**NModBus Report**

1. RTU

[**aster\_NS\_Modbus\_Protocol-2.htm**](https://product-help.schneider-electric.com/ED/ES_Power/NT-NW_Modbus_IEC_Guide/EDMS/DOCA0054EN/DOCA0054xx/Master_NS_Modbus_Protocol/Master_NS_Modbus_Protocol-2.htm)

What is Modbus and How does it work?

**What is Modbus?**

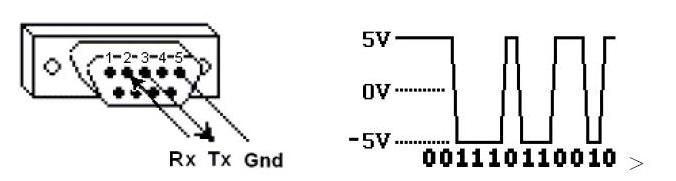
Modbus is a serial communication protocol developed by Modicon published by Modicon® in 1979 for use with its programmable logic controllers (PLCs). In simple terms, it is a method used for transmitting information over serial lines between electronic devices. The device requesting the information is called the Modbus Client and the devices supplying information are Modbus Servers. In a standard Modbus network, there is one Client and up to 247 Servers, each with a unique Server Address from 1 to 247. The Client can also write information to the Servers.  
  
The official Modbus specification can be found at <http://www.modbus.org/>

**What is it used for?**

Modbus is an open protocol, meaning that it's free for manufacturers to build into their equipment without having to pay royalties. It has become a standard communications protocol in industry, and is now the most commonly available means of connecting industrial electronic devices. It is used widely by many manufacturers throughout many industries. Modbus is typically used to transmit signals from instrumentation and control devices back to a main controller or data gathering system, 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. Versions of the Modbus protocol exist for serial lines (Modbus RTU and Modbus ASCII) and for Ethernet (Modbus TCP).

**How does it work?**

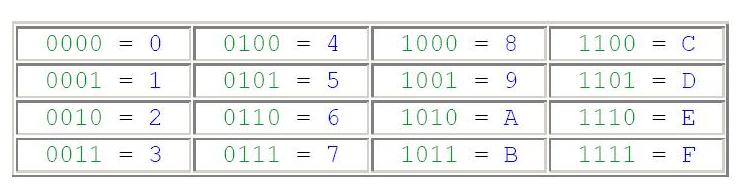
Modbus is transmitted over serial lines between devices. The simplest setup would be a single serial cable connecting the serial ports on two devices, a Client and a Server.



The data is sent as series of ones and zeroes called bits. Each bit is sent as a voltage. Zeroes are sent as positive voltages and a ones as negative. The bits are sent very quickly. A typical transmission speed is 9600 baud (bits per second).

**What is hexadecimal?**

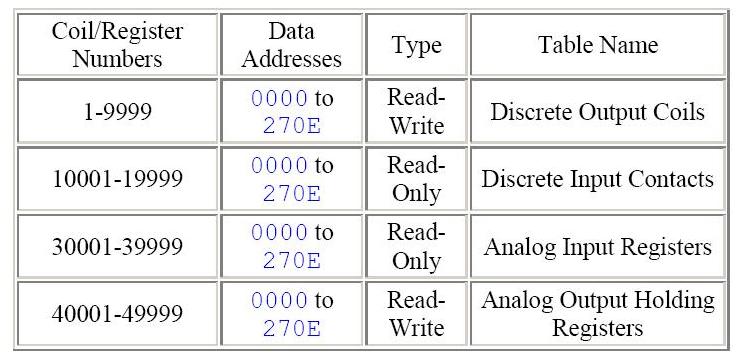
When troubleshooting problems, it can be helpful to see the actual raw data being transmitted. Long strings of ones and zeroes are difficult to read, so the bits are combined and shown in hexadecimal. Each block of 4 bits is represented by one of the sixteen characters from 0 to F.



Each block of 8 bits (called a byte) is represented by one of the 256 character pairs from 00 to FF.

**How is data stored in Standard Modbus?**

Information is stored in the Server device in four different tables. Two tables store on/off discrete values (coils) and two store numerical values (registers). The coils and registers each have a read-only table and read-write table. Each table has 9999 values. Each coil or contact is 1 bit and assigned a data address between 0000 and 270E. Each register is 1 word = 16 bits = 2 bytes and also has data address between 0000 and 270E.



Coil/Register Numbers can be thought of as location names since they do not appear in the actual messages. The Data Addresses are used in the messages. For example, the first Holding Register, number 40001, has the Data Address 0000. The difference between these two values is the **offset**. Each table has a different offset. 1, 10001, 30001 and 40001.

**What is the Server ID?**

Each server in a network is assigned a unique unit address from 1 to 247. When the client requests data, the first byte it sends is the Server address. This way each server knows after the first byte whether or not to ignore the message.

**Modbus YouTube tutorial**

[**https://www.youtube.com/watch?v=Pteeec3yCrk**](https://www.youtube.com/watch?v=Pteeec3yCrk)

[**https://www.rfwireless-world.com/Tutorials/Modbus-Protocol-tutorial.html**](https://www.rfwireless-world.com/Tutorials/Modbus-Protocol-tutorial.html)

**Note**

**Reduce reputable code.**

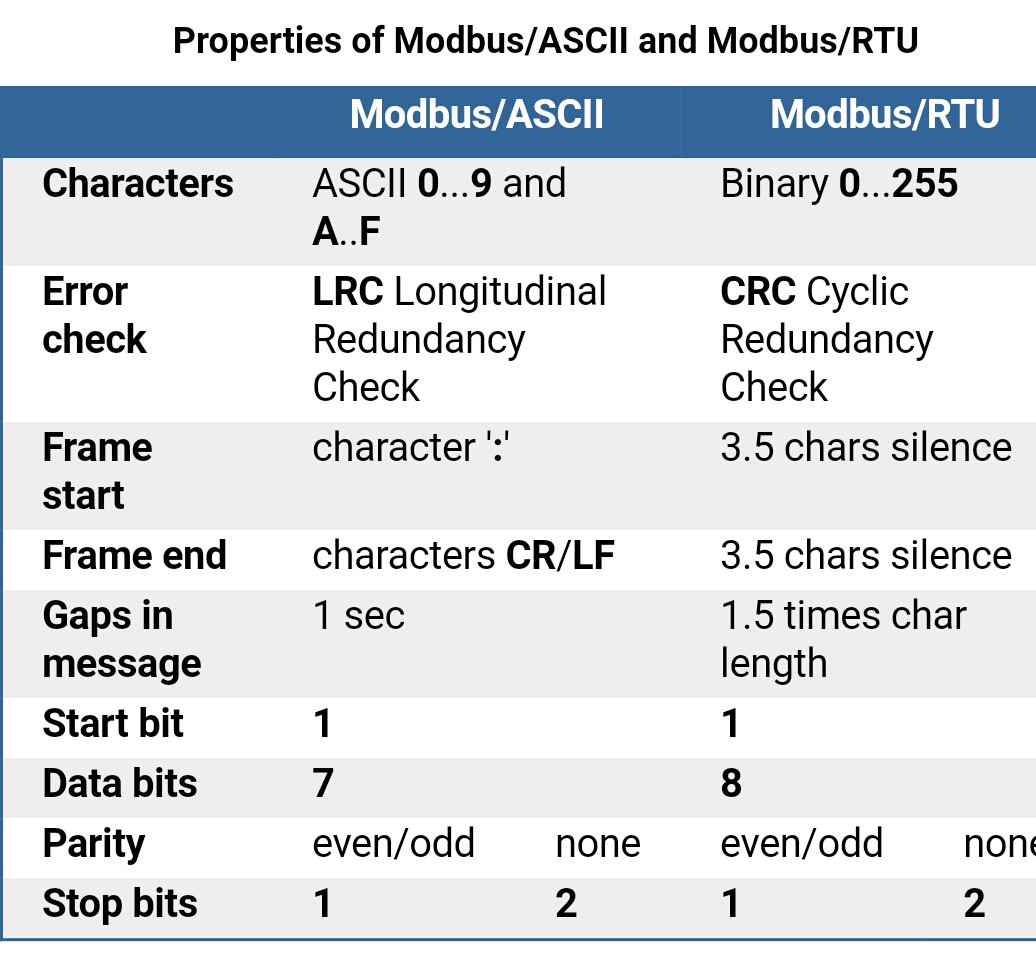
**Check open connection time show message connection available or not**

**Project Working in Rushikesh branch**

[**https://www.c-sharpcorner.com/UploadFile/201fc1/creating-a-serversharp47client-application-using-only-tcp-prot/**](https://www.c-sharpcorner.com/UploadFile/201fc1/creating-a-serversharp47client-application-using-only-tcp-prot/)

**Error:**

* Even and odd parity



**Read Multiple Register Notes**

[**https://www.codeproject.com/Articles/20929/Simple-Modbus-Protocol-in-C-NET-2-0**](https://www.codeproject.com/Articles/20929/Simple-Modbus-Protocol-in-C-NET-2-0)

**Solve Not Responding Problem.**

[**https://www.youtube.com/watch?v=3BxkMkWIIgY**](https://www.youtube.com/watch?v=3BxkMkWIIgY)

**Note Bosale sir la call :**

**Continue communicate hot ahe he kas kalel. Mhanje te band vagere zal ki aplyala kas kalnar disconnect zal mhanun. Continuous monitoring karnyasathi kay karav lagel.**

**Note For Code Updation :**

**Add timer to display coil status after specific time**

**11/10/2022**

**New development start inside UpdateModbus branch**

**Threading GUI Notes Vidio link**

[**https://www.youtube.com/watch?v=7F3OGd-JSO8**](https://www.youtube.com/watch?v=7F3OGd-JSO8)

[**https://www.codeproject.com/Articles/18702/Threading-in-NET-and-WinForms**](https://www.codeproject.com/Articles/18702/Threading-in-NET-and-WinForms)

**Thread sef application**

[**https://www.c-sharpcorner.com/UploadFile/1d42da/thread-safe-calls-using-windows-form-controls-in-C-Sharp/**](https://www.c-sharpcorner.com/UploadFile/1d42da/thread-safe-calls-using-windows-form-controls-in-C-Sharp/)

**Changes in Bar Concertized Bar Inspection to Convert Rs232**

**display\_Counts();**

**Err :**

PLC.WriteSingleRegister((ushort)reg, (ushort)ImageIndex);

Total Err 3

**private void cmb\_part\_name\_SelectedIndexChanged(object sender, EventArgs e)**

**Err:**

PLC.WriteSingleRegister(1714, 8);

Total Err 8

**btn\_start\_inspection\_Click(object sender, EventArgs e)**

**Err:**

PLC.WriteSingleRegister(1700, 1); // inspection start

Total Err 1

**private void StopInspection()**

**Err:**

PLC.WriteSingleRegister(1700, 0); // inspection off

Total Err 1

**private void MasterForm\_FormClosing(object sender, FormClosingEventArgs e)**

**Err:**

PLC.WriteSingleCoil(8892, false); // software close

**public void Maintain\_Connection()**

**Err:**

CycleCompleted = PLC.ReadHoldingRegisters(6408, 2);

temp = PLC.ReadHoldingRegisters(6396, 2);

ErrorCodes = PLC.ReadHoldingRegisters(1716, 2);

PLC.WriteSingleRegister(6408, (ushort)0);