MOORE®

CONFIGURATION GUIDE

CG39FDI-2
Rev: 2
May 1997

APACS®
QUADLOG®

MODBUS MASTER FUNCTION BLOCK LIBRARY FOR APACS VERSION 4.00 OR HIGHER QUADLOG VERSION 3.20 OR HIGHER

MOORE PRODUCTS CO., Spring House, PA 19477-0900
An ISO 9001 registered company.

TABLE OF CONTENTS

SECTION AND TITLE **PAGE**

LIST OF FIGURES

FIGURE AND TITLE **PAGE**

LIST OF TABLES

TABLE AND TITLE **PAGE**

SIGNIFICANT CHANGES FOR REV. 2

Significant changes for rev. 2 are indicated by change bars located in the page margins. Major changes are listed here.

1.0 INTRODUCTION - This section has been updated to accommodate the QUADLOG version of the library. Section descriptions were updated to include the new appendices as outlined on the next page.

1.2.2 PRODUCT SUPPORT - This section has been enhanced with the addition of a telephone listing of international subsidiaries.

2.0 SOFTWARE INSTALLATION - This section has been rewritten to accommodate the different versions of MS-Windows (i.e. Windows 3.x, Windows 95, and Windows NT).

SIGNIFICANT CHANGES FOR REV. 2 (Continued)

4.0 FUNCTION BLOCKS - This section has updated graphics for the Modbus Master (MODBUS_M) and Serial (SERIAL) function blocks. Also, three new soft list parameters (ModiconLoopback, PrioritizeWrites, and ExtendedAddress) have been added to the MODBUS_M block.

A.0 APPENDIX A, APPLICATION EXAMPLE 1 - This section has been revised with updates to graphics which show examples of on-line information.

B.0 APPENDIX B, APPLICATION EXAMPLE 2 - This section has been revised with updates to graphics which show examples of on-line information.

C.0 APPENDIX C, ERROR CODES - This section has been expanded. It now lists the error code messages that appear at the error code (E_CODE) output of the MODBUS_M function block, a description of each code, and recommended user actions.

D.0 APPENDIX D, SPECIFICATIONS - This section has been updated. It lists the general specifications of the Modbus Master Function Block Library.

F.0 APPENDIX F, ENTRELEC RS232/485 CONVERTER CONNECTIONS - This is a new section that shows the connections needed for wiring an RS-485 foreign device to an ACM/CCM's RS-232 serial port using an Entrelec RS-232 to RS-485 converter.

G.0 APPENDIX G, REDUNDANT SYSTEM WIRING CONFIGURATION - This is a new section that shows the general wiring needed for a dual Modbus network in a redundant system.

1.0 INTRODUCTION

The Modbus Master Function Block Library is add-on software for use with the *4-mation*™ configuration software, version 3.00 or higher. The library software is available in both APACS® and QUADLOG® versions. The APACS version (P/N 15939-623V4.xx) is for use with the Advanced Control Module (ACM) software, version 4.01 or higher. The QUADLOG version (P/N 15939-681V3.2x) is for use with either the Advanced Control Module Plus (ACM+) software or the Critical Control Module (CCM) software. Each must be version 3.20 or higher. *

The library software provides *4-mation* with a set of function blocks for configuring a control module (ACM/CCM) to operate as a Modbus master.

This document describes how to identify, register, backup, and install the Modbus Master Function Block Library. It also provides reference information for each function block. This information is intended to be used in conjunction with *Using the 4-mation Configuration Software*, as well as the other documents listed in section 1.3, Related Literature. After the Modbus Master Function Block Library software has been installed, most of the information presented in this document is also available through *4-mation*'s on-line help system.

This Guide is organized into the following sections:

- Section 1, Introduction—Summarizes the information presented in this Guide, describes the software package, provides the software registration procedure, and lists reference literature to consult for additional information.

- Section 2, Software Installation—Describes the general disk handling and backup requirements and the software installation procedure.

- Section 3, Configuration—Gives a general configuration objective, refers to the needed configuration software and associated literature, indicates how the Function Block Library is accessed, and provides guidelines to carry out the MODBUS_M function block configuration.

- Section 4, Function Blocks—Describes the inputs, outputs, and soft list parameters of the MODBUS_M and SERIAL function blocks.

- Appendix A, Application Example 1—Describes a control module configured as a Modbus master that reads thirty floating-point (REAL) values from a slave device.

- Appendix B, Application Example 2—Describes a control module configured as a Modbus master that reads 100 unsigned integer values from a slave address and converts these to REAL values for use in the control module.

- Appendix C, Error Codes—Lists the error code messages that appear at the error code (E_CODE) output of the MODBUS_M function block whenever an error condition is encountered.

- Appendix D, Specifications—Lists general specifications for the Modbus Master Function Block (FB) Library.

- Appendix E, Cable Connections—Shows the pin-out of the ACM/CCM serial ports and the minimum cable requirements for connecting to the serial port of a foreign device.

* • Appendix F, Converter Connections—Shows the connections needed for wiring an RS-485 foreign
* device to the ACM/CCM's RS-232 serial port using an Entrelec RS-232 to RS-485 converter.
*
* • Appendix G, Redundant System Wiring Configuration—Shows the general wiring needed for a dual
* Modbus network in a redundant system.

1.1 PRODUCT DESCRIPTION

The Modbus Function Block (FB) Libraries permit the control module and the APACS/QUADLOG system to exchange data with foreign devices that use the Modbus RTU protocol. The Modbus RTU protocol is used by AEG Schneider Automation (formerly Modicon/Gould) RTU programmable logic controllers (PLCs) and by many types of specialty instruments and interfaces such as analyzers, chromatographs, weigh scales, human-machine interfaces (HMIs), remote termination units, and other distributed control systems (DCSs). Modbus is a master/slave protocol in which a single master can address multiple slaves. The slave devices do not initiate any communication; they wait for a command from the master, which requests the slave to read or write data values. The control module can be configured to be **either** the master or the slave, but not both from the same port.

The Modbus Master FB Library contains the Modbus Master (MODBUS_M) function block, which allows the control module to act as a Modbus master. The Modbus Slave FB Library contains the Modbus Slave (MODBUS_S) function block, which allows the control module to act as a Modbus slave. To use a control module as a Modbus slave, refer to the *Modbus Slave Function Block Library* (document number CG39FDI-3). Both libraries also contain a Serial (SERIAL) function block, which is used to initialize the Modbus communications port.

The function blocks included in Modbus Master FB Library are configured on a configuration sheet using the *4-mation*™ configuration software. The configuration of the MODBUS_M function block establishes, inside the control module, a data array to hold the foreign device data. The control module configuration, another control module, or an operator interface can then access and use the foreign device data by referencing the array element. Configuration of the SERIAL function block enables the control module's serial port to communicate with the Modbus network and its foreign devices.

1.2 SOFTWARE REGISTRATION AND SUPPORT

The following subsections present guidelines for registering your software package and for contacting Moore Products Co. for product support.

1.2.1 Software Registration

Before reading further, complete and mail the brief Software Registration Form included in the package. Registration will:

- Enter your software's part number/version number and serial number in the master database of active software packages
- Initiate product support
- Identify you as the person to whom information about future software enhancements and updates should be sent

Without proper registration, it may be impossible to provide product support and to inform you of enhancements and updates.

Each software update or enhancement will include a Software Registration Form so that your package's current part number/version number and serial number can be recorded in the database. Additional registration forms are provided for subsequent users who become responsible for the software.

1.2.2 Product Support

Product support can be obtained from the Moore Products Co. Technical Information Center (TIC). TIC is a customer service center that provides direct telephone support on technical issues related to the functionality, application, and integration of all products supplied by Moore Products Co.

To contact TIC for support, either call **215-646-7400, extension 4TIC (4842)** or leave a message in the bulletin board service (BBS) by calling **215-283-4968**. The following information should be at hand when contacting TIC for support:

- Caller ID number, or name and company name

  (When someone calls for support for the first time, a personal caller number is assigned. This number is mailed in the form of a caller card. Having the number available when calling for support will allow the TIC representative taking the call to use the central customer database to quickly identify the caller's location and past support needs.)

- Product part number, software version, and serial number, all of which are identified on the software's disk label
- Computer brand name, model number, and hardware configuration (types of disk drives, memory size, video adapter, etc.)
- Version number of operating system (e.g. MS-DOS 6.22, Windows 3.x, Windows 95, Windows NT)

- If there is a problem with software operation:
  - The steps performed before the problem occurred
  - Any error messages displayed
  - A copy of the computer's CONFIG.SYS file
  - A copy of the computer's AUTOEXEC.BAT file

  It would also be helpful to have the following information:
  - *4-mation* version number
  - Control module ROM version
  - Documentation about your system's architecture
  - Information about the foreign device (e.g. company name, model number, software version, Modbus protocol implementation, etc.) to which the ACM\CCM and the Modbus software are trying to communicate.

For product support outside of North America, an alternative support system is available by contacting the appropriate Moore Products Co. subsidiary:

Australia
Moore Products Co. (Australia) Pty.Ltd.
Tel: (61) (2) 9319 4877

Canada
Moore Products Co. (Canada) Inc.
Tel: (905) 457 9638

France
Moore Products Co. (France)
Tel: (33) 475 05 44 62

India
Moore Controls Pvt. Limited
Tel: (91) (212) 770171

Italy
Moore Products Co. (Italia) S.r.l.
Tel: (39) (2) 2940 1094

Japan
Moore Products Co.(Japan) K.K.
Tel: (81) (3) 5484 4390

Mexico
Moore Products de Mexico S.A. de C.V.
Tel: 6-11-98-58; 6-15-19-48;
6-15-02-62; or 6-15-02-38

The Netherlands
Moore Products Co. B.V.
Tel: (00) (31) 180 461111

Singapore
Moore Products Co. (S) Pte.Ltd.
Tel: (65) 299 6051

South Africa
Moore Controls S.A. (Pty.)
Tel: (27) 466 1673/9

United Kingdom
Moore Products Co. (U.K.) Ltd.
Tel: (44) (1935) 706262

1.3 RELATED LITERATURE

The following Moore Products Co. literature is available for reference:

- *4-mation, Installation and Operation* (version 3.x binder number UM39-6, version 4.x binder number UM39-11) *
- *Function Block Language* (version 3.x binder number UM39-7, version 4.x binder number UM39-12) *
- *APACS ACM Installation & Service Instruction* (document number SD39ACM-2) *
- *QUADLOG CCM Installation & Service Instruction* (document number SDQLCCM-1) *

The following vendor literature should be available as needed:

- *Microsoft MS-DOS Operating System Reference*
- *Microsoft Windows User's Guide*
- Foreign Device's *Modbus Protocol Communications Manual*
- AEG Schneider Automation's *"Modicon Modbus Protocol Reference Guide PI-MBUS-300"* *

■

2.0 SOFTWARE INSTALLATION *

This section provides the procedures for making a backup copy of the Modbus Master FB Library software, installing the software, and performing initial software set-up. The software must be installed from floppy disks to a permanent storage medium, such as the computer's hard disk.

2.1 DISK IDENTIFICATION AND BACKUP

The Modbus Master FB Library software is shipped on a high density (HD) 3½" floppy disk. The label on the disk lists the product's part number, release number, and date, as well as the individual disk number. It is recommended that you make backup copies of the original disk. Make the copies using the disk copying procedure appropriate for your computer's operating system. Once copied, safeguard the original by storing it separately from the copies. Be sure to correctly label and write-protect the copies.

2.2 INSTALLATION PROCEDURE

The following procedures are used to install the Modbus Master FB Library software from a floppy drive to a permanent storage medium, such as a hard drive. The installation is performed from Windows Program Manager or Explorer. This is accomplished with the installation program (SETUP.EXE) resident on the library disk. Essentially, you are to load and run this program and respond to prompts. The prompts will be in the form of pop-up dialog boxes which will query you to enter information regarding your computer system. The method used to run the setup program varies depending on the specific version of Windows on the host computer. Use the appropriate procedure as needed.

IMPORTANT

> An ACM/CCM configuration created before the Modbus Master FB Library installation can be exported/imported to include the new function blocks, or a new configuration can be created and copy/paste used to place the Modbus Master and Serial function blocks into it.

To install the library under Windows 3.x:

1. Insert the library disk into the floppy drive.

2. From Program Manager, select File, Run.

3. At the Run command line, enter `drive:\SETUP.EXE` to start and initialize the setup program.

   Examples: A:\SETUP.EXE or B:\SETUP.EXE

4. When the setup program starts, you are presented with an installation screen. Pop-up dialog boxes will prompt you to enter information about your computer. Respond as needed.

To install the library under Windows 95:

1. Click on the Start button, point to Settings, and then click Control Panel.

2. Double-click on the Add/Remove Programs icon. This opens the Add/Remove Program Properties dialog box.

3. Click the Install button. This opens the Install Program From Floppy Disk or CD-ROM dialog box.

4. Insert the library disk into the floppy drive then click the Next button. This opens the Run Installation Program dialog box.

5. Verify that the command line is displaying the path and name of the setup program, then click the Finish button to commence installation.

   Examples: A:\SETUP.EXE or B:\SETUP.EXE

6. When the setup program starts, you are presented with a series of dialog boxes that prompt you for information about your computer. Respond as needed.

To install the library under Windows NT:

The procedure for installing this library under Windows NT (also called Windows NT Workstation) depends on which version of this operating system you are using. For Windows NT up to and including version 3.5.1, use the procedure for Windows 3.x above. For Windows NT version 4.0 and higher, use the procedure for Windows 95 above.

■

3.0 CONFIGURATION

The Modbus Master Function Block Library is used to configure a control module to act as a Modbus master. In this role, the control module is able to read data values from one or more slave devices. The function block can also send commands that originate elsewhere in the APACS or QUADLOG system to change data values in a Modbus slave.

3.1 PROCEDURE

The configuration procedure for the Modbus Master Library requires the same skills and knowledge needed for the configuration of other function blocks used by either an APACS or QUADLOG system. The configuration is developed by using the *4-mation* configuration software. Instructions for proper installation and use of the *4-mation* software are listed in section 1.3, Related Literature.

3.2 ACCESSING THE FUNCTION BLOCK LIBRARY

With the Modbus Master Library properly installed (see section 2.0, Installation), *4-mation* automatically includes the Modbus Master (MODBUS_M) and Serial (SERIAL) function blocks when it is started. These function blocks can then be selected from *4-mation*'s **Standard** function block list (they do not appear on the icon bar). From within *4-mation*, use the **DERIVED** key and then select the **Standard** option button to display the list of function blocks including the MODBUS_M and SERIAL blocks.

IMPORTANT

An ACM/CCM configuration created before the Modbus Master FB Library installation can be exported/imported to include the new function blocks, or a new configuration can be created and copy/paste used to place the Modbus Master and Serial function blocks into it.

3.3 GENERAL CONCEPTS

The MODBUS_M function block is needed to allow the control module to issue Modbus commands to a Modbus slave device. The control module configuration (function block diagram, ladder logic, etc.) is needed to configure the MODBUS_M block and its supporting data arrays.

Multiple MODBUS_M function blocks can reside in a control module configuration to execute multiple commands and to communicate with multiple slaves. Each MODBUS_M block is configured to communicate with a particular Modbus slave device by providing the slave's address to the block's (ADDR) input. You are to enter the desired Modbus Function Code number at the master block's command (CMD) input. Commonly supported Modbus codes include functions to read/write coils and read/write registers. The block also supports reading and writing of floating point and 32-bit integer data types. Command and response data values are handled by arrays. Input and output scaling for data values is also provided.

The purpose of the SERIAL function block is to setup one of the control module's serial ports for serial

communications with a foreign device. This function block handles the baud rate, parity, stop bits, etc. along with flow control, timeout, and buffer size parameters. The SERIAL block provides an ID output to be referenced by other serial communication function blocks that use the control module serial port. Multiple serial communication function blocks, such as the MODBUS_M, can make use of a single SERIAL block.

3.4 CONFIGURATION GUIDELINES

A common application for the MODBUS_M function block is to read values from the slave device and display and/or use these values in the APACS or QUADLOG system. Section 4.0, Function Blocks, describes the inputs, outputs, and soft list parameters of the MODBUS_M and SERIAL function blocks used in configuring the Modbus master. When a continuous update of slave values is needed, configure the CONT input to be TRUE. This causes the Modbus command to be executed as often as the control module's scan rate permits. When multiple MODBUS_M function blocks are configured in a control module, the scanning order controls the execution sequence.

The Modbus slave address, to which the control module is communicating, is entered using the ADD input. The particular Modbus Function Code to be executed is configured in the command (CMD) input. A reference address is associated with most Modbus Function Codes. The starting reference address used for a command, is entered at the START input. The quantity of sequential addresses to be read or written is entered using the number (NUM) input. When reading data from a slave, the actual data values received are placed in the configured DATA array. When sending a command to change a register value or coil state, the DATA array input contains the new value.

IMPORTANT

When structuring the database in your Modbus slave device, it is strongly suggested that a **sequential** block of discretes, registers, etc. be allocated as the data to be read by the control module Modbus master. A sequential block of data is read much more efficiently by the master resulting in improved performance. If necessary, use instructions in the slave device to move the data from scattered origins to a sequential or tightly packed group of registers, discretes, etc. to be read by the master.

Detailed descriptions of two sample applications are provided in the appendices of this Guide. Appendix A contains Application Example 1, which describes a control module configured as a Modbus master that reads thirty floating point (REAL) values from a slave device. Appendix B contains Application Example 2, which describes a control module configured as a Modbus master that reads 100 unsigned integer values from a slave address and converts them to REAL values. Additional supplemental information is provided in appendices C through G as described in section 1.0, Introduction.

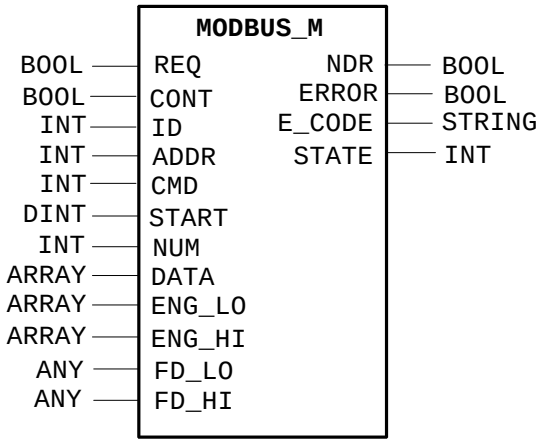
■

4.0 FUNCTION BLOCKS *

The Modbus Master Function Block Library provides the *4-mation* configuration software with a set of function blocks for configuring a control module to operate as a Modbus master. This permits an APACS or QUADLOG system to exchange data with foreign devices that use the Modbus RTU protocol. Modbus RTU is a master/slave protocol in which a single master can address multiple slaves. A listing of the library blocks is presented here. Reference information pertaining to individual blocks is provided in the sections to follow.

- Modbus Master block (MODBUS_M)
- Serial block (SERIAL)

4.1 MODBUS MASTER FUNCTION BLOCK (MODBUS_M)



The symbol for the Modbus Master (MODBUS_M) function block is shown above. This block allows the control module to operate as a Modbus master. The following subsections describe the block's inputs, outputs, and soft list parameters.

4.1.1 Inputs

This subsection defines the inputs of the MODBUS_M block.

REQ Data Type: BOOL

When this REQuest input makes a FALSE to TRUE transition, the block executes a single time. This input is used when a Modbus command is issued at discrete intervals (not continuously).

CONT Data Type: BOOL

When the CONTinuous input is TRUE, the function block executes as often as the control module's scan permits. The CONT input would typically be used when reading dynamic data values from the slave device.

ID Data Type: INT

A valid ID input must be configured for each MODBUS_M function block to execute. The ID originates from a SERIAL connect block, which can be used for multiple MODBUS_M blocks in a control module.

ADDR Data Type: INT

This is the Address input. It accepts the slave address of the Modbus device to which the block communicates. According to the Modbus protocol, valid slave addresses are 1 to 247. Note that the block does not check for an address that is outside of this range.

CMD DataType: INT

Enter a valid Modbus **Function Code** for the command to be executed by the function block. Consult the slave device's Modbus implementation regarding supported function codes, valid reference types and ranges, maximum number of values read in a single command, etc. to ensure compatibility. The **Function Codes** supported by this block are described below.

01 - Read Coil Status

DATA Array: BOOL, WORD, INT, UINT

Reads Modbus reference type Output Coil (addresses 00001 to 09999 decimal). Note that the STARTing address for MODBUS_M function block does not have the leading zeroes (e.g. Coil 00001 would be entered as 1). Maximum coils read by a function block using Command 01 is 2000; however, be aware that the slave may have a maximum read limit that is less than 2000.

02 - Read Input Status

DATA Array: BOOL, WORD, INT, UINT

Reads Modbus reference type Input Coil (addresses 10001 to 19999 decimal). The maximum coils read by a function block using Command 02 is 2000; however, be aware that the slave may have a maximum read limit that is less than 2000.

03 - Read Holding Registers

DATA Array: WORD, INT, UINT, REAL, DINT, UDINT, STRING, or
BOOL (lowest number coil is first BOOL element in array)

Reads Modbus reference type Holding Register (addresses 40001 to 49999 decimal). Maximum registers read by a function block using Command 03 are:

126	DATA Array:	WORD, INT, UINT, BOOL, REAL, DINT, or UDINT (NumBytesPerReg=2)
63	DATA Array:	REAL, DINT, UDINT (NumBytesPerReg=4)

04 - Read Input Registers

DATA Array: WORD, INT, UINT, REAL, DINT, UDINT, STRING, or
BOOL (lowest number coil is first BOOL element in array)

Reads Modbus reference type Input Register (addresses 30001 to 39999 decimal). Maximum registers read by a function block using Command 04 are:

126 DATA Array: WORD, INT, UINT, BOOL, REAL, DINT, or
UDINT (NumBytesPerReg=2)

63 DATA Array: REAL, DINT, UDINT (NumBytesPerReg=4)

05 - Force Single Coil

DATA Array: BOOL

Sends a command to change the state of a single 0xxxx coil.

06 - Preset Single Register

DATA Array: WORD, INT, UINT, REAL, DINT, UDINT, STRING, or
BOOL (lowest number coil is first BOOL element in array)

Sends a command to change the value of a single 4xxxx holding register.

08 - Loopback Test

DATA Array: WORD, INT, UINT

This is a Modbus command that provides a simple test of the communications system between the control module master and a Modbus slave. The Loopback Test command must be supported by the slave also. The block uses a Diagnostic Code of 00, which results in the query data being returned to the MODBUS_M block. In other words, what is transmitted is “loopbacked” to the master by the slave. The START address should be configured as an Input (3xxxx) or Holding Register (4xxxx), and a valid length (1 to 126) for the NUM input should be entered.

NOTE

The value of the NUM input automatically overwrites the first element of the DATA array for transmission to the slave. If the loopback test was successful, the DATA array contains the command sent, with the exception of the first element. If the loopback test failed, verify the command is supported by the slave device and the function block is configured properly.

15 - Force Multiple Coils

DATA Array: BOOL

Sends a command to change the state of multiple 0xxxx coils.

16 - Preset Multiple Registers

DATA Array: WORD, INT, UINT, REAL, DINT, UDINT, STRING, or
BOOL (lowest number coil is first BOOL element in array)

Sends a command to change the value of multiple 4xxxx holding registers. When changing the value of a floating point or double integer address that uses two consecutive registers, this command must be used (Function Code 06 will not work).

17 - Report Slave I.D.

DATA Array: WORD, INT, UINT

Allows the control module master to determine the type of slave with which it is communicating, if the command is supported by the slave. The START address should be configured as an Input (3xxxx) or Holding Register (4xxxx), although it is not explicitly used in the command, and a valid length (≥ 1) for the NUM input should be entered. When a response is returned from the slave, the Slave ID value is the first element in the DATA array, with the remaining response in subsequent elements. When viewing the response using *4-mation*'s Variable Control, select the option to display the values in hex to aid in determining the response.

65 - Coil Status Read/Write

DATA Array: BOOL

This command is unique to Moore Products Co.'s MODBUS_M function block but is implemented from two **standard** Modbus function codes, Code 01 - Coil Read and Code 05 - Force Single Coil. By combining the two standard codes, it is much easier to configure the Modbus master to change coil states in the slave device. Most of the time, this function code reads coil statuses (using code 01) and returns the values to the DATA array. If a command is sent by logic in the control module (SET_VAL block) or by a client (operator interface) on the APACS/QUADLOG side to the DATA array element, the function block automatically sends code 05 to change the coil state, then resumes the reading process.

66 - Holding Register Read/Write

DATA array: WORD, INT, UINT, REAL, DINT, UDINT, STRING

This command is unique to Moore Products Co.'s MODBUS_M function block but is implemented from two **standard** Modbus function codes, Code 03 - Holding Register Read and Code 16 - Preset Multiple Registers. By combining the two standard codes, it is much easier to configure the Modbus master to change register values in the slave device. Most of the time, this function code reads register values (using code 03) and returns the values to the DATA array. If a command is sent by logic (SET_VAL block) in the control module or by a client (operator interface) on the APACS/QUADLOG side to the DATA array element, the function block automatically sends code 16 to change the register value, then resumes the reading process.

START Data Type: DINT

This input defines the starting address for the configured command. When the ExtendedAddress soft list parameter is FALSE, the valid Address Ranges are defined as follows:

00001-09999 Read/Write Coils
10001-19999 Read Input Status
30001-39999 Read Input Registers
40001-49999 Read/Write Holding Registers

When the ExtendedAddress soft list parameter is TRUE, the valid Address Ranges are defined as follows:

000001-065535 Read/Write Coils
100001-165535 Read Input Status
300001-365535 Read Input Registers
400001-465535 Read/Write Holding Registers

NUM Data Type: INT

This input defines the quantity of coils or registers to be read or written by the configured command. When reading floating point or double integer values from the slave device, give careful notice to this entry and the NumBytesPerReg soft list entry. If the FP or DINT value is composed of two register addresses (NumBytesPerReg=2), the NUMber of registers configured should be twice the number of values desired (see Application Example 1 in Appendix A).

DATA Data Type: Single dimension array of REAL, UDINT, DINT, STRING, UINT, WORD, or BOOL, depending upon the Modbus Command number (see the Command descriptions above).

When issuing a command that reads data from the slave device, this array contains the actual data values received in the response from the slave. When the command performs a write to change a value in the slave, this array contains the new values to be written to the slave.

When declaring and handling this array, it is important to be aware of the following **important rules**:

- It must be a single dimensional array (e.g. DATAONE[1..100]).
- Do not use the Modbus reference type (0,1,3,4) as part of the DATA array size declaration. For example, if the MODBUS_M block is reading 50 unsigned integer registers from a slave (START=40301, NUM=50), the DATA array could be declared as SLVDATA[301..400]; that is, the 4 is omitted from the array size.
- Values communicated via Modbus are placed or extracted from the DATA array according to how the array is declared, the value of the START input and/or the ArrayOffset soft list parameter.

  If the ArrayOffset parameter is FALSE, the first Modbus value corresponds to the first DATA array element number. In the example above, the ArrayOffset should be set to FALSE so that the value for register 40301 is placed in the first DATA array element, in this case SLVDATA[301]. Since the SLVDATA array was declared with a range 301..400, element [301] is still the first element even though its value is not 1.

  Declaring the DATA array with element numbers that correspond to the Modbus reference may provide a convenient way to identify an array element number with the Modbus reference, depending upon the data type.

  If the ArrayOffset parameter is TRUE, the value of the START input determines the corresponding DATA array element number. Continuing with the example given above, if a second MODBUS_M block was configured to read registers 351 to 400 (START=40351, NUM=50), the same SLVDATA array could be used. However, ArrayOffset should be set to TRUE so that register 40351 is placed in SLVDATA[351]. The value of the START input, ignoring the Modbus reference type, is used to determine the offset.
- When 32-bit values are handled and the slave uses **two** registers for one value, the number of Modbus registers required will be **twice** the number of actual 32-bit values. Therefore, a one-to-one correspondence between register number and DATA array element number **will not** exist.

  As an example using 32-bit values where NumBytesPerReg=2, if the START value is 48001 and the NUM input is 100, the DATA array could be created as DATAONE[8001..8050] REALs. That is, it takes 100 registers to produce 50 floating point (REAL) values when NumBytesPerReg=2. The composite value from registers 48001 and 48002 are placed in DATAONE[8001], the value from registers 48003 and 48004 are placed in DATAONE[8002],

etc. The composite value for the last two registers, 48099 and 48100, are found in DATAONE[8050].

As an alternative, the register number association to the DATA array element number can be eliminated by declaring your DATA array to be [1..50] in the example above. In this manner, the 10th floating point value read would be placed in DATAONE[10].

- The DATA array dimensions must encompass the range of addresses configured by the START and NUM inputs. For example, if the START input is 16000 (Input Coil 6000) and the NUM is 200, the DATA array could be declared as INPUTS[6000..6199]. Again, note that the reference type designator (0,1,3, or 4) is not declared as part of the array element number.

- When reading and writing text strings, refer to the **Special Application** section under ENG_LO for details on proper DATA array declaration.

ENG_LO Data Type: Single dimension array of REALs

Some applications may require that an unsigned integer value from a slave device be scaled to a REAL value in the APACS or QUADLOG system. An element from the ENG_LO array, the corresponding element from the ENG_HI array scale the register using linear equation **y = mx + b**, where **y** is the scaled REAL value and **x** is the raw register value. The resulting scaled REAL value is placed in the DATA array in the appropriate element location.

For example, if FD_LO=0, FD_HI=65535, ENG_LO=0.00, and ENG_HI=100.00, a register value of 32768 is scaled to produce a REAL value of 50.00. This can be accomplished as follows:

DATA[a] = [(ENG_HI[a] - LO[a])/FD_HI-LO (register value - FD_LO) + ENG_LO[a]

Where: [a] is the array element number.

IMPORTANT

ENG_LO and ENG_HI array inputs and FD_LO and FD_HI values, should not be configured when 32-bit values are sourced from the Modbus slave device. These scaling factors are primarily intended for use when converting a 16-bit value (unsigned integer) from a slave device into a REAL value in the APACS or QUADLOG system.

ENG_LO Special Application - Accessing Text Strings

Data Type: INT (with STRING DATA array)

When reading a text string from a Modbus slave, the DATA array must be declared with a STRING data type and the ENG_LO with an INT data type. Each text string in your Modbus slave has a certain character length. Each Modbus register typically holds two characters. Furthermore, the Modbus slave may group several text strings into a sequential block of Modbus registers. The MODBUS_M block can be configured to read one or several text strings in these applications.

Each text string corresponds to a STRING array element in your MODBUS_M DATA array. The number of **characters** for each text string, and therefore each STRING array element, is specified by the corresponding array element in the ENG_LO array. The ENG_LO array should be declared as data type INT with the same number of elements as the DATA array. The value in the ENG_LO array element specifies the number of **characters** for the corresponding STRING.

For example, assume your Modbus slave device has three text strings, as shown below, that is read into the control module. The START input still specifies the starting Modbus register, in this case, 40325.

The DATA array is declared as SLV1TXT1[1..3], STRING data type. The ENG_LO array is declared as LENGTH1[1..3], INT data type.

SLAVE TEXT DATA	**FB DATA**	**READ RESULT**
Starting Register 40325	START = 40325	
Text String A: 8 characters (4 registers)	LENGTH[1] = 8	SLV1TXT[1]
Text String B: 16 characters	LENGTH[2] = 16	SLV1TXT[2]
Text String C: 4 characters	LENGTH[3] = 4	SLV1TXT[3]

The MODBUS_M block, when triggered, sends the configured Modbus command (usually Function Code 03), to read 14 registers starting at 40325. The NUM input does not need to be configured because the MODBUS_M block calculates the correct number of registers for you (based on two characters per register, the ENG_LO array values, and DATA array length).

ENG_HI Data Type: Single dimension array of REALs

See ENG_LO input for a description of its use.

FD_LO Data Type: REALs

See ENG_LO input for a description of its use.

FD_HI Data Type: REALs

See ENG_LO input for a description of its use.

4.1.2 Outputs

This subsection defines the outputs of the MODBUS_M block.

NDR Data Type: BOOL

When a command is received through the serial port for the configured function block, the boolean output toggles from FALSE to TRUE and then back to FALSE the next scan. This NDR (New Data Ready) output can be used to inform other logic that new data has been received or a command has completed executing.

ERROR Data Type: BOOL

An error condition received by this block causes the output to go TRUE for one control module scan. If desired, this output can be used to indicate to outside logic that an error has occurred and to assist in troubleshooting communication problems.

E_CODE Data Type: INT

This output provides a string description of an error condition. A value of 0 indicates no errors. Other error values are described in Appendix C, Error Codes.

STATE Data Type: INT

This output provides an integer value that indicates the current communication status of the function block as follows:

1 = Idle
2 = Building a command
3 = Sending a command
4 = Waiting for a response

4.1.3 Soft list Parameters

This subsection provides specifications and important supplementary information for the MODBUS_M function block soft list parameters.

NumBytesPerReg Data Type: INT (valid values are 2 and 4)

This parameter configures the function block to handle each register as either a 2-byte entity or a 4-byte entity (a byte is 8 bits). When the MODBUS_M block is reading a 32-bit value from the slave, the DATA array should be configured as REAL, DINT or UDINT, as appropriate. Some slaves, like Modicon PLCs, use two consecutive registers to represent a single 32-bit value, while others use 1 register. For proper reading (and writing) of data values, this parameter must be set correctly. In addition, the byte ordering is very important in determining a 32-bit value. This characteristic is handled by the **ModiconByteOrder** parameter discussed below.

2 Registers for one 32 bit value: NumBytesPerReg=2
1 Register for one 32 bit value: NumBytesPerReg=4

NOTE

The **NumBytesPerReg** parameter is ignored unless the DATA array is configured as either REAL, DINT, or UDINT.

ModiconByteOrder Data Type: BOOL

When this parameter is TRUE, the MODBUS_M function block assumes that the byte ordering for 32-bit floating point values from the slave is similar to Modicon PLCs. That is, a Modicon 32-bit value is sent as two registers (NumBytesPerReg=2) and the transmitted byte order is:

Register x = bytes 1 and 0, Register x+1 = bytes 3 and 2

For example, a floating point value in a Modicon PLC-compatible slave device is stored in Registers 40001 and 40002. The current value is 100.00. This value would be transmitted via Modbus protocol as:

40001	40002
00 00	42 C8

Using standard IEEE single-precision floating point format, this would be interpreted as the value 100.00 REAL in an APACS or QUADLOG system.

When this parameter is FALSE, the MODBUS_M function block expects the byte order to be "word-swapped" from the Modicon byte order. That is, the value above would be transmitted as:

40001	40002
42 C8	00 00

NOTE

The **ModiconByteOrder** parameter is ignored unless the DATA array is configured as either REAL, DINT, or UDINT. The Byte Order also operates on values that use four bytes per register.

ArrayOffset Data Type: BOOL

When this parameter is FALSE, the data values are placed in the first element location of the declared DATA array, regardless of the START input value. When ArrayOffset is set to TRUE, an offset is used when placing values in the DATA array. This offset process actually causes values to be placed in the array according to the configured START input value. For example, if the START value is 41626, ArrayOffset=TRUE, and the DATA array is declared as UINT data type (DATA_ONE[501..750]), the values from a register read command are placed in the DATA_ONE array, starting at element [626] not the first element [501].

This feature is useful in applications where several full-capacity MODBUS_M blocks are used to bring data into the control module and build a single DATA array (for example, MODBUS_M_1 reads registers 41501 to 625 and places data in DATA_ONE[501] to [625]. MODBUS_M_2 reads registers 41626 to 750 and places data in DATA_ONE [626] to [750]. The complete range of values can then be accessed by an APACS or QUADLOG client by reading array DATA_ONE. The ArrayOffset=TRUE must be set for MODBUS_M_2).

Default: FALSE

ModiconLoopback Data Type: BOOL

When this parameter is TRUE, the MODBUS_M function block uses the MODICON Loopback Diagnostic code of 0, which echoes back to the master the same number of bytes received.

When this parameter is FALSE, the MODBUS_M function block uses the MPCO (Moore Products Co.) Loopback Diagnostic code of 5, which uses a word count in the message to determine the amount of data to echo. This format is well suited for radio modem testing, which often has additional but unneeded data bytes on the end of each message.

Default: TRUE

PrioritizeWrites Data Type: BOOL

When this parameter is set to FALSE, Modbus writes have the same priority as reads. When set to TRUE, any writes generated by this function block have priority over any other Modbus reads from any other Modbus function block.

Set this parameter to FALSE in applications where there may be a concern of continuous Modbus writes locking out normal Modbus reads. However, in most applications, Modbus master writes to a slave are usually infrequent and/or triggered by operator action.

Default: TRUE

ExtendedAddress Data Type: BOOL

When this parameter is FALSE, the valid address ranges are defined as follows:

00001-09999 Read/Write Coils
10001-19999 Read Input Status
30001-39999 Read Input Registers
40001-49999 Read/Write Holding Registers

When this parameter is TRUE, the valid address ranges to support extended addressing are defined as follows:

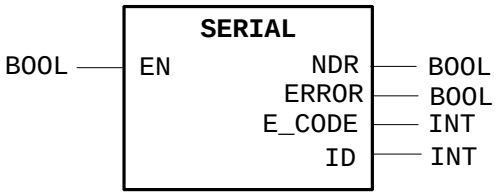
000001-065535 Read/Write Coils
100001-165535 Read Input Status
300001-365535 Read Input Registers
400001-465535 Read/Write Holding Registers

Default: FALSE

Version Data Type: STRING

This parameter is intended to specify the minimum APACS or QUADLOG version compatibility level for using this function block. Its use is flexible. For example, you can use it to enter the control module firmware revision number, *4-mation*'s software version number, or both. Once entered, you can determine compatibility by checking a software number against the number(s) in this parameter.

4.2 SERIAL FUNCTION BLOCK (SERIAL)



The symbol for the Serial (SERIAL) function block is shown above. This block is used to initialize the Modbus communications port. Specifically, it enables a control module's serial port to communicate with the Modbus network and its foreign devices. This section defines the block's input, outputs, and soft list parameters.

4.2.1 Input

This subsection provides specifications and important supplementary information for the SERIAL function block input.

EN Data Type: BOOL

When this input is set to FALSE, the SERIAL block is not active. When TRUE, the function block generates an ID output, which is used by the other serial communication function blocks. When active, the function block handle requests from the communication function blocks and external devices connected to the serial port.

4.2.2 Outputs

This subsection provides specifications and important supplementary information for the SERIAL function block outputs.

NDR Data Type: BOOL

Reserved for future use.

ERROR Data Type: BOOL

This output is TRUE when the last transmitted or received message contains an error. More information about the error is provided by the E_CODE output.

E_CODE Data Type: INT

A 0 (zero) output indicates that no general serial port errors exist. Error conditions particular to a serial port command are passed to the appropriate communication function block (e.g. MODBUS_M), which has its own E_CODE output.

ID Data Type: INT

If the SERIAL block is functioning, it provides an ID output which must be used by other serial communication function blocks as the ID input. An output of 0 indicates the SERIAL block is not functioning.

4.2.3 Soft List Parameters

This subsection provides specifications and important supplementary information for the SERIAL function block soft list parameters.

BaudRate Data Type: UDINT

This parameter sets the communication baud rate between the control module and the foreign serial device. The baud rates must be the same for all the serial communication devices.

Valid Entries: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 125000, 143000, 170000, 200000, 250000

Default: 9600

DataBits Data Type: UINT

The number of data bits used to represent a character must be specified. For Modbus, the data bits should be set to 8.

Valid Entries: 7, 8

Default: 8

Parity Data Type: UINT

The parity mode must be the same for all serial communication devices. Traditional Modbus devices use even parity but be sure to check the foreign device's documentation.

Valid Entries: 0=None, 1=Odd, 2=Even

Default: 0=None

StopBits Data Type: UINT

The number of stop bits used for a serial transmission must be specified.

Valid Entries: 1, 2

Default: 1

FlowControl Data Type: UINT

Flow control specifies what type of hardware handshaking is used between the serial communication devices. If the options below do not satisfy your application, you may have to hardwire the necessary handshaking lines. Refer to Appendix E for typical serial cable pin-outs.

Valid Entries: 0=RTS (Pin 8) held asserted
1=RTS asserted when control module transmits
2=Full RTS/CTS operation
3=RTS held asserted with CTS observed before transmitting

Default: 0

IMPORTANT

When using a FlowControl setting other than 0, the control module serial port Data Carrier Detect (DCD) input, pin 4, must be connected to a logic TRUE pin from the Modbus device serial port, such as a DCD output or a Data Terminal Ready (DTR) signal. If the Modbus device does not have this signal, pin 4 of the control module serial port can be tied back to pin 8, the Request to Send (RTS) output.

Timeout Data Type: UINT

When a control module initiates a serial transmission, it waits for the specified period of time for a response from the foreign device before indicating a timeout.

Valid Entries: 0 to 65535 (time base is milliseconds)

Default: 3000 (3.0 seconds)

ReceiveBufSize Data Type: UINT

Reserved for future use.

TransmitBufSize Data Type: UINT

Reserved for future use.

SIM_RackNum Data Type: UINT

Reserved for future use.

SIM_SlotNum Data Type: UINT

Reserved for future use.

PortNum Data Type: UINT

This parameter should be set to the control modules's serial port number connected to the Modbus device.

Valid Entries: 1 to 8

Default: 2

Version Data Type: STRING

This parameter is intended to specify the minimum APACS or QUADLOG version compatibility level for using this function block. Its use is flexible. For example, you can use it to enter the control module firmware revision number, *4-mation*'s software version number, or both. Once entered, you can determine compatibility by checking a software number against the number(s) in this parameter.

■

A.0 APPENDIX A — APPLICATION EXAMPLE 1 *

In this application example, the Modbus master (ACM or CCM) reads 30 floating point (FP) values from slave address 5, starting at register 40101. The slave device handles FP values similar to a Modicon PLC; therefore, each value requires two consecutive registers for reading (and writing). See Figures A-1 through A-4 for configuration examples.

A Serial block, SERIAL1, is configured with the EN input variable SER1_EN. When this is TRUE, the function block sets up and handles serial communications with the control module's serial port 2. If successful, the ID output has a non-zero value, which can be used as the ID input to a communication function block. Be sure to configure the BaudRate, Parity, and StopBits parameters correctly. In this example, the BaudRate is 19200, Parity = 0 (none), and StopBits = 1.

For the MODBUS_M block in Figure A-3, the READ1 variable is set with an initial value of TRUE and configured as the CONT input since the application requires continuous update of these dynamic floating point (FP) values. By using a variable as the CONT input rather than a constant TRUE, the application can easily stop this function by writing a FALSE to READ1. The ID1 input comes from the SERIAL block configured for control module serial port 2. The ADDR input is set to 5 by the ADDR5 integer variable. The Modbus register read function code 3 is requested by entering the CMD3 variable as the CMD input. The double integer variable R40101 has a value of 40101 for the START input, and the NUM_60 input is set to 60. Note that the number of **registers** (60) is twice the number of **values** (30) needed since two registers compose one floating point value.

The values received from the read command are placed in the DATA1 array. This array is defined as a one-dimensional array of REALs with dimensions 101 to 130. Therefore, each FP value can be found in its corresponding array element location. For example, the tenth FP value (slave registers 40119 and 40120) are placed in the APACS/QUADLOG variable DATA1[110], which can be read like any other variable in the control module.

Note how the soft list is configured for this command. NumBytesPerReg is set to 2 since each 2-byte register address represents half the floating point value (4-byte total). ModiconByteOrder is selected as TRUE since the slave device returns a FP value with the same byte order as a Modicon PLC floating point value. No scaling is needed for this configuration since FP values are handled by both master and slave devices.

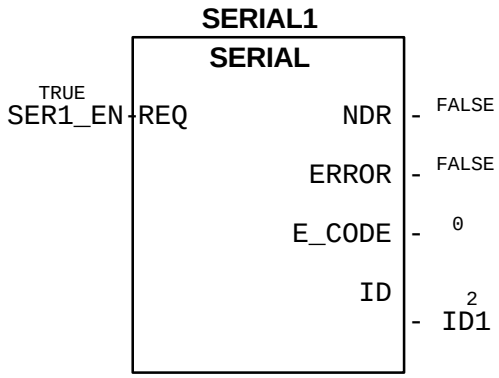


FIGURE A-1 SERIAL Block Configuration Showing On-line Data for Example 1

Soft List

Param Name	Data Type	Privilege	Value
BaudRate	UDINT	READ_WRITE	19200
Parity	UINT	READ_WRITE	0
StopBits	UINT	READ_WRITE	1
FlowControl	UINT	READ_WRITE	0
Timeout	UINT	READ_WRITE	3000
ReceiveBufSize	UINT	READ_WRITE	128
TransmitBufSize	UINT	READ_WRITE	128
SIM_RackNum	UINT	READ_WRITE	0
SIM_RackNum	UINT	READ_WRITE	0
PortNum	UINT	READ_WRITE	2
Version	STRING	READ_ONLY	'4.00'

FIGURE A-2 SERIAL Block Soft List for Example 1

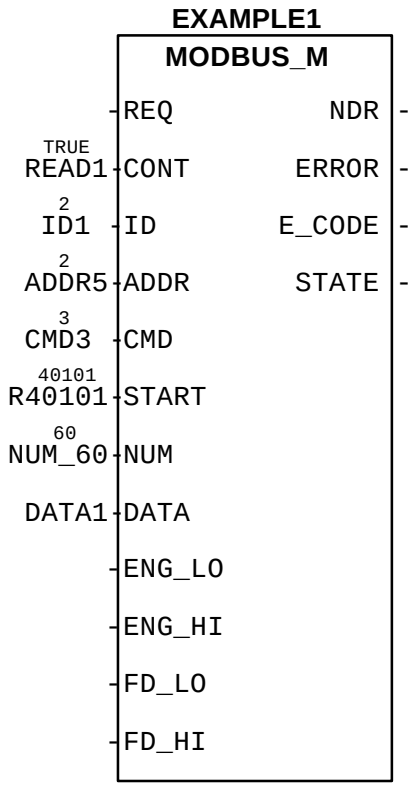


FIGURE A-3 MODBUS_M Configuration Showing On-line Data for Example 1

Soft List

Param Name	Data Type	Privilege	Value
NumBytesPerReg	INT	READ_WRITE	2
ModiconByteOrder	BOOL	READ_WRITE	TRUE
ArrayOffset	BOOL	READ_WRITE	FALSE
PrioritizeWrites	BOOL	READ_WRITE	TRUE
ModiconLoopback	BOOL	READ_WRITE	TRUE
ExtendedAddress	BOOL	READ_WRITE	FALSE
Version	STRING	READ_ONLY	'4.00'

FIGURE A-4 MODBUS_M Soft List for Example 1

■

B.0 APPENDIX B — APPLICATION EXAMPLE 2 *

In this application example, the control module is reading 100 unsigned integer values from slave address 5, starting at register 40001, and converting those to REAL values for use in the control module. Similar to standard register values in a Modicon PLC, each register represents one value with a possible range from 0 to 65535 (16-bit). Internally, the slave device may scale these values to an engineering range, such as 0 to 200°F (0 = 0°F and 4095 = 200°F). See Figure B-1 for the associated MODBUS_M block configuration.

The READ2 variable is set with an initial value of TRUE and configured as the CONT input since the application requires continuous update of these dynamic unsigned integer values. By using a variable as the CONT input rather than a constant TRUE, the application can easily stop this function by writing a FALSE to READ2.

The ID1 input comes from the SERIAL block configured for control module serial port 2. The ADDR input is set to 5 by the ADDR5 integer variable. The Modbus register read function code 3 is requested by entering the CMD3 variable as the CMD input. The R40001 variable with a DINT value of 40001 is configured as the START input, and the NUM_100 input is set to 100.

The values received from the read command are placed in the DATA2 array. This array is defined as a one-dimensional array of REALs with dimensions 1 to 100. Since most analog values in an APACS or QUADLOG system are REALs, this block is configured to scale each unsigned integer value from the slave into a REAL value. The scaling information is provided by the FD_LO and FD_HI values and the ENG_LO and ENG_HI values. Since the block is performing a read function, the scaling values are used as follows: the FD_LO and FD_HI are used as the “from” values, the ENG_LO and ENG_HI are used as the “to” values. Note that the “from” values are the same for all registers read by this block, but each REAL value has its own set of ENG_LO and ENG_HI factors. For example, if FD_LO=0, FD_HI=65535, ENG_LO=0.00, and ENG_HI=100.00, a register value of 32768 is scaled to produce a REAL value of 50.00.

An ability is provided to scale each value differently as it is stored in the DATA2 array, since an ENG_LO and ENG_HI pair exists for each array element. The ENG_LO and ENG_HI are also arrays. However, the raw data values coming from the slave device are all subject to the same scaling factors since only one set of FD_LO and FD_HI values exist. In this example, FD_LO and FD_HI are declared as REAL variables and the ENG_LO and ENG_HI are arrays of REALs.

The soft list parameters are not used by the block in this case since the FD_LO and FD_HI entries are configured, which indicates that 16-bit values are being read from the slave device.

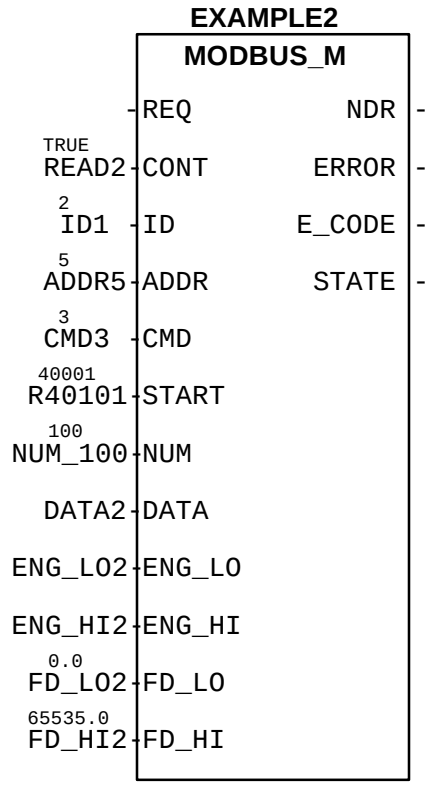


FIGURE B-1 MODBUS_M Configuration Showing On-line Data for Example 2

■

C.0 APPENDIX C — ERROR CODES *

The error code (E_CODE) output from the Modbus Master (MODBUS_M) function block provides a STRING that lists one or more of the following error code messages listed in Table C-1.

The E_CODE output clears with each execution of the block. To view the string in *4-mation*'s on-line mode, use its Variable Control function to set the block's Request (REQ) and Continuous (CONT) inputs to FALSE. Then, toggle the REQ input TRUE (do not pulse the input). The block executes once and the error string can be read until the REQ input is changed.

TABLE C-1 Error Codes

ERROR CODE (E_CODE)	DESCRIPTION OR POSSIBLE CAUSE	RECOMMENDED ACTION
03 Backup_Confirmation_Failed	• A message was received by only one side of a redundant pair. • A message was discarded. • There is a cable or signal converter problem.	Check all cables, signal converters (RS-422/RS-232), or modems used.
04 Checksum_Problem	• A partial message was received. • There was a CRC error in the message. • An unknown message was received.	Check source of message for proper setup. Check hardware connections.
06 Message_Problem	• This is a loopback message failure. • There is a possible communication hardware problem.	Check all cables, signal converters (RS-422/RS-232), or modems used.

TABLE C-1 Error Codes (Continued)

ERROR CODE (E_CODE)	DESCRIPTION OR POSSIBLE CAUSE	RECOMMENDED ACTION
07 Transmit_Busy	The serial port is busy servicing another function block.	No action required if multiple function blocks are configured for one serial port.
08 Old_Message_Removed	• A message was discarded from the serial port that was not serviced within three controller scans. • Possibly, a MODBUS block was not configured to handle this message.	• Check the source of message for proper setup. • Use a protocol analyzer to decipher the message for additional information. • Verify the MODBUS block configuration.
09 General_Uart_Problem	• There is a problem with the ACM/CCM's serial port. • There is a conflicting configuration using the same serial port. • An attempt was made to use a serial port other than 1 or 2.	• Check for multiple SERIAL blocks using one serial port. • Check for conflicting function blocks using the same serial port.
10 Re_Initializing_Port	A MODBUS function block is reinitializing because of a configuration change.	If this code persists, check for changing inputs or soft list parameters on MODBUS function blocks.
11 No_ID_Present	The ID input of a MODBUS function block is not configured or zero. This would be a normal code if you choose to disable a set of function blocks so they do not use a particular serial port.	Configure the ID input of the MODBUS block as the output of the SERIAL block and enable the SERIAL block.

TABLE C-1 Error Codes (Continued)

ERROR CODE (E_CODE)	DESCRIPTION OR POSSIBLE CAUSE	RECOMMENDED ACTION
12 Invalid_Range_Inputs	• The ENG_LO or ENG_HI inputs are not configured properly. These inputs should only be configured with a REAL array and two bytes per register. • The span is equal to zero. • The string size is greater than 255 characters. • The ENG_HI input was configured using signed integer conversion.	Check for proper configuration of the ENG_LO and ENG_HI inputs. See section 4.1.1, Inputs.
13 Default_Message_Problem	An unknown Modbus command was received.	Check the sending device.
14 Array_Problem	On the MODBUS block, either the DATA, ENG_LO, or ENG_HI input array is not configured properly. Arrays must be one dimensional. If the input array is a string array, the ENG_LO input requires an INT array. The ENG_LO and ENG_HI arrays must have the same number of elements as the DATA array.	Check the configuration of the MODBUS block's ENG_LO, ENG_HI, and DATA inputs. See section 4.1.1, Inputs.
15 Array_DataType_Problem	The DATA input array variable type is invalid.	Verify the MODBUS block's DATA input with respect to its CMD, START, and NUM inputs. See section 4.1.1, Inputs.
18 Max_Size_Exceeded	The NUM input exceeds the maximum allowed for the DATA and CMD inputs.	Verify the MODBUS blocks's NUM input based on the DATA input variable type and CMD input. See section 4.1.1, Inputs.

TABLE C-1 Error Codes (Continued)

ERROR CODE (E_CODE)	DESCRIPTION OR POSSIBLE CAUSE	RECOMMENDED ACTION
21 Invalid_Inputs	The function code specified on the MODBUS block's CMD input is not compatible with the START input's address range.	Verify the MODBUS block's CMD and START inputs. See section 4.1.1, Inputs.
22 Invalid_Command	The MODBUS block's CMD input is invalid.	Verify the CMD input is supported by the MODBUS function block.
23 Timeout_Error	The Modbus device did not respond within the configured time-out period.	Use a protocol analyzer to verify communications between the APACS/QUADLOG system and the Modbus device.
24 Unknown_Exception_Received	The MODBUS_M block has received an exception response with an unsupported exception code.	• Check the source of message for proper setup. • Use a protocol analyzer to decipher the message for additional information.
25 Invalid_Address	The MODBUS block's START input address may be invalid. The START and NUM input values may not match the DATA, ENG_LO or ENG_HI array sizes.	Verify the START, NUM and DATA inputs for valid address ranges.
28 CTS_Lost_During_Transmission	When using FlowControl options 2 or 3, the CTS line went low before transmission was completed.	Monitor serial communications parameters and check modem if problem persists.
29 Unknown_TX_Error	This is an unknown serial port driver transmission problem.	Check for misconfiguration of serial port, bad wiring or incompatible communication parameters (e.g. baud rate, parity, etc).

TABLE C-1 Error Codes (Continued)

ERROR CODE (E_CODE)	DESCRIPTION OR POSSIBLE CAUSE	RECOMMENDED ACTION
30 Unknown_RX_Error	This is an unknown serial port driver reception problem.	Check for misconfiguration of serial port, bad wiring or incompatible communication parameters (e.g. baud rate, parity).
31 Illegal_Function_Exception	The Modbus slave device does not understand the function code.	Verify that the function code used is supported by the Modbus slave device.
32 Illegal_Data_address_Exception	The starting or ending Modbus address, data register, or coils is out of range.	Verify that the value used is within the defined range.
33 Illegal_Data_value_Exception	• An illegal data value was written. • A coil or register write failed for security reasons.	• Correct the data value. • Check the MODBUS_S block's security settings.
34 Connection_Lost	The MODBUS_M block has lost contact with its Modbus slave device.	• Check all cables, signal converters (RS-422/RS-232), or modems used to connect to the slave device. • Check the Modbus slave device for proper operation.

■

D.0 APPENDIX D — SPECIFICATIONS *

General specifications for the Modbus Function Block Library are listed in Table D-1. Where applicable, default values are shown in **bold** type.

TABLE D-1 Library Specifications

ITEM	SPECIFICATION
Library Software Part Numbers	APACS 15939-623V4.00 Modbus Master QUADLOG 15939-681V3.20 Modbus Master Note that the Vx.xx suffix refers to the software version number. It will differ for later versions.
Required *4-mation* Version	3.00 or later
Electrical I/O Port	RS232 (Other physical layers can be accommodated with external converters, such as RS422 for multi-drop applications.) For RS422/RS485, use 4-wire only.
Transmission Mode	Modbus RTU (binary)
Baud Rate	110, 300, 600, 1200, 2400, 4800, **9600**, 19200, 38400, 57600, 115200, 125000, 143000, 170000, 200000, 250000
Start Bits	1
Data Bits	7 or **8**
Parity	Odd, Even, Space, Mark, or **None**
Stop Bits	**1** or 2
Error Checking	CRC-16
Slave Address	1 to 247
Supported Function Codes	01 - Read Coil Status 02 - Read Input Status 03 - Read Holding Registers 04 - Read Input Registers 05 - Force Single Coil 06 - Preset Single Register 08 - Loopback Test 15 - Force Multiple Coils 16 - Preset Multiple Registers 17 - Report Slave I.D. 65 - Coil Status Read/Write * 66 - Holding Register Read/Write* * MODBUS_M only

TABLE D-1 Library Specifications (Continued)

ITEM	SPECIFICATION
Supported Address Types	00001-09999 Read/Write Coils 10001-19999 Read Input Status 30001-39999 Read Input Registers 40001-49999 Read/Write Holding Registers Extended Addressing: 000001-065535 Read/Write Coils 100001-165535 Read Input Status 300001-365535 Read Input Registers 400001-465535 Read/Write Holding Registers
Device Data Types	Boolean Word Signed Integer Unsigned Integer Double Signed Integer Double Unsigned Integer Floating Point (2-register and 1-register formats; two different byte orders) String

■

E.0 APPENDIX E — CABLE CONNECTIONS

The pin-out for a control module serial port (DB9 female) is shown in the Table E-1. As a minimum, the cable between the control module serial port and the foreign device will require TD, RD and SG as shown in Figure E-1. The foreign device may require additional handshaking signals for proper operation (consult the foreign device documentation.) The FlowControl soft list parameter of the SERIAL block may also be useful in selecting the proper handshaking mode or these signals may have to be tied back to the foreign device (RTS to CTS, and DTR to DSR and DCD) as shown in Figure E-1.

TABLE E-1 Control Module Serial Port Pin-Out

PIN #	DESCRIPTION	DIRECTION
1	No Connection	
2	Transmitted Data (TD)	Output
3	Received Data (RD)	Input
4	Data Carrier Detect (DCD)	Input
5	Signal Ground (SG)	N/A
6	No Connection	
7	Clear to Send (CTS)	Input
8	Request to Send (RTS)	Output
9	Ring Indicator (RI)	Input

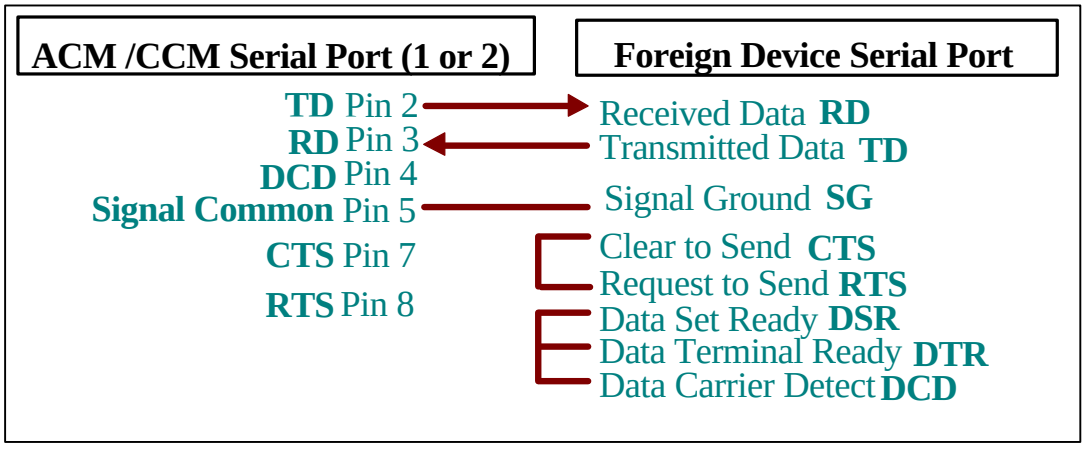


FIGURE E-1 Minimum Serial Cable Requirements

NOTES

The control module's serial port pin-out is not the same as that typically found on a personal computer. The control module's serial port is a DB9 female connection, using different pins than a PC for transmitting (TD) and receiving (RD) data.

When using a FlowControl setting other than 0, the control module's serial port Data Carrier Detect (DCD) input, pin 4, must be connected to a logic TRUE pin from the Modbus device serial port such as a Data Carrier Detect (DCD) output or a Data Terminal Ready (DTR) signal. If the Modbus device does not have this signal, pin 4 of the control module can be tied back to pin 8, the Request to Send (RTS) output.

■

F.0 APPENDIX F — ENTRELEC RS232/485 CONVERTER CONNECTIONS *

Cable connections needed to interface an RS-485 foreign device to the ACM/CCM RS-232 serial port by means of the Entrelec RS232/485 Converter are shown in Figure F-1. This converter is recommended by and available from Moore Products Co.

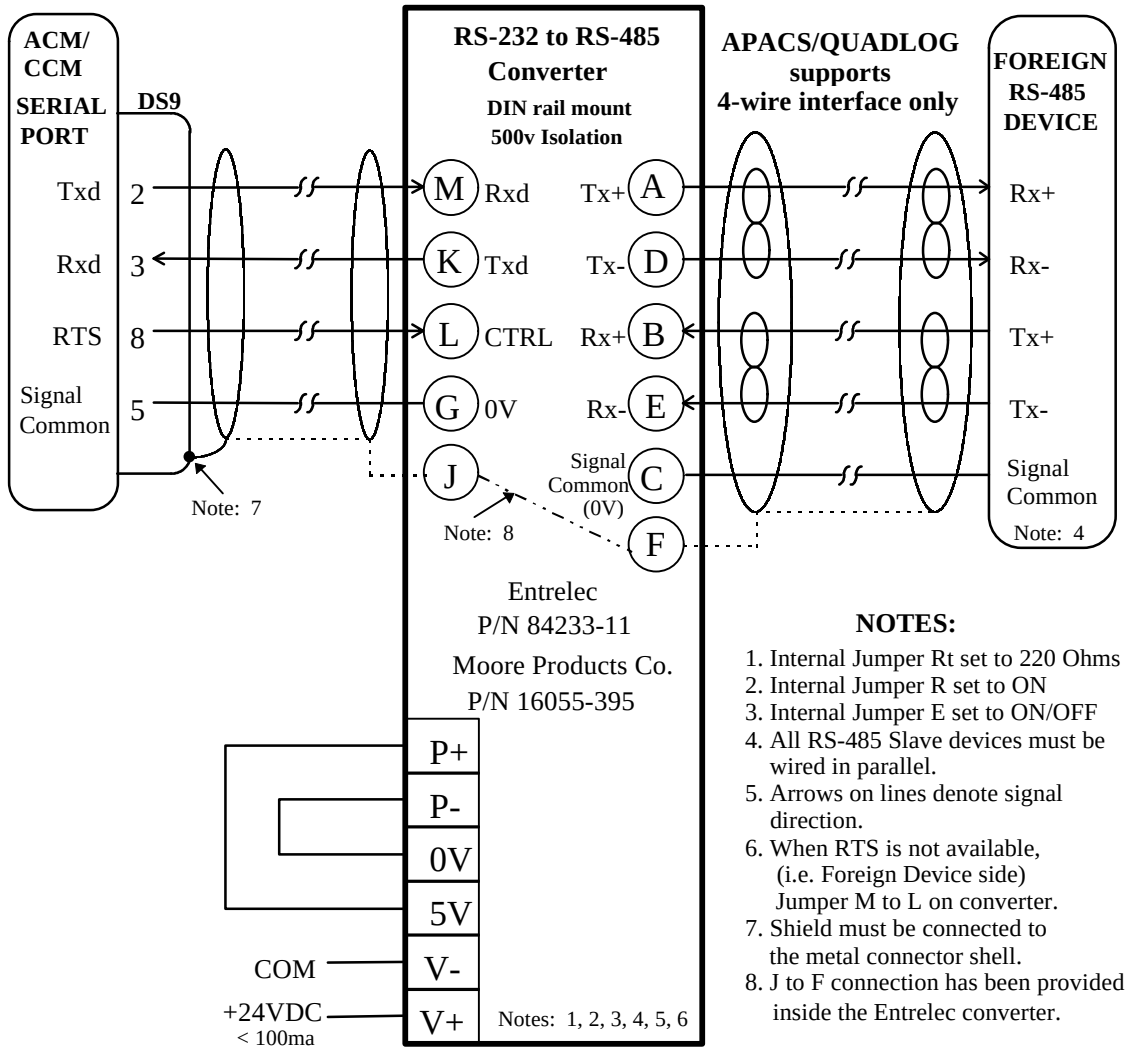


FIGURE F-1 Entrelec RS-232/RS-485 Converter Cable Connections

G.0 APPENDIX G — REDUNDANT SYSTEM WIRING CONFIGURATION *

Figure G-1 shows the general wiring needed for a dual Modbus network in a redundant system. For typical RS-232 and RS-485 cable connections, refer to Appendix F.

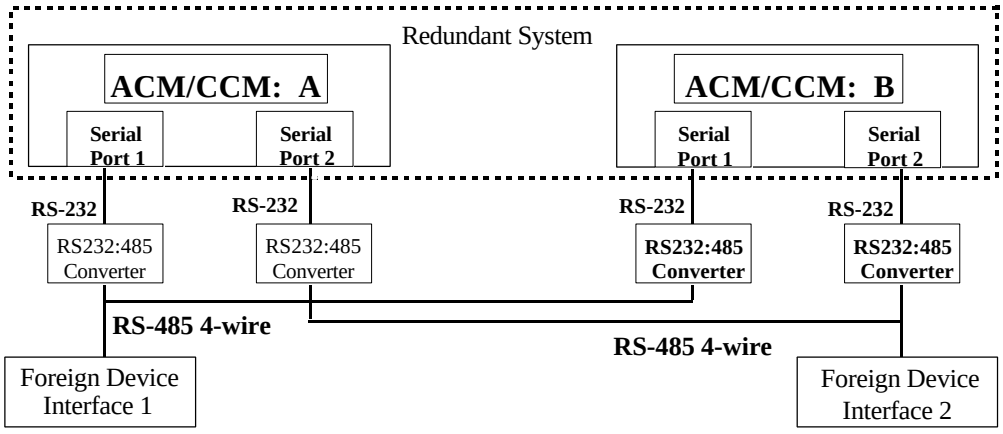


FIGURE G-1 Redundant System Wiring Configuration

■