Where Automation Connects.

inRAx®

MVI46-MCM

SLC Platform

Modbus Communication Module

March 29, 2011

USER MANUAL

Your Feedback Please

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

How to Contact Us

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

MVI46-MCM User Manual

March 29, 2011

ProSoft Technology® Product Documentation

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

Important Installation Instructions

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

A WARNING - EXPLOSION HAZARD - SUBSTITUTION OF COMPONENTS MAY IMPAIR SUITABILITY FOR CLASS I, DIV. 2;
B WARNING - EXPLOSION HAZARD - WHEN IN HAZARDOUS LOCATIONS, TURN OFF POWER BEFORE REPLACING OR WIRING MODULES
C WARNING - EXPLOSION HAZARD - DO NOT DISCONNECT EQUIPMENT UNLESS POWER HAS BEEN SWITCHED OFF OR THE AREA IS KNOWN TO BE NON-HAZARDOUS.
D THIS DEVICE SHALL BE POWERED BY CLASS 2 OUTPUTS ONLY.

MVI (Multi Vendor Interface) Modules

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

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

Warnings

North America Warnings

A Warning - Explosion Hazard - Substitution of components may impair suitability for Class I, Division 2.
B Warning - Explosion Hazard - When in Hazardous Locations, turn off power before replacing or rewiring modules.
Warning - Explosion Hazard - Do not disconnect equipment unless power has been switched off or the area is known to be nonhazardous.
C Suitable for use in Class I, division 2 Groups A, B, C and D Hazardous Locations or Non-Hazardous Locations.

ATEX Warnings and Conditions of Safe Usage:

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

A Warning - Explosion Hazard - When in hazardous locations, turn off power before replacing or wiring modules.
B Warning - Explosion Hazard - Do not disconnect equipment unless power has been switched off or the area is known to be non-hazardous.
C These products are intended to be mounted in an IP54 enclosure. The devices shall provide external means to prevent the rated voltage being exceeded by transient disturbances of more than 40%. This device must be used only with ATEX certified backplanes.
D DO NOT OPEN WHEN ENERGIZED.

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

Battery Life Advisory

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

Note: The battery is not user replaceable.

Markings

Electrical Ratings

- Backplane Current Load: 800 mA @ 5 Vdc
- Operating Temperature: 0°C to 60°C (32°F to 140°F)
- Storage Temperature: -40°C to 85°C (-40°F to 185°F)
- Shock: 30g Operational; 50g non-operational; Vibration: 5 g from 10 Hz to 150 Hz
- Relative Humidity 5% to 95% (without condensation)
- All phase conductor sizes must be at least 1.3 mm(squared) and all earth ground conductors must be at least 4mm(squared).

Label Markings

Agency Approvals and Certifications

Agency	**Applicable Standards**
ANSI / ISA	ISA 12.12.01 Class I Division 2, GPs A, B, C, D
CSA/cUL	C22.2 No. 213-1987
CSA CB Certified	IEC61010
ATEX	EN60079-0 Category 3, Zone 2 EN60079-15

243333

Contents

Guide to the MVI46-MCM User Manual

Function		Section to Read	Details
Introduction (Must Do)	→	Start Here (page 11)	This section introduces the customer to the module. Included are: package contents, system requirements, hardware installation, and basic configuration.
Diagnostic and Troubleshooting	→	Diagnostics and Troubleshooting	This section describes Diagnostic and Troubleshooting procedures.
Reference Product Specifications Functional Overview	→	Reference (page 57) Product Specifications (page 57) Functional Overview (page 60)	These sections contain general references associated with this product, Specifications, and the Functional Overview.
Support, Service, and Warranty Index	→	Support, Service and Warranty (page 97) Index	This section contains Support, Service and Warranty information. Index of chapters.

1 Start Here

In This Chapter

To get the most benefit from this User Manual, you should have the following skills:

- **Rockwell Automation® RSLogix™ software:** launch the program, configure ladder logic, and transfer the ladder logic to the processor
- **Microsoft Windows:** install and launch programs, execute menu commands, navigate dialog boxes, and enter data
- **Hardware installation and wiring:** install the module, and safely connect Modbus Master/Slave and SLC devices to a power source and to the MVI46-MCM module's application port(s)

1.1 System Requirements

The MVI46-MCM module requires the following minimum hardware and software components:

- Rockwell Automation SLC 5/02 M0/M1 capable processors (or newer), with compatible power supply and one free slot in the rack, for the MVI46-MCM module. The module requires 800mA of available power.
- Rockwell Automation RSLogix 500 programming software.
- Rockwell Automation RSLinx communication software
- Pentium® II 500 MHz minimum. Pentium III 733 MHz (or better) recommended
- Supported operating systems:
  - Microsoft® Windows 98
  - Windows NT® (version 4 with SP4 or higher)
  - Windows 2000
  - Windows XP
- 32 Mbytes of RAM minimum, 64 Mbytes of RAM recommended
- 50 Mbytes of free hard disk space (or more based on application requirements)
- 16-color VGA graphics adapter, 640 x 480 minimum resolution (256 Color 800 × 600 recommended)
- CD-ROM drive
- 3.5 inch floppy disk drive
- HyperTerminal or other terminal emulator program capable of file transfers using Ymodem protocol.

1.2 Package Contents

The following components are included with your MVI46-MCM module, and are all required for installation and configuration.

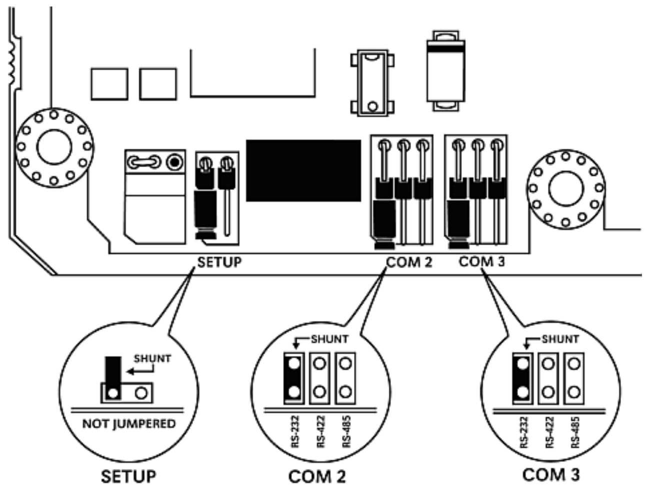
Important: Before beginning the installation, please verify that all of the following items are present.

Qty.	Part Name	Part Number	Part Description
1	MVI46-MCM Module	MVI46-MCM	Modbus Communication Module
1	Cable	Cable #15, RS232 Null Modem	For RS232 Connection to the CFG Port
3	Cable	Cable #14, RJ45 to DB9 Male Adapter cable	For DB9 Connection to Module's Port
2	Adapter	1454-9F	Two Adapters, DB9 Female to Screw Terminal. For RS422 or RS485 Connections to Port 1 and 2 of the Module
1	ProSoft Solutions CD		Contains sample programs, utilities and documentation for the MVI46-MCM module.

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

1.3 Setting Jumpers

If you use an interface other than RS-232 (default), you must change the jumper configuration to match the interface. The following illustration shows the MVI46-MCM jumper configuration:



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

1.4 Installing the Module in the Rack

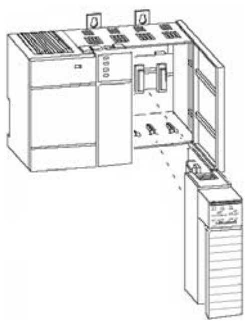
If you have not already installed and configured your SLC processor and power supply, please do so before installing the MVI46-MCM module. Refer to your Rockwell Automation product documentation for installation instructions.

Warning: You must follow all safety instructions when installing this or any other electronic devices. Failure to follow safety procedures could result in damage to hardware or data, or even serious injury or death to personnel. Refer to the documentation for each device you plan to connect to verify that suitable safety procedures are in place before installing or servicing the device.

After you have checked the placement of the jumpers, insert MVI46-MCM into the SLC™ chassis. Use the same technique recommended by Rockwell Automation to remove and install SLC™ modules.

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

1 Turn power OFF.
2 Align the module with the top and bottom guides, and slide it into the rack until the module is firmly against the backplane connector.



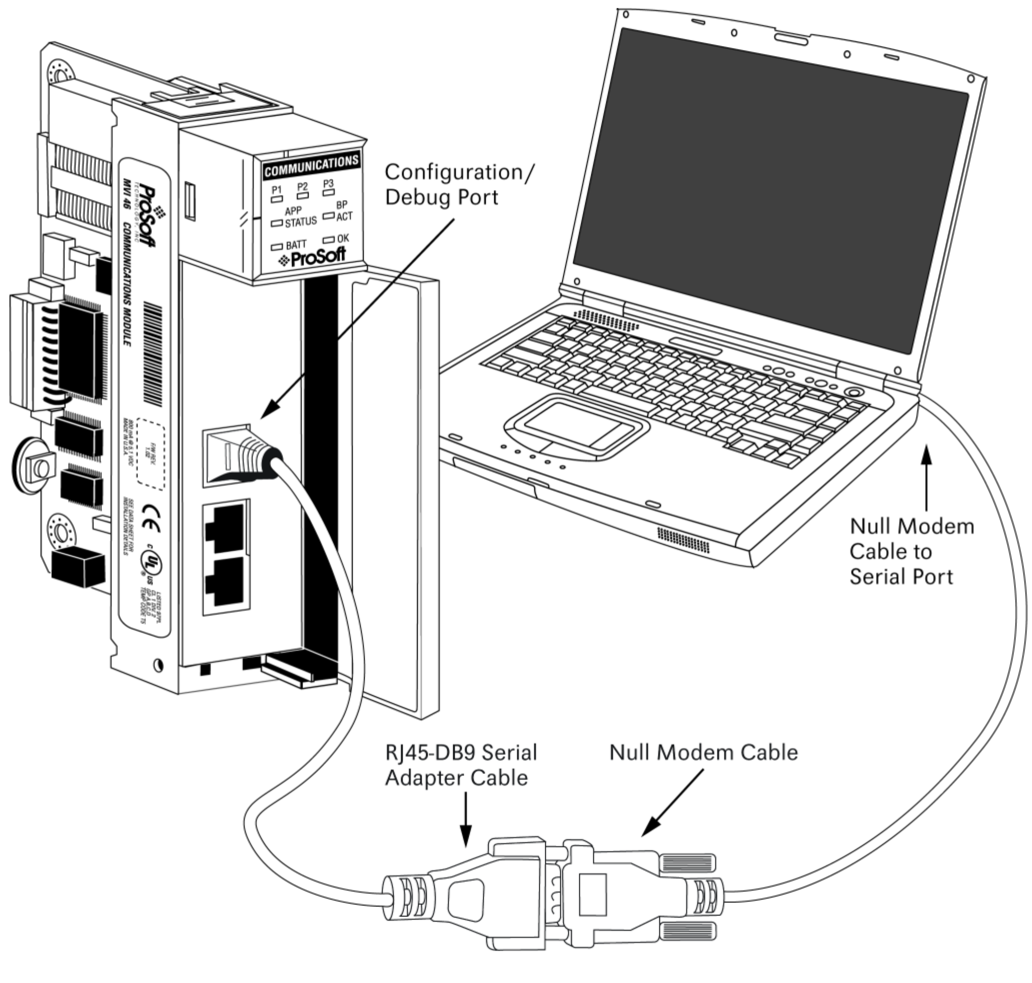
3 With a firm but steady push, snap the module into place.
4 Check that the holding clips on the top and bottom of the module are securely in the locking holes of the rack.
5 Make a note of the slot location. You will need to identify the slot in which the module is installed in order for the sample program to work correctly. Slot numbers are identified on the green circuit board (backplane) of the SLC rack.
6 Turn power ON.

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

1.5 Connecting Your PC to the Module

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

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



2 Configuring the MVI46-MCM Module

In order for the MVI46-MCM module to function, a minimum amount of configuration data must be transferred to the module. The following table describes the configuration data that the module will require, depending on the operating modes to be supported.

Module Register Address	Functional Modes Affected	Name	Description
5000 to 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 SLC processor.
5010 to 5039 and 5040 to 5069	Master and Slave	Port Configuration	These sections 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 to 6199 and 6200 to 7199	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 Installing and Configuring the Module section for a description of the configuration of the module. The MVI46-MCM module must be configured at least once when the card is first powered, and any time thereafter when the parameters must be changed.

2.1 Configuration Data

Configuration of the module is performed by filling in a user defined data table. In the example ladder logic, file N10 stores the general module configuration information. N11 stores the command list for port 1. N12 stores the command list for port 2. Each register in the files has an associated symbol and description to aid in filling in the data. Refer to MVI46-MCM Configuration Data for a list of items that must be configured for the module and their associated location in the M0 file.

Backplane Setup

File	M0 Offset	Register	Content	Description
N10:0	1	5000	Write Start Reg	Not used in this version of the software
N10:1	2	5001	Write Reg Count	Not used in this version of the software
N10:2	3	5002	Read Start Reg	Not used in this version of the software
N10:3	4	5003	Read Reg Count	Not used in this version of the software
N10:4	5	5004	Backplane Fail	Not used in this version of the software
N10:5	6	5005	Error Status Pointer	This parameter specifies the register location in the module's database where module status data is stored. If a value less than 0 is entered, the data will not be stored in the database. If the value specified is in the range of 0 to 4940, the data is placed in the user data area.
N10:6	7	5006	Spare	
N10:7	8	5007	Spare	
N10:8	9	5008	Spare	
N10:9	10	5009	Spare	

Port 1 Setup

File	M0 Offset	Register	Content	Description
N10:10	11	5010	Enable	This parameter defines if this port will be used. If the parameter is set to 0, the port is disabled. A value of 1 enables the port.
N10:11	12	5011	Type	This parameter defines if the port emulates a master or slave device. Enter 0 to emulate a master device and 1 to emulate a slave device.
N10:12	13	5012	Float Flag	This flag specifies if the floating-point data access functionality is to be used. If the float flag is set to Y, Modbus functions 3,6, and 16 will interpret floating point values for registers as specified by the two following parameters.
N10:13	14	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. For example, if a value of 7000 is entered, all requests for registers 7000 and above will be considered floating-point data.
N10:14	15	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. For example, if the Float Offset value is set to 3000 and the float start parameter is set to 7000, data requests for register 7000 will use the internal Modbus register 3000.
N10:15	16	5015	Protocol	0=RTU, 1=ASCII

File	M0 Offset	Register	Content	Description
N10:16	17	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. Exceptions: 38400 baud, enter 384. 57600 enter 576. 115000 enter 115.
N10:17	18	5017	Parity	This is the parity code to be used on the port. The coded values are as follows: 0=None 1=Odd 2=Even
N10:18	19	5018	Data Bits	This parameter sets the number of data bits for each word used by the protocol. Enter a value in the range of 5 to 8.
N10:19	20	5019	Stop Bits	This parameter sets the number of stop bits for each data value sent. Enter a value of 1 or 2.
N10:20	21	5020	RTS On Delay	This parameter sets the number of milliseconds to delay after RTS is asserted before the data will be transmitted. Enter a value in the range of 0 to 65535.
N10:21	22	5021	RTS Off Delay	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. Enter a value in the range of 0 to 65535.
N10:22	23	5022	Minimum Response Delay	This parameter sets the number of milliseconds to wait before a response message is sent out of the port. This parameter is required when interfacing to a slow responding device. Enter a value in the range of 0 to 65535.
N10:23	24	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. Normally, this parameter is required when half-duplex modems are used for communication (2-wire).
N10:24	25	5024	Slave ID	This parameter defines the virtual Modbus slave address for the internal database. Any requests received by the port with this address will be processed by the module. Verify that each device has a unique address on the network.
N10:25	26	5025	Bit Input Offset	This parameter specifies the offset address in the internal Modbus database for 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.
N10:26	27	5026	Word Input Offset	This parameter specifies the offset address in the internal Modbus database for network requests 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.

File	M0 Offset	Register	Content	Description
N10:27	28	5027	Output Offset	This parameter specifies the offset address in the internal Modbus database for 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 return the value at register 100 in the database.
N10:28	29	5028	Holding Register Offset	This parameter specifies the offset address in the internal Modbus database for network requests for Modbus 3, 6, or 16 commands. For example, if the value is set to 50, an address request of 0 will return the value at register 50 in the database.
N10:29	30	5029	Command Count	This parameter specifies the number of commands to be processed for the port. Enter a value of 0 to 100.
N10:30	31	5030	Minimum Command Delay	This parameter specifies the number of milliseconds to wait between the initial issuance of a command. This parameter can be used to delay all commands sent to slaves to avoid "flooding" commands on the network. This parameter does not affect retries of a command as they will be issued when failure is recognized. Enter a value in the range of 0 to 65535.
N10:31	32	5031	Command Error Pointer	This parameter sets the address in the internal Modbus database where the command error data will be placed. If the value is set to -1, the data will not be transferred to the database. Enter a value of 0 to 4999.
N10:32	33	5032	Response Timeout	This parameter represents the message response timeout period in 1 ms 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 on to communication network used and the expected response time of the slowest device on the network.
N10:33	34	5033	Retry Count	This parameter specifies the number of times a command will be retried if it fails. Enter a value in the range of 0 to 10.
N10:34	35	5034	Error Delay Count	This parameter specifies the number of polls to be skipped 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 the parameter. Enter a value in the range of 0 to 65535.
N10:35	36	5035	Reserved	
N10:36	37	5036	Use Guard Band Timer	Use packet gap timeout for messages (Yes or No). Use only in multi-drop applications.
N10:37	38	5037	Guard Band Timeout	A value of 0 uses the default baud rate or you can set a value in milliseconds (0 to 65535)
N10:38	39	5038	Spare	
N10:39	40	5039	Spare	

Port 2 Setup

File	M0 Offset	Register	Content	Description
N10:40	41	5040	Enable	This parameter defines if this port will be used. If the parameter is set to 0, the port is disabled. A value of 1 enables the port.
N10:41	42	5041	Type	This parameter defines if the port emulates a master or slave device. Enter 0 to emulate a master device and 1 to emulate a slave device.
N10:42	43	5042	Float Flag	This flag specifies if the floating-point data access functionality is to be used. If the float flag is set to Y, Modbus functions 3,6, and 16 will interpret floating point values for registers as specified by the two following parameters.
N10:43	44	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. For example, if a value of 7000 is entered, all requests for registers 7000 and above will be considered floating-point data.
N10:44	45	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. For example, if the Float Offset value is set to 3000 and the float start parameter is set to 7000, data requests for register 7000 will use the internal Modbus register 3000.
N10:45	46	5045	Protocol	0=RTU, 1=ASCII
N10:46	47	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. Exceptions: 38400 baud, enter 384. 57600 enter 576. 115000 enter 115.
N10:47	48	5047	Parity	This is the parity code to be used on the port. The coded values are as follows: 0=None 1=Odd 2=Even
N10:48	49	5048	Data Bits	This parameter sets the number of data bits for each word used by the protocol. Enter a value in the range of 5 to 8.
N10:49	50	5049	Stop Bits	This parameter sets the number of stop bits for each data value sent. Enter a value of 1 or 2.
N10:50	51	5050	RTS On Delay	This parameter sets the number of milliseconds to delay after RTS is asserted before the data will be transmitted. Enter a value in the range of 0 to 65535.
N10:51	52	5051	RTS Off Delay	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. Enter a value in the range of 0 to 65535.

File	M0 Offset	Register	Content	Description
N10:52	53	5052	Minimum Response Delay	This parameter sets the number of milliseconds to wait before a response message is sent out of the port. This parameter is required when interfacing to a slow responding device. Enter a value in the range of 0 to 65535.
N10:53	54	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. Normally, this parameter is required when half-duplex modems are used for communication (2-wire).
N10:54	55	5054	Slave ID	This parameter defines the virtual Modbus slave address for the internal database. Any requests received by the port with this address will be processed by the module. Verify that each device has a unique address on the network.
N10:55	56	5055	Bit Input Offset	This parameter specifies the offset address in the internal Modbus database for 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.
N10:56	57	5056	Word Input Offset	This parameter specifies the offset address in the internal Modbus database for network requests 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.
N10:57	58	5057	Output Offset	This parameter specifies the offset address in the internal Modbus database for 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 return the value at register 100 in the database.
N10:58	59	5058	Holding Register Offset	This parameter specifies the offset address in the internal Modbus database for network requests for Modbus 3, 6, or 16 commands. For example, if the value is set to 50, an address request of 0 will return the value at register 50 in the database.
N10:59	60	5059	Command Count	This parameter specifies the number of commands to be processed for the port. Enter a value of 0 to 100.
N10:60	61	5060	Minimum Command Delay	This parameter specifies the number of milliseconds to wait between the initial issuance of a command. This parameter can be used to delay all commands sent to slaves to avoid "flooding" commands on the network. This parameter does not affect retries of a command as they will be issued when failure is recognized. Enter a value in the range of 0 to 65535.
N10:61	62	5061	Command Error Pointer	This parameter sets the address in the internal Modbus database where the command error data will be placed. If the value is set to -1, the data will not be transferred to the database. Enter a value of 0 to 4999.

File	M0 Offset	Register	Content	Description
N10:62	63	5062	Response Timeout	This parameter represents the message response timeout period in 1 ms 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 on to communication network used and the expected response time of the slowest device on the network.
N10:63	64	5063	Retry Count	This parameter specifies the number of times a command will be retried if it fails. Enter a value in the range of 0 to 10.
N10:64	65	5064	Error Delay Count	This parameter specifies the number of polls to be skipped 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 the parameter. Enter a value in the range of 0 to 65535.
N10:65	66	5065	Reserved	
N10:66	67	5066	Use Guard Band Timer	Use packet gap timeout for messages (Yes or No). Use only in multi-drop applications.
N10:67	68	5067	Guard Band Timeout	A value of 0 uses the default baud rate or you can set a timeout value in milliseconds (0 to 65535).
N10:68	69	5068	Spare	
N10:69	70	5069	Spare	

Guard Band parameters

With multiple nodes in the network (multidrop), the module must be able to define the time period used to properly recognize the received data sequence as a valid Modbus message. This time period is defined through the guardband timer parameter. Failure to set the Guard Band functionality may cause successive communication errors in the Modbus network.

This functionality is only used for Modbus RTU mode, when the module is configured as a slave device in a multidrop network (RS-422 or RS-485 wiring). In RTU mode, when the module receives sequence of data from another node, it will recognize the end of the message when it detects a 3.5 character gap. The next byte received after a 3.5 character gap will be recognized as the start of a new Modbus message.

The 3.5 character gap depends on the port communication settings, especially the baud rate. The higher the baud rate, the lower the corresponding time period equivalent to the 3.5 character delay.

To enable the Guard Band Timeout parameter:

1 First of all, enable the functionality through the Use Guard Band Timer parameter as follows:

```
Use Guard Band Timer = Y
```

2 Then, set the Guard Band Timeout as follows:

```
Guard Band Timeout = 0
```

This configuration will set the module to use a pre-defined time interval for Modbus message recognition calculated for each baud rate. This should provide optimal performance for most applications.

The following guard band timer values will be automatically used by the module with the default parameter value (Guard Band Timeout = 0) is selected:

Baud	**Guard Band Timer (ms)**
110	350
150	256
300	128
600	64
1200	32
2400	16
4800	8
9600	4
19200	2
28800	2
38400	2
57600	1
115200	1

If you still observe a considerable number of communication errors, try to increase the Guard Band Timeout value (in milliseconds) until the network performance is improved.

Port 1 Commands

File	M0 Offset	Register	Content	Description
N11:0 to N11:09	71 to 80	5200 to 5209	Command #1	This set of registers contains the parameters for the first command in the master command list. Refer to the data object section of the documentation.
N11:10 to N11:19	81 to 90	5210 to 5219	Command #2	Command #2 dataset
	1061 to 1070	6190 to 6199	Command #100	Command #100 dataset

Port 2 Commands

File	M0 Offset	Register	Content	Description
N12:0 to N12:09	1071 to 1080	6200 to 6209	Command #1	This set of registers contains the parameters for the first command in the master command list. Refer to the data object section of the documentation.
N12:10 to N12:19	1081 to 1090	6210 to 6219	Command #2	Command #2 dataset
	2061 to 2070	7190 to 7199	Command #100	Command #100 dataset

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 Command Control (page 69, page 25, page 64) for more information.
7801 to 7999	Command Data	Reserved for future use.

3 Ladder Logic

In This Chapter

Ladder logic is required for application of the MVI46-MCM module. Tasks that must be handled by the ladder logic are module data transfer, special block handling, and status data receipt. Additionally, a power-up handler may be needed to handle the initialization of the module's data and to clear any processor fault conditions.

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

3.1 Module Data

All data related to the MVI46-MCM module is stored in user defined data areas. The user is responsible for setting up the data areas to match the specific application for which the module is used. Each data area is discussed in the following topics.

3.1.1 Backplane Parameters

In this revision of the module, all data to be transferred between the SLC processor and the module is held in the module's M1 file. This simplifies the ladder logic. In order to read data in the module, copy the specific data area in the M1 file into a user defined file. Repeat this operation for each data area. Remember, the maximum data area that can be copied with the COP instruction is 128 words. In order to write data to the module, copy the data in the user defined file to the specific data area in the M1 file. The read and write data operations should be limited to the M1 words 0 to 4999. The modules status data can be read from the M1 file starting at register 7200.

Only one parameter is used in this section of the configuration.

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

3.1.2 Port Parameters

These parameters define the operation of each of the Modbus ports on the module. Refer to MVI46-MCM Configuration Data Definition for a definition of each parameter.

3.1.3 Master Commands

These records define the commands in the master command list. Each parameter is described in the following table.

Parameter	Description
Enable	This parameter defines if the command is executed or disregarded. The following values are valid: 0=Disables the command and it will not execute. 1=The command will be considered for execution each scan of the command list and will be controlled by the PollInt parameter. And 2=The command will only execute if the data associated with the command has changed since the command was last issued. This option is only available for write commands.
IntAddress	This parameter specifies the starting internal register address to be associated with the command. Valid entry for this parameter is 0 to 4999 for Register Address (0 to 65535 for Bit Address).
PollInt	This parameter defines the minimum number of seconds to wait between the execution of continuous commands (Enable=1). This poll interval command can be used to lighten the communications load on a busy network. Valid entry for this parameter is 0 to 65535.
Count	This parameter defines the number of registers to be considered by the command. Valid entry for this parameter depends on the Modbus specification for the command.
Swap	This parameter specifies if the data used in the command must be altered when a Modbus function code 3 reads data from a node on the network. Values that can be assigned are as follows: 0=no swapping of data, 1=swap word values, 2=swap word and byte values and 3=swap byte values. This option is used when interfacing the module with ASCII and floating-point data on other devices.
Node	This parameter assigns the Modbus slave node address for the module to reach with the command on the Modbus network. This parameter can be assigned values from 0 to 255. Most Modbus networks limit the upper value to 247.
Func	This parameter specifies the Modbus function to be performed by the command. Valid entries are 1, 2, 3, 4, 5, 6, 15 and 16.
DevAddress	This parameter defines the starting address in the device being considered by the command. Values entered in this field are dependent on the node's database definition. Refer to the specific manufacture's database definition for the device to determine the location of the data to be interfaced.

3.1.4 Status Data

This data area views the status of the module. Use this data to monitor the state of the module at a "real-time rate". Refer to the MVI46-MCM Status Data Definition (page 82) for a complete listing of the data stored in this object. This data can be read from the module's M1 file starting at register 7200. User data file N30 is defined in the example ladder logic for this purpose.

3.1.5 User Data

All user data is stored in the module’s M1 file in registers 0 to 4999. This 5000-word area is directly accessible from the ladder logic. The COP instruction should be used to move blocks of data between the user data files and the module’s M1 file. This limits the number of accesses to the M1 data area and provides faster system response. In the example ladder logic, N31 holds data read from the module, and N32 stores data to write to the module.

3.1.6 Slave Polling Control and Status

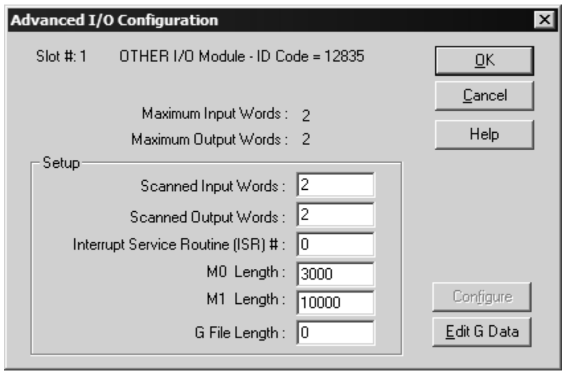
Two data areas can be allocated in the SLC to hold the polling status of each slave on the master ports. This status data can be used to determine which slaves are currently active on the port, are in communications error, or have their polling suspended and disabled. If the configuration supplies an address where this data resides, copy the data from the M1 file to the reserved files in the SLC. Using special blocks, the processor can enable or disable the polling of selected slaves.

3.2 Adding the Module to an Existing Project

1 Copy the Ladder Logic and data files from the sample program and paste them into your existing program.

Important: Take care not to overwrite existing data files in your application with data files in the sample application. Rename either the source or the destination data files, and then search and replace references in the ladder for instances of any renamed files.

2 Save and Download the new application to the controller and place the processor in run mode.



Slot #: 1 OTHER I/O Module - ID Code = 12835

Maximum Input Words : 2
Maximum Output Words : 2

Scanned Input Words : 2
Scanned Output Words : 2
Interrupt Service Routine (ISR) # : 0
M0 Length : 3000
M1 Length : 10000
G File Length : 0

Field	Value
Scanned Input Words	2
Scanned Output Words	2
Interrupt Service Routine (ISR)#	0
M0 Length	3000
M1 Length	10000
G File Length	0

3 Click OK to save your configuration.
4 Copy the Ladder Logic and data files from the sample program and paste them into your existing program.

Important: Take care not to overwrite existing data files in your application with data files in the sample application. Rename either the source or the destination data files, and then search and replace references in the ladder for instances of any renamed files.

5 Save and Download the new application to the controller and place the processor in run mode.

4 Diagnostics and Troubleshooting

In This Chapter

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

- LED status indicators on the front of the module provide general information on the module's status.
- Status Data contained in the module can be viewed through the Configuration/Debug port, using the diagnostic capabilities of *Microsoft Hyperterminal*.

Status data values can be transferred from the module to processor memory and can be monitored there manually or by customer-created logic.

4.1 LED Status Indicators

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

LED	Color	Status	Indication
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 its Modbus 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 its Modbus Port 2.
		Off	No data is being transferred on the port.
APP	Amber	On	The MVI46-MCM is working normally.
		Off	The MVI46-MCM module program has recognized a communication error on one of its Modbus ports.
BP ACT	Amber	On	The LED is on when the module is performing a write operation on the backplane.
		Off	The LED is off when the module is performing a read operation on the backplane. Under normal operation, the LED should blink rapidly on and off.
OK	Red/ Green	Off	The card is not receiving any power and is not securely plugged into the rack.
		Green	The module is operating normally.
		Red	The program has detected an error or is being configured. If the LED remains red for over 10 seconds, the program has probably halted. Remove the card from the rack and re-insert the card to restart the module's program.
BAT	Red	Off	The battery voltage is OK and functioning.
		On	The battery voltage is low or battery is not present. Allow battery to charge by keeping module plugged into rack for 24 hours. If BAT LED still does not go off, contact ProSoft Technology, as this is not a user serviceable item.

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

M0 Offset	Description	Length
0	9000	1
1	Spare	1
2	Port 1 Configuration Errors	1
3	Port 2 Configuration Errors	1

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	Protocol parameter is not valid.	0x0002
2	Termination type parameter is not valid.	0x0004
3	Baud rate parameter is not valid.	0x0008
4	Parity parameter is not valid.	0x0010
5	Data bits parameter is not valid.	0x0020
6	Stop bits parameter is not valid.	0x0040
7	Command count parameter is not valid.	0x0080
8	Retry count parameter is not valid.	0x0100
9	Spare	0x0200
10	Spare	0x0400
11	Spare	0x0800
12	Spare	0x1000
13	Spare	0x2000
14	Spare	0x4000
15	Spare	0x8000

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

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

4.1.1 Clearing a Fault Condition

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

To clear the condition, follow these steps:

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

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

4.1.2 Troubleshooting

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

Processor Errors

Problem description	**Steps to take**
Processor fault	Verify that the module is plugged into the slot that has been configured for the module in the I/O Configuration of RSLogix. Verify that the slot location in the rack has been configured correctly in the ladder logic.
Processor I/O LED flashes	This indicates a problem with backplane communications. A problem could exist between the processor and any installed I/O module, not just the MVI46-MCM. Verify that all modules in the rack are correctly configured in the ladder logic.

Module Errors

Problem description	**Steps to take**
BP ACT LED (not present on MVI56E modules) remains OFF or blinks slowly MVI56E modules with scrolling LED display: *<Backplane Status>* condition reads ERR	This indicates that backplane transfer operations are failing. Connect to the module's Configuration/Debug port to check this. To establish backplane communications, verify the following items: ▪ The processor is in RUN or REM RUN mode. ▪ The backplane driver is loaded in the module. ▪ The module is configured for read and write data block transfer. ▪ The ladder logic handles all read and write block situations. ▪ The module is properly configured in the processor I/O configuration and ladder logic.
OK LED remains RED	The program has halted or a critical error has occurred. Connect to the Configuration/Debug port to see if the module is running. If the program has halted, turn off power to the rack, remove the card from the rack and re-insert it, and then restore power to the rack.

4.2 The Configuration/Debug Menu

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

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

4.2.1 Using the Configuration/Debug Port

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

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

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

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

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

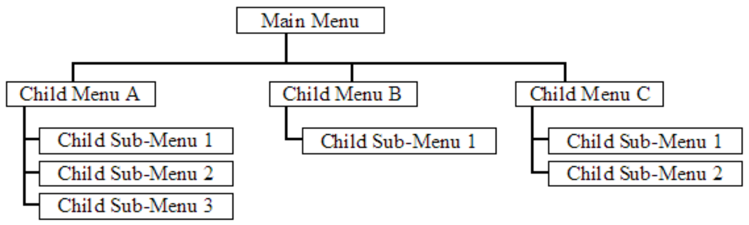
1 Verify that the null modem cable is connected properly between your computer's serial port and the module. A regular serial cable will not work.
2 Verify that RSLinx is not controlling the COM port. Refer to Disabling the RSLinx Driver for the Com Port on the PC.
3 Verify that your communication software is using the correct settings for baud rate, parity and handshaking.
4 On computers with more than one serial port, verify that your communication program is connected to the same port that is connected to the module.

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

Navigation

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

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



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

Keystrokes

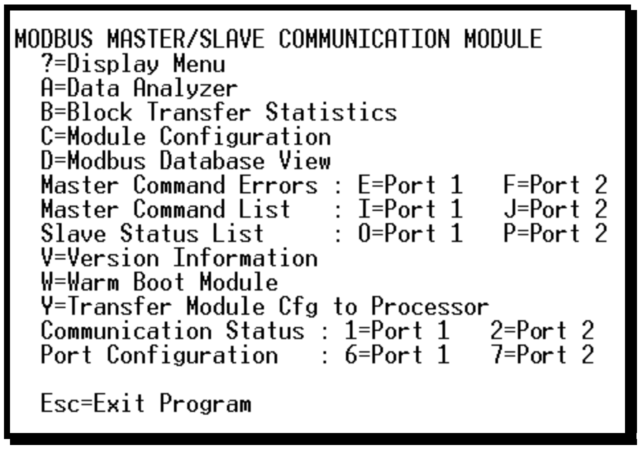
The keyboard commands on these menus are usually not case sensitive. You can enter most commands in lowercase or uppercase letters.

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

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

4.2.2 Main Menu

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



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

  Esc=Exit Program
```

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

Opening the Data Analyzer Menu

Press **[A]** to open the Data Analyzer Menu. Use this command to view all bytes of data transferred on each port. Both the transmitted and received data bytes are displayed. Refer to Data Analyzer (page 43) for more information about this menu.

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

Viewing Block Transfer Statistics

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

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

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

Viewing Module Configuration

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

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

Opening the Database View Menu

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

Use this menu command to view the current contents of the module's database. For more information about this submenu, see Database View Menu (page 48).

Opening the Master Command Error List (Ports 1 and 2)

Press **[E]** (port 1) or **[F]** (port 2) to view the Command Error List menu. Use this command to master command list error/status data.

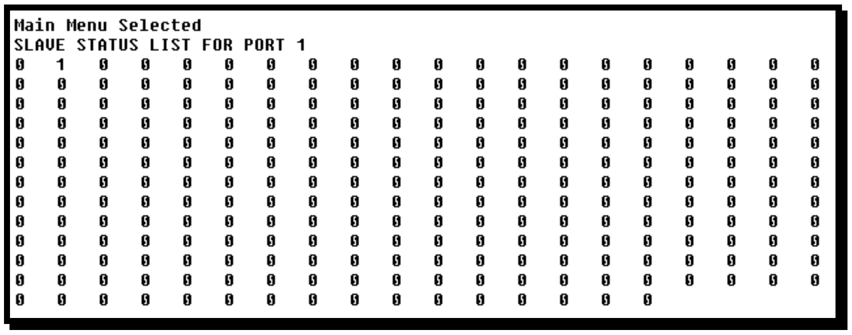
Opening the Command List Menu (Ports 1 and 2)

Press **[I]** (port 1) or **[J]** (port 2) to open the Master Command List menu. Use this command to view master command list data.

Viewing the Slave Status List (Port 1 and 2)

Press **[O]** (port 1) or **[P]** (port 2) to view the 256 slave status values associated with the ports.

- 0 = slave is not used
- 1 = slave being actively polled
- 2 = slave suspended
- 3 = slave disabled.



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

Viewing Version Information

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

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

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

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

Warm Booting the Module

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

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

Transferring Module Configuration to the Processor

Press **[Y]** to transfer the module's configuration data to the processor. Ladder logic is required in the processor to receive and implement the updated configuration. You will be prompted to confirm the transfer.

If the operation is not successful, an error code will be returned.

Code	Description
0	Transfer successful
-1	Error transferring module configuration data (block -9000)
-2	Error transferring device definition data (blocks -9100 to -9103)
-3	Error transferring master command list data (blocks -6000 to -6007)

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

Viewing Port Communication Status

Press **[1]** or **[2]** from the Main Menu to view the port communication status for Ports 1 and 2.

Use this command to view communication status and statistics for the selected port. This information can be informative when troubleshooting communication problems.

Viewing Port Configuration

Press **[6]** or **[7]** from the Main Menu to view configuration information for ports 1 and 2.

Use this command to display detailed configuration information for the selected port.

Exiting the Program

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

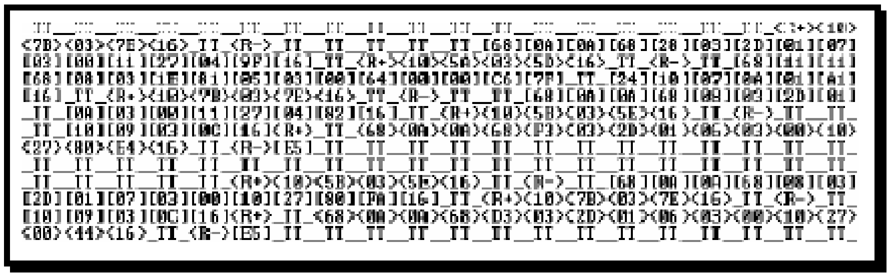
4.2.3 Data Analyzer

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

Note: The Port selection commands on the Data Analyzer menu differs very slightly in different modules, but the functionality is basically the same. Use the illustration above as a general guide only. Refer to the actual data analyzer menu on your module for the specific port commands to use.
Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please press [S] to stop the data analyzer, and then press [M] to return to the main menu. This action will allow the module to resume its normal high speed operating mode.

Analyzing Data for the first application port

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



```
_TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT_<R+><10>
<7B><03><7E><16>_TT_<R->_TT__TT__TT__TT__TT_[68][0A][0A][68][28][03][2D][01][07]
[03][00][11][27][04][9F][16]_TT_<R+><10><5A><03><5D><16>_TT_<R->_TT_[68][11][11]
[68][08][03][1E][81][05][03][00][64][00][00][C6][7F]_TT_[24][10][07][0A][01][A1]
[16]_TT_<R+><10><7B><03><7E><16>_TT_<R->_TT__TT_[68][0A][0A][68][08][03][2D][01]
_TT_[0A][03][00][11][27][04][82][16]_TT_<R+><10><5B><03><5E><16>_TT_<R->_TT__TT_
_TT_[10][09][03][0C][16]<R+>_TT_<68><0A><0A><68><F3><03><2D><01><06><03><00><10>
<27><00><E4><16>_TT_<R->[E5]_TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT_
_TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT_
_TT__TT__TT__TT__TT_<R+><10><5B><03><5E><16>_TT_<R->_TT_[68][0A][0A][68][08][03]
[2D][01][07][03][00][10][27][80][FA][16]_TT_<R+><10><7B><03><7E><16>_TT_<R->_TT_
[10][09][03][0C][16]<R+>_TT_<68><0A><0A><68><D3><03><2D><01><06><03><00><10><27>
<00><44><16>_TT_<R->[E5]_TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT_
```

Analyzing Data for the second application port

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

Displaying Timing Marks in the Data Analyzer

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

Key	**Interval**
[5]	1 milliseconds ticks
[6]	5 milliseconds ticks
[7]	10 milliseconds ticks
[8]	50 milliseconds ticks
[9]	100 milliseconds ticks
[0]	Turn off timing marks

Removing Timing Marks in the Data Analyzer

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

Viewing Data in Hexadecimal Format

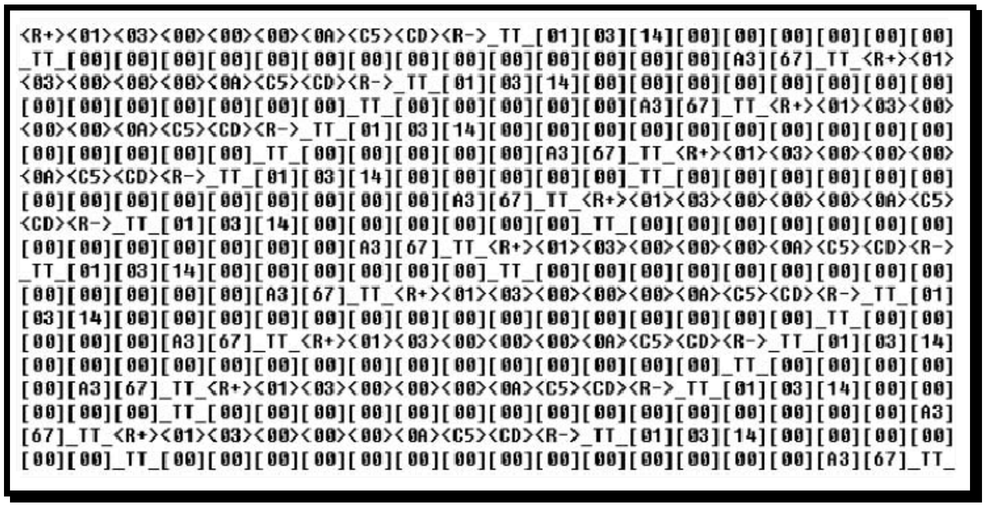
Press **[H]** from the *Database View* menu to display the data on the current page in hexadecimal format.

Viewing Data in ASCII (Text) Format

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

Starting the Data Analyzer

Press **[B]** to start the data analyzer. After the key is pressed, all data transmitted and received on the currently selected port will be displayed. The following illustration shows an example.



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

The Data Analyzer displays the following special characters:

Character	Definition
[ ]	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.

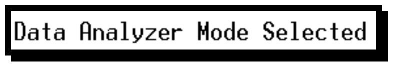
Stopping the Data Analyzer

Press **[S]** to stop the data analyzer. Use this option to freeze the display so the data can be analyzed. To restart the analyzer, press **[B].**

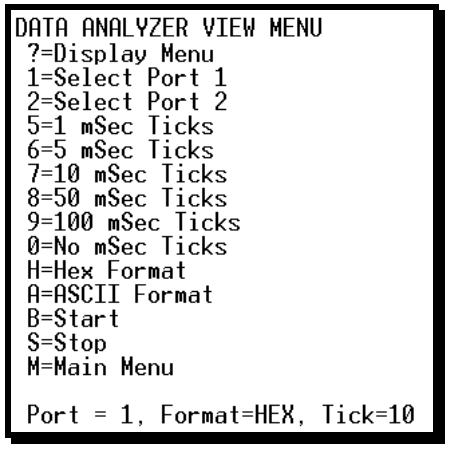
Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please press [S] to stop the data analyzer, and then press [M] to return to the main menu. This action will allow the module to resume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

Data Analyzer Tips

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


```
Data Analyzer Mode Selected
```

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

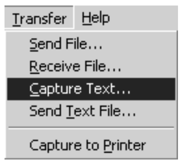

```
DATA ANALYZER VIEW MENU
 ?=Display Menu
 1=Select Port 1
 2=Select Port 2
 5=1 mSec Ticks
 6=5 mSec Ticks
 7=10 mSec Ticks
 8=50 mSec Ticks
 9=100 mSec Ticks
 0=No mSec Ticks
 H=Hex Format
 A=ASCII Format
 B=Start
 S=Stop
 M=Main Menu

 Port = 1, Format=HEX, Tick=10
```

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

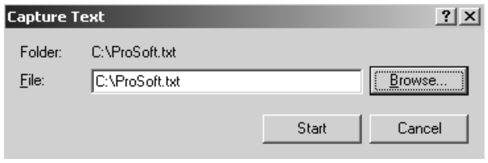
For most applications, HEX is the best format to view the data, and this does include ASCII based messages (because some characters will not display on HyperTerminal and by capturing the data in HEX, we can figure out what the corresponding ASCII characters are supposed to be).

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

After you have selected the Port, Format, and Tick, we are now ready to start a capture of this data. The easiest way to do so is to go up to the top of you HyperTerminal window, and do a **TRANSFER / CAPTURE TEXT** as shown below:



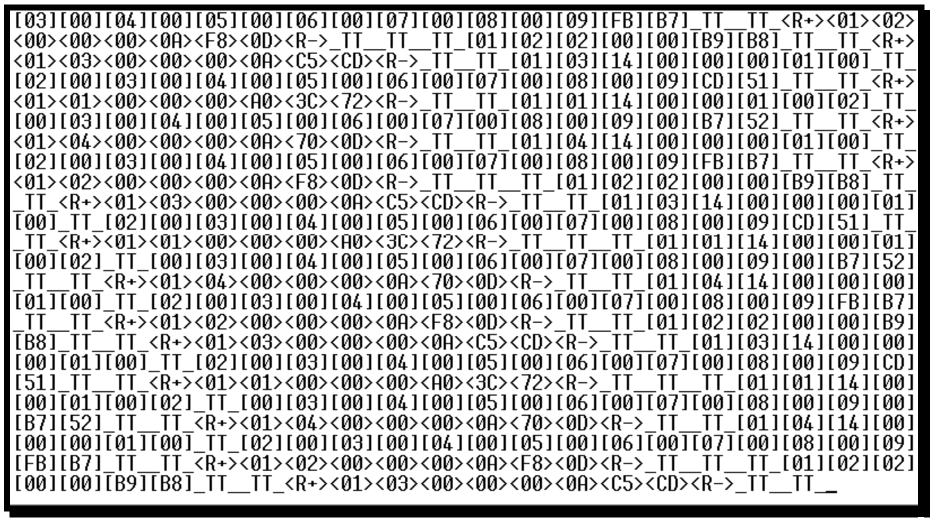
After selecting the above option, the following window will appear:



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

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

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



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

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

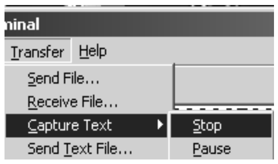
All characters shown in <> brackets are characters being sent out by the module.

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

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

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

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



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

Returning to the Main Menu

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

4.2.4 Database View Menu

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

```
DB Menu Selected

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
```

Viewing Register Pages

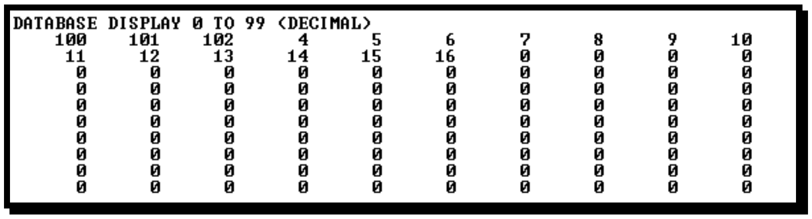
To view sets of register pages, use the keys described below:

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

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

Displaying the Current Page of Registers Again

Press **[S]** from the *Database View* menu to show the current page of registers again.



```
DATABASE DISPLAY 0 TO 99 (DECIMAL)
     100     101     102       4       5       6       7       8       9      10
      11      12      13      14      15      16       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
```

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

Moving Back Through 5 Pages of Registers

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

Moving Forward (Skipping) Through 5 Pages of Registers

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

Viewing the Previous Page of Registers

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

Viewing the Next Page of Registers

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

Viewing Data in Decimal Format

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

Viewing Data in Hexadecimal Format

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

Viewing Data in Floating-Point Format

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

Viewing Data in ASCII (Text) Format

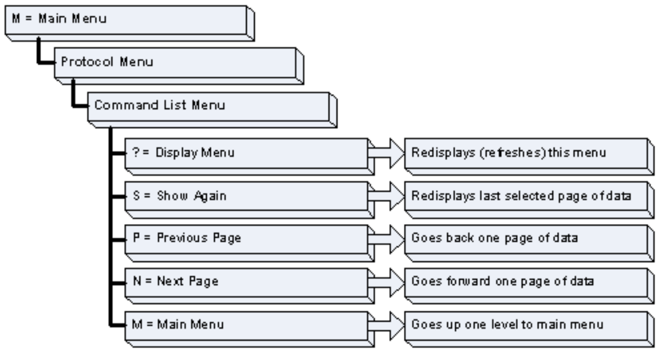
Press **[A]** from the *Database View* menu to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Returning to the Main Menu

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

4.2.5 Master Command Error List Menu

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


M = Main Menu
Protocol Menu
Command List Menu
? = Display Menu
Redisplays (refreshes) this menu
S = Show Again
Redisplays last selected page of data
P = Previous Page
Goes back one page of data
N = Next Page
Goes forward one page of data
M = Main Menu
Goes up one level to main menu

Redisplaying the Current Page

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

Moving Back Through 5 Pages of Commands

Press **[-]** to display data for last 5 page commands.

Viewing the Previous Page of Commands

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

Moving Forward (Skipping) Through 5 Pages of Commands

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

Viewing the Next Page of Commands

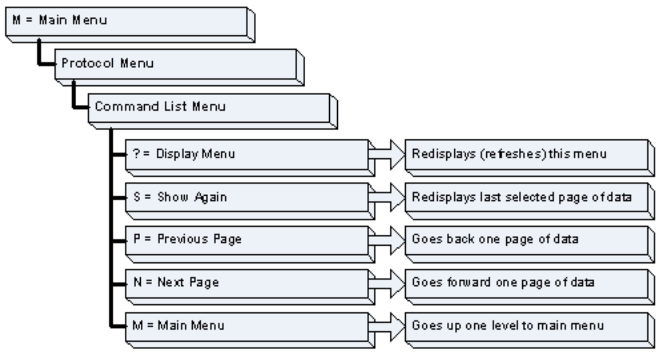
Press **[N]** to display the next page of commands.

Returning to the Main Menu

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

4.2.6 Master Command List Menu

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


M = Main Menu
Protocol Menu
Command List Menu
? = Display Menu
Redisplays (refreshes) this menu
S = Show Again
Redisplays last selected page of data
P = Previous Page
Goes back one page of data
N = Next Page
Goes forward one page of data
M = Main Menu
Goes up one level to main menu

Redisplaying the Current Page

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

Viewing the Previous 50 Commands

Press **[-]** to view the previous 50 commands.

Viewing the Previous Page of Commands

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

Viewing the Next 50 Commands

Press **[+]** to view the next 50 commands from the master command list.

Viewing the Next Page of Commands

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

Returning to the Main Menu

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

4.3 Reading Status Data from the Module

The MVI46-MCM module returns a 33-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 7200 to 7232.

4.3.1 MVI46-MCM Status Data Definition

This section contains a description of the members present in theStatus Table area of module memory

Offset	Content	Description
7200	Program Scan Count	This value is incremented each time a complete program cycle occurs in the module.
7201 to 7202	Product Code	These two registers contain the product code of "MCM".
7203 to 7204	Product Version	These two registers contain the product version for the current running software.
7205 to 7206	Operating System	These two registers contain the month and year values for the program operating system.
7207 to 7208	Run Number	These two registers contain the run number value for the currently running software.
7209	Port 1 Command List Requests	This field contains the number of requests made from this port to Slave devices on the network.
7210	Port 1 Command List Response	This field contains the number of Slave response messages received on the port.
7211	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.
7212	Port 1 Requests	This field contains the total number of messages sent from the port.
7213	Port 1 Responses	This field contains the total number of messages received on the port.
7214	Port 1 Errors Sent	This field contains the total number of message errors sent from the port.
7215	Port 1 Errors Received	This field contains the total number of message errors received on the port.
7216	Port 2 Command List Requests	This field contains the number of requests made from this port to Slave devices on the network.
7217	Port 2 Command List Response	This field contains the number of Slave response messages received on the port.
7218	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.
7219	Port 2 Requests	This field contains the total number of messages sent out the port.
7220	Port 2 Responses	This field contains the total number of messages received on the port.
7221	Port 2 Errors Sent	This field contains the total number of message errors sent out the port.

Offset	Content	Description
7222	Port 2 Errors Received	This field contains the total number of message errors received on the port.
7223	Read Block Count	This field contains the total number of read blocks transferred from the module to the processor.
7224	Write Block Count	This field contains the total number of write blocks transferred from the module to the processor.
7225	Parse Block Count	(Not Used)
7226	Command Event Block Count	This field contains the total number of command event blocks received from the processor.
7227	Command Block Count	This field contains the total number of command blocks received from the processor.
7228	Error Block Count	This field contains the total number of block errors recognized by the module.
7229	Port 1 Current Error	For a Master Port, this field contains the command index number of the most recently executed command that failed. To find what kind of error occurred, see the Command Error List entry for this command index number.
7230	Port 1 Last Error	For a Master Port, this field contains the command index number of the previous most recently executed command that failed. To find what kind of error occurred, see the Command Error List entry for this command index number.
7231	Port 2 Current Error	For a Master Port, this field contains the command index number of the most recently executed command that failed. To find what kind of error occurred, see the Command Error List entry for this command index number.
7232	Port 2 Last Error	For a Master Port, this field contains the command index number of the previous most recently executed command that failed. To find what kind of error occurred, see the Command Error List entry for this command index number.

4.3.2 Command Error Codes

The MVI46-MCM module will return an individual error code for every configured command. The location of these error codes are determined by the valued entered for the parameter, **COMMAND ERROR POINTER.** This parameter determines where in the module's 5000-register database the error codes for each command will be placed. The amount of error codes returned into the database is determined by the **COMMAND COUNT** parameter; therefore if the maximum number of commands have been selected (100), then 100 registers will be placed into the module memory.

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

The "Standard Modbus Protocol Errors" are error codes returned by the device itself. This means that the Slave device understood the command, but replied with an Exception Response, which indicates that the command could not be executed. These responses typically do not indicate a problem with port settings or wiring.

The most common values are Error Code 2 and Error Code 3.

Error Code 2 means that the module is trying to read an address in the device that the Slave does not recognize as a valid address. This is typically caused by the Slave device skipping some registers. If you have a Slave device that has address 40001 to 40005, and 40007 to 40010, you cannot issue a read command for addresses 40001 to 40010 (function code 3, DevAddress 0, Count 10) because address 40006 is not a valid address for this Slave.

Instead, try reading just one register, and see if the error code goes away. You can also try adjusting your DevAddress -1, as some devices have a 1 offset.

An Error Code of 3 is common on Modbus Write Commands (Function Codes 5,6,15, or 16). Typically, this is because you are trying to write to a parameter that is configured as read only in the Slave device, or the range of the data you are writing does not match the valid range for that device.

Refer to the documentation for your Slave device, or contact ProSoft Technical Support for more help with these types of error codes.

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

"Module Communication Errors" are generated by the MVI46-MCM module, and indicate communication errors with the Slave device.

Error Code -11 indicates that the module is transmitting a message on the communications wire. However, it is not receiving a response from the addressed Slave. This error is typically caused by one or more of the following conditions.

- Parameter mismatch, for example the module is set for 9600 baud, Slave is set for 19,200, parity is set to none, Slave is expecting even, and so on.
- Wiring problem, for example the port jumper on the module is set incorrectly, or + and - lines on RS485 are switched)
- The Slave device is not set to the correct address, for example the Master is sending a command to Slave 1 and the Slave device is configured as device 10.

With a -11 error code, check all of the above parameters, wiring, and settings on the Slave device. Also make sure that you cycle power to the module.

Error codes of 253 to 255 typically indicate noise on RS485 lines. Make sure that you are using the proper RS485 cable, with termination resistors installed properly on the line. If termination resistors are installed, try removing them as they are usually only required on cable lengths of more than 1000 feet.

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

The above error codes indicate that the module has detected an error when parsing the command.

For all commands that have not been configured (all parameters set to a value of 0) you will receive an error code of -44. To remove this error code, you can change your **COMMAND COUNT** parameter to the number of commands that are actually configured and cycle power to the module to transfer the new values.

5 Reference

In This Chapter

5.1 Product Specifications

The MVI46 Modbus Master/Slave Communication Module allows Rockwell Automation® SLC® processors to interface easily with other Modbus protocol compatible devices.

The module acts as an input/output module between the Modbus network and the SLC backplane. Compatible devices include not only Modicon® PLCs (almost all support the Modbus protocol) but also a wide range of process and control devices from a variety of manufacturers. Many SCADA packages also support the Modbus protocol.

5.1.1 General Specifications

- Single Slot - 1746 backplane compatible (Local or extended I/O rack only. Remote rack not supported)
- The module is recognized as an Input/Output module and has access to processor memory for data transfer between processor and module using M0/M1 files
- Ladder Logic is used for data transfer between module and processor
- Configuration data obtained through user-defined ladder. Sample ladder file included

5.1.2 Hardware Specifications

Specification	Description
Backplane Current Load	800 ma @ 5V (from backplane)
Operating Temperature	0 to 60°C (32 to 140°F)
Storage Temperature	-40 to 85°C (-40 to 185°F)
Shock	30g operational, 50g non-operational
Relative Humidity	5% to 95% (non-condensing)
Vibration	5 g from 10150 Hz
Processor	Compatible with Rockwell Automation SLC 5/02 M0/M1 capable processors or newer
LED indicators	Module status, Backplane transfer status, Application status, Serial activity and error LED status
Debug/Configuration port (CFG)	
CFG Port (CFG)	RJ45 (DB-9M with supplied cable) RS-232 only
Configuration Connector	RJ45 RS-232 Connector (RJ45 to DB-9 cable shipped with unit)
Application Ports	
Application Serial port (PRT1, PRT2) (Serial Modules)	Two RJ45 RS-232/422/485 Application ports

5.1.3 General Specifications - Modbus Master/Slave

Communication parameters	Baud Rate: 110 to 115K baud Stop Bits: 1 or 2 Data Size: 7 or 8 bits Parity: None, Even, Odd RTS Timing delays: 0 to 65535 milliseconds	
Modbus Modes	RTU mode (binary) with CRC-16 ASCII mode with LRC error checking	
Floating Point Data	Floating point data movement supported, including configurable support for Enron, Daniel®, and other implementations	
Modbus Function Codes Supported	1: Read Coil Status 2: Read Input Status 3: Read Holding Registers 4: Read Input Registers 5: Force (Write) Single Coil 6: Preset (Write) Single Holding Register 8: Diagnostics (Slave Only, Responds to Subfunction 00)	15: Force(Write) Multiple Coils 16: Preset (Write) Multiple Holding Registers 17: Report Slave ID (Slave Only) 22: Mask Write Holding Register (Slave Only) 23: Read/Write Holding Registers (Slave Only)

5.1.4 Functional Specifications

Modbus Master

A port configured as a virtual Modbus Master actively issues Modbus commands to other nodes on the Modbus network, supporting up to 100 commands on each port. The Master ports have an optimized polling characteristic that polls slaves with communication problems less frequently.

Command List	Up to 100 command per Master port, each fully configurable for function, slave address, register to/from addressing and word/bit count.
Polling of command list	Configurable polling of command list, including continuous and on change of data, and dynamically user or automatic enabled.
Status Data	Error codes available on an individual command basis. In addition, a slave status list is maintained per active Modbus Master port.

Modbus Slave

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 SLC processor.

Node address	1 to 247 (software selectable)
Status Data	Error codes, counters and port status available per configured slave port

5.2 Functional Overview

5.2.1 About the MODBUS Protocol

MODBUS is a widely-used protocol originally developed by Modicon in 1978. Since that time, the protocol has been adopted as a standard throughout the automation industry.

The original MODBUS specification uses a serial connection to communicate commands and data between Master and Slave devices on a network. Later enhancements to the protocol allow communication over other types of networks.

MODBUS is a Master/Slave protocol. The Master establishes a connection to the remote Slave. When the connection is established, the Master sends the MODBUS commands to the Slave. The MVI46-MCM module can work as a Master and as a Slave.

The MVI46-MCM module also works as an input/output module between itself and the Rockwell Automation backplane and processor. The module uses an internal database to pass data and commands between the processor and Master and Slave devices on MODBUS networks.

5.2.2 General Concepts

The following topics describe several concepts that are important for understanding the operation of the MVI46-MCM module.

1 On power up the module begins performing the following logical functions:
2 Initialize hardware components
   - o Initialize SLC backplane driver
   - o Test and Clear all RAM
   - o Initialize the serial communication ports

3 Wait for module configuration from the SLC processor
4 Allocate and initialize Module Register space
5 Enable Slave Driver on selected ports
6 Enable Master Driver on selected port if configured

After the module has received the Module Configuration, the module will begin communicating with other nodes on the network, depending on the configuration.

About the MODBUS Protocol

MODBUS is a widely-used protocol originally developed by Modicon in 1978. Since that time, the protocol has been adopted as a standard throughout the automation industry.

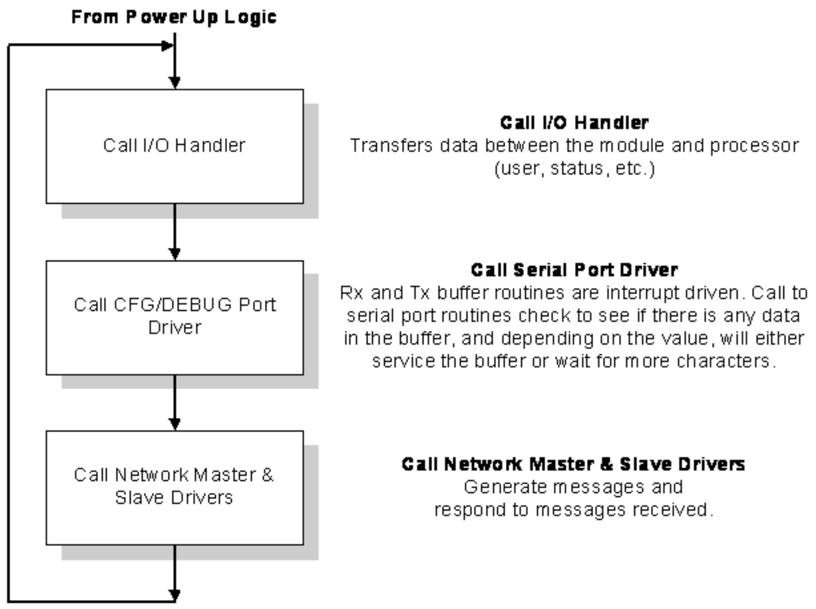
The original MODBUS specification uses a serial connection to communicate commands and data between Master and Slave devices on a network. Later enhancements to the protocol allow communication over other types of networks.

MODBUS is a Master/Slave protocol. The Master establishes a connection to the remote Slave. When the connection is established, the Master sends the MODBUS commands to the Slave. The MVI46-MCM module can work as a Master and as a Slave.

The MVI46-MCM module also works as an input/output module between itself and the Rockwell Automation backplane and processor. The module uses an internal database to pass data and commands between the processor and Master and Slave devices on MODBUS networks.

Main Logic Loop

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


From Power Up Logic
Call I/O Handler
Call I/O Handler
Transfers data between the module and processor (user, status, etc.)
Call CFG/DEBUG Port Driver
Call Serial Port Driver
Rx and Tx buffer routines are interrupt driven. Call to serial port routines check to see if there is any data in the buffer, and depending on the value, will either service the buffer or wait for more characters.
Call Network Master & Slave Drivers
Call Network Master & Slave Drivers
Generate messages and respond to messages received.

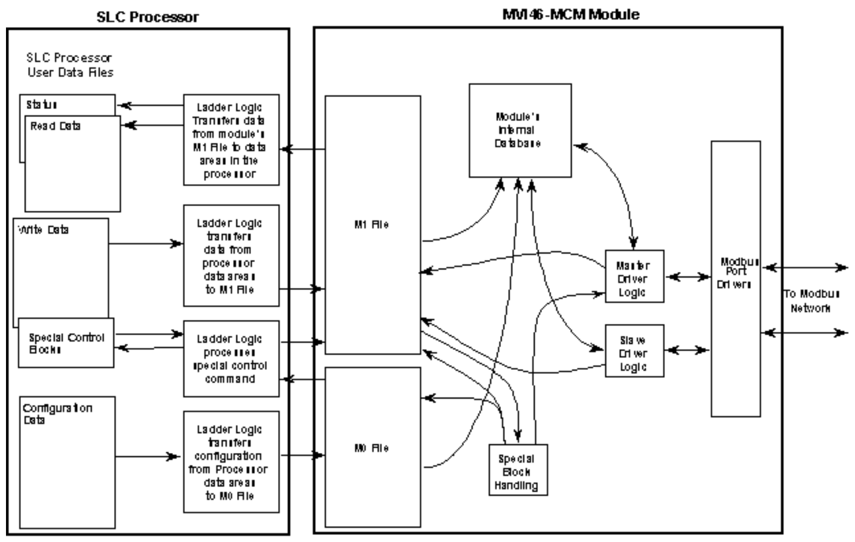
SLC Processor Not in Run

Whenever the module detects that the processor has gone out of the Run mode (that is, Fault or PGM), the Modbus ports will be shut down. When the processor is returned to a running state, the module will resume communications on the network. No backplane data transfers occur when the processor is not in run mode.

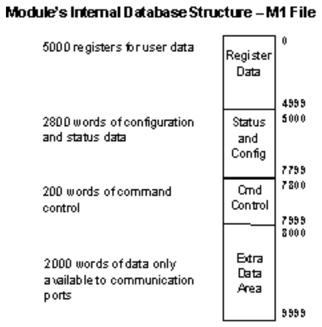
Backplane Data Transfer

The MVI46-MCM module communicates directly over the SLC backplane. All data for the module is contained in the module's M1 file. Data is moved between the module and the SLC processor across the backplane using the module's M-files. The SLC scan rate and the communication load on the module determine the update frequency of the M-files. The COP instruction can be used to move data between user data files and the module's M1 file.

The following illustration shows the data transfer method used to move data between the SLC processor, the MVI46-MCM module and the MCM network.


SLC Processor
MVI46-MCM Module
SLC Processor User Data Files
Status
Read Data
Write Data
Special Control Blocks
Configuration Data
Ladder Logic Transfers data from module's M1 File to data areas in the processor
Ladder Logic transfers data from processor data areas to M1 File
Ladder Logic processes special control command
Ladder Logic transfers configuration from Processor data areas to M0 File
M1 File
M0 File
Module's Internal Database
Master Driver Logic
Slave Driver Logic
Special Block Handling
Modbus Port Drivers
To Modbus Network

All data transferred between the module and the processor over the backplane is through the M0 and M1 files. Ladder logic must be written in the SLC processor to interface the M-file data with data defined in the user-defined data files in the SLC. All data used by the module is stored in its internal database. The following illustration shows the layout of the database:


Module's Internal Database Structure – M1 File
5000 registers for user data
Register Data
0
4999
2800 words of configuration and status data
Status and Config
5000
7799
200 words of command control
Cmd Control
7800
7999
2000 words of data only available to communication ports
Extra Data Area
8000
9999

User data contained in this database is continuously read from the M1 file. The configuration data is only updated in the M1 file after each configuration request by the module to the SLC. All data in the M1 file is available to devices on the Modbus Master/Slave networks. This permits data to be transferred from these devices to the SLC using the user data area. Additionally, remote devices can alter the module's configuration, read the status data, and issue control commands. Block identification codes define specific functions to the module.

The module uses the following block numbers:

M0 Offset	Description	Length
0	9001	1
1 to 6	Backplane Setup	6
11 to 40	Port 1 Configuration	30
41 to 70	Port 2 Configuration	30
71 to 80	Port 1 Command # 0 Definition	10
81 to 90	Port 1 Command # 1 Definition	10
91 to 1060	Port 1 Command # 2 to # 98	980
1061 to 1070	Port 1 Command # 99 Definition	10
1071 to 1080	Port 2 Command # 0 Definition	10
1081 to 1090	Port 2 Command # 1 Definition	10
1091 to 2060	Port 2 Command # 2 to # 98	980
2061 to 2070	Port 2 Command # 99 Definition	10

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

5.2.3 Normal Data Transfer

This version of the module provides for direct access to the data in the module. All data related to the module is stored in the module's M1 file. To read data from the module, use the COP instruction to copy data from the module's M1 file to a user data file. To write data to the module, use the COP instruction to copy data from a user file to the module's M1 file. Registers 0 to 4999 should be used for user data. All other registers are reserved for other module functions.

5.2.4 Special Function Blocks

Special Function blocks are special blocks used to control the module or request special data from the module. The current version of the software supports the following special function blocks:

- Event Command blocks
- Slave Status blocks
- Command Control blocks
- Write Configuration block
- Warm Boot block
- Cold Boot block

Event Command Blocks (1000 or 2000)

Event command control blocks send Modbus Master/Slave commands directly from the ladder logic to one of the master ports.

Block Request from Processor to Module

M1 Offset	**Description**	**Length**
7800	1000 or 2000	1
7801	Internal DB Address	1
7802	Point Count	1
7803	Swap Code	1
7804	Node Address	1
7805	Function Code	1
7806	Device Address	1

The block number defines the Modbus Master/Slave port to be considered. Block 1000 commands are directed to Port 1, and block 2000 commands are directed to Port 2. Use the parameters passed with the block 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 registers for the command. The **Swap Code** changes the word or byte order. The **Node Address** parameter defines the device on the Modbus Master/Slave network to consider. The **Function Code** parameter is one of those defined in the ProSoft Modbus Master/Slave Command Set documentation. When the block is received, the module will process it and place the command in the command queue.

Block Response from Module to Processor

M0 Offset	**Description**	**Length**
0	1000 or 2000	1
1	0=Fail, 1=Success	1

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

Slave Status Blocks (3000 to 3003 or 3100 to 3103)

Slave status data sends 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.
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 (Error Delay Counter value in the module configuration for each port). Each time a command in the list is scanned that has the address of a suspended slave, the delay counter value will be decremented. When the value reaches zero, the slave state will be set to one. This will enable polling of the slave.

In order to read the slave status table, ladder logic must be written and the slave status data must be located in the user data area. The module will constantly update the user defined data area with the slave data for each Modbus Master/Slave master port. This data can be transferred to a user-defined file in the processor using the COP instruction.

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

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. This does not indicate that the slave has responded to this message.
2	The master port has failed to communicate with the slave device. Communications with the slave is suspended for a user defined period based on the scanning of the command list.
3	Communications with the slave has been disabled by the ladder logic. No communication will occur with the slave until this state is cleared by the ladder logic.

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

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

Block Request from Processor to Module

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

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

Block Response from Module to Processor

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

Ladder logic can be written to override the value in the slave status table to disable slaves (state value of 3) by sending a special block of data from the processor to the slave. Port 1 slaves are disabled using block 3000, and Port 2 slaves are disabled using block 3100. Each block contains the slave node addresses to disable.

Block Request from Processor to Module

M1 Offset	Description	Length
7800	3000 or 3100	1
7801	Number of Slaves in Block	1
7802 to 7927	Slave Indexes	126

The module will respond with a block with the same identification code received and indicate the number of slaves acted on with the block.

Block Response from Module to Processor

M0 Offset	Description	Length
0	3000 or 3100	1
1	Number of Slaves Processed	1

Ladder logic can be written to override the value in the slave status table to enable the slave (state value of 1) by sending a special block. Port 1 slaves are enabled using block 3001, and Port 2 slaves are enabled using block 3101. Each block contains the slave node addresses to enable.

Block Request from Processor to Module

M1 Offset	Description	Length
7800	3001 or 3101	1
7801	Number of Slaves in Block	1
7802 to 7927	Slave Indexes	126

The module will respond with a block with the same identification code received and indicate the number of slaves acted on with the block.

Block Response from Module to Processor

M0 Offset	Description	Length
0	3001 or 3101	1
1	Number of Slaves Processed	1

Command Control Blocks (5001 to 5006 or 5101 to 5106)

Command Control blocks place commands from 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.

Block Request from Processor to Module

M1 Offset	Description	Length
7800	5001 to 5006 or 5101 to 5106	1
7801	Command Index	1
7802	Command Index	1
7803	Command Index	1
7804	Command Index	1
7805	Command Index	1
7806	Command Index	1

Blocks in the range of 5001 to 5006 are used for Port 1, while blocks in the range of 5101 to 5106 are used for Port 2. The last digit in the block code defines the number of commands to process in the block. For example, a block code of 5003 contains 3 command indexes for 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.

Block Response from Module to Processor

M0 Offset	Description	Length
0	5000 to 5006 or 5100 to 5106	1
1	Number of commands added to command queue	1

Configuration Data Transfer

When the module performs a restart operation, it will request configuration information from the SLC processor. This data is transferred to the module in a specially formatted write block in the M0 file. The module will poll for this information by placing the value 9000 in word 0 of the M0 file. The ladder logic must construct the requested block in order to configure the module. Refer to Module Data (page 28) for a description of the data objects used with the blocks and the ladder logic required. The format of the block for configuration is given in the following section.

Module Configuration Data (9001)

This block sends configuration information from the processor to the module. The data is transferred in a block with an identification code of 9001.

Configuration Block from Processor to Module

M0 Offset	Description	Length
0	9001	1
1 to 6	Backplane Setup	6
11 to 40	Port 1 Configuration	30
41 to 70	Port 2 Configuration	30
71 to 80	Port 1 Command # 0 Definition	10
81 to 90	Port 1 Command # 1 Definition	10
91 to 1060	Port 1 Command # 2 to # 98	980
1061 to 1070	Port 1 Command # 99 Definition	10
1071 to 1080	Port 2 Command # 0 Definition	10
1081 to 1090	Port 2 Command # 1 Definition	10
1091 to 2060	Port 2 Command # 2 to # 98	980
2061 to 2070	Port 2 Command # 99 Definition	10

If there are any errors in the configuration, the bit associated with the error will be set in one of the two configuration error words. The error must be corrected before the module starts operating.

Write Configuration Block (9001)

This block is sent from the SLC processor, and causes 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 9997 in the first word. The module responds with a block containing the module configuration data. Ladder logic must handle the receipt of the block.

Block Response from Module to Processor

M0 Offset	Description	Length
0	9001	1
1 to 9	Backplane Setup	9
11 to 40	Port 1 Configuration	30
41 to 70	Port 2 Configuration	30
71 to 80	Port 1 Command # 0 Definition	10
81 to 90	Port 1 Command # 1 Definition	10
91 to 1060	Port 1 Command # 2 to # 98	980
1061 to 1070	Port 1 Command # 99 Definition	10
1071 to 1080	Port 2 Command # 0 Definition	10
1081 to 1090	Port 2 Command # 1 Definition	10
1091 to 2060	Port 2 Command # 2 to # 98	980
2061 to 2070	Port 2 Command # 99 Definition	10

Ladder logic must process this block of information and place the data received in the correct data files in the SLC.

Block Request from Processor to Module

M1 Offset	Description	Length
7800	9997	1

Warm Boot Block (9998)

This block is sent from the SLC processor to the module 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 configuration data area. This causes the module to read the new configuration information and restart. The following table describes the format of the Warm Boot block.

Block Request from Processor to Module

M1 Offset	Description	Length
7800	9998	1

Cold Boot Block (9999)

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

Block Request from Processor to Module

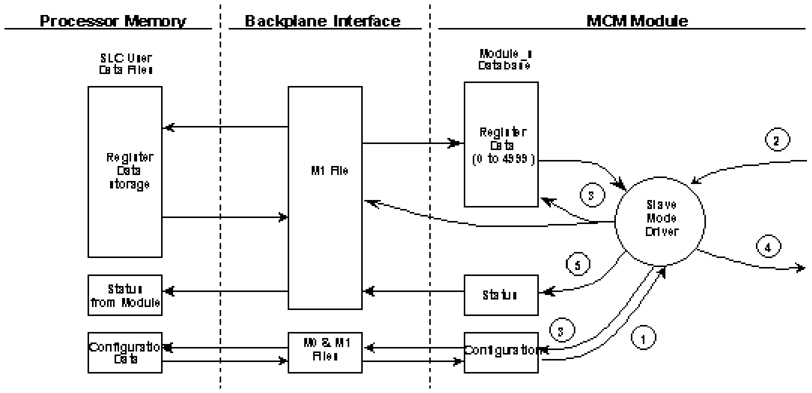
M1 Offset	Description	Length
7800	9999	1

5.2.5 Data Flow Between MVI46-MCM Module and SLC Processor

The following topics describe the flow of data between the two pieces of hardware (SLC processor and MVI46-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 depends on this configuration.

Slave Driver

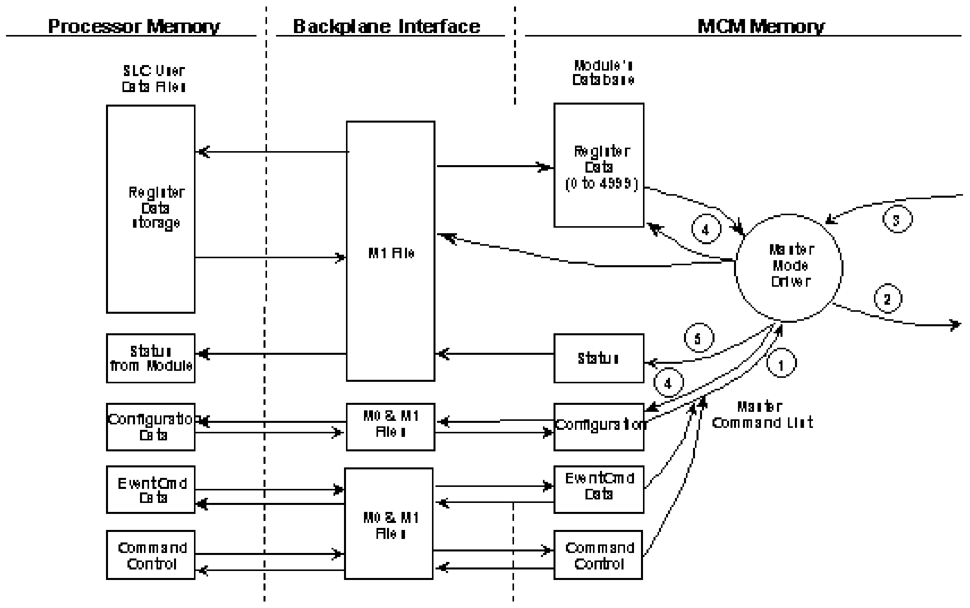
The Slave Driver Mode allows the MVI46-MCM module to respond to data read and write commands issued by a master on the MODBUS network. The following flow chart and associated table describe the flow of data in and out of the module.



Step	Description
1	The MODBUS slave port driver receives the configuration information from the SLC processor. This information configures the serial port and defines the slave node characteristics.
2	A Host device, such as the Rockwell Automation PLC or an HMI application issues a read or write command to the module's node address. The port driver qualifies the message before accepting it into the module.
3	After the module accepts the command, the data is immediately transferred to or from the internal database in the module. If the command is a read command, the data is read out of the database and a response message is built. If the command is a write command, the data is written directly into the database and the M1 file and a response message is built.
4	After the data processing has been completed in Step 3, 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.

Master Driver Mode

In the Master mode, the MVI46-MCM module issues read or write commands to slave devices on the MODBUS network. These commands are user-configured in the module via the Master Command List received from the SLC processor or issued directly from the SLC processor (Event Commands or Command Control). Command status is returned to the processor for each individual command in the command list status block. The location of this status block in the module's internal database is user-defined. The following flow chart and associated table describe the flow of data in and out of the module.



Step	Description
1	The Master driver obtains configuration data from the SLC 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 (Refer to the Module Set Up section).
2	After configuration, the Master driver begins transmitting read and/or write commands to the other nodes on the network. If writing data to another node, the data for the write command is obtained from the module's internal database to build the command.
3	Presuming successful processing by the node specified in the command, a response message is received into the Master driver for processing.
4	Data received from the node on the network is passed into the module's internal database and the M1 file, assuming a read command.
5	Status is returned to the SLC processor for each command in the Master Command List.

Refer to the **Module Set Up** section for a complete description of the parameters required to define the virtual Modbus master port. Refer to Modbus Protocol Specification (page 83) 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 may not be as desired. All commands containing invalid data are ignored by the module.

Master Command List

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

- Command enable mode: (0) disabled, (1) continuous or (2) conditional
- Slave Node Address
- Command Type: Read or Write up to 125 words (16000 bits) per command
- Database Source and Destination Register Address: The addresses where data will be written or read.
- Count: The number of words to be transferred - 1 to 125 on FC 3, 4, or 16. Select the number of bits on FC 1, 2, 15.

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

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

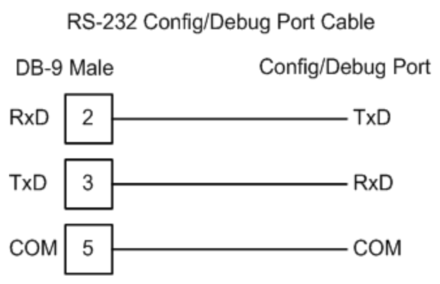
5.3 Cable Connections

The application ports on the MVI46-MCM module support RS-232, RS-422, and RS-485 interfaces. Please inspect the module to ensure that the jumpers are set correctly to correspond with the type of interface you are using.

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

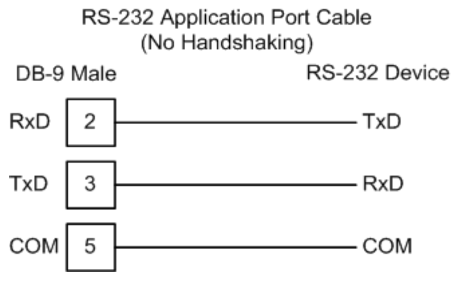
5.3.1 RS-232 Configuration/Debug Port

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



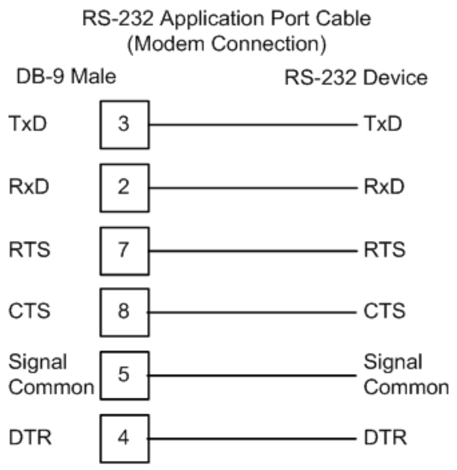
5.3.2 RS-232 Application Port(s)

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



RS-232: Modem Connection (Hardware Handshaking Required)

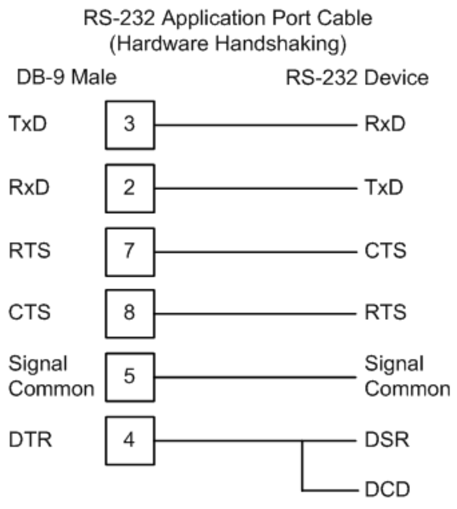
This type of connection is required between the module and a modem or other communication device.



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

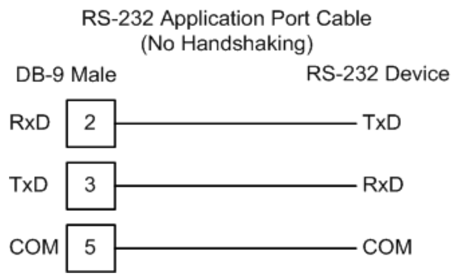
RS-232: Null Modem Connection (Hardware Handshaking)

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

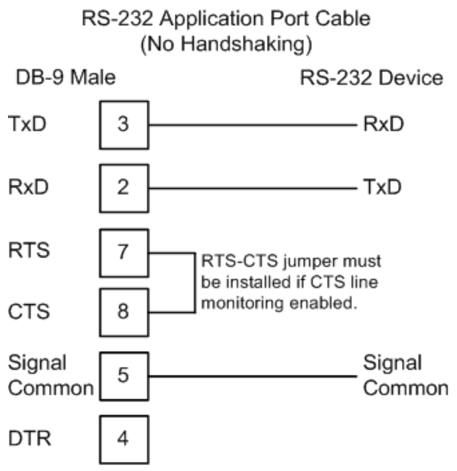


RS-232: Null Modem Connection (No Hardware Handshaking)

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

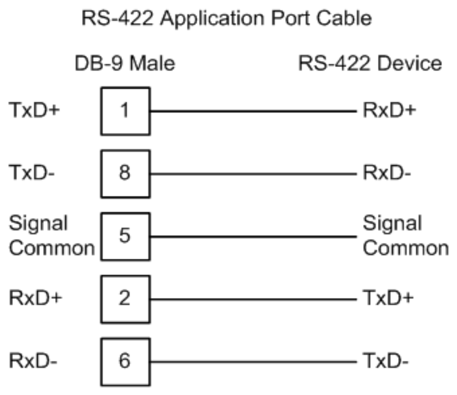


Note: For most null modem connections where hardware handshaking is not required, the *Use CTS Line* parameter should be set to N and no jumper will be required between Pins 7 (RTS) and 8 (CTS) on the connector. If the port is configured with the *Use CTS Line* set to Y, then a jumper is required between the RTS and the CTS lines on the port connection.


RS-232 Application Port Cable
(No Handshaking)
DB-9 Male
RS-232 Device
TxD
3
RxD
RxD
2
TxD
RTS
7
CTS
8
RTS-CTS jumper must be installed if CTS line monitoring enabled.
Signal Common
5
Signal Common
DTR
4

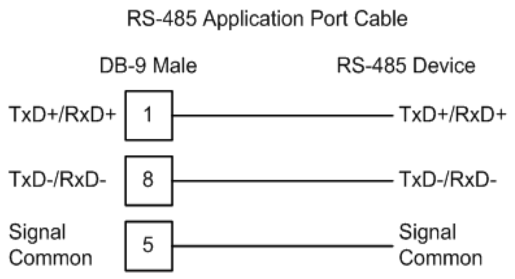
5.3.3 RS-422

The RS-422 interface requires a single four or five wire cable. The Common connection is optional, depending on the RS-422 network devices used. The cable required for this interface is shown below:



5.3.4 RS-485 Application Port(s)

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

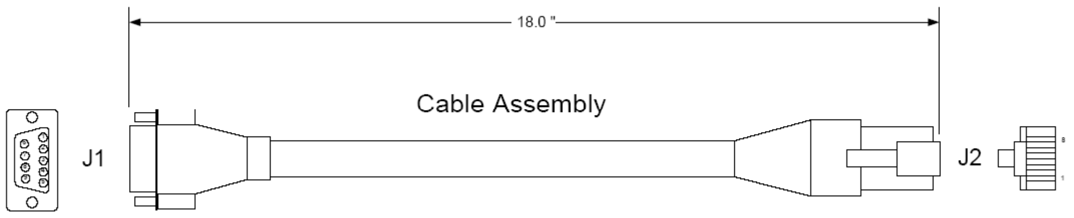


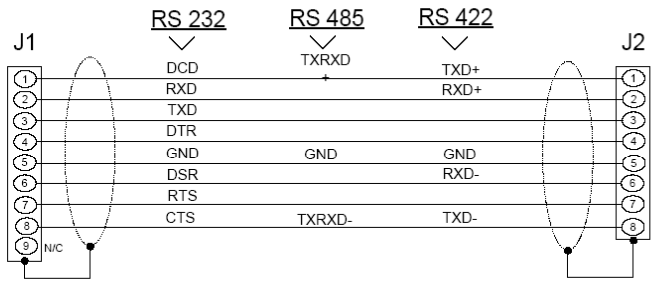
Note: Terminating resistors are generally not required on the RS-485 network, unless you are experiencing communication problems that can be attributed to signal echoes or reflections. In these cases, installing a 120-ohm terminating resistor between pins 1 and 8 on the module connector end of the RS-485 line may improve communication quality.

RS-485 and RS-422 Tip

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

5.3.5 DB9 to RJ45 Adaptor (Cable 14)





Wiring Diagram

5.4 MVI46-MCM Database Definition

This section contains a listing of the internal database of the MVI46-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 to 4999	40001	45000	User Data	5000
5000 to 5009	45001	45010	Backplane Configuration	10
5010 to 5039	45011	45040	Port 1 Setup	30
5040 to 5069	45041	45070	Port 2 Setup	30
5200 to 6199	45071	46070	Port 1 Commands	1000
6200 to 7199	46071	47070	Port 2 Commands	1000
7200 to 7232	47201	47233	Misc. Status Data	33
7600 to 7799	47601	48000	Command Control	200

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

5.5 MVI46-MCM Remote Configuration

Remote configuration data can be received from other nodes on the network that can control the MVI46-MCM module. Specific values are written to regions of this block to change the module's configuration parameters. 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.

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

5.6 Modbus Protocol Specification

The following pages give additional reference information regarding the Modbus protocol commands supported by the MVI46-MCM.

5.6.1 Commands Supported by the Module

The format of each command in the list depends on the MODBUS Function Code being executed.

The following table lists the functions supported by the module.

Function Code	**Definition**	**Supported in Master**	**Supported in Slave**
1	Read Coil Status	X	X
2	Read Input Status	X	X
3	Read Holding Registers	X	X
4	Read Input Registers	X	X
5	Set Single Coil	X	X
6	Single Register Write	X	X
8	Diagnostics		X
15	Multiple Coil Write	X	X
16	Multiple Register Write	X	X
17	Report Slave ID		X
22	Mask Write 4X		X
23	Read/Write		X

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

5.6.2 Read Coil Status (Function Code 01)

Query

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

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

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

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

Response

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

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

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

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

5.6.3 Read Input Status (Function Code 02)

Query

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

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

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

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

Response

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

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

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

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

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

5.6.4 Read Holding Registers (Function Code 03)

Query

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

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

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

Response

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

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

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

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

5.6.5 Read Input Registers (Function Code 04)

Query

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

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

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

Response

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

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

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

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

5.6.6 *Force Single Coil (Function Code 05)*

Query

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

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

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

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

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

Response

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

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

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

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

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

5.6.7 Preset Single Register (Function Code 06)

Query

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

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

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

Response

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

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

5.6.8 *Diagnostics (Function Code 08)*

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

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

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

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

Sub-function Codes Supported

Only Sub-function 00 is supported by the MVI46-MCM module.

00 Return Query Data

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

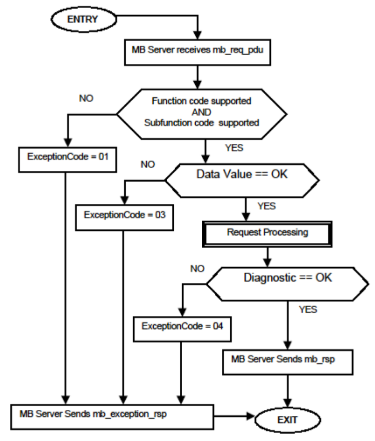
Sub-function	**Data Field (Request)**	**Data Field (Response)**
00 00	Any	Echo Request Data

Example and State Diagram

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

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

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



5.6.9 Force Multiple Coils (Function Code 15)

Query

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

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

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

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

Response

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

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

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

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

5.6.10 Preset Multiple Registers (Function Code 16)

Query

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

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

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

Response

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

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

5.6.11 Modbus Exception Responses

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

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

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

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

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

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

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

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

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

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

Modbus Exception Codes

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

Code	Name	Meaning
0a	Gateway Path Unavailable	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.
0b	Gateway Target Device Failed To Respond	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

6 Support, Service & Warranty

In This Chapter

Contacting Technical Support

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

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

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

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

Note: *For technical support calls within the United States, an after-hours answering system allows 24-hour/7-days-a-week pager access to one of our qualified Technical and/or Application Support Engineers. Detailed contact information for all our worldwide locations is available on the following page.*

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

6.1 Return Material Authorization (RMA) Policies and Conditions

The following Return Material Authorization (RMA) Policies and Conditions (collectively, "RMA Policies") apply to any returned product. These RMA Policies are subject to change by ProSoft Technology, Inc., without notice. For warranty information, see Limited Warranty (page 101). In the event of any inconsistency between the RMA Policies and the Warranty, the Warranty shall govern.

6.1.1 Returning Any Product

a) In order to return a Product for repair, exchange, or otherwise, the Customer must obtain a Return Material Authorization (RMA) number from ProSoft Technology and comply with ProSoft Technology shipping instructions.
b) In the event that the Customer experiences a problem with the Product for any reason, Customer should contact ProSoft Technical Support at one of the telephone numbers listed above (page 97). A Technical Support Engineer will request that you perform several tests in an attempt to isolate the problem. If after completing these tests, the Product is found to be the source of the problem, we will issue an RMA.
c) All returned Products must be shipped freight prepaid, in the original shipping container or equivalent, to the location specified by ProSoft Technology, and be accompanied by proof of purchase and receipt date. The RMA number is to be prominently marked on the outside of the shipping box. Customer agrees to insure the Product or assume the risk of loss or damage in transit. Products shipped to ProSoft Technology using a shipment method other than that specified by ProSoft Technology, or shipped without an RMA number will be returned to the Customer, freight collect. Contact ProSoft Technical Support for further information.
d) A 10% restocking fee applies to all warranty credit returns, whereby a Customer has an application change, ordered too many, does not need, etc. Returns for credit require that all accessory parts included in the original box (i.e.; antennas, cables) be returned. Failure to return these items will result in a deduction from the total credit due for each missing item.

6.1.2 Returning Units Under Warranty

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

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

6.1.3 Returning Units Out of Warranty

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

The following is a list of non-repairable units:

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

6.2 LIMITED WARRANTY

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

6.2.1 What Is Covered By This Warranty

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

6.2.2 What Is Not Covered By This Warranty

a) ProSoft makes no representation or warranty, expressed or implied, that the operation of software purchased from ProSoft will be uninterrupted or error free or that the functions contained in the software will meet or satisfy the purchaser's intended use or requirements; the Customer assumes complete responsibility for decisions made or actions taken based on information obtained using ProSoft software.

b) This Warranty does not cover the failure of the Product to perform specified functions, or any other non-conformance, defects, losses or damages caused by or attributable to any of the following: (i) shipping; (ii) improper installation or other failure of Customer to adhere to ProSoft's specifications or instructions; (iii) unauthorized repair or maintenance; (iv) attachments, equipment, options, parts, software, or user-created programming (including, but not limited to, programs developed with any IEC 61131-3, "C" or any variant of "C" programming languages) not furnished by ProSoft; (v) use of the Product for purposes other than those for which it was designed; (vi) any other abuse, misapplication, neglect or misuse by the Customer; (vii) accident, improper testing or causes external to the Product such as, but not limited to, exposure to extremes of temperature or humidity, power failure or power surges; or (viii) disasters such as fire, flood, earthquake, wind and lightning.

c) The information in this Agreement is subject to change without notice. ProSoft shall not be liable for technical or editorial errors or omissions made herein; nor for incidental or consequential damages resulting from the furnishing, performance or use of this material. The user guide included with your original product purchase from ProSoft contains information protected by copyright. No part of the guide may be duplicated or reproduced in any form without prior written consent from ProSoft.

6.2.3 Disclaimer Regarding High Risk Activities

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

6.2.4 Intellectual Property Indemnity

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

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

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

6.2.5 Disclaimer of all Other Warranties

The Warranty set forth in What Is Covered By This Warranty (page 101) are in lieu of all other warranties, express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

6.2.6 Limitation of Remedies **

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

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

6.2.7 Time Limit for Bringing Suit

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

6.2.8 No Other Warranties

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

6.2.9 Allocation of Risks

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

6.2.10 Controlling Law and Severability

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

Index