**MODBUS PROTOCOL**

**What is MODBUS?**

Modbus is a serial communication PROTOCOL developed by Modicon in 1979. It was developed to use with its programmable logic controllers (PLCs).

Basically It is a method used for transmitting information over serial lines between electronic devices.  
The device requesting the information is called the Modbus Client (or Master) and the devices supplying information are Modbus Servers (or Slave).  
In a standard Modbus network, there is one Client and up to 247 Servers, each with a unique Server Address from 1 to 247.  
In addition to requesting, the Client can also write information to the Servers.

Modbus has become a standard communications protocol in industries, 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 in transmitting the data from control devices back to the main controller. For eg- the STM32 serves as the Master and it can gather the data from a temperature sensor. Which it will then utilize to regulate the colling system.

There are 2 major versions of the Modbus protocol, for serial lines (Modbus RTU and Modbus ASCII) and for Ethernet (Modbus TCP).

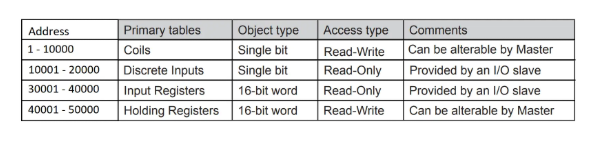
We will start with modbus RTU (Remote Terminal Unit).

**The Memory Arrangement**

In the end everything comes down to memories. Whether the master is requesting the data from the slave or it is writing the data to the slave, only the information stored a particular memory location is being read or written.

A standard modbus device have 4 types of memory Areas. They are: Coils, Discrete Inputs, Input Registers and Holding Registers.

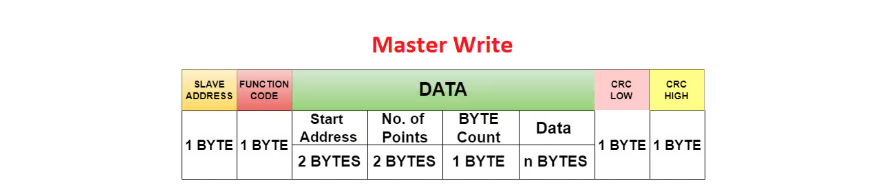
Below is the picture explaining the address and the types of these memory Areas

****

* **Discrete inputs** are similar to coils and this region also have the memory blocks of 1 bit.
  + But the Discrete inputs are only readable, so the master can not modify the bits in these locations.
  + The Address for the Discrete inputs ranges from 10001 to 20000, which means there are totally 10000 (1 bit) memory blocks present in this region.
  + For example, the master could read an input switch whether it’s pressed or open, but it can not control the switch.
* **Input Registers** memory blocks are 16 bits in size. They are only readable, so the master can not write any value to these blocks.
  + The Address for the Input Registers ranges from 30001 to 40000, which means there are totally 10000 (16 bit) memory blocks present in this region.
  + Input Registers are generally used to store the analog values of the sensors, like temp or pressure sensors. The master can only read these values and ca not modify them.
* **Holding Registers** memory blocks are also 16 bits in size. They are both readable and writable, so the master can modify any value in these blocks.
  + The Address for the Holding Registers ranges from 40001 to 50000, which means there are totally 10000 (16 bit) memory blocks present in this region.
  + Holding Registers can be used to give inputs to the modules that uses the analog values. Like servo motors.

**The Data Format**

The maximum data that can be transferred in modbus is 256 bytes. The format for the data is shown below



As shown above, out of 256 Bytes, 1 byte is occupied by the Slave ID, another byte by the Function code, and the 2 bytes are occupied by the CRC. These bytes are fixed irrespective of whether the master or slave is transmitting the data. The rest (upto 252) are used by the data field.

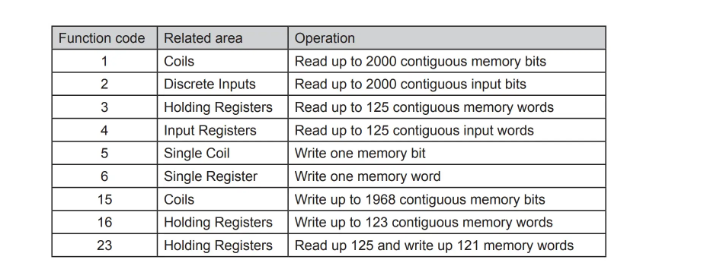
The data filed varies with a lot of things. For eg- It is different for master sending a query or sending the data to be written. It changes again when the slave responds to the query.

Basically data field is where we will discover different set of variation of each types.

**Function Codes**

The function code is something which defines what the master wants to do. There are different function codes for when the master requests data from the particular memory area. Again the code changes when the master wants to write to a memory area.

The function codes are shown in the picture below



* The function code **1 and 2** are used to read the data from the coils and Discrete inputs respectively
* Function Code **3 and 4** are used to read the Holding and input Registers
* Function code **5 and 6** are used to write 1 memory block of the coil and Holding Registers respectively
* Function codes **15 and 16** can be used to write multiple memory blocks of coils and Holding Registers.

We will cover all the function codes in upcoming tutorials in this series.

We have already seen the addresses of the memory blocks in the **Memory Arrangement Section**. But when the master sends a query to read some data at a specific address, it does not send a complete address.

The modbus device have something called the Offset. As per the modbus standard, the offset is same as the address of the first memory block. For eg- In case of holding Registers, the Offset would be 40001.

The master sends the address, ha corresponds to the difference between the Actual Register address (that the master wants to read) and the offset.  
For eg- If the master want to read the Register 40008, it will send the address 7 (40008 – 40001).

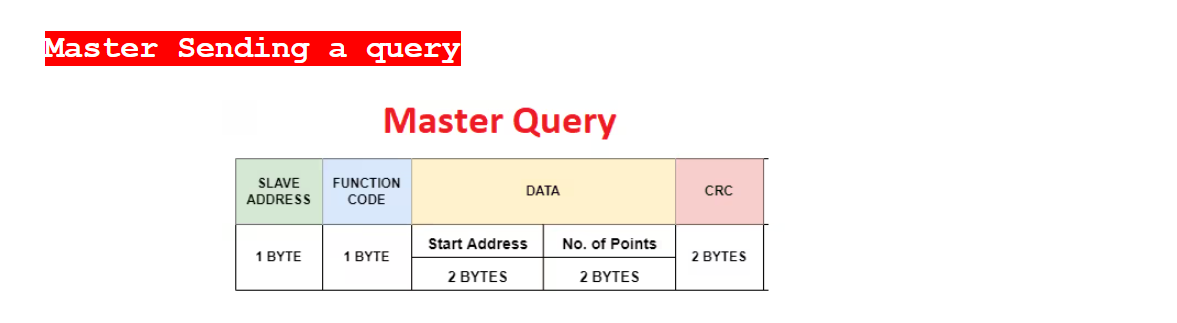
**So the question is**, if the master only sends 7 how do the slave know that it wants to read 40008, and not 30008 or just 8 ?

Well this is where the function code comes in. The master send the function code, 3, followed by the address, 7.  
The slave now is able to figure out that the master wants to read the holding Registers (which starts at 40001) and the address is 7 (7+40001 = 40008).

**The Data Field**

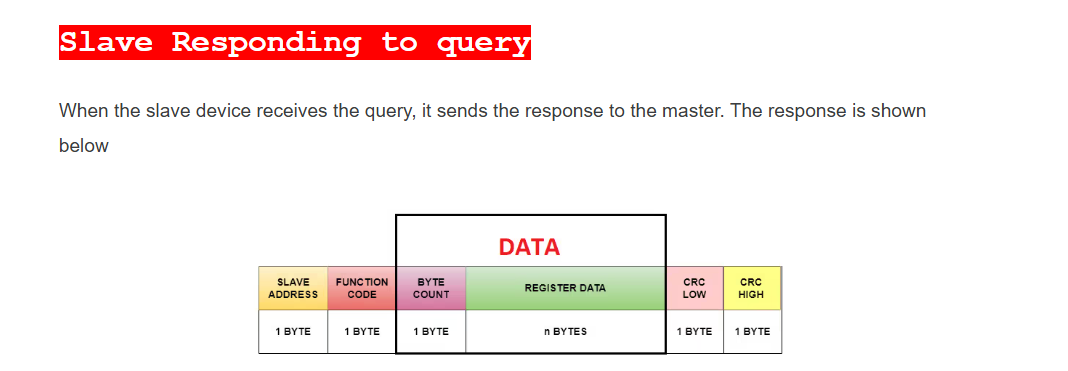
As I mentioned above the data field varies depending on the what type of operation is being performed on the modbus. We will cover all the operations in the upcoming tutorials.

Below are some examples of the data field being varied.



When the master send a query to read some data from a specific location in the slave, the data field consists of the following:

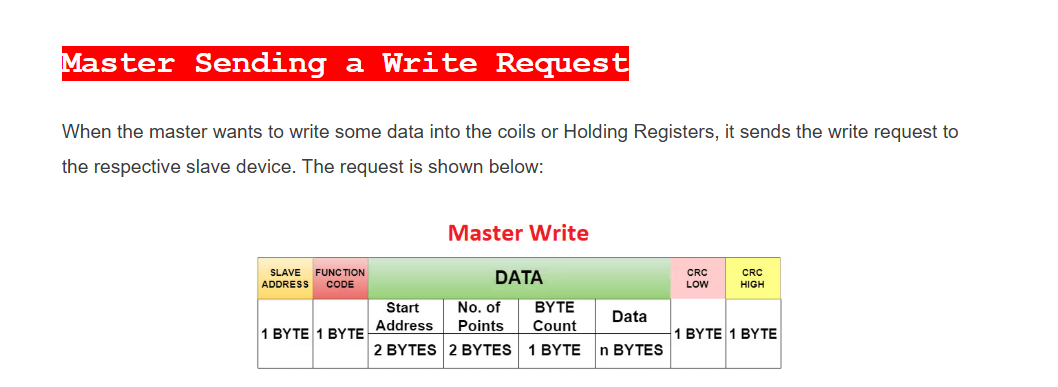
* The **start address** of the register (2 Bytes). This is the starting register from where the master wants to read the data.
* The **number of points** it wants to read (2 Bytes)



The pattern remains the same with slave address and function code occupying 1 byte each and CRC occupying 2 bytes.

The only change is again in the Data field. It consists of the following:

* **Byte count** (The number of data bytes slave is going to send) occupying 1 Byte
* The actual **data** itself. The occupancy here depends on how many data bytes the slave is sending.



Again the rest of the pattern is same other than the data field. It contains the following:

* **Start address** of the Register/Coil where the master wants to start wring the data from. (2 Bytes)
* **Number of points** the master wants to write (2 Bytes)
* **Byte count**, the number of bytes master wants to write (1 Byte)
* And the actual **data** itself.

One thing you might have noticed that I have used the term “Number of Points” master wants to read or write.  
We already know that the memory areas are divided into coils, which occupy 1 bit, and registers, which occupy 16 bits.

So when the master wants to read or write the coils, number of points signifies number of coils. And in case of Registers, it signify the number of registers.