MODBUS Organization
MODBUS Master Series
MODBUS Serial Master Driver

Supported version TOP Design Studio V1.4.5 or higher

CONTENTS

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

1. System configuration

This driver is the "Serial Master Driver" among the "MODBUS Protocol" of "MODBUS Organization".

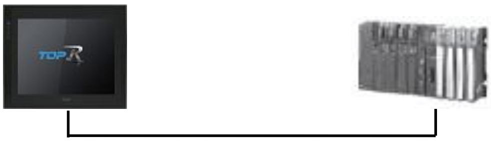
Depending on the external device (MODBUS Slave Protocol supported), you may set the "command code", "protocol frame format" etc., of the driver separately. In this case, set the detailed settings according to the external device side based on the communication method.

The system configuration with an external device supported by this driver is as follows:

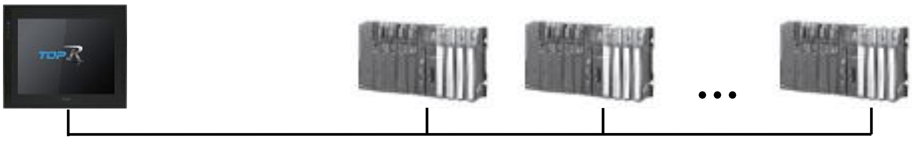
Series	CPU	Link I/F	Communication method	System setting	Cable
MODBUS Slave Device			RS-232C	3. TOP communication setting 4. External device setting	5. Cable table
			RS-422 (4 wire)		
			RS-485 (2 wire)		

■ Connectable configuration

- 1:1 connection

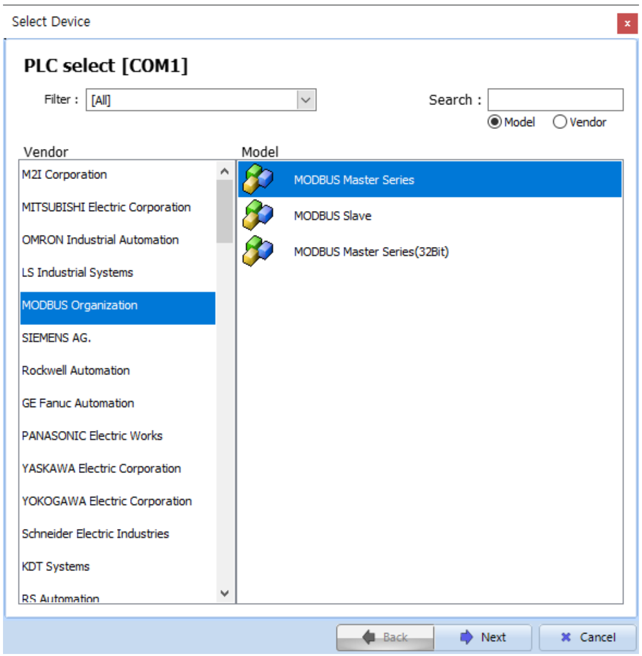


- 1:N connection



2. External device selection

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



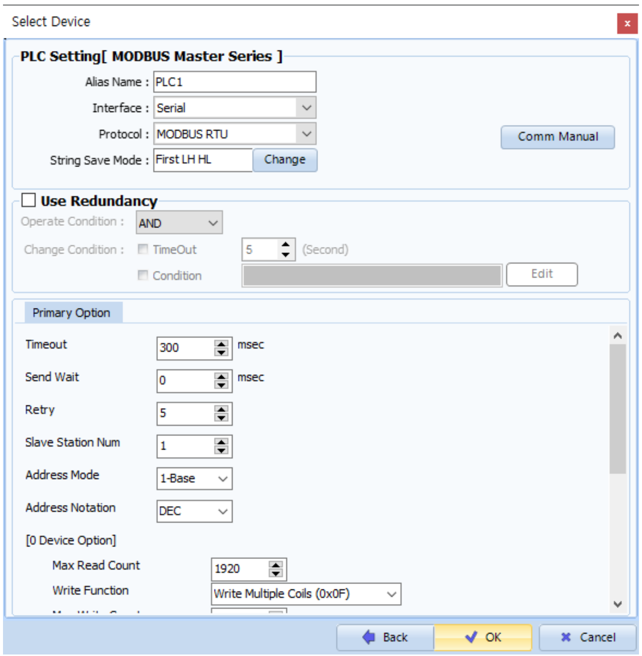
Select Device

PLC select [COM1]

Filter : [All]   Search :
● Model ○ Vendor

Vendor:
- M2I Corporation
- MITSUBISHI Electric Corporation
- OMRON Industrial Automation
- LS Industrial Systems
- MODBUS Organization
- SIEMENS AG.
- Rockwell Automation
- GE Fanuc Automation
- PANASONIC Electric Works
- YASKAWA Electric Corporation
- YOKOGAWA Electric Corporation
- Schneider Electric Industries
- KDT Systems
- RS Automation

Model:
- MODBUS Master Series
- MODBUS Slave
- MODBUS Master Series(32Bit)



Select Device

PLC Setting[MODBUS Master Series]

Alias Name : PLC1
Interface : Serial
Protocol : MODBUS RTU
String Save Mode : First LH HL Change

☐ Use Redundancy
Operate Condition : AND
Change Condition : ☐ TimeOut 5 (Second)
☐ Condition Edit

Primary Option

Timeout	300	msec
Send Wait	0	msec
Retry	5	
Slave Station Num	1	
Address Mode	1-Base	
Address Notation	DEC	
[0 Device Option]		
Max Read Count	1920	
Write Function	Write Multiple Coils (0x0F)	

Settings		Contents
TOP	Model	Select the TOP model.
External device	Vendor	Select the vendor of the external device to be connected to the TOP. Select “MODBUS Organization”.
	PLC	Select the external device to be connected to the TOP.

Model	Interface	Protocol
MODBUS Master Series	Serial	Set Users

Supported Protocol	
MODBUS RTU	MODBUS ASCII

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

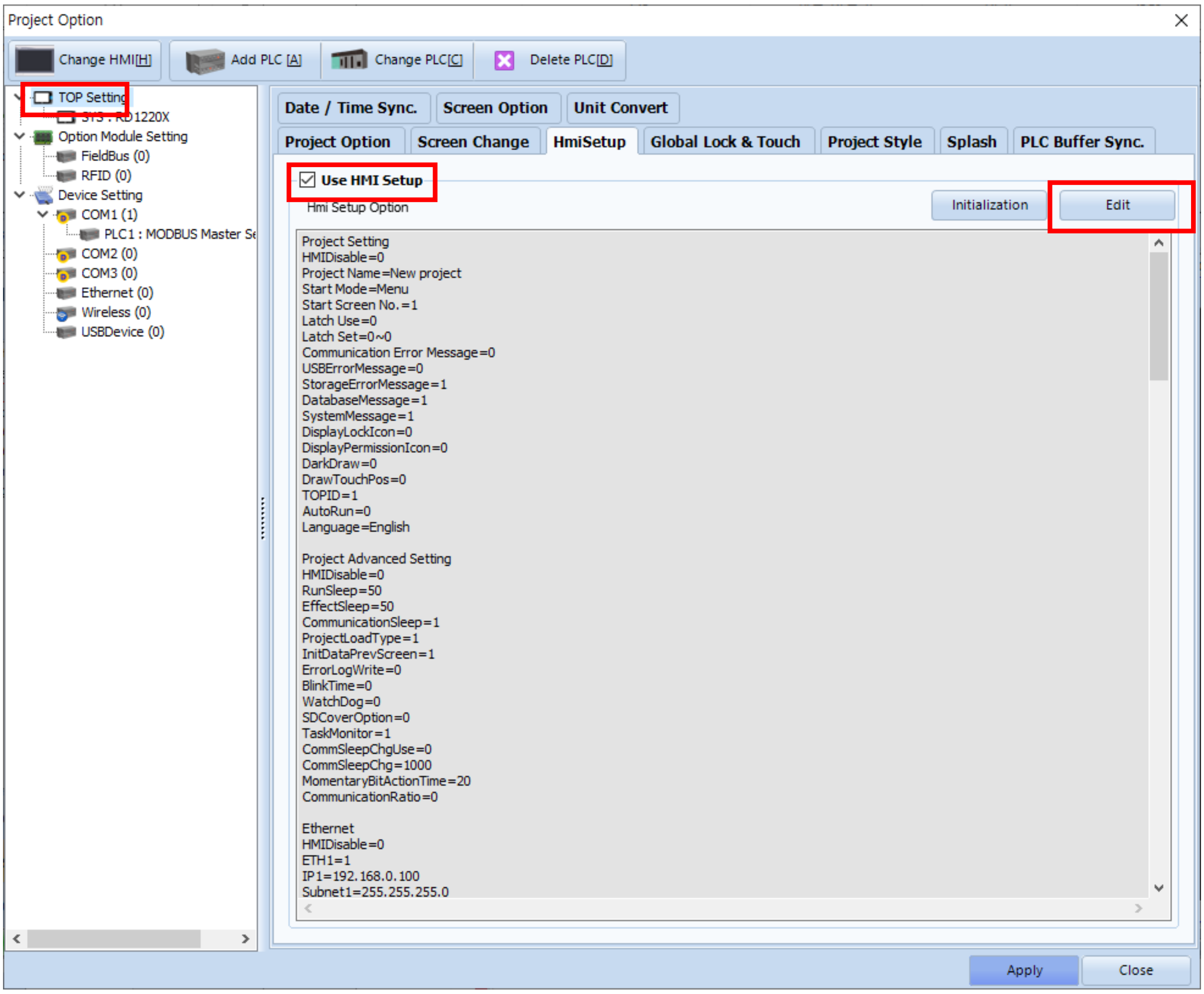
The communication can be set in TOP Design Studio or TOP main menu. The communication should be set in the same way as that of the external device.

3.1 Communication setting in TOP Design Studio

(1) Communication interface setting

■ [Project] → [Properties] → [TOP Settings] → [HMI Setup] → [Check Use HMI Setup] → [Edit] → [Serial]

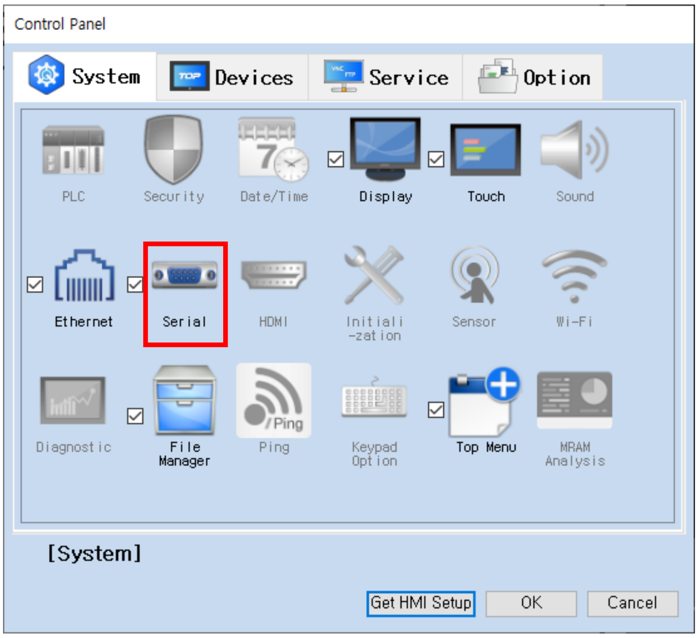
– Set the TOP communication interface in TOP Design Studio.

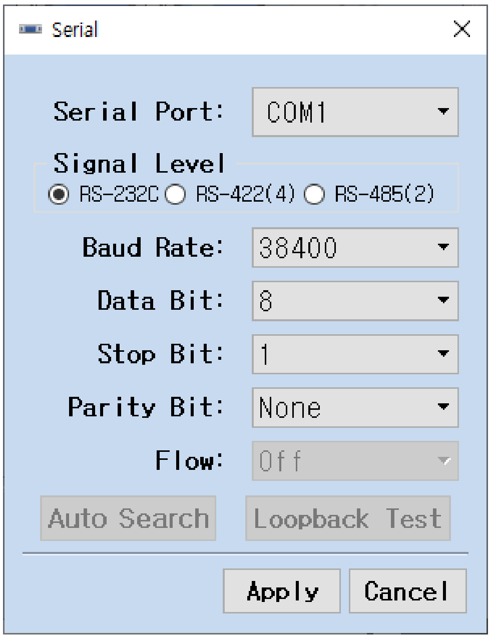


```
Project Setting
HMIDisable=0
Project Name=New project
Start Mode=Menu
Start Screen No.=1
Latch Use=0
Latch Set=0~0
Communication Error Message=0
USBErrorMessage=0
StorageErrorMessage=1
DatabaseMessage=1
SystemMessage=1
DisplayLockIcon=0
DisplayPermissionIcon=0
DarkDraw=0
DrawTouchPos=0
TOPID=1
AutoRun=0
Language=English

Project Advanced Setting
HMIDisable=0
RunSleep=50
EffectSleep=50
CommunicationSleep=1
ProjectLoadType=1
InitDataPrevScreen=1
ErrorLogWrite=0
BlinkTime=0
WatchDog=0
SDCoverOption=0
TaskMonitor=1
CommSleepChgUse=0
CommSleepChg=1000
MomentaryBitActionTime=20
CommunicationRatio=0

Ethernet
HMIDisable=0
ETH1=1
IP1=192.168.0.100
Subnet1=255.255.255.0
```





Serial Port: COM1

Signal Level ● RS-232C ○ RS-422(4) ○ RS-485(2)

Baud Rate: 38400

Data Bit: 8

Stop Bit: 1

Parity Bit: None

Flow: Off

Items	TOP	External device	Remarks
Signal Level	RS-232C RS-422/485	RS-232C RS-422/485	
Baud Rate	38400		
Data Bit	8		
Stop Bit	1		
Parity Bit	None.		

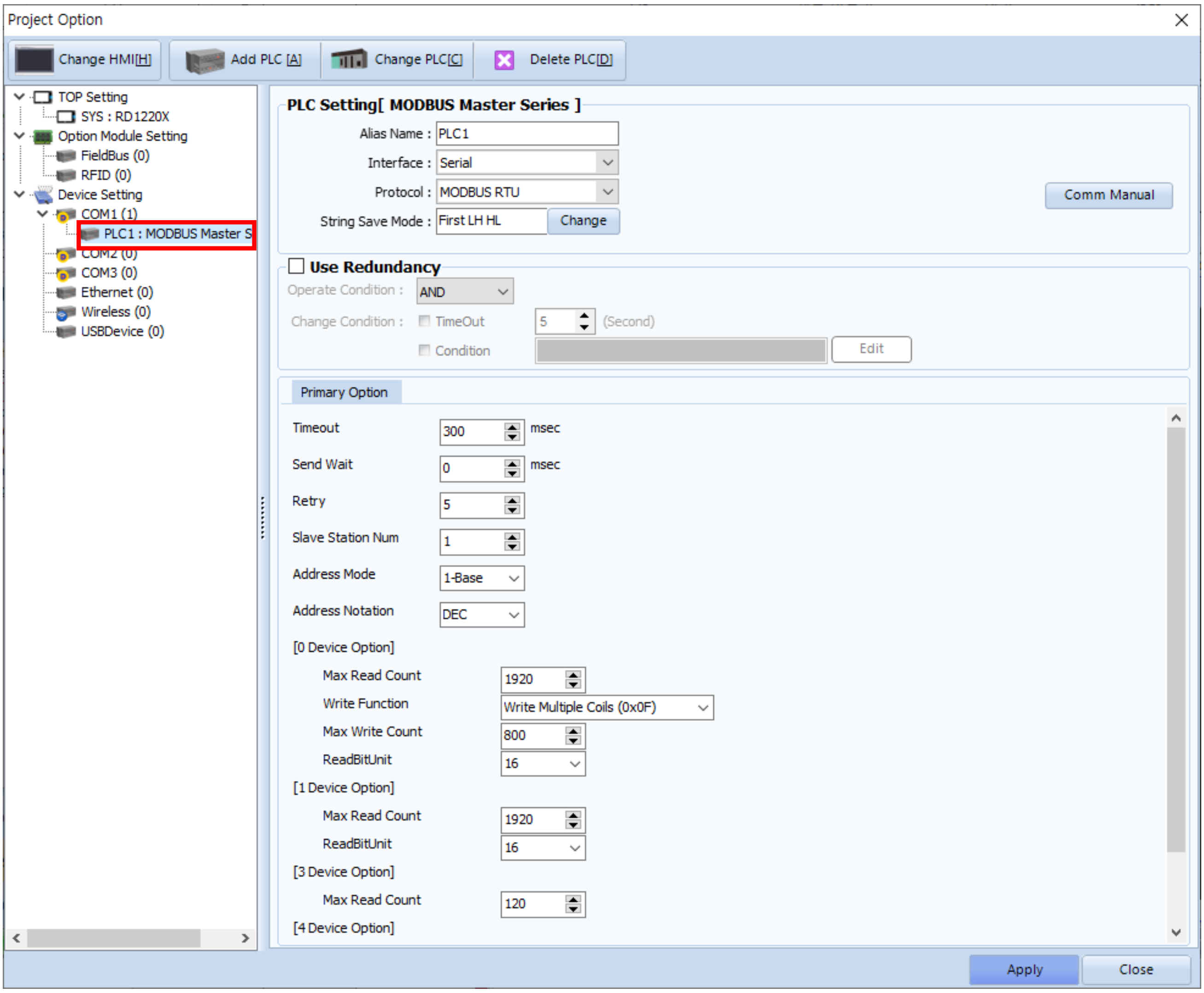
* The above settings are examples recommended by the company.

Items	Description
Signal Level	Select the serial communication method between the TOP and an external device. (COM3 supports only RS-485.)
Baud Rate	Select the serial communication speed between the TOP and an external device.
Data Bit	Select the serial communication data bit between the TOP and an external device.
Stop Bit	Select the serial communication stop bit between the TOP and an external device.
Parity Bit	Select the serial communication parity bit check method between the TOP and an external device.

(2) Communication option setting

■ [Project] → [Project Property] → [Device Setting > COM1 > PLC1 : MODBUS Master Series]

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



PLC Setting[MODBUS Master Series]

Alias Name : PLC1

Interface : Serial

Protocol : MODBUS RTU

String Save Mode : First LH HL Change

☐ Use Redundancy

Operate Condition : AND

Change Condition : ☐ TimeOut 5 (Second)

☐ Condition

Timeout	300	msec
Send Wait	0	msec
Retry	5	
Slave Station Num	1	
Address Mode	1-Base	
Address Notation	DEC	
[0 Device Option]		
Max Read Count	1920	
Write Function	Write Multiple Coils (0x0F)	
Max Write Count	800	
ReadBitUnit	16	
[1 Device Option]		
Max Read Count	1920	
ReadBitUnit	16	
[3 Device Option]		
Max Read Count	120	
[4 Device Option]		

Items		Settings	Remarks
Interface		Select "Serial".	Refer to "2. External device selection".
Protocol		Select the communication protocol between the TOP and an external device.	
String Save Mode		Set the byte order of data when entering the string data.	
Redundancy	Use Redundancy	Check whether redundancy settings are used or not.	
	Operation Condition	Set the operation condition for the change condition. AND: change Primary ↔ Secondary if all change conditions checked are satisfied. OR: change Primary ↔ Secondary if any of change conditions checked are satisfied.	
	Change Condition	Set Primary ↔ Secondary change condition.	
TimeOut (ms)		Set the time for the TOP to wait for a response from an external device.	
SendWait (ms)		Set the waiting time between TOP's receiving a response from an external device and sending the next command request.	
Retry		Set the number of request retries when the data request result is no response/negative response.	
Slave Station Num		Enter the prefix of an external device.	
Address Mode		Select the address input method. 1-base: The memory address of an device at 1. Request data to registered address-1. 0-base: The memory address of an device at 0. Request data to registered address.	
Address Notation		Select the address notation.	

[0 Device Option]	**Coil**	
Max Read Count	Set the maximum count at which a request can be made at one time when coil read is requested.	*Note 1) Note 2)
Write Function	Set the coil write request command. 0x05 : Force Single Coil (Write in 1-bit unit. Only bit unit operation can be used.) 0x0F : Force Multiple Coils (Write in 16-bit unit) Auto : Request as 0x05 or 0x0F depending on the number of data.	*Note 3)
Max Write Count	Sets the maximum count at which a request can be made at one time when Coil Write is requested.	*Note 2)
Read Bit Unit	Sets the number of bits requested when Coil Write is requested. If the set value is 16 and the address following the screen is registered, data is requested up to "Max Read Count" at one time.	
[1 Device Option]	**Discrete Input**	
Max Read Count	Set the maximum count at which a request can be made at one time when Discrete Input is requested.	*Note 1) Note 2)
Read Bit Unit	Sets the number of bits requested when Discrete Input is requested. If the set value is 16 and the address following the screen is registered, data is requested up to "Max Read Count" at one time.	
[3 Device Option]	**Input Register**	
Max Read Count	Set the maximum count at which a request can be made at one time when Input Register Read is requested.	*Note 1) Note 2)
[4 Device Option]	**Holding Register**	
Max Read Count	Set the maximum count at which a request can be made at one time when Holding Register Read is requested.	*Note 1)
Write Function	Set the Holding Register write request command. 0x06 : Preset Single Register (write 1) 0x10 : Preset Multiple Registers (write n) Auto : Request as 0x06 or 0x10 depending on the number of data.	*Note 3)
Max Write Count	Set the maximum count at which a request can be made at one time when requesting Holding Register data write with command 0x10.	*Note 2)

*Note 1)

- The Max Read Count of each device is also used as the address range to requested at one time without communicating several times when the addresses registered on the screen are not consecutive.

Ex. 1) If 400001, 400002, 400003, 400004, 400005, 400120 are registered as number objects on the screen, and the Max Read Count of 4 devices is set to 120, assume that the addresses are consecutive from 400001 through 400120 and read the data in 120 words from 400001 at one request.

Ex. 2) If 400001, 400002, 400003, 400004, 400005, 400120 are registered as number objects on the screen, and the Max Read Count of 4 devices is set to 3, read the data in 3 words from 400001 to 400003, 2 words from 400004 to 400005, and 1 word for 400120 at three requests.

Ex. 3) If 400001, 400010, 400011, 400021, 400031, 400041 are registered as number objects on the screen, and the Max Read Count of 4 devices is set to 10, read the data in 10 words from 400001 to 400010, 1 word for 400011, 1 word for 400021 and 1 word for 400041 at five requests.

- If Max Read Count is set to 0, up to 120 words only for consecutive addresses are requested.

*Note 2)

- Refer to the manual for the external device to check how many data can be read/written from the registered address at a time.

If a setting is made larger than the range supported by an external device, communication is not made normally.

Ex. If the Holding Register (4 devices) of an external device can respond only in the maximum of 10 words in one communication, set the Max Read Count of 4 devices to 10 according to the specification of an external device among the communication setting of the TOP.

*Note 3)

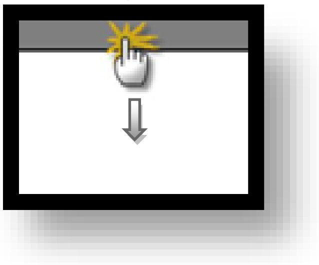
- Refer to the manual of the external device and set it according to the supported write command.

If you set a write command which is not supported, data write operation is not made.

3.2. Communication setting in TOP

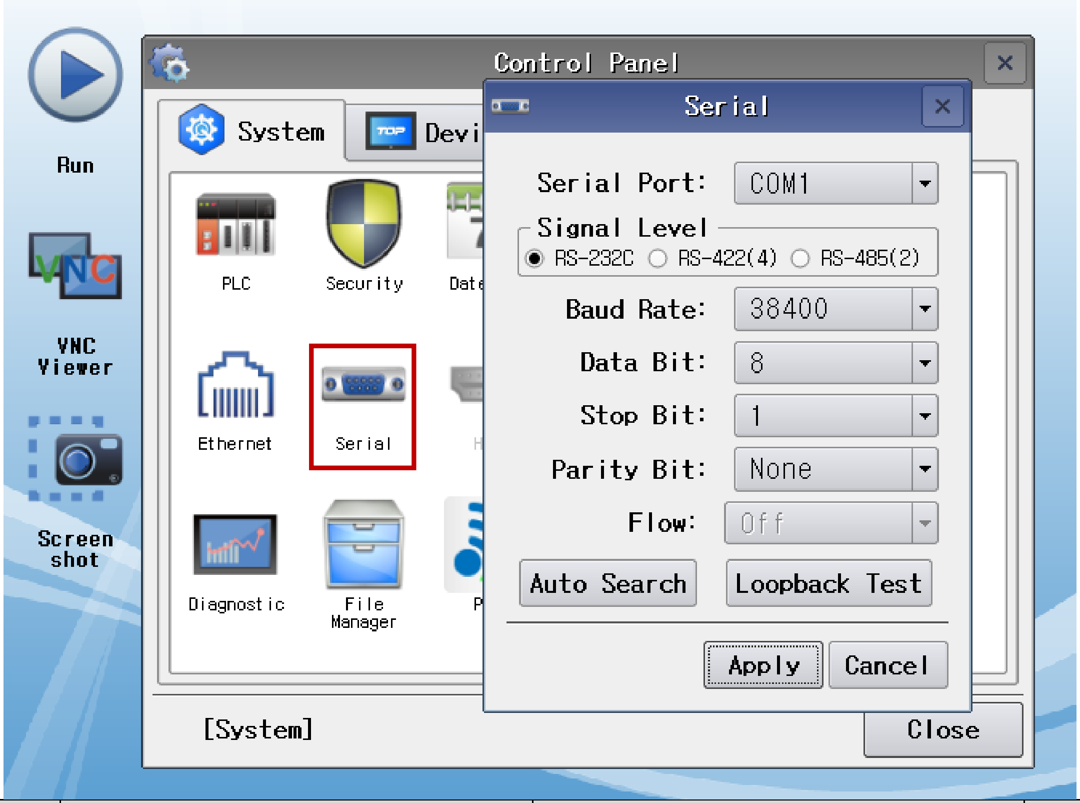
*** This is a setting method when "Use HMI Setup" in the setting items in "3.1 TOP Design Studio" is not checked.**

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



(1) Communication interface setting

■ [Control Panel] → [Serial]



Serial Port: COM1

Signal Level: ● RS-232C ○ RS-422(4) ○ RS-485(2)

Baud Rate: 38400

Data Bit: 8

Stop Bit: 1

Parity Bit: None

Flow: Off

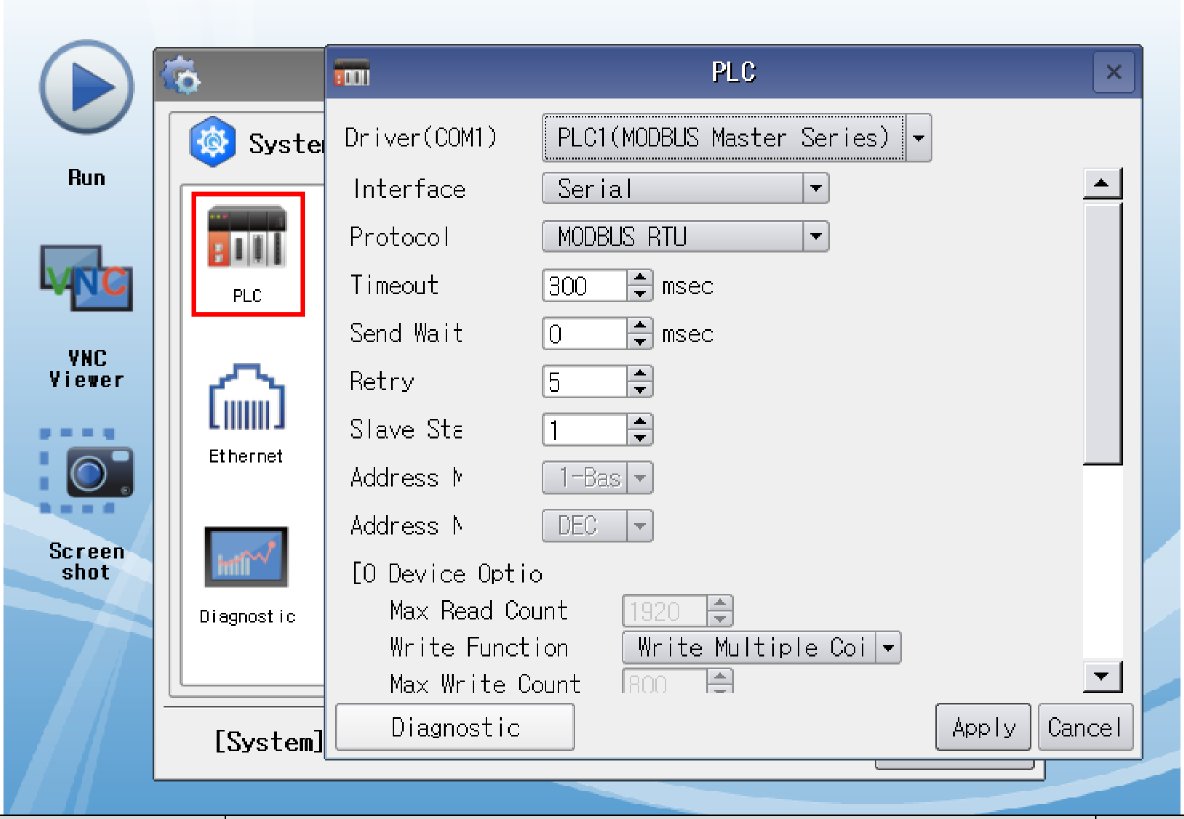
Items	TOP	External device	Remarks
Signal Level	RS-232C RS-422/485	RS-232C RS-422/485	
Baud Rate	38400		
Data Bit	8		
Stop Bit	1		
Parity Bit	None.		

* The above settings are examples recommended by the company.

Items	Description
Signal Level	Select the serial communication method between the TOP and an external device. (COM3 supports only RS-485.)
Baud Rate	Select the serial communication speed between the TOP and an external device.
Data Bit	Select the serial communication data bit between the TOP and an external device.
Stop Bit	Select the serial communication stop bit between the TOP and an external device.
Parity Bit	Select the serial communication parity bit check method between the TOP and an external device.

(2) Communication option setting

■ [Control Panel] → [PLC]



Driver(COM1) PLC1(MODBUS Master Series)
Interface Serial
Protocol MODBUS RTU
Timeout 300 msec
Send Wait 0 msec
Retry 5
Slave Sta 1
Address M 1-Bas
Address N DEC
[0 Device Optio
Max Read Count 1920
Write Function Write Multiple Coi
Max Write Count 800

Items		Settings	Remarks
Interface		Select "Serial".	Refer to "2. External device selection".
Protocol		Select the communication protocol between the TOP and an external device.	
String Save Mode		Set the byte order of data when entering the string data.	
Redundancy	Use Redundancy	Check whether redundancy settings are used or not.	
	Operation Condition	Set the operation condition for the change condition. AND: change Primary ↔ Secondary if all change conditions checked are satisfied. OR: change Primary ↔ Secondary if any of change conditions checked are satisfied.	
	Change Condition	Set Primary ↔ Secondary change condition.	
TimeOut (ms)		Set the time for the TOP to wait for a response from an external device.	
SendWait (ms)		Set the waiting time between TOP's receiving a response from an external device and sending the next command request.	
Retry		Set the number of request retries when the data request result is no response/negative response.	
Slave Station Num		Enter the prefix of an external device.	
Address Mode		Select the address input method. 1-base: The memory address of an device at 1. Request data to registered address -1. 0-base: The memory address of an device at 0. Request data to registered address.	
Address Notation		Select the address notation.	
[0 Device Option]		**Coil**	
Max Read Count		Set the maximum count at which a request can be made at one time when coil read is requested.	*Note 1) Note 2)
Write Function		Set the coil write request command. 0x05 : Force Single Coil (Write in 1-bit unit. Only bit unit operation can be used.) 0x0F : Force Multiple Coils (Write in 16-bit unit) Auto : Request as 0x05 or 0x0F depending on the number of data.	*Note 3)
Max Write Count		Sets the maximum count at which a request can be made at one time when Coil	*Note 2)

	Write is requested.	
Read Bit Unit	Sets the number of bits requested when Coil Write is requested. If the set value is 16 and the address following the screen is registered, data is requested up to "Max Read Count" at one time.	
[1 Device Option]	**Discrete Input**	
Max Read Count	Set the maximum count at which a request can be made at one time when Discrete Input is requested.	*Note 1) Note 2)
Read Bit Unit	Sets the number of bits requested when Discrete Input is requested. If the set value is 16 and the address following the screen is registered, data is requested up to "Max Read Count" at one time.	
[3 Device Option]	**Input Register**	
Max Read Count	Set the maximum count at which a request can be made at one time when Input Register Read is requested.	*Note 1) Note 2)
[4 Device Option]	**Holding Register**	
Max Read Count	Set the maximum count at which a request can be made at one time when Holding Register Read is requested.	*Note 1)
Write Function	Set the Holding Register write request command. 0x06 : Preset Single Register (write 1) 0x10 : Preset Multiple Registers (write n) Auto : Request as 0x06 or 0x10 depending on the number of data.	*Note 3)
Max Write Count	Set the maximum count at which a request can be made at one time when requesting Holding Register data write with command 0x10.	*Note 2)

*Note 1)

- The Max Read Count of each device is also used as the address range to requested at one time without communicating several times when the addresses registered on the screen are not consecutive.

Ex. 1) If 400001, 400002, 400003, 400004, 400005, 400120 are registered as number objects on the screen, and the Max Read Count of 4 devices is set to 120, assume that the addresses are consecutive from 400001 through 400120 and read the data in 120 words from 400001 at one request.

Ex. 2) If 400001, 400002, 400003, 400004, 400005, 400120 are registered as number objects on the screen, and the Max Read Count of 4 devices is set to 3, read the data in 3 words from 400001 to 400003, 2 words from 400004 to 400005, and 1 word for 400120 at three requests.

Ex. 3) If 400001, 400010, 400011, 400021, 400031, 400041 are registered as number objects on the screen, and the Max Read Count of 4 devices is set to 10, read the data in 10 words from 400001 to 400010, 1 word for 400011, 1 word for 400021 and 1 word for 400041 at five requests.

- If Max Read Count is set to 0, up to 120 words only for consecutive addresses are requested.

*Note 2)

- Refer to the manual for the external device to check how many data can be read/written from the registered address at a time.
If a setting is made larger than the range supported by an external device, communication is not made normally.
Ex. If the Holding Register (4 devices) of an external device can respond only in the maximum of 10 words in one communication, set the Max Read Count of 4 devices to 10 according to the specification of an external device among the communication setting of the TOP.

*Note 3)

- Refer to the manual of the external device and set it according to the supported write command.
If you set a write command which is not supported, data write operation is not made.

3.3 Communication diagnostics

■ Check the interface setting status between the TOP and external device.

- Touch the top of the TOP screen and drag it down. Touch "EXIT" in the pop-up window to go to the main screen.
- Check that the settings of the connected ports in [Control Panel] → [Serial] are the same as the settings of the external device.

■ Diagnosis of whether the port communication is normal or not

- Touch "Communication Diagnostics" in [Control Panel] → [PLC].
- Check whether communication is connected or not.

Communication diagnostics succeeded	**Communication setting normal**
Error message	**Communication setting abnormal** - Check the cable, TOP, and external device settings. (Refer to Communication diagnostics sheet.)

■ Communication diagnostics sheet

- If there is a problem with the communication connection with an external terminal, please check the settings in the sheet below.

Items	Contents		Check		Remarks
System configuration	How to connect the system		OK	NG	1. System configuration
	Cable		OK	NG	
TOP	Version information		OK	NG	2. External device selection 3. TOP communication setting
	Communication port		OK	NG	
	Communication driver and protocol		OK	NG	
	Other detailed settings		OK	NG	
	Relative prefix	Project setting	OK	NG	
		Communication diagnostics	OK	NG	
	Serial Parameter	Transmission Speed	OK	NG	
		Data Bit	OK	NG	
		Stop Bit	OK	NG	
		Parity Bit	OK	NG	
External device	CPU name		OK	NG	4. External device setting
	Communication port		OK	NG	
	Protocol		OK	NG	
	Prefix		OK	NG	
	Other detailed settings		OK	NG	
	Serial Parameter	Transmission Speed	OK	NG	
		Data Bit	OK	NG	
		Stop Bit	OK	NG	
		Parity Bit	OK	NG	
	Check address range		OK	NG	6. Supported addresses (For details, please refer to the PLC vendor's manual.)

4. External device setting

Refer to the vendor's user manual to set as a Modbus Slave (server).

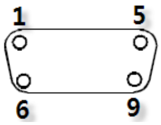
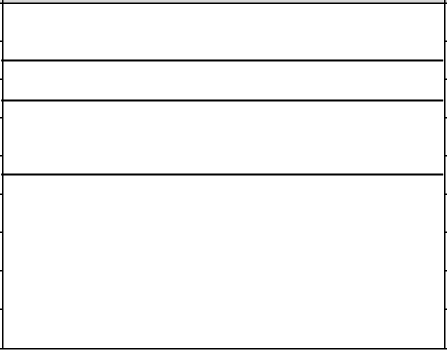
⚠

- Take caution when selecting RTU/ASCII during protocol setting.
- Check the memory address of the external device.

5. Cable table

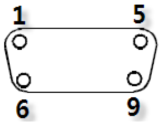
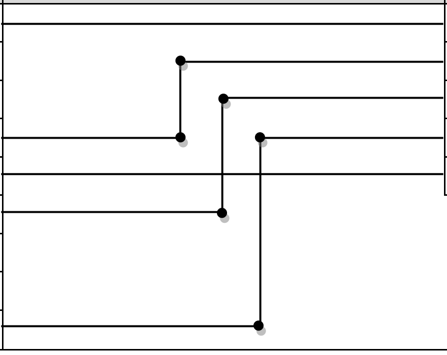
This chapter introduces a cable diagram for normal communication between the TOP and the corresponding device.
(The cable diagrams in this section may differ from the external device vendor's recommendations.)

■ RS-232C (1:1 connection)

COM1 / COM2			Cable connection	External device	
Pin arrangement*Note 1)	Signal name	Pin number		Signal name	
 Based on communication cable connector front, D-SUB 9 Pin male (male, convex)	CD	1			
	RD	2		SD	
	SD	3		RD	
	DTR	4		DTR	
	SG	5		SG	
	DSR	6		DSR	
	RTS	7		RTS	
	CTS	8		CTS	
		9			

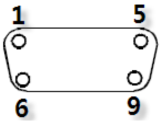
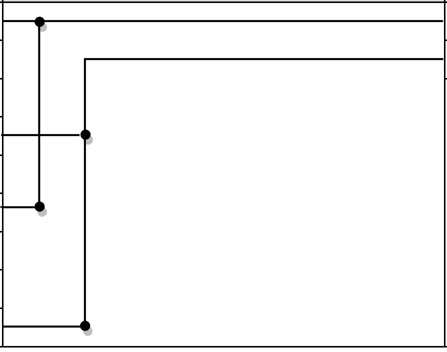
*Note 1) The pin arrangement is as seen from the connecting side of the cable connection connector.

■ RS-422 (1:1 connection)

COM1 / COM2			Cable connection	External device	
Pin arrangement*Note 1)	Signal name	Pin number		Signal name	
 Based on communication cable connector front, D-SUB 9 Pin male (male, convex)	RDA(+)	1		SDA(+)	
		2		SDB(-)	
		3		RDA(+)	
	RDB(-)	4		RDB(-)	
	SG	5		SG	
	SDA(+)	6			
		7			
		8			
	SDB(-)	9			

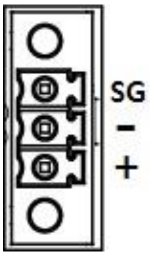
*Note 1) The pin arrangement is as seen from the connecting side of the cable connection connector.

■ RS-485 (1:1 connection)

COM1 / COM2			Cable connection	External device	
Pin arrangement*Note 1)	Signal name	Pin number		Signal name	
 Based on communication cable connector front, D-SUB 9 Pin male (male, convex)	RDA(+)	1		+	
		2		-	
		3			
	RDB(-)	4			
	SG	5			
	SDA(+)	6			
		7			
		8			
	SDB(-)	9			

*Note 1) The pin arrangement is as seen from the connecting side of the cable connection connector.

■ **RS-485** (1:1 connection)

COM3		Cable connection	PLC	
Pin arrangement	Signal name		Signal name	
	+		+	
	-		-	
	SG			

■ **RS-422** (1:N connection)

TOP	Cable connection and signal direction	External device	Cable connection and signal direction	External device
Signal name		Signal name		Signal name
RDA(+)		SDA(+)		SDA(+)
RDB(-)		SDB(-)		SDB(-)
SDA(+)		RDA(+)		RDA(+)
SDB(-)		RDB(-)		RDB(-)
SG		SG		SG

■ **RS-485** (1:N connection)

TOP	Cable connection and signal direction	External device	Cable connection and signal direction	External device
Signal name		Signal name		Signal name
RDA(+)		+		+
RDB(-)		-		-
SDA(+)				
SDB(-)				
SG				

6. Supported addresses

The devices available in TOP are as follows:

The device range (address) may differ depending on the CPU module series/type. The TOP series supports the maximum address range used by the external device series. Please refer to each CPU module user manual and be take caution to not deviate from the address range supported by the device you want to use.

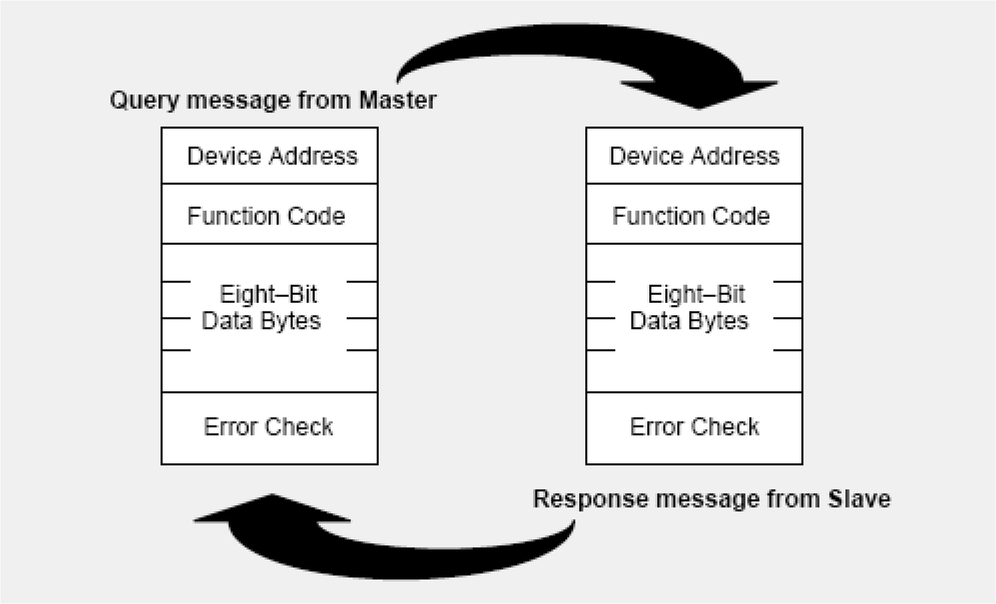
	Bit	Word	Remarks
Coil	000001 – 065536	000001 – 065521	
Discrete Input	100001 – 165536	100001 – 165521	*Note 1)
Input Register	300001.00 – 365536.15	300001 – 365536	*Note 1)
Holding Register	400001.00 – 465536.15	400001 – 465536	

***Note 1)** Read-only

Appendix A. Standard MODBUS Protocol

Describes MODBUS protocol commands and devices supported by "MODBUS Serial Master Driver" of this device.

At the message level, the MODBUS protocol still applies the master–slave principle even though the network communication method is peer–to–peer. If a controller originates a message, it does so as a master device, and expects a response from a slave device. Similarly, when a controller receives a message it constructs a slave response and returns it to the originating controller.



The Query: The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain any additional information that the slave will need to perform the function. For example, function code 03 will query the slave to read holding registers and respond with their contents. The data field must contain the information telling the slave which register to start at and how many registers to read. The error check field provides a method for the slave to validate the integrity of the message contents.

The Response: If the slave makes a normal response, the function code in the response is an echo of the function code in the query. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that describes the error. The error check field allows the master to confirm that the message contents are valid.

A.1 “0” Device (Coil)

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	Slave prefix	Command	Leading device		Device score		Check code (CRC)	
			H	L	H	L	L	H
Hex	11	01	00	13	00	25	—	—

(Slave → Master: response frame)

Comment	Slave prefix	Command	Number of data	Data					Check code (CRC)	
				Coils 27~20	Coils 35~28	Coils 43~36	Coils 51~44	Coils 56~52		
				L	–	–	–	H	L	H
Hex	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 / 1:ON

■ ASCII Mode

(Master → Slave: request frame)

comment	Header	Slave prefix		Command		Leading device				Device score				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	1	0	0	1	3	0	0	2	5			CR	LF
Hex	3A	31	31	30	31	30	30	31	33	30	30	32	35	—	—	0D	0A

(Slave → Master: response frame)

Comment	Header	Slave prefix		Command		Number of data (bytes)		Data										Check code (LRC)		Tail	
								Coils 27~20		Coils 35~28		Coils 43~36		Coils 51~44		Coils 56~52					
								H	L	H	L	H	L	H	L	H	L	L	H		
ASCII	:	1	1	0	1	0	5	C	D	6	B	B	2	0	E	1	B			CR	LF
Hex	3A	31	31	30	31	30	35	43	44	36	42	42	32	30	45	31	42	—	—	0D	0A

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	Slave prefix	Command	Leading device		Force data		Check code (CRC)	
			H	L	H	L	L	H
Hex	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	Slave prefix	Command	Leading device		Force data		Check code (CRC)	
			H	L	H	L	L	H
Hex	11	05	00	AC	FF	00	—	—

■ ASCII Mode

(Master → Slave: request frame)

comment	Header	Slave prefix		Command		Leading device				Force data				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	5	0	0	1	3	0	0	2	5			CR	LF
Hex	3A	31	31	30	31	30	30	41	43	45	45	30	30	—	—	0D	0A

(Slave → Master: response frame)

comment	Header	Slave prefix		Command		Leading device				Force data				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	5	0	0	1	3	0	0	2	5			CR	LF
Hex	3A	31	31	30	31	30	30	41	43	45	45	30	30	—	—	0D	0A

A.2 “1” Device (Discrete Input)

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.

■ RTU Mode

(Master → Slave: request frame)

Comment	Slave prefix	Command	Leading device		Device score		Check code (CRC)	
			H	L	H	L	L	H
Hex	11	02	00	C4	00	16	—	—

(Slave → Master: response frame)

Comment	Slave prefix	Command	Number of data	Data (Inputs)			Check code (CRC)	
				10204~10197	10212~10205	10218~10213	L	H
Hex	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

■ ASCII Mode

(Master → Slave: request frame)

comment	Header	Slave prefix		Command		Leading device				Device score				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	2	0	0	C	4	0	0	1	6			CR	LF
Hex	3A	31	31	30	32	30	30	43	34	30	30	31	36	—	—	0D	0A

(Slave → Master: response frame)

Comment	Header	Slave prefix		Command		Number of data (bytes)		Data (Inputs)						Check code (LRC)		Tail	
								10204~10197		10212~10205		10218~10213					
								H	L	H	L	H	L	L	H		
ASCII	:	1	1	0	2	0	3	A	C	D	B	3	5			CR	LF
Hex	3A	31	31	30	31	30	35	41	43	44	42	33	35	—	—	0D	0A

A.3 “3” Device (Input Register)

Read Input Registers : 04

Describes “03” command frame through an example where “300009 Register” data of the Slave device side (prefix: 17) is read from the MASTER device.

■ RTU Mode

(Master → Slave: request frame)

Comment	Slave prefix	Command	Leading device		Device score (Word Count)		Check code (CRC)	
			H	L	H	L	L	H
Hex	11	04	00	08	00	01	—	—

(Slave → Master: response frame)

Comment	Slave prefix	Command	Number of data (bytes)	Data Register 30009		Check code (CRC)	
				H	L	L	H
Hex	11	04	02	00	0A	—	—

■ ASCII Mode

(Master → Slave: request frame)

comment	Header	Slave prefix		Command		Leading device				Device score (Word)				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	1	0	0	0	8	0	0	0	1			CR	LF
Hex	3A	31	31	30	31	30	30	30	38	30	30	30	31	—	—	0D	0A

(Slave → Master: response frame)

Comment	Header	Slave prefix		Command		Number of data (bytes)		Data Register 40108				Check code (LRC)		Tail	
								H	–	–	L	L	H		
ASCII	:	1	1	0	4	0	2	0	0	0	A			CR	LF
Hex	3A	31	31	30	31	30	35	30	30	30	41	—	—	0D	0A

A.4 "4" Device (Holding Register)

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.

■ RTU Mode

(Master → Slave: request frame)

Comment	Slave prefix	Command	Leading device		Device score		Check code (CRC)	
			H	L	H	L	L	H
Hex	11	03	00	6B	00	03	—	—

(Slave → Master: response frame)

Comment	Slave prefix	Command	Number of data (bytes)	Data						Check code (CRC)	
				Register 40108		Register 40109		Register 40110			
				H	L	H	L	H	L	L	H
Hex	11	03	06	02	2B	00	00	00	64	—	—

■ ASCII Mode

(Master → Slave: request frame)

comment	Header	Slave prefix		Command		Leading device				Device score (Word)				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	1	0	0	1	3	0	0	2	5			CR	LF
Hex	3A	31	31	30	31	30	30	31	33	30	30	32	35	—	—	0D	0A

(Slave → Master: response frame)

Comment	Header	Slave prefix		Command		Number of data (bytes)		Data												Check code (LRC)		Tail	
								Register 40108				Register 40109				Register 40110							
								H	–	–	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	3	0	6	0	2	2	B	0	0	0	0	0	0	6	4			CR	LF
Hex	3A	31	31	30	31	30	35	30	32	32	42	30	30	30	30	30	30	36	34	—	—	0D	0A

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 .

■ RTU Mode

(Master → Slave: request frame)

Comment	Slave prefix	Command	Leading device		Preset data		Check code (CRC)	
			H	L	H	L	L	H
Hex	11	06	00	01	00	03	—	—

(Slave → Master: response frame)

Comment	Slave prefix	Command	Leading device		Preset data		Check code (CRC)	
			H	L	H	L	L	H
Hex	11	06	00	01	00	03	—	—

■ ASCII Mode

(Master → Slave: request frame)

comment	Header	Slave prefix		Command		Leading device				Preset data				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	6	0	0	0	1	0	0	0	3			CR	LF
Hex	3A	31	31	30	36	30	30	30	31	30	30	30	33	—	—	0D	0A

(Slave → Master: response frame)

comment	Header	Slave prefix		Command		Leading device				Preset data				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	0	6	0	0	0	1	0	0	0	3			CR	LF
Hex	3A	31	31	30	36	30	30	30	31	30	30	30	33	—	—	0D	0A

Preset Multiple Register : 10

Describes "10" command frame through an example where two consecutive data, "00 0A (hex)", "01 02 (hex)" are entered in 400002 Register of the Slave device side. (Error Code : 90$_H$)

■ RTU Mode

(Master → Slave: request frame)

Comment	Slave prefix	Command	Leading device		Quantity of Register (Word Count)		Number of data	Data				Check code (CRC)	
								Register 40002		Register 40003			
			H	L	H	L		H	L	H	L	L	H
Hex	11	10	00	01	00	02	04	00	0A	01	02	—	—

(Slave → Master: response frame)

Comment	Slave prefix	Command	Leading device		Quantity of Register (Word Count)		Check code (CRC)	
			H	L	H	L	L	H
Hex	11	10	00	01	00	02	—	—

■ ASCII Mode

(Master → Slave: request frame)

comment	Header	Slave prefix		Command		Leading device				Quantity of Register (Word Count)				Number of data (bytes)		Data							
																Register 40002				Register 40003			
		H	L	H	L	H	–	–	L	H	–	–	L	–	L	H	–	–	L	H	–	–	L
ASCII	:	1	1	1	0	0	0	0	1	0	0	0	2	0	4	0	0	0	A	0	1	0	2
Hex	3A	31	31	31	30	30	30	41	43	30	30	30	32	30	34	30	30	30	41	30	31	30	32

Continued...

	Check code (LRC)		Tail	
	L	H		
ASCII			CR	LF
Hex	—	—	0D	0A

(Slave → Master: response frame)

comment	Header	Slave prefix		Command		Leading device				Quantity of Register (Word Count)				Check code (LRC)		Tail	
		H	L	H	L	H	–	–	L	H	–	–	L	L	H		
ASCII	:	1	1	1	0	0	0	0	1	0	0	0	2			CR	LF
Hex	3A	31	31	30	31	30	30	30	31	30	30	30	32	—	—	0D	0A

A.5 LRC/CRC Generation

(1) LRC Generation

The Longitudinal Redundancy Check (LRC) field is one byte, containing an 8–bit binary value. The LRC value is calculated by the transmitting device, which appends the LRC to the message. The receiving device recalculates an LRC during receipt of the message, and compares the calculated value to the actual value it received in the LRC field. If the two values are not equal, an error results.

The LRC is calculated by adding together successive 8–bit bytes in the message, discarding any carries, and then two's complementing the result. The LRC is an 8–bit field, therefore each new addition of a character that would result in a value higher than 255 decimal simply 'rolls over' the field's value through zero. Because there is no ninth bit, the carry is discarded automatically.

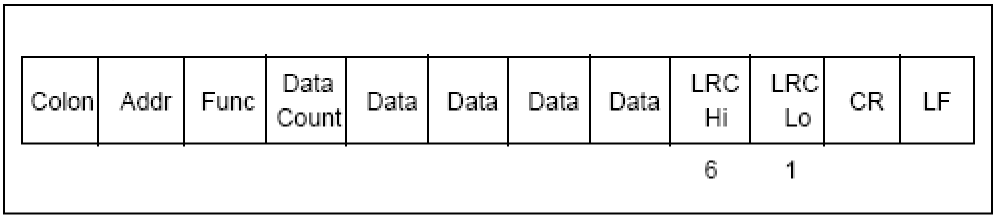
A procedure for generating an LRC is:

1. Add all bytes in the message, excluding the starting 'colon' and ending
CRLF. Add them into an 8–bit field, so that carries will be discarded.
2. Subtract the final field value from FF hex (all 1's), to produce the
ones–complement.
3. Add 1 to produce the twos–complement.

– Placing the LRC into the Message

When the 8–bit LRC (2 ASCII characters) is transmitted in the message, the high–order character will be transmitted first, followed by the low–order character.

For example, if the LRC value is 61 hex (0110 0001):



Colon	Addr	Func	Data Count	Data	Data	Data	Data	LRC Hi	LRC Lo	CR	LF
								6	1		

– Example

An example of a C language function performing LRC generation is shown below.

The function takes two arguments:

```
unsigned char *auchMsg ;                // A pointer to the message buffer containing
                                       // binary data to be used for generating the LRC
unsigned short usDataLen ;              // The quantity of bytes in the message buffer.
```

The function returns the LRC as a type unsigned char.

– LRC Generation Function

```
static unsigned char LRC(auchMsg, usDataLen)
unsigned char *auchMsg ;                    /* message to calculate LRC upon */
unsigned short usDataLen ;                  /* quantity of bytes in message */
{
   unsigned char uchLRC = 0 ;               /* LRC char initialized */
   while (usDataLen—)                       /* pass through message buffer */
   uchLRC += *auchMsg++ ;                   /* add buffer byte without carry */
   return ((unsigned char)(–((char)uchLRC))) ;   /* return twos complement */
}
```

(2) CRC Generation

The Cyclical Redundancy Check (CRC) field is two bytes, containing a 16–bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The CRC is started by first preloading a 16–bit register to all 1's. Then a process begins of applying successive 8–bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit, do not apply to the CRC.

During generation of the CRC, each 8–bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8–bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, is the CRC value.

A procedure for generating a CRC is:

1. Load a 16–bit register with FFFF hex (all 1's). Call this the CRC register.
2. Exclusive OR the first 8–bit byte of the message with the low–order byte of the 16–bit CRC register, putting the result in the CRC register.
3. Shift the CRC register one bit to the right (toward the LSB), zero–filling the MSB. Extract and examine the LSB.
4. (If the LSB was 0): Repeat Step 3 (another shift). (If the LSB was 1): Exclusive OR the CRC register with the polynomial value A001 hex (1010 0000 0000 0001).
5. Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8–bit byte will have been processed.
6. Repeat Steps 2 through 5 for the next 8–bit byte of the message. Continue doing this until all bytes have been processed.
7. The final contents of the CRC register is the CRC value.
8. When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

– Placing the CRC into the Message

When the 16–bit CRC (two 8–bit bytes) is transmitted in the message, the low-order byte will be transmitted first, followed by the high-order byte.

For example, if the CRC value is 1241 hex (0001 0010 0100 0001):

Addr	Func	Data Count	Data	Data	Data	Data	CR CLo	CR CHi
							41	12

– Example

An example of a C language function performing CRC generation is shown on the following pages. All of the possible CRC values are preloaded into two arrays, which are simply indexed as the function increments through the message buffer.

One array contains all of the 256 possible CRC values for the high byte of the 16–bit CRC field, and the other array contains all of the values for the low byte. Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

Note This function performs the swapping of the high/low CRC bytes internally. The bytes are already swapped in the CRC value that is returned from the function. Therefore the CRC value returned from the function can be directly placed into the message for transmission.

The function takes two arguments:

```
unsigned char *puchMsg ;                  //A pointer to the message buffer containing
                                          //binary data to be used for generating the CRC
unsigned short usDataLen ;                //The quantity of bytes in the message buffer.
```

The function returns the CRC as a type unsigned short.

– **CRC Generation Function**

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ;                         /* message to calculate CRC upon */
unsigned short usDataLen ;                       /* quantity of bytes in message */
{
    unsigned char uchCRCHi = 0xFF ;              /* high byte of CRC initialized */
    unsigned char uchCRCLo = 0xFF ;              /* low byte of CRC initialized */
    unsigned uIndex ;                            /* will index into CRC lookup table */
    while (usDataLen—)                           /* pass through message buffer */
    {
        uIndex = uchCRCHi ^ *puchMsgg++ ;        /* calculate the CRC */
        uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex} ;
        uchCRCLo = auchCRCLo[uIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

– **High-Order Byte Table**

```
/* Table of CRC values for high–order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
```

– **Low-Order Byte Table**

```
/* Table of CRC values for low–order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;
```