Acromag®

BusWorks™ 900MB Series
Modbus/RS485 Network I/O Modules

Technical Reference

INTRODUCTION TO MODBUS

ACROMAG INCORPORATED
30765 South Wixom Road
P.O. BOX 437
Wixom, MI 48393-7037 U.S.A.

Tel: (248) 624-1541
Fax: (248) 624-9234

8500-648-A02F000

TABLE OF CONTENTS

INTRODUCTION TO MODBUS

This information is provided as a service to our customers and to others interested in learning more about Modbus. Acromag assumes no responsibility for any errors that may occur in this document, and makes no commitment to update or keep this information current.

Be sure to visit Acromag on the web at **www.acromag.com.**

The following information describes the operation of Modbus as it relates to Acromag Series 900MB I/O modules. For more detailed information on Modbus, you may also refer to the "Modicon Modbus Reference Guide", PI-MBUS-300 Rev J, available via download from www.public.modicon.com.

Acromag manufactures a line of I/O modules that support Modbus over RS485. Feel free to visit our website at www.acromag.com to obtain the latest information about these and other Acromag products.

ABOUT MODBUS

The Modbus protocol provides an industry standard method that Modbus devices use for parsing messages. This protocol was developed by Modicon, Incorporated, for industrial automation systems and Modicon programmable controllers.

Modbus devices communicate using a master-slave technique in which only one device (the master) can initiate transactions (called queries). The other devices (slaves) respond by supplying the requested data to the master, or by taking the action requested in the query. A slave is any peripheral device (I/O transducer, valve, network drive, or other measuring device) which processes information and sends its output to the master using Modbus. Acromag Series 900MB I/O Modules are slave devices, while a typical master device is a host computer running appropriate application software.

Masters can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a response to all queries addressed to them individually, but do not respond to broadcast queries.

A master's query consists of a slave address (or broadcast address), a function code defining the requested action, any required data, and an error checking field. A slave's response consists of fields confirming the action taken, any data to be returned, and an error checking field. Note that the query and response both include a device address, plus a function code, plus applicable data, and an error checking field. If no error occurs, the slave's response contains the data requested. If an error occurs in the query received, or if the slave is unable to perform the action requested, the slave will return an exception message as its response (see Modbus Exceptions). The error check field of the message frame allows the master to confirm that the contents of the message are valid. Additionally, parity checking is also applied to each transmitted character in its data frame.

MODBUS REGISTER MAP

Modbus devices usually include a Register Map. Modbus functions operate on register map registers to monitor, configure, and control module I/O. You should refer to the register map for your device to gain a better understanding of its operation. You will also find it helpful to refer to the register map as you review the Modbus functions described later in this document.

Modbus registers are organized into reference types identified by the leading number of the reference address:

The "x" following the leading character represents a four-digit address location in user data memory.

The leading character is generally implied by the function code and omitted from the address specifier for a given function. The leading character also identifies the I/O data type.

Reference	Description
0xxxx	Read/Write Discrete Outputs or Coils. A 0x reference address is used to drive output data to a digital output channel.
1xxxx	Read Discrete Inputs. The ON/OFF status of a 1x reference address is controlled by the corresponding digital input channel.
3xxxx	Read Input Registers. A 3x reference register contains a 16-bit number received from an external source—e.g. an analog signal.
4xxxx	Read/Write Output or Holding Registers. A 4x register is used to store 16-bits of numerical data (binary or decimal), or to send the data from the CPU to an output channel.

Not all Modbus functions operate on register map registers. For example, with respect to Acromag 900MB modules, the Report Slave ID and Reset Slave functions do not operate on register map locations.

Note that Acromag modules provide an alternate method of accomplishing reset if your application software does not support the Reset Slave function (see the Module Reset Register).

MODBUS SERIAL TRANSMISSION MODES

Standard Modbus networks employ one of two types of transmission modes: ASCII Mode, or RTU Mode. The transmission mode defines the bit contents of the message bytes transmitted along the network, and how the message information is to be packed into the message stream and decoded. The mode of transmission is usually selected along with other serial port communication parameters (baud rate, parity, etc.) as part of the device configuration. Some devices may only support one mode or the other. However, the transmission mode and serial parameters must be the same for all devices connected to a Modbus network.

ASCII Mode (American Standard Code for Information Interchange)

In the ASCII Transmission Mode (American Standard Code for Information Interchange), each character byte in a message is sent as 2 ASCII characters. This mode allows time intervals of up to a second between characters during transmission without generating errors. The format of each byte in ASCII mode is outlined as follows:

MODBUS SERIAL TRANSMISSION MODES

ASCII Mode Byte Format	
Coding System	Hexadecimal of ASCII characters 0-9, and A-F. One hexadecimal character contained in each ASCII character (7 bits) of the message.
Bits Per Byte	1 start bit + 7 data bits, lsb sent first + 1bit for even/odd parity or no bit for no parity + 1 stop bit if parity is used or 2 stop bits with no parity.
Error Check Field	Longitudinal Redundancy Check (LRC)

Keep in mind that in ASCII Mode, it takes two data words (two 7-bit characters) to transfer an equivalent 8-bit byte in RTU Mode. That is, each 4-bit nibble of the 8-bit RTU byte requires one ASCII character (7 bits). One RTU Byte is essentially two bytes in ASCII Mode. For example, the value 6AH is sent as one 8-bit byte in RTU mode (0110 1010), but two bytes in ASCII Mode, one for ASCII “6” (“6” = 36H = 011 0110), one for ASCII “A” (“A” = 41H = 100 0001). This makes the ASCII Mode less efficient and less favored than RTU Mode for most applications.

Acromag 900MB modules do not use the ASCII Mode of transmission, they use the RTU Mode described below.

RTU Mode (Remote Terminal Unit)

In RTU (Remote Terminal Unit) Mode, each 8-bit message byte contains two 4-bit hexadecimal characters, and the message is transmitted in a continuous stream. The greater effective character density increases throughput over ASCII mode at the same baud rate. The format for each byte in RTU mode is outlined as follows:

RTU Mode Byte Format	
Coding System	8-bit binary, hexadecimal 0-9, A-F, two hexadecimal characters in each 8-bit field of the message.
Bits Per Byte	1 start bit + 8 data bits, lsb sent first + 1bit for even/odd parity or no bit for no parity + 1 stop bit if parity is used or 2 stop bits with no parity.
Error Check Field	Cyclical Redundancy Check (CRC)

Acromag Series 900MB modules utilize the widely accepted Modbus network protocol in the more efficient RTU (Remote Terminal Unit) serial transmission mode.

MODBUS MESSAGE FRAMING

A Modbus message is placed in a message frame by the transmitting device. Do not confuse the message frame with the data frame of a single byte (RTU Mode) or 7-bit character (ASCII Mode). A message frame is used to mark the beginning and ending point of a message allowing the receiving device to determine which device is being addressed and to know when the message is completed. It also allows partial messages to be detected and errors flagged as a result. Each word of this message (including the frame) is also placed in a data frame that appends a start bit, stop bit, and parity bit. In ASCII mode, the word size is 7 bits, while in RTU mode, the word size is 8 bits. Thus, every 8 bits of an RTU message is effectively 11 bits when accounting for the start, stop, and parity bits of the data frame (with one exception—see note below).

MODBUS MESSAGE FRAMING

Note: Some Modbus software will misinterpret "no parity" as equivalent to "no bit transmitted". According to the Modbus standard, no parity will add an additional stop bit (maintaining an 11 bit data frame in RTU mode). To prevent potential incompatibilities, Acromag 900MB modules are designed to accept messages with no parity and 1 or 2 stop bits.

The structure of the data frame depends on the transmission mode (ASCII or RTU). Note that on some other network types and on Modbus Plus, the network protocol handles the framing of messages and uses start and end delimiters specific to the network.

ASCII Mode Message Frames

ASCII Mode messages start with a colon character ":" (ASCII 3AH) and end with a carriage return-line feed pair of characters (CRLF, ASCII 0DH & 0AH). The only allowable characters for all other fields are hexadecimal 0-9 & A-F. Recall that it only takes 7 significant bits to represent an ASCII character. Likewise, the Modbus ASCII Mode data 'byte' or character is only 7 bits long.

Allowable ASCII Mode Characters

ASCII	HEX	ASCII	HEX
0	30H	**:**	3AH
1	31H	**CR**	0DH
2	32H	**LF**	0AH
3	33H	**A**	41H
4	34H	**B**	42H
5	35H	**C**	43H
6	36H	**D**	44H
7	37H	**E**	45H
8	38H	**F**	46H
9	39H		

For ASCII Mode transmission, each character requires 7 data bits. Thus, each character is 10 bits when accounting for the start bit, parity bit, and stop bit of the data frame.

In ASCII Mode, all network devices continuously monitor the network for the 'start of message' colon (:) character. When it is received, every network device decodes the next field to determine if it is the addressed device. Intervals of up to one second may occur between message characters and if this interval is more than one second, the receiving device will assume that an error has occurred and the message is ignored (the device resumes looking for the another start of message colon character).

ASCII Mode Message Frame

Start	Addr.		Function		Data	LRC		End	
1 char (colon)	2 char		2 char		N char	2 char		2 char (CRLF)	
3A	X	X	X	X	x1…xN	X	X	0D	0A

Note that each character noted above requires an ASCII code for 0-9 & A-F, and this requires 7 bits, plus 3 bits for the start bit, parity bit, and stop bit of the data frame. For example, even though the LRC is an 8 bit value, it effectively requires that 2 bytes be transmitted (two 7-bit bytes) in ASCII mode, one for each digit.

Note that some controllers may terminate the ASCII message after the LRC field with no CR LF attached (an interval of at least 1 second replaces the CR LF pair in this case).

Acromag Series 900MB modules do not support ASCII Mode of transmission and use the more efficient RTU Mode described below.

MODBUS MESSAGE FRAMING

RTU Mode Message Frames

RTU mode messages start with a silent interval of at least 3.5 character times implemented as a multiple of character times at the baud rate being used on the network (indicated as t1t2t3t4 below). The first field transmitted is the device address. The allowable characters transmitted for all fields are hexadecimal values 0-9, A-F.

A networked device continuously monitors the network, including the silent intervals, and when the first field is received (the address) after a silent interval of at least 3.5 character times, the device decodes it to determine if it is the addressed device. Following the last character transmitted, a similar silent interval of 3.5 character times marks the end of the message and a new message can begin after this interval. A typical message frame is shown below.

Character-Time: *the time it takes to transmit one character at the chosen baud rate. In RTU mode, there are 11 bits per character and this would be 11 bit-times. For example, at 9600 baud, 1 character-time is 11 bit times or 11bits/char* 1/9600bits/sec = 1146us/char.*

RTU Message Frame

Start	Addr.	Function	Data	CRC	End
t1t2t3t4	8 bits	8 bits	nx8 bits	16 bits	t1t2t3t4

The entire message must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame (not a continuous stream), the receiving device flushes the incomplete message and assumes the next byte will be the address field of a new message.

In similar fashion, if a new message begins earlier than 3.5 character times following a previous message, the receiving device assumes it is a continuation of the previous message. This will generate an error, as the value in the final CRC field will not be valid for the combined messages.

HOW CHARACTERS ARE TRANSMITTED SERIALLY

When messages are transmitted on Modbus serial networks, each character or byte is sent in the order of Least Significant Bit (LSB) to Most Significant Bit (MSB) as outlined below (left to right):

RTU Character Framing (No Parity)

Start	0	1	2	3	4	5	6	7	Stop	Stop

RTU Character Framing (With Parity)

Start	0	1	2	3	4	5	6	7	Parity	Stop

Note that an additional stop bit is transmitted to fill out the character frame for no parity. This is clearly defined in the Modbus standard, but often violated in industry. That is, some programmer's interpret "No Parity" as "No Bit", and fail to realize that the absence of parity is equivalent to adding another stop bit. Acromag Series 900MB modules were designed to accept messages with no parity and 1 or 2 stop bits so that this will not be an issue.

The character frame for ASCII Mode transmission is identical to RTU Mode, except that ASCII Mode uses only 7 data bits, versus 8 bits for RTU mode as shown above.

MODBUS ADDRESSES

The master device addresses a specific slave device by placing the 8-bit slave address in the address field of the message (RTU Mode). The address field of the message frame contains two characters (in ASCII mode), or 8 binary bits (in RTU Mode). Valid addresses are from 1-247. When the slave responds, it places its own address in this field of its response to let the master know which slave is responding. Address 0 is reserved for the broadcast address, which all slave devices on a network recognize. Some functions do not support the broadcast address. A slave device does not issue a response to a broadcast message.

All data addresses in Modbus messages are referenced to 0, with the first occurrence of a data item addressed as item number zero. Further, a function code field already specifies which register group it operates on (i.e. 0x, 1x, 3x, or 4x reference addresses). For example, holding register 40001 is addressed as register 0000 in the data address field of the message. The function code that operates on this register specifies a “holding register” operation and the “4xxxx” reference group is implied. Thus, holding register 40108 is actually addressed as register 006BH (107 decimal).

MODBUS FUNCTIONS

The function code field of the message frame will contain two characters (in ASCII mode), or 8 binary bits (in RTU Mode) that tell the slave what kind of action to take. Valid function codes are from 1-255, but not all codes will apply to a module and some codes are reserved for future use. The following table highlights the subset of standard Modbus functions supported by the Acromag 900MB modules (the reference register addresses that the function operates on are also indicated):

The functions below are used to access the registers outlined in the register map of the module for sending and receiving data. The Report Slave ID and Reset Slave commands do not operate on register map registers.

CODE	FUNCTION	REFERENCE
01 (01H)	Read Coil (Output) Status	0xxxx
03 (03H)	Read Holding Registers	4xxxx
04 (04H)	Read Input Registers	3xxxx
05 (05H)	Force Single Coil (Output)	0xxxx
06 (06H)	Preset Single Register	4xxxx
08 (08H)	Reset Slave	*Hidden*
15 (0FH)	Force Multiple Coils (Outputs)	0xxxx
16 (10H)	Preset Multiple Registers	4xxxx
17 (11H)	Report Slave ID	*Hidden*

When the slave device responds to the master, it uses the function code field to indicate either a normal (error-free) response, or that some kind of error has occurred (an exception response).

A normal response simply echoes the original function code of the query, while an exception response returns a code that is equivalent to the original function code with its most significant bit (msb) set to logic 1. For example, the Read Holding Registers command has the function code 0000 0011 (03H).

MODBUS FUNCTIONS

If the slave device takes the requested action without error, it returns the same code in its response. However, if an exception occurs, it returns 1000 0011 (83H) in the function code field and appends a unique code in the data field of the response message that tells the master device what kind of error occurred, or the reason for the exception (see Modbus Exceptions).

The master's application program must handle the exception response. It may choose to post subsequent retries of the original message, it may try sending a diagnostic query, or it may simply notify the operator of the exception error.

The following paragraphs describe some of the Modbus functions supported by the Acromag 900MB modules. To gain a better understanding of Modbus, please refer to your module's register map as you review this material.

Read Coil Status (01)

This command will read the ON/OFF status of discrete outputs or coils (0x reference addresses) in the slave. For Acromag 900MB modules, its response is equivalent to reading the on/off status of solid-state output relays or switches. Broadcast transmission is not supported.

The Read Coil Status query specifies the starting coil (output channel) and quantity of coils to be read. Coils correspond to the discrete solid-state relays of this transmitter and are addressed starting from 0 (up to 4 coils addressed as 0-3 for this model). The Read Coil Status in the response message is packed as one coil or channel per bit of the data field. For 900MB modules, the output status is indicated as 1 for ON (conducting current), and 0 for OFF (not conducting). The LSB of the first data byte corresponds to the status of the coil addressed in the query. The other coils follow sequentially, moving toward the high order end of the byte. Since this example has only 4 outputs, the remaining bits of the data byte will be set to zero toward the unused high order end of the byte.

Read Coil Status Example Query

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	1 (01)
Starting Address High Order	0 (00)
Starting Address Low Order	0 (00)
Number Of Points High Order	0 (00)
Number Of Points Low Order	4 (04)
CRC Error Check (LRC in ASCII Mode)	--

This example reads the output channel status of coils 0-3 at slave device 247.

Note that the leading character of the 0x reference address is implied by the function code and omitted from the address specified. In this example, the first address is 00001, referenced via 0000H, and corresponding to coil 0.

MODBUS FUNCTIONS

Read Coil Status Example Response

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	1 (01)
Byte Count	1 (01)
Data (Coils 3-0)	10 (0A)
CRC Error Check (LRC in ASCII Mode)	--

To summarize, the status of coils 3-0 is shown as the byte value 0A hex, or 00001010 binary. Coil 3 is the fifth bit from the left of this byte, and coil 0 is the LSB. The four remaining bits (toward the high-order end) are zero. Reading left to right, the output status of coils 3..0 is ON-OFF-ON-OFF.

Bin	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
Hex	**0**				**A**			
Coil	**NA**	**NA**	**NA**	**NA**	**3**	**2**	**1**	**0**

Read Holding Registers (03)

Reads the binary contents of holding registers (4x reference addresses) in the slave device. Broadcast transmission is not supported.

The Read Holding Registers query specifies the starting register and quantity of registers to be read. Note that registers are addressed starting at 0 (registers 1-16 addressed as 0-15). The Read Holding Registers response message is packed as two bytes per register, with the binary contents right-justified in each byte. For each register, the first byte contains the high order bits and the second byte the low order bits.

This example reads holding registers 40006…40008 (Channel 0 high limit value, low limit value, & deadband value) at slave device 247.

Read Holding Register Example Query

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	3 (03)
Starting Address High Order	0 (00)
Starting Address Low Order	5 (05)
Number Of Points High Order	0 (00)
Number Of Points Low Order	3 (03)
CRC Error Check (LRC in ASCII Mode)	--

Read Holding Register Example Response

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	3 (03)
Byte Count	6 (06)
Data High (Register 40006)	(3A)
Data Low (Register 40006)	75%=15000 (98)
Data High (Register 40007)	(13)
Data Low (Register 40007)	25%=5000 (88)
Data High (Register 40008)	(00)
Data Low (Register 40008)	1%=200 (C8)
CRC Error Check (LRC in ASCII Mode)	--

To summarize our example, the contents of register 40006 (2 bytes) is the channel 0 high limit of 75% (15000=3A98H). The contents of register 40007 (2 bytes) is the channel 0 low limit of 25% (5000=1388H). The contents of register 40008 is the channel 0 deadband value (2 bytes) of 1% (200=00C8H).

MODBUS FUNCTIONS

Read Input Registers (04)

This command will read the binary contents of input registers (3x reference addresses) in the slave device. Broadcast transmission is not supported.

The Read Input Registers query specifies the starting register and quantity of registers to be read. Note that registers are addressed starting at 0. That is, registers 1-16 are addressed as 0-15. The Read Input Registers response message is packed as two bytes per register, with the binary contents right-justified in each byte. For each register, the first byte contains the high order bits and the second byte the low order bits.

Read Input Registers Example Query

This example reads input registers 30003 & 30004 (Channel 0 input value and status) at slave device 247.

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	4 (04)
Starting Address High Order	0 (00)
Starting Address Low Order	2 (02)
Number Of Points High Order	0 (00)
Number Of Points Low Order	2 (02)
CRC Error Check (LRC in ASCII Mode)	--

Read Input Registers Example Response

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	4 (04)
Byte Count	4 (04)
Data High (Register 30003)	(3E)
Data Low (Register 30003)	80%=16000 (80)
Data High (Register 30004)	(00)
Data Low (Register 30004)	136 (88)
CRC Error Check (LRC in ASCII Mode)	--

To summarize our example, the contents of register 30003 (2 bytes) is the channel 1 input value of 80% (16000=3E80H). The contents of register 30004 (2 bytes) is the channel 0 status flags of 136 (0088H)—i.e. flagging high limit exceeded.

Force Single Coil (05)

Forces a single coil/output (0x reference address) ON or OFF. With broadcast transmission (address 0), it forces the same coil in all networked slaves.

The Force Single Coil query specifies the coil reference address to be forced, and the state to force it to. The ON/OFF state is indicated via a constant in the query data field. A value of FF00H forces the coil to be turned ON (i.e. the corresponding solid-state relay is turned ON or closed), and 0000H forces the coil to be turned OFF (i.e. the solid-state output relay is turned OFF or opened). All other values are invalid and will not affect the coil.

MODBUS FUNCTIONS

The following example forces discrete output 3 ON at slave device 247.

Coils are referenced starting at 0—up to 4 coils are addressed as 0-3 for our example and this corresponds to the discrete output channel number.

Force Single Coil Example Query And Response

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	5 (05)
Coil Address High Order	0 (00)
Coil Address Low Order	3 (03)
Force Data High Order	255 (FF)
Force Data Low Order	0 (00)
CRC Error Check (LRC in ASCII Mode)	--

The Force Single Coil response message is simply an echo (copy) of the query as shown above, but returned after executing the force coil command. No response is returned to broadcast queries from a master device.

Preset Single Register (06)

This command will preset a single holding register (4x reference addresses) to a specific value. Broadcast transmission is supported by this command and will act to preset the same register in all networked slaves.

The Preset Single Register query specifies the register reference address to be preset, and the preset value. Note that registers are addressed starting at 0--registers 1-16 are addressed as 0-15. The Preset Single Registers response message is an echo of the query, returned after the register contents have been preset.

This example writes a baud rate of 9600bps to holding register 40002 (Baud Rate) at slave device 247.

Preset Holding Register Example Query (And Response)

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	6 (06)
Register Address High Order	0 (00)
Register Address Low Order	1 (01)
Preset Data High Order	0 (00)
Preset Data Low Order	2 (02)
CRC Error Check (LRC in ASCII Mode)	--

The response message is simply an echo (copy) of the query as shown above, but is returned after the register contents have been preset. No response is returned to broadcast queries from a master device.

Force Multiple Coils (15)

Simultaneously forces a sequence of coils (0x reference addresses) either ON or OFF. Broadcast transmission is supported by this command and will act to force the same block of coils in all networked slaves.

The Force Multiple Coils query specifies the starting coil reference address to be forced, the number of coils, and the force data to be written in ascending order. The ON/OFF states are specified by the contents in the query data field. A logic 1 in a bit position of this field requests that the coil turn ON, while a logic 0 requests that the corresponding coil be turned OFF.

MODBUS FUNCTIONS

Unused bits in a data byte should be set to zero. Note that coils are referenced starting at 0—up to 4 coils are addressed as 0-3 for this example and this also corresponds to the discrete output channel number.

This example forces outputs 1 & 3 OFF, and 0 & 2 ON for coils 0-3 at slave device 247.

Force Multiple Coils Example Query

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	15 (0F)
Coil Address High Order	0 (00)
Coil Address Low Order	0 (00)
Number Of Coils High Order	0 (00)
Number Of Coils Low Order	4 (04)
Byte Count	01
Force Data High (First Byte)	5 (05)
Error Check (LRC or CRC)	--

Note that the leading character of the 0x reference address is implied by the function code and omitted from the address specified. In this example, the first address is 00001 corresponding to coil 0 and referenced via 0000H. Thus, in this example the data byte transmitted will address coils 3...0, with the least significant bit addressing the lowest coil in this set as follows (note that the four unused upper bits of the data byte are set to zero):

Bin	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
Hex	**0**				**5**			
Coil	**NA**	**NA**	**NA**	**NA**	**3**	**2**	**1**	**0**

Force Multiple Coils Example Response

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	15 (0F)
Coil Address High Order	0 (00)
Coil Address Low Order	0 (00)
Number Of Coils High Order	0 (00)
Number Of Coils Low Order	4 (04)
Error Check (LRC or CRC)	--

The Force Multiple Coils normal response message returns the slave address, function code, starting address, and the number of coils forced, after executing the force instruction. Note that it does not return the byte count or force value. No response is returned to broadcast queries from a master device.

Preset Multiple Registers (16)

Presets a block of holding registers (4x reference addresses) to specific values. Broadcast transmission is supported by this command and will act to preset the same block of registers in all networked slaves.

The Preset Multiple Registers query specifies the starting register reference address, the number of registers, and the data to be written in ascending order. Note that registers are addressed starting at 0--registers 1-16 are addressed as 0-15.

MODBUS FUNCTIONS

This example writes a new slave address of 200, a baud rate of 28800bps, and sets parity to even, by writing to holding registers 40001 through 40003 at slave device 247 (changes to slave address, baud rate, and parity will take effect following the next software or power-on reset).

Preset Multiple Registers Example Query

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	16 (10)
Starting Register High Order	0 (00)
Starting Register Low Order	0 (00)
Number Of Registers High Order	0 (00)
Number Of Registers Low Order	3 (03)
Preset Data High (First Register)	0 (00)
Preset Data Low (First Register)	200 (C8)
Preset Data High (Second Reg)	0 (00)
Preset Data Low (Second Reg)	5 (05)
Preset Data High (Third Reg)	0 (00)
Preset Data Low (Third Reg)	2 (02)
Error Check (LRC or CRC)	--

Preset Multiple Registers Example Response

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	16 (10)
Starting Register High Order	0 (00)
Starting Register Low Order	0 (00)
Number Of Registers High Order	0 (00)
Number Of Registers Low Order	3 (03)
Error Check (LRC or CRC)	--

The Preset Multiple Registers normal response message returns the slave address, function code, starting register reference, and the number of registers preset, after the register contents have been preset. Note that it does not echo the preset values. No response is returned to broadcast queries from a master device.

Report Slave ID (17)

This command returns the model, serial, and firmware number for the Acromag slave device, the status of the Run indicator, and any other information specific to the device. This command does not address Register Map registers and broadcast transmission is not supported.

The command query simply sends the slave address and function code with the error check (CRC) as shown here:

Report Slave ID Example Query

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	17 (11)
CRC Error Check (LRC in ASCII Mode)	--

MODBUS FUNCTIONS

Report Slave ID Example Response (Specific To Acromag 900MB)

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	17 (11)
Byte Count	26 (1A)
Acromag Slave Model ID	0 (00H) = 924MB-0900; 1 (01H) = 913MB-0900 2 (02H) = 914MB-0900; 3 (03H) = 917MB-0900 4 (04H) = 918MB-0900; 5 (05H) = 901MB-0900 6 (06H) = 902MB-0900; 7 (07H) = 903MB-0900 8 (08H) = 904MB-0900; 9 (09H) = 905MB-0900 10 (0AH) = 906MB-0900; 11 (0BH) = 932MB-0900 12 (0CH) = 934MB-0900; 13 (0DH) = 942MB-0900
Run Indicator Status (ON)	255 (FF)
Firmware ASCII Data String (Data Field)	"ACROMAG, 9300-043A, 942MB-0900," (41 43 52 4F 4D 41 47 2C 39 33 30 30 2D 30 34 33 41 2C 39 34 32 4D 42 2D 30 39 30 30 2CH)
Serial Number ASCII String (Unique Per Module)	Six Numbers + Revision "123456A" (31 32 33 34 35 36 41H)
CRC Error Check	--

Reset Slave (08)

This command is used to trigger a reset of the module and its effect is equivalent to a power-on reset of the module. Note that with respect to Acromag 900MB modules, changes to baud rate, slave address, and parity are initiated following reset.

The Reset Slave command uses sub-function 01 (Restart Communications) of the standard Modbus Diagnostics Command (08) to accomplish a reset and does not operate on a register map location. Broadcast transmission is not supported. Allow a few seconds following reset to re-initiate communication with a module.

Reset Slave Example Query & Echoed Response

Field Name	Example Value (Hex)
Slave Address	247 (F7)
Function Code	08 (08)
Sub-Function High Order Byte	0 (00)
Sub-Function Low Order Byte	1 (01)
Data Field High-Order Byte	0 (00)
Data Field Low Order Byte	0 (00)
CRC Error Check (LRC in ASCII Mode)	--

The Reset Slave query simply sends the slave address, function code, sub-function code, and data (data is ignored and simply echoed back), with error check (CRC) as shown here.

A Reset Slave response is simply an echoed acknowledge that is returned just before the reset is executed.

Acromag 900MB modules provide an alternate method of reset via a write to the Module Reset Register (See Register Map of particular unit) for Modbus software that does not support the Reset Slave function.

MODBUS DATA FIELD

The data field provides the slave with any additional information required by the slave to complete the action specified by the function code. The data is formed from a multiple of character bytes (a pair of ASCII characters in ASCII Mode), or a multiple of two hex digits in RTU mode, in range 00H-FFH. The data field typically includes register addresses, count values, and written data. For some messages, this field may not exist (has zero length), as not all messages require data.

If no error occurs, the data field of a response from a slave will return the requested data. If an error occurs, the data field returns an exception code (see Modbus Exceptions) that the master's application software can use to determine the next action to take.

Supported Data Types (Acromag 900MB Modules)

All I/O values are accessed via 16-bit Input Registers or 16-bit Holding Registers (see Register Map). Input registers contain information that is read-only. For example, the current input value read from a channel, or the states of a group of digital inputs. Holding registers contain read/write information that may be configuration data or output data. For example, the high limit value of an alarm function operating at an input, or an output value for an output channel.

I/O values of Acromag 900MB modules are represented by the following simple data types for temperature, percentage, and discrete on/off.

Summary Of Data Types Used By 900MB Modules

Data Types	Description
Count Value	A 16-bit signed integer value representing an A/D count, a DAC count, time value, or frequency with a range of –32768 to +32767.
Count Value	A 16-bit unsigned integer value representing an A/D count, a DAC count, time value, or frequency with a range of 0 to 65535.
Percentage	A 16-bit signed integer value with resolution of 0.005%/lsb. ±20000 is used to represent ±100%. For example, -100%, 0% and +100% are represented by decimal values –20000, 0, and 20000, respectively. The full range is –163.84% (-32768 decimal) to +163.835% (+32767 decimal).
Temperature	A 16-bit signed integer value with resolution of 0.1°C/lsb. For example, a value of 12059 is equivalent to 1205.9°C, a value of –187 equals –18.7°C. The maximum possible temperature range is –3276.8°C to +3276.7°C.
Discrete	A discrete value is generally indicated by a single bit of a 16-bit word. The bit number/position typically corresponds to the discrete channel number for this model. Unless otherwise defined for outputs, a 1 bit means the corresponding output is closed or ON, a 0 bit means the output is open or OFF. For inputs, a value of 1 means the input is in its high state (usually >> 0V), while a value of 0 specifies the input is in its low state (near 0V).

MODBUS ERROR CHECKING

Modbus networks employ two methods of error checking: parity checking of the data character frame (even, odd, or no parity), and frame checking within the message frame (Cyclical Redundancy Check in RTU Mode, or Longitudinal Redundancy Check in ASCII Mode).

Parity Checking

A Modbus device can be configured for even or odd parity, or for no parity checking. This determines how the parity bit of the character's data frame is set.

If even or odd parity checking is selected, the number of 1 bits in the data portion of each character frame is counted. Each character in RTU mode contains 8 bits. The parity bit will then be set to a 0 or a 1, to result in an even (even parity), or odd (odd parity) total number of 1 bits.

For example, if an RTU character frame contains the following eight data bits: 1100 0011, then since the total number of 1 bits is 4 (already an even number), the frame's parity bit will be 0 if even parity is selected. If odd parity is used, then the parity bit will be set to 1, making the total number of bits an odd number (five).

When a message is transmitted, the parity bit is calculated and applied to the data frame of each character transmitted (every 7 bits in ASCII Mode, or every 8 bits in RTU Mode). The receiving device counts the quantity of 1 bits in the data portion of the frame and sets an error flag if the count differs from that received. As such, parity checking can only detect an error if an odd number of bits are picked up or dropped off from a character frame during transmission. For example, if odd parity is employed and two 1 bits are dropped from a character, the result is still an odd count of 1 bits and no parity error is detected. Note that all devices on a Modbus network must use the same parity. If no parity checking is selected, then no parity bit is transmitted and no parity check is made. However, an additional stop bit is transmitted to fill out the character frame for the no parity selection.

LRC Longitudinal Redundancy Check (ASCII Mode Only)

In the ASCII transmission mode, the character frame includes an LRC field as the last field preceding the CRLF characters. This field contains two ASCII characters that represent the result of a longitudinal redundancy calculation for all the fields except the starting colon character and ending CR LF pair of characters.

The LRC field contains an 8-bit binary value that is calculated by the transmitting device and appended to the outgoing message. The receiving device also calculates an LRC value upon receipt of the message and compares its calculation to value the sent to it. If the two values are different, an error is generated.

The LRC value is calculated by first adding the successive 8-bit bytes of the message and discarding the carry bits to produce an 8-bit result (do not include the starting colon character or the ending CR LF characters). Then take the two's complement of this 8-bit result by subtracting it from FFH (one's complement), and adding 1 to get the two's complement. The 8-bit value that results (2 ASCII characters) is then transmitted in the message with the high-order character first, followed by the low-order character. For example, if the 8-bit LRC value is A1H (1010 0001), the LRC High character is A and this precedes the LRC low character of 1 in the data frame just prior to the CR LF characters.

MODBUS ERROR CHECKING

Acromag 900MB modules do not support the ASCII mode and do not use LRC error checking. These modules use the more efficient RTU mode and employ CRC error checking as described below.

CRC Error Checking (RTU Mode Only)

RTU Mode message frames include an error checking method that is based on a Cyclical Redundancy Check (CRC). The error checking field of a message frame contains a 16-bit value (two 8-bit bytes) that contain the result of a Cyclical Redundancy Check (CRC) calculation performed on the message contents.

The CRC value is calculated by the transmitting device and appended to the message as the last field in the message—the low order byte is appended first, followed by the high-order byte. Thus, the CRC high-order byte is the last byte to be sent in a message. The receiving device calculates a CRC during receipt of a message and compares the calculated value to that received in the CRC field. If the two values differ, an error will result.

The CRC is started by first preloading the 16-bit CRC register to all 1's. Successive 8-bit bytes of the message (only the 8-data bits in each character--no start, stop, or parity bits) are applied to the current contents of the register, and each 8-bit character is exclusive OR'ed with the register contents. The exclusive OR result is shifted in the direction of the least significant bit (lsb) of the CRC, with a zero placed into the most significant bit (msb). The lsb is then extracted and examined, if the lsb is a 1, the register is exclusive OR'ed with a preset fixed value. If the lsb is a 0, no exclusive OR takes place.

This process is repeated until 8 shifts have been performed. After the last (eighth) shift, the next 8-bit byte is exclusive OR'ed with the register's current contents, and the process repeats itself for 8 more shifts as described above. The final contents of the CRC register after all the message bytes have been applied is the CRC value.

MODBUS EXCEPTIONS

If an unsupported function code is sent to a module, then the exception code 01 (Illegal Function) will be returned in the data field of the response message. If a holding register is written with an invalid value, then exception code 03 (Illegal Data Value) will be returned in the response message.

Modbus Exception Codes

Code	Exception	Description
01	Illegal Function	The function code received in the query is not allowed or invalid.
02	Illegal Data Address	The data address received in the query is not an allowable address for the slave or is invalid.
03	Illegal Data Value	A value contained in the query data field is not an allowable value for the slave or is invalid.
04	Slave Device Failure	An unrecoverable error occurred while the slave was attempting to perform the requested action.

MODBUS EXCEPTIONS

Modbus Exception Codes…continued

Code	Exception	Description
05	Acknowledge	The slave has accepted the request and is processing it, but a long duration of time is required to do so. This response is returned to prevent a timeout error from occurring in the master.
06	Slave Device Busy	The slave is engaged in processing a long-duration program command. The master should retransmit the message later when the slave is free.
07	Negative Acknowledge	The slave cannot perform the program function received in the query. This code is returned for an unsuccessful programming request using function code 13 or 14 (codes not supported by this model). The master should request diagnostic information from the slave.
08	Memory Parity Error	The slave attempted to read extended memory, but detected a parity error in memory. The master can retry the request, but service may be required at the slave device.

In a normal response, the slave echoes the function code of the original query in the function field of the response. All function codes have their most-significant bit (msb) set to 0 (their values are below 80H). In an exception response, the slave sets the msb of the function code to 1 in the returned response (i.e. exactly 80H higher than normal) and returns the exception code in the data field. This is used by the master's application to recognize an exception response and to direct an examination of the data field for the applicable exception code.