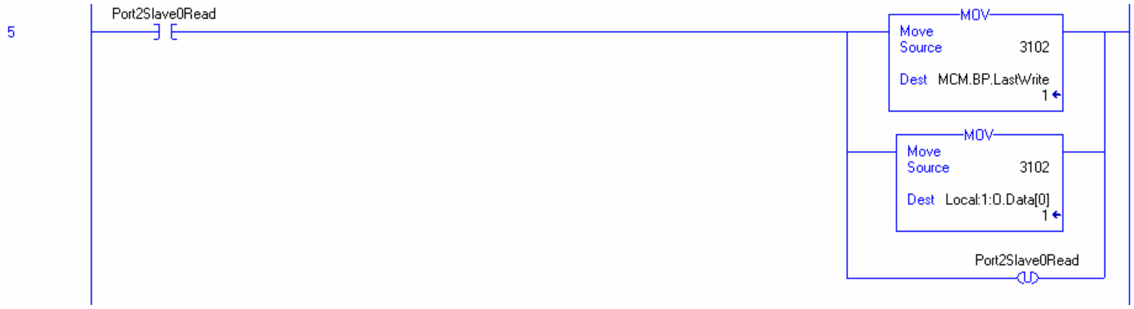


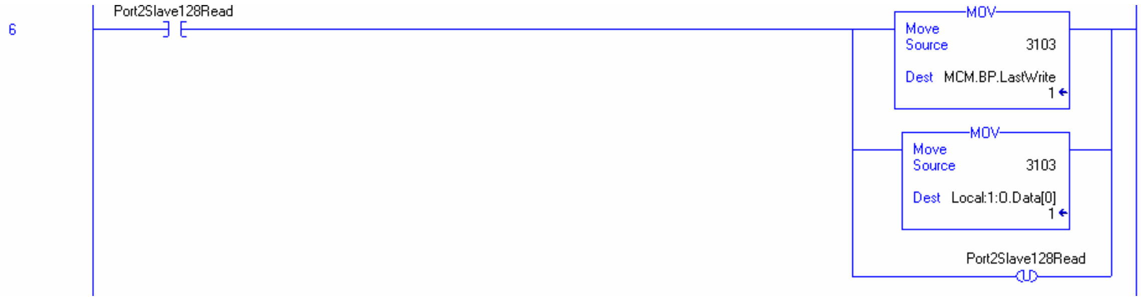
The next four rungs are used to request the slave node status data associated with each master port. Two requests are required for each port in order to obtain the data for the potential 256-slave addresses on a port. The following ladder logic displays what is required to obtain the data for the Modbus Port 1.



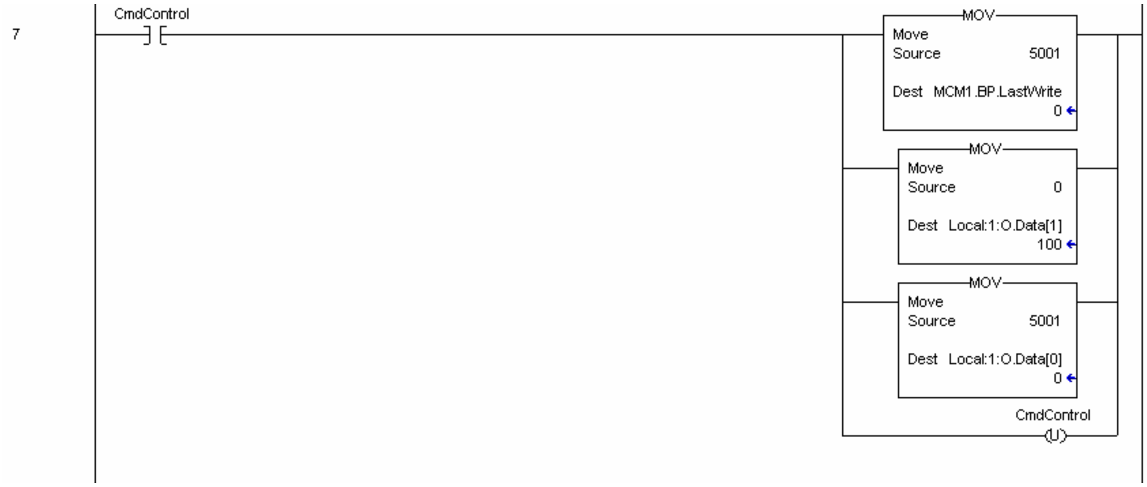


The next two rungs display the logic for the Modbus Port 2 slave status/control data.



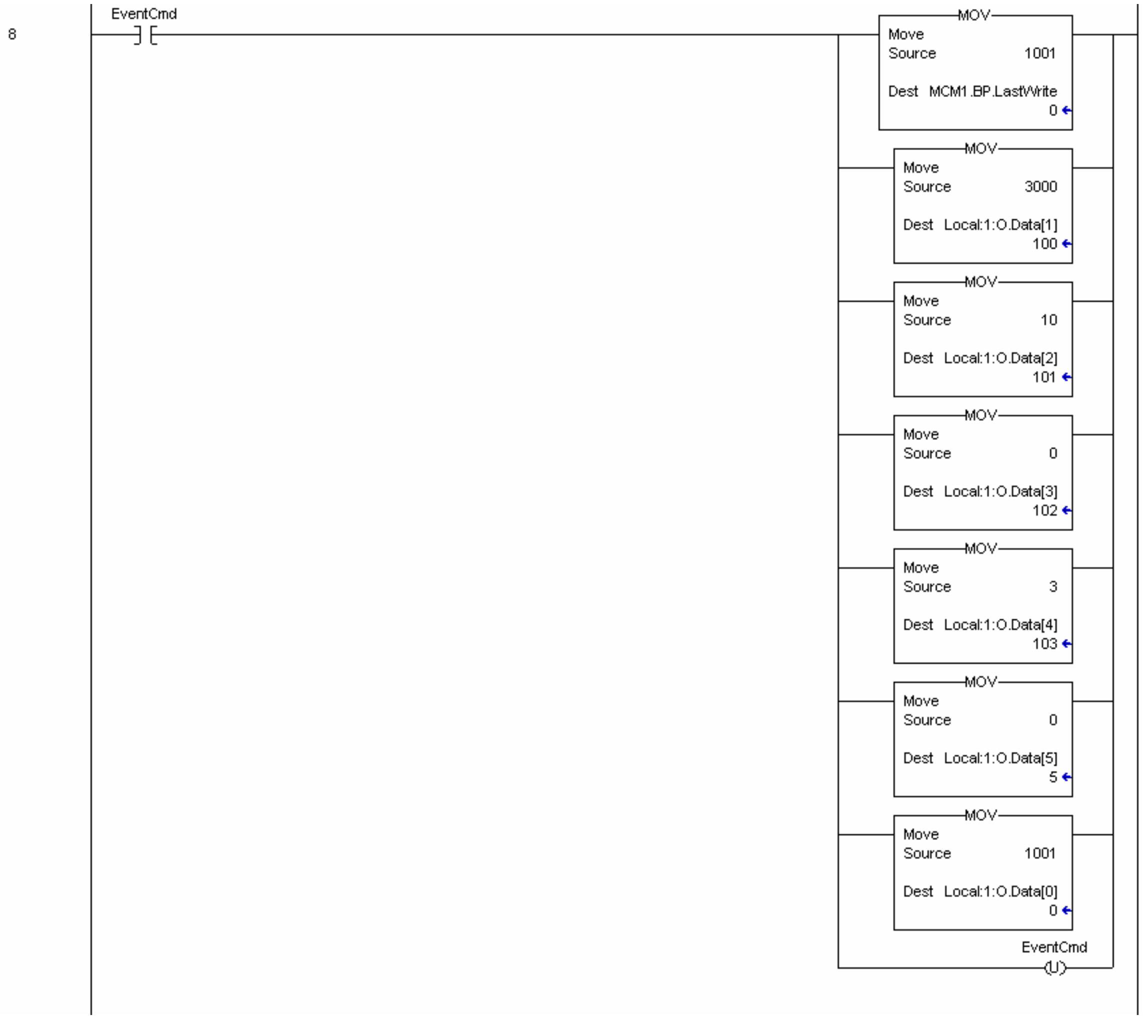


The next rung displays an example of command control. This block of data is passed from the processor to the module to execute a command in a master port's command list.



When the CmdControl bit is set, the Port 1 master command (index 0) will be placed in the command queue and executed. Up to six commands can be transferred from the command list to the command queue with one request.

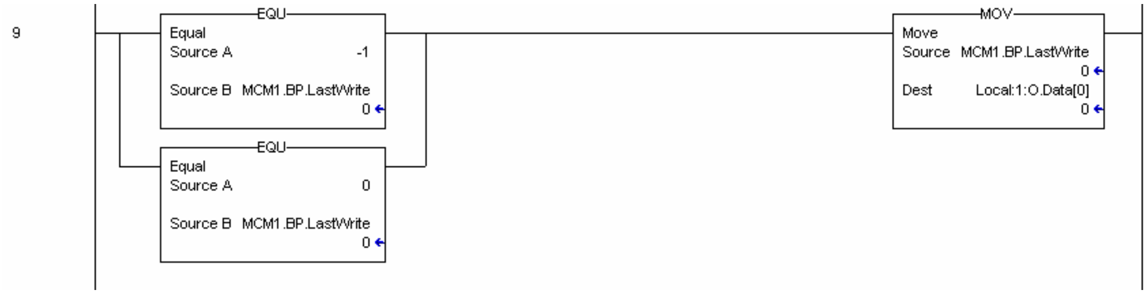
The next rung is used to issue an event message (user constructed message) on Port 1.



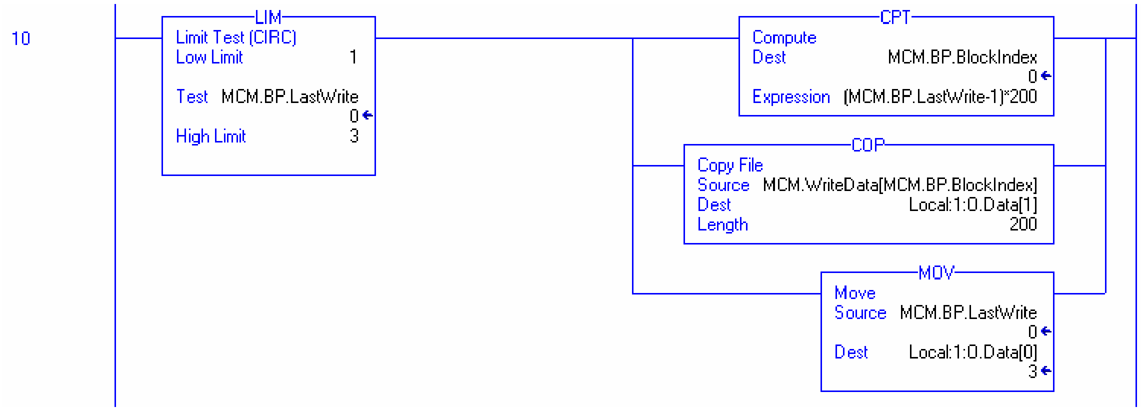
```
8   EventCmd
    ] [
        MOV  Move  Source 1001  Dest MCM1.BP.LastWrite  0
        MOV  Move  Source 3000  Dest Local:1:O.Data[1]  100
        MOV  Move  Source 10    Dest Local:1:O.Data[2]  101
        MOV  Move  Source 0     Dest Local:1:O.Data[3]  102
        MOV  Move  Source 3     Dest Local:1:O.Data[4]  103
        MOV  Move  Source 0     Dest Local:1:O.Data[5]  5
        MOV  Move  Source 1001  Dest Local:1:O.Data[0]  0
        EventCmd
        (U)
```

When the EventCmd bit is set, the rung will execute. It places the command contained in the rung in the command queue for execution. This technique can be used to issue commands on a port without constructing a master command list, or to execute commands that are to be issued under special conditions (e.g., a reset command that must be executed once a day, week, etc.).

If the module is configured for no block transfer, or only one block transfer, special processing is required. The module must observe the first word of the module's output image changing in order to recognize the receipt of new data. If the value never changes, the module will not process the data. This presents a problem when fewer than two blocks are to be transferred to the module from the processor. To overcome this problem, the module will send -1 and 0 in the input word. When the module is configured for zero write blocks, the following block request sequence will be present: -1, 0, -1, 0... When the module is configured for one write block, the following block request sequence will be present: 1, 0, 1, 0, 1, 0... The following rung is required to handle these conditions.

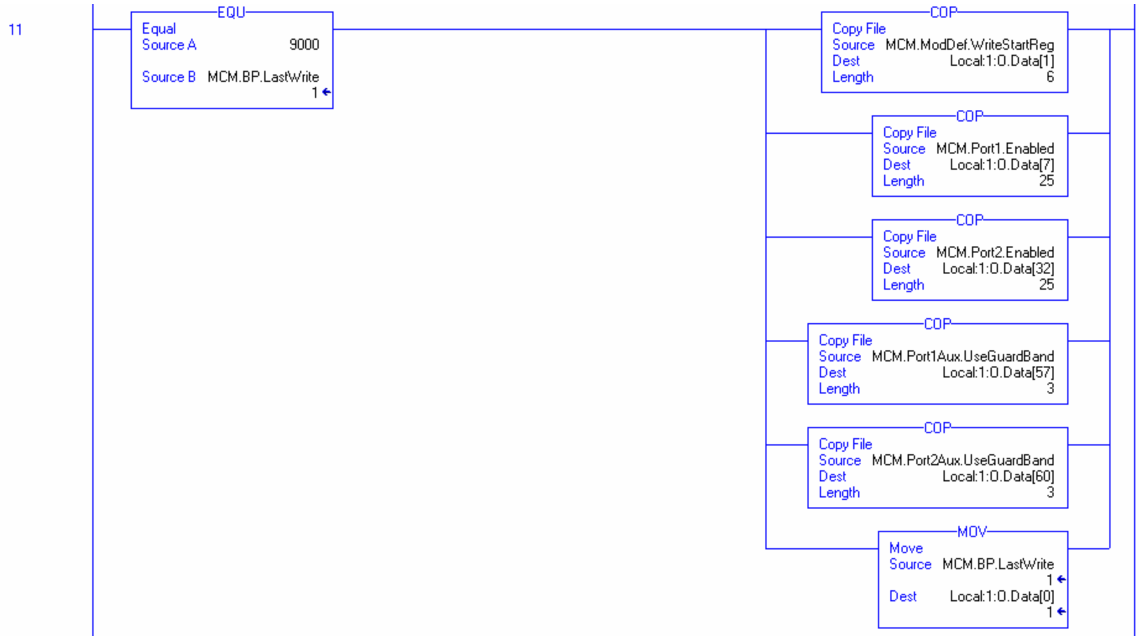


The next rung in the ladder logic is the most important. It handles the transfer of processor data to the module. Up to 200 words of user data held in the processor (**MCM.WriteData[ ]**) can be transferred to the module at one time.



In order to configure the module, the configuration must be transferred from the processor data table to the module. Several blocks are required to transfer all the information required by the module. Each of these blocks must be programmed and handled for the module to run.

The first configuration block has a code value of 9000. This block is used to transfer the data block size information (**MCM.ModDef**) and the Modbus port configuration information (**MCM.Port[ ]**). This is the first data set requested by the module when it boots.



11

EQU
Equal
Source A 9000
Source B MCM.BP.LastWrite
1 ←

COP
Copy File
Source MCM.ModDef.WriteStartReg
Dest Local:1:O.Data[1]
Length 6

COP
Copy File
Source MCM.Port1.Enabled
Dest Local:1:O.Data[7]
Length 25

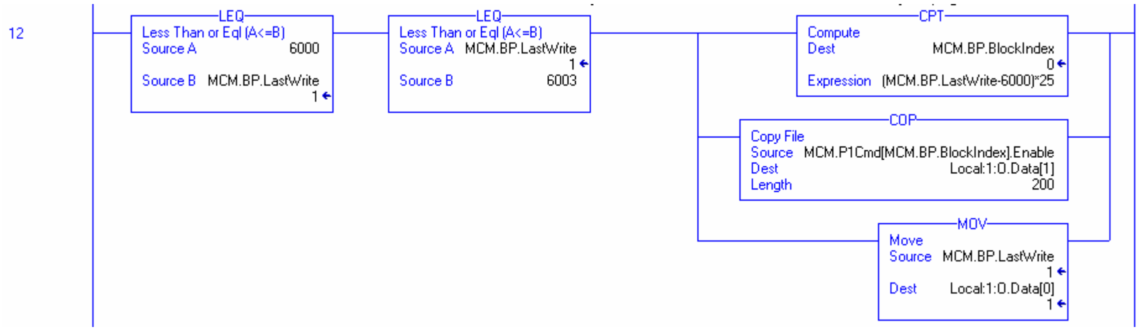
COP
Copy File
Source MCM.Port2.Enabled
Dest Local:1:O.Data[32]
Length 25

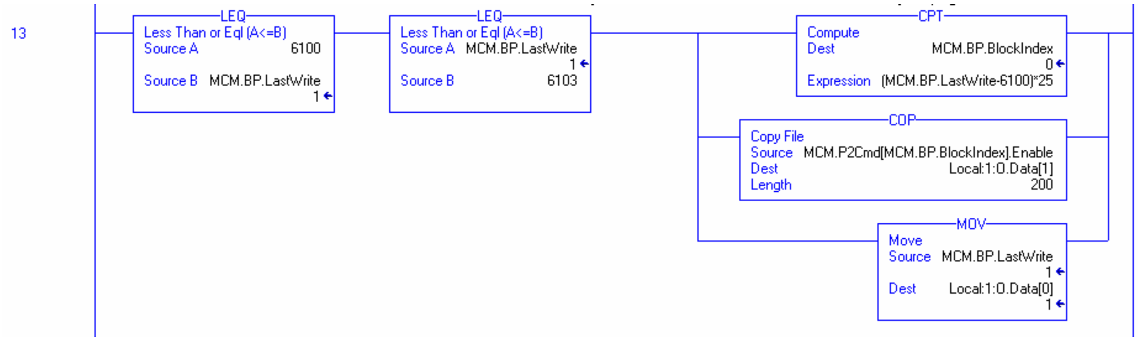
COP
Copy File
Source MCM.Port1Aux.UseGuardBand
Dest Local:1:O.Data[57]
Length 3

COP
Copy File
Source MCM.Port2Aux.UseGuardBand
Dest Local:1:O.Data[60]
Length 3

MOV
Move
Source MCM.BP.LastWrite
1 ←
Dest Local:1:O.Data[0]
1 ←

The last set of configuration information required from the module is the master command list for each port. This list is transferred to the module 25 commands at one time. The ladder logic to transfer the command list to the module is shown in the following rungs:





5 Diagnostics and Troubleshooting

In This Chapter

The module provides diagnostic information in three forms to the user. 1) Status Data values are transferred from the module to the data files defined in the ControlLogix processor. 2) All data contained in the module can be viewed through the configuration/debug port to an attached terminal emulator. 3) LED status indicators on the front of the module yield information on the modules status.

The following sections explain how to obtain the Status Data from the module and the meaning of the individual LED's on the module.

5.1 Reading Status Data From the Module

The MVI56-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 ControlLogix processor continuously with each read block. For a complete listing of the status data object, refer to the **Module Set Up** section.

5.1.1 Required Hardware

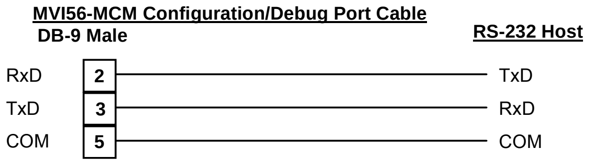
The hardware requirements to interface with the configuration/debugger port are not too stringent. A personal computer with a standard serial port should suffice. For optimal performance, the minimum is required:

80468 based processor (Pentium preferred)

1 megabyte of memory

At least one serial communications port available

Additionally, a null-modem cable is shipped with the module to provide a connection between your PC and the port. The module's port has a DB-9 male connector at the end of a RJ-45 to DB-9 pigtail. The RJ-45 end of the cable is to be placed in the MVI56-MCM port 1 connector (top port). The cable required is displayed in the following diagram:



5.1.2 Required Software

The software required on your personal computer to interface with the configuration/debugger port is operating system dependent. Tested software includes the following:

DOS	ProComm, PS-Term and several other terminal emulation programs
Windows 3.1	Terminal
Windows 95/98	HyperTerminal and PS-Term
Windows NT/2000/XP	HyperTerminal
Linux	Minicom

Any ASCII terminal emulation software package provided with your operating system should work as long as it can be configured as follows:

Baud Rate	57,600
Parity	None
Data Bits	8
Stop Bits	1
File Transfer Protocol	Zmodem

5.1.3 Using the Port

The following steps are required to interface with the configuration/debugger port:

1 Connect your computer to the module's port using a null-modem cable.

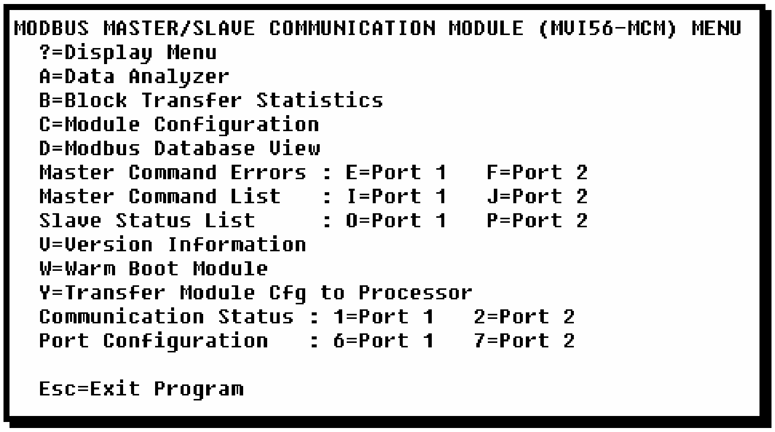
2 Start the terminal emulation program on your computer and configure the communication parameters to those shown in the Required Software section (57600K, N, 8, 1).

3 Enter the '?' character on your computer. If everything is set up correctly, the port's menu will be displayed.

If there is no response from the module, check the communication setup and the cable. In addition, make sure you are connected to the correct port on your computer and the module.

5.1.4 Menu Options

Features available through the use of the configuration/debug port on the MVI56-MCM module are all reached using single keystrokes on your computer. There is a single main menu and several sub-menus presented on the port. To view the current selections available, press the '?' key on your computer. If you are in main menu mode, the following menu appears:

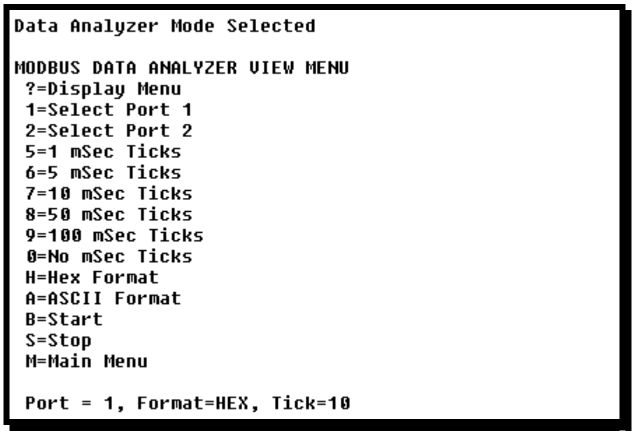


```
MODBUS MASTER/SLAVE COMMUNICATION MODULE (MVI56-MCM) MENU
  ?=Display Menu
  A=Data Analyzer
  B=Block Transfer Statistics
  C=Module Configuration
  D=Modbus Database View
  Master Command Errors : E=Port 1   F=Port 2
  Master Command List   : I=Port 1   J=Port 2
  Slave Status List     : O=Port 1   P=Port 2
  V=Version Information
  W=Warm Boot Module
  Y=Transfer Module Cfg to Processor
  Communication Status : 1=Port 1   2=Port 2
  Port Configuration   : 6=Port 1   7=Port 2

  Esc=Exit Program
```

If this menu is not displayed, press the 'M' key to display the main menu. All facilities offered by the configuration/debugger are shown on the Main menu. Each option is discussed in the following sections.

A=Data Analyzer

Selection of this menu option places the program in analyzer menu mode. This mode of operation is used to display Modbus messages generated and received by the module. To view the menu options available in this mode, press the '?' key and the following menu appears:



```
Data Analyzer Mode Selected

MODBUS 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
```

This tool is extremely useful in determining the operation of the module and nodes on the network of each port. The parameters at the bottom of the display show the current analyzer settings. Each of the menu options is discussed in the following sections.

1=Select Port 1

This option selects Modbus Port 1 for analysis. Data displayed when in analyzer mode will relate to this port.

2=Select Port 2

This option selects Modbus Port 2 for analysis. Data displayed when in analyzer mode will relate to this port.

5=1 mSec Ticks

This option generates 1-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

6=5 mSec Ticks

This option generates 5-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

7=10 mSec Ticks

This option generates 10-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

8=50 mSec Ticks

This option generates 50-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

9=100 mSec Ticks

This option generates 100-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

0=No mSec Ticks

This option turns the display of timing marks off.

H=Hex Format

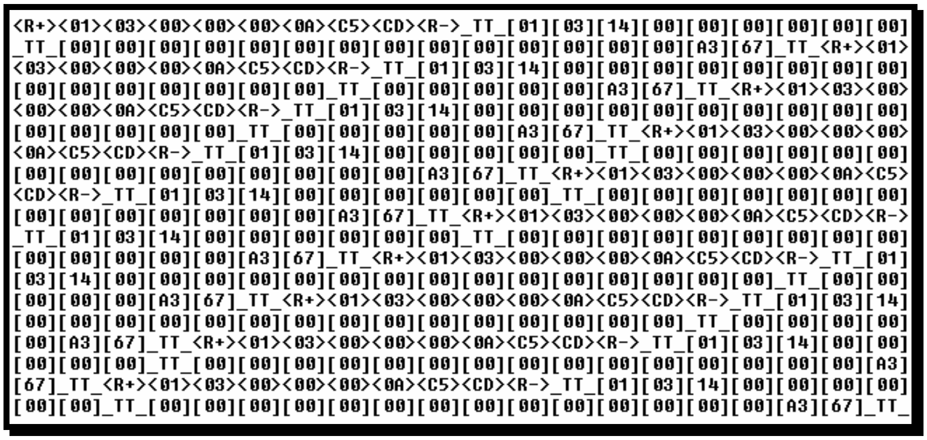
This option selects the display of the data in hexadecimal format. This format is most useful when viewing Modbus RTU protocol messages.

A=ASCII Format

This option selects the display of the data in ASCII format. This format is most useful when viewing Modbus ASCII protocol messages.

B=Start

This option starts the data analyzer. After the key is pressed, all data transmitted and received on the currently selected port is displayed. An example display is shown:



```
<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01][03][14][00][00][00][00][00][00]
_TT_[00][00][00][00][00][00][00][00][00][00][00][00][00][00][A3][67]_TT_<R+><01>
<03><00><00><00><0A><C5><CD><R->_TT_[01][03][14][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00]_TT_[00][00][00][00][00][A3][67]_TT_<R+><01><03><00>
<00><00><0A><C5><CD><R->_TT_[01][03][14][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00]_TT_[00][00][00][00][00][A3][67]_TT_<R+><01><03><00><00><00>
<0A><C5><CD><R->_TT_[01][03][14][00][00][00][00][00]_TT_[00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5>
<CD><R->_TT_[01][03][14][00][00][00][00][00][00]_TT_[00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->
_TT_[01][03][14][00][00][00][00][00][00]_TT_[00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01]
[03][14][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]_TT_[00][00]
[00][00][00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01][03][14]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]_TT_[00][00][00][00]
[00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01][03][14][00][00]
[00][00][00]_TT_[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][A3]
[67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01][03][14][00][00][00][00]
[00][00]_TT_[00][00][00][00][00][00][00][00][00][00][00][00][00][00][A3][67]_TT_
```

The following table describes the special characters used in the display:

[ ]	Data enclosed in these characters represent data received on the port.
< >	Data enclosed in these characters represent data transmitted on the port.
<R+>	These characters are inserted when the RTS line is driven high on the port.
<R->	These characters are inserted when the RTS line is dropped low on the port.
<CS>	These characters are displayed when the CTS line is recognized high.
_TT_	These characters are displayed when the timing mark interval has been reached. This parameter is user defined.

S=Stop

This option stops the analyzer. Use this option to freeze the display so the data can be analyzed. To restart the analyzer, press the 'B' key.

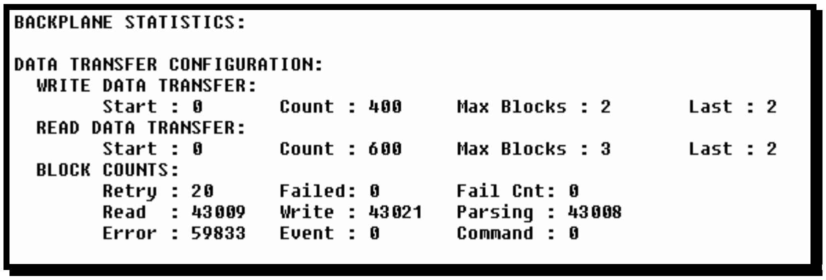
Warning -- When in analyzer mode, program execution will slow down. Only use this tool during a trouble-shooting session. Disable the analyzer before leaving the module to run in its normal mode.

M = Main Menu

This menu option is used to return to the main menu mode.

B=Block Transfer Statistics

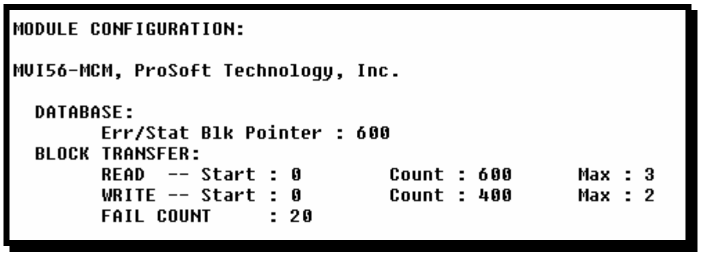
This menu option displays the configuration and statistics of the backplane data transfer operations. After selecting this option, the following is displayed. Selecting this option at one-second intervals can be used to determine the number of blocks transferred each second.



```
BACKPLANE STATISTICS:

DATA TRANSFER CONFIGURATION:
  WRITE DATA TRANSFER:
        Start : 0       Count : 400     Max Blocks : 2      Last : 2
  READ DATA TRANSFER:
        Start : 0       Count : 600     Max Blocks : 3      Last : 2
  BLOCK COUNTS:
        Retry : 20      Failed: 0       Fail Cnt: 0
        Read  : 43009   Write : 43021   Parsing : 43008
        Error : 59833   Event : 0       Command : 0
```

C=Module Configuration

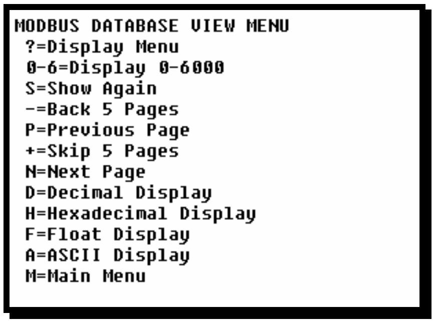
This option displays the general module configuration information for the MVI56-MCM module. After selecting the option, the following screen appears.



```
MODULE CONFIGURATION:

MVI56-MCM, ProSoft Technology, Inc.

  DATABASE:
        Err/Stat Blk Pointer : 600
  BLOCK TRANSFER:
        READ  -- Start : 0        Count : 600      Max : 3
        WRITE -- Start : 0        Count : 400      Max : 2
        FAIL COUNT     : 20
```

D=Modbus Database View

Selecting this option places the program in database view menu mode. This mode of operation displays the module's internal database values. To view the menu options available in this mode, press the '?' key and the following menu appears.



```
MODBUS DATABASE VIEW MENU
 ?=Display Menu
 0-6=Display 0-6000
 S=Show Again
 -=Back 5 Pages
 P=Previous Page
 +=Skip 5 Pages
 N=Next Page
 D=Decimal Display
 H=Hexadecimal Display
 F=Float Display
 A=ASCII Display
 M=Main Menu
```

All data contained in the module's database is available for viewing using the menu options. Each option available on the menu is discussed in the following sections.

0-9 Register Pages 0-9000

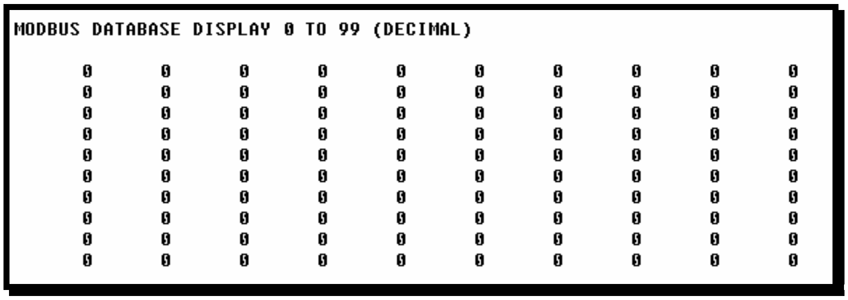
This menu option jumps to a specific set of registers in the database and displays the data. The keys perform the following functions.

Key	FUNCTION
0	Display registers 0 to 99
1	Display registers 1000 to 1099
2	Display registers 2000 to 2099
3	Display registers 3000 to 3099
4	Display registers 4000 to 4099

Key	FUNCTION
5	Display registers 5000 to 5099
6	Display registers 6000 to 6099
7	Display registers 7000 to 7099
8	Display registers 8000 to 8099
9	Display registers 9000 to 9099

S=Show Again

This menu option displays the current page of 100 registers in the database. Example output of the database display is shown in the following example:



```
MODBUS DATABASE DISPLAY 0 TO 99 (DECIMAL)

      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
```

- = Back 5 Pages

This menu option skips the previous 500 registers of data for viewing and displays the data.

P = Previous Page

This menu option selects and displays the previous 100 registers of data.

+ = Skip 5 Pages

This menu option skips 500 registers of data and displays the new page of data.

N = Next Page

This menu option selects the next 100 registers of data for viewing and displays the data.

D = Decimal Display

This menu option displays the data on the current page in decimal format.

H = Hexadecimal Display

This menu option displays the data on the current page in hexadecimal format.

<u>F = Float Display</u>

This menu option displays 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 will not be displayed properly.

<u>A = ASCII Display</u>

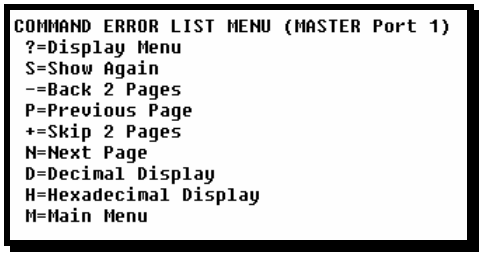
This menu option displays the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

<u>M = Main Menu</u>

This menu option returns to the main menu mode.

<u>E and F=Master Command Errors (Ports 1 and 2)</u>

Selecting these menu options places the program in master command error menu mode for the specified port. This mode of operation displays multiple pages of master command list error/status data. To view the menu options available in this mode, press the '?' key and the following menu appears.

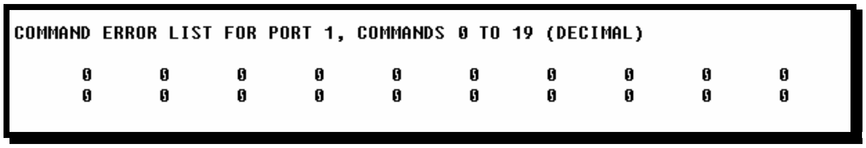


```
COMMAND ERROR LIST MENU (MASTER Port 1)
 ?=Display Menu
 S=Show Again
 -=Back 2 Pages
 P=Previous Page
 +=Skip 2 Pages
 N=Next Page
 D=Decimal Display
 H=Hexadecimal Display
 M=Main Menu
```

Each menu option is discussed in the following sections:

<u>S = Show Again</u>

This option displays the current page of master command error/status data. After selecting the option, the following screen appears.



```
COMMAND ERROR LIST FOR PORT 1, COMMANDS 0 TO 19 (DECIMAL)

      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
```

Each value displayed on the screen corresponds to the error/status code for the associated master command list index. Refer to the **Module Set Up** section for complete listing and interpretation of the codes displayed.

- = Back 2 Pages

This option skips back 20 commands and displays the data.

P = Previous Page

This option displays the previous page of data.

+ = Skip 2 Pages

This option skips past the next 20 commands and displays the data.

N = Next Page

This option displays the next page of master command list error/status data.

D = Decimal Display

This option changes the display of the data to decimal format.

H = Hexadecimal Display

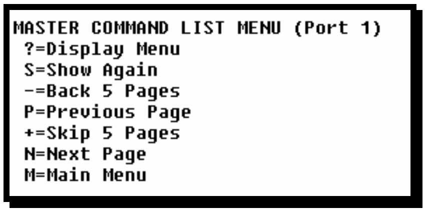
This option changes the display of error/status data to hexadecimal format.

M = Main Menu

This option returns the program to main menu mode.

I and J=Master Command List (Ports 1 and 2)

Selecting these menu options places the program in master command list menu mode for the specified port. This mode of operation is used to display multiple pages of master command list data. To view the menu options available in this mode, press the '?' key and the following menu appears.

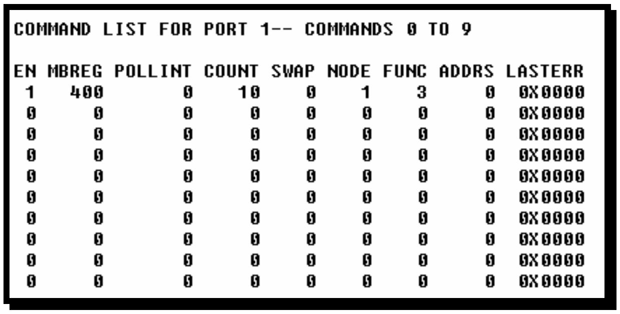


```
MASTER COMMAND LIST MENU (Port 1)
 ?=Display Menu
 S=Show Again
 -=Back 5 Pages
 P=Previous Page
 +=Skip 5 Pages
 N=Next Page
 M=Main Menu
```

Each option on the menu is discussed in the following sections:

S = Show Again

This option displays the current page of master commands. Ten commands are displayed on each page as shown in the following example:



```
COMMAND LIST FOR PORT 1-- COMMANDS 0 TO 9

EN MBREG POLLINT COUNT SWAP NODE FUNC ADDRS LASTERR
 1   400       0    10    0    1    3     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
```

- = Back 5 Pages

This menu option displays the master command list data after skipping the previous 50 commands.

P = Previous Page

This menu option displays the previous page of master command list data.

+ = Skip 5 Pages

This menu option displays the master command list data after skipping the next 50 commands.

N = Next Page

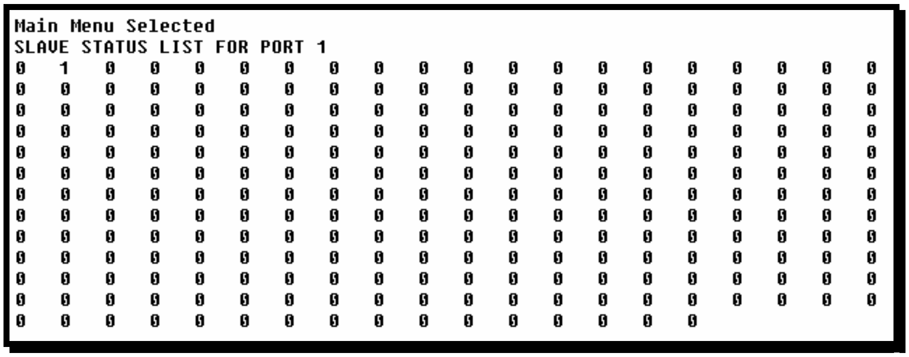
This menu option displays the next page of master command list data.

M = Main Menu

This option returns to the main menu mode of operation.

O and P=Slave Status List (Port 1 and 2)

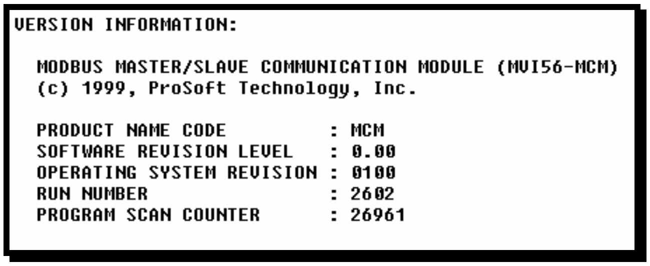
Selecting these menu options displays the 256 slave status values associated with the ports. Values shown have the following definitions: 0 = slave is not used, 1 = slave being actively polled, 2 = slave suspended and 3 = slave disabled.



```
Main Menu Selected
SLAVE STATUS LIST FOR PORT 1
0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

V=Version Information

This option displays the current version of the software for the module and other important values. After selecting the option, the following screen appears.

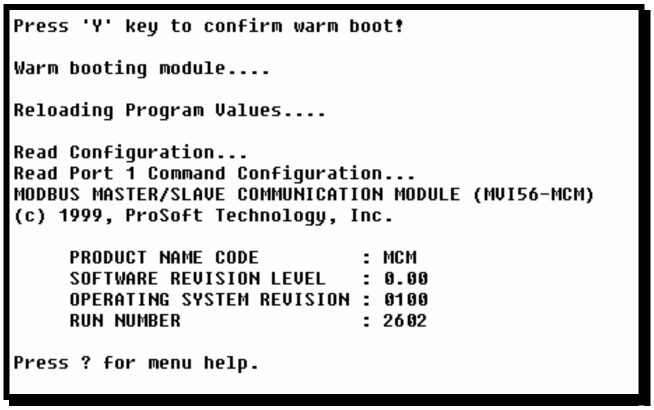


```
VERSION INFORMATION:

  MODBUS MASTER/SLAVE COMMUNICATION MODULE (MVI56-MCM)
  (c) 1999, ProSoft Technology, Inc.

  PRODUCT NAME CODE         : MCM
  SOFTWARE REVISION LEVEL   : 0.00
  OPERATING SYSTEM REVISION : 0100
  RUN NUMBER                : 2602
  PROGRAM SCAN COUNTER      : 26961
```

This information may be requested 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. This value can be used to determine the frequency of program execution by pressing the 'V' key at one-second intervals.

W=Warm Boot Module

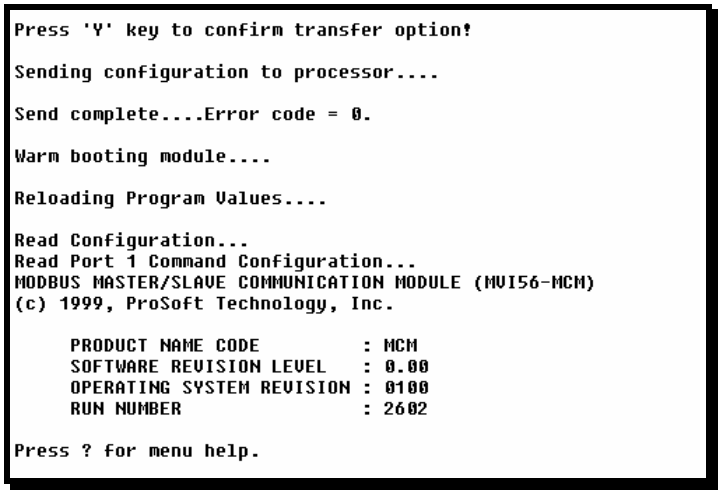
This option is usedwhen a warm-boot operation is required of the module. This request is usually made after configuration changes are set in the ControlLogix processor's Controller Tags data area to implement the changes. After selecting the option, the following screen appears.



```
Press 'Y' key to confirm warm boot!

Warm booting module....

Reloading Program Values....

Read Configuration...
Read Port 1 Command Configuration...
MODBUS MASTER/SLAVE COMMUNICATION MODULE (MVI56-MCM)
(c) 1999, ProSoft Technology, Inc.

     PRODUCT NAME CODE         : MCM
     SOFTWARE REVISION LEVEL   : 0.00
     OPERATING SYSTEM REVISION : 0100
     RUN NUMBER                : 2602

Press ? for menu help.
```

Y=Transfer Module Cfg to Processor

This option transfers the current configuration data in the module to the ControlLogix processor. Ladder logic must exist in the processor to successfully implement this option. After selecting the option, the following is displayed indicating successful operation:



```
Press 'Y' key to confirm transfer option!

Sending configuration to processor....

Send complete....Error code = 0.

Warm booting module....

Reloading Program Values....

Read Configuration...
Read Port 1 Command Configuration...
MODBUS MASTER/SLAVE COMMUNICATION MODULE (MVI56-MCM)
(c) 1999, ProSoft Technology, Inc.

     PRODUCT NAME CODE         : MCM
     SOFTWARE REVISION LEVEL   : 0.00
     OPERATING SYSTEM REVISION : 0100
     RUN NUMBER                : 2602

Press ? for menu help.
```

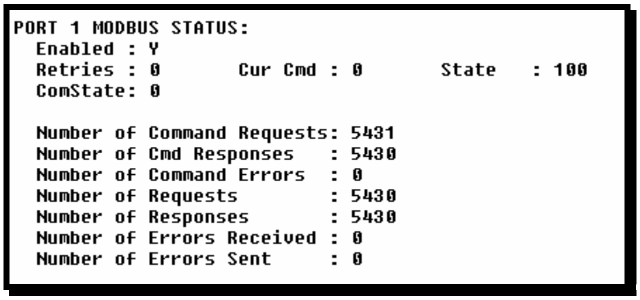
If the operation is not successful, an error code is returned. Error codes returned are as follows:

Code	Description
0	Transfer Successful
-1	Error transferring module configuration data (block –9000)
-2	Error transferring master command list data for Port 1 (blocks -6000 to -6003)
-3	Error transferring master command list data for Port 2 (blocks -6100 to -6103)

After successful data transfer, the module performs a warm boot operation to read in the new data.

1 and 2=Communication Status (Ports 1 and 2)

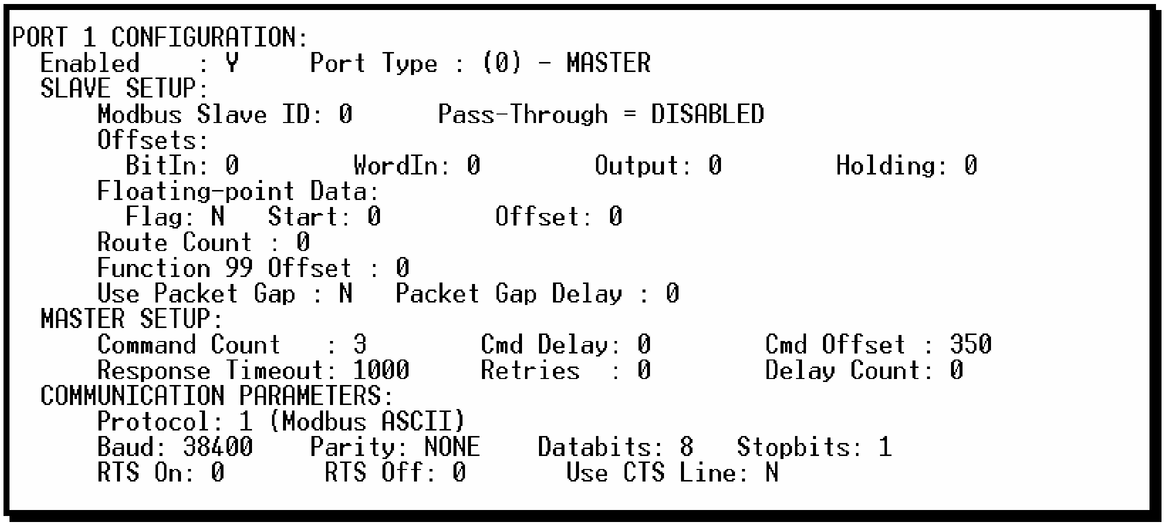
These options display the communication status and statistics of the specified Modbus port. This information can be informative when troubleshooting network problems. After selecting the option, the following information is displayed.



```
PORT 1 MODBUS STATUS:
  Enabled : Y
  Retries : 0       Cur Cmd : 0       State    : 100
  ComState: 0

  Number of Command Requests: 5431
  Number of Cmd Responses   : 5430
  Number of Command Errors  : 0
  Number of Requests        : 5430
  Number of Responses       : 5430
  Number of Errors Received : 0
  Number of Errors Sent     : 0
```

6 and 7=Port Configuration (Ports 1 and 2)

These options are used to display the configuration information for the selected Modbus port. After selecting the option, the following information will be displayed.



```
PORT 1 CONFIGURATION:
  Enabled    : Y     Port Type : (0) - MASTER
  SLAVE SETUP:
      Modbus Slave ID: 0      Pass-Through = DISABLED
      Offsets:
        BitIn: 0       WordIn: 0       Output: 0       Holding: 0
      Floating-point Data:
        Flag: N   Start: 0       Offset: 0
      Route Count : 0
      Function 99 Offset : 0
      Use Packet Gap : N   Packet Gap Delay : 0
  MASTER SETUP:
      Command Count   : 3       Cmd Delay: 0       Cmd Offset : 350
      Response Timeout: 1000    Retries  : 0       Delay Count: 0
  COMMUNICATION PARAMETERS:
      Protocol: 1 (Modbus ASCII)
      Baud: 38400    Parity: NONE    Databits: 8   Stopbits: 1
      RTS On: 0       RTS Off: 0       Use CTS Line: N
```

Esc=Exit Program

This option exits the program and displays the operating system prompt. This option should only be selected if instructed by the ProSoft Technical Support Group. If you select the option, the module ceases operation. Data is no longer transferred between the Modbus ports and the module and between the ControlLogix processor and the module. This might cause an upset to a currently running process.

5.2 LED Status Indicators

The LED's will indicate the module's operating status as follows:

ProSoft Module	**Color**	**Status**	**Indication**
CONFIG	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 its Modbus 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 its Modbus Port 2.
		Off	No data is being transferred on the port.

ProSoft Module	Color	Status	Indication
APP	Amber	On	The MVI56-MCM is working normally.
		Off	The MVI56-MCM module program has recognized a communication error on one of its Modbus ports.
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 the battery is not present. Replace the battery on the module.

During module configuration, the OK will be red and the APP and BP ACT LED’s will be on. If the LED’s are latched in this mode for a long period of time, check the configuration error words in the configuration request block. The structure of the block is shown in the following table:

Offset	Description	Length
0	Reserved	1
1	9000	1
2	Module Configuration Errors	1
3	Port 1 Configuration Errors	1
4	Port 2 Configuration Errors	1
5 – 248	Spare	244
249	-2 or –3	1

The bits in each configuration word are shown in the following table. The module configuration error word has the following definition:

Bit	Description	Value
0	Read block start value is greater than the database size.	0x0001
1	Read block start value is less than zero.	0x0002
2	Read block count value is less than zero.	0x0004
3	Read block count + start is greater than the database size.	0x0008
4	Write block start value is greater than the database size.	0x0010
5	Write block start value is less than zero.	0x0020
6	Write block count value is less than zero.	0x0040
7	Write block count + start is greater than the database size.	0x0080

Bit	Description	Value
8		0x0100
9		0x0200
10		0x0400
11		0x0800
12		0x1000
13		0x2000
14		0x4000
15		0x8000

The port configuration error words have the following definitions:

Bit	Description	Value
0	Type code is not valid. Enter a value from 0 (master) to 1 (slave).	0x0001
1	The float flag parameter is not valid.	0x0002
2	The float start parameter is not valid.	0x0004
3	The float offset parameter is not valid.	0x0008
4	Protocol parameter is not valid.	0x0010
5	Baud rate parameter is not valid.	0x0020
6	Parity parameter is not valid.	0x0040
7	Data bits parameter is not valid.	0x0080
8	Stop bits parameter is not valid.	0x0100
9	Slave ID is not valid.	0x0200
10	Input bit or word, output word and/or holding register offset(s) are not valid.	0x0400
11	Command count parameter is not valid.	0x0800
12	Spare	0x1000
13	Spare	0x2000
14	Spare	0x4000
15	Spare	0x8000

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

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

5.2.1 Clearing a Fault Condition

Typically, if the OK LED on the front of the module becomes illuminated red for over ten seconds, a hardware problem has been detected in the module or the program has exited. To attempt to clear the condition:

1. Remove the card from the rack and re-insert the card in the rack
2. Verify the configuration data being transferred to the module from the ControlLogix processor

If the module's OK LED does not turn green, make sure the module is inserted completely into the rack and a valid ladder program is downloaded into the processor (Processor must be in Run mode). If this does not cure the problem, contact the factory.

5.2.2 Troubleshooting

The following table is designed to assist you in troubleshooting the module. Please use this table to attempt to correct a problem. However, if you have additional questions or problems, please do not hesitate to contact us.

The entries in this section have been placed in the order in which the problems would most likely occur after powering up the module.

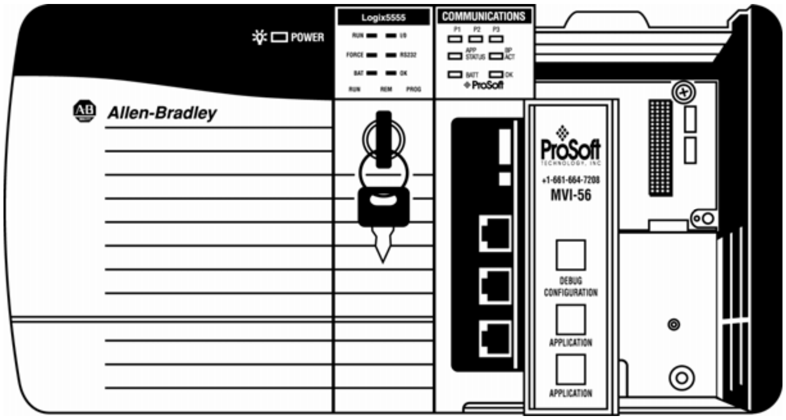
Problem Description	Steps to take
Processor Fault	Be sure that the module is plugged into the slot that has been configured for the MVI56-MCM module. Assure that the slot in the rack configuration has been set up correctly:
Processor I/O LED flashes	This indicates there is a problem with backplane communications. Be certain this and all modules in the rack are configured in the processor.
BP ACT LED remains off or blinks slowly	This indicates that backplane transfer operations are failing. Use the Configuration/Debug port facility to check this. To establish backplane communications make sure of the following: 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, remove the card from the rack and re-insert the card in the rack.

6 Cable Connections

In This Chapter

The MVI56-MCM module has the following communication connections on the module:

- Two Modbus communication ports (RJ45 connector)
- One RS-232 Configuration/Debug port (RJ45 connector)



6.1 Modbus Communication Ports

The MVI56-MCM module has two physical Modbus connectors with a RJ45 plug located on the front of the module.

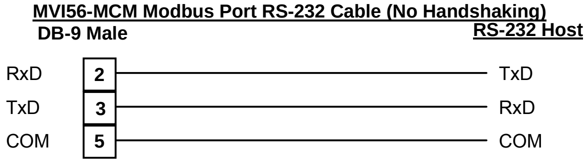
6.1.1 Connecting the Cable to the Connector

ProSoft provides two (2) RJ45 to male DB-9 pigtails to permit simpler interfacing to other devices. The module's Modbus ports can be configured to operate in RS-232, RS-422 or RS-485 mode. The interface to be associated with a port is set with jumpers on the module. There is a jumper for each of the two ports. Additionally, the use of the modem control lines is user definable by configuring the MCM.PORT1.UseCTS and MCM.PORT2.UseCTS variables to a value of one (1). The following sections describe each interface.

RS-232

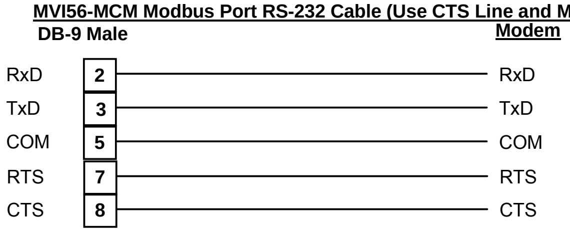
When the RS-232 interface is selected, the use of the modem control lines is user definable. If no modem control lines will be used, the cable to connect to the port is as shown in the following example:

MVI56-MCM Modbus Port RS-232 Cable (No Handshaking)



The RTS line is controlled by the RTS on and off parameters set for the port. If the CTS line is used (usually only required for half-duplex modems), the RTS and CTS lines must either be connected together or connected to the modem. The following diagram displays the cable required when connecting the port to a modem.

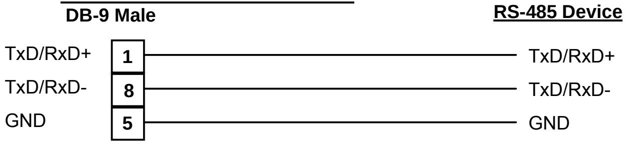
MVI56-MCM Modbus Port RS-232 Cable (Use CTS Line and Modem)



RS-485

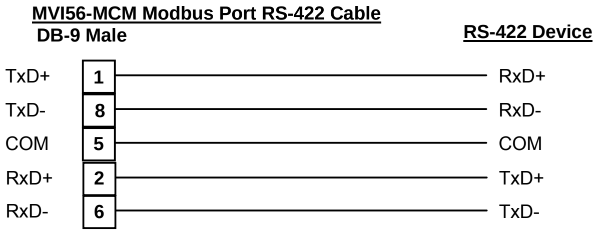
When the RS-485 interface is used, a single two or three wire cable is required. The use of the ground is optional and dependent on the RS-485 network. The cable required for this interface is shown in the following diagram:

MVI56-MCM Modbus Port RS-485



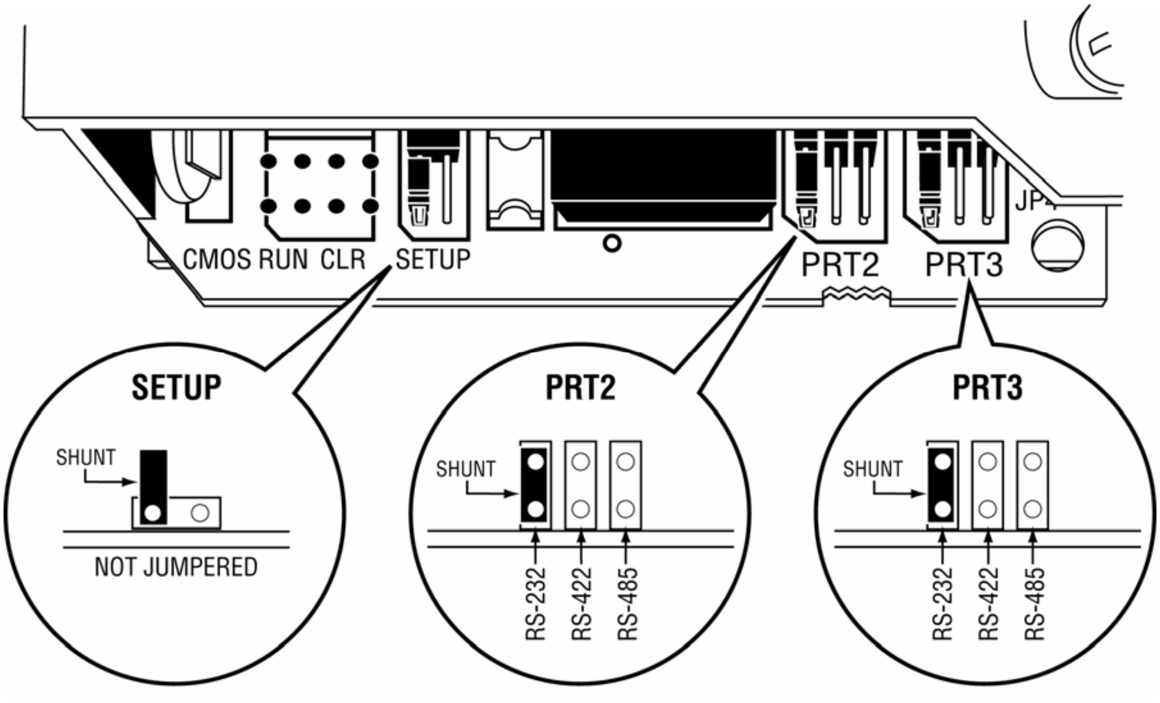
RS-422

When the RS-422 interface is used, a four or five wire cable is required. The use of the ground is optional and dependent on the RS-422 network. The cable required for this interface is shown in the following diagram:



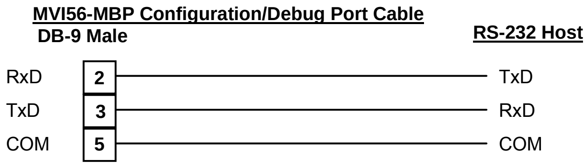
6.1.2 Setting Jumpers

If you use an interface other than RS-232 (default), you must change the jumper configuration to match the interface. The following diagram shows the MVI56-MCM jumper configuration:



6.2 RS-232 Configuration/Debug Port

This port is physically an RJ-45 connection. An RJ-45 to DB-9 pigtail cable is shipped 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:



Appendix A - MVI56-MCM Database Definition

This appendix contains a listing of the internal database of the MVI56-MCM module. This information can be used to interface other devices to the data contained in the module.

Register Range	Modbus Low	Modbus High	Content	Size
0-4999	40001	45000	User Data	5000
5000-5009	45001	45010	Backplane Configuration	10
5010-5039	45011	45040	Port 1 Setup	30
5040-5069	45041	45070	Port 2 Setup	30
5070-5869	45071	46070	Port 1 Commands	800
5870-6669	46071	47070	Port 2 Commands	800
6750-6752	46751	46753	Port 1 Aux. Setup	3
6760-6762	46761	46763	Port 2 Aux Setup	3
7600-7632	47601	47633	Misc. Status Data	33
7200-7232	47801	7999	Command Control	200

The User Data area is used to hold 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 Appendix B.

Definition of the configuration data areas can be found in the data definition section of this document and in Appendix C.

Appendix D contains a discussion of the command control section of the database.

Appendix B – MVI56-MCM Status Data Definition

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

Offset	Content	Description
202	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
203 –204	Product Code	These two registers contain the product code of "MCM".
205 – 206	Product Version	These two registers contain the product version for the current running software.
207 –208	Operating System	These two registers contain the month and year values for the program operating system.
209 – 210	Run Number	These two registers contain the run number value for the currently running software.
211	Port 1 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
212	Port 1 Command List Response	This field contains the number of slave response messages received on the port.
213	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.
214	Port 1 Requests	This field contains the total number of messages sent out of the port.
215	Port 1 Responses	This field contains the total number of messages received on the port.
216	Port 1 Errors Sent	This field contains the total number of message errors sent out of the port.
217	Port 1 Errors Received	This field contains the total number of message errors received on the port.
218	Port 2 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
219	Port 2 Command List Response	This field contains the number of slave response messages received on the port.
220	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.
221	Port 2 Requests	This field contains the total number of messages sent out the port.
222	Port 2 Responses	This field contains the total number of messages received on the port.
223	Port 2 Errors Sent	This field contains the total number of message errors sent out the port.

Offset	Content	Description
224	Port 2 Errors Received	This field contains the total number of message errors received on the port.
225	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
226	Write Block Count	This field contains the total number of write blocks transferred from the module to the processor.
227	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
228	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
229	Command Block Count	This field contains the total number of command blocks received from the processor.
230	Error Block Count	This field contains the total number of block errors recognized by the module.
231	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.
232	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.
233	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.
234	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.

Appendix C – MVI56-MCM Configuration Data Definition

This appendix contains listings of the MVI56-MCM module's database related to the module's configuration. This data is available to any node on the network and is read from the ControlLogix processor when the module first initializes.

Backplane Setup

Register	Content	Description
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 zero, the communication ports will continue to operate under all conditions. If the value is set larger than 0 (1 – 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 zero is entered, the data will not be stored in the database. If the value specified in the range of 0 to 4940, the data will be placed in the user data area.
5006	Spare	
5007	Spare	
5008	Spare	
5009	Spare	

Port 1 Setup

Register	Content	Description
5010	Enable	This parameter is used to define 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, 57600, 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.

Register	Content	Description
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.
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. Be certain 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	

Port 2 Setup

Register	Content	Description
5040	Enable	This parameter is used to define 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.

Register	Content	Description
5045	Protocol	This parameter specifies the Modbus protocol to be used on the port. Valid protocols are: 0 = Modbus RTU and 1 = Modbus ASCII.
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, 38400, 57600, and 115.
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. Be certain 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.

Register	Content	Description
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.
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	

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

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

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	

Register	Content	Description
6750	Port 1 Use Guard Band	Use packet gap timeout for messages (Yes or No). Use only in multi-drop applications.
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	

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). Refer to Appendix D for more information.
6801	Command Data	Not Used
-	-	-
6999	Command Data	Not Used

Appendix D – MVI56-MCM Command Control

Command Control data is received from other nodes on the network that can control the MVI56-MCM module. Specific values are written to regions of this block to control the module. Currently, the module is programmed to handle the receipt of the following requests: write configuration to processor, warm boot and cold boot.

The remote node controls the module by writing one of the following values to register 7800 (Modbus address 47801):

9997	Write configuration in database to the processor and warm boot the module.
9998	Warm boot the module.
9999	Cold boot the module.

The control register is cleared (a value of 0) after the operation is executed with the exception of the 9997 command. If the module fails to successfully transfer the configuration to the processor, an error code will be returned in the control register as follows:

0	No error, transfer successful
-1	Error transferring general configuration information.
-2	Error transferring Modbus Port 1 master command list
-3	Error transferring Modbus Port 2 master command list

Ladder logic must be written to handle the 9997 command. No ladder logic is required when using the warm or cold boot commands.

Appendix E – Product Specifications

General Specifications

The MVI56-MCM module acts as a gateway between the Modbus network and the Allen-Bradley backplane. The data transfer from the ControlLogix processor is asynchronous from the actions on the Modbus network. A 5000-word register space in the module is used to exchange data between the processor and the Modbus network.

Some of the general specifications include:

- Support for the storage and transfer of up to 5000 registers to/from the ControlLogix processor's data files
- Module memory usage that is completely user definable
- Two ports to emulate any combination of Modbus master or slave device
- Configurable parameters include:

Protocol	:	RTU or ASCII
Baud Rate	:	110 to 115,200
Parity	:	None, Odd and Even
Data Bits	:	5 to 8
Stop Bits	:	1 or 2
RTS On and Off Timing	:	0 to 65535 milliseconds
Minimum Response Delay	:	0 to 65535 milliseconds
Use of CTS Modem Line	:	Yes or No
Device Routing Paths	:	1 to 64
Floating-Point Support		

Slave Functional Specifications

The MVI56-MCM module accepts Modbus function code commands of 1, 2, 3, 4, 5, 6, 8, 15 and 16 from an attached Modbus master unit. A port configured as a Modbus slave permits a remote master to interact with all data contained in the module. This data can be derived from other Modbus slave devices on the network through a master port or from the ControlLogix processor. The module can be configured to pass write commands (functions 5, 6, 15, 16, 22, and 23) directly from the remote host to the processor. This mode of operation is referred to as pass-through mode.

Modbus Master Functional Specifications

A port configured as a virtual Modbus master device on the MVI56-MCM module will actively issue Modbus commands to other nodes on the Modbus network. One hundred commands are supported on each port. Additionally, the master ports have an optimized polling characteristic that will poll slaves with communication problems less frequently. The ControlLogix processor can be programmed to control the activity on the port by actively selecting commands from the command list to execute or issuing commands directly from the ladder logic. The ControlLogix processor also has the ability to control the scanning of slaves on the port. Polling of individual slaves can be selectively controlled (enabled/disabled) through the ladder logic.

Physical

This module is designed by ProSoft Technology and incorporates licensed technology from Allen-Bradley (ControlLogix backplane technology).

- ControLogix Form Factor - Single Slot
- Connections:

2– RJ45 connectors for support of RS-232, RS-422 or RS-485 interfaces

1– RJ45 RS-232 Configuration Tool Connector

ControlLogix Interface

- Operation via simple ladder logic
- Complete set up and monitoring of module through RSLogix 5000 software
- ControlLogix backplane interface via I/O access
- All data related to the module is contained in a single controller tag with defined objects to ease in the configuration, monitoring and interfacing with the module
- Module configuration and communication configuration data is transferred to the MVI56-MCM via a predefined user data type in the processor.

Hardware Specifications

The MVI56-MCM module is designed by ProSoft Technology and incorporates licensed technology from Allen-Bradley (ControlLogix backplane technology).

- Current Loads: 800 ma @ 5V (from backplane)
- Operating Temperature: 0 to 60 Deg C (32 to 140 Deg F)
- Storage Temperature: -40 to 85 Deg C (-40 to 185 Def F)
- Relative Humidity: 5-95% (w/o condensation)
- Modbus Port Connector: Two RJ45 Connectors (RJ45 to DB9 cable shipped with unit) supporting RS-232, RS-422 and RS-485 interfaces (RJ45 to DB9 cables shipped with unit)
- Configuration Connector: RJ45 RS-232 Connector (RJ45 to DB9 cable shipped with unit)

Support, Service & Warranty

Technical Support

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	(661) 716-5100 (661) 716-5101 (Fax)
Postal Mail	ProSoft Technology, Inc. 1675 Chester Avenue, Second 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 (you may wish to fax it to us prior to calling):

1. Product Version Number
2. System hierarchy
3. Module configuration and contents of MCM.CFG file
4. Module Operation
   - Configuration/Debug status information
   - LED patterns
5. Information about the processor and data areas as viewed through RSLogix 500 and LED patterns on the processor
6. Details about the serial network

Module Service and Repair

The MVI56-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 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 need to 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

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.

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 misapplication or misuse of the Product; (b) failure of Customer to adhere to any of ProSoft's specifications or instructions; (c) neglect of, abuse of, or accident to, the Product; or (d) any associated or complementary equipment or software not furnished by ProSoft.

Limited warranty service may be obtained by delivering the Product to ProSoft and providing proof of purchase or 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 further information.

Limitation of Liability

EXCEPT AS EXPRESSLY PROVIDED HEREIN, PROSOFT MAKES NO WARRANT 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 MERCHANT ABILITY 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.

Hardware Product Warranty Details

Warranty Period: ProSoft warranties hardware Product for a period of 1 year.

Warranty Procedure: Upon return of the hardware Product ProSoft will, at its option, repair or replace 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 current ProSoft standard rates for parts and labor, and return the Product freight collect.

Index