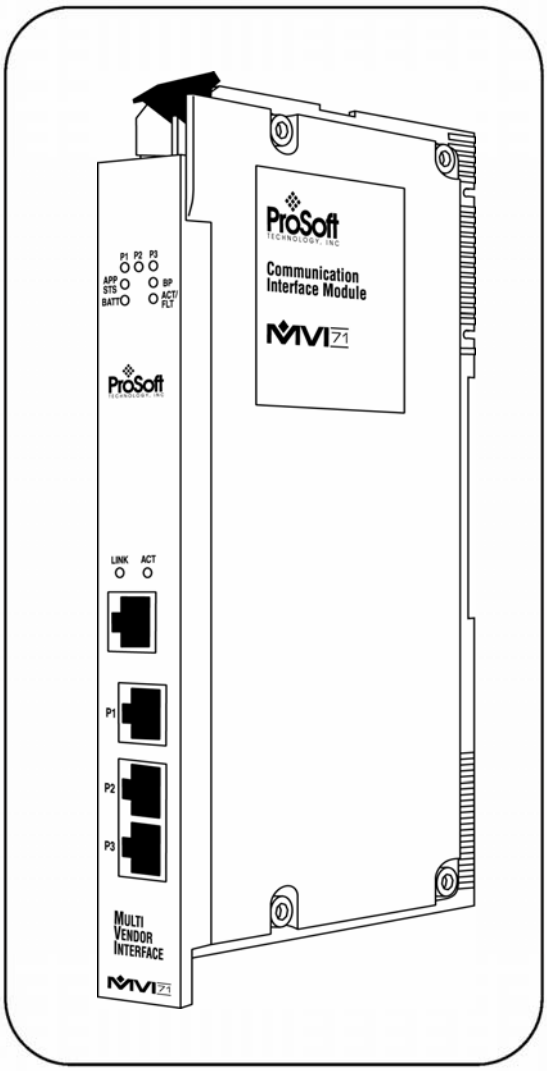
inRAx



MVI71-MCM

PLC Platform

Modbus Communication Module

User Manual

January 11, 2005

ProSoft TECHNOLOGY

Please Read This Notice

Successful application of this module requires a reasonable working knowledge of the PLC 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 A-B 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.

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.

ProSoft Technology, Inc.

1675 Chester Avenue, 2nd Floor

Bakersfield, CA 93301

(661) 716-5100

(661) 716-5101 (Fax)

www.prosoft-technology.com

MVI71-MCM User Manual

January 11, 2005

Table of Contents

1 Introduction

The MVI71-MCM (“Modbus Communication Module”) product allows Allen-Bradley PLC 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 MVI71-MCM module acts as a gateway between the Modbus network and the Allen-Bradley backplane. The data transfer from the PLC 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 MVI71-MCM is a module that allows the communication between the PLC and a Modbus network. The module has 2 ports that can be individually configured as a Modbus Master or as a Modbus Slave.

The module uses the rack backplane in order to transfer data to the PLC. Ladder logic is used for different tasks including:

- Downloading configuration data
- Updating the module’s internal database (read or write)
- Executing control blocks

The module uses block transfer instructions in order to read and write data to the PLC. The MVI71-MCM also supports side connect transfer (requires a side connect adapter) which allows a faster data transfer and does not require ladder logic to transfer data from/to the PLC.

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

1. Initialize hardware components
    a. Initialize PLC backplane driver or side-connect drivers
    b. Test and Clear all RAM
    c. Initialize the serial communication ports
2. Determine the interface to the backplane (side-connect or block transfer).
3. Wait for Module Configuration from PLC processor
4. Initialize Module Register space
5. Enable Slave Driver on selected ports
6. 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 Quick Start Guide

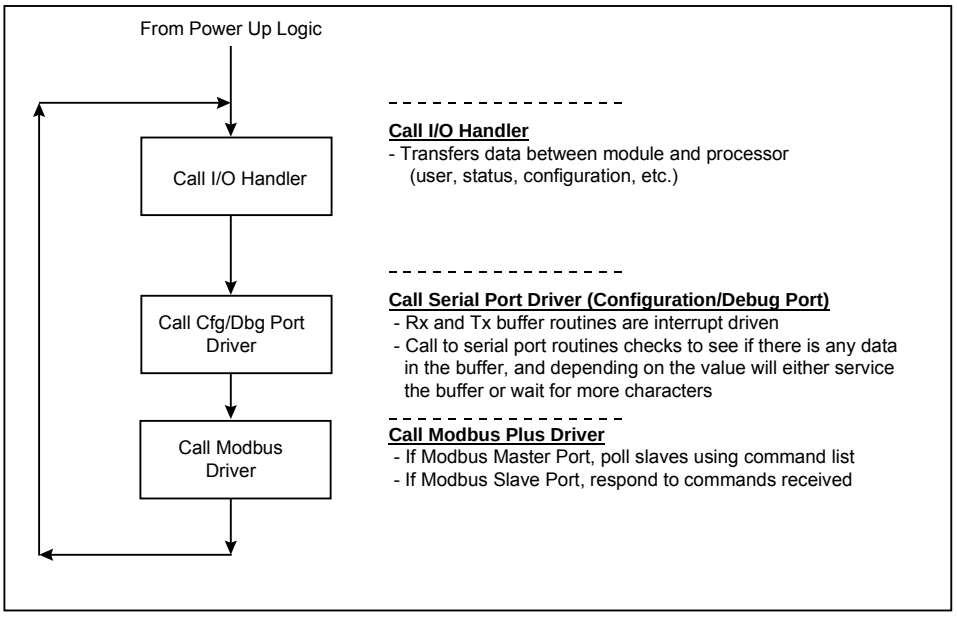
Appendix G contains a Quick Start Guide that provides step-by-step instructions on how to get your MVI71-MCM module up and running. However, it is important that you understand the contents of this manual. The Quick Start Guide provides the overview steps required to get your module up and running.

2 Understanding the Architecture

This section gives the reader a functional overview of the MVI71-MCM module. Details associated with the ladder logic and the memory-map are not covered in this section (refer to the section entitled **Configuring the Module**). 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 **Configuring the Module** 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:



2.2 PLC Processor Not in Run

Anytime the module detects that the processor has gone out of the Run mode (i.e., 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

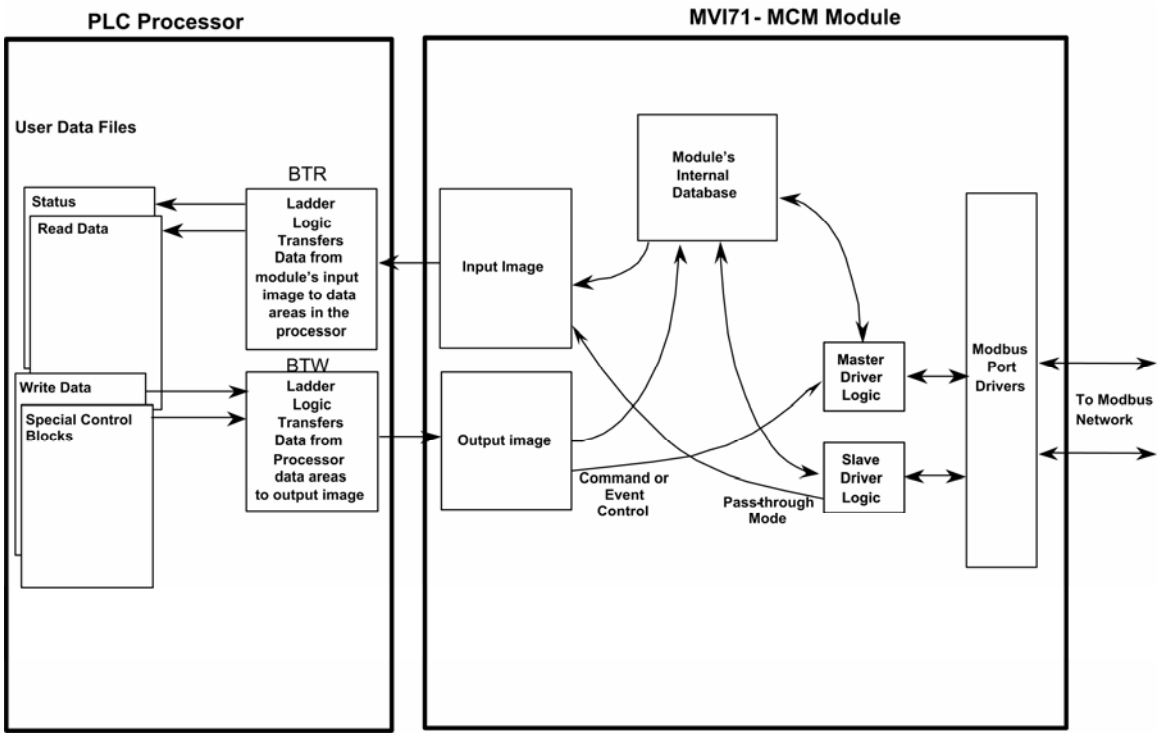
The MVI71-MCM module is unique in the way that the PLC backplane is utilized. Data is paged between the module and the PLC processor across the backplane using the module's input and output images or directly to the processor using the side-connect interface (requires a side-connect adapter). The update 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 data registers in the processor by the ladder logic. The input image for the module is set to 64 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 64 words. This large data area permits fast throughput of data from the processor to the module.

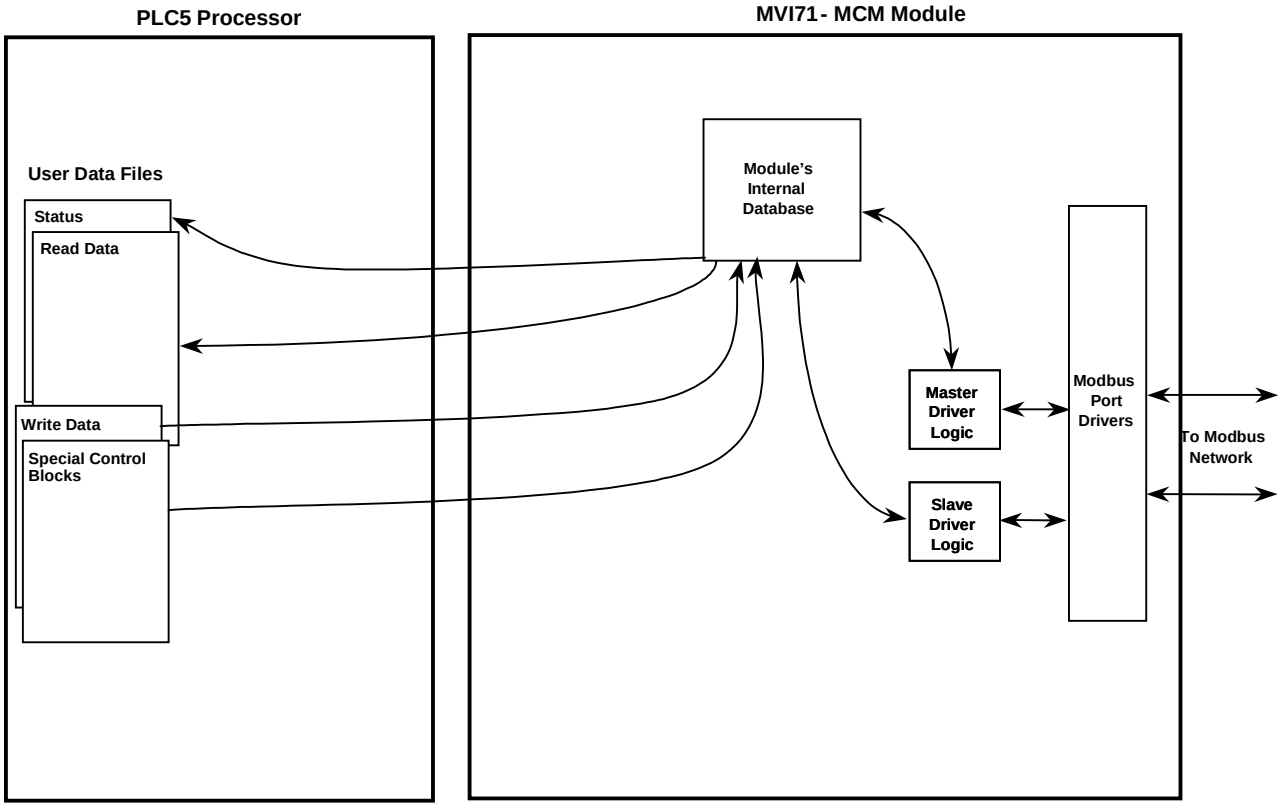
The following diagram shows the data transfer method used to move data between the PLC processor, the MVI71-MCM module and the Modbus Network.

Block Transfer



The following diagram shows the data transfer operations used when using the side-connect interface (requires the side-connect adapter):

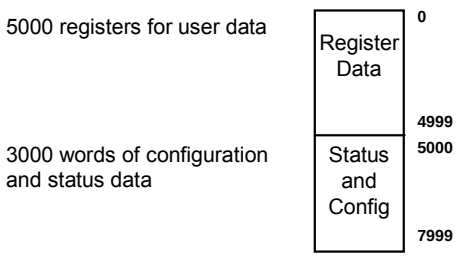
Side-Connect



When the side connect interface is used, data is transferred directly between the processor and the module. The module's program interfaces directly to the set of user data files established in the PLC to pass all data between the two devices. No ladder logic is required for data transfer, only the establishment of the data files.

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 PLC processor to interface the input and output image data with data defined in the data registers. 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



Data contained in this database is paged through the input and output images by coordination of the PLC ladder logic and the MVI71-MCM module's program. Up to 60 words of data can be transferred from the module to the processor at once. Up to 60 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 Modbus Addressing

It is important to familiarize yourself with the Modbus addressing concepts when using the MVI71-MCM module. The module supports the following address ranges for each Modbus function:

Function	Description	Modbus Address Range	MVI71-MCM Addressing
1	Read Output Status	0001-9999	bit
2	Read Input Status	10001-29999	bit
3	Read Holding Registers	40001-49999	word (16 bits)
4	Read Input Registers	30001-39999	word (16 bits)
5	Force Single Coil	0001-9999	bit
6	Preset Single Register	40001-49999	word (16 bits)
15	Force Multiple Coils	0001-9999	bit
16	Preset Multiple Registers	40001-49999	word (16 bits)

The table above also shows the MVI71-MCM addressing for each function. Depending on which function is used, the database address should be interpreted in bits or words.

Examples

MVI71-MCM as a Slave

- When a Modbus master device connected to a MVI71-MCM slave port sends a command function 5 to Modbus address 32, it will write to bit 32 in the MVI71-MCM database (bit 0 from word 2).
- When a Modbus master device connected to a MVI71-MCM slave port sends a command function 6 to Modbus address 40032, it will write to word 32 in the MVI71-MCM database.

MVI71-MCM as a Master

- When an MVI71-MCM Master Port sends a command with the following parameters:

  Internal DB Address: 10

  Count: 3

  Function: 15

It will write 3 coils to the specified slave using the source bits 10, 11, 12 in the MVI71-MCM database.

- When an MVI71-MCM Master Port sends a command with the following parameters:

  Internal DB Address: 10

  Count: 3

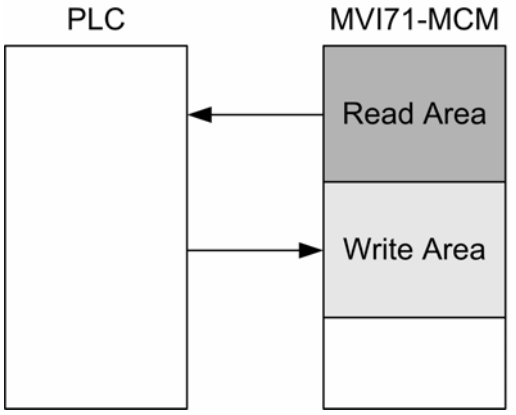
  Function: 16

  It will write 3 holding registers to the specified slave using as the source words 10, 11, 12 in the MVI71-MCM database.

2.5 Using the Read and Write Data Areas

The MVI71-MCM module separates user data into two regions; the Read Data area and the Write Data area. These areas must be defined inside the User Data area. The user can set up both areas when transferring the configuration data to the module.

The Read Data area is constantly transferred from the MVI71-MCM to the PLC using BTR instructions while the Write Data area is constantly transferred from the PLC to the MVI71-MCM module using BTW instructions:



For example, if the user sets the Read and Write areas as:

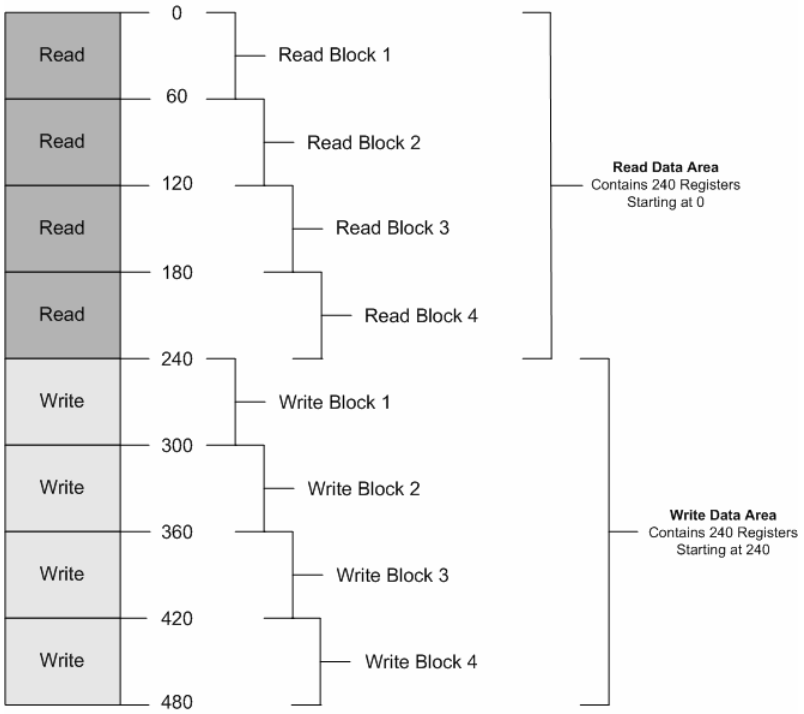
Read Register Start : 0

Read Register Count : 240

Write Register Start : 240

Write Register Count : 240

This configuration would set the module internal database as:



The configuration first creates four possible blocks for each Read and Write area since every block contains 60 registers. Therefore, the Read and Write Block ID generated would be:

Read Block ID	Write Block ID
0	3
1	4
2	1
3	2
4	3
-1	4
0	1
1	2
2	3

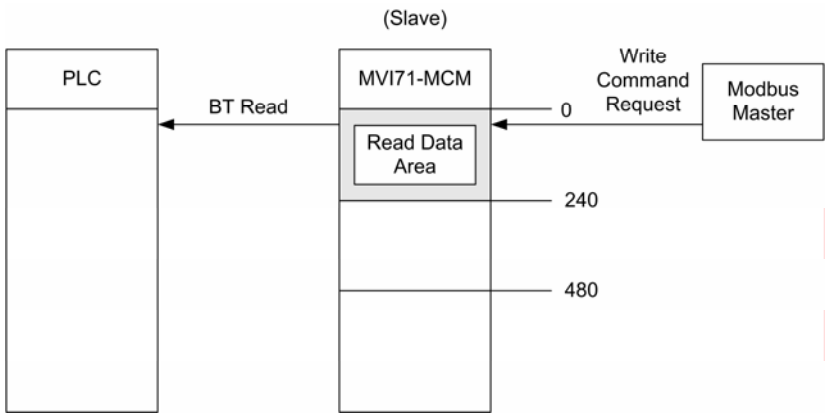
If the ladder logic takes too much time to send a BTW instruction, the module sends a new BTR instruction requesting the same Write Block ID.

The Read Block ID 0 is a null block to guarantee ladder logic consistency if the user sets an empty Read Data area. The Read Block ID –1 is used to transfer the configuration data.

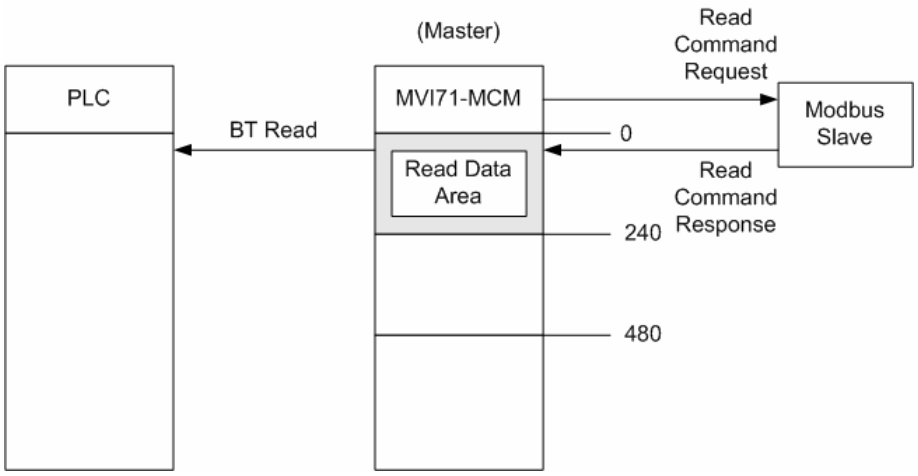
2.5.1 Read Data Area Application Examples

Ex. 1 -- The following example shows a Read Data Area application; A Modbus Master device sends a Modbus Write Command to an MVI71-MCM slave port. The

command destination address must be located inside the Read Data Region (between 0 and 239).

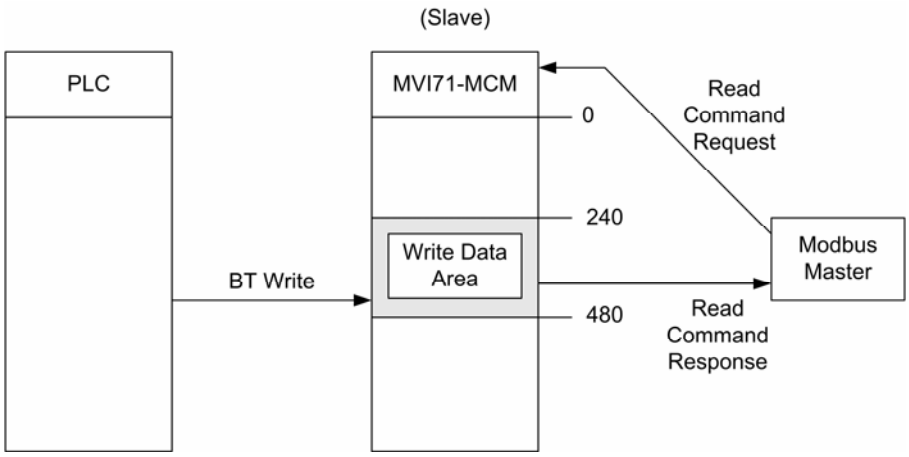


Ex. 2 -- The next figure shows another example of how to use the Read Data Area; an MVI71-MCM port configured as a Master sends a Modbus Read Command to a Slave device. The Modbus Read Command destination address must be located inside the Read Data Area (between addresses 0 and 239).

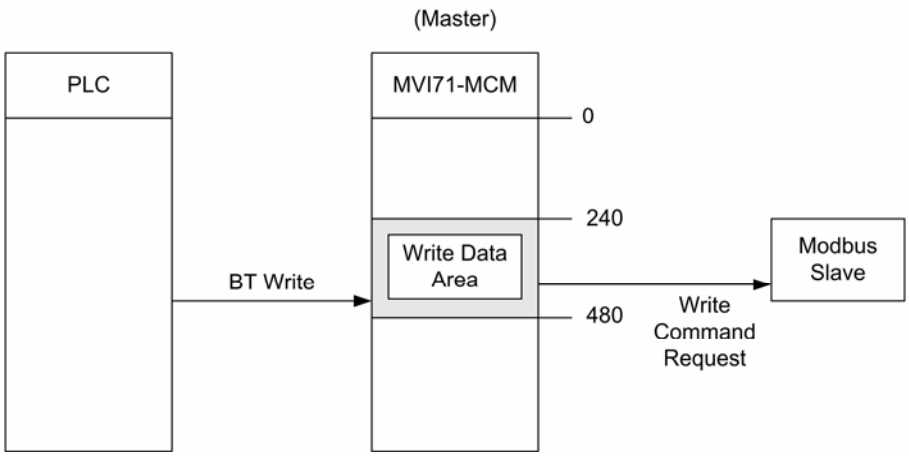


2.5.2 Write Data Area Application Examples

Ex. 1 -- The following figure shows a Write Data Area application; a Modbus Master device sends a Modbus Read Command to an MVI71-MCM slave port. The command source address (in the MVI71-MCM database) must be located inside the Read Data Region (between 240 and 479).



Ex. 2 -- The next figure shows another example of how to use the Write Data Area; an MVI71-MCM port configured as a Master sends a Modbus Write Command to a slave device. The Modbus Write Command source address in the MVI71-MCM database must be located inside the Write Data Area (between addresses 240 and 479).



2.6 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.

2.6.1 Read Block

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

Offset	Description	Length
0	Read Block ID	1
1	Write Block ID	1
2 – 61	Read Data	60
62 – 63	Spare	2

The Read Block ID is an index value used to determine the location of where the data will be placed in the PLC processor user data table. Each transfer can move up to 60 words (block offsets 2 to 61) of data.

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

If the ladder logic does not send a BTW instruction to the module quickly enough, it is possible for the MVI71-MCM module to send a new BTR instruction requesting the same write block ID.

2.6.2 Write Block

These blocks of data are used to transfer information from the PLC 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 – 60	Write Data	60
61 to 63	Spare	3

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 60 words (block offsets 1 to 60) of data.

2.6.3 Status Data Block (Read Block ID = -1)

After the last Read Block is sent, the module builds a BTR block (ID = -1) to transfer the module’s status information to the processor. This information can be used by

the PLC program to determine the current status of the module. Ladder logic should be constructed to transfer the information in this block to a user data file. The structure of this block is shown in the following table:

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

Offset	Content	Description
27	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
28	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
29	Command Block Count	This field contains the total number of command blocks received from the processor.
30	Error Block Count	This field contains the total number of block errors recognized by the module.
31	Port 1 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.
32	Port 1 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.
33	Port 2 Current Error	For a slave port, this field contains the value of the current error code returned. For a master port, this field contains the index of the currently executing command.
34	Port 2 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.

2.6.4 Master Command Blocks

Each port on the module can be configured as a Modbus master device containing its own list of one hundred commands.

Each command has the following structure:

Offset	Parameter
0	Enable
1	Internal Address
2	Poll Interval
3	Count
4	Swap
5	Slave Address
6	Function
7	Destination Address
8	Spare
9	Spare

For information about these parameters, refer to the **Configuring the Module** section under Modbus Master Commands.

The commands are read from the processor using the following Write Block ID's: Modbus Port 1 -- 6000 to 6016 and Modbus Port 2 -- 6100 to 6116. 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 sample ladder logic already handles this). The structure of each block is shown in the following table.

Offset	Description	Length
0	6000 to 6016 and 6100 to 6116	1
1 – 10	Command Definition	10
11 – 20	Command Definition	10
21 – 30	Command Definition	10
31 – 40	Command Definition	10
41 – 50	Command Definition	10
51 – 60	Command Definition	10
61 – 63	Command Definition Spare	3

Each block contains six commands. Since there are 100 possible commands, the last block (6016 or 6116) should contain up to four commands.

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 **Appendix D**.

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. 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 slave status values for Port 1 slaves 0 to 59

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

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

Offset	Description	Length
0	3002 – 3006 or 3102 – 3106	1
1 – 63	Spare	63

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	3002 – 3006 or 3102 – 3106	1
1	Write Block ID	1
2 – 61	Slave Poll Data Status	60
62 to 63	Spare	2

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
2 – 61	Slave indexes	60
62 – 63	Spare	2

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	3000 or 3100	1
1	Write Block ID	1
2	Number of slaves processed	1

Offset	Description	Length
3 – 63	Spare	61

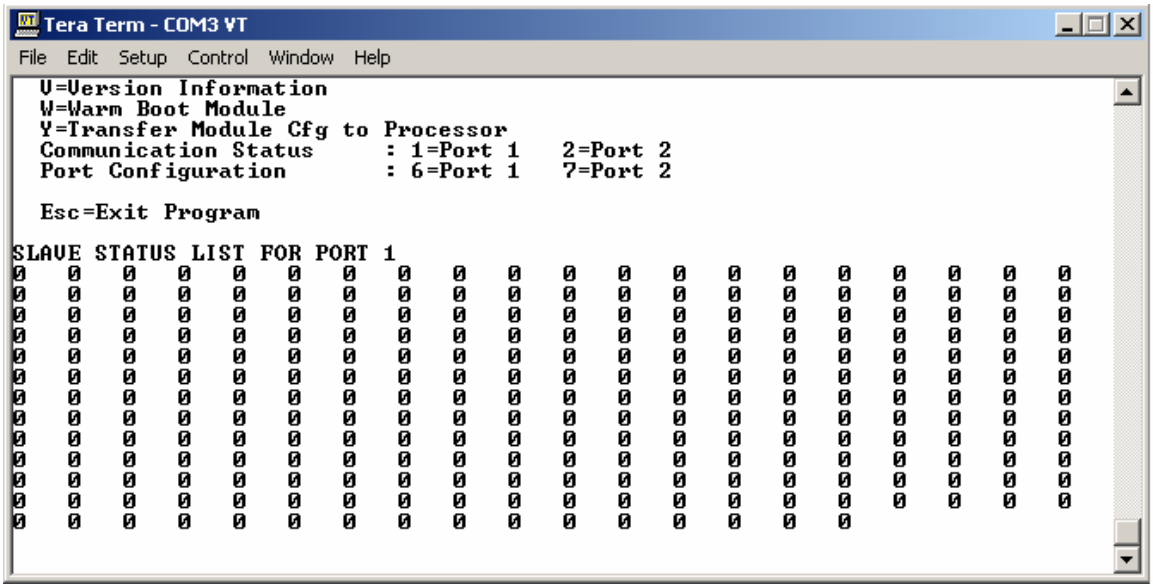
Ladder logic can be written to override the value in the slave status table to enable the slave (state value of 1) by sending a special block. Port 1 slaves are enabled using block 3001, and Port 2 slaves are enabled using block 3101. Each block contains the slave node addresses to enable. The format 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 – 61	Slave indexes	60
62 – 63	Spare	2

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 – 63	Spare	61

The user can monitor the slave status table using the Debug port .< option Slave Status List>:



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 Using the Command Control Blocks

Ladder logic must be considered in order to use command control blocks. The ladder logic is responsible for moving the control block data to the correct location in order for the data to be read by the module.

2.8.1.1. Block Transfer

The ladder logic should copy the command control block to the write block transfer data file. For example, if using N7:310 as the write block transfer data file, the structure would look as follows:

Offset	Data File
0	N7:310
1	N7:311
2	N7:312
3	N7:313
4	N7:314
5	N7:315
…	…
63	N7:363

2.8.1.2. Side-Connect (Requires Side-Connect Adapter)

The ladder logic should copy the command control block to the side-connect data file starting at register 80.

For example, if the user selected N10 as the side connect data file, the ladder logic should move the command control block according to the following structure:

Offset	Data File
0	N10:80
1	N10:81
2	N10:82
3	N10:83
4	N10:84
5	N10:85
…	…
63	N10:143

2.8.2 Event Command

Event command control blocks are used to send Modbus commands directly from the ladder logic to one of the master ports. This control block should be only be used under special circumstances. Normally, the user should use the configuration command list to send commands. The format for these blocks is displayed in the following table:

Write Block

Offset	Description	Length
0	1000 – 1255 or 2000 – 2255	1
1	Internal DB Address	1
2	Point Count	1
3	Swap Code	1
4	Function	1
5	Device DB Address	1
6 - 63	Spare	57

The block number defines the Modbus port to be considered and the slave node to be accessed. For example, if sending a command to Slave 1, Block ID 1001 should be used (from Port 1). 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 (100 registers or 800 coils). 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:

Read Block

Offset	Description	Length
0	1000 or 2000	1
1	Write Block ID	1
2	0 = Fail, 1 = Success	1
3 – 63	Spare	61

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).

2.8.3 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:

Write Block

Offset	Description	Length
0	5001 – 5006 or 5101 – 5106	1
1	Command index	1
2	Command index	1
3	Command index	1
4	Command index	1
5	Command index	1
6	Command index	1
7 – 63	Spare	57

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:

Read Block

Offset	Description	Length
0	5001 – 5006 or 5101 – 5106	1
1	Write Block ID	1
2	Number of commands added to command queue	1
3 – 63	Spare	61

2.8.4 Read Current Configuration

This block is sent from the PLC 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	-9000	1
1	-9000	1
2 – 7	Backplane Setup	6
8 – 32	Port 1 Configuration	25
33 – 57	Port 2 Configuration	25
58 – 63	Spare	6

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

Offset	Description	Length
0	-6000 to 6016 and –6100 to 6116	1
1	-6000 to 6016 and –6100 to 6116	1
2 – 11	Command Definition	10
12 – 21	Command Definition	10
22 – 31	Command Definition	10
32 – 41	Command Definition	10
42 – 51	Command Definition	10
52 – 61	Command Definition	10
62 – 63	Spare	2

Each of these blocks must be handled by the ladder logic for proper module operation. The processor can request the module's configuration by sending a configuration read request block, block code 9997, to the module. The format of this request block is as follows:

Offset	Description	Length
0	9997	1
1 – 63	Spare	63

When the module receives this command block, it transfers the module's current configuration to the processor. If the block transfer interface is used, the blocks defined in the previous tables (-9000 and –6000 series blocks) will be sent from the module. If the side-connect interface is used, the user data files will be updated directly by the module.

2.8.5 Warm Boot

This block is sent from the PLC 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 data registers. 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 – 63	Spare	63

2.8.6 Cold Boot

This block is sent from the PLC 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 – 63	Spare	63

2.9 Pass-Through Control Blocks

The pass-through feature allows a Master device to write commands directly to the PLC ladder logic.

2.9.1 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:

2.9.1.1. Function 5

Offset	Description	Length
0	0	1
1	9958	1
2	1	1
3	Bit Address	1
4	Data	1

Offset	Description	Length
5 – 58	Modbus message received	59

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 – 63	Spare	63

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

2.9.1.2. 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 – 63	Data	60

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 – 63	Spare	63

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

2.9.1.3. 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 – 63	Data	60

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 – 63	Spare	63

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

2.10 Remote Command Control

Command Control data is received from other nodes on the network that can control the MVI71-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

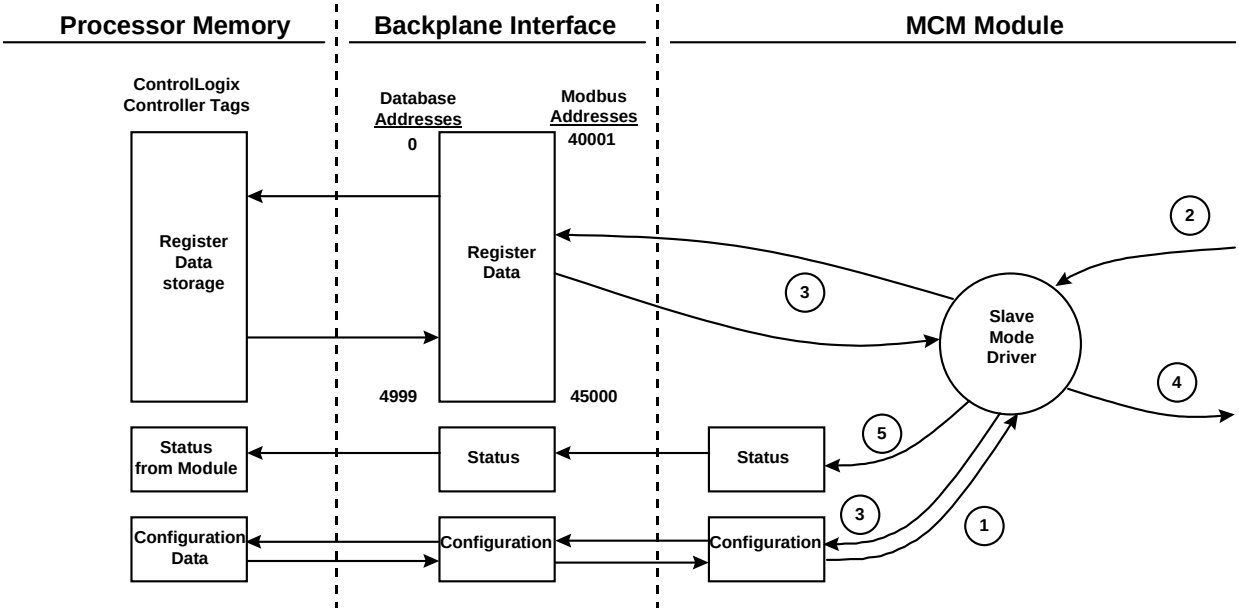
Ladder logic must be written to handle the 9997 command. No ladder logic is required when using the warm or cold boot commands.

2.11 Data Flow Between MVI71-MCM Module and PLC Processor

The following discussion details the flow of data between the two pieces of hardware (PLC processor and MVI71-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.

2.11.1 Slave Driver

The Slave Driver Mode allows the MVI71-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 PLC 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 **Configuring the Module** section for a complete list of the parameters that must be defined for a slave port.

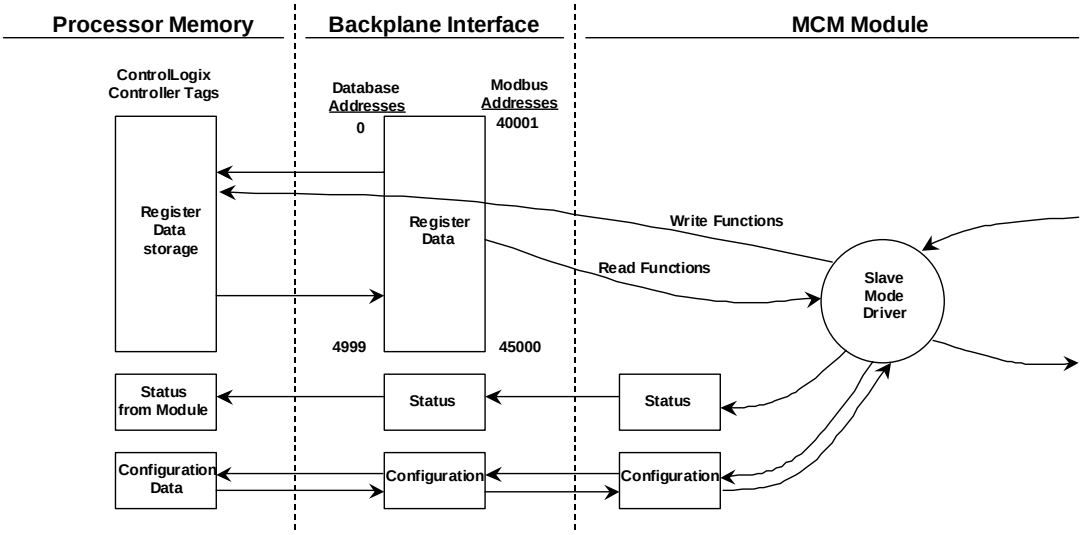
2.11.1.1. Pass-Through

Each port can have the type parameter configured for pass-through mode where:

Swap Code	Description
2	Pass-Through Formatted
3	Pass-Through Formatted with Byte Swap

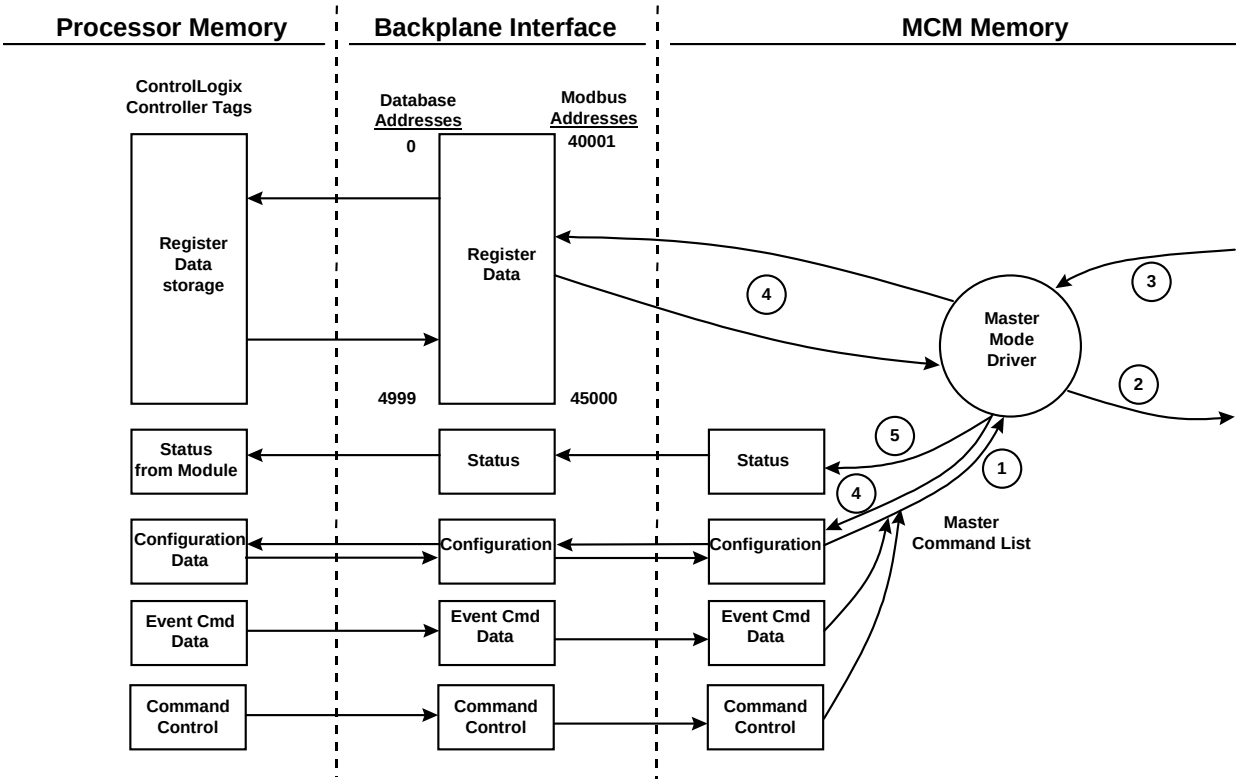
Option 3 swaps all bytes in the data received from a Modbus Master command.

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:



2.11.2 Master Driver Mode

In the Master Mode of operation, the MVI71-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 PLC processor or issued directly from the PLC 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 PLC 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 Module Set Up section).
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.
5	Status is returned to the PLC processor for each command in the Master Command List.

Refer to the **Configuring the Module** 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.

3 Configuring the Module

In order for the MVI71-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 PLC 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.
5200-6199 and 6400-7399	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.

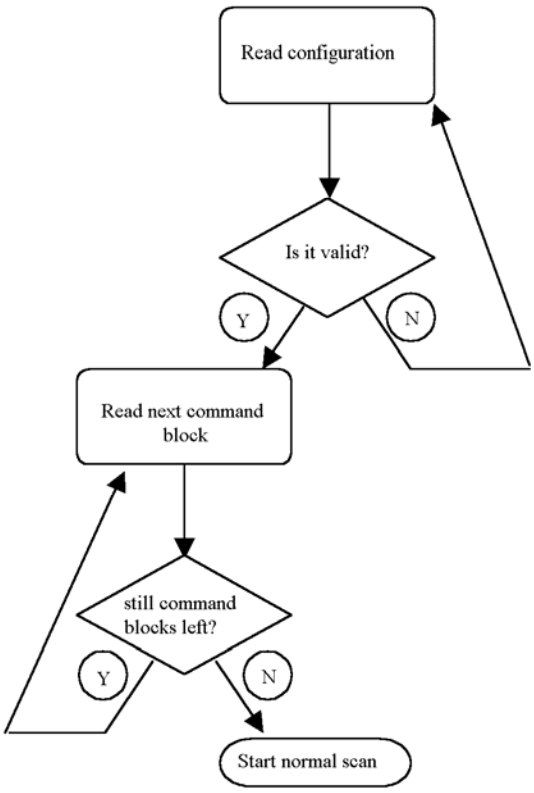
Refer to the Setting Up the Module section for a description of the configuration of the module. The MVI71-MCM module must be configured at least once when the card is first powered, and any time thereafter when the parameters must be changed.

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 Configuration Data Transfer

When the module performs a restart operation, it will request configuration information from the PLC 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 **Configuring the Module** section for a description of the data objects used with the blocks and the ladder logic required. The module will request all command blocks, according to the number of commands configured by the user for each Master port. The following diagram shows this procedure:



The format of the blocks for configuration is given in the following sections.

3.2.1 Module Configuration

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:

Write Block

Offset	Description	Length
0	9000	1
1 – 6	Backplane Setup	6
7 – 31	Port 1 Configuration	25
32 – 56	Port 2 Configuration	25
57 – 63	Spare	7

The read block used to request the configuration has the following structure (Please refer to **Appendix C** for a listing of configuration data):

Read Block

Offset	Description	Length
0	-2	1
1	9000	1
2	Module Configuration Errors	1
3	Port 1 Configuration Errors	1
4	Port 2 Configuration Errors	1
5 – 63	Spare	59

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.

Please refer to **Appendix E** for a listing of error words associated with invalid configuration blocks.

3.3 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 PLC 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 PLC 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.

3.4 Setting Up the Module

Set up of the MVI71-MCM module only requires software configuration using the RSLogix 5 program. The easiest method to implement the module is to start with the example provided with the module (MVI71MCM_BT.RSP (block transfer example) or MVI71MCM_SC.RSP (side-connect example)). 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.

The first step in setting up the module is to define whether the block transfer or side-connect interface will be used. If the block transfer interface is used, remove the Compact Flash Disk from the module if present and insert the module into the rack with the power turned off.

3.5 Module Data Files

All data related to the MVI71-MCM is stored in user-defined data files. The user is responsible for setting up the data areas to match the specific application for the module. Each data area is discussed in the following sections.

3.5.1 Configuration Data

Configuration of the module is performed by filling in a user-defined data table. In the example ladder logic (block transfer), file N10 is used to store the general module configuration information. N11 is used to store the command list for Port 1. N12 is used to store the command list for Port 2. Each register in the files has an associated symbol and description to aid in filling in the data. The appendix contains a list of items that must be configured for the module and their associated location in the file. When the side-connect interface is used, the file used for configuration must match that set in the SC_DATA.TXT file using the setdnpsc.exe program.

3.5.1.1. Data Transfer Parameter

The first five parameters of the configuration relate to the data transfer between the module and the processor as follows:

- **Write start register –** Offset in the module's database where write data is placed
- **Write register count –** Number of registers to transfer from processor to module
- **Read start register –** Offset in the module's database where read data is sourced
- **Read register count –** Number of registers to transfer from the module to the processor
- **Backplane fail –** Number of successive transfer failures that cause communication shutdown
- **Error Status Pointer –** Offset in the module's database where the error status is stored.

These parameters apply to both the side-connect and block transfer interfaces. For the block transfer interface, the number of blocks to transfer between the module and the processor is determined by the count values set in these parameters. Each block can transfer a maximum of 60 words. For example, if the Write Register Count parameter is set to 240, four write blocks will be transferred (1 to 4) between the processor and the module. When the side-connect interface is used, each block can transfer up to 1000 words of data. The user data files must be set to match the values entered in this parameter set. For example, if the Read Register Count parameter is set to 2100, three user data files must be defined. The first two must contain 1000 elements and the last one must contain at least 100 elements.

The Backplane Fail 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 0, 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 resume communication 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 resumes communications.

The Error Status Pointer parameter is used to define the location in the module's database where the error/status data will be stored. Ir 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.

3.5.1.2. Modbus Port Parameters

These parameters are used to define the operation of each of the Modbus ports on the module. Refer to the appendix for a definition of each parameter. These parameters are contained in the configuration at offsets 6 to 55. Care must be taken when filling in this data area for successful operation of the module in a user application.

3.5.1.3. Modbus Master Commands

These records are used to define the commands in the master command list.

Each command has the following structure:

Offset	Parameter
0	Enable
1	Internal Address
2	Poll Interval
3	Count
4	Swap
5	Slave Address
6	Function
7	Destination Address
8	Spare
9	Spare

Enable – This parameter is used to define if the command will be executed or disregarded. The following values are valid:

- 0=Disables the command
- 1=The command is considered for execution at each scan of the command list and will be controlled by the PollInt parameter
- 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.

Internal Address – This parameter specifies the starting internal register address to be associated with the command. Valid entries for this parameter are 0 to 9999.

For example, if using a read holding register function (function 3), the Internal Address parameter will have the address in the MVI71-MCM database in which the value read from a Modbus slave is located. If using a write function (function 16), the Internal Address will have the source address in the MVI71-MCM database which contains the value to be written to a Modbus slave device.

Important: The Internal Address must be used according to the Modbus function. When using holding registers (Functions 3,6,16), or input registers (Function 4), the Internal Address indicates the word (16 bits) address in the MVI71-MCM internal database. For example, if using Internal Address = 500, this indicates that word 500 is the Start Internal Address. When using bits (functions 1,2,5,15), the Internal Address parameter indicates the bit address in the MVI71-MCM internal database. For example, if using Internal Address = 32 and function 15, this indicates that the source address starts at bit 0 at word 2

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 entries for this parameter are 0 through 65535.

Count – This parameter defines the number of registers to be considered by the command. Valid entries for this parameter are 1 to 100 for functions 3, 4, and 16 and 800 for functions 1, 2, and 15. The Count parameter must be interpreted according to the Modbus function used. Functions 1, 2, and 15: Count = Number of Coils (1 bit). Functions 3, 4, and 16: Count = Number of Words (16 bits).

Swap – This parameter is used to specify if the data used in the command must be altered when reading data from a node on the network (Function 3). Values that can be assigned are as follows:

- 0=No data swapping
- 1=Swap word values
- 2=Swap word and byte values
- 3=Swap byte values

This option is used when interfacing the module with ASCII and floating-point data on other devices.

Slave Address – This parameter is used to assign the Modbus slave node address for the module to reach with the command on the network. This parameter can use values from 0 to 255. Most Modbus networks limit the upper value to 247.

Func – This parameter specifies the function to be performed by the command. Valid entries are 1, 2, 3, 4, 5, 6, 15, and 16.

Destination Address -- This parameter stores the destination address in the slave associated with the command.

For example, if using a Read holding register function (function 3), the Destination Address parameter will contain the address to be read in the Modbus slave

connected to a MVI71-MCM master port. If using a Write function (function 16), the Destination Address will contain the slave address to which to command will be written.

Important: The Destination Address must be used according to the Modbus function in the command. When using holding registers (functions 3,6,16), or input registers (function 4), the Destination Address is expressed as the word (16 bits) address in the slave. For example, Destination Address = 500 indicates that word 500 is the start Destination Address. When using bits (functions 1,2,5,15,) the Destination Address parameter is expressed as bit address in the MVI71-MCM Destination database. For example, Destination Address = 32, with function 15, indicates that the source address starts at the bit 0 at word 2.

When the side-connect interface is used, the command must be contiguous in the file, each one occupying a 10-word area. The file numbers used are also fixed. Port 1 commands 0 to 99 will reside in the file after the configuration file. Port 2 commands 0 to 99 will reside in the next file.

The following provides some command examples:

Enable	Internal Address	Poll	Count	Swap	Slave Address	Function	Destination Address
1	100	0	10	0	1	16	55

This command continuously writes 10 words starting at address 100 to 109 in the MVI71-MCM internal database to addresses 55 to 64 at slave address 1.

Enable	Internal Address	Poll	Count	Swap	Slave Address	Function	Destination Address
2	0	0	1	0	5	6	1000

This command conditionally writes 1 word from address 0 in the MVI71-MCM internal database to address 1000 in slave 5. This means that the command is issued only when the value at address 0 changes.

Enable	Internal Address	Poll	Count	Swap	Slave Address	Function	Destination Address
1	0	15	1	0	5	6	1000

This command continuously writes 1 word from address 0 in the MVI71-MCM internal database to address 1000 in slave 5 every 15 seconds.

Enable	Internal Address	Poll	Count	Swap	Slave Address	Function	Destination Address
1	320	0	20	0	5	15	1600

This command continuously writes 20 bits from bit address 320-339 (word 20, bit 0) in the MVI71-MCM internal database to bit address 1600-1619 (word 100, bit 0) in slave 5.

Enable	Internal Address	Poll	Count	Swap	Slave Address	Function	Destination Address
1	320	15	1	0	5	5	1600

This command continuously writes 1 bit from bit address 320 (word 20, bit 0) in the MVI71-MCM internal database to bit address 1600 (word 100, bit 0) in slave 5 every 15 seconds.

Enable	Internal Address	Poll	Count	Swap	Slave Address	Function	Destination Address
1	850	0	10	3	5	3	700

This command continuously reads 10 holding registers from addresses 700-709 in slave 5 to addresses 850-859 in MVI71-MCM internal database after swapping the bytes in each word.

3.5.2 Status Data

This data area is used to view the status of the module. Refer to the appendix for a complete listing of the data stored in this object. When the side-connect interface is used, this data is automatically updated in the configuration file (for example, N10:) starting at offset 200 approximately every second and does not include the first two registers. For the block transfer interface, the module generates blocks with a BTR block identification code of –1. Ladder logic must be written to transfer this information into a user data file.

3.6 User Data

Data in the module's internal database in the register range of 0 to 4999 is available to the processor. The parameters set in the configuration determine the set of data that is transferred from the module to the processor (read data) and that transferred from the processor to the module (write data). If the block transfer interface is used, ladder logic is required to handle the transfer of data between the processor and the module. BTR messages are required to handle data read from the module, and BTW messages are required to handle data written to the module. When the side-connect interface is used, the data is directly transferred between the module and the user data files without the ladder logic requirement.

3.7 Slave Polling Control and Status

The 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. Special blocks (block transfer interface) or control command (side-connect interface) are required to interface with this data. Using block (command) 3000 or 3100, slaves

can be disabled for polling. They can be enabled using block (command) 3001 or 3101. Blocks 3002 to 3006 or 3102 to 3106 request the current status of each slave in the module.

3.8 Using Side-Connect (Requires Side-Connect Adapter)

If the side-connect interface is used, make sure the file SC_DATA.TXT on the Compact Flash Disk contains the correct configuration file number. You can run the setdnpsc.exe program to set the configuration file number to be used with your application.

Install the module in the rack and turn on the power. Connect the terminal server to the module's debug/configuration port and exit the program by pressing the Esc key followed by the "Y" key. This causes the program to exit and remain at the operating system prompt. Run the setdnpsc.exe program with a command line argument of the file number to use for the configuration file. For example, to select N10: as the configuration file, enter the following:

SETDNPSC 10

Note: The SETDNPSC.EXE utility will only set the N file number between 10 and 933.

The program will build the SC_DATA.TXT on the Compact Flash Disk (C: drive in the root directory).

The next step in module setup is to define the data files to be used with the application. If the block transfer interface is used, define the data files to hold the configuration, status, and user data. Enter the module's configuration in the user data files. Enter the ladder logic to handle the blocks transferred between the module and the PLC. Download the program to the PLC and test the program with the module.

If the side-connect interface is used, no ladder logic is required for data transfer. The user data files to interface with the module must reside in contiguous order in the processor. The first file to be used by the interface is the configuration file. This is the file number set in the SC_DATA.TXT file using the SETDNPSC.EXE program. The following table lists the files used by the side-connect interface:

File Number	Example	Size	Description
Cfg File	N10	300	Configuration/Control/Status File
Cfg File+1	N11	to 1000	Port 1 commands 0 – 99
Cfg File+2	N12	to 1000	Port 2 commands 0 – 99
Cfg File+5	N15	to 1000	Data transferred from the module to the processor.
			Other files for read data.
Cfg File+5+n	N16	to 1000	Data transferred from the processor to the module.
Cfg File +5+n+m			Other files for write data.

n is the number of read data files minus one. Each file contains up to 1000 words.

m is the number of write data files minus one. Each file contains up to 1000 words.

Even if both files are not required for a port's commands, they are still reserved and should only be used for that purpose. The read and write data contained in the last set of files possess the data transferred between the module and the processor. The number of files required for each is dependent on the number of registers configured for each operation. Two examples follow:

Example of 240 words of read and write data (cfg file=10)

Data Files	Description
N15:0-239	Read Data
N16:0-239	Write Data

Example of 2300 read and 3500 write data registers (cfg file=10)

Data Files	Description
N15:0-999	Read data words 0-999
N16:0-999	Read data words 1000-1999
N17:0-299	Read data words 2000-2299
N18:0-999	Write data words 0-999
N19:0-999	Write data words 1000-1999
N20:0-999	Write data words 2000-2999
N21:0-499	Write data words 3000-3499

Special care must be taken when defining the files for the side-connect interface. Because the module directly interacts with the PLC processor and its memory, any errors in the configuration may cause the processor to fault and it may even lose its configuration program. After defining the files and populating them with the correct data, download the program to the processor, and place the processor in Run mode. If everything is configured properly, the module should start its normal operation.

If all the configuration parameters are set correctly, and the module is attached to a Modbus network, the modules application LED (OK LED) should remain off and the backplane activity LED (BP ACT) should blink rapidly. Refer to the Diagnostics and Troubleshooting of this manual if you encounter errors. Attach a terminal to Port 1 on the module and check the status of the module using the resident debugger in the module.

4 Sample Ladder Logic

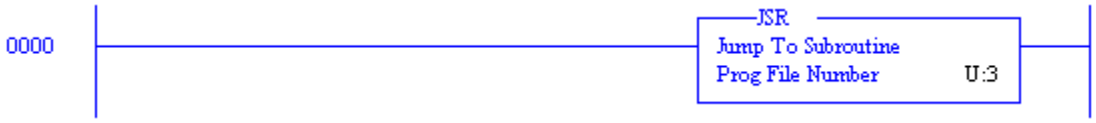
Ladder logic is required for application of the MVI71-MCM module when using the block transfer interface. Ladder logic is only required when using the side-connect interface to perform special functions. Tasks that must be handled by the ladder logic are module configuration, data transfer, special block handling and status data receipt. This section discusses each aspect of the ladder logic as required by the module. The sections that follow describe the simple ladder logic example provided for each interface.

4.1 Block Transfer Interface

When the block transfer interface is used, ladder logic is required to transfer all data between the module and the processor.

4.1.1 Main Routine

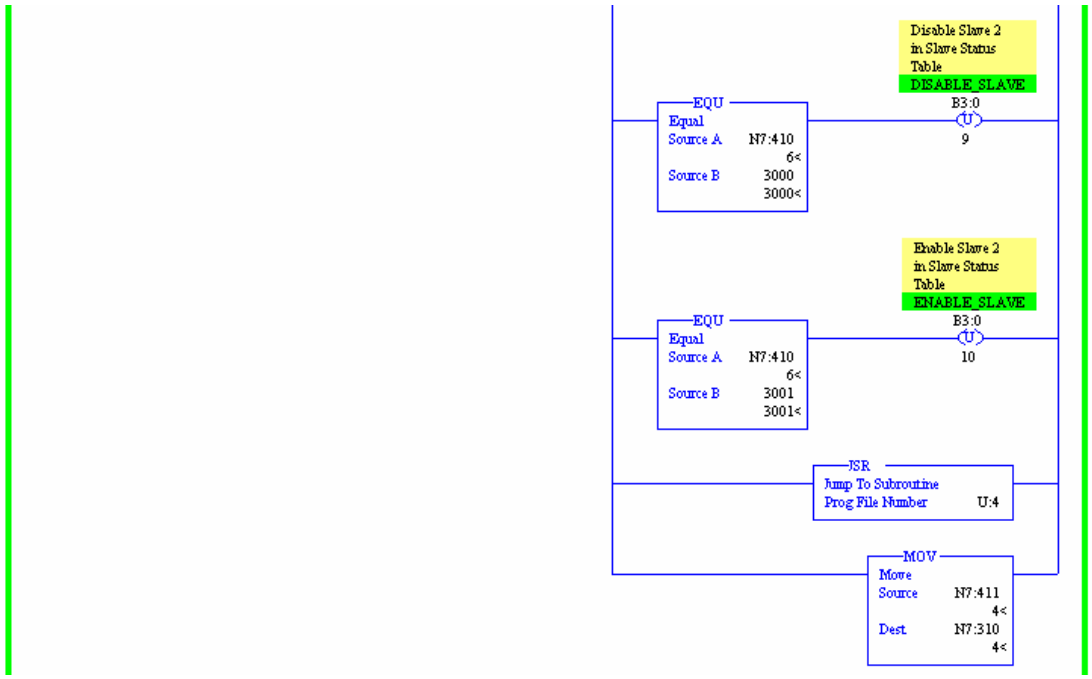
The Main program file is used to jump to the routine that processes the BTR and BTW functions for the interface. Ladder logic to accomplish this task is shown below:



4.1.2 Block Transfer Routine

The Block Transfer Routine is used to handle the BTR and BTW operations to transfer data between the processor and the module. Each block to be interfaced between the processor and the module must be addressed in this logic. The example ladder logic displays the minimum application of the module and does not use any of the special features offered by the module. The first rung of the routine is used to handle the BTR operation (data read from the module). The rung is shown below:


0000
N7:300
15
N7:400
15
BTR
Block Transfer Read
Module Type Generic Block Transfer
Rack 000
Group 0
Module 0
Control Block N7:400
Data File N7:410
Length 64
Continuous No
Setup Screen
EN
DN
ER
Recrive configuration from module
#WRITE_START_REG
EQU
Equal
Source A N7:410
6<
Source B -9000
-9000<
COP
Copy File
Source #N7:412
Dest #N10:0
Length 56
EQU
Equal
Source A N7:410
6<
Source B -2
-2<
COP
Copy File
Source #N7:412
Dest #N10:100
Length 3
EQU
Equal
Source A N7:410
6<
Source B -1
-1<
COP
Copy File
Source #N7:412
Dest #N9:0
Length 60
EQU
Equal
Source A N7:410
6<
Source B 1
1<
COP
Copy File
Source #N7:412
Dest #N15:0
Length 60
EQU
Equal
Source A N7:410
6<
Source B 2
2<
COP
Copy File
Source #N7:412
Dest #N15:60
Length 60
EQU
Equal
Source A N7:410
6<
Source B 3
3<
COP
Copy File
Source #N7:412
Dest #N15:120
Length 60
EQU
Equal
Source A N7:410
6<
Source B 4
4<
COP
Copy File
Source #N7:412
Dest #N15:180
Length 60



This rung will only execute when a BTR or BTW message is not enabled. This logic is required to alternate between the BTR and BTW messages. When it is time to perform a BTR operation, the 64-word data block will be transferred to N7:410. The remaining branches of the rung then process this data.

The first branch examines the block identification code to see it the data contained in the block is status data. If the block code is set to –1, the status data is copied N10:200, the status data area.

The next four branches check to see if the block identification code corresponds to a read data block (1 to 4). If the block contains a valid code, the 60-word data set is copied to the user data file.

The last branch is very important, as it copies the BTW block identification code received from the module into the BTW block. This code is used to request data from the processor for the module.

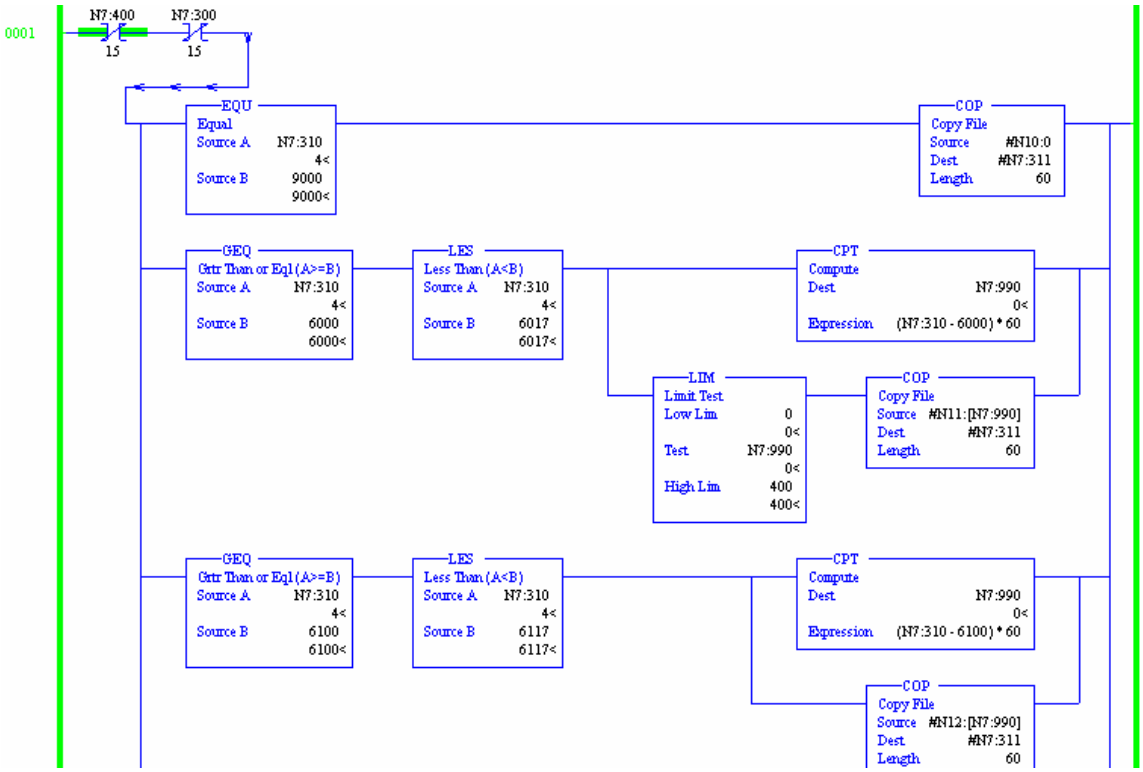
The next rung in the ladder logic is used to handle the BTW message blocks. An example rung is displayed below. As with the BTR rung, execution of this rung alternates between the BTR and BTW operation with the contacts in rung guaranteeing this mode of operation. The topmost branch of the rung is used to check if the module is requesting the configuration information (block 9000). The module requests this block each time a module restart operation occurs. The branch will execute when the block is requested and will copy the module configuration information into the BTW block.

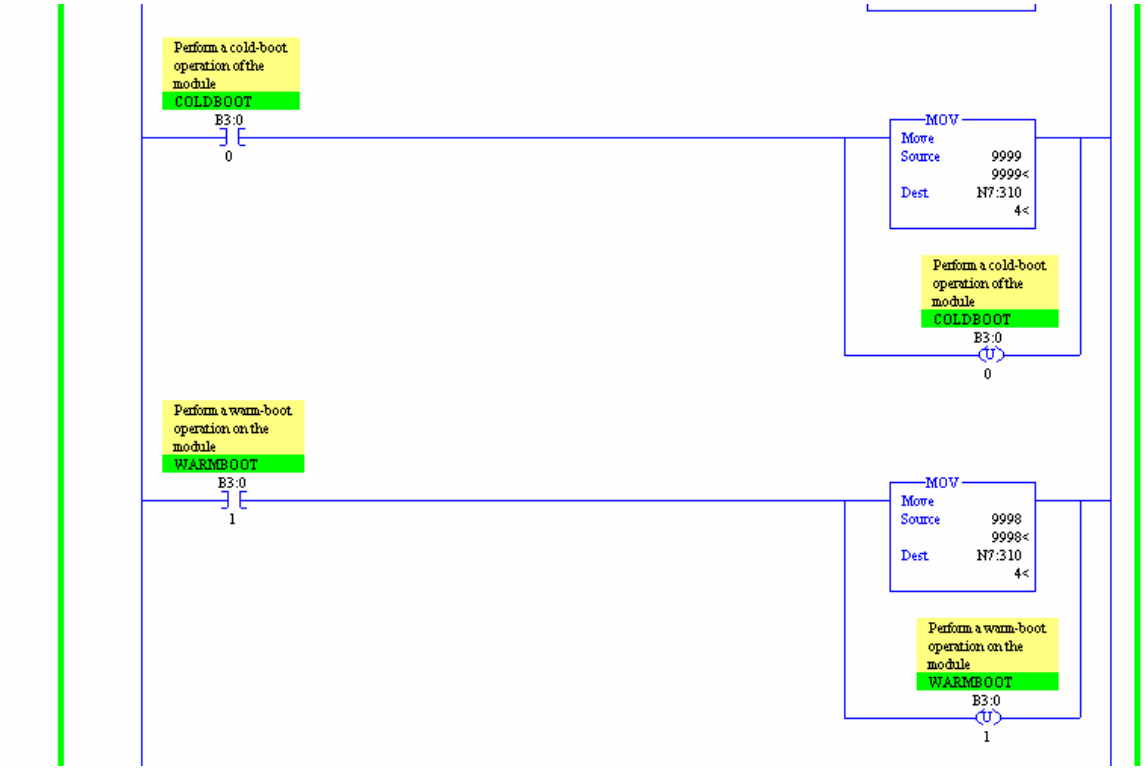
The next branch is used to copy the command list information to the module when requested. The example ladder uses only 5 commands for the port so only one block needs to be transferred. The two next branches are used to handle the request of other command lists blocks for port 1 and 2. The ladder logic should be written to handle all configuration blocks even though they may not all be used.

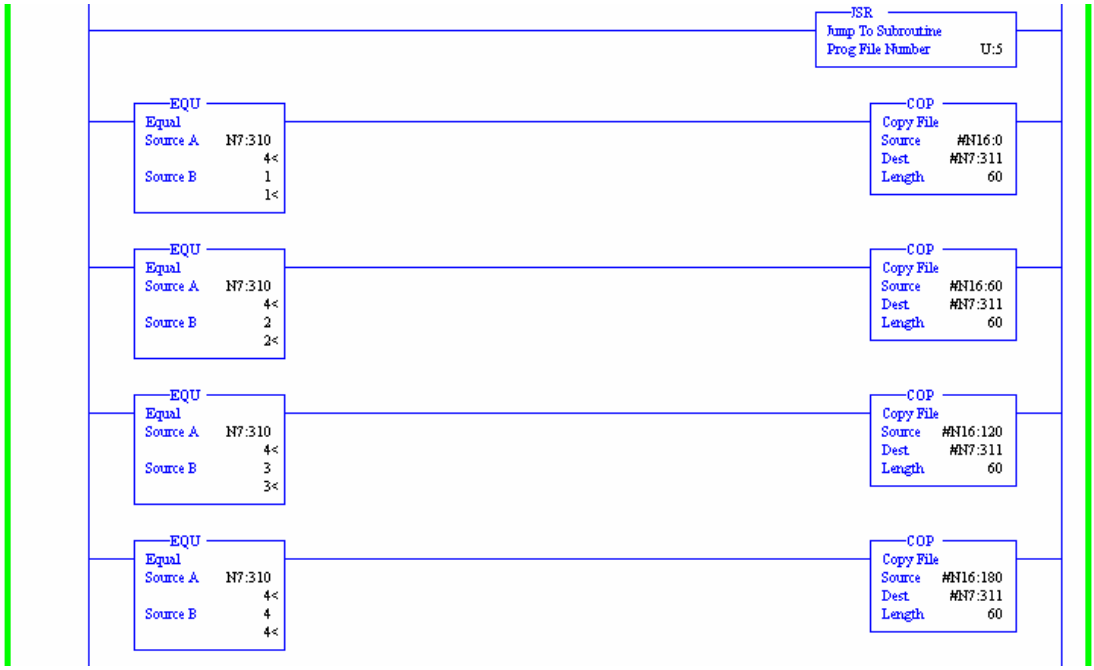
The next two branches are used to clear the cold-boot and warm-boot request bits in the processor. The block numbers for these special functions are set in the BTR rung above.

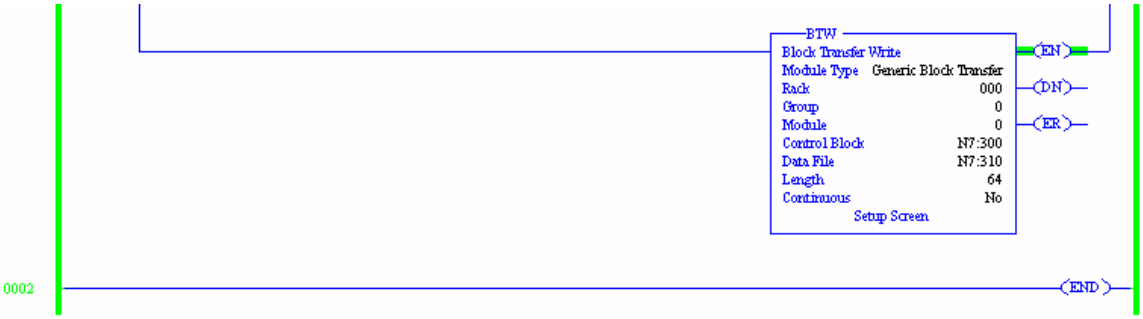
The next four branches are used to transfer the write data from the processor to the module. The branches determine the block to write (1 to 4) and copy the associated data into the BTW block.

The last branch of the rung is used to perform the BTW message operation. This operation will be recognized by the module and the data contained in the received BTW block will be processed by the module. If the data contained in the block is normal write data, the data will be placed in the module's internal database. If the block is a special control block (i.e., warm-boot block), the module will perform the selected operation.




Perform a cold-boot operation of the module
COLDBOOT
B3:0
0
MOV
Move
Source 9999
9999<
Dest N7:310
4<
Perform a cold-boot operation of the module
COLDBOOT
B3:0
U
0
Perform a warm-boot operation on the module
WARMBOOT
B3:0
1
MOV
Move
Source 9998
9998<
Dest N7:310
4<
Perform a warm-boot operation on the module
WARMBOOT
B3:0
U
1


JSR
Jump To Subroutine
Prog File Number U:5
EQU
Equal
Source A N7:310
4<
Source B 1
1<
COP
Copy File
Source #N16:0
Dest #N7:311
Length 60
EQU
Equal
Source A N7:310
4<
Source B 2
2<
COP
Copy File
Source #N16:60
Dest #N7:311
Length 60
EQU
Equal
Source A N7:310
4<
Source B 3
3<
COP
Copy File
Source #N16:120
Dest #N7:311
Length 60
EQU
Equal
Source A N7:310
4<
Source B 4
4<
COP
Copy File
Source #N16:180
Dest #N7:311
Length 60

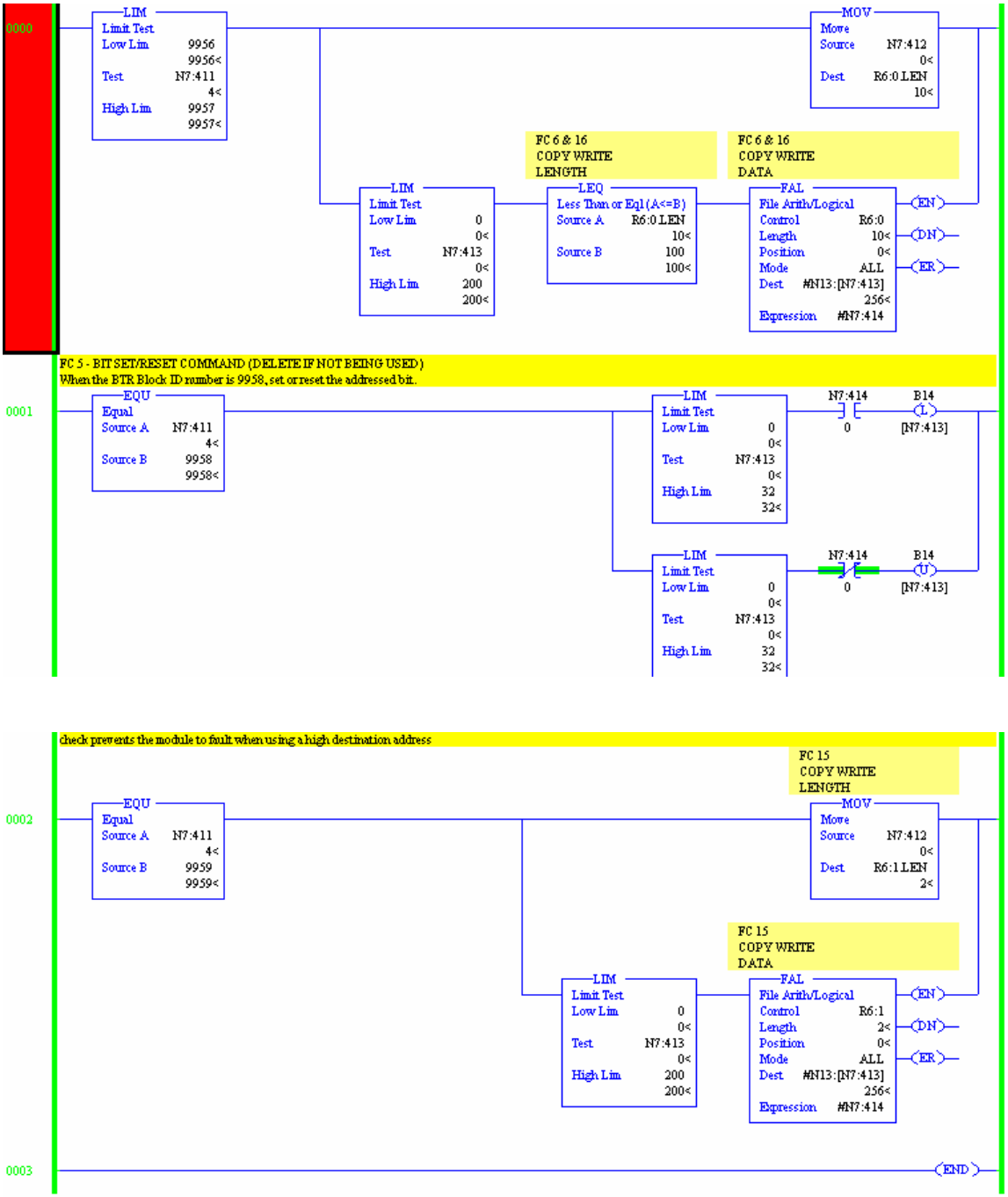


4.1.3 Pass-Through Routine

When a Slave port is configured to support the Pass-Through mode, Modbus command 5, 15, 6/16 which are addressed to the local address will be passed across the backplane for processing by the ladder logic. The ladder logic below is how to handle the pass-through routine.

The first rung is used to check if the pass-through mode is requested for the function code 6 and 16 (either integer (9956) or floating-point (9957)). The below example is only to handle integer value function code 6 and 16. In order for the ladder logic to handle floating-point value function code 6 and 16, similar logic can be used with an appended branch of COP instruction to move the data to a floating-point file.

The second rung is for function code 5, and the third is for the function code 15. The result from execute any of the function code the result will be located in N13 integer data file. In addition to the function 5, the written bit will also be appeared in B14 binary data file.

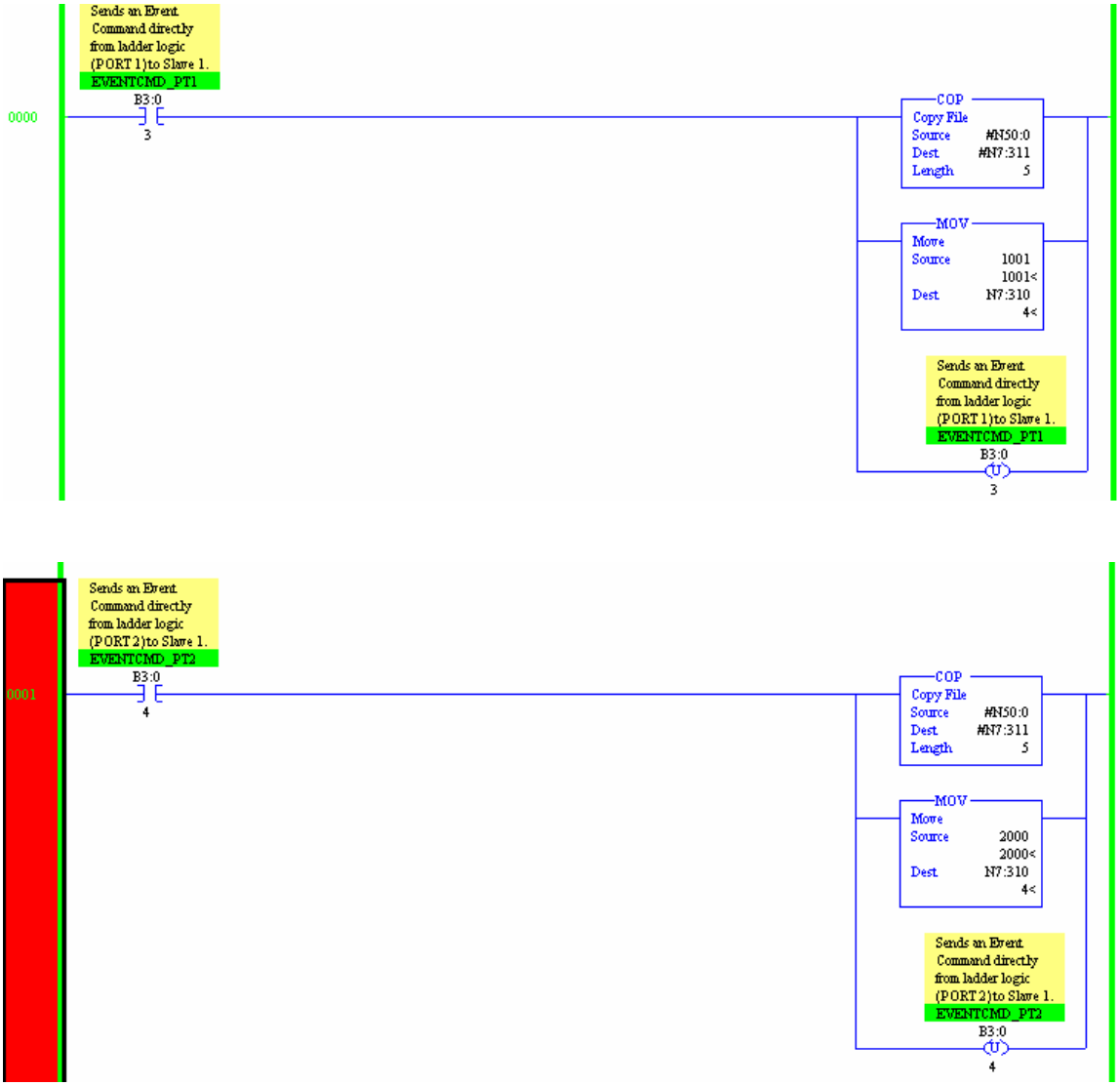


4.1.4 Command Control Routine

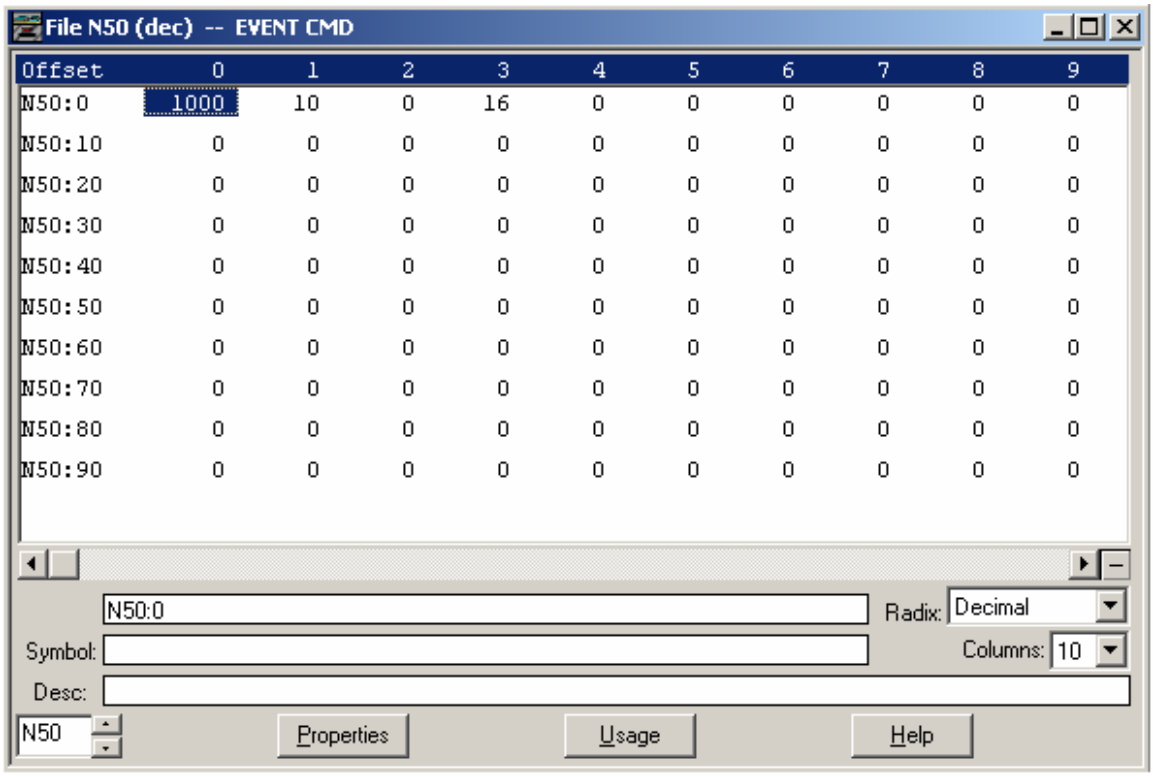
This routine describes how to use command control blocks. The following sections describe available command control routine features.

Event Command

The following rungs show how to create and send a command from ladder logic. The first rung shows a command from Port 1 (Block 1000) and the next rung shows a command for Port 2 (Block 2000).



In this case, N50:0 – N50:4 contains a command:



In this example, the module writes (Function 16)10 words from internal address 1000 to address 0 in slave 1 (since we are using blocks 1001 and 2001).

Command Console

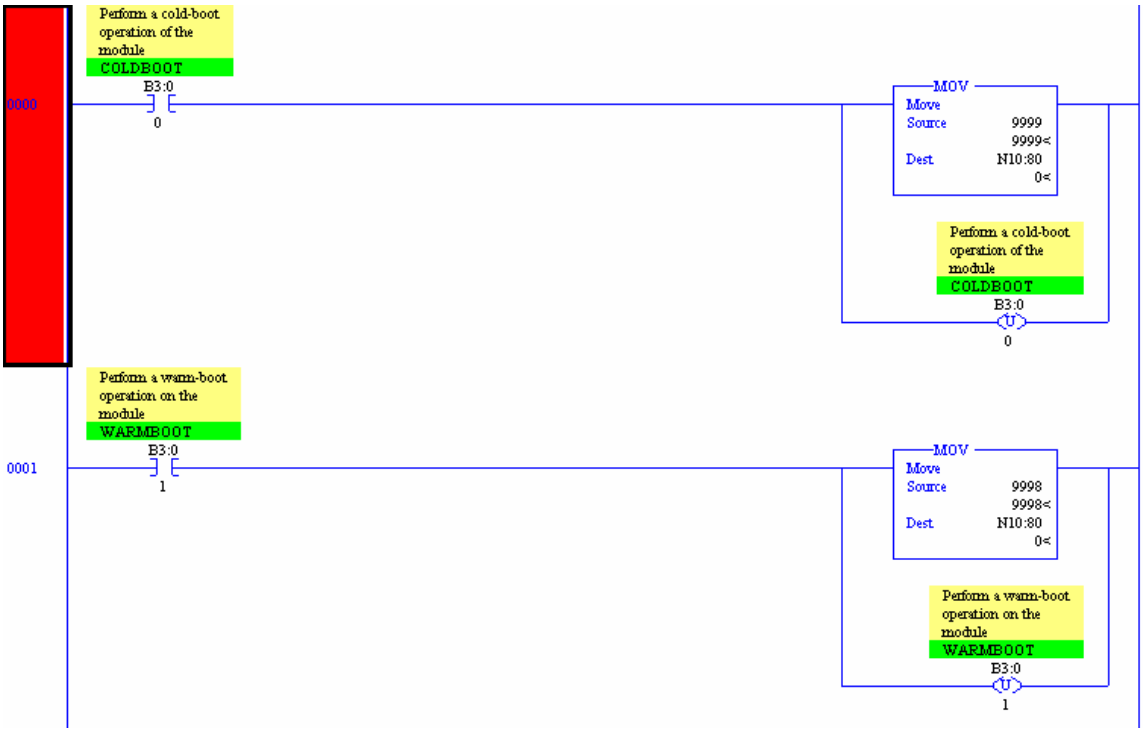
The next two rungs show how to enable commands that are currently disabled in the ladder logic:

4.2 Side-Connect Interface (Requires side-connect adapter)

When the side-connect interface is used, no ladder logic is required for normal data transfer. The module directly reads and writes information between the module and the processor using the user data files defined. The SC_DATA.TXT file contains the file number to be used for the configuration file. This file number and the module configuration determine the set of user data files required in the PLC.

In order to perform special control of the module (i.e., warm-boot operation), ladder logic is required. A reserved area in the configuration file is constantly monitored by the module (elements 80 to 139). If the module recognizes a valid control command code in element 80, it will use the data in the block to perform the requested operation. For example, to perform a warm-boot operation on the module, copy a value of 9998 into element 80 of the configuration file. The module should perform the warm-boot operation and reset the register value back to zero.

Boot



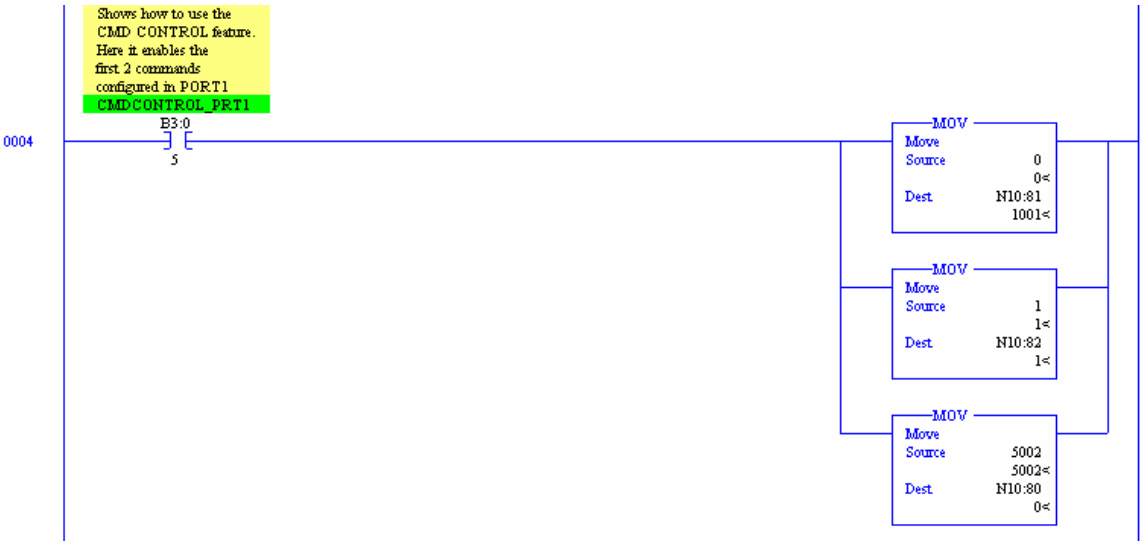
Event Command

Port 1 as the Master

Port 2 as the Master

Command Control

Port 1 as the Master

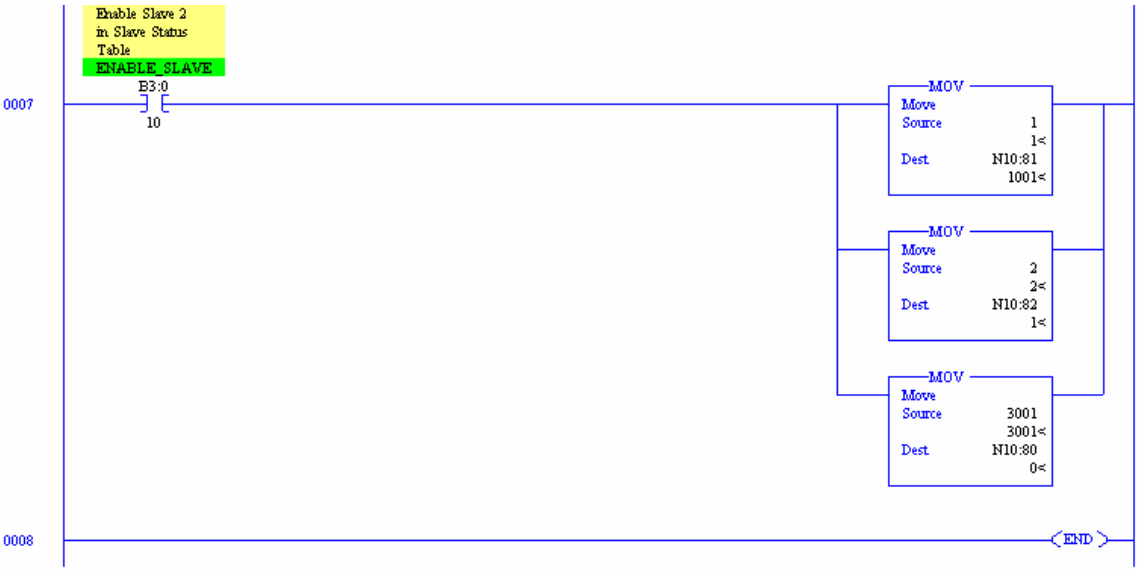


Port 2 as the Master

Slave Status Control

Disable a Slave (2)

Enable a Slave (2)



5 Diagnostics and Troubleshooting

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 PLC 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 MVI71-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 7600 to 7628 and at the location specified in the configuration. This data is transferred to the PLC 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

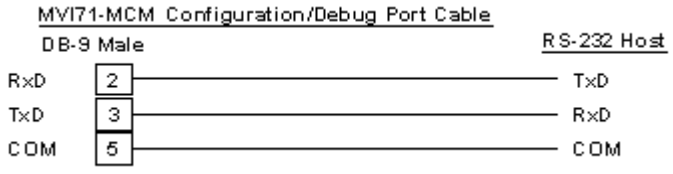
The hardware requirements to interface with the configuration/debugger port are not too stringent. A personal computer with a standard serial port should suffice. For optimal performance, the minimum is required:

80486 based processor (Pentium preferred)

1 megabyte of memory

At least one serial communications port available

Additionally, a null-modem cable is required 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 MVI71-MCM port 1 connector (top port, P1). The cable required is displayed below:



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 include 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	HyperTerminal
Windows 2000	HyperTerminal
Windows 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	57600
Parity	None
Data Bits	8
Stop Bits	1

5.1.3 Using the Port

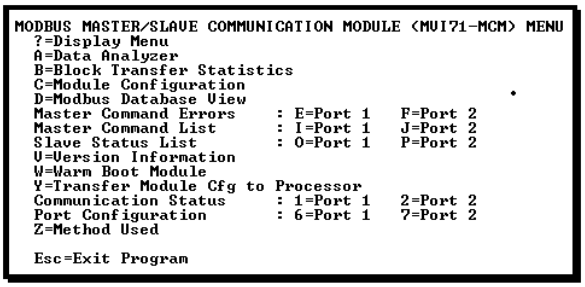
The following steps are required to interface with the configuration/debugger port:

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 (57.6K, 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

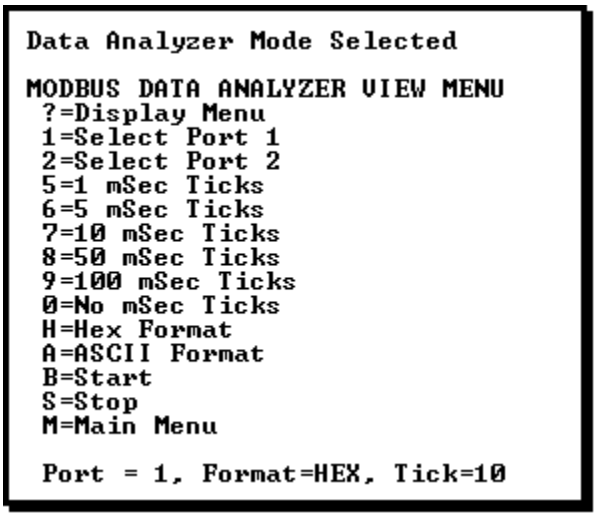
Features available through the use of the configuration/debug port on the MVI71-MCM module are all reached using single keystrokes on your computer. There is a single main menu and several sub-menus presented on the port. To view the current selections available, press the '?' key on your computer. If you are in main menu mode, the following menu will be displayed:



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 below:

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 will be displayed:



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

1=Select Port 1

This option is used to select Port 1 for analysis. Data displayed when in analyzer mode will relate to this port.

2=Select Port 2

This option is used to select the Port 2 for analysis. Data displayed when in analyzer mode will relate to this port.

5=1 mSec Ticks

This option is used to generate 1-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

6=5 mSec Ticks

This option is used to generate 5-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

7=10 mSec Ticks

This option is used to generate 10-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

8=50 mSec Ticks

This option is used to generate 50-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

9=100 mSec Ticks

This option is used to generate 100-millisecond timing marks on the display. This may help when determining communication-timing characteristics.

0=No mSec Ticks

This option is used to turn the display of timing marks off.

H=Hex Format

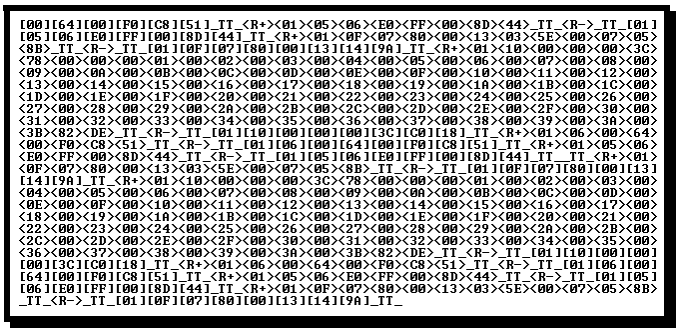
This option is used to select the display of the data in hexadecimal format. This format is most useful when viewing MODBUS protocol messages.

A=ASCII Format

This option is used to select the display of the data in ASCII format. This format is most useful when viewing ASCII messages.

B=Start

This option is used to start the data analyzer. After the key is pressed, all data transmitted and received on the currently selected port will be displayed. An example display is shown below:



Special characters used in the display are as follows:

[ ]	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 is used to stop 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 is used to display the configuration and statistics of the backplane data transfer operations. After selecting this option, the following will be 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 : 240     Max Blocks : 1       Last : 0
  READ DATA TRANSFER:
        Start : 1000     Count : 240     Max Blocks : 1       Last : 0
  BLOCK COUNTS:
        Retry : 0        Failed: 0       Fail Cnt: 0
        Read  : 28905    Write : 28906   Parsing : 0
        Error : 0        Event : 0       Command : 0
```

C = Module Configuration

This option displays the general module configuration information for the MVI71-MCM module. After selecting the option, the following screen will be displayed for the block transfer set up:

```
MVI71-MCM, ProSoft Technology, Inc.

  DATABASE:
        Err/Stat Blk Pointer : 2500
  BLOCK TRANSFER:
        READ  -- Start : 0        Count : 240      Max : 4
        WRITE -- Start : 1000     Count : 240      Max : 4
        FAIL COUNT     : 0
```

When the module is configured for the side-connect interface, the following will be displayed:

```
MODULE CONFIGURATION (SIDE-CONNECT INTERFACE):

MVI71-MCM, ProSoft Technology, Inc.

  DATABASE:
    Err/Stat Blk Pointer : 2500
  SIDE-CONNECT: Configuration/Status/Control File: 10
    READ  -- Start : 0        Count : 240    File(s): 1    (15-15)
    WRITE -- Start : 1000     Count : 240    File(s): 1    (16-16)
    FAIL COUNT     : 0
```

D = Database View

Selection of this menu option places the program in database view menu mode. This mode of operation is used to display the module's internal database values. To view the menu options available in this mode, press the '?' key and the following menu will be displayed:

```
Modbus DB Menu Selected

MODBUS DATABASE VIEW MENU
 ?=Display Menu
 0-9=Display 0-9000
 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 sections below:

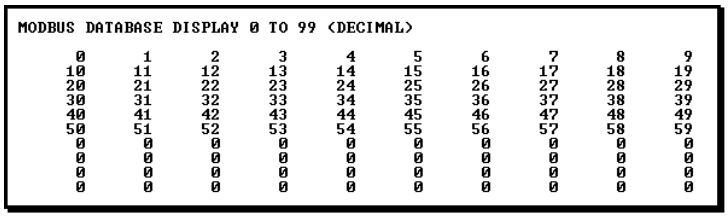
0-9 = Register pages 0-9000

This menu option is used to jump to a specific set of registers in the database and display 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
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 is used to display the current page of 100 registers in the database. Example output of the database display is shown below:



- = Back 5 Pages

This menu option is used to skip the previous 500 registers of data for viewing and display the data.

P = Previous Page

This menu option is used to select and display the previous 100 registers of data.

+ = Skip 5 Pages

This menu option is used to skip 500 registers of data and to display the new page of data.

N = Next Page

This menu option is used to select the next 100 registers of data for viewing and displays the data.

D = Decimal Display

This menu option is used to display the data on the current page in decimal format.

H = Hexadecimal Display

This menu option is used to display the data on the current page in hexadecimal format.

F = Float Display

This menu option is used to display the data on the current page in floating-point format. The program assumes that the values are aligned on even register boundaries. If floating-point values are not aligned as such, they will not be displayed properly.

A = ASCII Display

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

M = Main Menu

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

E and F = Master Command Errors (Ports 1 and 2)

Selection of these menu options places the program in master command error menu mode for the specified port. This mode of operation is used to display 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 will be displayed:

```
Port 1 Cmd Err Menu Selected

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:

S = Show Again

This option is used to display the current page of master command error/status data. After selecting the option, the following screen will be displayed.

```
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 a complete listing and interpretation of the codes displayed.

- = Back 2 Pages

This option is used to skip back 20 commands and display the data.

P = Previous Page

This option is used to display the previous page of data.

+ = Skip 2 Pages

This option is used to skip past the next 20 commands and display the data.

N = Next Page

This option is used to display the next page of master command list error/status data.

D = Decimal Display

This option is used to change the display of the data to decimal format.

H = Hexadecimal Display

This option is used to change the display of error/status data to hexadecimal format.

M = Main Menu

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

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

Selection of 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 will be displayed:

```
Port 1 Command List Menu Selected

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 is used to display the current page of master commands. Ten commands are displayed on each page as shown below:

```
ENABLE MBREG POLLINT COUNT SWAP NODE FUNC ADDRS
     1  1000       0    60    0    1   16     0
     1  1239       0     1    0    1    6   100
     1 16016       0     1    0    1    5  1760
     1 17504       0    19    0    1   15  1920
     0     0       0     0    0    0    0     0
     0     0       0     0    0    0    0     0
     0     0       0     0    0    0    0     0
     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 is used to display the master command list data after skipping the previous 50 commands.

P = Previous Page

This menu option is used to display the previous page of master command list data.

+ = Skip 5 Pages

This menu option is used to display the master command list data after skipping the next 50 commands.

N = Next Page

This menu option is used to display the next page of master command list data.

M = Main Menu

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

<u>O and P = Slave Status List (Port1 and 2)</u>

Selection of 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.

```
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 is used to view the current version of the software for the module and other important values. After selecting the option, the following will be displayed:

```
VERSION INFORMATION:

  MODBUS MASTER/SLAVE COMMUNICATION MODULE (MVI71-MCM)
  (c) 1999, 2000, 2001, ProSoft Technology, Inc.

  PRODUCT NAME CODE          : MCM7
  SOFTWARE REVISION LEVEL    : 1.00
  OPERATING SYSTEM REVISION : 0302
  RUN NUMBER                 : 2701
  PROGRAM SCAN COUNTER       : 19292
  BACKPLANE DRIVER VERSION   :  0.01
  BACKPLANE API VERSION      :  1.09
  SIDE-CONNECT API VERSION   :  0.02
  MODULE NAME : MVI71 ProSoft Technology, Inc.
  VENDOR ID   : 309       DEVICE TYPE    : 12
  PRODUCT CODE: 13        SERIAL NUMBER : 00000459
  REVISION    : 1.01
```

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 selected when a warm-boot operation is required of the module. This request is usually made after configuration changes are set in the PLC5 processor's data registers to implement the changes. After selecting the option, the following will be displayed:

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

Warm booting module....

Reloading Program Values....
Side-connect interface used (N10).

Read Configuration...
MODBUS MASTER/SLAVE COMMUNICATION MODULE (MVI71-MCM)
       (c) 1999-2002, ProSoft Technology, Inc.

       PRODUCT NAME CODE          : MCM7
       SOFTWARE REVISION LEVEL    : 1.00
       OPERATING SYSTEM REVISION : 0302
       RUN NUMBER                 : 2701

Press ? for menu help.
```

Y = Transfer Module Cfg to Processor

This option is used to transfer the current configuration data in the module to the PLC5 processor. Ladder logic must exist in the processor to successfully implement this option. After selecting the option, the following will be displayed for successful operation:

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

Sending configuration to processor....

Send complete....Error code = 0.

Warm booting module....

Reloading Program Values....
Side-connect interface used (N10).

Read Configuration...
MODBUS MASTER/SLAVE COMMUNICATION MODULE (MVI71-MCM)
      (c) 1999-2002, ProSoft Technology, Inc.

      PRODUCT NAME CODE         : MCM7
      SOFTWARE REVISION LEVEL   : 1.00
      OPERATING SYSTEM REVISION : 0302
      RUN NUMBER                : 2701

Press ? for menu help.
```

If the operation is not successful, an error code will be 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 will perform a warm-boot operation to read in the new data.

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

These options are used to display the communication status and statistics of the specified port. This information can be informative when trouble-shooting network problems. After selecting the option, the following information will be displayed:

```
PORT 1 MODBUS STATUS:
  Enabled : Y
  Retries : 0       Cur Cmd : 0       State   : 2
  ComState: 4       Cur Err : -1      Last Err: -1

  Number of Command Requests: 10845
  Number of Cmd Responses   : 10844
  Number of Command Errors  : 0
  Number of Requests        : 10845
  Number of Responses       : 10844
  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 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
  MASTER SETUP:
        Command Count    : 4        Cmd Delay: 0        Cmd Offset : 3000
        Response Timeout: 1000     Retries  : 0        Delay Count: 0
  COMMUNICATION PARAMETERS:
        Protocol: 0 (Modbus RTU)
        Baud: 38400    Parity: NONE    Databits: 8   Stopbits: 1
        RTS On: 0       RTS Off: 1       Use CTS Line: N
```

Esc = Exit Program

This option is used to exit the program and to display 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 will cease operation. Data will no longer be transferred between the ports and the module and between the PLC5 processor and the module. This might cause an upset to a currently running process.

5.2 LED Status Indicators

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

ProSoft Module	Color	Status	Indication
P1	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.
P2	Green	On	Data is being transferred between the module and the MODBUS network on Port 1.
		Off	No data is being transferred on the port.
P3	Green	On	Data is being transferred between the module and the MODBUS network on Port 2.
		Off	No data is being transferred on the port.
APP	Amber	Off	The MVI71-MCM is working normally.
		On	The MVI71-MCM module program has recognized a communication error on one of its 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 ACT/FLT LED 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.

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 ACT/FLT 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. Turn off the power to the rack.
2. Remove the card from the rack and re-insert the card in the rack and turn the power on.
3. Verify the configuration data being transferred to the module from the PLC5 processor.

If the module's ACT/FAULT LED does not turn green, make sure the module is inserted completely into the rack. If this does not cure the problem, contact the factory.

5.2.2 Troubleshooting

In order to assist in the troubleshooting of the module, the following table has been put together to assist you. Please use the following to help in using the module, but 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.

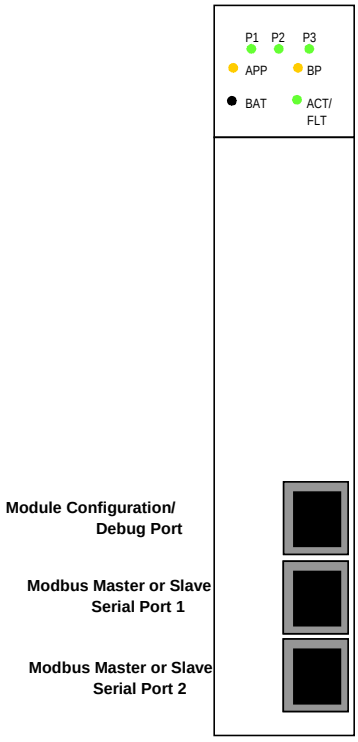
Problem Description	Steps to take
Processor Fault	Be sure that the module is plugged into the slot that has been configured for the MVI71-MCM module. Assure that the slot in the rack configuration has been set up correctly:
Processor I/O LED flashes	This indicates there is a problem with backplane communications. Be certain this and all modules in the rack are configured in the processor.

Problem Description	Steps to take
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 or the side-connect configuration file is set correctly. 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. Do not remove the module from the rack when power is applied to the rack.

6 Cable Connections

The MVI71-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

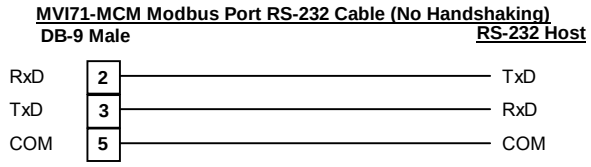
The MVI71-MCM module has two physical Modbus connectors with a RJ45 plug located on the front of the module.

6.1.1 Connecting the Cable to the Connector

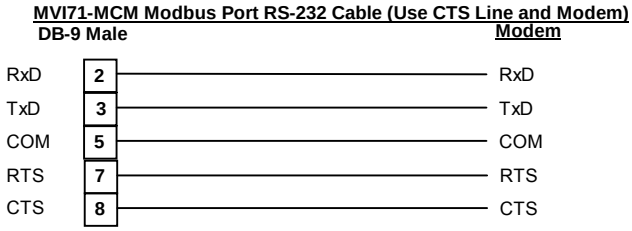
ProSoft provides two RJ45 to male DB-9 pigtails to permit simpler interfacing to other devices. The module's Modbus ports can be configured to operate in RS-232, RS-422 or RS-485 mode. The interface to be associated with a port is set with jumpers on the module. There is a jumper for each of the two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

6.1.1.1. 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:

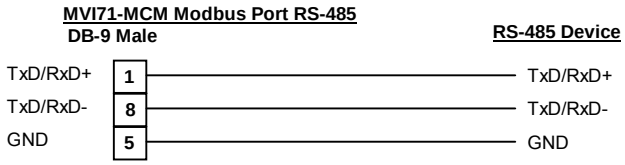


The RTS line is controlled by the RTS on and off parameters set for the port. If the CTS line is used (usually only required for half-duplex modems), the RTS and CTS lines must either be connected together or connected to the modem. The following diagram displays the cable required when connecting the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



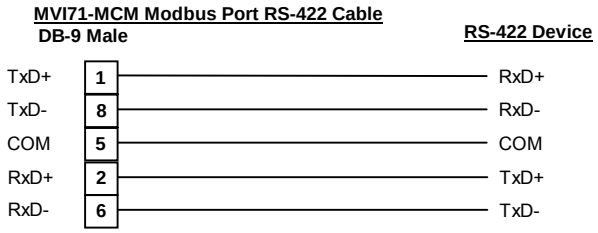
6.1.1.2. RS-485

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:



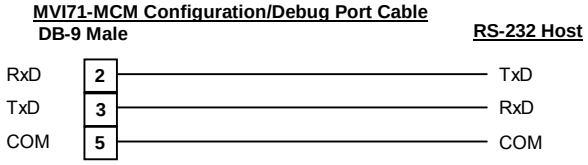
6.1.1.3. RS-422

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:



6.2 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 - MVI71-MCM Database Definition

This appendix contains a listing of the internal database of the MVI71-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
5000-5006	45001	45007	Backplane Configuration	10
5010-5039	45011	45040	Port 1 Setup	30
5040-5069	45041	45070	Port 2 Setup	30
5200-6199	45901	46900	Port 1 Commands	1000
6400-7399	46401	47400	Port 2 Commands	1000
7600-7632	47601	47633	Misc. Status Data	33
7800-7999	47801	4800	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 E contains a table of Configuration Error Codes.

Appendix B – MVI71-MCM Status Data Definition

This appendix contains a description of the data contained in the status data block. This data is transferred from the module to the processor either through a BTR block –1 (block transfer interface) or directly into the configuration file (side-connect interface).

Offset	Content	Description
0	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
1	Product Code	These two registers contain the product code of "DFCM"
2	Product Version	These two registers contain the product version for the current running software.
3-4	Operating System	These two registers contain the month and year values for the program operatin system.
5-6	Run Number	These two registers contain the run number value for the currently running software.
7-8	Port 1 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
9-10	Port 1 Command List Response	This field contains the number of slave response messages received on the port.
11	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.
12	Port 1 Requests	This field contains the total number of messages sent out of the port.
13	Port 1 Responses	This field contains the total number of messages received on the port.
14	Port 1 Errors Sent	This field contains the total number of message errors sent out of the port.
15	Port 1 Errors Received	This field contains the total number of message errors received on the port.
16	Port 2 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
17	Port 2 Command List Response	This field contains the number of slave response messages received on the port.
18	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.
19	Port 2 Requests	This field contains the total number of messages sent out the port.
20	Port 2 Responses	This field contains the total number of messages received on the port.
21	Port 2 Errors Sent	This field contains the total number of message errors sent out of the port.
22	Port 2 Errors Received	This field contains the total number of message errors received on the port.

Offset	Content	Description
23	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
24	Write Block Count	This field contains the total number of write blocks transferred from the processor to the module.
25	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
26	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
27	Command Block Count	This field contains the total number of command blocks received from the processor.
28	Error Block Count	This field contains the total number of block errors recognized by the module.
29	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.
30	Port 1 Last Error	For a slave port, this field contains the value of the last error code returned. For a master port, this field contains the index of the command with an error.
31	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.
32	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 – MVI71-MCM Configuration Data Definition

This appendix contains listings of the MVI71-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 PLC processor when the module first initializes.

Backplane Setup

Write Block Offset	Internal Database Register	Content	Description
1	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.
2	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.
3	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.
4	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.
5	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.
6	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.

Port 1 Setup

Write Block Offset	Internal Database Register	Content	Description
7	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.
8	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 byte swapping (3).

Write Block Offset	Internal Database Register	Content	Description
9	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.
10	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.
11	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.
12	5015	Protocol	This parameter specifies the Modbus protocol to be used on the port. Valid protocols are: 0 = Modbus RTU and 1 = Modbus ASCII.
13	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, 384 (38400), 576 (57600), 115 (115200).
14	5017	Parity	This is the parity code to be used for the port. Values are None, Odd, Even.
15	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.
16	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.
17	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.
18	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.
19	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.
20	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).
21	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).
22	5025	Bit Input 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.

Write Block Offset	Internal Database Register	Content	Description
23	5026	Word Input 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.
24	5027	Bit Output 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.
25	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.
26	5029	Command Count	This parameter specifies the number of commands to be processed by the Modbus master port.
27	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.
28	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.
29	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.
30	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.
31	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.
32	5035	Guard Band	Timeout. A value of 0 uses the default baud rate or you can set a timeout value in milliseconds (0 to 65535)
33	5036	Spare	Spare

Port 2 Setup

Write Block Offset	Internal Database Register	Content	Description
34	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.

Write Block Offset	Internal Database Register	Content	Description
35	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 byte swapping (3).
36	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.
37	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.
38	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.
39	5045	Protocol	This parameter specifies the Modbus protocol to be used on the port. Valid protocols are: 0 = Modbus RTU and 1 = Modbus ASCII.
40	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.
41	5047	Parity	This is the parity code to be used for the port. Values are None, Odd, Even.
42	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.
43	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.
44	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.
45	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.
46	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.
47	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).
48	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).

Write Block Offset	Internal Database Register	Content	Description
49	5055	Bit Input Offset	This parameter specifies the offset word 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 the first bit in register 150 in the database.
50	5056	Word Input 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.
51	5057	Bit Output Offset	This parameter specifies the offset word 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 the first bit at register 100 in the database.
52	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.
53	5059	Command Count	This parameter specifies the number of commands to be processed by the Modbus master port.
54	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.
55	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.
56	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.
57	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.
58	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.
59	5065	Guard Band	Timeout. A value of 0 uses the default baud rate or you can set a timeout value in milliseconds (0 to 65535)
60	5066	Spare	Spare

Port 1 Commands

Register	Content	Description
5200 – 5207	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.
5210 – 5217	Command #2	Command #2 data set
-	-	-
6190 – 6197	Command #100	Command #100 data set

Port 2 Commands

Register	Content	Description
6400 – 6407	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.
6410 – 6417	Command #2	Command #2 data set
-	-	-
7390 – 7397	Command #100	Command #100 data set

Misc. Status

Register	Content	Description
7600	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
7601 - 7602	Product Code	These two registers contain the product code of “MCM”.
7603 - 7604	Product Version	These two registers contain the product version for the current running software.
7605 - 7606	Operating System	These two registers contain the month and year values for the program operating system.
7607 - 7608	Run Number	These two registers contain the run number value for the currently running software.
7609	Port 1 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
7610	Port 1 Command List Response	This field contains the number of slave response messages received on the port.
7611	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.
7612	Port 1 Requests	This field contains the total number of messages sent out of the port.
7613	Port 1 Responses	This field contains the total number of messages received on the port.
7614	Port 1 Errors Sent	This field contains the total number of message errors sent out of the port.
7615	Port 1 Errors Received	This field contains the total number of message errors received on the port.

Register	Content	Description
7616	Port 2 Command List Requests	This field contains the number of requests made from this port to slave devices on the network.
7617	Port 2 Command List Response	This field contains the number of slave response messages received on the port.
7618	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.
7619	Port 2 Requests	This field contains the total number of messages sent out the port.
7620	Port 2 Responses	This field contains the total number of messages received on the port.
7621	Port 2 Errors Sent	This field contains the total number of message errors sent out the port.
7622	Port 2 Errors Received	This field contains the total number of message errors received on the port.
7623	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
7624	Write Block Count	This field contains the total number of write blocks transferred from the module to the processor.
7625	Parse Block Count	This field contains the total number of blocks successfully parsed that were received from the processor.
7626	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
7627	Command Block Count	This field contains the total number of command blocks received from the processor.
7628	Error Block Count	This field contains the total number of block errors recognized by the module.
7629	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.
7630	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.
7631	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.
7632	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.

Command Control

Register	Content	Description
7800	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.
7801	Command Data	Not Used
-	-	-
7999	Command Data	Not Used

Appendix D – MVI71-MCM Command Error Codes

The following tables list the MVI71-MCM Command Error Codes:

Standard Modbus Protocol Errors

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

Module Communication Error Codes

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

Command List Entry Errors

Code	Description
-41	Invalid enable code
-42	Internal address > maximum address
-43	Invalid node address (< 0 or > 255)
-44	Count parameter set to 0
-45	Invalid function code
-46	Invalid swap code

Appendix E -- Configuration Error Codes

The configuration update procedure starts when the module receives a configuration block (Write Block ID = 9000):

Offset	Description	Length
0	9000	1
1-6	Backplane Setup	6
7-31	Port 1 Configuration Errors	25
32-56	Port 2 Configuration Errors	25
57-63	Spare	7

The module then responds with a Read Block (ID = -2) with any possible errors:

BTR

Offset	Description	Length
0	-2	1
1	9000	1
2	Module Configuration Errors	1
3	Port 1 Configuration Errors	1
4	Port 2 Configuration Errors	1
5 – 63	Spare	58

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

Bit	Description	Value
0	Write block start value is greater than the database size.	0x0001
1	Write block start value is less than zero.	0x0002
2	Write block count value is less than zero.	0x0004
3	Write block count + start is greater than the database size.	0x0008
4	Read block start value is greater than the database size.	0x0010
5	Read block start value is less than zero.	0x0020
6	Read block count value is less than zero.	0x0040
7	Read block count + start is greater than the database size.	0x0080
8		0x0100
9		0x0200
10		0x0400
11		0x0800
12		0x1000
13		0x2000
14		0x4000

Bit	Description	Value
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

Appendix F – Product Specifications

General Specifications

The MVI71-MCM module acts as a gateway between the Modbus network and the Allen-Bradley backplane. The data transfer from the PLC 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 PLC 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
Floating-Point Support		

Slave Functional Specifications

The MVI71-MCM module accepts Modbus function code commands of 1, 2, 3, 4, 5, 6, 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 PLC processor. The module can be configured to pass write commands (functions 5, 6, 15, and 16,) 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 MVI71-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 PLC 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 PLC 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 (PLC backplane technology).

- PLC 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

PLC Interface

- Operation via simple ladder logic
- Complete set up and monitoring of module through RSLogix 5 software
- PLC backplane interface via I/O access
- All data related to the module is contained in data registers with defined objects to ease in the configuration, monitoring and interfacing with the module
- Module configuration and communication configuration data is transferred to the MVI71-MCM via a predefined user data type in the processor.

Hardware Specifications

The MVI71-MCM module is designed by ProSoft Technology and incorporates licensed technology from Allen-Bradley (PLC 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)

Appendix G - Quick Start Guide

This section lists the steps required to start module operation. The goal of this section is to provide the user with the necessary actions steps to get the module up and running, however, it is essential that the user read all of this manual in order to fully understand how the module operates.

1. Choose the slot where the module will be located in the PLC rack.
2. Open the sample ladder file using RSLogix 5.
3. Make sure the BTR and BTW instructions match the MVI71-MCM location.
4. The sample ladder Block Transfer instructions are configured as:
5. Rack=0 , Group=0,Module=0
6. Verify that the data files used in the sample ladder will not interfere with the data files you are using in your application. If necessary, edit the sample ladder in order to use a different data file number for the same task.
7. The sample ladder uses the following data files:

   B3: Auxiliary Bits

   N9: Status Data

   N10:Configuration Data

   N11: Port 1 Modbus Commands (if Port 1 is configured as a Modbus Master)

   N12: Port 2 Modbus Commands (if Port 2 is configured as a Modbus Master)

   N13: Pass-Thru

   B14: Pass-Thru Bit (used to bit write commands when using pass-thru)

   N15: Block Transfer Read Data

   N16: Block Transfer Write Data

   N50: Event Commands

8. Edit the Configuration Data to match your application. The configuration data contains the port configuration and block transfer parameters. In the sample ladder the Configuration Data is located at data file N10 and it is structured according to the table listed below. Use this table to help you configure the module:

Address	Description	Value
N10:0	Write Data Start	
N10:1	Write Data Count	
N10:2	Read Data Start	

Address	Description	Value
N10:3	Read Data Count	
N10:4	Backplane Fail Count	
N10:5	Error Start Address	
N10:6	Port 1 Enable	
N10:7	Port 1 Type	
N10:8	Port 1 Float Flag	
N10:9	Port 1 Float Start	
N10:10	Port 1 Float Offset	
N10:11	Port 1 Protocol	
N10:12	Port 1 Baud Rate	
N10:13	Port 1 Parity	
N10:14	Port 1 Data Bits	
N10:15	Port 1 Stop Bits	
N10:16	Port 1 RTS ON	
N10:17	Port 1 RTS OFF	
N10:18	Port 1 Minimum Response	
N10:19	Port 1 Use CTS	
N10:20	Port 1 Slave ID	
N10:21	Port 1 Bit Input Offset	
N10:22	Port 1 Word Input Offset	
N10:23	Port 1 Bit Output Offset	
N10:24	Port 1 Holding Register Offset	
N10:25	Port 1 Command Count	
N10:26	Port 1 Minimum Command Delay	
N10:27	Port 1 Command Error List Address	
N10:28	Port 1 Command Response Timeout	
N10:29	Port 1 Retry Count	
N10:30	Port 1 Error Delay Count	
N10:31	Port 2 Enable	
N10:32	Port 2 Type	
N10:33	Port 2 Float Flag	
N10:34	Port 2 Float Start	
N10:35	Port 2 Float Offset	
N10:36	Port 2 Protocol	
N10:37	Port 2 Baud Rate	
N10:38	Port 2 Parity	
N10:39	Port 2 Data Bits	
N10:40	Port 2 Stop Bits	
N10:41	Port 2 RTS ON	
N10:42	Port 2 RTS OFF	
N10:43	Port 2 Minimum Response	
N10:44	Port 2 Use CTS	

Address	Description	Value
N10:45	Port 2 Slave ID	
N10:46	Port 2 Bit Input Offset	
N10:47	Port 2 Word Input Offset	
N10:48	Port 2 Bit Output Offset	
N10:49	Port 2 Holding Register Offset	
N10:50	Port 2 Command Count	
N10:51	Port 2 Minimum Command Delay	
N10:52	Port 2 Command Error List Address	
N10:53	Port 2 Command Response Timeout	
N10:54	Port 2 Retry Count	
N10:55	Port 2 Error Delay Count	

9. In case a port is configured as a Modbus Master, you will create Modbus Commands for that port. For example, if Port 1 is configured as Master N11:0 file would be used as:

Start	End	Description
N11:0	N11:9	Command 1
N11:10	N11:19	Command 2
N11:20	N11:29	Command 3
N11:30	N11:39	Command 4
N11:40	N11:49	Command 5
…	…	…

Each Modbus Command has the following structure:

Offset	Description
0	Enable
1	Internal Address
2	Poll Interval
3	Command Count
4	Swap Code
5	Slave Address
6	Modbus Function
7	Destination Address
8	Spare
9	Spare

10. Copy the new ladder file to your existing ladder application.
11. Modify the communication set up jumper in the back of the MVI71-MCM, selecting the communication type for each port (485, 422 or 232).
12. Insert the module in the rack and connect the port(s) to the Modbus network. Make sure the connection correctly follows the wiring diagrams shown in this manual.
13. Turn on the rack power and download the new ladder file to the PLC.

Frequently Asked Questions

What is the MVI71-MCM?

The MVI71-MCM is a single slot solution that allows Modbus devices to easily interface with a PLC processor. The module has 2 ports individually configurable either as a Modbus Master or Modbus Slave.

The module has an internal database with 5000 registers that are accessible to external Modbus devices as well as accessible to the PLC through the rack backplane. Ladder logic is required, among other tasks, to access the MVI71-MCM internal database.

Why does the module have to break down the data in blocks of 64 words?

Because of a limitation of the PLC backplane that allows only 64 words at a time.

How do I make the module start working?

Please refer to the Quick Start Guide in Appendix G.

How do I read and write data between the internal MVI71-MCM database and the PLC?

Ladder logic is required to read and write blocks from the MVI71-MCM. Initially the user must configure the Read and Write Data areas inside the MVI71-MCM database. It creates 2 regions: the Read Data area which is constantly read from the module database to the PLC, and the Write Data area which is constantly written from the PLC ladder logic to the MVI71-MCM database.

Block Transfer instructions are used to transfer data: the BTR instruction is used to read data from the module and the BTW instruction is used to write data to the module.

If using side connect (requires side connect adapter) option, ladder logic is not required to transfer data to the MVI71-MCM.

What is the procedure to change the module's configuration?

A simple boot operation will force the module to request configuration data. The ladder logic can force the module to perform a Cold Boot or Warm Boot.

How do I set a Port Modbus Master command?

You can create Modbus commands using a PLC data file that is copied to the BTW data file by the ladder logic. The sample ladder logic uses:

N11: Port 1 Master Commands

N12: Port 2 Master Commands

Ladder logic is necessary to copy the commands to the MVI71-MCM during the boot operation (the sample ladder already handles it).

The Modbus Master Command is not working, what should I do?

Check the Command Error Menu in the debug port and write down the code associated with the command index. Look for the meaning of the error code in the Appendix and ensure that the parameters entered are valid.

If there is no error code in the Command Error Menu, the Modbus command could refer to a different location than the one considered.

What is the Pass-Through feature?

An MVI71-MCM slave port can be configured for pass-through mode. This causes every write command to write directly to the PLC, without modifying the MVI71-MCM internal database. In order to accomplish this, every time a slave pass-through port receives a write command, the module generates specific blocks that allows ladder logic to handle each write command internally (please refer to the Pass-Through section).

How can I check a Slave's status connected to an MVI71-MCM Master port?

The debug port allows the user to monitor every slave attached to a master port.

Should I use the sample ladder logic?

Yes, the sample ladder should always be used as the source for your application. You may simply copy and paste the ladder logic to your current application.

Is it possible to control the Master Modbus Commands from ladder logic?

Yes, the ladder logic can create and send Modbus commands (event command) or enable commands that are currently disabled in the Master Command List (command control).

Please refer to the User Manual for more information.

How is the MVI71-MCM database divided into Discrete Output, Discrete Input, Input Registers, and Holding Registers?

The MVI71-MCM uses the same database (0-4999) for all of these data types. The only difference is how you address each point. For Discrete Output or Discrete Input, use a bit addressing (1 bit) and for Input Registers or Holding Registers, use a word addressing (16 bits).

Support, Service, and Warranty

Technical Support

ProSoft Technology 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

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

Module Service and Repair

The MVI71-MCM card 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 card may require repair.

When purchased from ProSoft Technology, the module has a one-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 card for repair, obtain an RMA number from ProSoft

Technology. Please call the factory for this number and display the number prominently on the outside of the shipping carton used to return the card.

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 one (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.

- - - - - - - - - - - - - - - - - END OF MANUAL - - - - - - - - - - - - - - - - -