Process Control and Automation using Modbus Protocol

Modbus is the fundamental network protocol used in most industrial applications today. It is universal, open and an easy to use protocol. Modbus has been around industries for over 30 years, but it's still the dominant voluntary standard used in almost all major industrial process control, instrumentation and automation products in the market today. New industrial products such as PLC, PAC, I/O devices, and meters may have an Ethernet, serial or even perhaps wireless interface, but Modbus is still the preferred protocol

The pivotal advantage of Modbus is that it can run over virtually all communication media, including twisted pair wires, wireless, fibre optics, Ethernet, telephone modems, cell phones and microwave which would imply that a Modbus connection could be easily established on a new or existing factory floor.

This white paper examines the functionality of Modbus and ways that Modbus can be used in industrial applications.

What is MODBUS

Modbus is a serial communications protocol, developed by Modicon in 1979 as a means for communicating with industrial electronic devices such as PLC, PAC or DCS over twisted pair wires. Modbus is a simple and rugged protocol that has become the de-facto standard of the process control and automation industry.It allows for communication between many devices (approximately 240) connected on the same network, i.e. a system that measures physical phenomenon such as temperature, current, voltage and humidity communicating the results to a Master (computer). Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems.

A Modbus system comprises of a master and slave, where the "master" communicates with one or multiple "slaves." The master typically is a Programmable Logic Controller (PLC), PC, Programmable Automation Controller (PAC), Distributed Control System (DCS) or Remote Terminal Unit (RTU). The slaves are mostly field devices, all of which connect to the network in a multi-drop configuration as depicted in Figure 1.

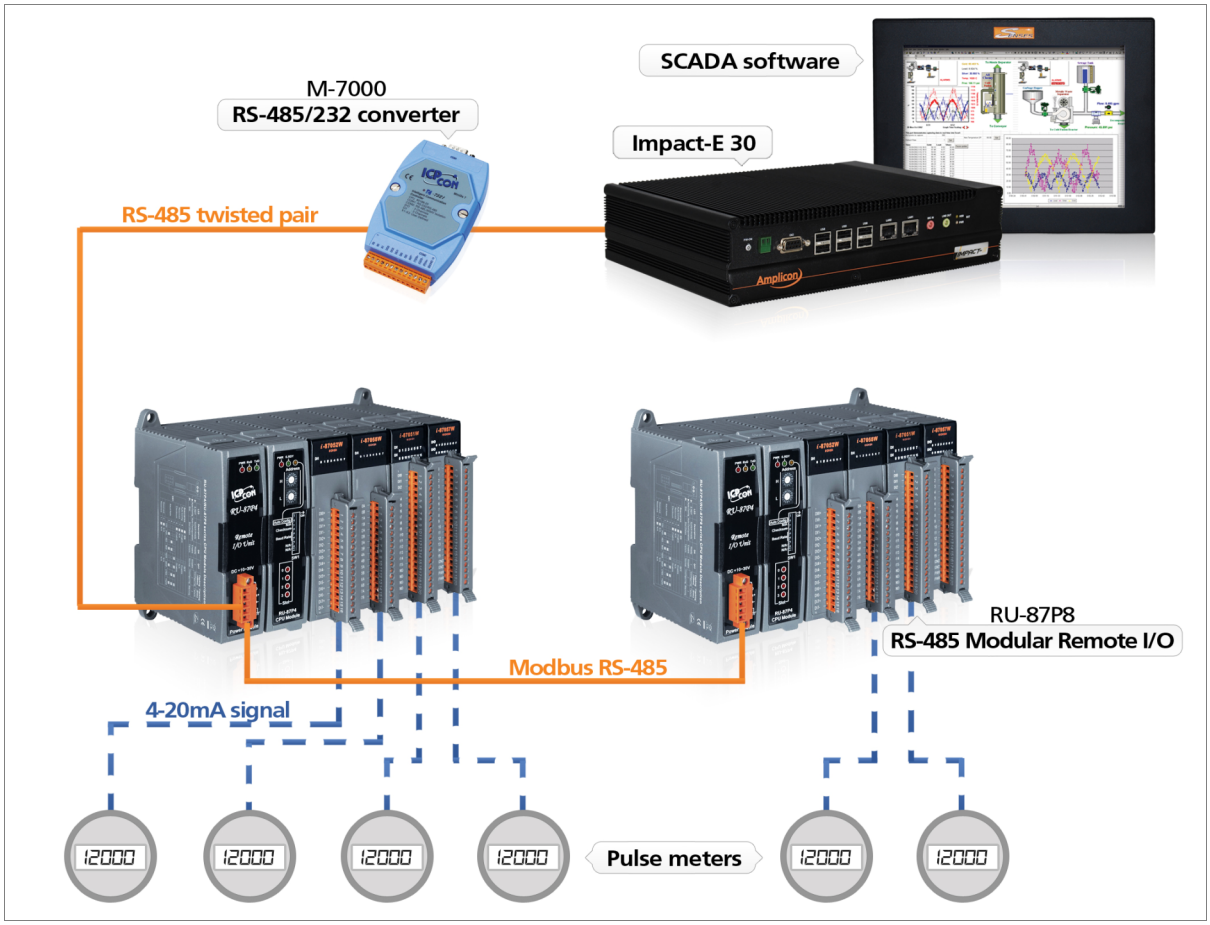

Figure 1. *Modbus RTU network consisting of a "Master," such as a PLC, PAC or DCS and up to 247 "Slave" devices*

When a Modbus RTU master wants information from a device, the master sends a message that contains the device's address, the data it wants and a checksum for error detection. Every other device on the network sees the message, but only the device that is addressed responds.

Slave devices on a Modbus network cannot initiate communication, these slave devices can only respond to a master.

The three most common Modbus types used today are as follows:

- Modbus ASCII
- Modbus RTU (based on serial communication like RS485 and RS232)
- Modbus/TCP (based on Ethernet communication)

Modbus ASCII, all messages are coded in hexadecimal using 4-bit ASCII characters. Modbus ASCII marks the start of each message with a colon character " : " (hex 3A). The end of each message is terminated with the carriage return and line feed characters (hex 0D and 0A). This allows the space between bytes to be variable making it suitable for transmission through some modems.

Modbus ASCII is the slowest of the three protocols, but is suitable when telephone modem or radio (RF) links are used. This is because ASCII uses characters to delimit a message. Because of this delimiting of the message, any delays in the transmission medium will not cause the message to be misinterpreted by the receiving device. This can be important when dealing with slow modems, mobile phones, noisy connections, or other difficult transmission mediums.

Modbus RTU, data is coded in binary and requires only one communication byte per data byte. This is ideal for use over RS232 or multi-drop RS485 networks, at speeds from 1,200 to 115K baud. The most common speeds are 9,600 and 19,200 baud. Modbus RTU is the most widely used industrial protocol hence most of this white paper will focus on Modbus RTU basics and application considerations.

Modbus/TCP is simply Modbus over Ethernet. Instead of using device addresses to communicate with slave devices, IP addresses are used. With Modbus/TCP, the Modbus data is simply encapsulated inside a TCP/IP packet hence, any Ethernet network that supports TCP/ IP should immediately support Modbus/TCP.

The difference between Modbus RTU and Modbus ASCII

There are two basic transmission modes found in serial Modbus connections, ASCII and RTU. These transmission modes determine the way in which the Modbus messages are coded. In ASCII format, the messages are readable, whereas in RTU the messages are in binary coding and cannot be read while monitoring. The trade-off is that the RTU messages are a smaller size, which allows for more data exchange in the same time span. However, all nodes within one Modbus network must be of the same transmission mode, meaning Modbus ASCII cannot communicate with Modbus RTU and vice versa.

In Modbus ASCII, messages are encoded with hexadecimal value, represented with comprehensive ASCII characters. The characters used for this encoding are 0…9 and A…F. For every byte of information, two communication-bytes are used because every communication-byte can only define 4 bits in the hexadecimal system. Modbus RTU, however, exchanges data in binary format where each byte of data is coded in one communication-byte.

The Modbus messages on a serial connection are not broadcast in plain format. They are constructed in a way that allows receivers an easy way to detect the beginning and end of a message. The characters start and end a frame when in ASCII mode. To flag the start of a message, a colon ‘:’ is used and each message is ended with a CR/LF combination, Modbus RTU uses a different method. In Modbus RTU, framing is constructed by measuring gaps of silence on the communication line. Before each message, there must be a minimum gap of 3.5 characters. To prepare for new messages, the receiver clears the buffer when a gap of 1.5 characters is detected. One of the main differences between Modbus ASCII and Modbus RTU is that ASCII allows gaps between the bytes of a message with a maximum length of 1 second. With Modbus RTU, continuous streams of messages must be sent.

Properties of Modbus ASCII and Modbus RTU

	Modbus/ASCII		Modbus/RTU	
Characters	ASCII **0**...**9** and **A**..**F**		Binary **0**...**255**	
Error check	**LRC** Longitudinal Redundancy Check		**CRC** Cyclic Redundancy Check	
Frame start	character '**:**'		3.5 chars silence	
Frame end	characters **CR**/**LF**		3.5 chars silence	
Gaps in message	1 sec		1.5 times char length	
Start bit	**1**		**1**	
Data bits	**7**		**8**	
Parity	even/odd	none	even/odd	None
Stop bits	**1**	**2**	**1**	**2**

Modbus RTU Fundamentals

To enable communication with a slave device, the master must send a message or query containing:-

- Device Address
- Function Code
- Data
- Error Check

The **Device Address** is a number from 0 to 247. Messages sent to address 0 (broadcast messages) can be accepted by all slaves, but numbers 1-247 are addresses of specific devices. With the exception of broadcast messages, a slave device always responds to a Modbus message so the master knows the message was received.

The **Function Code** defines the command that the slave device is to execute, such as read data, accept data, report status, etc. (Figure 2). Function codes are 1 to 255. Some function codes have sub-function codes.

Command	Function Code
01	Read Coils
02	Read Discrete Inputs
03	Read Holding Registers
04	Read Input Registers
05	Write Single Coil
06	Write Single Register
07	Read Exception Status
08	Diagnostics
*	
*	
Xx	Up to 255 function codes, depending on the device

Figure 2. *Typical Function Codes*

The **Data** defines addresses in the device's memory map or read functions, contains data values to be written into the device's memory, or contains other information needed to carry out the function requested.

The **Error Check** is a 16-bit numeric value representing the Cyclic Redundancy Check (CRC). The CRC is generated by the master (via a complex procedure involving ORing and shifting data) and checked by the receiving device. If the CRC values do not match, the device asks for a retransmission of the message. In some systems, a parity check can also be applied.

When the slave device performs the requested function, it then sends a message back to the master. The returning message contains the slave's address and requested function code (so the master knows which slave device is responding), the data requested and an Error Check value.

Modbus Memory Map

Each Modbus device has memory, where process variable data is stored. The Modbus specification dictates how data is retrieved and what type of data can be retrieved. However, it does not place a limitation on how and where the device manufacturer maps this data in its memory map. The below table is a common example of how a Modbus device manufacture might logically map different types of process variable data. Discrete inputs and coils are one-bit values and each has a specific address. Analog inputs (also called "Input Registers") are stored in 16-bit registers. By utilising two of these registers, Modbus can support the IEEE 32-bit floating point format. Holding Registers are also 16-bit internal registers that can support floating point.

Data in the memory map is defined in the Modbus specification. Assuming that the Modbus device manufacturer followed the Modbus specification (not all manufacturers do), then all data can easily be accessed by the Modbus master, which follows the specification. In most cases, the Modbus device manufacturer publishes the memory locations as shown in figure 3 making it easier for the engineer or scientist programming the master to communicate with the slave device.

Table Addresses	Type	Table Name
1-9999	Read or Write	Coils
10001-19999	Read Only	Discrete Inputs
20001-29999	Read Only	Input Registers
30001-39999	Read or Write	Holding Registers

Figure 3. *The operation manual for most Modbus compatible devices*

Reading and Writing Data

Modbus has up to 255 function codes, but 01 (read coils), 02 (read discrete inputs), 03 (read holding registers) and 04 (read input registers) are the most commonly used read functions that are used to collect data from Modbus slaves as shown in figure 2. For example, to read three 16-bit words of analog data from device 7's memory map, the master sends a command that looks like this:

7 03 2 4 CRC

Where 7 is the device address, 03 says to read input registers, 2 is the starting address (address 30,002), 4 means to read three contiguous data values starting at address 30,002 and CRC is the error check value for this message. The slave device upon receiving this command sends back a response that looks like this:

7 03 aa bb cc CRC

Where 7 is the device's address 03 is the repeated read command; aa, bb and cc are the three 16-bit data values; and CRC is the error check value for this message.

Reading and writing digital inputs and outputs is done in a similar manner using different read and write functions. Assuming that the device complies with the Modbus specification, it is a simple programming task to set up the master to read and write data, check status, obtain diagnostic information and perform various control and monitoring functions.

Wireless Modbus

The cost savings in installation costs (wiring infrastructure) is the pivotal benefit for using wireless devices in industrial applications. Wireless technology enables process control engineers to quickly and cost effectively obtain real-time data from anywhere in the field/factory floor or perhaps various other remote locations at any time which is essential for an industrial automation and process control system.

A Modbus network can be set up fairly easily to work over a wireless link as shown in figure 4. Essentially, all the wireless link does is replacing the twisted pair cables with a transmitter/receiver at each end of the network. It is imperative that you consult Amplicon industrial wireless specialist to assist you in selecting the right wireless hardware for your industrial applications.

Modbus via wireless is transparent to the control system or host and the slave devices. The host system doesn't know if a wireless Modbus network exists, as it doesn't have to deal with it. When a Modbus master makes a request to a slave and the packets arrive at the transmitting radio, that radio will usually re-order the packets and encrypt them before transmission. Once the RF (Radio Frequency) packets are received by the "slave" radio, it de-encrypts them and puts them back in order to represent a valid Modbus Packet. Assuming that the packet has not been damaged or corrupted, it will then be sent to the destined slave. The slave will respond back to the Master and the process starts again.

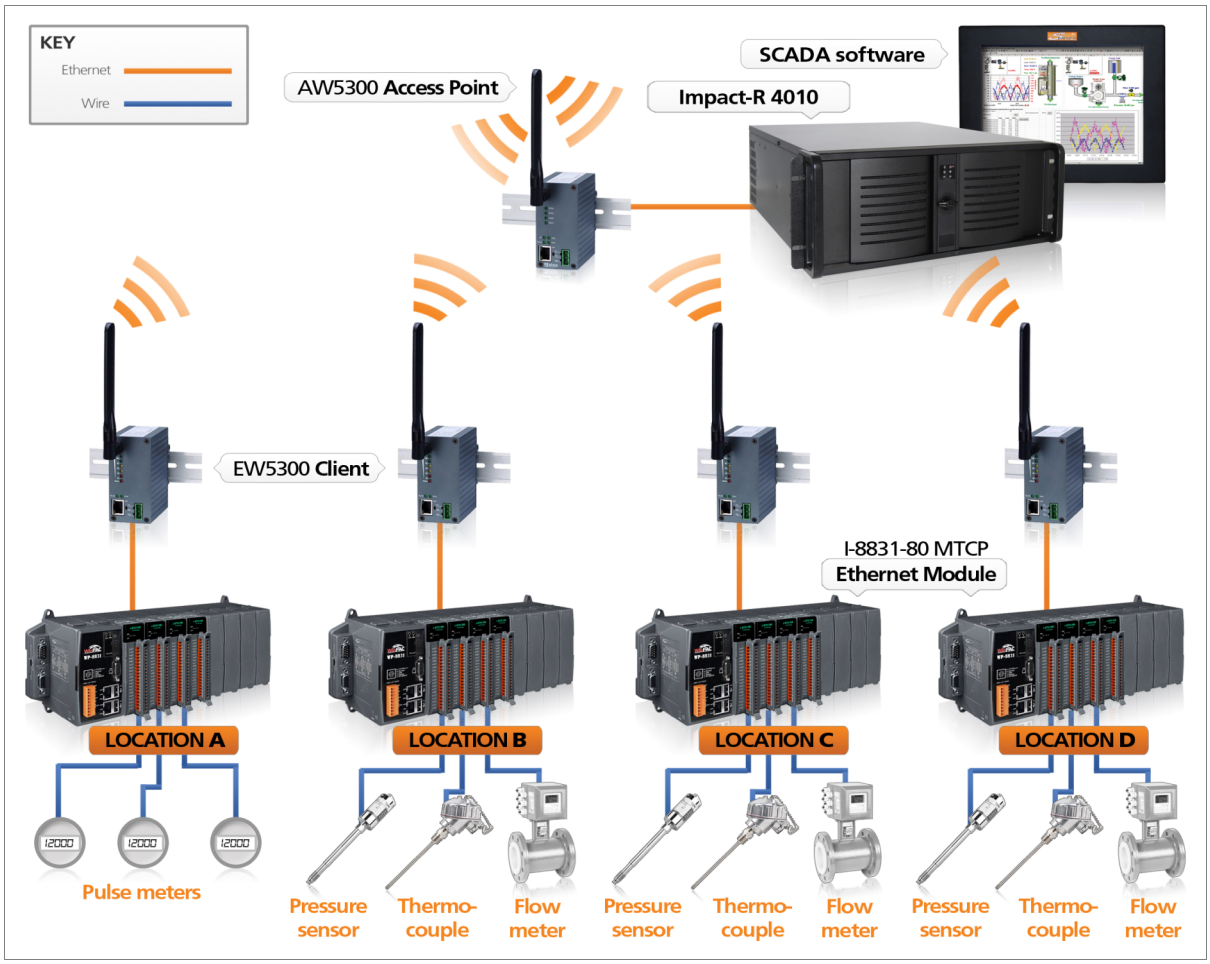

***Figure 4**. Wireless Modbus topology*

It is important to pay special attention to a Modbus communication parameter called "timeout." Timeout is the amount of time that the Modbus master will wait for a response from a slave device before attempting a re-transmission. Depending on how well the radio is communicating, packets can be delayed causing an unnecessary amount of retries and re-transmits. Today most of these parameters can be managed for efficient transfer of Modbus packets. However, it is essential to conduct a proper radio site survey that include signal strength and spectrum noise analysis before implementing a wireless Modbus network to alleviate any communication problems.

Modbus/TCP (Ethernet)

Modbus/TCP is often referred to as Modbus over Ethernet. Modbus/TCP (also Modbus-TCP/IP) is simply the Modbus RTU protocol with a TCP interface that runs on Ethernet. The Modbus messaging structure is the application protocol that defines the rules for organising and interpreting the data independent of the data transmission medium. TCP/IP refers to the Transmission Control Protocol and Internet Protocol, which provides the transmission medium for Modbus TCP/IP messaging.

This enables Modbus/TCP devices to immediately and easily connect and communicate over existing Ethernet and fibre networks. Modbus/TCP also allows many more addresses than RS485, the use of multiple Masters and speeds in the gigabit range. While Modbus

RTU has a limitation of 247 nodes per network, Modbus/TCP networks can have as many slaves as the physical layer can handle. Often this number is somewhere around 1,024. Ethernet's rapid adoption within the process control and automation industry has allowed Modbus/TCP to become the most widely used, fastest growing and supported industrial protocol over Ethernet.

Unlike Modbus RTU and Modbus ASCII, Modbus/TCP will allow multiple masters to poll the same slave device simultaneously. This is permitted because over Ethernet via TCP/IP, multiple messages can be sent, buffered and delivered without the requirement of token passing or total bus control, which is often the case with many RS485 and RS422 protocols.

Peer to Peer Communication

Peer-to-peer communications technology enables the reproduction of I/O signals and extended data transmissions over Ethernet without the need of a Master i.e host PC, PLC or PAC.

You can transmit data from an input module to an output module over copper, fibre, or wireless Ethernet infrastructure without the need of a Master, which significantly reduces the cabling and wiring costs. With this technology, digital and analog signals can be duplicated over virtually unlimited distances with no host PC, PLC, or controller needed and without any noise interference. Figure 5 depicts a typical peer-to-peer topology.

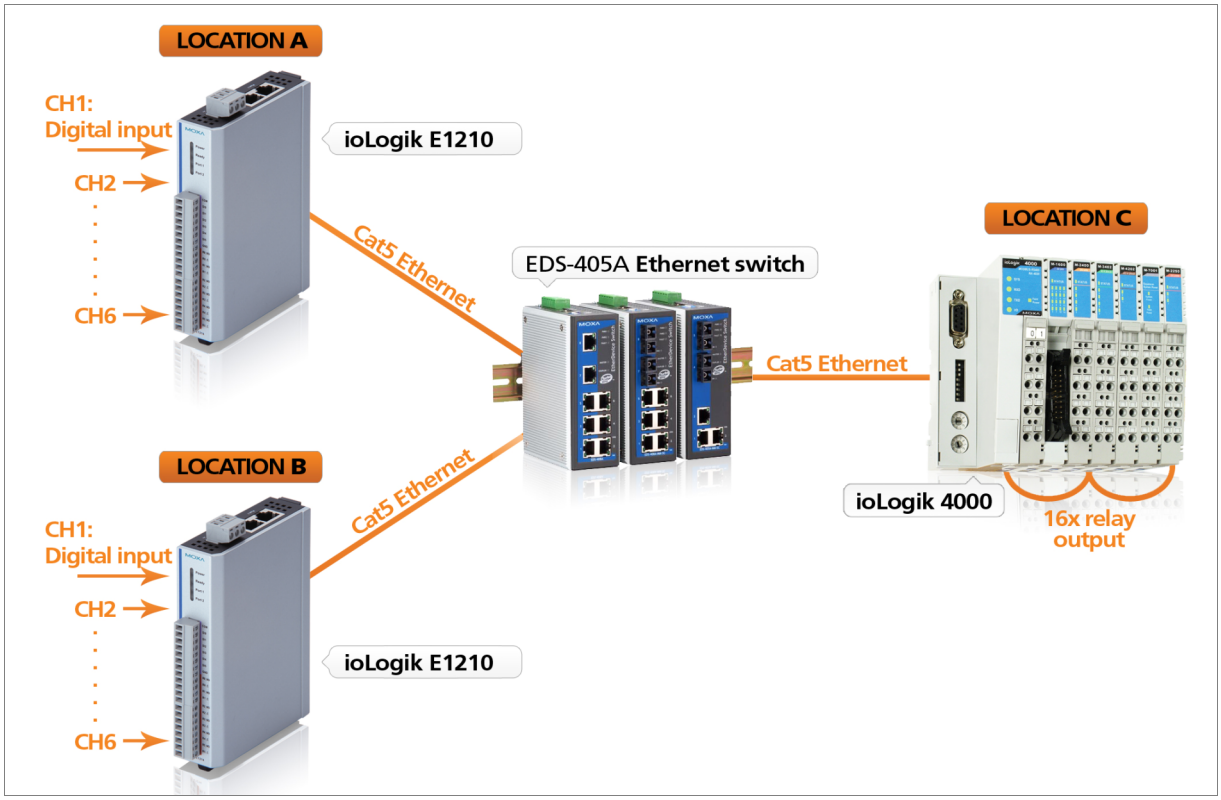

Figure 5. *Peer to peer topology*