

Modbus Message Formatting

The MODBUS protocol describes an industrial communications and distributed control system developed by Gould-Modicon to integrate PLC's, computers, terminals, and other monitoring, sensing, and control devices. MODBUS is a Master/Slave communications protocol, whereby one device, (the Master), controls all serial activity by selectively polling one or more slave devices. The protocol provides for one master device and up to 247 slave devices on a common line. Each device is assigned an address to distinguish it from all other connected devices.

Only the master initiates a transaction. Transactions are either a query/response type, (only a single slave is address), or a broadcast/no response type, (all slaves are addressed). A transaction comprises a single query and single response frame or a single broadcast frame.

Certain characteristics of the MODBUS protocol are fixed, such as the frame format, frame sequences, handling of communications errors and exception conditions, and the functions performed.

Other characteristics are user selectable. These include a choice of transmission media, baud rate, character parity, number of stop bits, and the transmission modes, (RTU or ASCII). The user selected parameters are set, (hardwired or programmed), at each station. These parameters cannot be changed while the system is running.

Modes of Transmission

The mode of transmission is the structure of the individual units of information within a message, and the numbering system used to transmit the data. Two modes of transmission are available for use in a MODBUS system. Both modes provide the same capabilities for communicating with PLC slaves; the mode is selected depending on the equipment used as a MODBUS Master. One mode must be used per MODBUS system; mixing of modes is not allowed. The modes are ASCII (American Standard Code for Information Interchange), and RTU, (Remote Terminal Unit.) The characteristics of the two transmission modes are defined below:

Characteristic	ASCII (7-bit)	RTU (8-bit)
Coding System	hexadecimal (uses ASCII printable characters (0-9, A-F)	8-bit binary
Number of bits per character:		
start bits	1	1
data bits (least significant first)	7	8
parity (optional)	1	1
	(1-bit sent for even or odd parity, no bits for no parity)	(1-bit sent for even or odd parity, no bits for no parity)
stop bits	1 or 2	1 or 2
Error Checking	LRC (Longitudinal Redundancy Check)	CRC (Cyclical Redundancy Check)

ASCII printable characters are easy to view when troubleshooting and this mode is suited to computer masters programmed in a high level language, such as FORTRAN, as well as PLC masters. RTU is suited to computer masters programmed in a machine language, as well as PLC masters.

In the RTU mode, data is sent in 8-bit binary characters. In the ASCII mode, each RTU character is first divided into two 4-bit parts, (high order and low order), and then represented by the hexadecimal equivalent. The ASCII characters representing the hexadecimal characters are used to construct the message. The

ASCII mode uses twice as many characters as the RTU mode, but decoding handling the ASCII data is easier. Additionally, in the RTU mode, message characters must be transmitted in a continuous stream. In the ASCII mode, breaks of up to one second can occur between characters to allow for a relatively slower master.

Error Detection

There are two types of errors which may occur in a communications system: transmission errors and programming errors. The MODBUS system has specific methods for dealing with either type of error.

Communications errors usually consist of a changed bit or bits within a message. The most frequent cause of communications errors is noise: unwanted electrical signals in a communications channel. These signals occur because of electrical interference from machinery, damage to the communications channel, impulse noise, (spikes), etc. Communications errors are detected by character framing, a parity check, and a redundancy check.

When the character framing, parity, or redundancy checks detect a communications error, processing of the message stops. A PLC slave will not act on or respond to the message. (The same occurs if a non-existent slave address is used.)

When a communications error occurs, the message is unreliable. The PLC slave cannot know for sure if this message was intended for it. So the CPU might be answering a message which was not its message to begin with. It is essential to program the MODBUS Master to assume a communications error has occurred if there is no response in a reasonable time. The length of this time depends upon the baud rate, type of message, and scan time of the PLC slave. Once this time is determined, the master may be programmed to automatically retransmit the message.

The MODBUS system provides several levels of error checking to assure the quality of the data transmission. To detect multibit errors where the parity has not changed, the system uses redundancy checks: Cyclical Redundancy Check, (CRC), for the RTU mode and Longitudinal Redundancy Check, (LRC), for the ASCII mode.

CRC-16 Cyclic Redundancy Check

The CRC-16 error check sequence is implemented as described in the following paragraphs.

The message, (data bits only, disregarding start/stop and parity bits), is considered as one continuous binary number whose most significant bit, (MSB), is transmitted first. The message is pre-multiplied by X^{16} , (shifted left 16 bits), then divided by $X^{16} + X^{15} + X^2 + 1$ expressed as a binary number (1100000000000101). The integer quotient digits are ignored and the 16-bit remainder (initialized to all ones at the start to avoid the case where all zeroes being an accepted message), is appended to the message, (MSB first), as the two CRC check bytes. The resulting message including the CRC, when divided by the same polynomial ($X^{16} + X^{15} + X^2 + 1$), at the receiver will give a zero remainder if no errors have occurred. (The receiving unit recalculates the CRC and compares it to the transmitted CRC). All arithmetic is performed modulo two, (no carries). An example of the CRC-16 error check for message HEX 0207, (address 2, function 7 or a status request to slave number 2) follows:

The device used to serialize the data for transmission will send the conventional LSB or right-most bit of each character first. In generating the CRC, the first bit transmitted is defined as the MSB of the dividend. For convenience then, and since there are no carries used in arithmetic, let's assume while computing the CRC that the MSB is on the right. To be consistent, the bit order of the generating polynomial must be reversed. The MSB of the polynomial is dropped since it affects only the quotient and not the remainder. This yields 1010 0000 0000 0001, (HEX A001).. Note that this reversal of the bit order will have no effect whatever on the interpretation or the bit order of characters external to the CRC calculations.

The step by step procedure to form the CRC-16 is as follows:

1. Load a 16-bit register with all 1's.
2. Exclusive OR the first 8-bit byte with the high order byte of the 16-bit register, putting the result in the 16-bit register.
3. Shift the 16-bit register one bit to the right.
- 4a. If the bit shifted out to the right is one, exclusive OR the generating polynomial 1010 0000 0000 0001 with the 16-bit register.
- 4b. If the bit shifted out to the right is zero; return to step 3.
5. Repeat steps 3 and 4 until 8 shifts have been performed.
6. Exclusive OR the next 8-bit byte with the 16-bit register.
7. Repeat step 3 through 6 until all bytes of the message have been exclusive OR'rd with the 16-bit register and shifted 8 times.
8. The contents of the 16-bit register are the 2 byte CRC error check and is added to the message most significant bits first.

	16-BIT REGISTER				MSB	Flag
(Exclusive OR)	1111	1111	1111	1111		
02				0000	0010	
Shift 1	1111	1111	1111	1111	1101	
Polynomial	0111	1111	1111	1111	1110	1
	1010	0000	0000	0000	0001	
Shift 2	1101	1111	1111	1111	1111	
Polynomial	0110	1111	1111	1111	1111	1
	1010	0000	0000	0000	0001	
Shift 3	1100	1111	1111	1111	1110	
Shift 4	0110	0111	1111	1111	1111	0
Polynomial	0011	0011	1111	1111	1111	1
	1010	0000	0000	0000	0001	
Shift 5	1001	0011	1111	1111	1110	
Shift 6	0100	1001	1111	1111	1111	0
Polynomial	0010	0100	1111	1111	1111	1
	1010	0000	0000	0000	0001	
Shift 7	1000	0100	1111	1111	1110	
Shift 8	0100	0010	0111	1111	1111	0
Polynomial	0010	0001	0011	1111	1111	1
	1010	0000	0000	0000	0001	
07	1000	0001	0011	1110	0000	
				0111		
	1000	0001	0011	1001		

Shift 1	0100	0000	1001	1100	1
Polynomial	1010	0000	0000	0001	
Shift 2	1110	0000	1001	1101	1
Polynomial	0111	0000	0100	1110	
	1010	0000	0000	0001	
Shift 3	1101	0000	0010	1111	1
Polynomml	0110	1000	0010	0111	
	1010	0000	0000	0001	
Shift 4	1100	1000	0010	0110	0
Shift 5	0110	0100	0001	0011	1
Polynomial	0011	0010	0000	1001	
	1010	0000	0000	0001	
Shift 6	1001	0010	0000	1000	0
Shift 7	0100	1001	0000	0100	0
Shift 8	0010	0100	1000	0010	0
	0001	0010	0100	0001	0

HEX 12

HEX 41

TRANSMITTED MESSAGE WITH CRC-16
(MESSAGE SHIFTED TO RIGHT TO TRANSMIT)

12 41 07 02

0001 0010 0100 0001 0000 0111 0000 0010

LRC (Longitudinal Redundancy Check)

The error check sequence for the ASCII mode is LRC. The error check is an 8-bit binary number represented and transmitted as two ASCII hexadecimal (hex) characters. The error check is produced by converting the hex characters to binary, adding the binary characters without wraparound carry, and two's complementing the result. At the received end the LRC is recalculated and compared to the sent LRC. The colon, CR, LF, and any imbedded non-ASCII hex characters are ignored in calculating the LRC.

Address	02		0000 0010
Function	01		0000 0001
Start Add H.O.	00		0000 0000
Start Add L.O.	00		0000 0000
Quantity of Pts	00		0000 0000
	08		0000 1000
		Sum	0000 1011
		1's complement	1111 0100
		+1	0000 0001
Error Check	F5	2's complement	1111 0101

MODBUS Message Types

ASCII Framing

Framing in ASCII Transmission mode is accomplished by the use of the unique colon, (:), character to indicate the beginning of frame and carriage return/line feed, (CRLF), to delineate end of frame. The line feed character also serves as a synchronizing character which indicates that the transmitting station is ready to receive an immediate reply.

BEGIN FRAME	ADDRESS	FUNCTION	DATA	ERROR CHECK	EOF	READY TO RECEIVE
:	2-CHAR 16- BIT	2-CHAR 16- BITS	N X 4-CHAR N X 16-BITS	2-CHAR 16-BITS	CR	LF

RTU Framing

Frame synchronization can be maintained in RTU transmission mode only by simulating a synchronous message. The receiving device monitors the elapsed time between receipt of characters. If three and one-half character times elapse without a new character or completion of the frame, then the device flushes the frame and assumes that the next byte received will be an address.

T1,T2,T3	ADDRESS	FUNCTION	DATA	CHECK	T1,T2,T3
	8-BITS	8-BITS	N X 8-BITS	16-BITS	

Address Field

The address field immediately follows the beginning of frame and consists of 8-bits, (RTU), or 2 characters, (ASCII). These bits indicate the user assigned address of the slave device that is to receive the message sent by the attached master.

Each slave must be assigned a unique address and only the addressed slave will respond to a query that contains its address. When the slave sends a response, the slave address informs the master which slave is communicating. In a broadcast message, an address of 0 is used. All slaves interpret this as an instruction to read and take action on the message, but not to issue a response message.

Function Field

The Function Code field tells the addressed slave what function to perform. MODBUS function codes are specifically designed for interacting with a PLC on the MODBUS industrial communications system. The high order bit in this field is set by the slave device to indicate an exception condition in the response message. If no exceptions exist, the high-order bit is maintained as zero in the response message.

The following table lists those functions supported by various WinTECH Software Applications:

CODE	MEANING	ACTION
01	READ COIL STATUS	Obtains current status, (ON/OFF), of a group of logic coils.
02	READ INPUT STATUS	Obtains current status, (ON/OFF), of a group of discrete inputs.
03	READ HOLDING REGISTER	Obtains current binary value in one or more holding registers.
04	READ INPUT REGISTER	Obtains current binary value in one or more input registers.
05	FORCE SINGLE COIL	Force logic coil to a state of ON or OFF.
06	PRESET SINGLE REGISTER	Place a specific binary value into a holding register.
15	WRITE MULTIPLE COILS	Force a group of logic coils to a defined state.
16	PRESET MULTIPLE REGISTERS	Place specific binary values into a group of holding registers.

Data Field

The data field contains information needed by the slave to perform the specific function or it contains data collected by the slave in response to a query. This information may be values, address references, or limits. For example, the function code tells the slave to read a holding register, and the data field is needed to indicate which register to start at and how many to read. The imbedded address and data information varies with the type and capacity of the PLC associated with the slave.

Error Check Field

This field allows the master and slave devices to check a message for errors in transmission. Sometimes, because of electrical noise or other interference, a message may be changed slightly while its on its way from one device to another. The error checking assures hat the slave or master does not react to messages that have changed during transmission. This increases the safety and the efficiency of the MODBUS system.

The error check field uses a Longitudinal Redundancy Check, (LRC), in the ASCII mode of transmission, and a CRC-16 check in the RTU mode.

Exception Responses

Programming or operation errors are those involving illegal data in a message, no response from the PLC to its interface unit, or difficulty in communicating with a slave. These errors result in an exception response from either the master computer software or the PLC slave, depending on the type of error. The exception response codes are listed below. When a PLC slave detects one of these errors, it sends a response message to the master consisting of the slave address, function code, error code, and error check fields. To indicate that the response is a notification of an error, the high-order bit of the function code is set to one.

CODE	NAME	MEANING
01	ILLEGAL FUNCTION	The message function received is not an allowable action for the addressed slave.
02	ILLEGAL DATA ADDRESS	The address referenced in the data field is not an allowable address for the addressed slave device.
03	ILLEGAL DATA VALUE	The value referenced in the data field is not allowable in the addressed slave location.
04	FAILURE IN ASSOCIATED DEVICE	The slave's PC has failed to respond to a message or an abortive error occurred.
05	ACKNOWLEDGE	The slave PLC has accepted and is processing the long duration program command.
06	BUSY, REJECTED MESSAGE	The message was received without error, but the PLC is engaged in processing a long duration program command.
07	NAK-NEGATIVE ACKNOWLEDGMENT	The PROGRAM function just requested could not be performed.

READ OUTPUT STATUS (FUNCTION CODE 01)

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

The addressing allows up to 2000 coils to be obtained at each request; however, the specific slave device may have restrictions that lower the maximum quantity. The coils are numbered from zero; (coil number 1 is address 0000, coil number 2 is address 0001, etc.)

The following is an example of a message to Read Output Status Coils 20-56 from slave device number 17.

ADDR	FUNC	DATA START PT HO	DATA START PT LO	DATA # OF PTS HO	DATA # OF PTS LO	ERROR CHECK FIELD
11	01	00	13	00	25	B6

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

ADDR	FUNC	BYTE COUNT	DATA COIL STATUS 20-27	DATA COIL STATUS 28-35	DATA COIL STATUS 36-43	DATA COIL STATUS 44-51	DATA COIL STATUS 52-56	ERROR CHECK FIELD
11	01	05	CD	6B	B2	0E	1B	D6

The status of coils 20-27 is shown as CD(HEX) = 1100 1101(Binary). Reading left to right, this shows that coils 27,26,23,22, and 20 are all on. The other coil data bytes are decoded similarly.

READ INPUT STATUS (FUNCTION CODE 02)

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

The following is an example of a message to Read Input Status Coils 10197-10218 from slave device number 17.

ADDR	FUNC	DATA START PT HO	DATA START PT LO	DATA # OF PTS HO	DATA # OF PTS LO	ERROR CHECK FIELD
11	02	00	C4	00	16	13

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

ADDR	FUNC	BYTE COUNT	DATA DISCRETE INPUT 10197-10204	DATA DISCRETE INPUT 10205-10212	DATA DISCRETE INPUT 10213-10218	ERROR CHECK FIELD	
11	02	03	AC	DB	35	2E	LRC

The status of inputs 10197-10204 is shown as AC (HEX) = 1010 1100 (Binary). Reading left to right, this shows that inputs 10204, 10202, 10200 and 10099 are all on. The other input data bytes are decoded similarly.

READ OUTPUT REGISTERS (FUNCTION CODE 03)

Read Output Registers allows the user to obtain the binary contents of holding registers in the addressed slave.

These registers can store the numerical values of associated timers and counters which can be driven to external devices.

The addressing allows up to 125 registers to be obtained at each request; however, the specified slave device may have restrictions that lower this maximum quantity. The registers are numbered from zero, broadcast mode is not allowed.

The following example reads registers 40108 through 40110 from slave number 17.

ADDR	FUNC	DATA START PT HO	DATA START PT LO	DATA # OF REGS HO	DATA # OF REGS LO	ERROR CHECK FIELD
11	03	00	6B	00	03	7E

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

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

ADDR	FUNC	BYTE COUNT	DATA OUTPUT REG H.O. 40108	DATA OUTPUT REG L.O. 40108	DATA OUTPUT REG H.O. 40109	DATA OUTPUT REG L.O. 40109	DATA OUTPUT REG H.O. 40110	DATA OUTPUT REG L.O. 40110	ERROR CHECK FIELD
11	03	06	02	2B	00	00	00	64	55

READ INPUT REGISTERS (FUNCTION CODE 04)

Function Code 04 obtains the contents of the controllers input registers. These locations receive their values from devices connected to the I/O structure and can only be referenced, not altered from within the controller nor via MODBUS.

The example below requests the contents of register 30009 in slave number 17.

ADDR	FUNC	DATA START PT HO	DATA START PT LO	DATA # OF REGS HO	DATA # OF REGS LO	ERROR CHECK FIELD
11	04	00	08	00	01	E2

In the response message, the contents of register 30009 is decimal value 0.

ADDR	FUNC	BYTE COUNT	DATA INPUT REG HO 30009	DATA INPUT REG LO 30009	ERROR CHECK FIELD
11	04	02	00	00	E9

FORCE SINGLE COIL (FUNCTION CODE 05)

This message forces a single coil either On or OFF. Any coil that exists within the controller can be forced to either state, (ON or OFF). Coils are numbered from zero (i.e. coil 1 is address 0000, coil 2 is address 0001, etc.). The data value 65,280, (FF00 HEX) will set the coil ON and the value zero will turn it off. All other values are illegal and will not effect the coil. The use of slave address 00, (Broadcast mode), will force all attached slaves to modify the desired coil.

The example below requests slave number 17 to turn coil number 0173 ON.

ADDR	FUNC	DATA COIL HO	DATA COIL LO	DATA # ON/OFF	DATA	ERROR CHECK FIELD
11	05	00	AC	FF	00	3F

The normal response to the command request is to retransmit the message as received, after the coil state has been altered.

ADDR	FUNC	DATA COIL HO	DATA COIL LO	DATA # ON/OFF	DATA	ERROR CHECK FIELD
11	05	00	AC	FF	00	3F

PRESET SINGLE REGISTER (FUNCTION CODE 06)

Function 06 allows the user to modify the contents of a holding register. Any holding register that exists within the controller can have its contents changed by this message. The values are provided in binary up to the maximum capacity of the controller. Unused high-order bits must be set to zero. When used with slave address 00, all slave controllers will load the specified register with the contents specified.

ADDR	FUNC	DATA REG HO	DATA REG LO	DATA VALUE HO	DATA VALUE LO	ERROR CHECK FIELD
11	06	00	87	03	9E	C1

The normal response to a preset single register request is to retransmit the query message after the register has been altered.

ADDR	FUNC	DATA REG HO	DATA REG LO	DATA VALUE HO	DATA VALUE LO	ERROR CHECK FIELD
11	06	00	87	03	9E	C1

FORCE MULTIPLE COILS (FUNCTION CODE 15)

Function 15 allows the user to modify the contents of a group of consecutively addressed coils. The following example forces 10 coils starting at address 20, (13 HEX). The two data fields, CD = 1100 1101 and 00 = 0000 0000, indicate that coils 27, 26, 23, 22 and 20 are to be forced on.

ADDR	FUN C	H.O. ADDR	L.O. ADDR	QUANTITY		BYTE CNT	DATA COIL STATUS	DATA COIL STATUS	ERROR CHECK FIELD
11	0F	00	13	00	0A	02	CD	00	F4

The normal response to a FORCE MULTIPLE COILS request is to echo the slave address, function code, starting address, and quantity of coils set.

ADDR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY		ERROR CHECK FIELD
11	0F	00	13	00	0A	C3

PRESET MULTIPLE REGISTERS (FUNCTION CODE 16)

Holding registers existing within the controller can have their contents changed via function code 16. Sixteen bits of data for each register is contained within the message.

ADDR	FUN C	H.O. ADDR	L.O. ADDR	QUANTITY		BYTE CNT	H.O. DATA	L.O. DATA	etc.	ERROR CHECK FIELD
11	10	00	87	00	02	04	00	0A		??

The normal response to a PRESET MULTIPLE REGISTERS request is to echo the slave address, function code, starting address, and quantity of registers set.

ADDR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY		ERROR CHECK FIELD
11	10	00	87	00	02	56