MODBUS Organization

MODBUS/TCP Server

Supported version TOP Design Studio V1.0 or higher

CONTENTS

We want to thank our customers who use the Touch Operation Panel.

1. System configuration

This driver allows the TOP to operate by adding the MODBUS/TCP server feature.

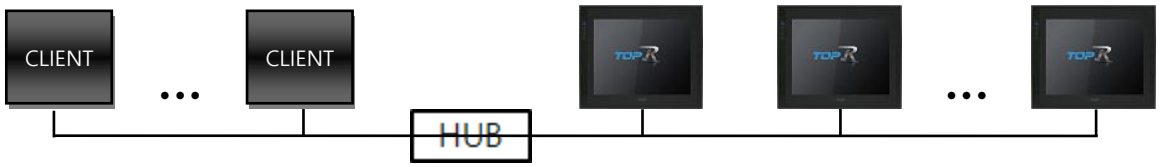
External device	Communication method	System setting	Cable
MODBUS/TCP Client	Ethernet (TCP)	3. TOP communication setting	Twisted pair cable *Note 1)

*Note 1) Twisted pair cable

- Refer to STP (Shielded Twisted Pair Cable) or UTP (Unshielded Twisted Pair Cable) Category 3, 4, 5.
- Depending on the network configuration, you can connect to components such as the hub and transceiver, and in this case, use a direct cable.

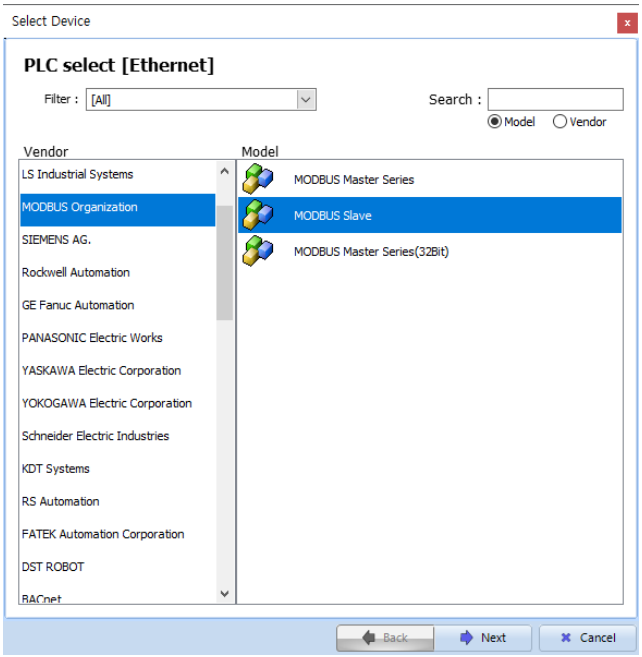
■ Connectable configuration

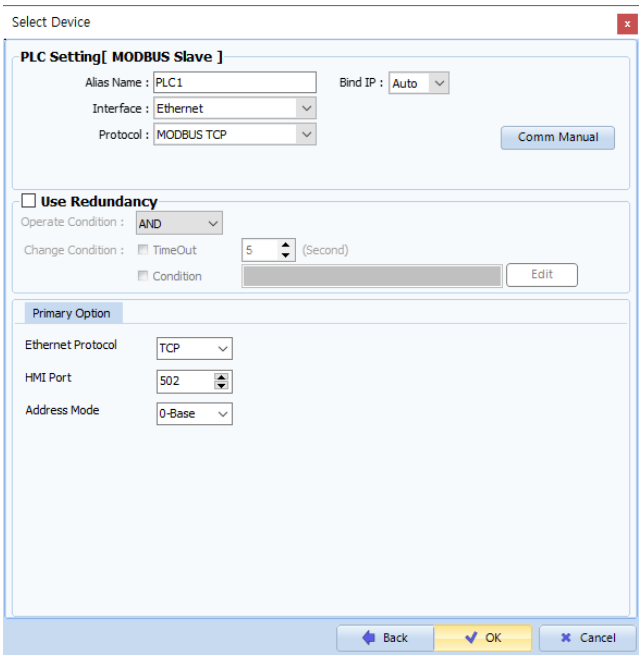
・N:N connection



2. External device selection

■ Select a TOP model and a port, and then select an external device.





Settings		Contents
TOP	Model	Check the display and process of TOP to select the touch model.
External device	Vendor	Select the vendor of the external device to be connected to TOP. Select "MODBUS Organization".
	PLC	Select an external device to connect to TOP. **Model** / **Interface** / **Protocol** MODBUS Slave / Ethernet / MODBUS TCP Please check the system configuration in Chapter 1 to see if the external device you want to connect is a model whose system can be configured.

3. TOP communication setting

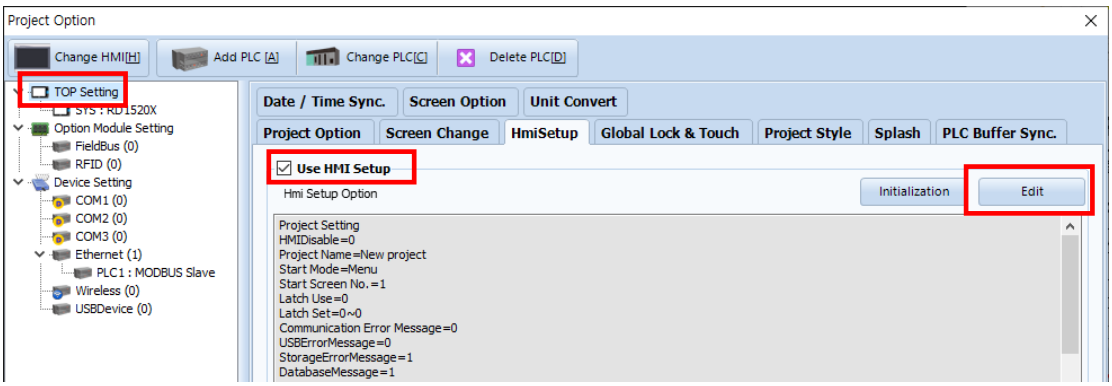
The communication can be set in TOP Design Studio or TOP main menu. The communication should be set in the same way as that of the external device.

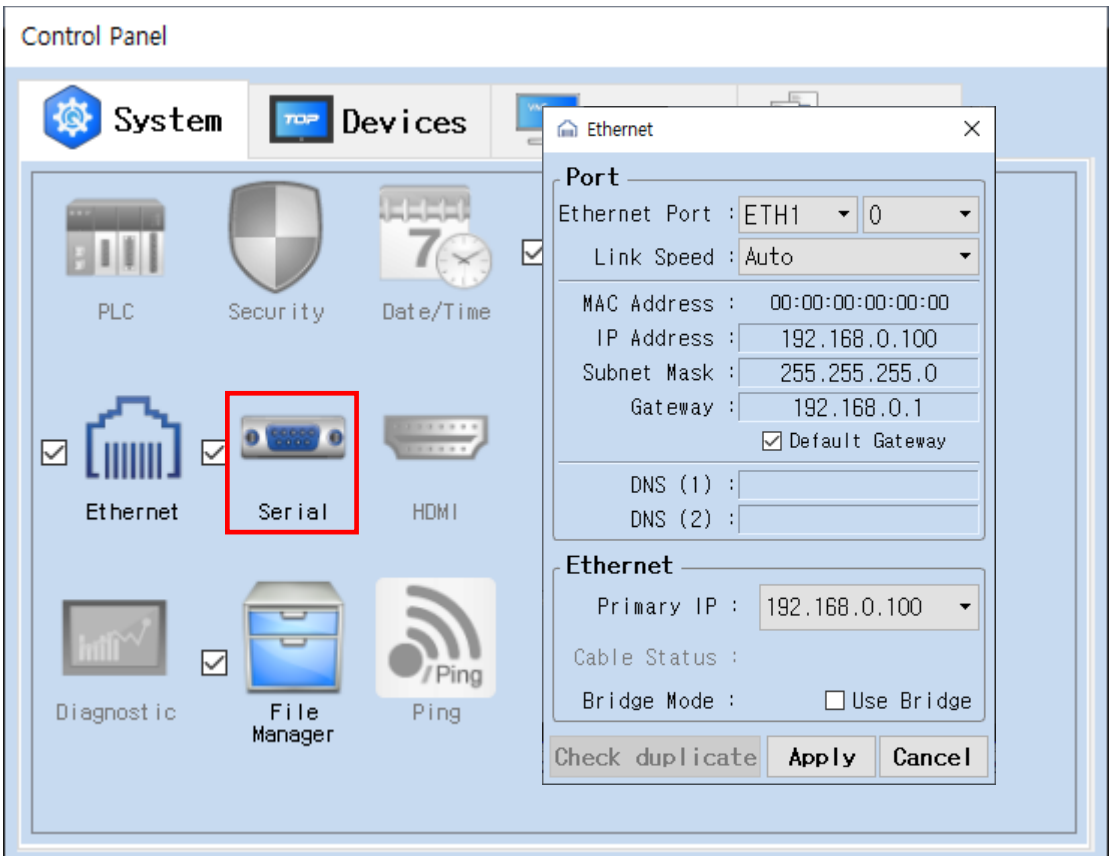
3.1 Communication setting in TOP Design Studio

(1) Communication interface setting

■ [Project] → [Property] → [TOP Setting] → [HMI Setup] → [Use HMI Setup Check] → [Edit] → [Ethernet]

– Set the TOP communication interface in TOP Design Studio.





Items	TOP	External device	Remarks
IP Address	192.168.0.100	192.168.0.50	
Subnet Mask	255.255.255.0	255.255.255.0	
Gateway	192.168.0.1	192.168.0.1	

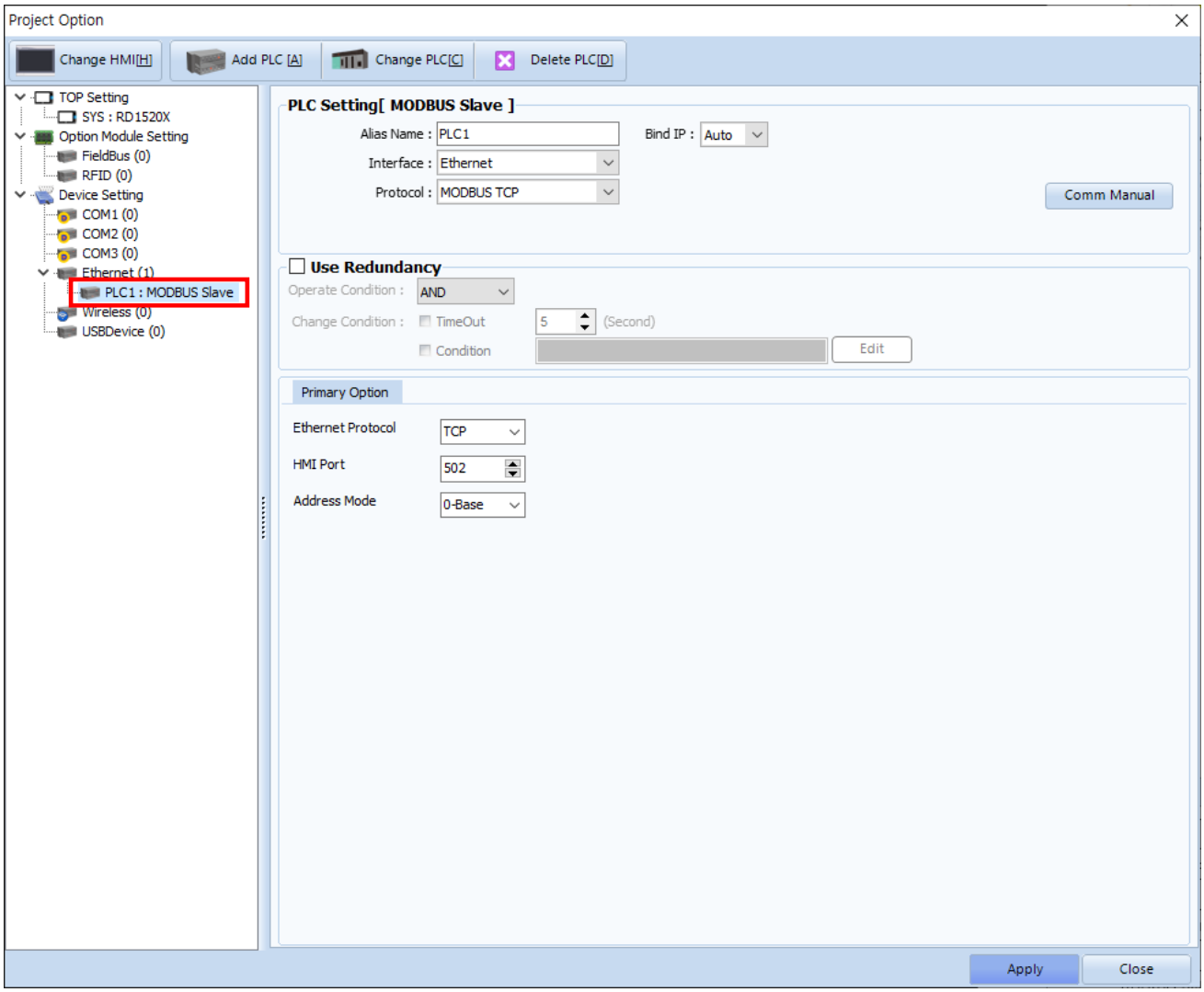
* The above settings are examples recommended by the company.

Items	Description
IP Address	Set the IP address of the TOP.
Subnet Mask	Enter the subnet mask of the network.
Gateway	Enter the gateway of the network.

(2) Communication option setting

■ [Project] → [Project Property] → [Device Setting > Ethernet > PLC1 : MODBUS Slave]

– Set the options of the MODBUS Slave communication driver in TOP Design Studio.



Items	Settings	Remarks
Interface	Select “Ethernet”.	Refer to “2. External device selection”.
Protocol	Select the communication protocol between the TOP and an external device.	
Ethernet Protocol	Enter the IP address of the external device.	
HMI Port	Set the MODBUS communication port number of TOP.	
Address Mode	Set the -1 discrepancy of the MODBUS PDU Address.	*Note 1)

*Note 1) Configure according to client specifications.

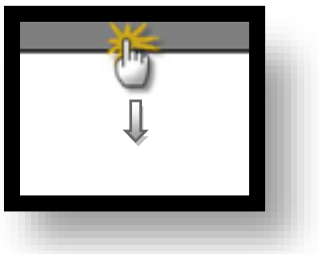
In order to read SYSO0 data of TOP, select 0-Base by requesting Address 0.

In order to read SYSO0 data of TOP, select 1-Base by requesting Address 0.

3.2. Communication setting in TOP

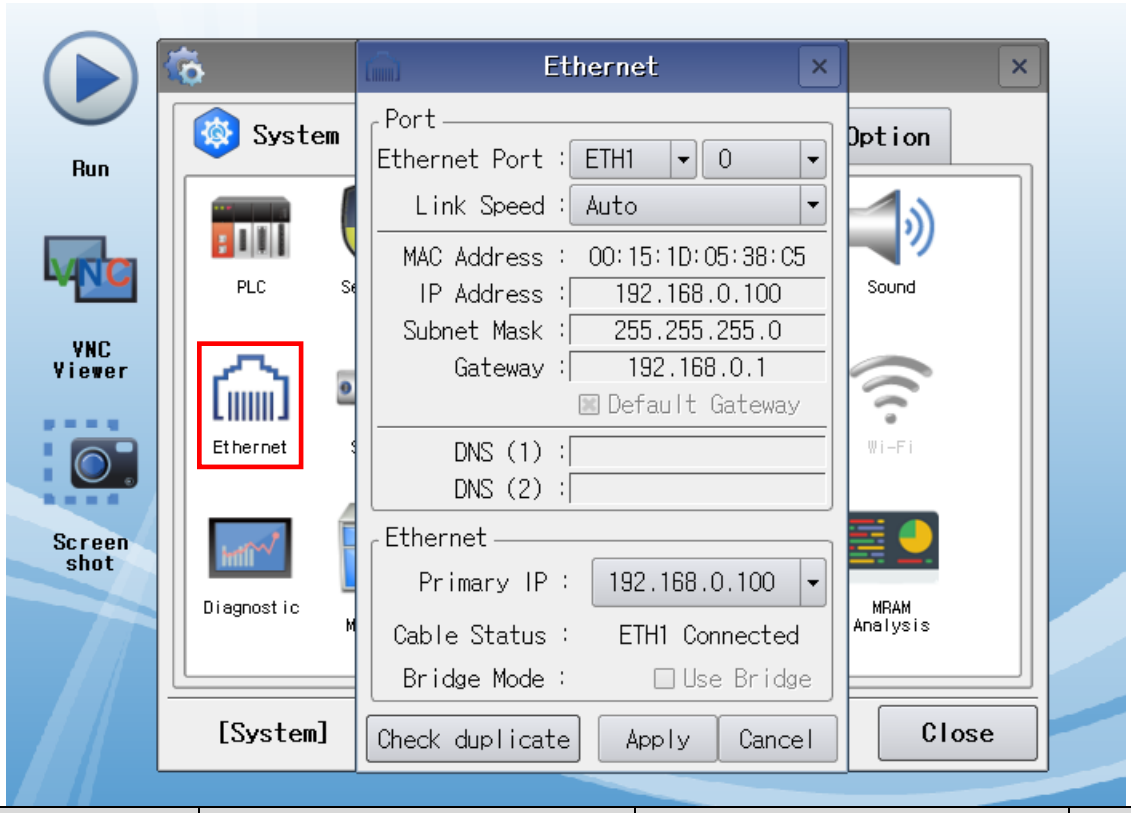
*** This is a setting method when "Use HMI Setup" in the setting items in "3.1 TOP Design Studio" is not checked.**

■ Touch the top of the TOP screen and drag it down. Touch "EXIT" in the pop-up window to go to the main screen.



(1) Communication interface setting

■ [Control Panel] → [Ethernet]



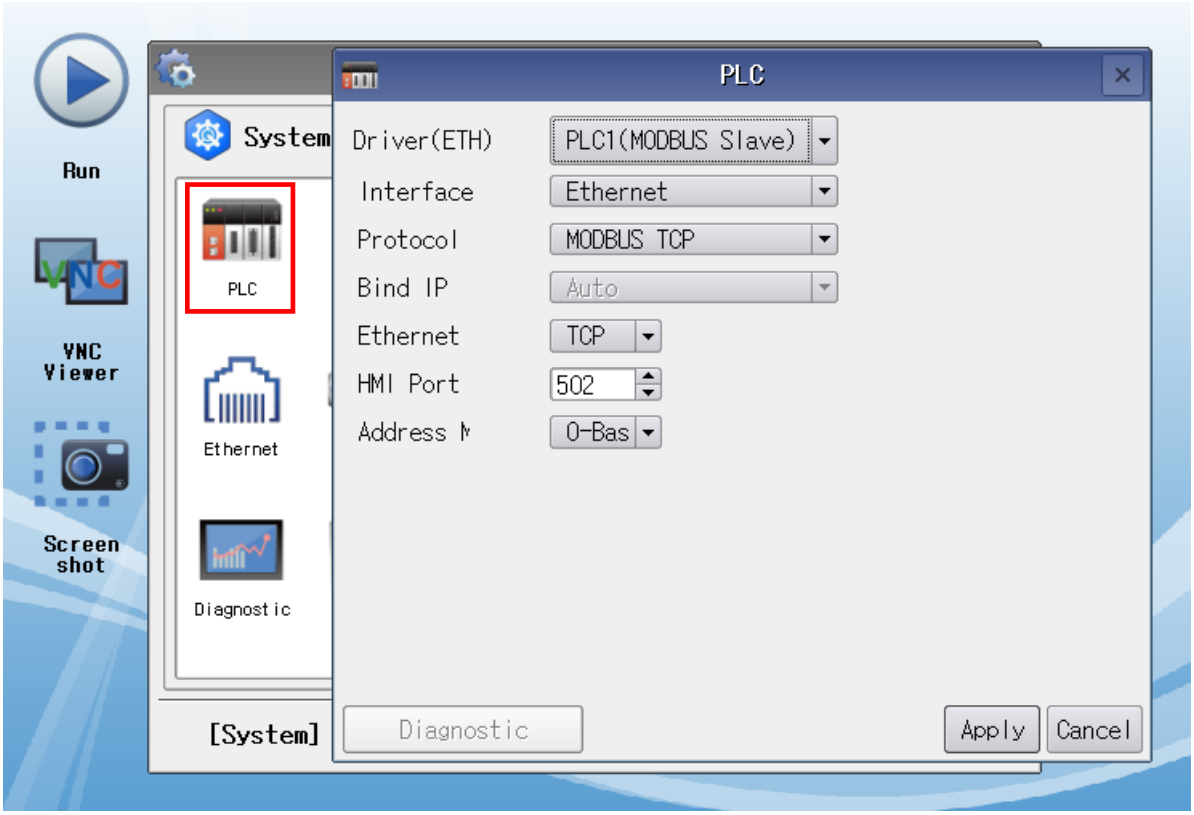
Items	TOP	External device	Remarks
IP Address	192.168.0.100	192.168.0.50	
Subnet Mask	255.255.255.0	255.255.255.0	
Gateway	192.168.0.1	192.168.0.1	

* The above settings are examples recommended by the company.

Items	Description
IP Address	Set the IP address of the TOP.
Subnet Mask	Enter the subnet mask of the network.
Gateway	Enter the gateway of the network.

(2) Communication option setting

■ [Control Panel] → [PLC]



* The above settings are examples recommended by the company.

Items	Settings	Remarks
Interface	Select "Ethernet".	Refer to "2. External device selection".
Protocol	Select the communication protocol between the TOP and an external device.	
Ethernet Protocol	Enter the IP address of the external device.	
HMI Port	Configure the HMI MODBUS communication port number.	
Address Mode	Set the -1 discrepancy of the MODBUS PDU Address.	*Note 1)

*Note 1) Configure according to client specifications.

In order to read SYSO0 data of TOP, select 0-Base by requesting Address 0.

In order to read SYSO0 data of TOP, select 1-Base by requesting Address 0.

3.3 Communication diagnostics

This driver does not support communication diagnostics.

Check the communication connection by attempting a connection and data read request from the client.
Caution) TOP must be running.

4. Supported addresses

Describes the data supported by TOP.

Address	Bit	Word	Remarks
SYS	0.0 – 10239.15	0 – 10239	*Note 1)

*Note 1) TOP-VIEW supports 0–65535.

※ TOP internal memory → MODBUS data modeling

If the TOP internal memory is expressed as MODBUS data, it counts as Holding Register.
Can be read using command 0x03, or values can be changed using command 0x06, 0x10.
Commands for accessing coil, discrete input, and input register are supported, however,
even if the commands are different, it ultimately accesses the same memory.

■ Supported Commands

Code (hex)	Descriptions
01	Read Coils
02	Read Discrete Inputs
03	Read Holding Registers
04	Read Input Registers
05	Write Single Coil
06	Write Single Register
0F	Write Multiple Coils
10	Write Multiple Registers

Appendix A. MODBUS TCP

WHAT IS MODBUS?

The MODBUS protocol was developed in 1979 by Modicon, Incorporated, for industrial automation systems and Modicon programmable controllers. It has since become an industry standard method for the transfer of discrete/analog I/O information and register data between industrial control and monitoring devices. MODBUS is now a widely-accepted, open, public-domain protocol that requires a license, but does not require royalty payment to its owner.

MODBUS devices communicate using a master-slave (client-server) technique in which only one device (the Client(Master)) can initiate

transactions (called queries). The other devices (slaves/servers) respond by supplying the requested data to the master, or by taking the action requested in the query. A slave is any peripheral device (I/O transducer, valve, network drive, or other measuring device) which processes information and sends its output to the master using MODBUS. The Acromag I/O Modules form slave/server devices, while a typical master device is a host computer running appropriate application software. Other devices may function as both clients (masters) and servers (slaves).

Masters can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a response to all queries addressed to them individually, but do not respond to broadcast queries. Slaves do not initiate messages on their own, they only respond to queries from the master.

A master's query will consist of a slave address (or broadcast address), a function code defining the requested action, any required data, and an error checking field. A slave's response consists of fields confirming the action taken, any data to be returned, and an error checking field. Note that the query and response both include a device address, a function code, plus applicable data, and an error checking field. If no error occurs, the slave's response contains the data as requested. If an error occurs in the query received, or if the slave is unable to perform the action requested, the slave will return an exception message as its response (see MODBUS Exceptions). The error check field of the slave's message frame allows the master to confirm that the contents of the message are valid. Traditional MODBUS messages are transmitted serially and parity checking is also applied to each transmitted character in its data frame.

At this point, It's important to make the distinction that MODBUS itself is an application protocol, as it defines rules for organizing and interpreting data, but remains simply a messaging structure, independent of the underlying physical layer. As it happens to be easy to understand, freely available, and accessible to anyone, it is thus widely supported by many manufacturers.

WHAT IS MODBUS TCP/IP?

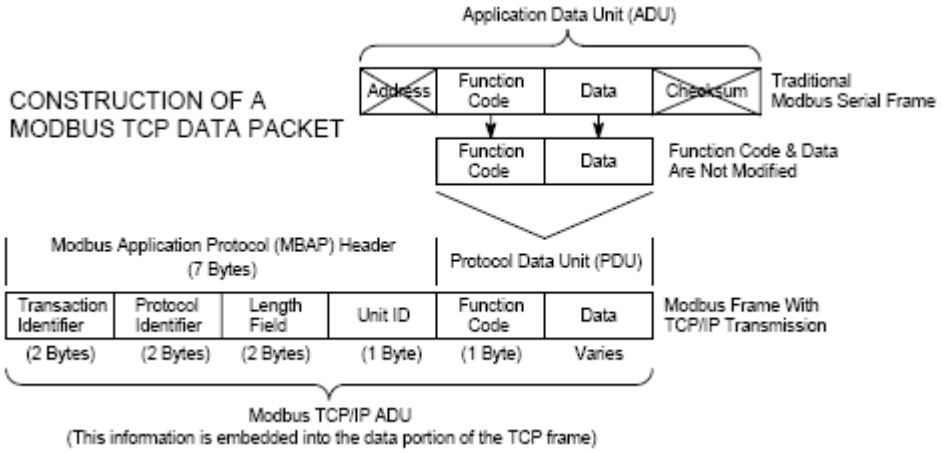
MODBUS TCP/IP (also MODBUS-TCP) is simply the MODBUS RTU protocol with a TCP interface that runs on Ethernet.
The MODBUS messaging structure is the application protocol that defines the rules for organizing and interpreting the data independent of the data transmission medium.

TCP/IP refers to the Transmission Control Protocol and Internet Protocol, which provides the transmission medium for MODBUS TCP/IP messaging.

Simply stated, TCP/IP allows blocks of binary data to be exchanged between computers. It is also a world-wide standard that serves as the foundation for the World Wide Web. The primary function of TCP is to ensure that all packets of data are received correctly, while IP makes sure that messages are correctly addressed and routed. Note that the TCP/IP combination is merely a transport protocol, and does not define what the data means or how the data is to be interpreted (this is the job of the application protocol, MODBUS in this case).

So in summary, MODBUS TCP/IP uses TCP/IP and Ethernet to carry the data of the MODBUS message structure between compatible devices. That is, MODBUS TCP/IP combines a physical network (Ethernet), with a networking standard (TCP/IP), and a standard method of representing data (MODBUS as the application protocol). Essentially, the MODBUS TCP/IP message is simply a MODBUS communication encapsulated in an Ethernet TCP/IP wrapper.

In practice, MODBUS TCP embeds a standard MODBUS data frame into a TCP frame, without the MODBUS checksum, as shown in the following diagram.



The MODBUS commands and user data are themselves encapsulated into the data container of a TCP/IP telegram without being modified in any way. However, the MODBUS error checking field (checksum) is not used, as the standard Ethernet TCP/IP link layer checksum methods are instead used to guaranty data integrity. Further, the MODBUS frame address field is supplanted by the unit identifier in MODBUS TCP/IP, and becomes part of the MODBUS Application Protocol (MBAP) header (more on this later).

From the figure, we see that the function code and data fields are absorbed in their original form. Thus, a Modbus TCP/IP Application Data Unit (ADU) takes the form of a 7 byte header (transaction identifier + protocol identifier + length field + unit identifier), and the protocol data unit (function code + data). The MBAP header is 7 bytes long and includes the following fields:

- **Transaction/invocation Identifier (2 Bytes):** This identification field is used for transaction pairing when multiple messages are sent along the same TCP connection by a client without waiting for a prior response.
- **Protocol Identifier (2 bytes):** This field is always 0 for MODBUS services and other values are reserved for future extensions.
- **Length (2 bytes):** This field is a byte count of the remaining fields and includes the unit identifier byte, function code byte, and the data fields.
- **Unit Identifier (1 byte):** This field is used to identify a remote server located on a non TCP/IP network (for serial bridging). In a typical MODBUS TCP/IP server application, the unit ID is set to 00 or FF, ignored by the server, and simply echoed back in the response.

The complete MODBUS TCP/IP Application Data Unit is embedded into the data field of a standard TCP frame and sent via TCP to well-known system port 502, which is specifically reserved for MODBUS applications. MODBUS TCP/IP clients and servers listen and receive MODBUS data via port 502.
We can see that the operation of MODBUS over Ethernet is nearly transparent to the MODBUS register/command structure. Thus, if you are already familiar with the operation of traditional MODBUS, then you are already very with the operation of MODBUS TCP/IP.

A.1 "0" Device (Coil)

(1) Read Single Coil : 01

Describes "01" command frame through the example where "**0**00020–**0**00056 Coil" data of the Slave device side (prefix: 17) is read from the MASTER device.

■ RTU Mode

(Master → Slave: request frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Device score	
	H	L	H	L	H	L			H	L	H	L
Hex	00	01	00	00	00	06	11	01	00	13	00	25

(Slave → Master: response frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Number of data (bytes)	Data				
										Coils 2720	Coils 3528	Coils 4336	Coils 5144	Coils 5652
	H	L	H	L	H	L				L	–	–	–	H
Hex	00	01	00	00	00	08	11	01	05	CD	6B	B2	0E	1B

■ Coils data status

Coils	27	26	25	24	23	22	21	20
on/off	1	1	0	0	1	1	0	1
Coils	35	34	33	32	31	30	29	28
on/off	0	1	1	0	1	0	1	1
Coils	43	42	41	40	39	38	37	36
on/off	1	0	1	1	0	0	1	0
Coils	51	50	49	48	47	46	45	44
on/off	0	0	0	0	1	1	1	0
Coils	59	58	57	56	55	54	53	52
on/off	–	–	–	1	1	0	1	1

0: OFF /

(2) Force Single Coil : 05

Describes "05" command frame through an example where FORCE "ON" is done on Coil 000173 of the Slave device side in the MASTER device.

■ RTU Mode

(Master → Slave: request frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Force data	
	H	L	H	L	H	L			H	L	H	L
Hex	00	02	00	00	00	06	11	05	00	AC	FF	00

■ Force Data

	High	Low
Force ON	FF_H	00_H
Force OFF	00_H	00_H

(Slave → Master: response frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Force data	
	H	L	H	L	H	L			H	L	H	L
Hex	00	02	00	00	00	06	11	05	00	AC	FF	00

A.2 "1" Device (Discrete Input)

(1) Read Input Status : 02

Describes "02" command frame through an example where "**1**00197–**1**00218 Input" data of the Slave device side (prefix: 17) is read from the MASTER device.

(Master → Slave: request frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Device score	
	H	L	H	L	H	L			H	L	H	L
Hex	00	03	00	00	00	06	11	02	00	C4	00	16

(Slave → Master: response frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Number of data	Data (Inputs)		
										10204-10197	10212-10205	10218-10213
	H	L	H	L	H	L						
Hex	00	03	00	00	00	06	11	02	03	AC	DB	35

■ Coils data status

Coils	204	203	202	201	200	199	198	197
on/off	1	0	1	0	1	1	0	0
Coils	212	211	210	209	208	207	206	205
on/off	1	1	0	1	1	0	1	1
Coils	220	219	218	217	216	215	214	213
on/off	–	–	1	1	0	1	0	1

0: OFF / 1:ON

A.3 "3" Device (Input Register)

(1) Read Input Registers : 04

Describes "03" command frame through an example where "**3**00009 Register" data of the Slave device side (prefix: 17) is read from the MASTER device.

(Master → Slave: request frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Device score (Word Count)	
	H	L	H	L	H	L			H	L	H	L
Hex	00	04	00	00	00	06	11	04	00	08	00	01

(Slave → Master: response frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Number of data	Data 30009 Register	
	H	L	H	L	H	L				H	L
Hex	00	04	00	00	00	05	11	04	02	00	0A

A.4 "4" Device (Holding Register)

(1) Read Holding Registers : 03

Describes "03" command frame through an example where "**4**00108 – **4**00110 Register" data of the Slave device side (prefix: 17) is read from the MASTER device.

(Master → Slave: request frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Device score	
	H	L	H	L	H	L			H	L	H	L
Hex	00	05	00	00	00	06	11	03	00	6B	00	03

(Slave → Master: response frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Number of data	Data					
										Register 40108		Register 40109		Register 40110	
	H	L	H	L	H	L				H	L	H	L	H	L
Hex	00	05	00	00	00	09	11	03	06	02	2B	00	00	00	64

(2) Preset Single Register : 06

Describes "06" command frame through an example where 00 03 (hex) data is entered in **4**00002 Register of the Slave device side .

(Master → Slave: request frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Preset data	
	H	L	H	L	H	L			H	L	H	L
Hex	00	06	00	00	00	06	11	06	00	01	00	03

(Slave → Master: response frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Preset data	
	H	L	H	L	H	L			H	L	H	L
Hex	00	06	00	00	00	06	11	06	00	01	00	03

(3) Preset Multiple Register : 10

Describes "10" command frame through an example where two consecutive data, "00 0A (hex)", "01 02 (hex)" are entered in **4**00002 Register of the Slave device side. (Error Code : 90_H)

(Master → Slave: request frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Quantity of Register (Word Count)		Number of data	Data			
														40002 Register		40003 Register	
	H	L	H	L	H	L			H	L	H	L		H	L	H	L
Hex	00	07	00	00	00	0B	11	10	00	01	00	02	04	00	0A	01	02

(Slave → Master: response frame)

Comment	Transaction Identifier		Protocol Identifier		Length Field		Unit ID (Slave prefix)	Command	Leading device		Quantity of Register (Word Count)	
	H	L	H	L	H	L			H	L	H	L
Hex	00	07	00	00	00	06	11	10	00	01	00	02