An Introduction to Modbus RTU Addressing, Function Codes, and Modbus RTU Networking Overview

Let’s Break Down Modbus RTU

Modbus RTU is an open serial protocol derived from the Master/Slave architecture originally developed by Modicon (now Schneider Electric). It is a widely accepted serial level protocol due to its ease of use and reliability. Modbus RTU is widely used within Building Management Systems (BMS) and Industrial Automation Systems (IAS).

Modbus RTU messages are a simple 16-bit structure with a CRC (Cyclic-Redundant Checksum). The simplicity of these messages is to ensure reliability. Due to this simplicity, the basic 16-bit Modbus RTU register structure can be used to pack in floating point, tables, ASCII text, queues, and other unrelated data.

This protocol primarily uses an RS-232 or RS-485 serial interfaces for communications and is supported by almost every commercial SCADA, HMI, OPC Server and data acquisition software program in the marketplace. This makes it very easy to integrate Modbus compatible equipment into new or existing monitoring and control applications.

A Little Modbus RTU History

You might call the Modbus protocol the grandfather of industrial networking. It truly is as old as the hills and has the whiskers to prove it. In today’s age of Internet connectivity and Web Services, Modbus’ unconnected message and simple request-response communication structure are almost quaint. Almost as old as the first Programmable Logic Controller, the Modicon 084, which in those days was called a PC for Programmable Controller.

Modbus RTU is an open standard, meaning that manufacturers can build it into their equipment without having to pay royalties. It is the most pervasive communications protocol in industrial automation and is now the most commonly available means of connecting industrial electronic devices.

Modbus is used widely by many manufacturers throughout many industries. Modbus is typically used to transmit data from control instrumentation to a logic controller or a system for archiving data. In building automation, for example, temperature and humidity are often communicated to a computer for long term storage. Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems.

Why is Modbus so Popular?

Simplicity is the reason Modbus is so widespread. It also didn’t hurt that Modbus was created by one of the largest PLC manufacturers at that time and made it open and widely available. Modbus also requires very little in the way of processor code space or RAM. While this isn’t as
important today given the powerful processors and technology available to us, it was very important in the early years of industrial automation when processors used 8-bit technology and resources like RAM and ROM were extremely expensive and scarce.

Message checking is another reason why Modbus has been so popular. CRC and LRC checking means that transmission errors are checked to 99% accuracy.

**Modbus Communication**

The Modbus RTU protocol uses a Master/Slave technique to communicate between devices. Meaning, any application that utilizes the Modbus RTU protocol will have a Modbus Master and at least one Modbus Slave. A Modbus Master is typically a host supervisory computer running software that will communicate with one or more Modbus Slave devices.

Modbus enables Master/Slave communication between devices connected through buses or networks. On the OSI model, Modbus is positioned at level 7. Modbus is intended to be a request/reply protocol and delivers services specified by function codes. The function codes of Modbus are elements of Modbus’ request/reply PDUs (Protocol Data Unit).

In order to build the Modbus application data unit, the client must initiate a Modbus transaction. It is the function which informs the server as to which type of action to perform. The format of a request initiated by a Master is established by the Modbus application protocol. The function code field is then coded into one byte. Only codes within the range of 1 through 255 are considered valid, with 128-255 being reserved for exception responses. When the Master sends a message to the Slave, it is the function code field which informs the server of what type of action to perform.

To define multiple actions, some functions will have sub-function codes added to them. For instance, the Master can read the ON/OFF states of a group of discreet outputs or inputs. It could also read/write the data contents of a group of MODBUS registers. When the Master receives the Slave response, the function code field is used by the Slave to indicate either an error-free response or an exception response. The Slave echoes to the request of the initial function code in the case of a normal response.
Modbus RTU Data Representation

Like everything else about Modbus, the data representation is simple. In fact, data is represented more simply in Modbus than in any other industrial protocol you’ll ever find. The Bit of least importance is sent and received first. All devices within the network must interpret each transmitted byte analogously in this manner.

There are no methods for automated recognition of baud rates. The same baud rate must be utilized by the Slave(s) and Master connected to the bus. No specific baud rate is specified by the Modbus: typical baud rates are 9600 or 19200.

A transmitted Byte is coded as 8 Bit binary value, hexadecimal 0 - 9 and A - F. The least significant Bit is sent and received first.

There are only two data types in Modbus: coils and registers. Coils are simply single bits. The bits can be ON (1) or they can be OFF (0). Some coils represent inputs, meaning they contain the status of some physical discrete input. Or they represent outputs, meaning that they hold the state of some physical discrete output signal. Registers are simply 16-bit unsigned register data. Registers can have a value from 0 to 65535 (0 to FFFF hexadecimal). There is no representation for negative values, no representation for values greater than 65535, and no representation for real data like 200.125.

Registers are grouped into Input Registers and Holding Registers. Like Input Coils, Input Registers report the state of some external input as a value between 0 and 65535. The original intent of an Input Register was to reflect the value of some analog input. It is a digital representation of an analog signal like a voltage or a current. Most Modbus devices today are not I/O devices, and Input Registers simply function identically to Holding Registers.

Holding Registers were originally designed as temporary program storage for devices like Modbus controllers. Today, Holding Registers function as data storage for devices.

Modbus RTU packets are only intended to send data; they do not have the capability to send parameters such as point name, resolution, units, etc. If the ability to send such parameters is needed, one should investigate a BACnet, EtherNet/IP, or other modern protocols.

Modbus RTU Address Requirements and Station Identification

Standard Modbus RTU node addresses are 1-255, with 0 being reserved for broadcast messages and write only. However, the 0 address is rarely used since there is no confirmation that the message was properly received at the slave node. This doesn’t have much affect if your physical layer is RS-232 as only one node can be implemented anyway. RS-485 limits the number of nodes to 32, though some drivers will allow you to extend the amount.
Serial Modbus Slave devices are identified by a station number which precedes the general message structure. Generally, up to 32 stations are supported as that is the limit imposed by most RS485 serial drivers. There is no software limit to the number of stations that could be supported. Valid Slave addresses are assigned in the range of 1 to 255 with station number 0 reserved for broadcast messages, messages processed by all Slaves.

**Transport Layers**

There are several standard transports used to move Modbus protocol messages: RS232 and RS485. You can use others, but these are the common ones.

RS485 is a successor to RS232. It works in a similar fashion regarding the synchronizing bits that synchronize the transfer of bits from a sending station to a receiving station. There are, however, two defining characteristics that make RS485 different from RS232. The first is the ability to drive multiple destinations. RS485 transmitters can electrically signal up to 32 destination devices. That makes RS485 the preferred way to serially transport Modbus messages.

The other defining characteristic of RS485 is enhanced noise immunity. RS485 does not use the electrical common as the reference for its electrical signal. Instead, RS485 uses a pair of wires and drives a signal by setting a voltage potential across the pair. By doing that, any environmental electrical noise affects both wires equally and the potential across the two wires isn’t changed.

**Modbus RTU Data Encoding**

An encoding mechanism describes how bit patterns are formed from the control and data values that are encoded into the packet. Both the sender and the receiver must use the same encoding to correctly understand the contents of the data. There are two mechanisms for encoding Modbus messages: ASCII and RTU.

RTU encoding is the much more common encoding mechanism used on Modbus. RTU simply means that values are encoded as standard big-endian binary. That means that in the case of 16-bit values, the Most Significant Byte (MSB) is encoded prior to the Least Significant byte (LSB). An 8-bit value like decimal 41 (29 hex) is encoded simply as 0010 1001. Whereas a 16-bit value like decimal 300 (12C hex) is encoded as 0000 0001 0010 1100. The MSB of 01 is encoded and transmitted prior to the LSB of 2C.
MODBUS RTU Memory Map

<table>
<thead>
<tr>
<th>Modbus RTU Data Type</th>
<th>Common name</th>
<th>Starting address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modbus Coils</td>
<td>Bits, binary values, flags</td>
<td>00001</td>
</tr>
<tr>
<td>Digital Inputs</td>
<td>Binary inputs</td>
<td>10001</td>
</tr>
<tr>
<td>Analog Inputs</td>
<td>Binary inputs</td>
<td>30001</td>
</tr>
<tr>
<td>Modbus Registers</td>
<td>Analog values, variables</td>
<td>40001</td>
</tr>
</tbody>
</table>

The Difference Between Modbus RTU and Modbus TCP

The most basic difference between Modbus RTU and Modbus TCP (Also known as Modbus IP, Modbus EtherNet, and Modbus TCP/IP) is that Modbus TCP runs on an Ethernet physical layer and Modbus RTU is a serial level protocol. Modbus TCP also uses a 6-byte header to allow routing.

A Modbus RTU Master is a single Master bus. It sends a message to an RTU slave device and gets an answer. Modbus RTU is limited to a single master. Only one set of signals can be on the RS485 link at any one time. Either the single RTU Master is transmitting or one of the RTU Client devices is transmitting.

With the introduction of Modbus TCP, everything was simplified and easier. With Modbus TCP, controllers can much more efficiently use the bandwidth on Ethernet to be the Master to hundreds of Modbus TCP devices. Modbus TCP allows for multiple Clients. Where RS485 had an electrical limitation of 32 devices, Ethernet is unlimited. Operating RAM is the only practical limitation. With Modbus TCP, there is the ability for a network designer to use multiple Clients/Masters if they so choose.

With Modbus TCP (Ethernet), you need to get an expensive switch involved. With Modbus RTU (serial), you can just daisy chain all the devices together. Devices with old 8-bit processors and a tiny bit of memory can easily do Modbus serial but you’ll need a more expensive platform to do Ethernet.

Did you know?


For more information on our Modbus RTU products and services call 1-800-249-1612 or email us at Solutions@rtautomation.com.
Looking to add Modbus RTU to your project?

See Modbus RTU Solutions