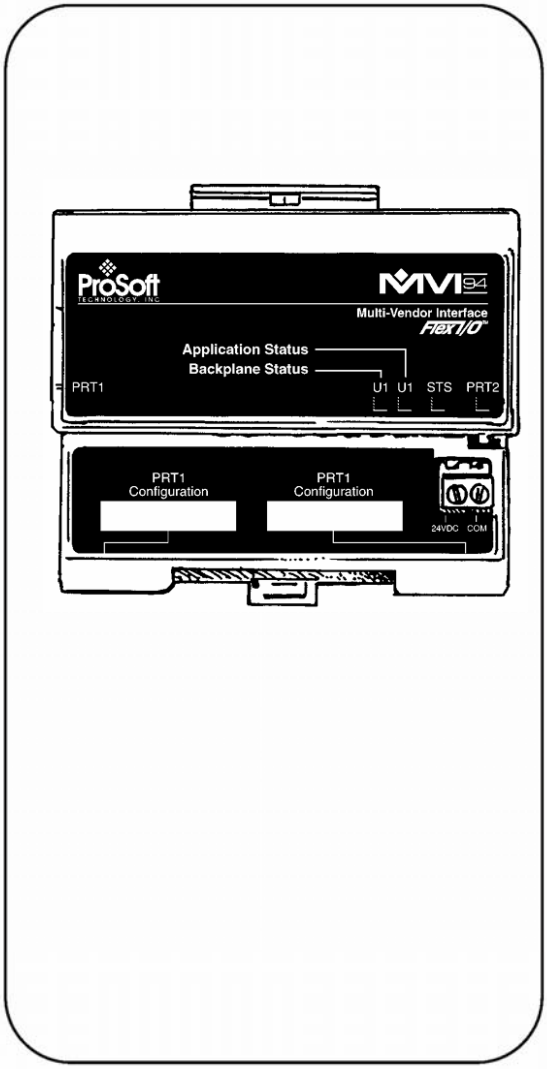
inRAx



MVI94-MBM

Modbus Master Serial Communications

Flex I/O Serial Communications Module

User Manual

November 18, 2004

ProSoft TECHNOLOGY

Please Read This Notice

Successful application of this module requires a reasonable working knowledge of the Modbus Master Serial Communications Flex I/O Serial Communications Module hardware and the application in which the combination is to be used. For this reason, it is important that those responsible for implementation satisfy themselves that the combination will meet the needs of the application without exposing personnel or equipment to unsafe or inappropriate working conditions.

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

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

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

ProSoft Technology, Inc.

1675 Chester Avenue, 2nd Floor

Bakersfield, CA 93301

(661) 716-5100

(661) 716-5101 (Fax)

www.prosoft-technology.com

MVI94-MBM User Manual

November 18, 2004

Table of Contents

1 Introduction

The MVI94-MBM communication module is used to interface Modbus slave devices with the Flex I/O system. The module contains a virtual Modbus database that is defined by the user. This database is used for the request and command messages sent from the Modbus master port to Modbus slave devices. Data areas in the virtual Modbus database can be reserved for status and error information generated by the module under user control.

The virtual Modbus database also interfaces with the Flex I/O system using the Flex I/O bus (backplane). Data is made available to the PLC or any processor on a ControlNet network using this backplane interface. Input and output image tables in the module are used to present the data in the virtual Modbus database to the backplane.

A Modbus master port is present on the communication module to continuously poll Modbus slave devices. Up to 100 user-defined commands can be defined for the port. Data read from Modbus slave devices are placed in the virtual Modbus database. Any write requests for the Modbus slave devices are sourced with data from the virtual Modbus database.

The module can be configured to place slave devices that are not responding to commands at a lower priority. If the module recognizes a slave device has failed to respond to a message after the user defined retry count, it will mark the slave as "in communication failure" and set the error delay counter to the user specified value. Each time the module encounters this slave in the command list, the counter will be decremented. When the value reaches zero, the slave will be placed in an active status. This facility can improve communication throughput on the Modbus network.

Commands can be activated in the module under processor control. This feature permits the processor to issue a command in the command list under program control. When a command is activated, it will be placed in the command queue for immediate execution. Normal command polling will begin after the command queue is completely processed.

Additionally, the processor can send a command directly to any slave attached to the Modbus master port. This feature can be used for commands that are not issued on a regular basis and are not in the command list. Commands submitted as events are placed in the command queue for immediate execution. They will preempt normal operation of the poll list. When the command queue is completely empty, normal command polling will resume.

The module provides a Configuration/Debug port for use with an external computer executing a terminal emulation program. The terminal emulation program provided with the module permits uploading and downloading of the configuration information required by the module. Additionally, the Configuration/Debug port provides a view into the virtual Modbus database, communication statistics and the configuration.

1.1 Reference and Resources

Several resources are available to assist with the configuration and support of the MVI94 module. The following files are available from the ftp site:

ftp://ftp.prosoft-technology.com/pub/Manuals/

MVI94MBM.CFG	Example configuration file
MVI94_MBM_User_Manual.pdf	This User Manual in pdf format

ftp://ftp.prosoft-technology.com/pub/Ladder/

mvi94mbmc.rsp	Example PLC5 Ladder program

ftp://ftp.prosoft-technology.com/pub/Utilities/PSTerm/

DOS_PSTerm.ZIP	DOS program for configuration
WIN_PSTerm.	ZIPWin95/98/NT program for config

The following references are available for the MVI94-GSC from the ftp/web site:

Documentation (ftp.prosoft-technology.com/pub/Manuals/MVI94_GSC)

- MVI94-GSC Flex I/O Serial Communications Module Installation Instructions
- MVI94-GSC Quick Start Guide

Example Ladder (ftp.prosoft-technology.com/pub/Ladder/MVI94_GSC)

- Example PLC 5 program (as documented in Section 5)

1.2 Definitions

Backplane	Refers to the electrical interface, or bus, to which IO modules connect when inserted into the rack. The IO modules communicate with the control processor through the backplane.
BIOS	Basic Input Output System. The BIOS firmware initializes the module at power-on and provides a DOS compatible interface to the console and ROM disk.
Control	The PLC or other controlling processor which communicates with the MVI
Processor	module directly over the backplane or via a network or remote IO adapter.
Input Image	Refers to a contiguous block of data that is written by the module application and read by the control processor. The input image is read by the control processor once each scan. Also called the input file.
Output Image	Refers to a contiguous block of data that is written by the control processor and read by the module application. The output image is written by the control processor once each scan (only in Run mode). Also called the output file.

1.3 MVI94 Functional Features

Hardware

Port 1 : RS-232 only

Port 2 : RS-232/422/485 (Jumper configurable)

Software

- Bi-directional, full duplex serial data transfer
- Each serial port independently configurable for baud rate, word length, parity, stop bits, handshaking, and data termination condition
- Supports baud rates of 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600, and 115200
- Three data termination conditions available: termination character, data length, and data timeout
- Supports transmitted or received data up to 1023 bytes in length
- LED diagnostic indicators aid debugging
- Error messages help debug invalid configurations
- Enhanced reliability provided by the 'heartbeat' - module is reinitialized in the event of any communications disruption between the controller and module (for example, if the module power is cycled off and on)

1.4 Configuration Jumpers

One set of jumpers is used to configure the user serial port PRT2 for RS-232, RS-485, or RS-422 (Note that PRT1 is RS-232 only). See the **Installation Instructions** for more information.

Another jumper, the ‘Setup’ jumper, is used to enable the console on PRT1. The Setup Jumper should be placed in the ‘removed’ position when operating the MVI94-GSC module (default position from factory). If the Setup Jumper is installed, the GSC module’s executable application will be disabled, and the module will boot to a DOS prompt. See the **Quick Start Guide** for more information.

2 Quick Start Guide

This section describes the procedure to be followed for setting up the module for communications. These steps should be followed for successful implementation of a module in a user application.

1. Define the communication characteristics of the Modbus master port.
2. Define the command lists to be used on the Modbus master port.
3. Fill in the blank configuration form for application using the data sets defined in steps one and two.
4. Edit the configuration text file **MVI94MBM.CFG** (using Notepad or some other text editor)to reflect the desired data from the configuration form and save the file under a different name. This example CFG file is shipped in the module's memory, and is available off the ftp site.
5. Connect the module to a 24 VDC power source.
6. Connect the MVI94-MBM module's Configuration/Debug Port to a computer containing ProSoft's terminal emulation program (PS_TERM.EXE) with a null-modem cable.
7. Select the directory containing the correct configuration file on the computer.
8. Start the terminal emulation program on the computer.
9. Press the '**?**' key on the terminal to be certain the module is communicating with the computer and that the main menu mode is current.
10. Press the '**R**' key on the terminal emulator to select the receive option. Immediately press the '**Y**' key.
11. Press the **[ALT-F3]** key on the terminal emulator and enter name of the configuration file to load into the module [**MVI94MBM.CFG** if using example file]. The configuration will be downloaded, and the module will restart using the new configuration.
12. Connect the module's Modbus master port to the Modbus network. If everything is configured correctly and the cable connections are correct, communications should be present on the master port.
13. Monitor the communication statistics for the port to be certain everything is working correctly.
14. View the virtual Modbus database in the module using the terminal emulator.
15. Create the ladder logic program for your system. An example PLC5 ladder program is available on the ftp site. This logic is responsible for transferring the data between the module and processor.

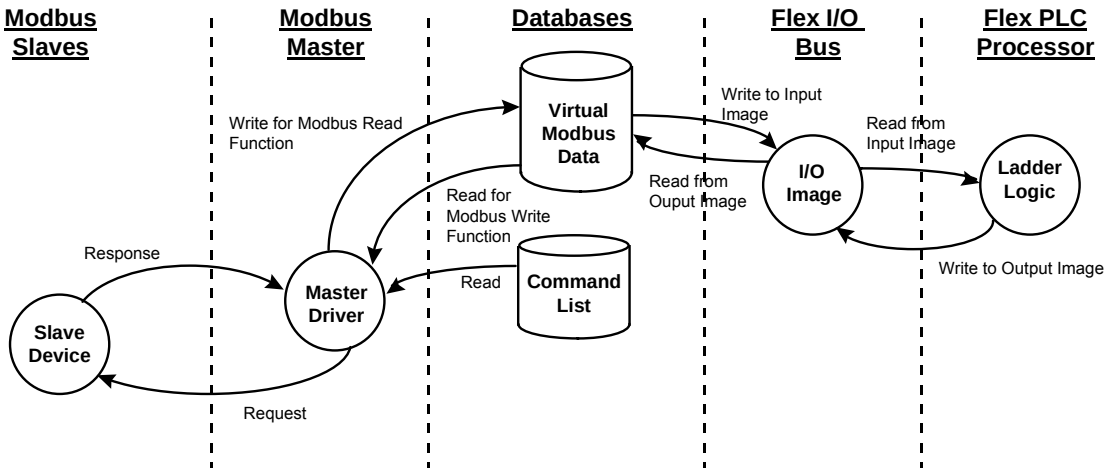
16. Connect the module to the Flex I/O communication adapter. If all is configured correctly, the data in the module should be visible in the processor.

17. Use the Configuration/Debug port to view the backplane transfer statistics.

3 Virtual Modbus Database Concept

Central to the functionality of the module is the virtual Modbus database. This database is used as the interface between remote Modbus slave devices and the Flex I/O bus. The size, content and structure of the database are completely user defined.

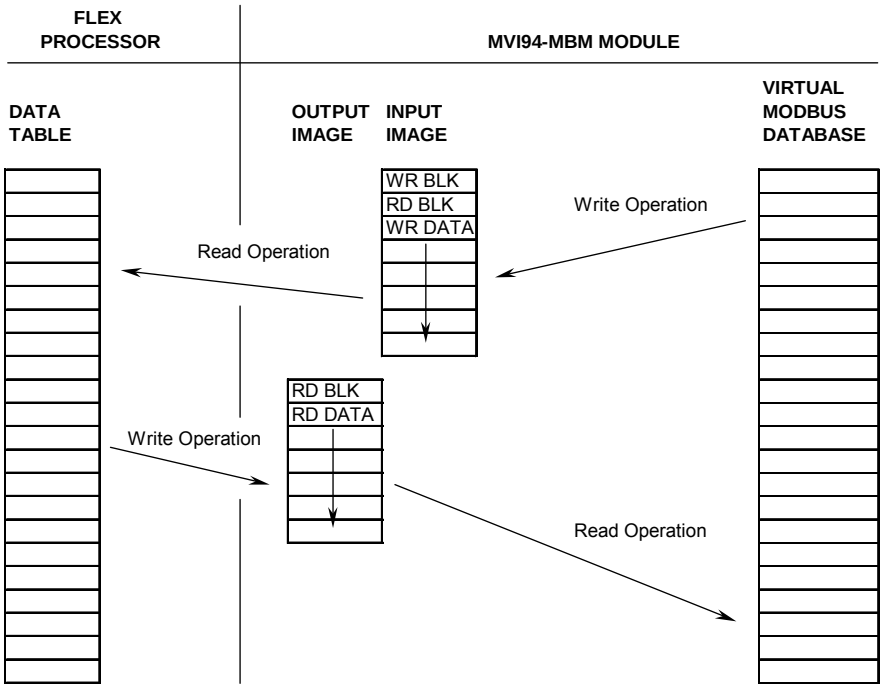
The Flex I/O bus is used to read data from and write data to the database using the backplane interface. The module interfaces data contained in remote Modbus slave devices to the virtual Modbus database using the Modbus master port. User commands are issued out the master port from a command list. These commands gather or control data in the Modbus slave devices. The diagram below displays the relationships discussed above:



4 DATA TRANSFER

Data is transferred over the backplane using the module's input and output images. The module is configured with an eight-word input image and a seven-word output image. The module and the Flex processor use these images to page data and commands. The input image is set (written) by the module and is read by the Flex processor. The output image is set (written) by the Flex processor and read by the module. The diagram below displays this relationship:

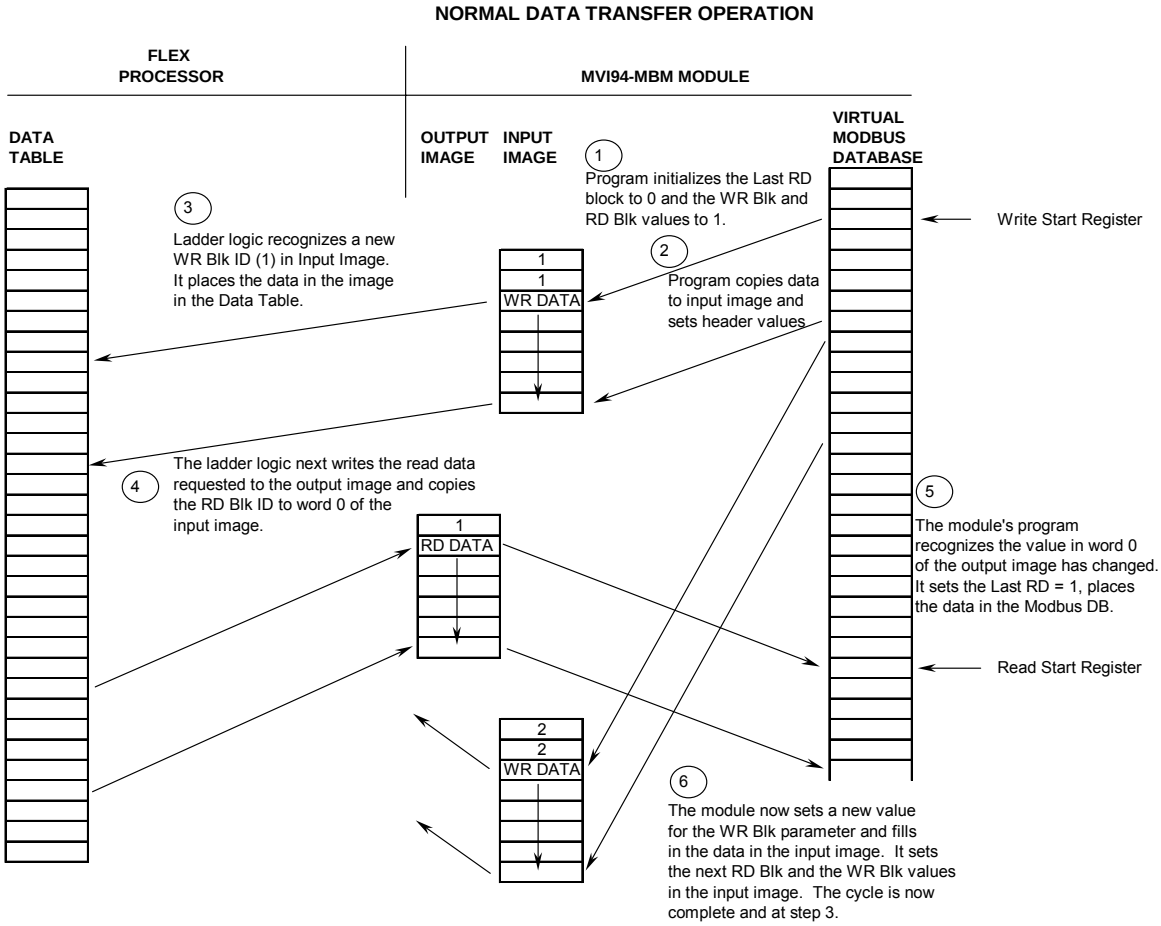
DATA TRANSFER OPERATION



The module's program is responsible for setting the block identification code used to identify the data block written and the block identification code of the block it wants to read from the processor. User configuration information determines the read (Read Start Register) and write (Write Start Register) locations in the virtual Modbus database and the amount of data transferred (Read Register Count and Write Register Count). Each read and write operation transfers a six-word data area. The write operation contains a two-word header that defines the block identification code of the write data and the block identification code of the read block requested. These identification codes are in the range of 0 to 666. A value of zero indicates that the block contains no data and should be ignored. The first valid block identification code is one and refers to the first block of six words to be read or written.

The module and the processor constantly monitor input and output images. How does either one know when a new block of data is available? Recognizing a change in the header information of the image (word 0) solves the problem. For example,

when the module recognizes a different value in the first word of the output image, new read data is available. When the processor recognizes a new value in the first word of the input image, new write data is available. This technique requires the storage of the previously processed data block identification code. The diagram below shows the normal sequence of events for data transfer:



The steps outlined in the diagram above are discussed below:

1. During program initialization, the write and read block identification codes are set to one. The last block read variable is set to zero.

2. The program copies the first six-word block of the virtual Modbus database starting at the user defined Write Start Register to the input image (words 2 to 7). It then sets the current read block code in word 1 of the input image. To "trigger" the write operation, the program places the current write block code into word 0 of the input image.

3. The Flex processor recognizes a new value in word 0 of the input image (based on the last_write_block_code not equal to write_block_code) in its ladder logic. The ladder logic computes the offset into the file based on the following formula:

write_file_offset = (write_block_code - 1) * 6

The new data contained in the input image (words 2 to 7) is copied to the offset in the processor's user data file. The last_write_block_code storage register in the processor is updated with the new write_block_code.

NOTE: If the data area transferred from the module exceeds the size of a single user file in the Flex processor, logic will be required to handle multiple files.

4. The ladder logic next examines the value of the read_block_code and computes the offset into the read data file as follows:

read_file_offset = (read_block_code - 1) * 6

The required 6-word, read data is copied to the module's output image (words 1 to 6). To "trigger" the transfer operation, the ladder logic moves the read_block_code into word 0 of the output image.

5. The module's program recognizes the new read_block_code. It transfers the data to the correct offset in the virtual Modbus database using the following function:

Modbus_offset = Read_Start_Register + (read_block_code - 1) * 6

The module sets the last_read_block_code to the value of read_block_code.

6. The module now selects the next read and write blocks. The data for the write operation is placed in the input image and the read_block_code is set. The module "triggers" the transfer operation by setting the new write_block_code in word 0 of the input image. The sequence continues at step 3.

The discussion above is for normal data transfer operation. The table below lists the block identification codes used by the module.

BLOCK IDENTIFICATION CODES

TYPE	NUMBER	DESCRIPTION
R/W	1 TO 666	Data blocks used to transfer data from the module to the backplane and from the backplane to the module. The module's input/output images are used for the data transfers.
R	1000 to 1255	These blocks are used to transfer event messages (Modbus commands) from a device on the backplane to the module. The block number contains the slave module to be used with the command. For example, to use slave device 5 attached to the master port, the block number 1005 should be used.
R	2001 to 2006	These blocks are used to transfer a list of commands to execute from a device on the backplane to the module. The number of commands to be considerd is coded in the block number. For example, to add three commands to the command queue, use block 2003.
R	9998	Warm boot the module. When the module receives this block, it will reset all program values using the configuration data.
R	9999	Cold boot the module. When the module receives this block, it will perform a hardware restart.

Data is transferred between the processor and the module using the block identification codes of 1 to 666. The other block codes are used to control the module from the processors ladder logic. They are implemented when the ladder logic needs to control the module. In order to use one of the blocks, the ladder logic inserts the data and code in the output image of the module. The data should be set before the code is placed in the block. This operation should be performed after the receipt of a new write block from the module. A discussion of each set of codes is given below:

4.1 EVENTS (BLOCKS 1000 TO 1255)

These control blocks are sent from the processor to the module to execute a Modbus command out the Modbus master port. It should be used for commands that are not found in the command list. Use blocks 2001 to 2006 to execute commands in the list under processor control. The format for this block is shown below:

BLOCK 1000 TO 1255 STRUCTURE

WORD	DESCRIPTION
0	This word contains the slave device address on the Modbus network to be considered with the command. This value is added (0 to 255) to the value of 1000. This generates a block identification code of 1000 to 1255.
1	This word contains the internal Modbus address in the module to be used with the command.
2	This word contains the count parameter that determines the number of digital points or registers to associate with the command.
3	The parameter specifies the swap type for the data. This function is only valid for function code 3.
4	This word contains the Modbus function code to be used with the command.
5	This word contains the Modbus address in the slave device to be associated with the command.
6	Reserved

The data contained in the block is used by the module to construct a valid Modbus command and defines what data in the virtual Modbus database to use with the command. These parameters are the same as defined for the command list. Refer to the Commands section of the documentation for further information.

When the module receives a block 1000 to 1255, it will check the command queue for an available position. If there is space in the command queue, the module will construct a command and place it in the queue. The module will respond with the following block in the input image after processing request.

BLOCK 1000 TO 1255 RESPONSE

WORD	DESCRIPTION
0	This word contains the block 1000 to 1255 requested by the processor.
1	This word contains the next read request block identification code.
2	This word contains the result of the event request. If a value of one is present, the command was issued. If a value of zero is present, no room was found in the command queue.
3	Not used.
4	Not used.
5	Not used.
6	Not used.

The ladder logic can examine word 2 of the input image to determine if the module was able to execute the command. If invalid parameters are set in the event request block, the command may still be placed in the queue and there will be no error indication.

4.2 COMMANDS (BLOCKS 2001 TO 2006)

These control blocks are sent from the processor to the module to execute one or more commands in the module's command list out the Modbus master port. Commands selected for execution need not have the Enable Code set (1 or 2) but must be valid commands. The format for this block is shown below:

BLOCK 2001 TO 2006 STRUCTURE

WORD	DESCRIPTION
0	Command queue block identification code of 2001 to 2006.
1	This word contains the index in the command list for the first command to be entered into the command queue.
2	This word contains the index in the command list for the second command to be entered into the command queue.
3	This word contains the index in the command list for the third command to be entered into the command queue.
4	This word contains the index in the command list for the fourth command to be entered into the command queue.
5	This word contains the index in the command list for the fifth command to be entered into the command queue.
6	This word contains the index in the command list for the sixth command to be entered into the command queue.

When the module receives one of these blocks, it examines word 0 of the output image. This word defines the number of commands contained in the block. The command count is determined by subtracting 2000 from the word value. This permits the controller to set from one to six commands into the command queue. The indexes submitted in the block should be valid for the command list. After the module determines the number of commands to consider, it inserts each command in the command queue. The response message sent from the module to the processor is as follows:

BLOCK 2001 TO 2006 RESPONSE

WORD	DESCRIPTION
0	This word contains the block 2001 to 2006 requested by the processor.
1	This word contains the next read request block identification code.
2	This word contains the number of commands in the block placed in the command queue.
3	Not used.
4	Not used.
5	Not used.
6	Not used.

The ladder logic can examine word 2 of the input image to determine the number of commands placed in the command queue. Any errors associated with the command can be viewed in the command list error table.

4.3 WARM-BOOT (BLOCK 9998)

This block does not contain any data. When the processor places a value of 9998 in word 0 of the output image, the module will perform a warm-start. This involves clearing the configuration and all program status data. Finally, the program will load in the configuration information from the Flash ROM and begin running. There is no positive response to this message other than the status data being set to zero and the block polling starting over.

4.4 COLD-BOOT (BLOCK 9999)

This block does not contain any data. When the processor places a value of 9999 in word 0 of the output image, the module will perform a hardware restart. This will cause the module to reboot and reload the program. There is no positive response to this message other than the status data being set to zero and the block polling starting over.

5 Master Command List

The MVI94-MBM communication module's primary services are data concentration and communication gateway. The Modbus master port polls Modbus slave devices based on user defined commands and places the data in the virtual Modbus database. The Flex I/O bus interfaces with this database to a Flex processor. The user is responsible for defining the structure and content of the virtual Modbus database.

In order to interface the virtual Modbus database with Modbus slave devices, the user must construct a command list. The commands in the list specify the Modbus slave device to be utilized, the function to be performed (read or write), the data area in the device to interface with and the position in the virtual Modbus database to be associated with the device data. Up to 100 commands can be entered for this purpose. The list is processed from top (command #0) to bottom. A poll interval parameter is associated with each command to specify a minimum delay time in seconds between the issuance of a command. If the user specifies a value of 10 for the parameter, the command will be executed no more frequently than every 10 seconds. Additionally, a user specified time delay can be inserted between the issuance of each command. This is useful for slow responding slave devices.

Write commands have a special feature, as they can be set to execute only if the data in the write command changes. If the data in the command has not changed since the command was last issued, the command will not be executed. If the data in the command has changed since the command was last issued, the command will be executed. Use of this feature can lighten the load on the Modbus network. In order to implement this feature; set the enable code for the command to a value of 2.

6 MVI Module Configuration

This section of the documentation describes the configuration data required by the communication module. It is important that the module be configured accurately for reliable and correct operation. Appendix A of the documentation contains a configuration form that can be used to aid in configuring the module.

All configuration information for the module is stored in the module's Flash ROM. This provides permanent storage of the information. The user configures the module using a text file and then using the terminal emulation software provided with the module to download it to the module's Flash ROM. The file contains the configuration for the virtual Modbus database, Flex backplane data transfer, master port and the command list. This file is downloaded to the module for each application. An example file (MVI_MBM.TXT) is shipped with the module and should be used as a starting point for configuration. Use any text editor you are familiar with to edit the data in the file. When you have completed editing the file, download it to the module using the terminal emulation software provided by ProSoft Technology, Inc. A different file can be used for each application.

The text file is separated into two sections with topic header names enclosed in the [ ] characters. The sections present in the file are as follows:

[Section]	Description
[MODBUS MASTER]	Configuration of the module database, backplane data transfer and master port.
[COMMANDS]	Command list data for the Modbus master port.

After each section header, the file contains a set of parameters. Unique labels are used under each section to specify a parameter. Each label in the file must be entered exactly as shown in the file for the parameter to be identified by the program. If the module is not considering a parameter, check the label for the data item. Each parameter's value is separated from the label with the ':' character. This character is used by the program to delimit the position in the data record where to start reading data. All data for a parameter must be placed after the ':' character. For numeric parameter values, any text located after the value will not be used. There must be at least on space character between the end of the parameter value and the following text. An example of a parameter entry is given below:

Baud Rate : 38400 #Baud rate for port 110 to 115K

The parameter label is "Baud Rate" and the parameter value is 38400. The characters after the parameter value are ignored and are used for internal documentation of the configuration file.

The table below displays the baud rate values expected by the program. Because the baud rate parameter can potentially exceed the value for a word value, high baud rates must be shortened.

Baud Rate Values

Baud Rate	Parameter Value
110	110
150	150
300	300
600	600
1200	12 or 1200
2400	24 or 2400
4800	48 or 4800
9600	96 or 9600
19,200	19, 192 or 19200
28,800	28, 288 or 28800
57,600	57 or 576
115,200	115 or 1152

Any record that begins with the '#' character is considered to be a comment record. These records can be placed anywhere in the file as long as the '#' character is found in the first column of the line. These lines are ignored in the file and can be used to provide documentation within the configuration file. Liberal use of comments within the file can ease the use and interpretation of the data in the file.

A description of each section and its contained data items (parameters) is given below.

6.1 [MODBUS MASTER] Section

The [MODBUS MASTER] section of the configuration file is used to set the Modbus master port communication parameters, to define the virtual Modbus database, to define the command list specifications for the master port and to specify the Flex backplane data transfer parameters. The table below lists the parameters defined in this section:

[Section]/Item	Value	Range	Description
[MODBUS MASTER]			Configuration header for Master Modbus Port.
Module Name:		Up to 80 chars	Name of the module for use on reports. Use this parameter to identify your module in your system.
Maximum Register:		1 to 3996	This parameter defines the maximum register in the virtual Modbus Database. You should size the database for your application leaving room for expansion in the future. Requests for registers outside of the range selected will be returned with an error message.
Float Flag:		Y or N	This flag specifies if the floating-point data access functionality is to be implemented. 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.
Float Start:		0 to 32767	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 as floating-point data.

Float Offset:		0 to 3995	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.
Error/Status Block Pointer:		-1 to 3995	This value represents the relative starting position in the module's internal Modbus database where the Error/Status data will be stored. The table can be placed anywhere in the module's data space. The content of the Error/Status table is updated at the frequency defined in the parameter below. If a value of -1 is set for the parameter, the data will not be placed in the database.
Error/Status Frequency:		0 to 65535	This parameter specifies the number of program cycles between each update of the Error/Status Block data in the module. If the parameter is set to a value of 0, the data is never updated.
Protocol:		0 or 1	This parameter specifies the Modbus protocol to be used on the port. Valid Protocols are 0=Modbus RTU and 1=Modbus ASCII.
Baud Rate:		110 to 115K	This is the baud rate to be used on the port. Enter the baud rate as a value. For example, to select 19K baud, enter 19200. Valid entry for this field include: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600 and 115.
Parity:		0 to 4	This is the Parity code to be used for the port. The coded values are as follows: 0=None, 1=Odd, 2=Even, 3=Mark and 4=Space.
Data Bits:		5 to 8	This parameter sets the number of data bits for each word used by the protocol.
Stop Bits:		1 or 2	This parameter sets the number of stop bits to be used with each data value sent.
RTS On:		0 to 65535	This parameter sets the number of milliseconds to delay after RTS is asserted before the data will be transmitted.
RTS Off:		0 to 65535	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.
Use CTS Line:		Y or N	This parameter specifies if the CTS modem control line is to be used. If the parameter is set to N, the CTS line will not be monitored. If the parameter is set to Y, the CTS line will be monitored and must be high before the module will send data. This parameter is normally only required when half-duplex modems are used for communication (2-wire).
Command Count:		0 to 100	This parameter specifies the number of commands to be processed by the Modbus Master port.
Command Delay:		0 to 65535	This parameter specifies the number of milliseconds to wait between issuing each command. This delay value is not applied to retries.
Error Block Pointer:		-1 to 3995	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.
Response Timeout:		0 to 65535	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 upon the communication network used and the expected response time of the slowest device on the network.
Retry Count:		0 to 10	This parameter specifies the number of times a command will be retried if it fails.

Error Delay Count:		0 to 65535	This parameter specifies the number of polls to skip on the slave before trying to re-establish communications. After the slave fails to respond, the master will skip commands to be sent to the slave the number of times entered in this parameter.
Write Start Register:		0 to 3995	This parameter specifies the starting register in the internal Modbus database to write over the backplane.
Write Register Count:		12 to 3996	This parameter specifies the number registers in the internal Modbus database to write over the backplane. This parameter is used to compute the number of blocks to transfer from the module to the backplane. The number of blocks must be >= 2 for proper backplane data transfer.
Read Start Register:		0 to 3995	This parameter specifies the starting register in the internal Modbus database to fill with data read over the backplane.
Read Register Count:		12 to 3996	This parameter specifies the number registers in the internal Modbus database to consider from the read operations over the backplane. This parameter is used to compute the number of blocks to transfer from the backplane to the module. The number of blocks must be >= 2 for proper backplane data transfer.

6.2 [COMMANDS] Section

The [COMMANDS] section of the configuration file is used to set the Modbus master port command list. This list is used to poll Modbus slave devices attached to the Modbus master port. The module supports numerous commands. This permits the module to interface with a wide variety of Modbus protocol devices.

The command list is formatted differently than the other sections of the configuration file. Commands are present in a block between the labels START and END. These labels are used to inform the program where the list resides. The module's program will parse all commands after the START label until it reaches the END label or until the command count entered for the port is reached.

The function codes used for each command are those specified in the Modbus protocol. Each command list record has the same format. The first part of the record contains the information relating to the MVI94-MBM, communication module and the second part contains information required to interface to the Modbus slave device. The command structure is displayed in the table below for all functions supported:

Module Information ◄——┼——► Device Information

MODBUS COMMAND STRUCTURE

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

node = destination address

Each parameter is discussed below:

6.2.1 Enable Code

This field is used to define whether or not the command is to be executed and under what conditions. If the parameter is set to 0, the command is disabled and will not be executed in the normal polling sequence. The command can be executed under the control of the PLC processor through the use of a Command Control block. Setting the parameter to a value of 1 for the command causes the command to be executed each scan of the command list if the Poll Interval Time is set to zero. If the Poll Interval time is set, the command will be executed, when the interval timer expires. If the parameter is set to 2, the command will execute only if the internal data associated with the command changes. This value is valid only for write commands.

6.2.2 Internal Address

This field specifies the virtual Modbus database register to be associated with the command. If the command is a read function, the data read from the slave device will be placed starting at the register value entered in this field. If the command is a write function, the data written to the slave device will be sourced from the address specified. Register addresses specified for commands must reside in the range specified by the Maximum Register parameter under the [MODBUS MASTER] section.

6.2.3 Poll Interval Time

This parameter specifies the minimum interval to execute continuous commands (Enable code of 1). The parameter is entered in units of seconds. Therefore, if a value of 10 is entered for a command, the command will execute no more frequently than every 10 seconds.

6.2.4 Count

This parameter specifies the number of registers or digital points to be associated with the command. Functions 5 and 6 ignore this field as they only apply to a single data point. For functions 1, 2 and 15, this parameter sets the number of digital points

(inputs or coils) to be associated with the command. For functions 3, 4 and 16, this parameter sets the number of registers to be associated with the command.

6.2.5 Swap Code

This parameter is used to define if the data received from the Modbus slave is to be ordered differently than received from the slave device. This parameter is helpful when dealing with floating-point or other multi-register values, as there is no standard method of storage of these data types in slave devices. This parameter can be set to order the register data received in an order useful by other applications. The table below defines the values and their associated operations:

Code	Operation
0	None -- No change is made in the byte ordering.
1	Words -- The words are swapped.
2	Words & Bytes -- The words are swapped and then, the bytes in each word are swapped.
3	Bytes -- The bytes in each word are swapped.

6.2.6 Node

This parameter is used to specify the Modbus slave node address on the network to be considered. Values of 1 to 255 are permitted. Most Modbus devices only accept an address in the range of 1 to 247 so be careful. If the value is set to zero, the command will be a broadcast message on the network. The Modbus protocol permits broadcast commands for write operations. Do not use this node address for read operations.

6.2.7 Function Code

This parameter specifies the Modbus function to be executed by the command. These function codes are defined in the Modbus protocol. The table below defines the purpose of each function supported by the module.

MODBUS FUNCTION LIST

FUNC	DESCRIPTION
1	Read Coil Status
2	Read Input Status
3	Read Holding Registers
4	Read Input Registers
5	Single Coil Write
6	Single Register Write
15	Multiple Coil Write
16	Multiple Register Write

6.2.8 Device Modbus Address

This parameter specifies the starting Modbus register or digital point address to be considered by the command in the Modbus slave device. Refer to the documentation of each Modbus slave device on the network for their register and digital point

address assignments. For example, in an Omni Flow Computer, points 1901 to 1936 are digital points used for alarm flags. These points can be read using Modbus function code 1 and placed in the virtual Modbus database of the module.

7 Troubleshooting

The MVI module is designed to allow devices with a serial port to be accessed on the Flex I/O platform. The MVI94-GSC provides simple serial communications without requiring programming of the MVI module in 'C'. This document assumes that the MVI module has been configured to run the GSC application on startup (this is done at the factory).

7.1 LED Indicators

The MVI module has 5 LEDs that are visible from the front of the module. The following table details the function of these LEDs.

MVI94-GSC LED Descriptions

COLOR	LEGEND	DESCRIPTION
Green	PRT1	PRT1 Activity (Configuration Port) This LED is used to indicate data transmit and receive activity on the configuration port. When the TXD or RXD pin is active on the port, the LED will be illuminated green. When the port is not active, the LED will be in the off state.
Green	PRT2	PRT2 Activity(Application Port) This LED is used to indicate data transmit and receive activity on the Application Port (Modbus, DF1,etc.). When the TXD or RXD pin is active on the port, the LED will be illuminated green. When the port is not active, the LED will be in the off state.
Red/Green	STS	Module Status This LED is used to indicate the "health" of the module. When power is applied to the module, the LED will be illuminated. Green If the LED is green, the program is working correctly and the user configuration is being used. Red If the LED is red, the program is halted. Try restarting the module by cycling power. This should cause the module to return to its normal state.
Yellow	U1	Backplane Activity This LED is used to indicate backplane data transfer operation. When the module is successfully writing data to the FLEX I/O backplane, the LED will be in the off state. When the module is reading a new block of data from the FLEX I/O backplane, the LED will be in the on state (amber). During normal operation of the module, this LED should turn on and off at a very rapid rate. If the LED never turns on, check your ladder logic to be certain the data transfer is set up correctly.

COLOR	LEGEND	DESCRIPTION
Yellow	U2	Communication Error This LED is used to indicate communication errors on the Modbus master port. The LED is illuminated (amber) when no error exists on the port. If a communication error is recognized on the port, the LED will turn off. If the LED is turned off, check for errors in the command list to determine the error condition recognized by the module.

7.2 Error and Status Data from Module

The module error/status data areas are discussed in this section. The module contains three areas related to this data. The user defines the location of two of these data sets in the virtual Modbus database of the module. The error/status data contains module data, the command error list data set contains the errors associated with the command list and the slave status list contains the current communication status of each slave on the master port.

7.2.1 ERROR/STATUS DATA

The error/status data table is located at the virtual Modbus address assigned by the user. If the address is set to -1 or the frequency parameter is set to 0, the data will not be placed in the database. It will only be available through the Configuration/Debug Port. If valid address and frequency values are assigned, the module will update the Modbus data area.

The data area will be initialized with zeros whenever the processor is initialized. This occurs during a cold-start (power-on), reset (reset push-button pressed) or a warm-boot operation (commanded or loading of new configuration).

The data area is a 32-word register block. The structure of the block is shown below:

ERROR/STATUS TABLE

WORD	DESCRIPTION
	SYSTEM STATUS
0	Program Cycle Counter
1	Last read block identification code.
2	Current read block identification code.
3	Current write block identification code.
4	Current command index value.
5	Total number of commands to process by the module.
6	Total number of commands in the command queue.
7	Current state code for the Modbus master port.
8	Current communication state code for the Modbus master port.

SYSTEM INFORMATION	
9	Product Name (ASCII) = MBAB
10	
11	Revision (ASCII)
12	
13	Operating System Rev (ASCII)
14	
15	Production Run Number (ASCII)
16	

MASTER PORT STATISTICS	
17	Number of Command Requests
18	Number of Command Responses
19	Number of Command Errors
20	Number of Requests
21	Number of Responses
22	Number of Errors Received
23	Number of Errors Sent
24	Reserved
25	Reserved

BLOCK TRANSFER STATISTICS	
26	Total number of read block operations
27	Total number of write block operations
28	Total number of blocks parsed
29	Total number of event blocks received
30	Total number of command blocks received.
31	Total number block transfer errors.

7.2.2 COMMAND ERROR LIST

Each command in the command list has a reserved word value for a status/error code. This error data list can be read using the Configuration/Debug Port. Additionally, the data can be placed in the virtual Modbus database of the module. The configuration parameter "Command Error Pointer" is used to define the register address in the virtual Modbus database where the data will be placed.

The first word in the register location defined contains the status/error code for the first command in the port's command list. Each successive word in the command error list is associated with the next command in the list. Therefore, the size of the data area is dependent upon the number of commands defined.

Refer to the following Error Codes section to interpret the status/error codes present in the data area.

7.2.3 SLAVE STATUS LIST

The slave status list is used to view the communication status of each slave device on a master port. Slaves attached to the master port can have one of the following states:

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

Slaves are defined to the system when the module initializes the master command list. Each slave defined will be set to a state of 1 in this initial step. If the master port fails to communicate with a slave device (retry count expired on a command), the master will set the state of the slave to a value of 2 in the status table. This suspends communication with the slave device for a user specified scan count (**Error Delay Count** value in the configuration). 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.

The slave status list can only be viewed using the module's configuration/debug port. The 'O' option on the main menu is used to display the status of all 256 slave units.

7.3 ERROR CODES

The module error codes are listed in this section. Error codes are separated into Modbus error codes and module error codes.

7.3.1 MODBUS ERROR CODES

These error codes are returned to the module from Modbus slave devices attached to the master port. These codes are the standard Modbus errors. The error codes are listed in the table below:

MODBUS ERRORS

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

7.3.2 MODULE ERROR CODES

Module error codes are generated by the module's program. These errors can result due to configuration or communication problems. These errors are stored in the Modbus master, command list error table. A word is allocated for each command in the memory area.

The table below lists the errors returned by the module for communication errors:

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

The table below lists the errors returned by the module for errors found when parsing the command list:

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	All parameters set to 0
-47	All parameters set to -1

Use the error codes returned for each command in the list to determine the success or failure of the command. If the command fails, use the error code to determine the cause of failure.

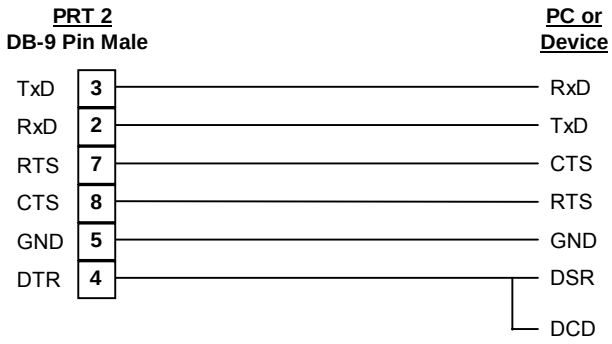
8 Serial Port Connections

This section contains information on cable and pin assignments for the MVI94-MBM, communication module. Port 1 (PRT 1, RS-232 interface) on the module is used for a Configuration/Debug port and requires no special cabling. A standard null-modem cable will work for the port. The only pins required on the connector are 2, 3, and 5 (RxD, TxD and GND).

Port 2 (PRT 2) is the Modbus master port. This port can be configured for RS-232, RS-422 and RS-485 communication interfaces. Before installing the module into the base, set the modules interface jumpers located on the bottom of the module. The diagrams below show the pin assignments for the three interfaces:

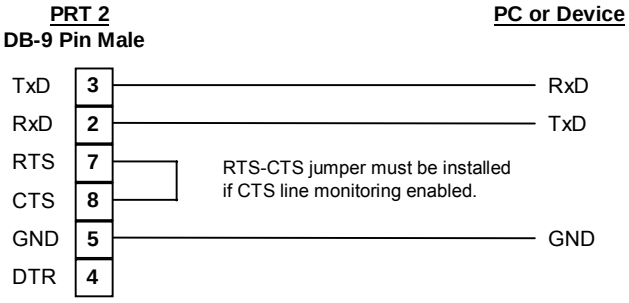
8.1 RS-232 -- Null Modem Connection (Hardware Handshaking)

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



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

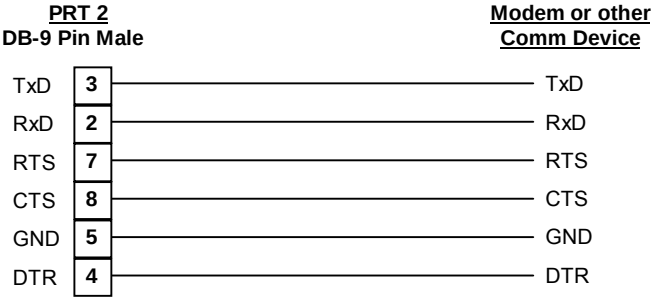
This type of connection can be used to connect the module to a computer or field device communication port.



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

8.3 RS-232 -- Modem Connection

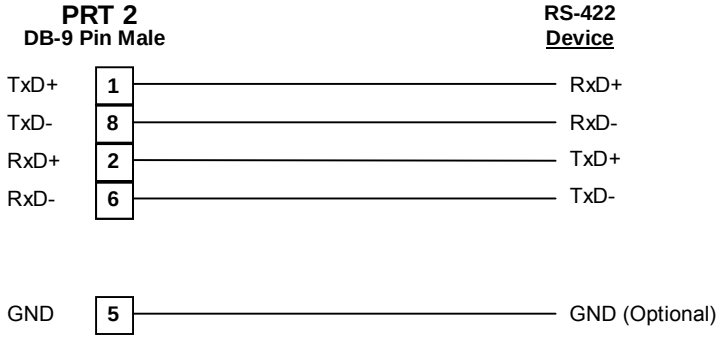
This type of connection is required between the module and a modem or other communication device.



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

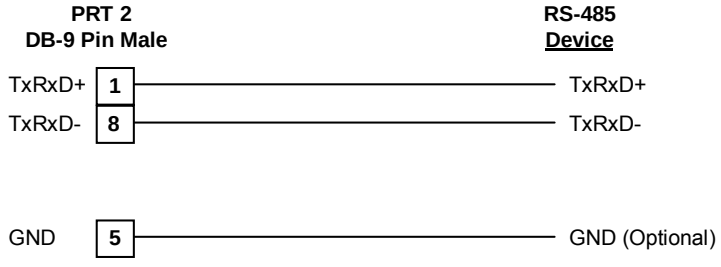
8.4 RS-422 INTERFACE CONNECTIONS

The diagram below applies when the RS-422 interface is selected.



8.5 RS-485 INTERFACE CONNECTIONS

The diagram below applies when the RS-485 interface is selected.



9 PLC Code Samples

Sample ladder logic for the MVI94-GSC module has been developed in order to ease the implementation in the field. This logic is available on our web/ftp site for the PLC 5, and is partially documented in the following section.

9.1 Ladder Program Files

Program File	Name	Functional Description
Main Sample logic		
The functionality performed by SBR2 can be moved anywhere in a customers ladder program, as long as there is a JSR call to the routine once per program scan.		
2	MAIN_SAMPL	This ladder program is where the application handles the setup and retrieval of data between the rx/tx buffers for each of the ports and makes the JSR call to SBR 20. The example program shows Port 1 being used as a Transmit only port (Sending out Alarm messages to a printer), and Port 2 as a receive only port. This functionality can easily be changed by the ladder programmer .
MVI94-GSC Driver Logic		
The following ladder program files can be moved anywhere in a customers application ladder program, but their functionality should not be changed without significant understanding of the results.		
20	MVI_MAIN	This routine is the main routine called by the application ladder program and generally serves as the primary interface with the MVI94-GSC. Functionality in this SBR includes timeout logic and initialization of the ports on the MVI94, calls to the buffer TX and RX handling routines, and the CIO instructions performing the actual data transfer to the MVI94
21	MVI Init	Initializes the necessary working registers when called
22	MVI Rcv	Handles building the receive buffer in the ladder program as data is received from the MVI94
23	MVI Xmit	Handles the movement of the transmit buffer to the MVI94 in the ladder program
24	MVI Send	Called whenever a string is ready to be sent out a port. Calls to the routine are conditioned based on several status conditions (See example ladder program 2)

9.2 Ladder Data Files

Data File	Name	Description
Demo Data Files		
B3	DEMO_ONLY	Used in Program file 2 as Alarm bits to trigger the movement of ASCII strings into the P1_TXBUF.

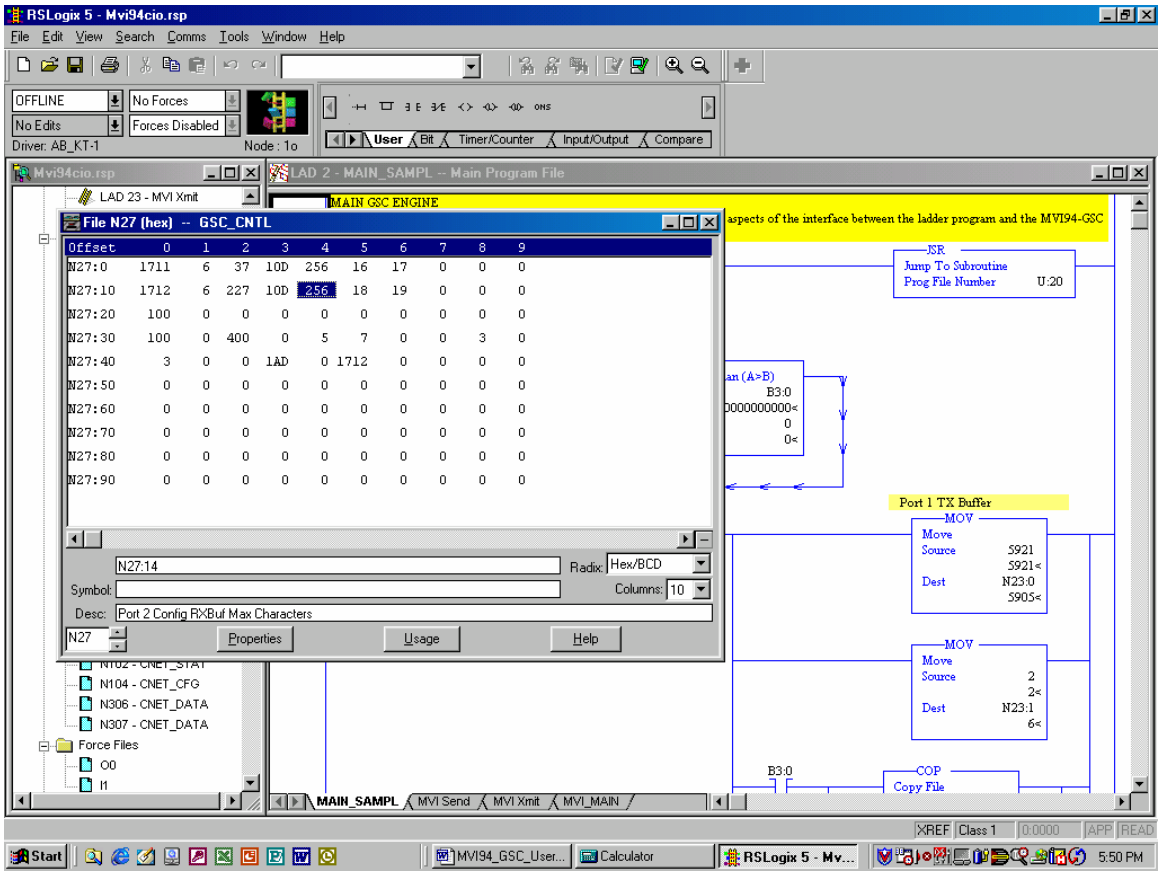
Data File	Name	Description
A9	DEMO_ONLY	Used in Program file 2 as a rollover buffer, receiving 20 words of data at a time from the receive buffer.
MVI94-GSC Driver Data Files		
N20	MVI INPUT	CIO input file. Use in a similar fashion if using BTR instructions
N21	MVI OUTPUT	CIO output file. Use in a similar fashion if using BTW instructions
N22	P1_RXBUF	Port 1 Receive buffer. File number easily changed during port setup in N27
N23	P1_TXBUF	Port 1 Transmit buffer. File number easily changed during port setup in N27
N24	P2_RXBUF	Port 2 Receive buffer. File number easily changed during port setup in N27
N25	P2_TXBUF	Port 2 Transmit buffer. File number easily changed during port setup in N27
N27	GSC_CNTL	MVI94-GSC module data file. The registers in this file are used to setup/store the port configurations for the GSC module, as well as the working registers for the ladder logic
T28	GSC_TMRS	Timeout timers for the ladder program
CT29	MVI_CNET	CIO instruction control data files
ControlNet Files		
N102	CNET_STAT	These data files are setup by the RSNetworks program, and are not used by the ladder program directly.
N103	CNET_CFG	
N306	CNET_DATA	
N307	CNET_DATA	

9.3 Port Configuration Setup

Configuration of the MVI94 ports is quite easy in the example program. Simply open N27 data file in the programming software, and using the data structure and layout of the configuration words detailed in Section 3.2, enter the appropriate values in N27:(2,3,4) for Port 1, and N27:(12,13,14) for Port 2.

Note that in addition to the Port Configuration values, the RX and TX buffer files can easily be changed by changing the file pointers in N27:5 and N27:6 for Port 1, and N27:15, N27:16 for Port 2. Note that if the file pointers are changed, the appropriate data files will need to created before the ladder program can be successfully run.

	Hex Data Representation					**Decimal Data Representation**	
	Message Signature	Msg Length (Bytes)	Baud Bits Parity	Handshake Termination	Data Timeout	RX File Buffer Pointer	TX File Buffer Pointer
Word	0	1	2	3	4	5	6
N27:0	1721h	0006h	0037h	010Dh	256h	22d	23d
N27:10	1722h	0006h	0037h	010Dh	256h	24d	25d



9.4 Initialization Values

The data registers in N27:20 through N27:29 are preset in our example and should never be changed by the user.

Important

Do not alter the values in N27:20 to N27:29. If creating a new data file, copy these values to the new data file. If this is not performed, the program will not interface correctly with the MVI94-GSC module.

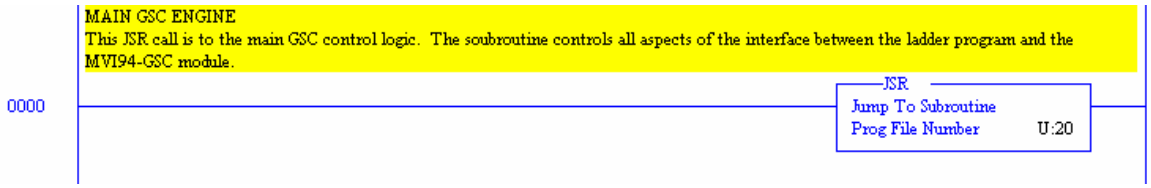
9.5 Using the Example Program

The key to successful implementation of the example ladder program is:

1) Understanding what to edit in File 2 to match the required application
2) Using the code in Program files 20 – 24 without alterations

Calling the MVI94-GSC Driver Logic

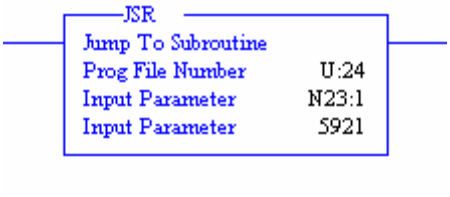
The SBR 20 should be called once per scan by the ladder program. In our example we have placed the call to SBR 20 in file 2.



Transmitting Data out Port 1

The transfer of data to be transmitted out a port on the MVI94 is coordinated by two status flags and a call to SBR 24.

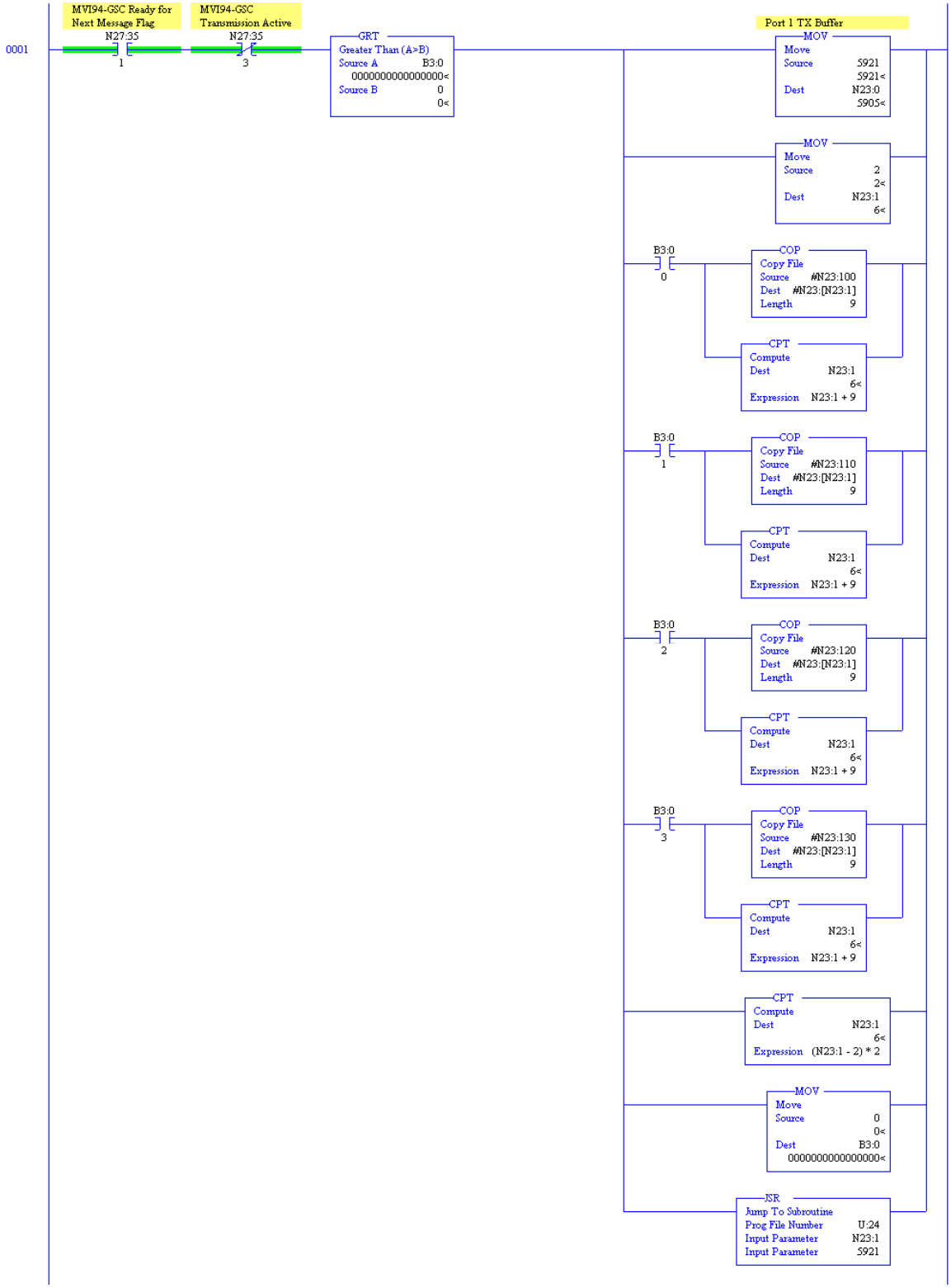
The following rung shows an example of some ladder logic which sets up the P1_TXBUF file and then makes a call to SBR 24. Note that the parameters passed to SBR24 are a function of the Port the data is to be transmitted out of.



The first Input Parameter is the length of the buffer to be transmitted, in bytes. This value is already setup in our example ladder program, therefore we pass the value in N23:1 (Message count field).

The second Input Parameter is the Message Signature for the port to be used.

Port 1	5921 dec (1721 Hex)
Port 2	5922 dec (1722 Hex)


MVI94-GSC Ready for Next Message Flag
N27:35
1
MVI94-GSC Transmission Active
N27:35
3
0001
GRT
Greater Than (A>B)
Source A B3:0
0000000000000000<
Source B 0
0<
Port 1 TX Buffer
MOV
Move
Source 5921
5921<
Dest N23:0
5905<
MOV
Move
Source 2
2<
Dest N23:1
6<
B3:0
0
COP
Copy File
Source #N23:100
Dest #N23:[N23:1]
Length 9
CPT
Compute
Dest N23:1
6<
Expression N23:1 + 9
B3:0
1
COP
Copy File
Source #N23:110
Dest #N23:[N23:1]
Length 9
CPT
Compute
Dest N23:1
6<
Expression N23:1 + 9
B3:0
2
COP
Copy File
Source #N23:120
Dest #N23:[N23:1]
Length 9
CPT
Compute
Dest N23:1
6<
Expression N23:1 + 9
B3:0
3
COP
Copy File
Source #N23:130
Dest #N23:[N23:1]
Length 9
CPT
Compute
Dest N23:1
6<
Expression N23:1 + 9
CPT
Compute
Dest N23:1
6<
Expression (N23:1 - 2) * 2
MOV
Move
Source 0
0<
Dest B3:0
0000000000000000<
JSR
Jump To Subroutine
Prog File Number U:24
Input Parameter N23:1
Input Parameter 5921

Receiving data on Port 2

The MVI94-GSC driver ladder logic fills data from the module into the appropriate Px_RXBUF file. Once a complete message has been received from the module, the Port RX Ready Flag is set.

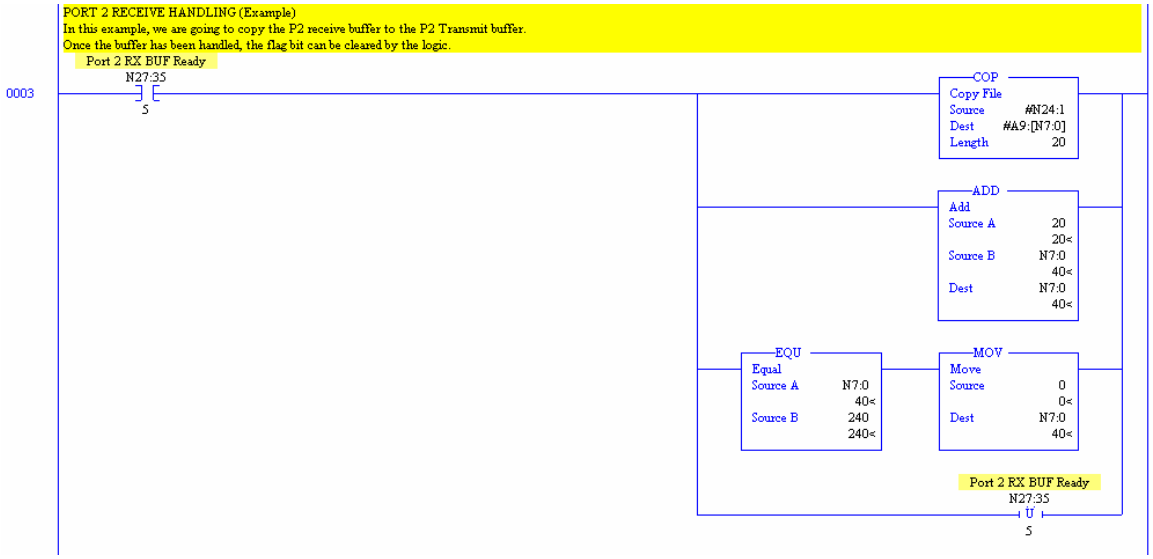
The following logic shows an example of how this flag can be trapped to execute some logic, after which the flag is cleared.

The example program does a simple building of a rollover buffer. The key points to get out of this example logic however is the conditional testing of the Port 2_RX BUF Ready flag (N27:35/5) and the clearing of this flag once the data has been handled.

The appropriate flags for testing for each port are:

Port 1 RX BUF ReadyN27:35/4

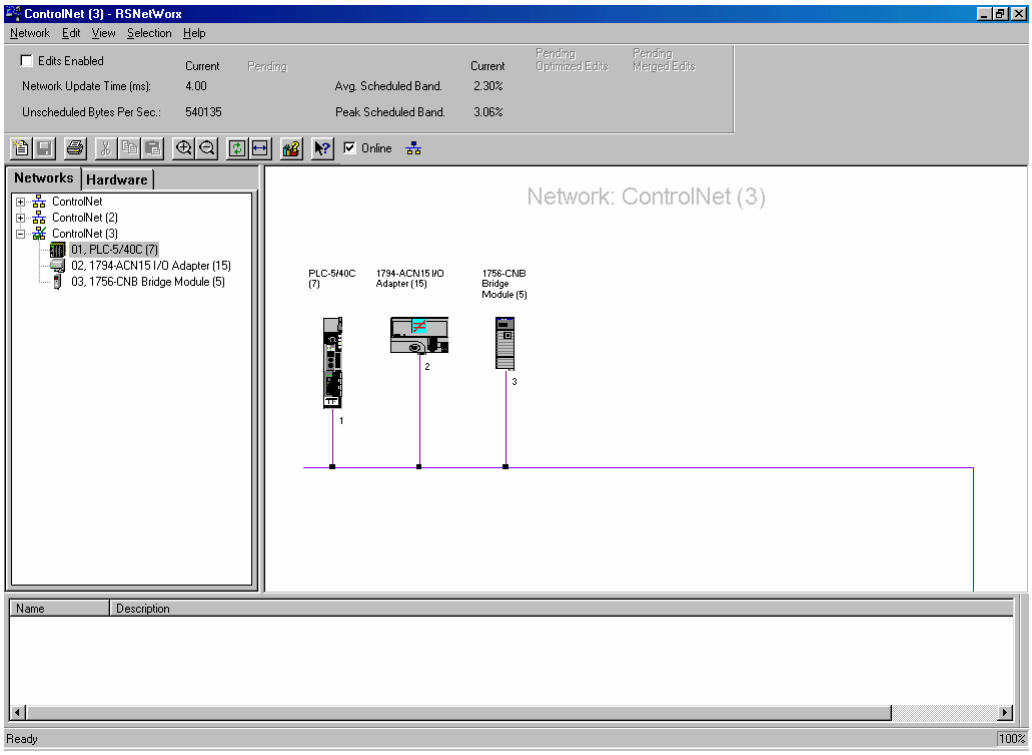
Port 2 RX BUF ReadyN27:35/5



Appendix A : PLC5 ControlNet Configuration

Adding the MVI94 to the ControlNet network

To access the MVI94 module and the 1794-ACRN15 ControlNet Flex I/O adapter, the PLC-5 must first be configured using the RSNetWorx tool. The following screens show the configuration selections which were used with the example program MVICIO.RSP when setting up the ControlNet network using RSNetWorx for ControlNet (Rev 1.80.38)



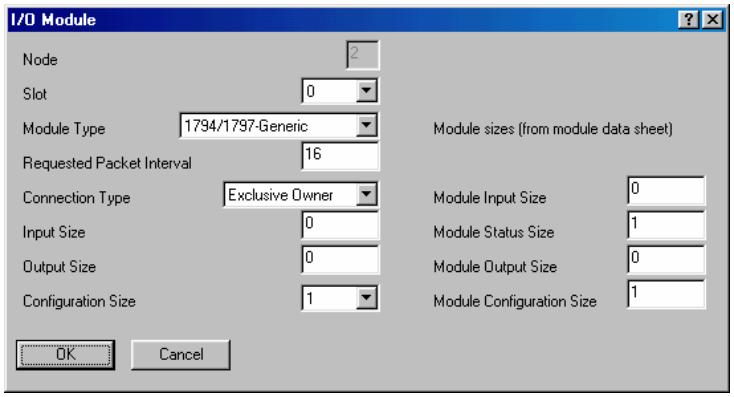
Notes from our experiences:

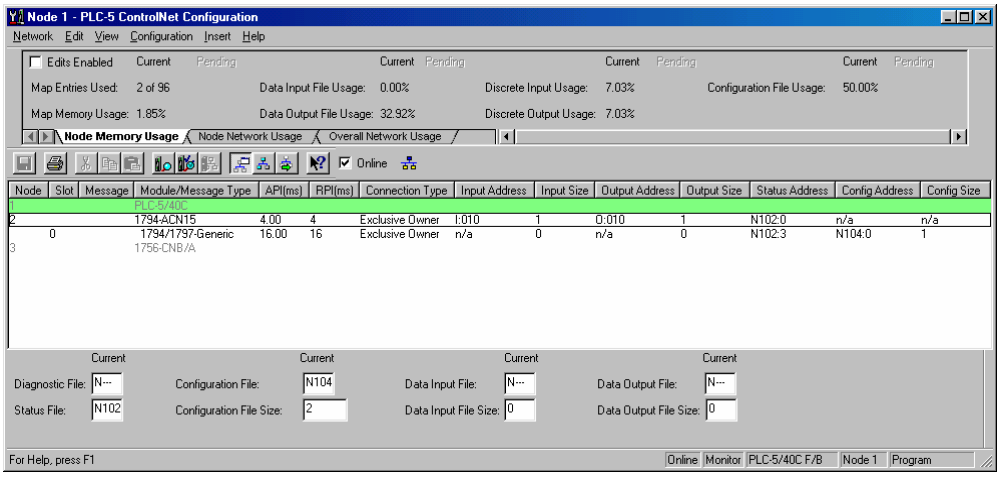
When setting up the MVI94 and the 1794-ACN15 adapter, we found a few configuration items to be somewhat critical to the success of our test setup. Of particular importance were:

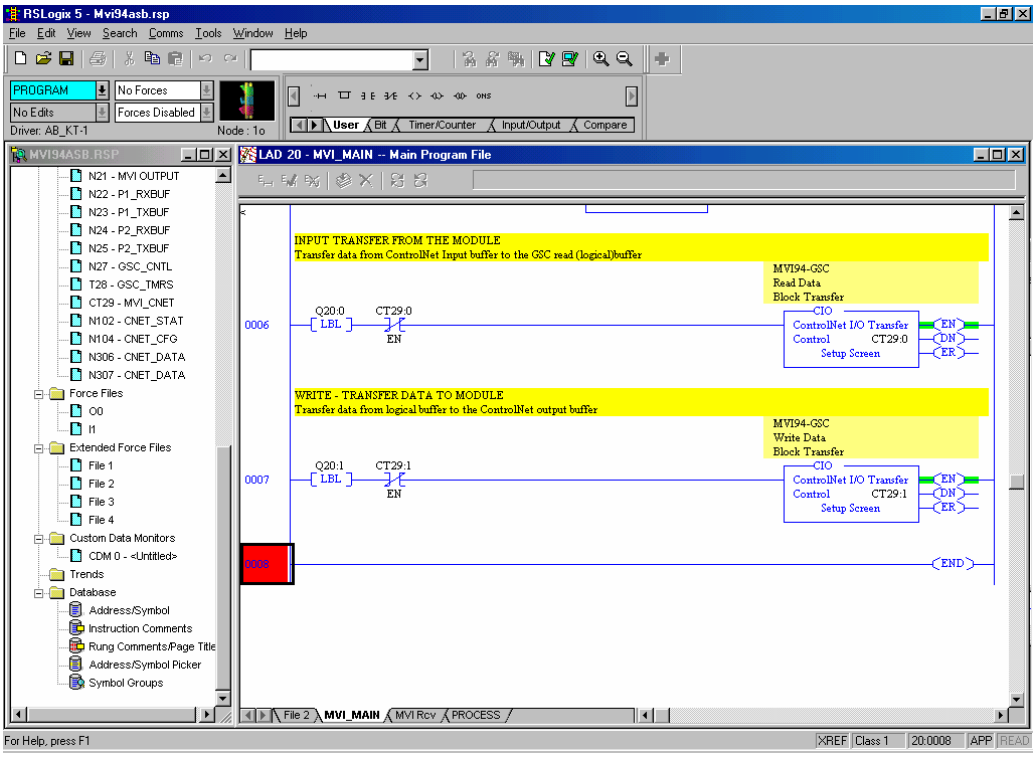
1794-CAN15 Input and Output File Size

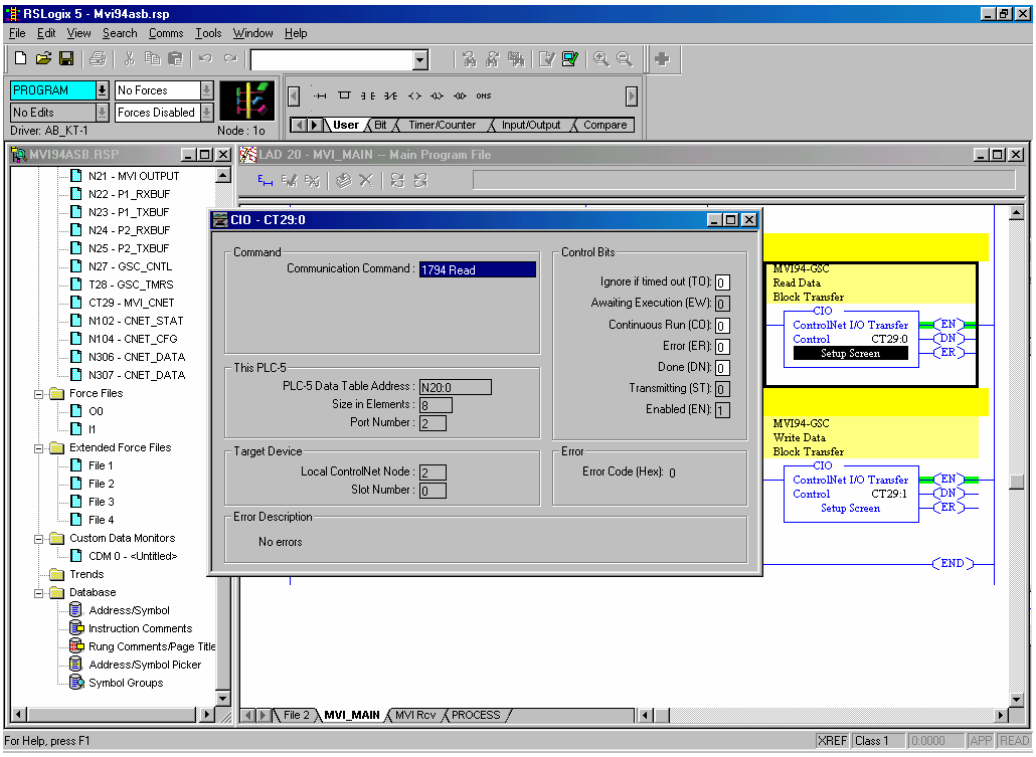
In the 1794-ACN15 adapter configuration, it was important to allocate one(1) input and one(1) output word per MVI94 to the adapter.

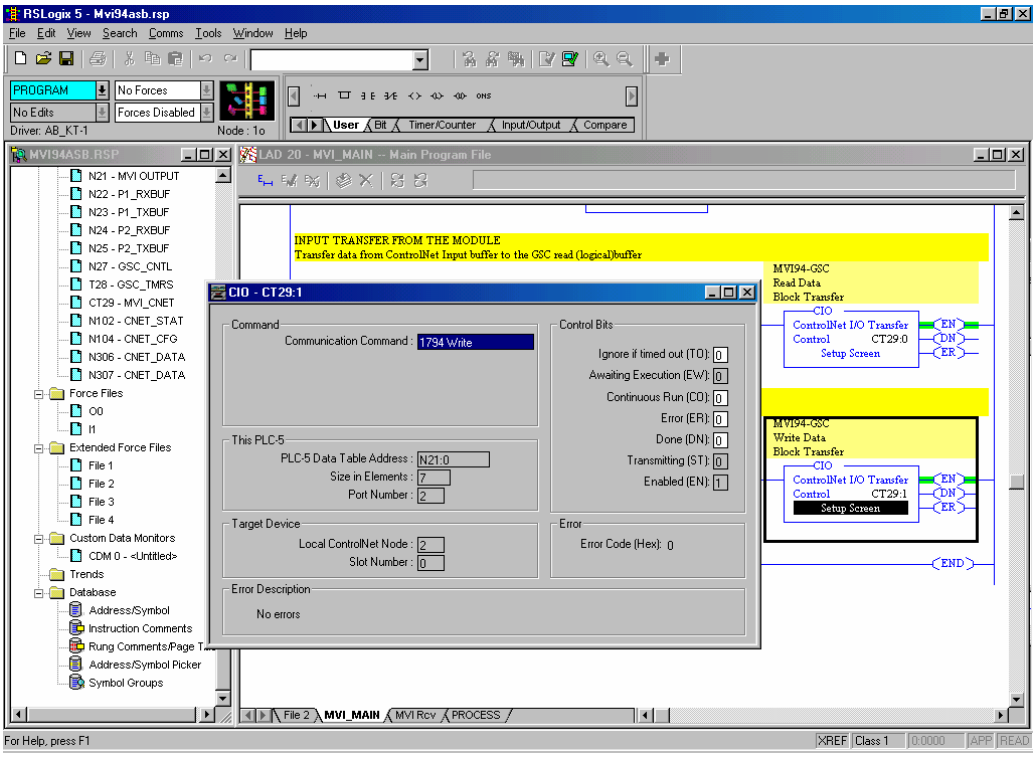
Note:
If the MVI94 does not show up in the RSNetWorx Module Type selection list, simply use the 1794/1797-Generic selection.


I/O Module
Node
2
Slot
0
Module Type
1794/1797-Generic
Module sizes (from module data sheet)
Requested Packet Interval
16
Connection Type
Exclusive Owner
Module Input Size
0
Input Size
0
Module Status Size
1
Output Size
0
Module Output Size
0
Configuration Size
1
Module Configuration Size
1
OK
Cancel


Node 1 - PLC-5 ControlNet Configuration
Network Edit View Configuration Insert Help
Edits Enabled
Current
Pending
Map Entries Used: 2 of 96
Map Memory Usage: 1.85%
Data Input File Usage: 0.00%
Data Output File Usage: 32.92%
Discrete Input Usage: 7.03%
Discrete Output Usage: 7.03%
Configuration File Usage: 50.00%
Node Memory Usage
Node Network Usage
Overall Network Usage
Online
Node Slot Message Module/Message Type API(ms) RPI(ms) Connection Type Input Address Input Size Output Address Output Size Status Address Config Address Config Size
1 PLC-5/40C
2 1794-ACN15 4.00 4 Exclusive Owner I:010 1 O:010 1 N102:0 n/a n/a
0 1794/1797-Generic 16.00 16 Exclusive Owner n/a 0 n/a 0 N102:3 N104:0 1
3 1756-CNB/A
Diagnostic File: N···
Status File: N102
Configuration File: N104
Configuration File Size: 2
Data Input File: N···
Data Input File Size: 0
Data Output File: N···
Data Output File Size: 0
For Help, press F1
Online Monitor PLC-5/40C F/B Node 1 Program


RSLogix 5 - Mvi94asb.rsp
File Edit View Search Comms Tools Window Help
PROGRAM
No Forces
No Edits
Forces Disabled
Driver: AB_KT-1
Node : 1o
User Bit Timer/Counter Input/Output Compare
MVI94ASB.RSP
LAD 20 - MVI_MAIN -- Main Program File
N21 - MVI OUTPUT
N22 - P1_RXBUF
N23 - P1_TXBUF
N24 - P2_RXBUF
N25 - P2_TXBUF
N27 - GSC_CNTL
T28 - GSC_TMRS
CT29 - MVI_CNET
N102 - CNET_STAT
N104 - CNET_CFG
N306 - CNET_DATA
N307 - CNET_DATA
Force Files
O0
I1
Extended Force Files
File 1
File 2
File 3
File 4
Custom Data Monitors
CDM 0 - <Untitled>
Trends
Database
Address/Symbol
Instruction Comments
Rung Comments/Page Title
Address/Symbol Picker
Symbol Groups
INPUT TRANSFER FROM THE MODULE
Transfer data from ControlNet Input buffer to the GSC read (logical)buffer
MVI94-GSC
Read Data
Block Transfer
0006
Q20:0
LBL
CT29:0
EN
CIO
ControlNet I/O Transfer
Control CT29:0
Setup Screen
EN
DN
ER
WRITE - TRANSFER DATA TO MODULE
Transfer data from logical buffer to the ControlNet output buffer
MVI94-GSC
Write Data
Block Transfer
0007
Q20:1
LBL
CT29:1
EN
CIO
ControlNet I/O Transfer
Control CT29:1
Setup Screen
0008
END
File 2 MVI_MAIN MVI Rcv PROCESS
For Help, press F1
XREF Class 1 20:0008 APP READ


RSLogix 5 - Mvi94asb.rsp
CIO - CT29:0
Command
Communication Command : 1794 Read
This PLC-5
PLC-5 Data Table Address : N20:0
Size in Elements : 8
Port Number : 2
Target Device
Local ControlNet Node : 2
Slot Number : 0
Control Bits
Ignore if timed out (TO): 0
Awaiting Execution (EW): 0
Continuous Run (CO): 0
Error (ER): 0
Done (DN): 0
Transmitting (ST): 0
Enabled (EN): 1
Error
Error Code (Hex): 0
Error Description
No errors
MVI94-GSC
Read Data
Block Transfer
CIO
ControlNet I/O Transfer
Control CT29:0
Setup Screen
MVI94-GSC
Write Data
Block Transfer
CIO
ControlNet I/O Transfer
Control CT29:1
Setup Screen
END


RSLogix 5 - Mvi94asb.rsp
File Edit View Search Comms Tools Window Help
PROGRAM
No Forces
No Edits
Forces Disabled
Driver: AB_KT-1
Node : 1o
User Bit Timer/Counter Input/Output Compare
MVI94ASB.RSP
N21 - MVI OUTPUT
N22 - P1_RXBUF
N23 - P1_TXBUF
N24 - P2_RXBUF
N25 - P2_TXBUF
N27 - GSC_CNTL
T28 - GSC_TMRS
CT29 - MVI_CNET
N102 - CNET_STAT
N104 - CNET_CFG
N306 - CNET_DATA
N307 - CNET_DATA
Force Files
O0
I1
Extended Force Files
File 1
File 2
File 3
File 4
Custom Data Monitors
CDM 0 - <Untitled>
Trends
Database
Address/Symbol
Instruction Comments
Rung Comments/Page T...
Address/Symbol Picker
Symbol Groups
LAD 20 - MVI_MAIN -- Main Program File
INPUT TRANSFER FROM THE MODULE
Transfer data from ControlNet Input buffer to the GSC read (logical)buffer
MVI94-GSC
Read Data
Block Transfer
CIO
ControlNet I/O Transfer
Control CT29:0
Setup Screen
EN
DN
ER
MVI94-GSC
Write Data
Block Transfer
CIO
ControlNet I/O Transfer
Control CT29:1
Setup Screen
EN
DN
ER
END
CIO - CT29:1
Command
Communication Command: 1794 Write
This PLC-5
PLC-5 Data Table Address: N21:0
Size in Elements: 7
Port Number: 2
Target Device
Local ControlNet Node: 2
Slot Number: 0
Control Bits
Ignore if timed out (TO): 0
Awaiting Execution (EW): 0
Continuous Run (CO): 0
Error (ER): 0
Done (DN): 0
Transmitting (ST): 0
Enabled (EN): 1
Error
Error Code (Hex): 0
Error Description
No errors
File 2 MVI_MAIN MVI Rcv PROCESS
For Help, press F1
XREF Class 1 0:0000 APP READ

Appendix B : Remote I/O with 1794-ASB Adapter

The same messaging example has also been implemented using a 1794-ASB Remote IO Flex I/O adapter instead of the ControlNet adapter. Only one of the program files must be modified to use the RIO adapter: Program File 20. In this file, rungs must be added to perform the block transfers necessary to copy the MVI module image data to the PLC data tables.

Figure 1 and Figure 2 show the RSLogix 5 IO chassis configuration dialogs for the 1794-ASB Flex I/O adapter. Chassis_2 in Figure 1 shows that communications channel 1B is connected to the 1794-ASB Flex adapter.

Figure 1 RSLogix 5 IO Configuration for 1794-ASB

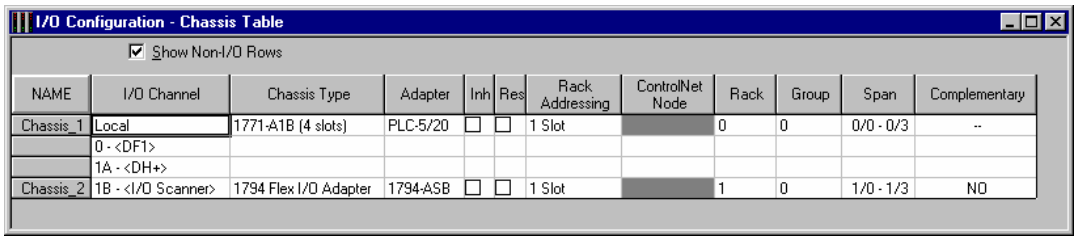
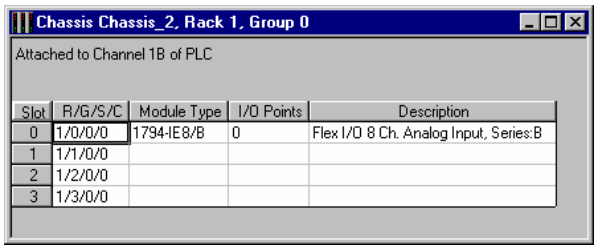


Figure 2 shows the configuration for Chassis_2. Note that older versions of RSLogix 5 will not have the MVI94-GSC module on the list of known modules; in this case, choose one of the 16-word analog modules such as the 1794-IE8, as shown. No other module configuration is necessary.

Figure 2 RSLogix 5 IO Chassis Configuration for 1794-ASB



Appendix C : Modbus Master Config Worksheet

This appendix contains configuration forms that will aid in the configuration of the module. If you design your system before trying to directly implement it, you will have a greater chance of success. Fill in the configuration form for your application, then edit the configuration text file.

MVI94-MBM MODBUS MASTER COMMUNICATION MODULE CONFIGURATION

[Section]/Item	Value	Range	Description
[MODBUS MASTER]			Configuration header for Master Modbus Port.
Module Name:		Up to 80 chars	Name of the module for use on reports. Use this parameter to identify your module in your system.
Maximum Register:		1 to 3996	This parameter defines the maximum register in the virtual Modbus Database. You should size the database for your application leaving room for expansion in the future. Requests for registers outside of the range selected will be returned with an error message.
Float Flag:		Y or N	This flag specifies if the floating-point data access functionality is to be implemented. 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.
Float Start:		0 to 32767	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 as floating-point data.
Float Offset:		0 to 3995	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.
Error/Status Block Pointer:		-1 to 3995	This value represents the relative starting position in the module's internal Modbus database where the Error/Status data will be stored. The table can be placed anywhere in the module's data space. The content of the Error/Status table is updated at the frequency defined in the parameter below. If a value of -1 is set for the parameter, the data will not be placed in the database.
Error/Status Frequency:		0 to 65535	This parameter specifies the number of program cycles between each update of the Error/Status Block data in the module. If the parameter is set to a value of 0, the data is never updated.

Protocol:		0 or 1	This parameter specifies the Modbus protocol to be used on the port. Valid Protocols are 0=Modbus RTU and 1=Modbus ASCII.
Baud Rate:		110 to 115K	This is the baud rate to be used on the port. Enter the baud rate as a value. For example, to select 19K baud, enter 19200. Valid entry for this field include: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600 and 115.
Parity:		0 to 4	This is the Parity code to be used for the port. The coded values are as follows: 0=None, 1=Odd, 2=Even, 3=Mark and 4=Space.
Data Bits:		5 to 8	This parameter sets the number of data bits for each word used by the protocol.
Stop Bits:		1 or 2	This parameter sets the number of stop bits to be used with each data value sent.
RTS On:		0 to 65535	This parameter sets the number of milliseconds to delay after RTS is asserted before the data will be transmitted.
RTS Off:		0 to 65535	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.
Use CTS Line:		Y or N	This parameter specifies if the CTS modem control line is to be used. If the parameter is set to N, the CTS line will not be monitored. If the parameter is set to Y, the CTS line will be monitored and must be high before the module will send data. This parameter is normally only required when half-duplex modems are used for communication (2-wire).
Command Count:		0 to 100	This parameter specifies the number of commands to be processed by the Modbus Master port.
Command Delay:		0 to 65535	This parameter specifies the number of milliseconds to wait between issuing each command. This delay value is not applied to retries.
Error Block Pointer:		-1 to 3995	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.
Response Timeout:		0 to 65535	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 upon the communication network used and the expected response time of the slowest device on the network.
Retry Count:		0 to 10	This parameter specifies the number of times a command will be retried if it fails.
Error Delay Count:		0 to 65535	This parameter specifies the number of polls to skip on the slave before trying to re-establish communications. After the slave fails to respond, the master will skip commands to be sent to the slave the number of times entered in this parameter.

Write Start Register:		0 to 3995	This parameter specifies the starting register in the internal Modbus database to write over the backplane.
Write Register Count:		12 to 3996	This parameter specifies the number registers in the internal Modbus database to write over the backplane. This parameter is used to compute the number of blocks to transfer from the module to the backplane. The number of blocks must be >= 2 for proper backplane data transfer.
Read Start Register:		0 to 3995	This parameter specifies the starting register in the internal Modbus database to fill with data read over the backplane.
Read Register Count:		12 to 3996	This parameter specifies the number registers in the internal Modbus database to consider from the read operations over the backplane. This parameter is used to compute the number of blocks to transfer from the backplane to the module. The number of blocks must be >= 2 for proper backplane data transfer.

Module Information ← | → Device Information

MODBUS COMMAND ENTRY FORM

Column #	1	2	3	4	5	6	7	8
	Enable Code	**Internal Address**	**Poll Interval Time**	**Count**	**Swap Code**	**Node**	**Function Code**	**Device Modbus Address**

Appendix D : MVI94MBM.CFG Example Configuration File

This appendix contains the contents of the example configuration file which is provided for use when setting up the MVI94-MBM. This file is downloadable off the ftp site (as referenced earlier in the manual).

```
# MVI94MBM.CFG
#
# Example file for use with the MVI94-MBM communication module
# with a Modbus Master port on COM2:.
#
# LOCATION : Example shipped with MVI94-MBM communication module
# DATE     : 07/03/2000
# MODIFIED :
#
#

[MODBUS MASTER]
MODULE NAME               : TEST OF MVI94-MBM MODULE (02/22/2000)
MAXIMUM REGISTER          :  4000  #Maximum # of register in virtual database
FLOAT FLAG                :     N  #Use floating-point data (Y=Yes, N=No)
FLOAT START               :  7000  #Start register of floats
FLOAT OFFSET              :  1500  #Virtual DB offset to floats
ERROR/STATUS BLOCK POINTER :   150  #Virtual DB offset to error/status
ERROR/STATUS FREQUENCY    :  1500  #Update freq. of error/status data

PROTOCOL                  :     0  #0=Modbus RTU, 1=Modbus ASCII
BAUD RATE                 :  9600  #Baud rate 110 to 115
PARITY                    :     0  #0=None, 1=Odd, 2=Even, 3=Mark, 4=Space
DATA BITS                 :     8  #5 to 8 data bits
STOP BITS                 :     1  #1 or 2 stop bits
RTS ON                    :     0  #millisecond RTS on delay (pre-data delay)
RTS OFF                   :     0  #millisecond RTS off delay (post-data delay)
USE CTS LINE              :     N  #Monitor status of CTS line (Y=Yes, N=No)

COMMAND COUNT             :     2  #Number of commands to process
COMMAND DELAY             :    30  #millisecond delay between commands
ERROR BLOCK POINTER       :  3000  #Virtual DB offset to error block
RESPONSE TIMEOUT          :  1000  #Message response timeout (milliseconds)
RETRY COUNT               :     3  #Message retry count
ERROR DELAY COUNT         :    10  #0-65535 delay for slave after comm error
```

```
WRITE START REGISTER       :     0  #Virtual DB offset where to start write
WRITE REGISTER COUNT       :    60  #Number of regs to write to backplane
READ START REGISTER        :    60  #Virtual DB offset where to start read
READ REGISTER COUNT        :    60  #Number of reg to read from backplane

[COMMANDS]
# TYPE   INTERNAL      POLL          SWAP   SLAVE  FUNCTION    DEVICE
# CODE    ADDRESS  INTERVAL  COUNT   CODE    NODE      CODE   ADDRESS
START
     1          0         0      10     0       1         3         0
     1          0         0      10     0       1        16        10
END
```

Appendix E : Configuration / Debug Port Operation

This section contains information on how to use the Configuration/Debug Port. This port provides the means to transmit or receive configuration data, view database information in the module and view configuration data. Use of this port can aid in locating problems that may exist in the user configuration, attached devices and the command list.

To communicate with the Configuration/Debug Port (PRT 1), a null-modem cable is required along with terminal emulation software. You should use the software provided by ProSoft, as it is required for transferring the configuration file. Connect the null-modem cable to the Configuration/Debug Port and start the software.

Communication parameters used for the port are as follows:

19,200 baud, no parity, 8 data bits, 1 stop bit and no hardware handshaking.

After the terminal emulation software is loaded, press the '**?**' key to display the menu.

If the main menu is not displayed, press the '**M**' key, and then press the '**?**' key. If the menu is not displayed, there may be a problem with the cable connection between the module and the terminal emulator. Check all connections and the communication parameters used by the terminal emulator.

Each option available on the Configuration/Debug Port is discussed below:

? = DISPLAY MENU

This option is used to display the menu options available in the current menu mode. If the menu mode is set to main, the following will be displayed.

```
MVI94-MBM -- MODBUS COMMUNICATION MODULE MENU
  ?=Display Menu
  A=Data Analyzer
  B=Backplane Transfer Statistics
  C=Modbus Configuration
  D=Modbus Database View
  E=Command Errs
  L=Command List
  O=Slave Status List
  R=Receive Configuration from Remote
  S=Send Configuration to Remote
  V=Version Information
  W=Warm Boot Module
  1=Modbus Port Status   6=Modbus Port Cfg
  Esc=Cold Boot Module
```

Select any of the options displayed by pressing the single key shown in the menu on the computer running the terminal emulation software.

A = Data Analyzer

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

```
DATA ANALYZER VIEW MENU
 ?=Display Menu
 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 = MODBUS MASTER PORT, Format=HEX, Tick=10
```

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

5 = 1 mSec Ticks

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

6 = 5 mSec Ticks

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

7 = 10 mSec Ticks

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

8 = 50 mSec Ticks

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

9 = 100 mSec Ticks

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

0 = No mSec Ticks

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

H = Hex Format

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

A = ASCII Format

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

B = Start Data Analyzer

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

```
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][58][37]_TT_<R+><01><03><A4><10><00><78><67><1D><R->
_TT__TT_[01][03][F0][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00]_TT_[00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][02][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][04][A2]<R+><01><03><80><E8><00><20><ED><E6><R->_TT__TT_
[01][03][96][00][01][00][88][00][00][00][00]_TT_[00][00][00][00][00][00][00][4E]
[00][58][00][B8][00][E8][00][64][00][00][00][09][00][00][00][03][00][00][00][00]
[00][00][00][4E][00][0A]_TT_[00][59][00][00][00][01][00][01][00][08][00][00][00]
[03][00][00][00][1E][00][05][00][4E][00][1E][00][04][00][32][00][4C][00][E8][00]
[03][00][64][00][59][00][01][00][01][00][01][00][07][00][00][00][03][00][00][00]
[01][00][01][00][4E][00][1E][00][03][00][00][00][B0][00][64][00][03][00][B8][00]
[59][00][01][00][00][00][00][00][08][00][00][00][03][00][00][00][05][00][05][00]
[4E][00][00][00][01][00][00][00][14][00][64][00][03][00][A0][A3][F1]_TT_<R+><01>
<03><84><D0><00><78><6C><E1><R->_TT_
```

Special characters used in the display are as follows:

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

S = Stop Data Analyzer

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

WARNING

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

M = Main Menu

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

B = BACKPLANE TRANSFER STATISTICS

This menu option is used to display the statistics for the backplane data transfer operation. Data displayed after selecting the option is shown below:

```
BACKPLANE TRANSFER STATISTICS:

  Total Read Blocks    :   30713      Read Register Start  :     100
  Total Write Blocks   :    2074      Read Register Count  :      60
  Total Parsed Blocks  :       0      Read Block Count     :      10
  Total Event Blocks   :       0      Write Register Start :       0
  Total Command Blocks :       0      Write Register Count :     100
  Total Block Errors   :    2074      Write Block Count    :      17
```

C = MODBUS CONFIGURATION

This menu option is used to display the module name and the configuration of the internal Modbus database. Data displayed after selecting the option is shown below:

```
MODBUS CONFIGURATION:

TEST OF MVI94-MBM MODULE

  Maximum Register     : 3996    Floating-point Flag  : N
  Floating-point Start : 7000    Floating-point Offset: 1500
  Err/Stat Blk Pointer : 3000    Err/Stat Blk Freq    : 1000
  Protocol Selected    : 0 (Modbus RTU)
```

D = MODBUS DATABASE VIEW

After selecting the option, the following text will be displayed: **Modbus DB Menu Selected**. This indicates that the database menu mode is selected. Options available in this mode are displayed by pressing the '**?**' key on the terminal emulator. If the key is pressed, the following will be displayed:

```
MODBUS DATABASE VIEW MENU
 ?=Display Menu
 0-3=Register Pages 0-3000
 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
```

Select an option by pressing the associated key. To display the current database page selected press the '**S**' key. A display similar to the one shown below will appear:

```
MODBUS DATABASE DISPLAY 3000 TO 3099 (DECIMAL)

  -3188      0      4     16      0      7      0      3      0  16973
   8269  11825  13872  12592  12336  12339  12592  13136  13104     93
  13198  13104      0      0      0      0 -16224   2383      0      0
      0   2383      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
```

The page of data displayed can be changed by using the '**-**', '**+**', '**N**' or '**P**' keys. The format of the data can be changed using the '**D**', '**H**', '**F**' and '**A**' keys. To return to the main menu mode, press the '**M**' key.

E = COMMAND ERRS

After selecting the option, the following text will be displayed: **Cmd Err Menu Selected**. The message indicates that the command error list menu mode is selected. Options available in this mode are displayed by pressing the '**?**' key on the terminal emulator. If the key is pressed, the following will be displayed:

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

Select an option by pressing the associated key. To display the current error list page selected, press the '**S**' key. A display similar to the one shown below will appear:

```
COMMAND ERROR LIST -- COMMANDS 0 TO 19 (DECIMAL)

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

The page of data displayed can be changed by using the '**-**', '**+**', '**N**' or '**P**' keys. The format of the data can be changed using the '**D**' and '**H**' keys. To return to the main menu mode, press the '**M**' key.

L = COMMAND LIST

After selecting the option, the following text will be displayed: **Command List Menu Selected**. This indicates that the command list menu mode is selected. Options available in this mode are displayed by pressing the '**?**' key on the terminal emulator. If the key is pressed, the following will be displayed:

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

Select an option by pressing the associated key. To display the current list page selected, press the '**S**' key. A display similar to the one shown below will appear:

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

EN MBREG POLLINT COUNT SWAP NODE FUNC ADDRS LASTERR
 1     0       0    32    0    1    3 33000  0X0000
 1   500       0   120    0    1    3 34000  0X0000
 1   700       0   120    0    1    3 34100  0X0000
 1   900       0   120    0    1    3 34200  0X0000
 1  1100       0   120    0    1    3 40000  0X0000
 1  1300       0   120    0    1    3 41000  0X0000
 1  1500       0   120    0    1    3 42000  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
 0     0       0     0    0    0    0     0  0X0000
```

The page of data displayed can be changed by using the '**-**', '**+**', '**N**' or '**P**' keys. The format of the data can be changed using the '**D**' and '**H**' keys. To return to the main menu mode, press the '**M**' key.

O = SLAVE STATUS LIST

This menu option is used to display the slave status list for the slaves on the master port. Each value displayed is associated with a slave node address. The first value displayed is associated with node address 0 and the last value is associated with node address 255. After selecting the option, the following display will be shown:

```
SLAVE STATUS LIST
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
```

Values in the list have the following definitions: 0 = inactive, 1 = active and 2 = in communication failure. The master port will only poll slaves with a status value of 1.

R = RECEIVE CONFIGURATION FROM REMOTE

This menu option is used to transfer the module configuration from the PC used for configuration to the module. This transfer requires the terminal software provided by ProSoft. After selecting the option, the following message will appear on the terminal display:

```
Press 'Y' key to confirm configuration receive!
Receiving configuration:


TRANSFERRING CONFIGURATION FROM REMOTE TO UNIT:
START MVIMB PROGRAM ON COMPUTER AND SELECT SEND CONFIGURATION OPTION
BY PRESSING THE [F3] KEY...
```

On the terminal emulator, press the **[F3]** key and enter the file name containing the configuration for the module when prompted for the filename. If the file is found, it will immediately be downloaded to the module. After the download operation is complete, the module will restart to read the new configuration.

S = SEND CONFIGURATION TO REMOTE

This menu option is used to transfer the module configuration from the module to the PC using the terminal software. This transfer requires the terminal software provided by ProSoft. After selecting the option and confirming the request, the following message will be displayed on the terminal:

```
Press 'Y' key to confirm configuration send!
Sending configuration:


TRANSFERRING CONFIGURATION FILES FROM UNIT TO REMOTE:
START MVIMB PROGRAM ON COMPUTER AND SELECT RECEIVE CONFIGURATION OPTION
BY PRESSING THE [F4] KEY....
```

On the terminal emulator, press the **[F4]** key and enter a file name when prompted. If the configuration is successfully transferred from the module, a file with the name

entered will be created on the PC. The file generated will be formatted as a standard configuration text file for the module.

V = VERSION INFORMATION

This menu option is used to display the version information for the software running on the module. This option should be used before calling ProSoft for support. Record the information and have it handy when calling for technical support. After selecting the option, the following information will be displayed:

```
VERSION INFORMATION:

  MVI94-MBM -- MODBUS COMMUNICATION MODULE
  (c) 1999, ProSoft Technology, Inc.

  PRODUCT NAME CODE          : MBM
  SOFTWARE REVISION LEVEL    : 1.06
  OPERATING SYSTEM REVISION  : 0100
  RUN NUMBER                 : 3001

  API LIBRARY VERSION        : 01.04
  BP DRIVER VERSION          : 01.04

  VENDOR ID                  : 22
  DEVICE TYPE                : 0
  PRODUCT CODE               : 2
  HARDWARE VERSION (DATE)    : 01.03 (09/03/1999)
  SERIAL NUMBER              : 816199927
  PRODUCT NAME               : 1794AV-MVI-DOS
```

W = WARM BOOT MODULE

This menu option is used to force the module software to read the current configuration information stored in the module's Flash ROM. The module will automatically perform this operation each time a new configuration is downloaded to the module. Therefore, this option is rarely used.

One possible use of this option is to reset all the program statistics. After the option is selected and the configuration is complete, all program counter values are reset to zero. This can be useful when monitoring a port's statistics.

1 = MODBUS PORT STATUS

This menu option is used to display the communication statistics for the Modbus master port. After selecting the option, the following will be displayed:

```
MODBUS MASTER STATUS:
  Retries : 0       Cur Cmd : 2       State   : 3
  ComState: 0

  Number of Command Requests: 18969
  Number of Cmd Responses   : 18937
  Number of Command Errors  : 94
  Number of Requests        : 19032
  Number of Responses       : 18937
  Number of Errors Received : 0
  Number of Errors Sent     : 0

  Program Scan Counter      : 31924
```

Use this information to aid in debugging Modbus slave port communication problems.

6 = MODBUS PORT CFG

This menu option is used to display the Modbus master port configuration information. After selecting the option, the following will be displayed:

```
CONFIGURATION OF MODBUS MASTER:

  Maximum Registers  : 3996

  Floating-point Data:
    Flag    : N        Start: 7000     Offset: 1500

  Error/Status Table:
    Offset  : 3000     Freq : 1000

  Communication Parameters:
    Protocol: 0 (Modbus RTU)
    Baud    : 38400    Parity : NONE     Databits    : 8    Stopbits: 1
    RTS On  : 0        RTS Off: 0        Use CTS Line: N

  Command Parameters:
    Commands: 7        Min Dly: 0        Cmd Offs : -1
    Resp TMO: 1000     Retries: 2        Err Delay: 20
```

The information displayed reflects that stored in the module's Flash ROM. This information should be reviewed each time a new configuration is downloaded to the module to be certain the configuration is correct.

ESC = COLD BOOT MODULE

This menu option is used to force the module to perform a cold boot operation. This will cause the module to restart and force all drivers to be loaded. The module will use the configuration stored in the module's Flash ROM to configure the module.

Support, Service and Warranty

Technical Support

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

Factory/Technical Support

1675 Chester Avenue, 2nd Floor

Bakersfield, CA 93301

(661) 716-5100

(661) 716-5101 (Fax)

E-mail address: prosoft@prosoft-technology.com

Web Site : http://www.prosoft-technology.com

Before calling for support, please prepare yourself for the call. In order to provide the best and quickest support possible, we will most likely ask for the following information (you may wish to fax it to us prior to calling):

1. Product Version Number
2. Configuration Information
    - Communication Configuration
    - Jumper positions
3. System hierarchy
4. Physical connection information
    - RS-232, 422 or 485
    - Cable configuration
5. Ladder Logic program
6. Module Operation
    - Block Transfers operation
    - LED patterns

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

Module Service and Repair

The MVI94-GSC is an electronic product, designed and manufactured to function under somewhat adverse conditions. As with any product, through age, misapplication, or any one of many possible problems, the card may require repair.

When purchased from ProSoft Technology, the module has a one year parts and labor warranty according to the limits specified in the warranty. Replacement and/or returns should be directed to the distributor from whom the product was purchased. If you need to return the card for repair, it is first necessary to obtain an RMA number from ProSoft Technology. Please call the factory for this number and display the number prominently on the outside of the shipping carton used to return the card.

General Warranty Policy

ProSoft Technology, Inc. (Hereinafter referred to as ProSoft) warrants that the Product shall conform to and perform in accordance with published technical specifications and the accompanying written materials, and shall be free of defects in materials and workmanship, for the period of time herein indicated, such warranty period commencing upon receipt of the Product.

This warranty is limited to the repair and/or replacement, at ProSoft's election, of defective or non-conforming Product, and ProSoft shall not be responsible for the failure of the Product to perform specified functions, or any other non-conformance caused by or attributable to: (a) any misapplication of misuse of the Product; (b) failure of Customer to adhere to any of ProSoft's specifications or instructions; (c) neglect of, abuse of, or accident to, the Product; or (d) any associated or complementary equipment or software not furnished by ProSoft.

Limited warranty service may be obtained by delivering the Product to ProSoft and providing proof of purchase or receipt date. Customer agrees to insure the Product or assume the risk of loss or damage in transit, to prepay shipping charges to ProSoft, and to use the original shipping container or equivalent. Contact ProSoft Customer Service for further information.

Limitation of Liability

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

Where directed by State Law, some of the above exclusions or limitations may not be applicable in some states. This warranty provides specific legal rights; other rights that vary from state to state may also exist. This warranty shall not be applicable to the extent that any provisions of this warranty is prohibited by any Federal, State or Municipal Law that cannot be preempted.

Hardware Product Warranty Details

Warranty Period : ProSoft warranties hardware product for a period of one (1) year.

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