# Analysis and Testing of Modbus on Serial and TCP

**Sowmiya G, B.Parimala Devi**

***Abstract*— Modbus is a serial communication protocol originally published by Modicon (now Schneider Electric)in 1979 for use with its programmable logic controllers (PLC's).Simple and robust, it has since become a de-facto standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices. Modbus enables communication between many(approximately 240) devices connected to the same network, for example a system that measures temperature and humidity and communicates the results to a computer. Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many of the data types are named from its use in driving relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or a contact. Modbus is a widely used in industry for a long time and slow-control devices in accelerator control system recently. Modbus protocol over Ethernet has advantages for non real-time applications due to its maturity.**

***Index Terms*— Modbus, Serial, TCP.**

## I. Introduction

The Modbus protocol provides the internal standard that the Modicon controllers use for parsing messages. During communications on a Modbus network, the protocol determines how each controller will know its device address, recognize a message addressed to it, determine the kind of action to be taken, and extract any data or other information contained in the message. If a reply is required, the controller will construct the reply message and send it using Modbus protocol.

On other networks, messages containing Modbus protocol are imbedded into the frame or packet structure that is used on the network. For example, Modicon network controllers for Modbus Plus or MAP, with associated application software libraries and drivers, provide conversion between the imbedded Modbus message protocol and the specific framing protocols those networks use to communicate between their node devices. This conversion also extends to resolving node addresses, routing paths, and error-checking methods specific to each kind of network. For example, Modbus device addresses contained in the Modbus protocol will be converted into node addresses prior to transmission of the messages. Error-checking fields will also be applied to message packets, consistent with each network's protocol.

## II. TRANSACTIONS ON MODBUS NETWORKS

Standard Modbus ports on Modicon controllers use an RS-232C compatible serial interface that defines connector pinouts, cabling, signal levels, transmission baud rates, and parity checking. Controllers can be networked directly or via modems. Controllers communicate using a master-slave technique, in which only one device (the master) can initiate transactions (queries). The other devices (the slaves) respond by supplying the requested data to the master, or by taking the action requested in the query. Typical master devices include host processors and programming panels. Typical slaves include programmable Modbus Protocol controllers. The master can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a message (response) to queries that are addressed to them individually. Responses are not returned to broadcast queries from the master. The Modbus protocol establishes the format for the master's query by placing into it the device(or broadcast) address, a function code defining the requested action, any data to be sent, and an error-checking field. The slave's response message is also constructed using Modbus protocol. It contains fields confirming the action taken, any data to be returned, and an error-checking field. If an error occurred in receipt of the message, or if the slave is unable to perform the requested action, the slave will construct an error message and send it as its response.

Table-1.Modbus over Serial Line Uses Three-Layer Model

| Layer | ISO/OSI Function | Modbus Function |
|---|---|---|
| 7 | Application | Modbus Application Protocol |
| 3-6 | Various | Null |
| 2 | Data-Link | Modbus Serial Line Protocol |
| 1 | Physical | EIA-232C or EIA-485 |

## III. SERIAL TRANSMISSION MODES OF MODBUS NETWORKS

Instead of traditional Seven-Layer ISO Open system Interconnection Reference Model, the Modbus over serial line is collapsed to three layers as shown in Table-1.At the Top is the Application Layer which is called as Modbus Application Protocol or simply Modbus Protocol. Layers 3-6 are not used instead, the model relies on the application layer to ensure end-to-end delivery of the message. The data link(Layer-2) is occupied by the Modbus Serial Line Protocol. Finally the Physical Layer allows for either the EIA-232C or EIA-485 Implementation. With only three layers Modbus Serial Line is easier to understand than other Industrial Protocols.

**Manuscript received Feb. 20, 2014.**

**Sowmiya G**, M.E (Pursuing) from Kongu Engineering College, Erode, India, 07676076673.

**B.Parimala Devi**, M.E (Pursuing) from Kongu Engineering College,Erode,India,8012432580.

### A.DATA LINK LAYER

Referring to Fig.1, note that the PDU consists of four elements. In the Middle is the Modbus PDU consisting of a function code and data. Most Modbus Implementations only use a subset of all the available function codes. The data structure may change depending upon the function code. Wrapping the Modbus PDU is an Error Check Field and Address field. The Address Field only contains slave address or the broadcast address. The Master Address is not required or not referenced since this is a Master/Slave protocol with commands originating from a unique master.

The Actual Framing of Modbus over Serial Line Messages depends on whether ASCII or RTU Transmission mode is used.

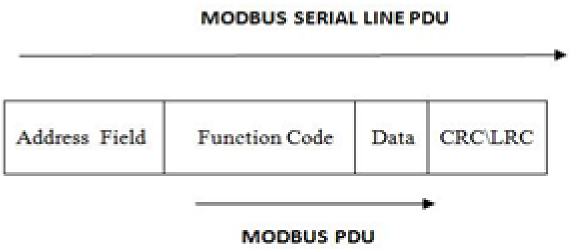


Fig.1.A Slave Address Field and Error Check Wrap around a Modbus PDU It is a very compact frame with only one byte reserved for the slave or broadcast address, one byte for function code and two bytes for CRC error check. A single byte carries the function code.

### B.PHYSICAL LAYER

The original Modbus called for a point-to-point EIA-232c link between a host computer and PLC.The EIA-485 standard supporting upto 32 devices over a common bus. This can be implemented either using two wires or four wire cabling configuration. With any of the serial line implementations, a wide range of baud rates from 1.25 kbps to 115 kbps are allowed, but all implementations must at least support 9.6 kbps and 19.2 kbps.The default rate is 19.2 kbps.

## IV. MODBUS OVER SERIAL LINE

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. Standard MODBUS networks employ one of two types of transmission modes:

1. ASCII Mode
2. RTU Mode.

The mode of transmission is usually selected along with other serial port communication parameters (baud rate, parity, etc.) as part of the device configuration.

### A.ASCII TRANSMISSION MODE

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.

### B.RTU TRANSMISSION MODE

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.

| Slave Address | Function Code | Data | CRC |
|---|---|---|---|
| 1 byte | 1 byte | 0 upto 252 bytes | 2 bytes |

Fig.2.RTU Framing is more condensed than ASCII Framing

## V. MODBUS MESSAGE FRAMING

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. A MODBUS message is placed in a message frame by the transmitting device. 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 character.

### A.ASCII MODE MESSAGE FRAME

ASCII mode message starts with 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. 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.

### B.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. 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. 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.

## VI. MODBUS/TCP

MODBUS/TCP is a communication protocol designed to allow industrial equipment such as Programmable Logic Controllers, computers, operator panels, motors, sensors, and other types of physical input/output devices to communicate over a network. Modbus/TCP was invented by Modicon/Group Schneider and is today is one of the most popular protocols embedded inside the TCP/IP frames of Ethernet. Modbus/TCP basically embeds a Modbus frame into a TCP frame in a simple manner. This is a connection-oriented transaction, which means every query expects a response. This query/response technique fits well with the master/slave nature of Modbus, adding to the deterministic advantage that Switched Ethernet offers industrial users. The use of OPEN Modbus within the TCP frame provides a totally scalable solution from ten nodes to ten thousand nodes without the risk of compromise that other multicast techniques would give.

Table-2.MODBUS TCP Uses a Five-Layer Internet Model
MODBUS TCP/IP has became an industry de facto standard because of its openness, simplicity, low cost development, and minimum hardware required to support it.

It is used to exchange information between devices, monitor and program them. It is also used to manage distributed I/Os,

| Layer | ISO/OSI Function | Modbus Function |
|---|---|---|
| 5,6,7 | Application | Modbus Application Protocol |
| 4 | Transport | Modbus Transport Protocol |
| 3 | Network | Internet Protocol |
| 2 | Data Link | IEEE 802.3 |
| 1 | Physical | IEEE 802.3 |

being the preferred protocol by the manufacturers of this type of devices. MODBUS TCP/IP uses TCP/IP and Ethernet to carry the MODBUS messaging structure. MODBUS/TCP requires a license but all specifications are public and open so there is no royalty paid for this license. Making use of TCP/IP also offers the use of embedded Web pages to make life even more user friendly! Simply 'surf' your plant intranet for the information you need by using your web browser.

## VII. THE QUERY-RESPONSE CYCLE

### A.THE QUERY

The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain any additional information that the slave will need to perform the function. For example, function code 03 will query the slave to read holding registers and respond with their contents. The data field must contain the information telling the slave which register to start at and how many registers to read. The error check field provides a method for the slave to validate the integrity of the message contents.

### B.THE RESPONSE

If the slave makes a normal response, the function code in the response is an echo of the function code in the query. The data bytes contain the data collected by the slave, such as register values or status.

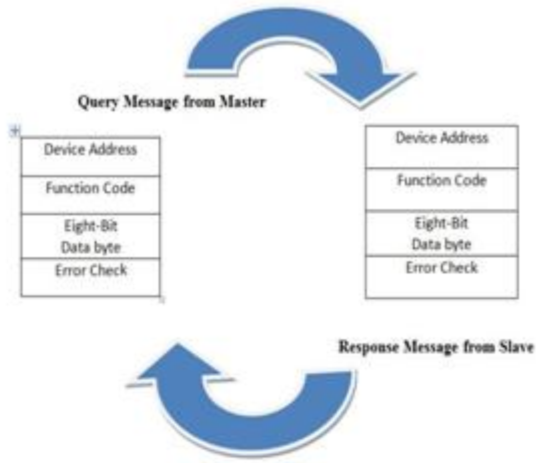


Fig-3.Query-Response Cycle

If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that describes the error. The error check field allows the master to confirm that the message contents are valid.

## VIII. ERROR CHECKING METHODS

Standard Modbus serial networks use two kinds of error checking. Parity checking (even or odd) can be optionally applied to each character. Frame checking (LRC or CRC) is applied to the entire message. Both the character check and message frame check are generated in the master device and applied to the message contents before transmission. The slave device checks each character and the entire message frame during receipt. The master is configured by the user to wait for a predetermined timeout interval before aborting the transaction. This interval is set to be long enough for any slave to respond normally. If the slave detects a transmission error, the message will not be acted upon. The slave will not construct a response to the master. Thus the timeout will expire and allow the master's program to handle the error.

Note that a message addressed to a nonexistent slave device will also cause a timeout. Other networks such as MAP or Modbus Plus use frame checking at a level above the Modbus contents of the message. On those networks, the Modbus message LRC or CRC check field does not apply. In the case of a transmission error, the communication protocols specific to those networks notify the originating device that an error has occurred, and allow it to retry or abort according to how it has been setup. If the message is delivered, but the slave device cannot respond, a timeout error can occur which can be detected by the master's program.

### A.PARITY CHECKING

Users can configure controllers for Even or Odd Parity checking, or for No Parity checking. This will determine how the parity bit will be set in each character. If either Even or Odd Parity is specified, the quantity of 1 bits will be counted in the data portion of each character (seven data bits for ASCII mode, or eight for RTU). The parity bit will then be set to a 0 or 1 to result in an Even or Odd total of 1 bit.

For example, these eight data bits are contained in an RTU character frame:

1100 0101

The total quantity of 1 bit in the frame is four. If Even Parity is used, the frame's parity bit will be a 0, making the total quantity of 1 bits still an even number (four). If Odd Parity is used, the parity bit will be a 1, making an odd quantity (five).

### B.LRC CHECKING

In ASCII mode, messages include an error–checking field that is based on a Longitudinal Redundancy Check (LRC) method. The LRC field checks the contents of the message, exclusive of the beginning 'colon' and ending CRLF pair. It is applied regardless of any parity check method used for the individual characters of the message. The LRC field is one byte, containing an 8–bit binary value. The LRC value is calculated by the transmitting device, which appends the LRC to the message. The receiving device calculates an LRC during receipt of the message, and compares the calculated value to the actual value it received in the LRC field. If the two values are not equal, an error results. The LRC is calculated by adding together successive 8–bit bytes of the message, discarding any carries, and then two's complementing the result. It is performed on the ASCII message field contents excluding the 'colon' character that begins the message, and excluding the CRLF pair at the end of the message. In ladder logic, the CKSM function calculates a LRC from the message contents.

### C.CRC CHECKING

In RTU mode, messages include an error–checking field that is based on a Cyclical Redundancy Check (CRC) method. The CRC field checks the contents of the entire message. It is applied regardless of any parity check method used for the individual characters of the message. The CRC field is two bytes, containing a 16–bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results. The CRC is started by first preloading a 16–bit register to all 1's. Then a process begins of applying successive 8–bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit, do not apply to the CRC. During generation of the CRC, each 8–bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place. This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8–bit byte is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the bytes of the message have been applied, is the CRC value. When the CRC is appended to the message, the low-order byte is appended first, followed by the high-order byte. In ladder logic, the CKSM function calculates a CRC from the message contents.

### REFERENCES


[1]Modbus Application Specification protocol V1.1b, http://www.modbus -IDA.org, December 28, 2006.

[2] Modbus over Specification and Implementation Guide V1.02, http://www.modbus -IDA.org, December 20, 2006.

[3] Modbus Messaging on TCP and Implementation Guide V1.0b, http://www.modbus -IDA.org, October 24, 2006.

[4] Modbus Protocol Reference Guide Rev.J, http://www.modbus -IDA.org, June 1996.


**Sowmiya G**, Pursuing M.E (Embedded Systems) in Kongu Engineering College. Completed B.E (ECE) in Excel College of Engineering.

**B.Parimala Devi**, Pursuing M.E (Embedded Systems) in Kongu Engineering College. Completed B.E (ECE) in Maharaja Engineering College for Women.