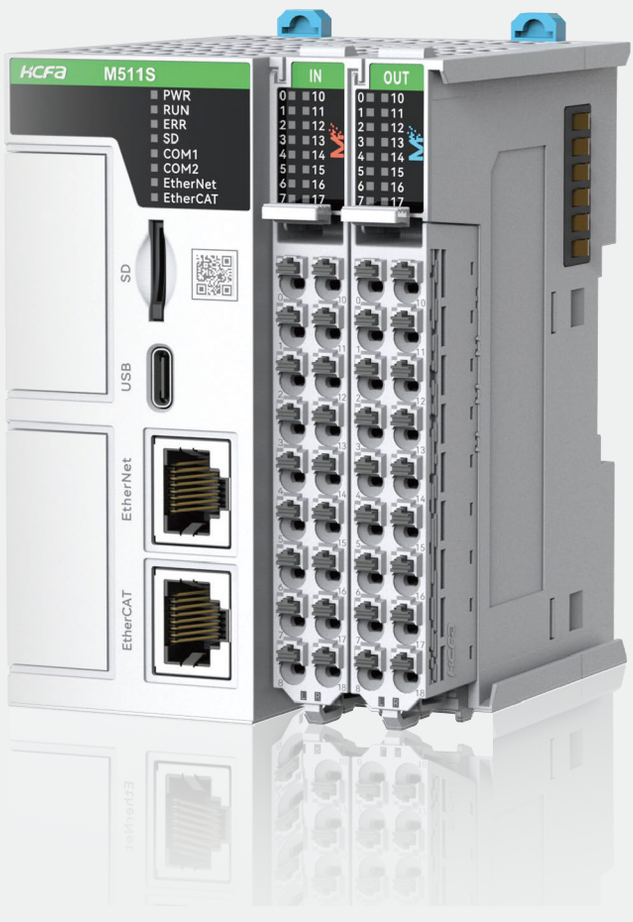
M-Series Controller

Communication

Instruction Manual



May 28, 2024 V1.00
Version:ATC/MMXC032410

※ Contents

※ Preface

Thank you very much for purchasing M series controllers. This manual mainly introduces the controller communication instructions, such as Ethernet, RS485, RS232, CAN, and other communication instructions.

- **Intended users**

This manual is intended for technicians who program and debug the M series motion controllers. Users need to have some basic knowledge and a programming mindset related to programmable controllers.

- **Manual revision information**

Version	Date	Content
V1.00	2024/03/06	First edition

Copyright statement

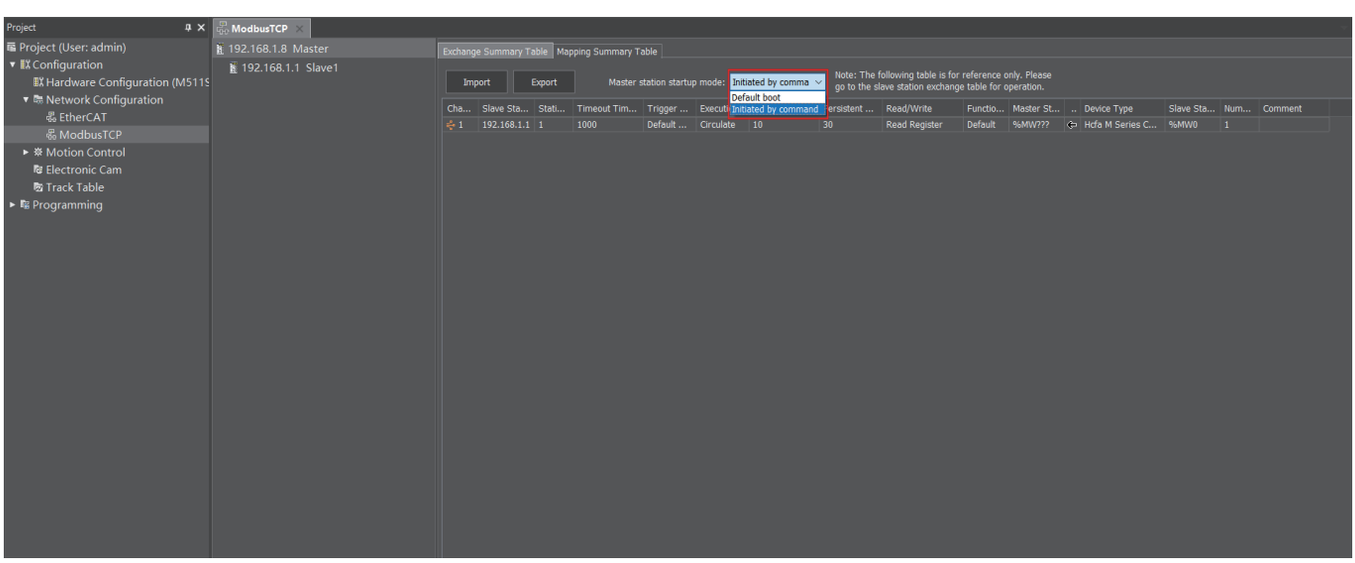
Chapter 1 Ethernet communication

1.1 ModbusTCP communication usage overview

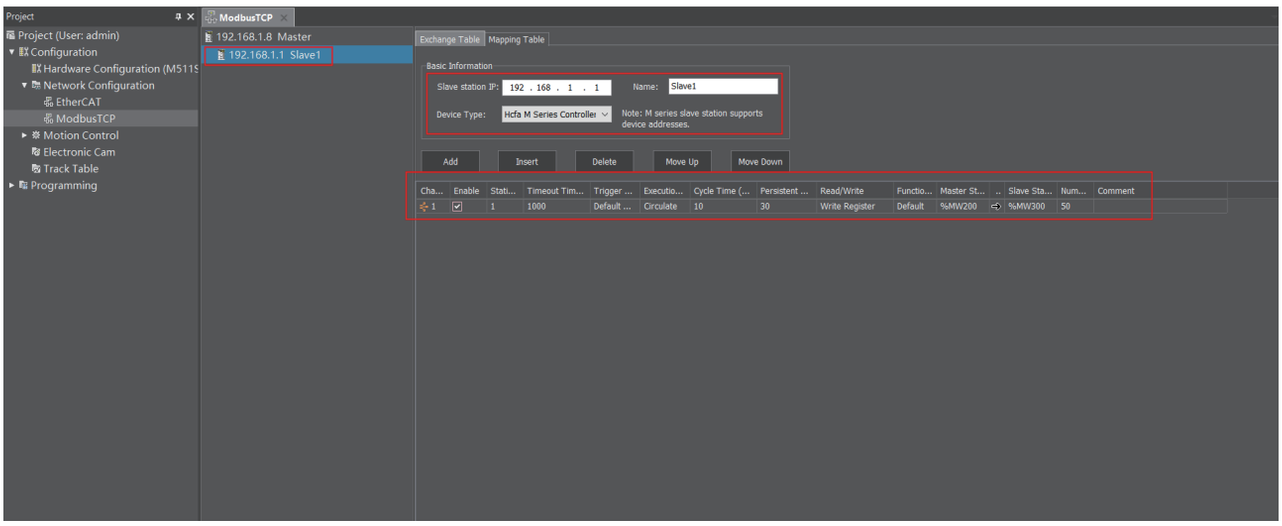
Software configuration using ModbusTCP functionality:

Step	Item	Usage		Description
1	Control master	Select “enable by instructions” in the software configuration interface	ModbusTCP_MasterRun	Enable ModbusTCP master function
		Select “enable by default” in the software configuration interface	Enable by default	Enable ModbusTCP master function
		ModbusTCP_MasterStop		Disable ModbusTCP master function
		ModbusTCP_Masterstatus		Get ModbusTCP master status
2	Configure channels	Software configuration interface		Configure the specified data exchange channels
3	Control channels	Select “trigger by program” in the software configuration interface	ModbusTCP_LinkRun	Open the specified data exchange channels
		Select “trigger by default” in the software configuration interface	Enable by default	Open the specified data exchange channels
		ModbusTCP_LinkStop		Close the specified data exchange channels
		ModbusTCP_GetLinkStatus		Get the status of the specified data exchange channels

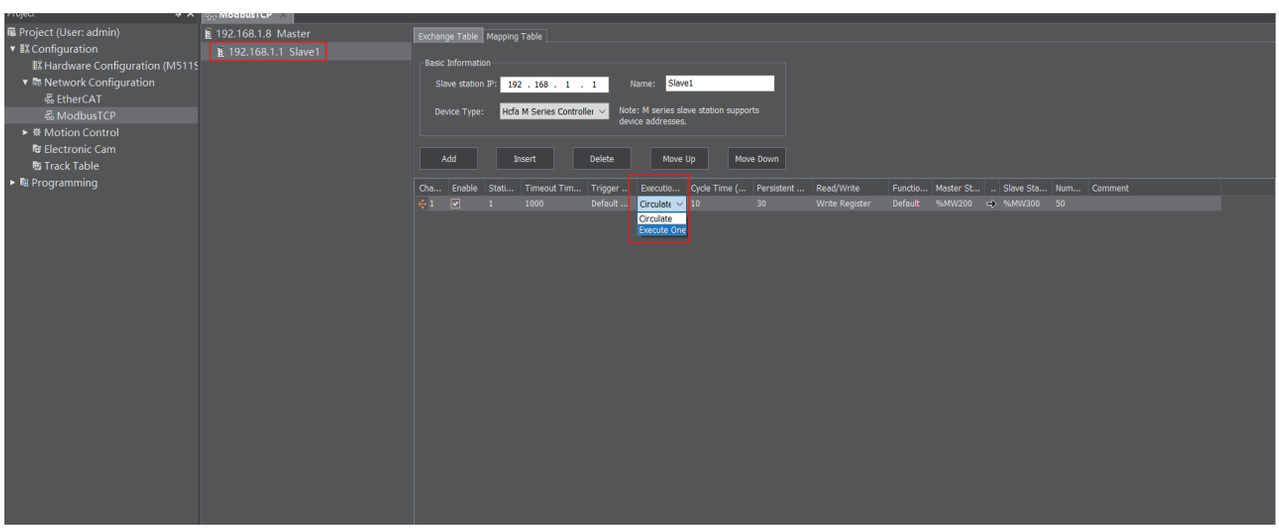
Step 1: After enabling the operation of the ModbusTCP master function master switch, ModbusTCP-related functions can be used. The software configuration can be enabled by default (default operation) or by the ModbusTCP_MasterRun instruction (executed after the instruction is triggered), disabled by the ModbusTCP_MasterStop instruction, and get the ModbusTCP master status by the ModbusTCP_Masterstatus instruction.



Step 2: There are multiple data exchange channels in the ModbusTCP master station, which are independent of each other. The channel parameters can be configured separately in the software configuration interface.



Step 3: There are multiple data exchange channels in the ModbusTCP master station, which are independent of each other. The channels configured by the software can be triggered by default (default running) or be triggered by program (executed after the ModbusTCP_LinkRun instruction is triggered) to open the specified channels. They can be closed by ModbusTCP_Link-Stop instruction, and get the status of the specified data exchange channels by the ModbusTCP_ GetLinkStatus instruction.



Instruction configuration using ModbusTCP functionality:

Step	Item	Instruction	Description
1	Control master	ModbusTCP_MasterRun	Enable ModbusTCP master function
		ModbusTCP_MasterStop	Disable ModbusTCP master function
		ModbusTCP_Masterstatus	Get ModbusTCP master status
2	Configure channels	ModbusTCP_LinkConfig	Configure the specified data exchange channels
3	Control channels	ModbusTCP_LinkRun	Open the specified data exchange channels
		ModbusTCP_LinkStop	Close the specified data exchange channels
		ModbusTCP_GetLinkStatus	Get the status of the specified data exchange channels

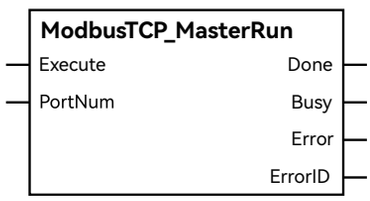
Step 1: After enabling the operation of the ModbusTCP master function master switch, ModbusTCP-related functions can be used. The software configuration can be enabled by the ModbusTCP_MasterRun instruction or by the ModbusTCP_Master-Stop instruction and get the ModbusTCP master status by the ModbusTCP_Masterstatus instruction.

Step 2: There are multiple data exchange channels in the ModbusTCP master station, which are independent of each other. The channel parameters can be configured by the ModbusTCP_LinkConfig instruction.

Step 3: There are multiple data exchange channels in the ModbusTCP master station, which are independent of each other and can be opened by the ModbusTCP_LinkRun instruction. It can be disabled by the ModbusTCP_LinkStop instruction, and get the status of the specified data exchange channels by the ModbusTCP_GetLinkStatus instruction.

1.2 ModbusTCP_MasterRun

Enable ModbusTCP master function. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ModbusTCP_ MasterRun	ModbusTCP management	FB		ModbusTCP_MasterRun0 (Execute:=parameter , PortNum:= parameter , Done=>parameter , Busy=>parameter , Error=>parameter , ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Reserved	UINT	Reserved	Reserved	Reserved

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Refer to "instruction error code description" for the meaning of the output error code value when an instruction execution error occurs.

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed.	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected.	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Error is TRUE, and Execute changes from TRUE to FALSE. The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE

◆ Function description

This instruction is used to enable the master function of ModbusTCP.

1.3 ModbusTCP_MasterStop

Disable the ModbusTCP master function. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ModbusTCP_ MasterStop	ModbusTCP management	FB	ModbusTCP_MasterStop_Instance **ModbusTCP_MasterStop** Execute — Done PortNum — Busy Error ErrorID	ModbusTCP_MasterStop_Instance (Execute:=parameter , PortNum:= parameter, Done=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Reserved	UINT	Reserved	Reserved	Reserved

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Refer to "instruction error code description" for the meaning of the output error code value when an instruction execution error occurs

◆ Output variable refreshing timing

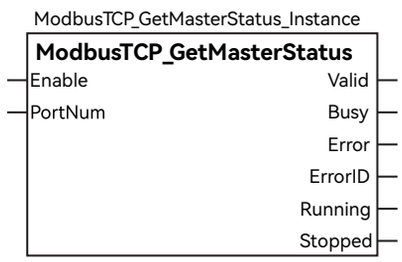
Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process	When Error is TRUE, Execute changes from TRUE to FALSE The Instruction is executed and the Execute become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE

◆ Function description

The instruction is used to disable the ModbusTCP master function.

1.4 ModbusTCP_GetMasterstatus

Get master status. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ModbusTCP_ Masterstatus	Get master status	FB		ModbusTCP_GetMasterStatus(Enable:=parameter, PortNum:=parameter, Valid=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter, Running=>parameter, Stopped=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	The instruction will be executed if it is set to TRUE, or not executed if it is set to FALSE.
PortNum	Reserved	UINT	Reserved	Reserved	Reserved

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the instruction is executed normally
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Refer to "instruction error code description" for the meaning of the output error code value when an instruction execution error occurs
Running	The master function is enabled	BOOL	TRUE / FALSE	TRUE when the master is enabled
Stopped	The master function is disabled	BOOL	TRUE / FALSE	TRUE when the master is disabled

◆ Output variable refreshing timing

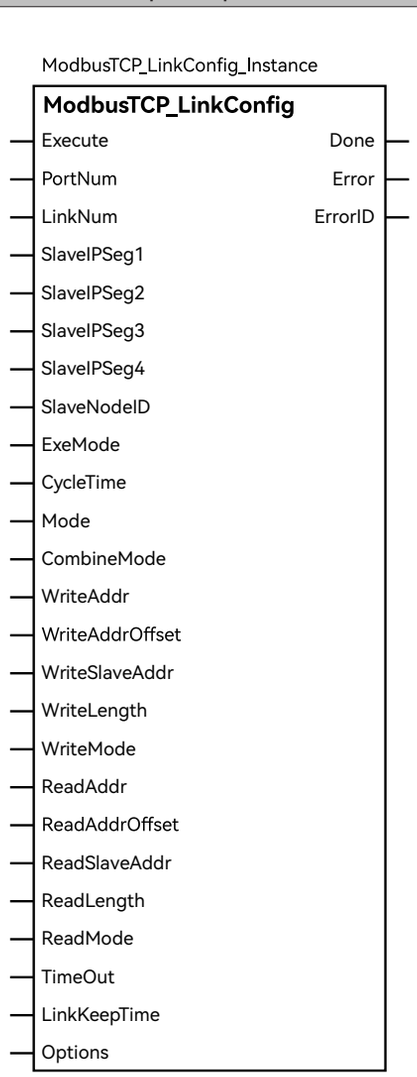
Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Busy	When Execute changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Execute changes from TRUE to FALSE
Running	When ModbusTCP_MasterRun is enabled	When ModbusTCP_MasterStop is enabled When Enable changes from TRUE to FALSE
Stopped	When ModbusTCP_MasterStop is enabled	When ModbusTCP_MasterRun is enabled When Enable changes from TRUE to FALSE

◆ Function description

Get ModbusTCP master status. This instruction can be used to check whether the ModbusTCP master function is enabled or disabled.

1.5 ModbusTCP_LinkConfig

Parameter configuration instruction. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ModbusTCP_LinkConfig	ModbusTCP parameter configuration	FB		ModbusTCP_LinkConfig_Instance(Execute:= parameter, PortNum:=parameter , LinkNum:= parameter, SlaveIPSeg1:= parameter, SlaveIPSeg2:= parameter, SlaveIPSeg3:= parameter, SlaveIPSeg4:= parameter, SlaveNodeID:= parameter, ExeMode:= parameter, CycleTime:= parameter, Mode:= parameter, CombineMode:= parameter, WriteAddr:= parameter, WriteAddrOffset:= parameter, WriteSlaveAddr:= parameter, WriteLength:= parameter, WriteMode:= parameter, ReadAddr:= parameter, ReadAddrOffset:= parameter, ReadSlaveAddr:= parameter, ReadLength:= parameter, ReadMode:= parameter, TimeOut:= parameter, LinkKeepTime:= parameter, Options:= parameter, Done=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Reserved	UINT	Reserved	Reserved	Reserved
LinkNum	Data exchange channel number	UINT	Refer to communication instruction specifications	Required field	Set the number of Modbus TCP periodic data exchange
SlaveIPseg1	IP address of the target device (1st segment)	USINT	0~255	0	Set the 1st segment of the IP address of the target device If the IP address is 192.168.1.2, the 1st segment is 192
SlaveIPseg2	IP address of the target device (2nd segment)	USINT	0~255	0	Set the 2nd segment of the IP address of the target device If the IP address is 192.168.1.2, the 2nd segment is 168

SlaveIPseg3	IP address of the target device (3rd segment)	USINT	0~255	0	Set the 3rd segment of the IP address of the target device If the IP address is 192.168.1.2, the 3rdsegment is 1
SlaveIPseg4	IP address of the target device (4th segment)	USINT	0~255	0	Set the 4th segment of the IP address of the target device If the IP address is 192.168.1.2, the 4th segment is 2
SlaveNodeID	Modbus station number of the target device	USINT	0~255	0	Set the Modbus station number of the target device
ExeMode	Cyclic mode	USINT	0~1	0	Set the mode of sending 0: Cyclic sending 1: Single sending
CycleTime	Cycle time	USINT	0~65535	10	Set the interval between the last data sending and the next data sending in the cyclic sending mode: Unit: ms
Mode	Type of read/write address	USINT	0~1	0	Set the read/write type of the slave address, and select read/write Word-type address or Bit-type address by this parameter 0 : Word 1 : Bit
Comebine-Mode	Enable read/write integration mode	BOOL	TRUE / FALSE	FALSE	This parameter is used to set whether to enable the read/write integration function 0: Disable (read and write data separately) 1: Enable (read and write data are integrated into one single message and use the 0x17 function code) Note: Valid only when Mode is 0
WriteAddr	Starting address of the write operation data cache	UINT	%MW0~%MW32767 %QW0~%QW63	0	Set the storage address in the controller for the data to be written to the target. For write operation, the controller writes the contents of this address (which is determined by both WriteAddr and WriteAddrOffset) to the target device. If the input corresponding to this parameter is a variable, the correct address must be specified when the variable is declared.
WriteAddrOff-set	Offset value of the write operation data cache	USINT	0~255	0	Set the offset value of the write operation data cache The actual content to be written is obtained from the address based on WriteAdd offset by WriteAddrOffset. If it is a Word read/write, the offset unit is 1Word. For example, if WriteAddr specifies the address as %MW0 and WriteAddrOffset is 1, it means the start address is %MW1. If it is a Bit read/write, the offset unit is 1Bit. For example, if WriteAddr specifies the address as %MW1 and WriteAddrOffset is 1, it means the start address is %MX2.1. 1, it means the starting address is %MX2.1.
WriteSlaveAd-dr	Destination address for write operation	UINT	16#0~16#FFFF	0	Set the starting address of the write operation, which is the Modbus address in the target device.
Writelength	Write operation length	UINT	Word device: 0~100 Bit device: 0~256	0	Set the length of the written data. If it is a Word read/write, the unit is Word; if it is a Bit read/write, the unit is Bit.
WriteMode	Write operation mode (function code)	USINT	16#0、16#06、16#10、16#05、16#0F	0	Set the function code to be used for write operation. If set to 0, the controller will select it automatically.

ReadAddr	Starting address of the read operation data cache	UINT	%MW0~%MW32767 %QW0~%QW63	0	Set the storage starting address of the data read from the target device in the controller, which is determined by both ReadAddr and ReadAddrOffset. If the input corresponding to this parameter is a variable, the correct address must be specified when the variable is declared.
ReadAddrOffset	Offset value of the read operation data cache	USINT	0~255	0	Set the offset value of the read operation data cache. The read content will be stored in the address based on ReadAddr and offset by ReadAddrOffset. If it is a Word read/write, the offset unit is 1Word. For example, ReadAddr specifies the address as %MW100, and ReadAddrOffset is 1, which means the starting address is %MW101. If it is a Bit read/write, the offset unit is 1Bit. For example, ReadAddr specifies the address as %MW100, and ReadAddrOffset is 1, which means the starting address is %MX200.1.
ReadSlaveAddr	Destination address for read operation	UINT	16#0~16#FFFF	0	Set the starting address of the read operation, which is the Modbus address in the target device.
Readlength	Read operation length	UINT	Word device: 0~100 Bit device: 0~256 （0）	0	Set the length of the read data. If it is a Word read/write, the unit is Word; if it is a Bit read/write, the unit is Bit.
ReadMode	Read operation mode (function code)	USINT	16#0、16#03、16#04、16#01、16#02	0	Set the function code to be used for read operation. If set to 0, the controller will select it automatically.
TimeOut	Communication timeout	UINT	0~65535	1000	Set the time to wait for a response from the target device. Unit: ms
LinkKeepTime	Time of link keeping	UINT	0~65535	30	Set the time for which the communication connection between this device and the target device can be kept.
Options	Reserved	Reserved	Reserved	Reserved	Reserved

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Refer to "instruction error code description" for the meaning of the output error code value when an instruction execution error occurs

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Execute changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Execute changes from TRUE to FALSE

◆ Function description

• Basic function description

This instruction is used to configure the parameters related to Ethernet ModbusTCP periodic data exchange (write and read), such as the IP address of the target device, the Modbus station number, the destination address for data read and write, the cache address, the data length, the function code, and so on.

Note: This instruction is for parameter configuration only. If the configuration is changed while the ModbusTCP master is enabled and the channel data is exchanged, and this instruction is re-triggered, it will take effect immediately.

- **LinkNum**

There are multiple ModbusTCP periodic data exchange channels in the controller, and these data exchange channels are independent of each other and can be configured with their parameters separately. The LinkNum parameter of this instruction can be used to specify the channel that needs to be configured with parameters.

- **EnableLink**

The parameter EnableLink enables or disables the Modbus TCP periodic data exchange channel specified by LinkNum. The channel is enabled when EnableLink is TRUE and disabled when FALSE.

Note: Like other parameters, EnableLink is latched when the instruction is triggered for execution (Execute rising edge moment), and any change in the value of EnableLink before or after this moment is invalid.

- **WriteAddr、WriteAddrOffset、WriteLength**

The periodic data exchange consists of write operations and read operations. Write operation is to write the data of the specified length from the write operation cache address to the target device. The write operation cache address is specified by the parameters WriteAddr, WriteAddrOffset. The data length of the write operation is specified by WriteLength. The parameters WriteAddr, WriteAddrOffset specify the starting address of the cache area, which is in the form of “base address+ offset”. The base address is WriteAddr and the offset is WriteAddrOffset.

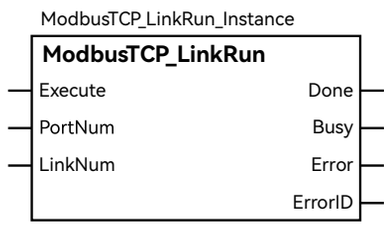
For Word read/write (Mode=0), the unit of WriteAddrOffset is Word, e.g., the base address specified by WriteAddr is %MW1000, and the length specified by WriteLength is 5. If WriteAddrOffset is 0, the cache start address is %MW1000.

- **Priority description of software configuration and instruction configuration**

The instruction-configured channels have a higher priority than the software-configured channels when they take effect. Software-configured channels will take effect only once when the program is downloaded and powered up, and instruction-configured channels do not have power-down persistence and require the instruction to be re-executed.

1.6 ModbusTCP_LinkRun

Enable the data exchange channels specified by ModbusTCP. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ModbusTCP_LinkRun	Slave link management	FB		ModbusTCP_LinkRun1(Execute:= parameter, PortNum:= parameter, LinkNum:= parameter, Done=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Reserved	UINT	Reserved	Reserved	Reserved
LinkNum	Data exchange channel number	UINT	Refer to communication instruction specifications	Required field	Set the number of Modbus TCP periodic data exchange.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Refer to "instruction error code description" for the meaning of the output error code value when an instruction execution error occurs

◆ Output variable refreshing timing

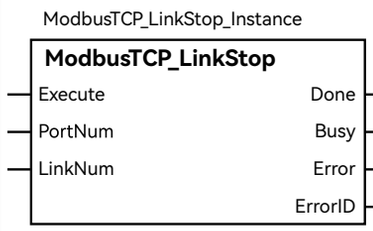
Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process	When Error is TRUE, and Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE

◆ Function description

Enable the specified data exchange channels.

1.7 ModbusTCP_LinkStop

Disable the data exchange channels specified by ModbusTCP. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ModbusTCP_LinkStop	Slave link management	FB		ModbusTCP_LinkStop(Execute:= parameter, PortNum:= parameter, LinkNum:= parameter, Done=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Reserved	UINT	Reserved	Reserved	Reserved
LinkNum	Data exchange channel number	UINT	Refer to communication instruction specifications	Required field	Set the number for Modbus TCP periodic data exchange.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Refer to "instruction error code description" for the meaning of the output error code value when an instruction execution error occurs

◆ Output variable refreshing timing

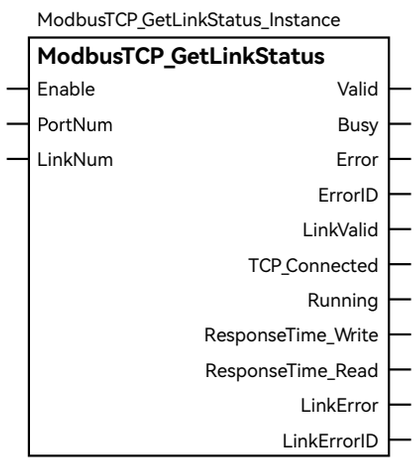
Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process	When Error is TRUE and Execute changes from TRUE to FALSE The Instruction is executed and the Execute becomes FALSE. When an exception is encountered during the execution, Error becomes TRUE. After one period, it becomes FALSE

◆ Function description

Disable the specified data exchange channels.

1.8 ModbusTCP_GetLinkStatus

Get the status of the data exchange channels specified by ModbusTCP. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_Manage	Get the status of the data exchange channels specified by ModbusTCP	FB		ModbusTCP_GetLinkStatus1(Enable:=parameter , PortNum:= parameter, LinkNum:= parameter, Valid=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter, LinkValid=>parameter, TCP_Connected=>parameter, Running=>parameter, ResponseTime_Write=>parameter, ResponseTime_Read=>parameter, LinkError=>parameter, LinkErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	The instruction will be executed if it is set to TRUE, or not executed if it is set to FALSE.
PortNum	Reserved	UINT	Reserved	Reserved	Reserved
LinkNum	Data exchange channel number	UINT	Refer to communication instruction specifications	Required field	Set the number of Modbus TCP periodic data exchange.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD		Output an error code when an instruction execution exception occurs. *1
LinkValid	Configuration completed	BOOL	TRUE / FALSE	Configuration of the channels specified by LinkNum is completed
TCP_Connected	TCP connection	BOOL	TRUE / FALSE	TRUE when a connection is established with the slave of the specified channels
Running	Normal data exchange	BOOL	TRUE / FALSE	TRUE when the specified channel data exchange is normal and FALSE when it is abnormal
ResponseTime_Write	Response time of write operation	UINT	0~65535	The time interval from when the master sends a write instruction to when the master receives the slave's response. Unit: ms
ResponseTime_Read	Response time of read operation	UINT	0~65535	The time interval from when the master sends a read instruction to when the master receives the slave's response. Unit: ms
LinkError	Abnormal data exchange	BOOL	BOOL/FALSE	TRUE when the specified channel data exchange is abnormal and FALSE when it is normal; if the slave responds abnormally to the data or has a timeout, this bit changes to TRUE, and when the slave responds normally, it automatically changes to FALSE.

LinkErrorID	Abnormal data exchange error code	WORD	0~65535	The corresponding error code is output when LinkError is TRUE for a specified channel data exchange exception. For the meaning of the value, refer to "Instruction error code description".

◆ Output variable refreshing timing

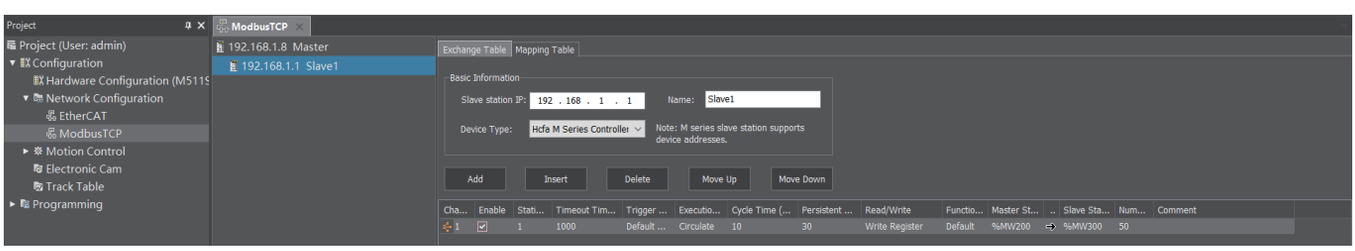
Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	TRUE when the instruction is executed normally
Busy	When Enable changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The value of the instruction input variable is out of the allowed range, the execution conditions of the instruction are not satisfied, or an exception is encountered during the execution of the instruction	When Execute changes from TRUE to FALSE
LinkValid	TRUE when the configuration of the LinkNum specified channels is valid.	FALSE when the configuration of the LinkNum specified channels is invalid.
TCP_Connected	TRUE when a connection is established with the slave of the specified channels	FALSE when disconnected with the slave of the specified channels
Running	TRUE when the specified channel data exchange is normal	FALSE when the specified channel data exchange is abnormal
LinkError	TRUE when the specified channel data exchange is abnormal	FALSE when the specified channel data exchange is normal

◆ Function description

This instruction is used to get the status of the specified ModbusTCP data exchange channels, including checking whether the current data exchange of the specified channels are normal or not, or whether the connection between master and slave is established or not. If the output bit of Running is TRUE, it means the data exchange is normal; if the output bit of LinkError is TRUE, it means the data exchange is abnormal. LinkErrorID outputs the corresponding error code. If the slave responds with abnormal data or timeout, the LinkError bit of the instruction changes to TRUE, and if the slave responds normally, it automatically changes to FALSE.

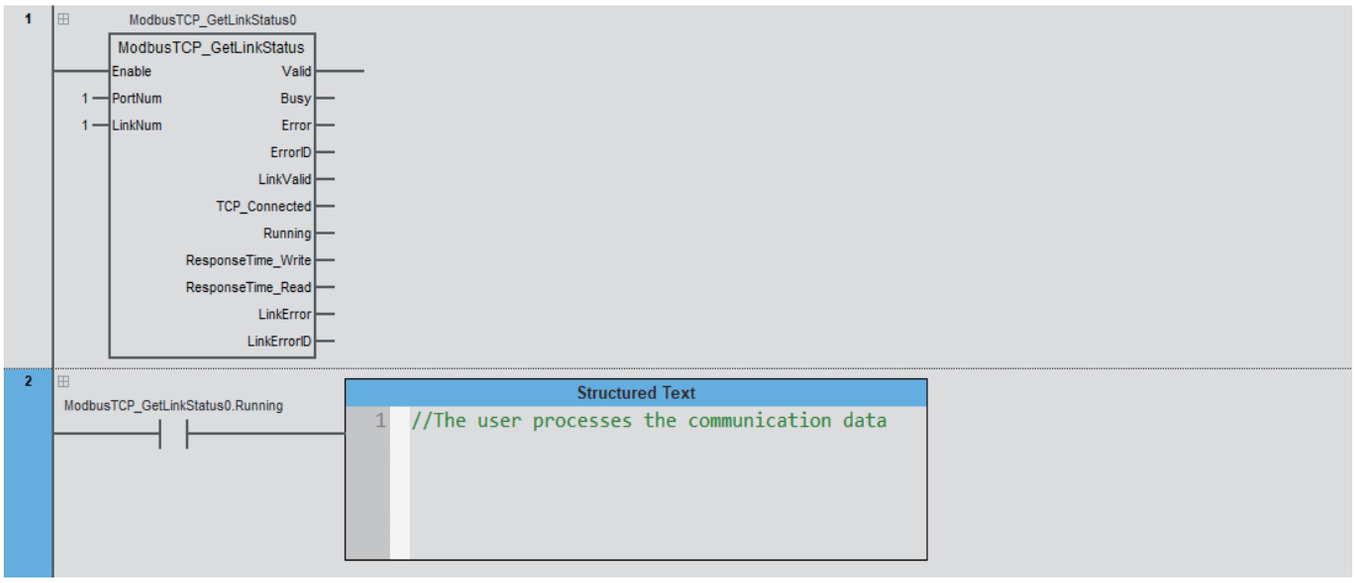
◆ Example program

As shown in the figure below, a sum of data is configured for channel 1 in the ModbusTCP configuration interface: the station number is 1, trigger by default, and the function code is 0x10. With this instruction, the status of channel 1 can be obtained.



Category	Name	Assigned to	Data type	Initial value	Comment
VAR	ModbusTCP_GetLinkStatus0		BOOL		

LD:



ST:

```
ModbusTCP_GetLinkStatus1(Enable:= 1,
    PortNum:= 1,
    LinkNum:= 1,
    Valid=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    LinkValid=> ,
    TCP_Connected=> ,
    Running=> ,
    ResponseTime_Write=> ,
    ResponseTime_Read=> ,
    LinkError=> ,
    LinkErrorID=>
    );
    IF ModbusTCP_GetLinkStatus1.Running THEN
        //The user processes the communication data
        ;
    END_IF;
```

- **Program description:**

When Running is TRUE, it means that the specified channel data is running for exchange, and users can process the communication data according to their actual demands.

1.9 ModbusTCP communication usage example

1.9.1 TCP data exchange example 1: Software configuration ModbusTCP

- **Target demand**

The two M-Series PLCs exchange data via the 0x10 function code and 0x03 function code of the TCP protocol.

The IP address and port of the controller are as follows:

Item	Master controller	Item	Slave controller
IP address	192.168.1.1	IP address	192.168.1.2
Port number	502	Port number	502
Address	MW200	Address	MW300
Address	MW500	Address	MW400

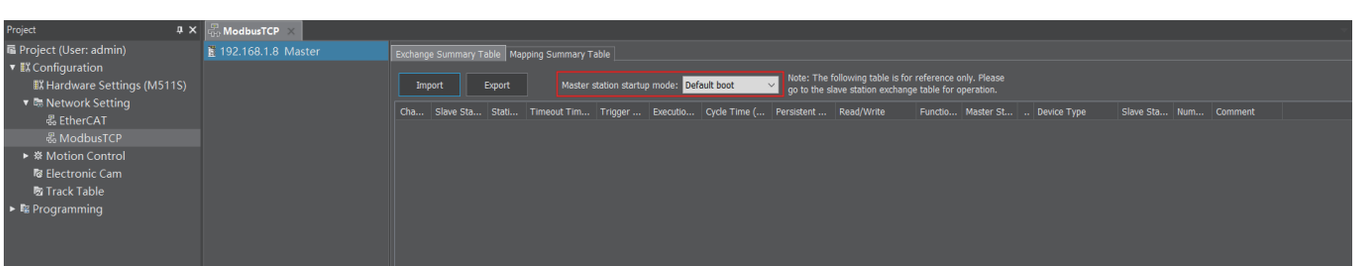
- **Demand analysis**

According to the demand, it is necessary for the master controller to use the 0x10 function code of channel 1 to send data to the slave controller and use the 0x03 function code to read data from the slave controller.

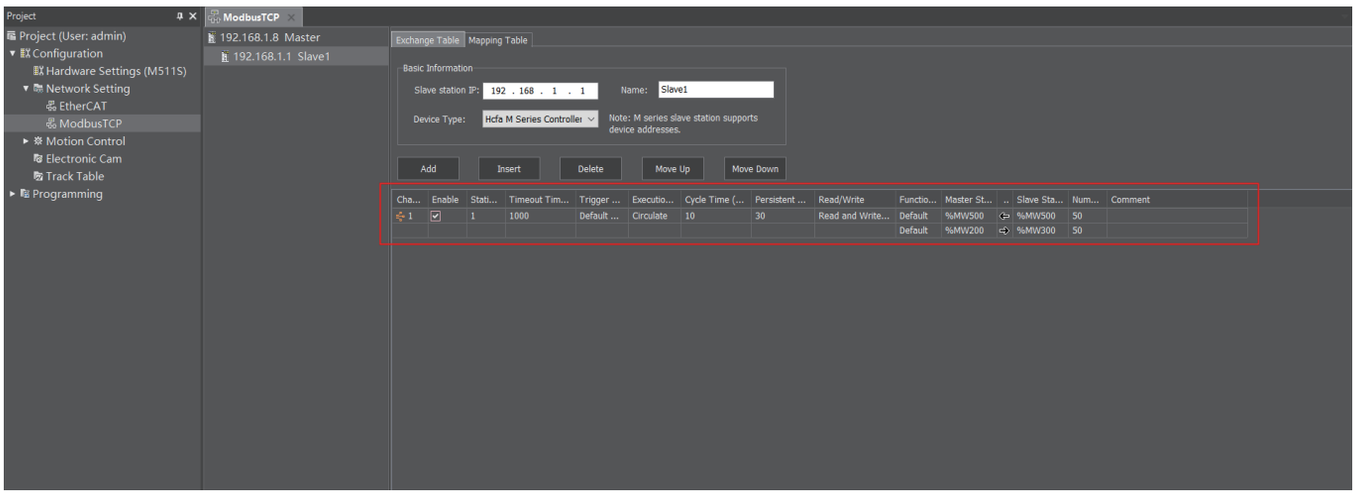
Step	Item	Usage		Description
1	Control master	Select “enable by instructions” in the software configuration interface	ModbusTCP_MasterRun	Enable ModbusTCP master function
		Select “enable by default” in the software configuration interface	Enable by default	Enable ModbusTCP master function
		ModbusTCP_MasterStop		Disable ModbusTCP master function
		ModbusTCP_Masterstatus		Get ModbusTCP master status
2	Configure channels	Software configuration interface		Configure the specified data exchange channels
3	Control channels	Select “trigger by program” in the software configuration interface	ModbusTCP_LinkRun	Open the specified data exchange channels
		Select “trigger by default” in the software configuration interface	Enable by default	Open the specified data exchange channels
		ModbusTCP_LinkStop		Close the specified data exchange channels
		ModbusTCP_GetLinkStatus		Get the status of the specified data exchange channels

- **Software configuration**

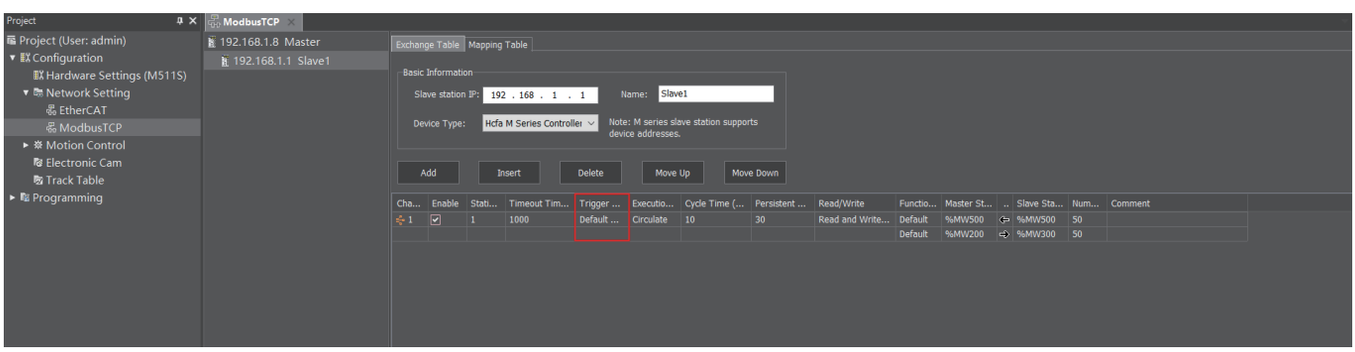
Step 1: Set the ModbusTCP master startup method to enable by default, and the ModbusTCP master function will be enabled automatically after the download.



Step 2: Add a slave and set the data parameters for channel 1.



Step 3: Set the trigger mode to trigger by default, and the data exchange of channel 1 will be automatically enabled after downloading or after powering up the controller.



1.9.2 TCP data exchange example 2: Software and instruction configuration ModbusTCP

- **Target demand**

Two M-Series PLCs exchange data via TCP protocol with 0x10 function code and 0x06 function code.

The IP address and port of the controller are as follows:

Item	Master controller	Item	Slave controller
IP address	192.168.1.1	IP address	192.168.1.2
Port number	502	Port number	502
Address	MW200	Address	MW300
Address	MW500	Address	MW400

- **Demand analysis**

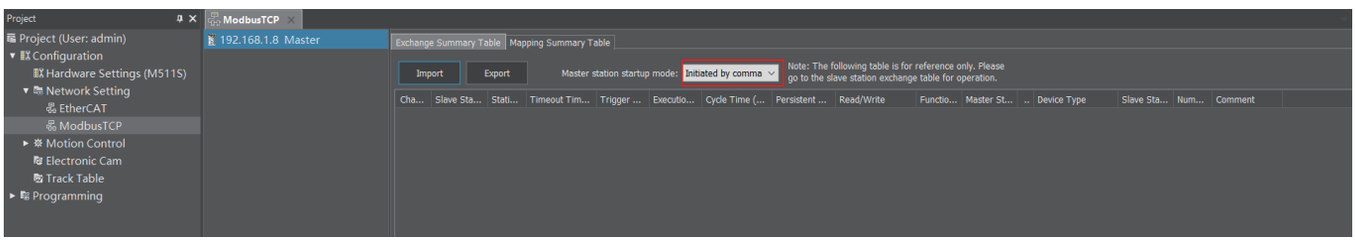
According to the demand, it is necessary for the master controller to use the 0x10 function code of channel 1 to send data to the slave controller and use the 0x03 function code to read data from the slave controller.

Step	Item	Usage		Description
1	Control master	Select “enable by instructions” in the software configuration interface	ModbusTCP_Master-Run	Enable ModbusTCP master function
		Select “enable by default” in the software configuration interface	Enable by default	Enable ModbusTCP master function
		ModbusTCP_MasterStop		Disable ModbusTCP master function
		ModbusTCP_Masterstatus		Get ModbusTCP master status
2	Configure channels	Software configuration interface		Configure the specified data exchange channels

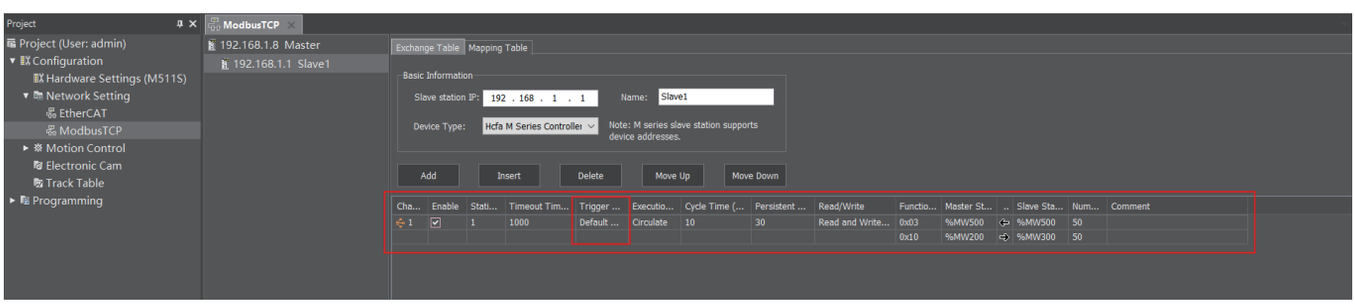
3	Control channels	Select “trigger by program” in the software configuration interface	ModbusTCP_LinkRun	Open the specified data exchange channels
		Select “trigger by default” in the software configuration interface	Enable by default	Open the specified data exchange channels
		ModbusTCP_LinkStop		Close the specified data exchange channels
		ModbusTCP_GetLinkStatus		Get the status of the specified data exchange channels

- **Software configuration**

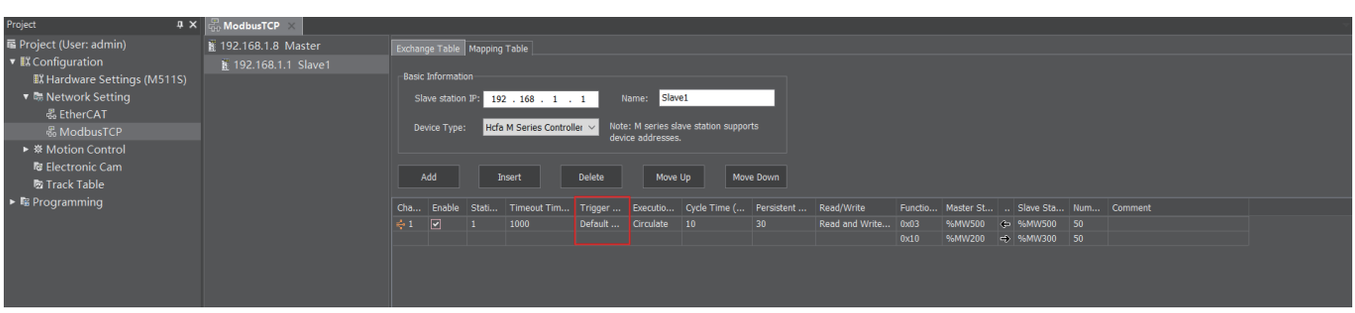
Step 1: Set the ModbusTCP master startup method to enable by instructions, which can enable the master functions.



Step 2: Add a slave and set the data parameters for channel 1.



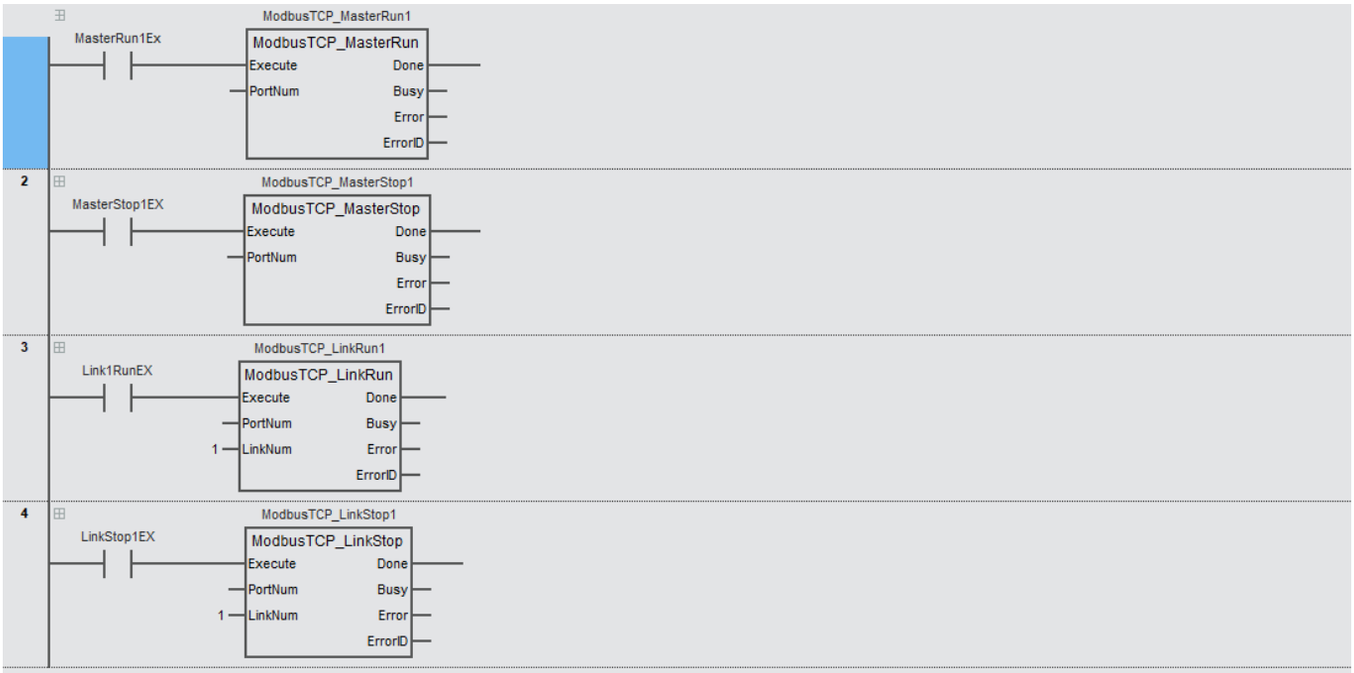
Step 3: Set the trigger method to trigger by program, and the data exchange of channel 1 is enabled by instructions.



- **Variable table**

Category	Name	Assigned to	Data type	Initial value	Comment
VAR	MasterRun1Ex		BOOL		
VAR	MasterStop1EX		BOOL		
VAR	ModbusTCP_MasterRun1		ModbusTCP_MasterRun		
VAR	ModbusTCP_MasterStop1		ModbusTCP_MasterStop		
VAR	Link1RunEX		BOOL		
VAR	LinkStop1EX		BOOL		
VAR	ModbusTCP_LinkRun1		ModbusTCP_LinkRun		
VAR	ModbusTCP_LinkStop1		ModbusTCP_LinkStop		

LD



ST

```
ModbusTCP_MasterRun1(Execute:= MasterRun1Ex,
    PortNum:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
ModbusTCP_MasterStop(Execute:=MasterStop1EX ,
    PortNum:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
ModbusTCP_LinkRun1(Execute:= Link1RunEX,
    PortNum:= ,
    LinkNum:= 1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
ModbusTCP_LinkStop1(Execute:=LinkStop1EX ,
    PortNum:= ,
    LinkNum:=1 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
```

- **Program description**

Step 1: The ModbusTCP_MasterRun1 instruction is triggered to enable the ModbusTCP master function when the variable MasterRun1Ex is set to TRUE.

Step 2: The ModbusTCP_LinkConfig1 instruction is triggered to write the configured data parameters to channel 1 when the variable LinkConfigEX is set to TRUE.

Step 3: To stop the data exchange of channel 1, set the variable LinkStop1EX to TRUE to trigger the ModbusTCP_LinkStop1 instruction, or set the variable MasterStop1EX to TRUE to trigger the ModbusTCP_MasterStop1 instruction to disable the ModbusTCP Master function. Use the ModbusTCP_LinkStop instruction to disable the specified channel data exchange. If users configure other channels, other channels can still exchange data. Use ModbusTCP_MasterStop to disable the ModbusTCP master function, and all data channel exchange will be stopped.

1.9.3 TCP data exchange example 3: Instruction configuration ModbusTCP

• Target demand

Two M-Series PLCs exchange data via TCP protocol with 0x10 function code and 0x06 function code.

The IP address and port of the controller are as follows:

Item	Master controller	Item	Slave controller
IP address	192.168.1.1	IP address	192.168.1.2
Port number	502	Port number	502
Address	MW200	Address	MW300
Address	MW500	Address	MW400

• Demand analysis

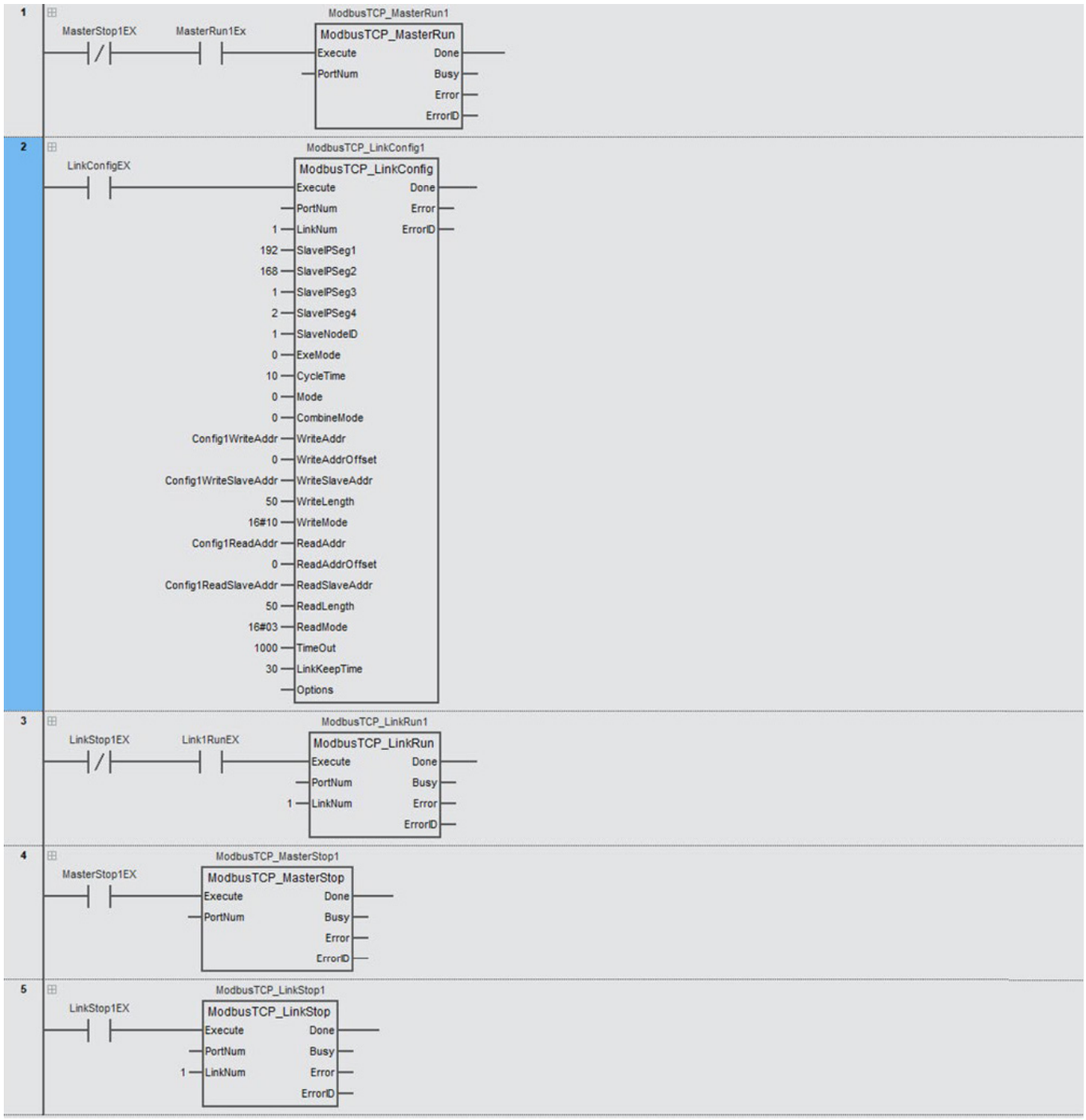
According to the demand, it is necessary for the master controller to use the 0x10 function code of channel 1 to send data to the slave controller and use the 0x03 function code to read data from the slave controller.

Step	Item	Instruction	Description
1	Control master	ModbusTCP_MasterRun	Enable ModbusTCP master function
		ModbusTCP_MasterStop	Disable ModbusTCP master function
		ModbusTCP_Masterstatus	Get ModbusTCP master status
2	Configure channels	ModbusTCP_LinkConfig	Configure data exchange channel 1 (see instructions for specific configuration)
3	Control channels	ModbusTCP_LinkRun	Open data exchange channel 1
		ModbusTCP_LinkStop	Close data exchange channel 1
		ModbusTCP_GetLinkStatus	Get the status of data exchange channel 1

• Variable table

Category	Name	Assigned to	Data type	Initial value	Comment
VAR	ModbusTCP_LinkConfig1		ModbusTCP_LinkConfig		
VAR	LinkConfigEX		BOOL		
VAR	Config1WriteAddr	%MW200	UINT		
VAR	Config1WriteSlaveAddr	%MW300	UINT		
VAR	Config1ReadAddr	%MW500	UINT		
VAR	Config1ReadSlaveAddr	%MW400	UINT		
VAR	ModbusTCP_MasterRun1		ModbusTCP_MasterRun		
VAR	MasterRun1Ex		BOOL		
VAR	ModbusTCP_MasterStop1		ModbusTCP_MasterStop		
VAR	MasterStop1EX		BOOL		
VAR	ModbusTCP_LinkRun1		ModbusTCP_LinkRun		
VAR	Link1RunEX		BOOL		
VAR	ModbusTCP_LinkStop1		ModbusTCP_LinkStop		
VAR	LinkStop1EX		BOOL		

LD


1
ModbusTCP_MasterRun1
MasterStop1EX
MasterRun1Ex
ModbusTCP_MasterRun
Execute
PortNum
Done
Busy
Error
ErrorID
2
ModbusTCP_LinkConfig1
LinkConfigEX
ModbusTCP_LinkConfig
Execute
PortNum
1 LinkNum
192 SlaveIPSeg1
168 SlaveIPSeg2
1 SlaveIPSeg3
2 SlaveIPSeg4
1 SlaveNodeID
0 ExeMode
10 CycleTime
0 Mode
0 CombineMode
Config1WriteAddr WriteAddr
0 WriteAddrOffset
Config1WriteSlaveAddr WriteSlaveAddr
50 WriteLength
16#10 WriteMode
Config1ReadAddr ReadAddr
0 ReadAddrOffset
Config1ReadSlaveAddr ReadSlaveAddr
50 ReadLength
16#03 ReadMode
1000 TimeOut
30 LinkKeepTime
Options
Done
Error
ErrorID
3
ModbusTCP_LinkRun1
LinkStop1EX
Link1RunEX
ModbusTCP_LinkRun
Execute
PortNum
1 LinkNum
Done
Busy
Error
ErrorID
4
ModbusTCP_MasterStop1
MasterStop1EX
ModbusTCP_MasterStop
Execute
PortNum
Done
Busy
Error
ErrorID
5
ModbusTCP_LinkStop1
LinkStop1EX
ModbusTCP_LinkStop
Execute
PortNum
1 LinkNum
Done
Busy
Error
ErrorID

ST

```
ModbusTCP_LinkConfig1(Execute:=LinkConfigEX ,
    PortNum:=  ,
    LinkNum:=1 ,
    SlaveIPSeg1:=192 ,
    SlaveIPSeg2:=168 ,
    SlaveIPSeg3:= 1,
    SlaveIPSeg4:= 1,
    SlaveNodeID:= 1,
    ExeMode:= 0,
    CycleTime:=10 ,
    Mode:= 0,
    CombineMode:= 0,
    WriteAddr:=Config1WriteAddr ,
    WriteAddrOffset:= 0,
    WriteSlaveAddr:= 0,
    WriteLength:=50 ,
    WriteMode:= 16#10,
    ReadAddr:= Config1ReadAddr,
    ReadAddrOffset:=0 ,
    ReadSlaveAddr:= Config1ReadSlaveAddr,
    ReadLength:=50 ,
    ReadMode:=16#05 ,
    TimeOut:= 1000,
    LinkKeepTime:=30 ,
    Options:= ,
    Done=> ,
    Error=> ,
    ErrorID=>
    );

    ModbusTCP MasterRun1(Execute:=  MasterRun1Ex AND (NOT MasterStop1EX) ,
    PortNum:=  ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );

    ModbusTCP_LinkRun1(Execute:=  Link1RunEX AND (NOT LinkStop1EX) ,
    PortNum:=  ,
    LinkNum:=1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );

    ModbusTCP_MasterStop1(Execute:= MasterStop1EX,
    PortNum:=  ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );

    ModbusTCP_LinkStop1(Execute:= LinkStop1EX,
    PortNum:=  ,
    LinkNum:=1 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );

```

- **Program description**

Step 1: Set variable MasterRun1Ex to TRUE to trigger ModbusTCP_MasterRun1 instruction to enable ModbusTCP master function.

Step 2: Set variable LinkConfigEX to TRUE to trigger ModbusTCP_LinkConfig1 instruction and write the configured data parameters to channel 1.

Step 3: Set variable Link1RunEX to TRUE to trigger ModbusTCP_LinkRun1 instruction to open channel 1 to start configured

data exchange.

Step 4: To stop the data exchange on channel 1, set the variable LinkStop1EX to TRUE to trigger the ModbusTCP_LinkStop1 instruction, thus disabling the data exchange on channel 1. Or set the variable MasterStop1EX to TRUE to trigger the ModbusTCP_MasterStop1 instruction to disable the ModbusTCP Master function. Use the ModbusTCP_LinkStop instruction to disable the specified channel data exchange, if users configure other channels, the other channels can still exchange data. Use ModbusTCP_MasterStop to disable ModbusTCP master function, and all data channel exchange will be stopped.

1.10 Socket communication usage overview

The instruction is configured to use the Socket function:

Step	Item	Instruction	Description
1	Configure the Socket parameter	Socket_Config	Configure communication parameters, including IP address, port number, and communication method. Ensure that the communication parameters of the PLC and the external device are matched
2	Connection establishment	Socket_Open	Configure the working mode of the Socket, and enable the Socket data exchange startup function
		Socket_GetStatus	Monitor Socket status
3	Data sending and receiving	Socket_Send	Sending data
		Socket_Receive	Receiving data
4	Close connection	Close Socket data exchange	Close Socket connection

Step 1: Configure Socket parameters. Configure communication parameters by executing the Socket_Config instruction to set the IP address, port number, connection mode (TCP or UDP), link keeping time and other parameters of the target device.

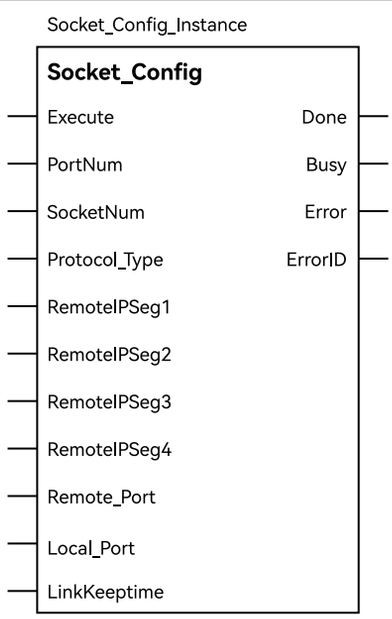
Step 2: To establish a connection, set the working mode and establish a connection by executing Socket_Open. After the Socket_Open instruction is executed, the Socket connection status can be checked and obtained by the Socket_GetStatus instruction.

Step 3: Send data by executing the Socket_Send instruction. Receive data by executing the Socket_Receive instruction.

Step 4: Close the connection by executing the Socket_Closed instruction.

1.11 Socket_Config

Configure the parameters related to the Socket function. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_Config	Socket parameter configuration	FB		Socket_Config_Instance(Execute:= parameter, PortNum:=parameter, SocketNum:= parameter, Protocol_Type:= parameter, RemoteIPSeg1:= parameter, RemoteIPSeg2:= parameter, RemoteIPSeg3:= parameter, RemoteIPSeg4:= parameter, Remote_Port:= parameter, Local_Port:= parameter, LinkKeeptime:= parameter, Done=> parameter, Busy=> parameter, Error=> parameter, ErrorID=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Ethernet hardware interface number	UINT	Reserved	Reserved	Reserved
SocketNum	Socket number	UINT	Refer to communication instruction specifications	Required field	Specify the number of the Socket. Different Sockets can be identified by SocketNum
Protocol_Type	Socket connection mode	USINT	0 or 1	0	Set the Socket connection mode 0: UDP 1: TCP
RemoteIPSeg1	IP address of the target device (1st segment)	USINT	0~255	0	Set the 1st segment of the IP address of the target device If the IP address is 192.168.1.2, the 1st segment is 192
RemoteIPSeg2	IP address of the target device (2nd segment)	USINT	0~255	0	Set the 2nd segment of the IP address of the target device If the IP address is 192.168.1.2, the 2nd segment is 168
RemoteIPSeg3	IP address of the target device (3rd segment)	USINT	0~255	0	Set the 3rd segment of the IP address of the target device If the IP address is 192.168.1.2, the 3rd segment is 1
RemoteIPSeg4	IP address of the target device (4th segment)	USINT	0~255	0	Set the 4th segment of the IP address of the target device If the IP address is 192.168.1.2, the 4th segment is 2
Remote_Port	Port number of the target device	UINT	0~65535	0	Set the port number of the target device
Local_Port	Local port number	UINT	0~65535	0	Set local port number of the controller
LinkKeeptime	Time of link keeping	UINT	0~65535	0	Set the Socket link keeping time, if no data is exchanged exceeding this time, the connection will be closed automatically (unit: seconds).

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

*1 Refer to "instruction error code description" for the meaning of the output error code value.

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Execute changes from TRUE to FALSE
Busy	When Execute changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Execute changes from TRUE to FALSE

◆ Function description

• Basic function description

This instruction is used to configure the parameters related to the Socket function, such as the IP address of the target device, the port number, the link keeping time, and so on.

• Local_Port

In UDP mode, Local_Port can be set to 0, in which case the controller will automatically assign the port number.

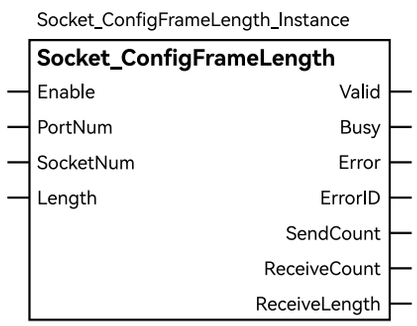
Local_Port must not be set to 0 in TCP mode.

• Precautions

This instruction cannot be executed after the Socket data exchange is enabled, and can be executed after the Socket data exchange is disabled.

1.12 Socket_ConfigFrameLength

Configure the data length related to the Socket function. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_Config-FrameLength	Socket data length configura-tion	FB		Socket_ConfigFrameLength_Instance(Enable:= parameter, PortNum:=parameter, SocketNum:= parameter, Length:= parameter, Valid=> parameter, Busy=> parameter, Error=> parameter, ErrorID=> parameter, SendCount=> parameter, ReceiveCount=> parameter, ReceiveLength=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	Execute the instruction when this parameter is TRUE
PortNum	Ethernet hardware interface number	UINT	Reserved	Reserved	Reserved
SocketNum	Socket number	UINT	Refer to communi-cation instruction specifications	Required field	Specify the number of the Socket. Different Socket can be identified by SocketNum.
Length	Socket data length	ARRAY[1..100] OF UINT	0~1460	0	Set the Socket data length (unit: Byte)

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the instruction output variable is valid
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD		Output an error code when an instruction execution exception occurs. *1
SendCount	Number of Socket sending	UINT		Number of Socket sending
ReceiveCount	Number of Socket receiving	UINT		Number of Socket receiving
ReceiveLength	Data length of Socket sending	ARRAY[1..100] OF UINT		Data length of Socket receiving (unit: Byte)

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Busy	When Execute changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE

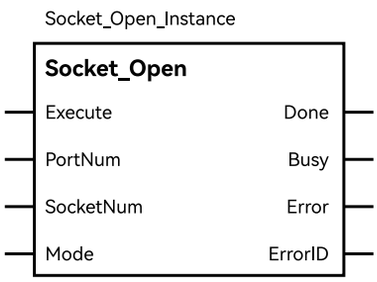
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Execute changes from TRUE to FALSE

◆ Function description

This instruction is an extension of the Socket_Config instruction and is used to configure the length of data sent by the Ethernet port Socket.

1.13 Socket_Open

Enable Socket function. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_Open	Enable Socket function	FB		Socket_Open_Instance(Execute:= parameter, PortNum:=parameter, SocketNum:= parameter, Mode:= parameter, Done=> parameter, Busy=> parameter, Error=> parameter, ErrorID=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Ethernet hardware interface number	UINT	Reserved	Reserved	Reserved
SocketNum	Socket number	UINT	Refer to communication instruction specifications	Required field	Specify the number of the Socket. Different Socket can be identified by SocketNum.
Mode	Socket working mode	BOOL	TRUE / FALSE	FALSE	Set the Socket working mode TRUE: Client FALSE: Server

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Execute changes from TRUE to FALSE
Busy	When Execute changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Execute changes from TRUE to FALSE

◆ Function description

• Basic function description

This instruction is used to enable the Socket function, i.e. to establish a TCP connection or to enable the UDP function. The

status after startup is available with the Socket_GetStatus instruction.

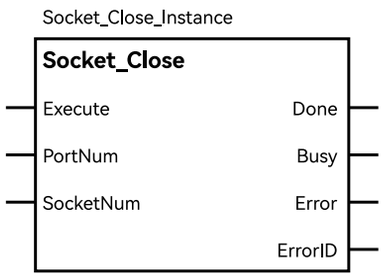
- **Working mode**

  Server Mode (Mode=FALSE): The controller is waiting for the target device to establish a connection with it.

  Client Mode (Mode=TRUE): The controller establishes a connection with the target device.

1.14 Socket_Close

Disable Socket function. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_Close	Disable Socket function	FB		Socket_Close_Instance(Execute:= parameter, PortNum:=parameter, SocketNum:= parameter, Done=> parameter, Busy=> parameter, Error=> parameter, ErrorID=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Ethernet hardware interface number	UINT	Reserved	Reserved	Reserved
SocketNum	Socket number	UINT	Refer to communication instruction specifications	Required field	Specify the number of the Socket. Different Socket can be identified by SocketNum.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

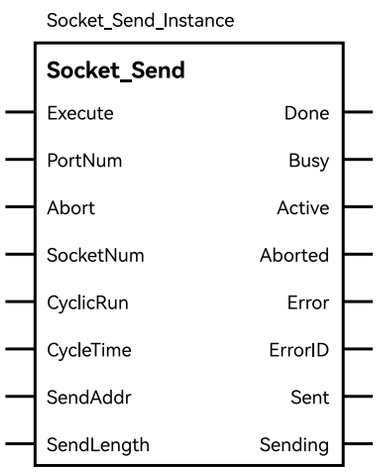
Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Execute changes from TRUE to FALSE
Busy	When Execute changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Execute changes from TRUE to FALSE

◆ Function description

This instruction is used to disable the Socket function, i.e., to disconnect the TCP connection or to disable the UDP function.

1.15 Socket_Send

Send Socket data. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_Send	Send Socket data	FB		Socket_Send_Instance(Execute:= parameter, PortNum:=parameter, Abort:= parameter, SocketNum:= parameter, CyclicRun:= parameter, CycleTime:= parameter, SendAddr:= parameter, SendLength:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Aborted=> parameter, Error=> parameter, ErrorID=> parameter, Sent=> parameter, Sending=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Ethernet hardware interface number	UINT	Reserved	Reserved	Reserved
Abort	Abort data sending	BOOL	TRUE / FALSE	FALSE	Abort when the rising edge of this parameter is detected
SocketNum	Socket number	UINT	Refer to communication instruction specifications	Required field	Specify the number of the Socket. Different Socket can be identified by SocketNum.
CyclicRun	Cyclic sending	BOOL	TRUE / FALSE	FALSE	Set whether to send the data in the buffer cyclically or not. TRUE: Cyclic sending FALSE: Single sending
CycleTime	Cyclic sending interval	UINT	0~65535	0	Set the cyclic sending interval (Unit: ms)
SendAddr	Send data cache address	USINT	%MB0~%MB65535	Required field	Set the starting address for storing sent data.
SendLength	Send data length	UINT	1~1000	0	Set the length of sent data (Unit: Byte)

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed (single sending only)	BOOL	TRUE / FALSE	In single sending mode, it changes to TRUE when data sending is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	The instruction is controlling the Socket.	BOOL	TRUE / FALSE	TRUE when the instruction controls the Socket normally

Aborted	The instruction is aborted	BOOL	TRUE / FALSE	TRUE when the instruction is aborted
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Sent	Sent successfully	BOOL	TRUE / FALSE	TRUE when data sending is completed
Sending	Sending	BOOL	TRUE / FALSE	TRUE in data sending

*1 Refer to "instruction error code description" for the meaning of the output error code value.

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	When sending is completed	When Execute changes from TRUE to FALSE
Busy	When Execute changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Aborted changes from FALSE to TRUE When Error changes from FALSE to TRUE
Active	When the instruction starts controlling the Socket	When Done changes from FALSE to TRUE When Aborted changes from FALSE to TRUE When Error changes from FALSE to TRUE
Aborted	When the instruction is aborted	When Execute changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Execute changes from TRUE to FALSE
Sent	When sending is completed	When Execute changes from TRUE to FALSE When the data sending is not yet completed
Sending	When the data sending is not yet completed	When data sending is completed

◆ Function description

• Basic function description

This instruction is used to send Socket data. When the instruction triggers execution, SendLength length data in the cache area is sent to the target device via Socket, where the starting address of the cache area is specified by the parameter SendAddr.

• Single sending

When the parameter CyclicRun is FALSE, it is a single sending. Each time the instruction is triggered, only one data sending will be executed, and the output variable Done changes to TRUE when the instruction is completed, and the instruction needs to be triggered again if it needs to be sent again.

• Cyclic sending

When the parameter CyclicRun is TRUE, it is cyclic sending. After the instruction triggers execution, data will be sent cyclically at the interval set by CycleTime. If CycleTime is set to 0, the next data will be sent immediately after one data sending. Every time the sending is completed, the output variable Sent becomes TRUE for one period, and then becomes FALSE. When the data is ready to be sent, Sending is TRUE and Sent is FALSE. When the data is completed, Sending is FALSE and Sent is TRUE.

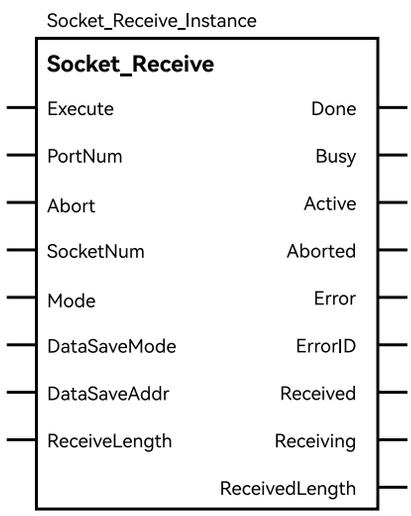
During the cyclic sending process, the cyclic sending can be aborted by setting Aborted from FALSE to TRUE, and the output variable Aborted will be changed to TRUE at the same time.

• Precautions

This instruction needs to be executed after the Socket function is enabled, and the Socket function can be enabled by the Socket_Open instruction.

1.16 Socket_Receive

Receive Socket data. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_Receive	Receive Socket data	FB		Socket_Receive_Instance(Execute:= parameter, PortNum:=parameter, Abort:= parameter, SocketNum:= parameter, Mode:= parameter, DataSaveMode:= parameter, DataSaveAddr:= parameter, ReceiveLength:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Aborted=> parameter, Error=> parameter, ErrorID=> parameter, Received=> parameter, Receiving=> parameter, ReceivedLength=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
PortNum	Ethernet hardware interface number	UINT	Reserved	Reserved	Reserved
Abort	Abort data receiving	BOOL	TRUE / FALSE	FALSE	Abort when the rising edge of this parameter is detected
SocketNum	Socket number	UINT	Refer to communication instruction specifications	0	Specify the number of the Socket. Different Socket can be identified by SocketNum.
Mode	Receiving mode	USINT	0, 1	0	Set the way of data receiving. 0: Cyclic receiving 1: Single receiving
DataSaveMode	Save mode of the received data	USINT	0, 1	0	Set the way in which received data is saved to the buffer. 0: Continuous (Mode is 0, valid for cyclic receiving) 1: Overwriting
DataSaveAddr	Cache starting address for the received data	USINT	%MB0~%MB65535	Required field	Set the starting address where the received data is stored.
ReceiveLength	Receive data length	UINT	1~1000	0	Set the length of the received data (Unit: Byte)

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when data receiving is completed

Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	The instruction is controlling the Socket	BOOL	TRUE / FALSE	TRUE when the instruction controls the Socket normally
Aborted	The instruction is aborted	BOOL	TRUE / FALSE	TRUE when the instruction is aborted
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Received	Receiving completed	BOOL	TRUE / FALSE	TRUE when data receiving is completed
Receiving	Receiving	BOOL	TRUE / FALSE	TRUE in data receiving
ReceivedLength	The data length of a message	UINT	0~65535	The data length of a message (Unit: Byte)

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Execute changes from TRUE to FALSE
Busy	When Execute changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Aborted changes from FALSE to TRUE When Error changes from FALSE to TRUE
Active	When the instruction starts controlling the Socket	When Done changes from FALSE to TRUE When Aborted changes from FALSE to TRUE When Error changes from FALSE to TRUE
Aborted	When the instruction is aborted	When Execute changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Execute changes from TRUE to FALSE
Received	When data receiving is completed	When Execute changes from TRUE to FALSE Data is ready to be received
Receiving	Data is ready to be received	When data receiving is completed

◆ Function description

• Basic function description

This instruction is used to receive Socket data. When the instruction triggers execution, the received data is stored in the cache area in the mode set by DataSaveMode, and the starting address of the cache area is specified by the parameter DataSaveAddr.

• Single receiving

When the parameter Mode is 1, it indicates a single receiving. When it is necessary to execute a single receiving, set Mode to 1 before executing the instruction (Execute changes from FALSE to TRUE).

During single receiving, the instruction only receives data once each time it triggers execution. After receiving the data, the output variables Received and Done become TRUE, and the execution of the instruction is completed. If another receiving is needed, it is necessary to re-trigger the execution of the instruction.

If the actual length of the received data exceeds the length specified by the input variable ReceiveLength, the instruction will report an error, and no data can be received at the address of the received data cache. In this case, make the length specified by the input variable ReceiveLength longer, and then re-execute the instruction.

• Continuous receiving

When the parameter Mode is 0, it is continuous receiving. When users need to execute continuous receiving data, set Mode to 0 before executing the instruction (Execute changes from FALSE to TRUE).

When receiving data continuously, a new sum of data is received, and Received becomes TRUE for one period and then changes to FALSE. Receiving and Received are mutually exclusive, i.e., when Receiving is TRUE, Received is FALSE; when Receiving is FALSE, Received is TRUE.

When receiving data continuously, the data storage mode can be set to continuous or overwriting with the parameter DataSaveMode.

When DataSaveMode (value=1) is set to overwriting Mode, newly received data is always stored from the starting address of the receiving data cache. ReceivedLength is the data length of the latest received data (the number of bytes of data in a message), Receiving and Received are mutually exclusive.

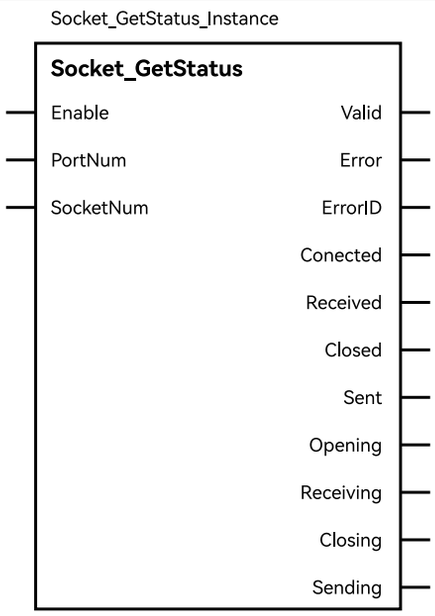
When DataSaveMode (value=0) is set to continuous mode, the newly received data is stored after the previous data. When the actual length of the received data is less than the length specified by ReceiveLength, the data is received in succession, ReceivedLength is the data length of the latest received data (the number of bytes of data in a message), Receiving and Received are mutually exclusive. When the actual length of the received data is greater than or equal to the length specified by ReceiveLength, the output variable Done becomes TRUE, Receiving is FALSE, Received is TRUE, and no more data is received. If there is a need to receive data again, it is necessary to re-trigger the execution of this instruction.

- **Precautions**

This instruction needs to be executed after the Socket function is enabled, and the Socket function can be enabled by the Socket_Open instruction.

1.17 Socket_GetStatus

Get Socket status. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_GetStatus	Get Socket status	FB		Socket_GetStatus_Instance(Enable:= parameter, PortNum:=parameter, SocketNum:= parameter, Valid=> parameter, Error=> parameter, ErrorID=> parameter, Conected=> parameter, Received=> parameter, Closed=> parameter, Sent=> parameter, Opening=> parameter, Receiving=> parameter, Closing=> parameter, Sending=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	The instruction will be executed if it is set to TRUE, or not executed if it is set to FALSE.
PortNum	Ethernet hardware interface number	UINT	Reserved	Reserved	Reserved
SocketNum	Socket number	UINT	Refer to communication instruction specifications	Required field	Specify the number of the Socket. Different Socket can be identified by SocketNum.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the instruction is executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Conected	Connection established	BOOL	TRUE / FALSE	TRUE if the connection is completed successfully, and FALSE if it is not
Received	Received successfully	BOOL	TRUE / FALSE	TRUE if the receiving is completed successfully, and FALSE if it is not
Closed	Connection closed	BOOL	TRUE / FALSE	TRUE when the connection is closed and FALSE when it is not.
Sent	Data sending is completed	BOOL	TRUE / FALSE	TRUE if the sending is completed successfully, and FALSE if it is not
Opening	Socket data exchange is opening	BOOL	TRUE / FALSE	TRUE if Socket data exchange is enabled and FALSE if it is not
Receiving	Data is being received	BOOL	TRUE / FALSE	TRUE in Socket data receiving and FALSE if it is not
Closing	Socket data exchange is closing	BOOL	TRUE / FALSE	TRUE if the Socket data exchange is closed, and FALSE if it is not
Sending	Data is being sent	BOOL	TRUE / FALSE	TRUE if the Socket data is being sent, and FALSE if it is not

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encountered an exception during the instruction execution process.	When Enable changes from TRUE to FALSE When the error is eliminated
Conected	When the connection is established successfully	When Enable changes from TRUE to FALSE When the connection is cut off
Received	When data receiving is completed	When Enable changes from TRUE to FALSE
Closed	When the connection is closed	When Enable changes from TRUE to FALSE When the connection is established successfully
Sent	Sent successfully	When Enable changes from TRUE to FALSE When re-sending
Opening	Socket data exchange is opening	When Enable changes from TRUE to FALSE When Socket data exchange opening is completed
Receiving	When data receiving is started	When Enable changes from TRUE to FALSE When data receiving is completed
Closing	Socket data exchange is closing	When Enable changes from TRUE to FALSE When Socket data exchange closing is completed
Sending	When data is started to be sent	When Enable changes from TRUE to FALSE When data sending is completed

◆ Function description

This instruction is used to get the current status of Socket.

1.18 Socket_Manage

Enable or disable Socket function. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Socket_Manage	Socket management	FB	Socket_Manage_Instance **Socket_Manage** Enable — Valid PortNum — Ready EnableSocket — PhysicalLinkBroken	Socket_Manage_Instance(Enable:= parameter, PortNum:=parameter, EnableSocket:= parameter, Valid=> parameter, Ready=> parameter, PhysicalLinkBroken=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	The instruction will be executed if it is set to TRUE, or not executed if it is set to FALSE.
PortNum	Ethernet hardware interface number	UINT	Reserved	Reserved	Reserved
EnableSocket	Enable or disable Socket function	BOOL	TRUE / FALSE	FALSE	Enable or disable Socket function. TRUE: Enable FALSE: Disable

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the instruction is executed normally
Ready	Socket function is enabled	BOOL	TRUE / FALSE	TRUE: Socket is enabled FALSE: Socket is disabled
PhysicalLinkBroken	Physical connection is disconnected	BOOL	TRUE / FALSE	TRUE: Physical connection is disconnected FALSE: Physical connection is connected normally

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Ready	When Enable is TRUE and the Socket function is successfully enabled	When Enable changes from TRUE to FALSE When EnableSocket changes from TRUE to FALSE When the Ethernet port is physically disconnected
PhysicalLinkBroken	When EnableSocket is TRUE and the host Ethernet port is physically disconnected	When Enable changes from TRUE to FALSE When the Ethernet port is physically re-connected

◆ Function description

This instruction is used to enable or disable the Socket function. The Socket function enables sending and receiving of customized Ethernet data.

Note 1: The version of the M511S controller is 1.01.07 and earlier, and the version of the M312 controller is 1.01.06 and earlier. It is necessary to enable the Socket function by executing the Socket_Manage instruction before using other Socket-re-

lated instructions.

Note 2: The version of M511S controller is 1.01.07 or later, and the version of M312 controller is 1.01.06 or later, so it is not recommended to use this instruction. When this instruction is not used, other Socket related instructions can be used directly; Socket_Open can be used to open the Socket function, and Socket_Close can be used to close the Socket function.

If using the instruction: When this instruction is executed (Enable is TRUE), the Socket function can still be enabled or disabled by the input variable EnableSocket. When EnableSocket is TRUE, the Socket function is enabled; when EnableSocket is FALSE, the Socket function is disabled.

1.19 Ethernet Socket communication example TCP

1.19.1 Socket data exchange example

- **Target demand**

The two M-Series PLCs exchange customized data using the Socket function.

The IP address and port of the controller are as follows:

Item	Client controller	Item	Server controller
IP address	192.168.1.1	IP address	192.168.1.10
Port number	10000	Port number	20000

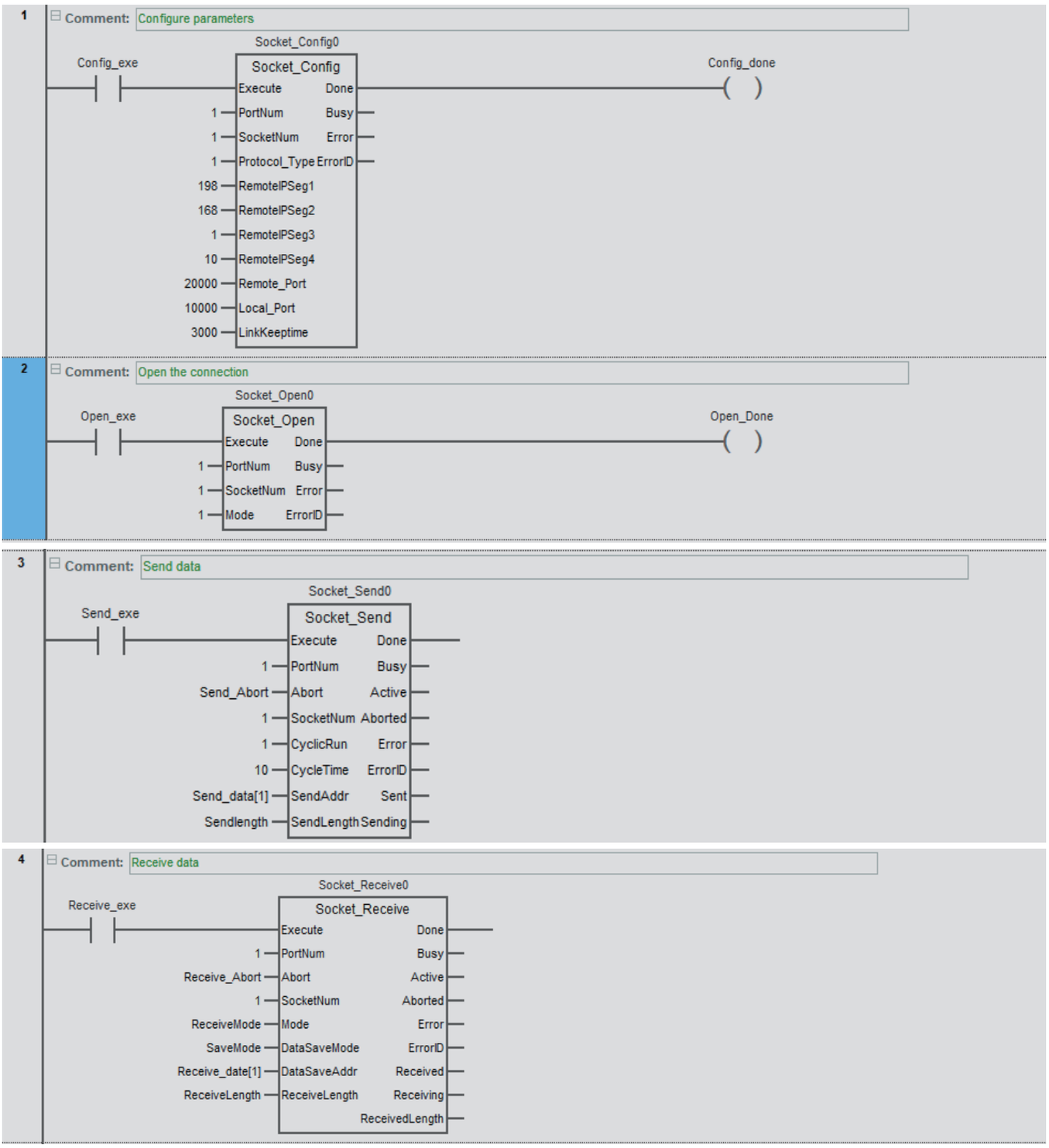
- **Demand analysis**

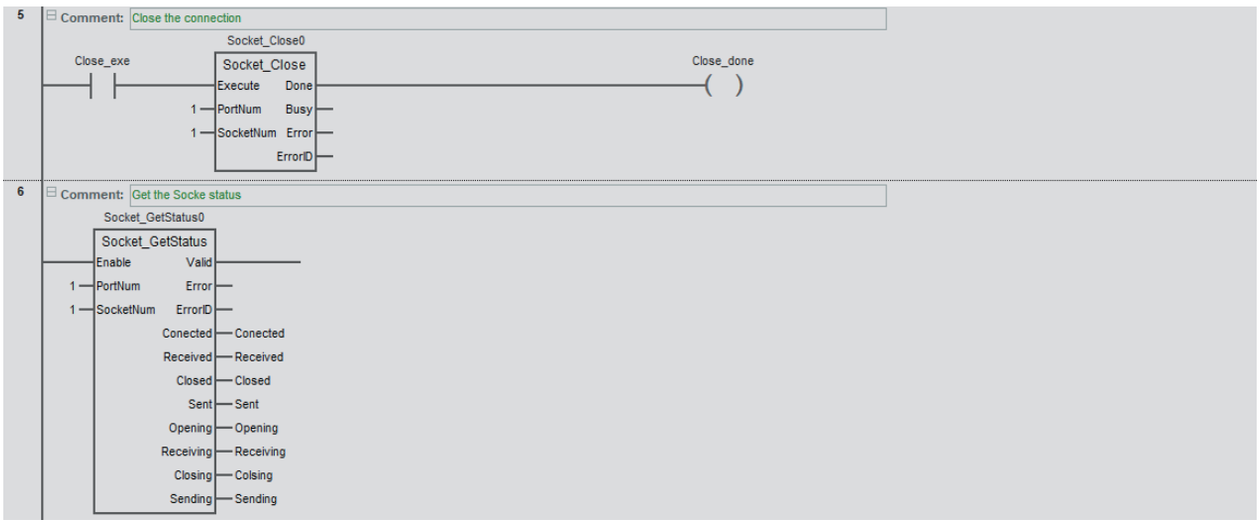
Write programs based on the demand analysis.

The client controller program variables are programmed as shown in the following table:

Required field	Name	Address	Category	Initial value	Comment
VAR	Socket_Config0		Socket_Config		Socket configuration instruction
VAR	Config_exe		BOOL		Execution conditions for Socket configuration instruction
VAR	Config_done		BOOL		Socket configuration parameter completion bit
VAR	Socket_Open0		Socket_Open		Socket port opening instruction
VAR	Open_exe		BOOL		Execution conditions for Socket port opening instruction
VAR	Open_Done		BOOL		Socket opening completion bit
VAR	Socket_Send0		Socket_Send		Socket data sending instruction
VAR	Send_Exe		BOOL		Execution conditions for Socket data sending
VAR	Send_Abort		BOOL		Abort Socket data sending
VAR	Send_data	%MB1000	ARRAY[1..100] OF USINT	[100(0)]	Cache address for Socket sending
VAR	SendLength		UINT	100	Data length of Socket sending
VAR	Socket_Receive0		Socket_Receive		Socket data receiving instruction
VAR	Receive_Exe		BOOL		Execution conditions for Socket data receiving
VAR	Receive_Abort		BOOL		Abort Socket data receiving
VAR	ReceiveMode		USINT		Socket data receiving method
VAR	SaveMode		USINT	1	Socket receiving data storage mode, 0 for the continuous data storage, 1 for overwriting data storage.
VAR	ReceiveBuffer	%MB2000	ARRAY[1..100] OF USINT	[100(0)]	Cache address for Socket data receiving
VAR	ReceiveLength		UINT	100	Data length of Socket receiving
VAR	Socket_Close0		Socket_Close		Socket port closing instruction
VAR	Close_exe		BOOL		Execution conditions for Socket port closing instruction
VAR	Close_Done		BOOL		Socket closing completion bit
VAR	Socket_GetStatus0		Socket_GetStatus		Instruction for getting the operating status of the Socket
VAR	Connected		BOOL		Connection established
VAR	Received		BOOL		Received successfully
VAR	Closed		BOOL		Connection closed
VAR	Sent		BOOL		Data sending is completed
VAR	Opening		BOOL		Socket connection port is opening
VAR	Receiving		BOOL		Data is being received
VAR	Closing		BOOL		Socket connection port is closed
VAR	sending		BOOL		Data is being sent

- **The client controller LD program is as follows:**





- **The client controller ST program is as follows:**

```
// Configure parameters
Socket_Config0(Execute:=Config_exe ,
PortNum:= 1,
SocketNum:=1 ,
Protocol_Type:=1 ,
RemoteIPSeg1:=192 ,
RemoteIPSeg2:=168 ,
RemoteIPSeg3:=1 ,
RemoteIPSeg4:=10 ,
Remote_Port:=20000 ,
Local_Port:=10000 ,
LinkKeeptime:=3000 ,
Done=>Config_done
);
// Open the connection
Socket_Open0(Execute:= Open_exe ,
PortNum:=1 ,
SocketNum:=1 ,
Mode:=1 ,
Done=>Open_Done
);
// Send data
Socket_Send0(Execute:=Send_Exe ,
PortNum:= 1,
Abort:=Send_Abort ,
SocketNum:=1 ,
```

```
CyclicRun:=1 ,
CycleTime:=10 ,
SendAddr:= Send_data[1] ,
SendLength:=SendLength ,
);
//Receive data
Socket_Receive0(Execute:=Receive_Exe ,
PortNum:=1 ,
Abort:= Receive_Abort,
SocketNum:=1 ,
Mode:=ReceiveMode ,
DataSaveMode:=SaveMode ,
DataSaveAddr:= Receive_date[1] ,
ReceiveLength:=ReceiveLength ,
);
// Close the connection
Socket_Close0(Execute:=Close_exe ,
PortNum:=1 ,
SocketNum:=1 ,
Done=>Close_done ,
);
//Get the Socket status
Socket_GetStatus0(Enable:=1,
PortNum:=1 ,
SocketNum:= 1,
Conected=>Connected,
Received=>Received,
Closed=>Closed,
Sent=>Sent,
Opening=>Opening,
Receiving=>Receiving,
Closing=>Closing,
Sending=>Sending
);
```

- **Program flow description:**

Step 1: Configure Socket parameters (Socket_Config), which include connection mode, remote device IP, port number, and link keeping time. Set the variable Config_exe to TRUE to trigger the execution of the Socket parameter configuration instruction, and if the variable Config_done is TRUE, the configuration is successful.

Step 2: Open the Socket port to establish a connection (Socket_Open), including the Socket working mode. When working in client mode (Mode=1), ensure that the external server is already open and in a connectable status, and then set the Open_exe variable to TRUE in order to trigger the Socket_Open instruction to open the Socket port and establish a connection. Use the Socket_GetStatus instruction to get the status of the Socket, and check whether a Socket connection has been successfully established by checking whether the Connected variable is TRUE.

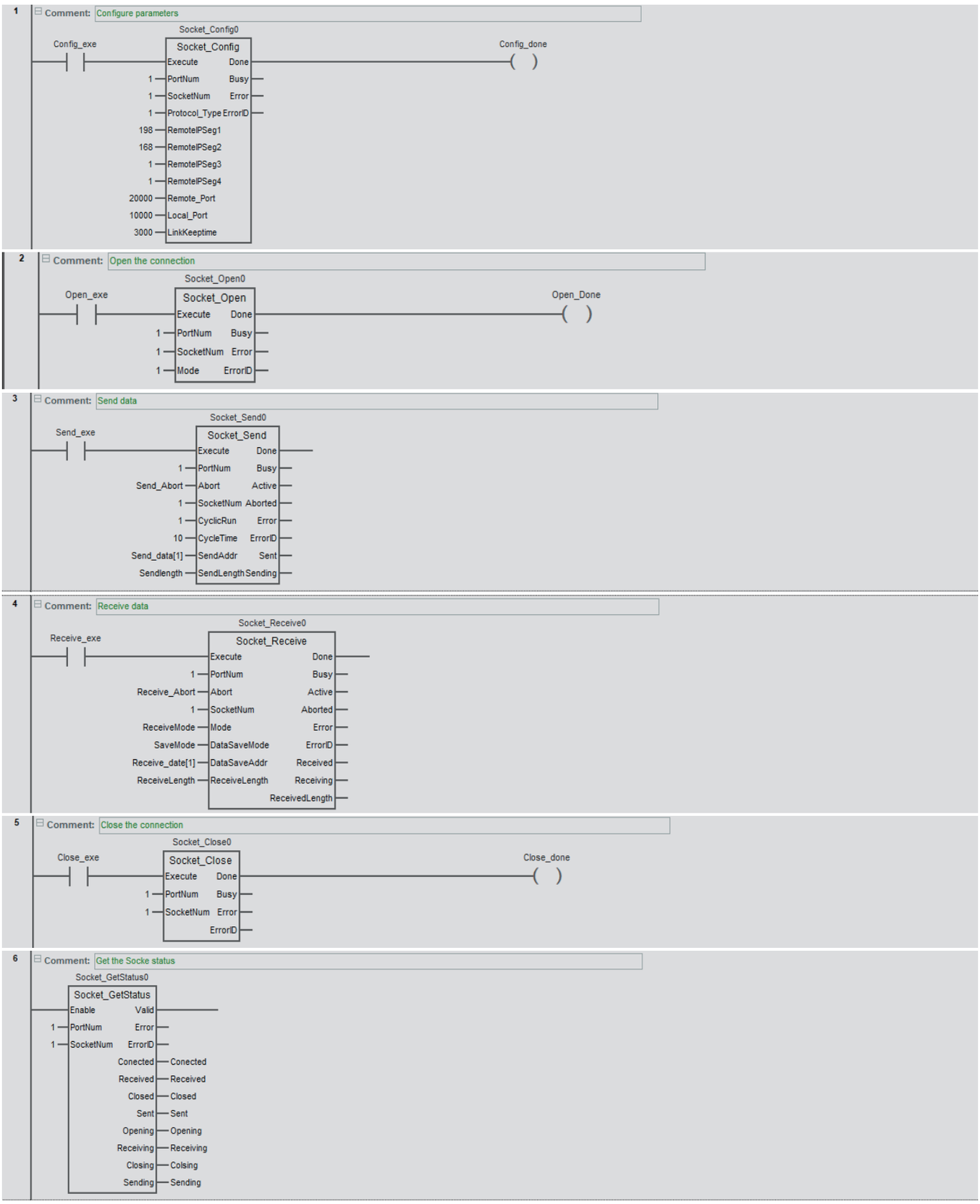
Step 3:Sending and receiving Socket data. The sending and receiving instructions can be executed only after the Socket port is opened and the connection is successfully established. In this example, the variable Send_Exe is set to TRUE to trigger Socket sending, (Socket_Send1.CyclicRun) is set to 1 to send cyclically, and the variable Receive1_Exe is set to TRUE to trigger Socket receiving.

Step 4: Close the Socket port to break the connection (Socket_Close). Set the variable Close_exe to TRUE to trigger the instruction for closing the Socket port, and if the variable Close_done is TRUE, it means that the closing is successful.

- **The server controller program variables are programmed as shown in the following table:**

Required field	Name	Address	Category	Initial value	Comment
VAR	Socket_Config0		Socket_Config		Socket configuration instruction
VAR	Config_exe		BOOL		Execution conditions for Socket configuration instruction
VAR	Config_done		BOOL		Socket configuration parameter completion bit
VAR	Socket_Open0		Socket_Open		Socket port opening instruction
VAR	Open_exe		BOOL		Execution conditions for Socket port opening instruction
VAR	Open_Done		BOOL		Socket opening completion bit
VAR	Socket_Send0		Socket_Send		Socket data sending instruction
VAR	Send_Exe		BOOL		Execution conditions for Socket data sending
VAR	Send_Abort		BOOL		Abort Socket data sending
VAR	Send_data	%MB3000	ARRAY[1..100] OF USINT	[100(0)]	Cache address for Socket sending
VAR	SendLength		UINT	100	Data length of Socket sending
VAR	Socket_Receive0		Socket_Receive		Socket data receiving instruction
VAR	Receive_Exe		BOOL		Execution conditions for Socket data receiving
VAR	Receive_Abort		BOOL		Abort Socket data receiving
VAR	ReceiveMode		USINT		Socket data receiving method
VAR	SaveMode		USINT	1	Socket receiving data storage mode, 0 for the continuous data storage, 1 for overwriting data storage
VAR	ReceiveBuffer	%MB4000	ARRAY[1..100] OF USINT	[100(0)]	Cache address for Socket data receiving
VAR	ReceiveLength		UINT	100	Data length of Socket receiving
VAR	Socket_Close0		Socket_Close		Socket port closing instruction
VAR	Close_exe		BOOL		Execution conditions for Socket port closing instruction
VAR	Close_Done		BOOL		Socket closing completion bit
VAR	Socket_GetStatus0		Socket_GetStatus		Instruction for getting the operating status of the Socket
VAR	Connected		BOOL		Connection established
VAR	Received		BOOL		Received successfully
VAR	Closed		BOOL		Connection closed
VAR	Sent		BOOL		Data sending is completed
VAR	Opening		BOOL		Socket connection port is opening
VAR	Receiving		BOOL		Data is being received
VAR	Closing		BOOL		Socket connection port is closed
VAR	Sending		BOOL		Data is being sent

- **The server controller LD program is as follows:**



- **The server controller ST program is as follows:**

// Configure parameters

Socket_Config0(Execute:=Config_exe ,

PortNum:= 1,

SocketNum:=1 ,

```
Protocol_Type:=1 ,
RemoteIPSeg1:=192 ,
RemoteIPSeg2:=168 ,
RemoteIPSeg3:=1 ,
RemoteIPSeg4:=1 ,
Remote_Port:=10000 ,
Local_Port:=20000 ,
LinkKeeptime:=3000 ,
Done=>Config_done
);
// Open the connection
Socket_Open0(Execute:= Open_exe ,
PortNum:=1 ,
SocketNum:=1 ,
Mode:=0 ,
Done=>Open_Done
);
// Send data
Socket_Send0(Execute:=Send_Exe ,
PortNum:= 1,
Abort:=Send_Abort ,
SocketNum:=1 ,
CyclicRun:=1 ,
CycleTime:=10 ,
SendAddr:= Send_data[1] ,
SendLength:=SendLength ,
);
//Receive data
Socket_Receive0(Execute:=Receive_Exe ,
PortNum:=1 ,
Abort:= Receive_Abort,
SocketNum:=1 ,
Mode:=ReceiveMode ,
DataSaveMode:=SaveMode ,
DataSaveAddr:= Receive_date[1] ,
ReceiveLength:=ReceiveLength ,
);
// Close the connection
```

```
Socket_Close0(Execute:=Close_exe ,
PortNum:=1 ,
SocketNum:=1 ,
Done=>Close_done ,
);
//Get the Socket status

Socket_GetStatus0(Enable:=1,
PortNum:=1 ,
SocketNum:= 1,
Conected=>Connected,
Received=>Received,
Closed=>Closed,
Sent=>Sent,
Opening=>Opening,
Receiving=>Receiving,
Closing=>Closing,
Sending=>Sending
);
```

- **Program flow description:**

Step 1: Configure Socket parameters (Socket_Config), which include connection mode, remote device IP, port number, and link keeping time. Set the variable Config_exe to TRUE to trigger the execution of the Socket parameter configuration instruction, and if the variable Config_done is TRUE, the configuration is successful.

Step 2: Open the Socket port to establish a connection (Socket_Open), including the Socket working mode. When working in client mode (Mode=0), set the Open_exe variable to TRUE to trigger the Socket_Open instruction , thus opening the Socket port and establishing a connection. Use the instruction Socket_GetStatus to get the status of the Socket, and check whether a Socket port is open by checking whether the Opening variable is TRUE.

Step 3: Sending and receiving Socket data. The sending and receiving instructions can be executed only after the Socket port is opened and the connection is successfully established. In this example, the variable Send_Exe is set to TRUE to trigger Socket sending, (Socket_Send1.CyclicRun) is set to 1 to send data cyclically, and the variable Receive1_Exe is set to TRUE to trigger Socket receiving.

Step 4: Close the Socket port to break the connection (Socket_Close). Set the variable Close_exe to TRUE to trigger the instruction for closing the Socket port, and if the variable Close_done is TRUE, it means that the closing is successful.

1.20 Ethernet Socket communication example UDP

1.20.1 Socket data exchange example

- **Target demand**

The two M-Series PLCs exchange customized data using the Socket function.

The IP address and port of the controller are as follows:

Item	Controller 1	Item	Controller 2
IP address	192.168.1.1	IP address	192.168.1.10
Port number	10000	Port number	20000

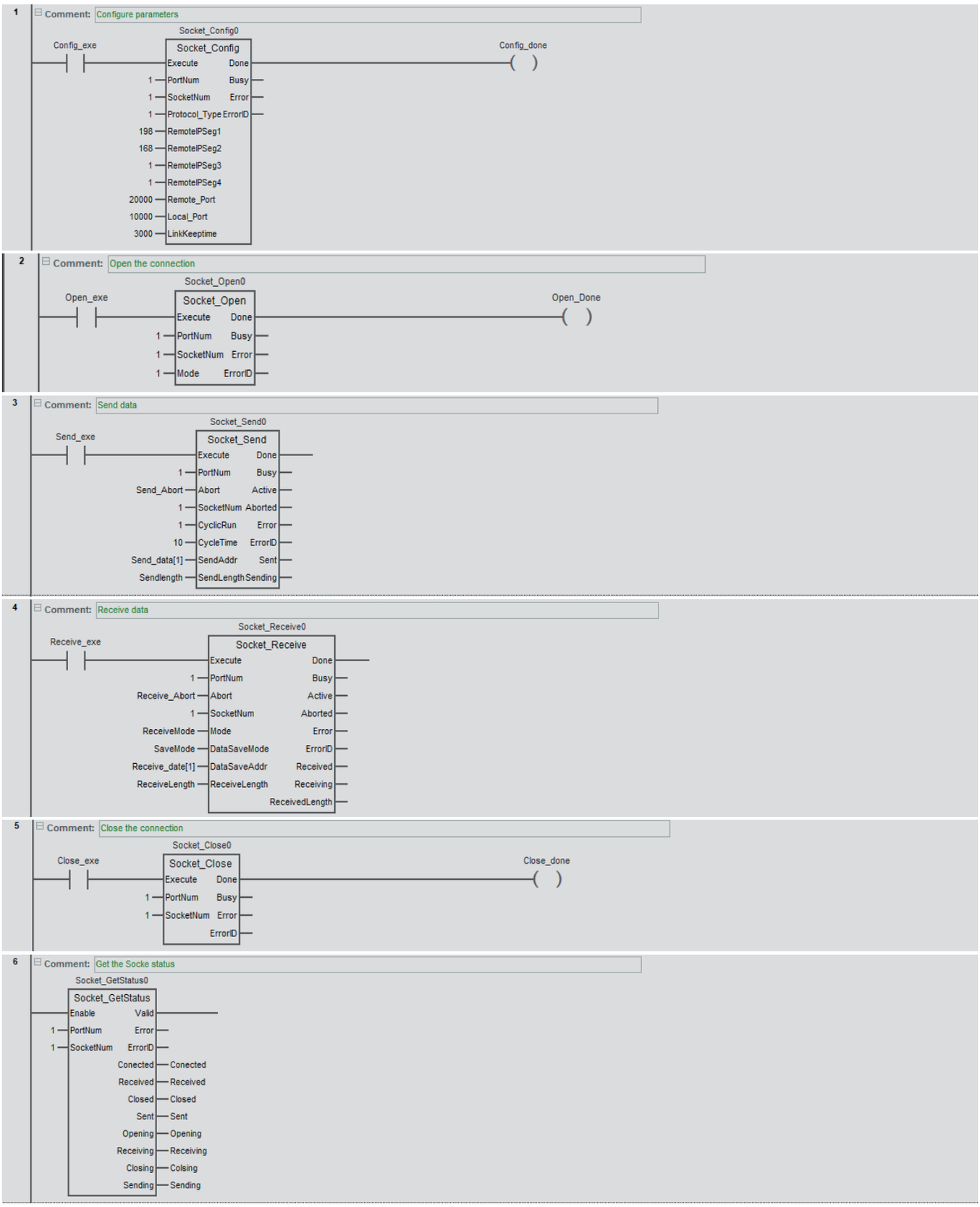
- **Demand analysis**

Write programs based on the demand analysis.

Controller 1 program variables are programmed as shown in the following table:

Required field	Name	Address	Category	Initial value	Comment
VAR	Socket_Config0		Socket_Config		Socket configuration instruction
VAR	Config_exe		BOOL		Execution conditions for Socket configuration instruction
VAR	Config_done		BOOL		Socket configuration parameter completion bit
VAR	Socket_Open0		Socket_Open		Socket port opening instruction
VAR	Open_exe		BOOL		Execution conditions for Socket port opening instruction
VAR	Open_Done		BOOL		Socket opening completion bit
VAR	Socket_Send0		Socket_Send		Socket data sending instruction
VAR	Send_Exe		BOOL		Execution conditions for Socket data sending
VAR	Send_Abort		BOOL		Abort Socket data sending
VAR	Send_data	%MB1000	ARRAY[1..100] OF USINT	[100(0)]	Cache address for Socket sending
VAR	SendLength		UINT	100	Data length of Socket sending
VAR	Socket_Receive0		Socket_Receive		Socket data receiving instruction
VAR	Receive_Exe		BOOL		Execution conditions for Socket data receiving
VAR	Receive_Abort		BOOL		Abort Socket data receiving
VAR	ReceiveMode		USINT		Socket data receiving method
VAR	SaveMode		USINT	1	Socket receiving data storage mode, 0 for the continuous data storage, 1 for overwriting data storage.
VAR	ReceiveBuffer	%MB2000	ARRAY[1..100] OF USINT	[100(0)]	Cache address for Socket data receiving
VAR	ReceiveLength		UINT	100	Data length of Socket receiving
VAR	Socket_Close0		Socket_Close		Socket port closing instruction
VAR	Close_exe		BOOL		Execution conditions for Socket port closing instruction
VAR	Close_Done		BOOL		Socket closing completion bit
VAR	Socket_GetStatus0		Socket_GetStatus		Instruction for getting the operating status of the Socket
VAR	Connected		BOOL		Connection established
VAR	Received		BOOL		Received successfully
VAR	Closed		BOOL		Connection closed
VAR	Sent		BOOL		Data sending is completed
VAR	Opening		BOOL		Socket connection port is opening
VAR	Receiving		BOOL		Data is being received
VAR	Closing		BOOL		Socket connection port is closed
VAR	sending		BOOL		Data is being sent

- **The controller 1 LD program is as follows:**



- **The controller 1 ST program is as follows:**

```
// Configure parameters
Socket_Config0(Execute:=Config_exe ,
PortNum:= 1,
```

```
SocketNum:=1 ,
Protocol_Type:=0 ,
RemoteIPSeg1:=192 ,
RemoteIPSeg2:=168 ,
RemoteIPSeg3:=1 ,
RemoteIPSeg4:=10 ,
Remote_Port:=20000 ,
Local_Port:=10000 ,
LinkKeeptime:=3000 ,
Done=>Config_done
);
// Open the connection
Socket_Open0(Execute:= Open_exe ,
PortNum:=1 ,
SocketNum:=1 ,
Mode:=0 ,
Done=>Open_Done
);
// Send data
Socket_Send0(Execute:=Send_Exe ,
PortNum:= 1,
Abort:=Send_Abort ,
SocketNum:=1 ,
CyclicRun:=1 ,
CycleTime:=10 ,
SendAddr:= Send_data[1] ,
SendLength:=SendLength ,
);
//Receive data
Socket_Receive0(Execute:=Receive_Exe ,
PortNum:=1 ,
Abort:= Receive_Abort,
SocketNum:=1 ,
Mode:=ReceiveMode ,
DataSaveMode:=SaveMode ,
DataSaveAddr:= Receive_date[1] ,
ReceiveLength:=ReceiveLength ,
);
```

```
// Close the connection
Socket_Close0(Execute:=Close_exe ,
PortNum:=1 ,
SocketNum:=1 ,
Done=>Close_done ,
);
//Get the Socket status
Socket_GetStatus0(Enable:=1,
PortNum:=1 ,
SocketNum:= 1,
Conected=>Connected,
Received=>Received,
Closed=>Closed,
Sent=>Sent,
Opening=>Opening,
Receiving=>Receiving,
Closing=>Closing,
Sending=>Sending
);
```

- **Program flow description:**

Step 1: Configure Socket parameters (Socket_Config), which include connection mode, remote device IP, port number, and link keeping time. Set the variable Config_exe to TRUE to trigger the execution of the Socket parameter configuration instruction, and if the variable Config_done is TRUE, the configuration is successful.

Step 2: Open the Socket port to establish a connection (Socket_Open). There is no need to set the connection mode under the UDP mode. Open_exe variable is set to TRUE to trigger the Socket_Open instruction, thus opening the Socket port and establishing a connection. Use the Socket_GetStatus instruction to get the status of the Socket. Check whether a Socket port is open by checking whether the Opening variable is TRUE. Establish a connection when both ends of the UDP are in the Opening state at the same time.

Step 3: Sending and receiving Socket data. The sending and receiving instructions can be executed only after the Socket port is opened and the connection is successfully established. In this example, the variable Send_Exe is set to TRUE to trigger Socket sending, (Socket_Send1.CyclicRun) is set to 1 to send data cyclically, and the variable Receive1_Exe is set to TRUE to trigger Socket receiving.

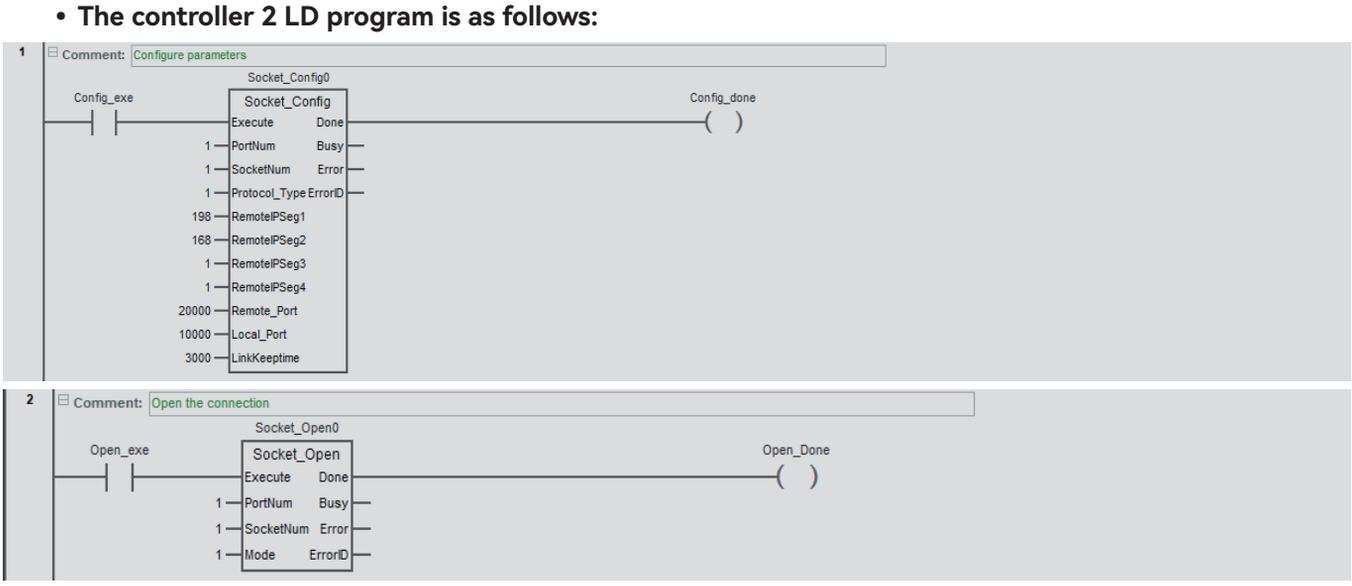
Step 4: Close the Socket port to break the connection (Socket_Close). Set the variable Close_exe to TRUE to trigger the instruction for closing the Socket port, and if the variable Close_done is TRUE, it means that the closing is successful.

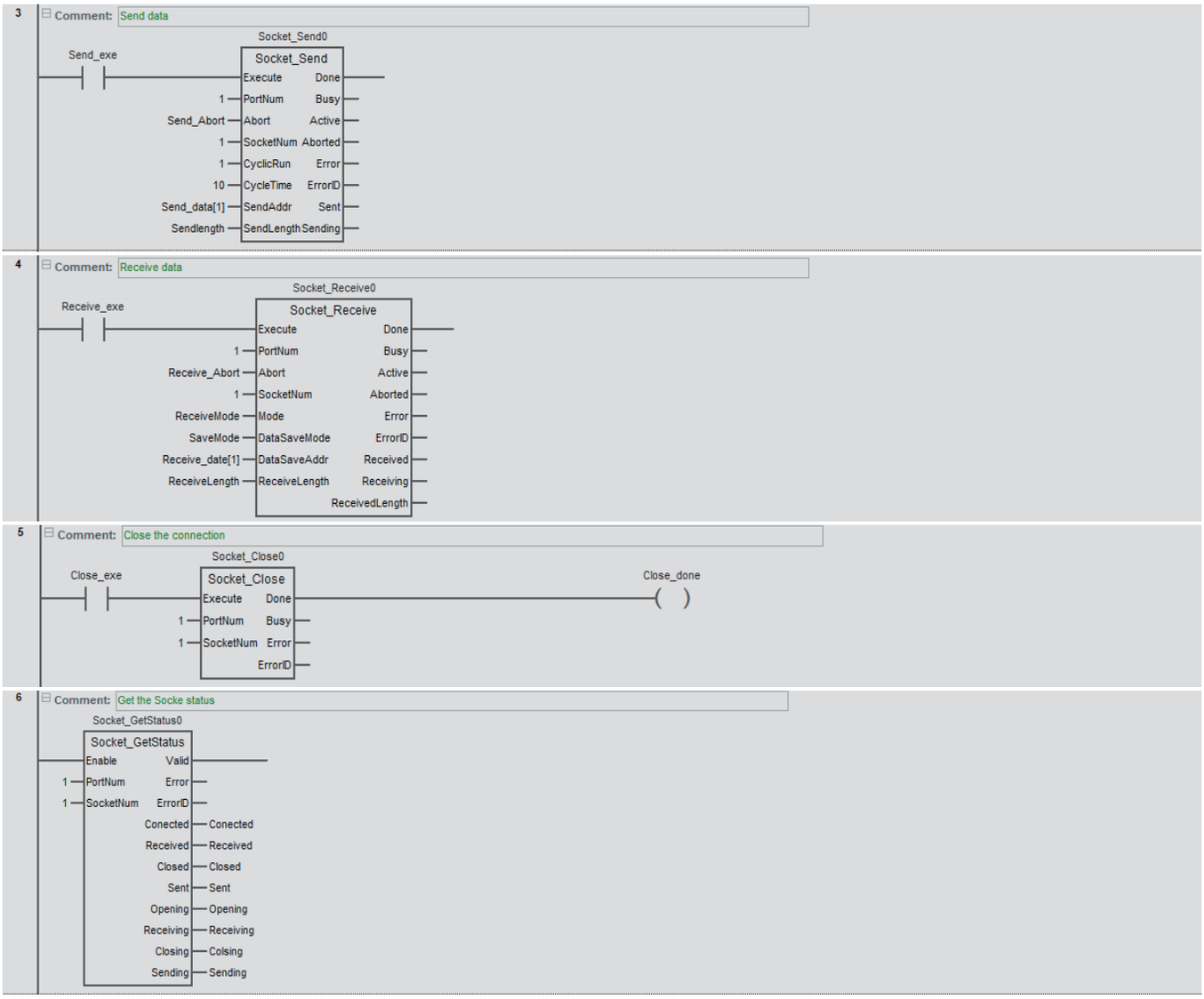
- **Controller 2 program variables are programmed as shown in the following table:**

Required field	Name	Address	Category	Initial value	Comment
VAR	Socket_Config0		Socket_Config		Socket configuration instruction
VAR	Config_exe		BOOL		Execution conditions for Socket configuration instruction
VAR	Config_done		BOOL		Socket configuration parameter completion bit
VAR	Socket_Open0		Socket_Open		Socket port opening instruction

VAR	Open_exe		BOOL		Execution conditions for Socket port opening instruction
VAR	Open_Done		BOOL		Socket opening completion bit
VAR	Socket_Send0		Socket_Send		Socket data sending instruction
VAR	Send_Exe		BOOL		Execution conditions for Socket data sending
VAR	Send_Abort		BOOL		Abort Socket data sending
VAR	Send_data	%MB3000	ARRAY[1..100] OF USINT	[100(0)]	Cache address for Socket sending
VAR	SendLength		UINT	100	Data length of Socket sending
VAR	Socket_Receive0		Socket_Receive		Socket data receiving instruction
VAR	Receive_Exe		BOOL		Execution conditions for Socket data receiving
VAR	Receive_Abort		BOOL		Abort Socket data receiving
VAR	ReceiveMode		USINT		Socket data receiving method
VAR	SaveMode		USINT	1	Socket receiving data storage mode, 0 for the continuous data storage, 1 for overwriting data storage.
VAR	ReceiveBuffer	%MB4000	ARRAY[1..100] OF USINT	[100(0)]	Cache address for Socket data receiving
VAR	ReceiveLength		UINT	100	Data length of Socket receiving
VAR	Socket_Close0		Socket_Close		Socket port closing instruction
VAR	Close_exe		BOOL		Execution conditions for Socket port closing instruction
VAR	Close_Done		BOOL		Socket closing completion bit
VAR	Socket_GetStatus0		Socket_GetStatus		Instruction for getting the operating status of the Socket
VAR	Connected		BOOL		Connection established
VAR	Received		BOOL		Received successfully
VAR	Closed		BOOL		Connection closed
VAR	Sent		BOOL		Data sending is completed
VAR	Opening		BOOL		Socket connection port is opening
VAR	Receiving		BOOL		Data is being received
VAR	Closing		BOOL		Socket connection port is closed
VAR	Sending		BOOL		Data is being sent

- **The controller 2 LD program is as follows:**





- **The controller 2 ST program is as follows:**

```
// Configure parameters
Socket_Config0(Execute:=Config_exe ,
PortNum:= 1,
SocketNum:=1 ,
Protocol_Type:=0 ,
RemoteIPSeg1:=192 ,
RemoteIPSeg2:=168 ,
RemoteIPSeg3:=1 ,
RemoteIPSeg4:=1 ,
Remote_Port:=10000 ,
Local_Port:=20000 ,
LinkKeeptime:=3000 ,
Done=>Config_done
);
// Open the connection
```

```
Socket_Open0(Execute:= Open_exe ,
PortNum:=1 ,
SocketNum:=1 ,
Mode:=0 ,
Done=>Open_Done
);
// Send data
Socket_Send0(Execute:=Send_Exe ,
PortNum:= 1,
Abort:=Send_Abort ,
SocketNum:=1 ,
CyclicRun:=1 ,
CycleTime:=10 ,
SendAddr:= Send_data[1] ,
SendLength:=SendLength ,
);
//Receive data
Socket_Receive0(Execute:=Receive_Exe ,
PortNum:=1 ,
Abort:= Receive_Abort,
SocketNum:=1 ,
Mode:=ReceiveMode ,
DataSaveMode:=SaveMode ,
DataSaveAddr:= Receive_date[1] ,
ReceiveLength:=ReceiveLength ,
);
// Close the connection
Socket_Close0(Execute:=Close_exe ,
PortNum:=1 ,
SocketNum:=1 ,
Done=>Close_done ,
);
//Get the Socket status

Socket_GetStatus0(Enable:=1,
PortNum:=1 ,
SocketNum:= 1,
Conected=>Connected,
```

Received=>Received,

Closed=>Closed,

Sent=>Sent,

Opening=>Opening,

Receiving=>Receiving,

Closing=>Closing,

Sending=>Sending

);

- **Program flow description:**

Step 1: Configure Socket parameters (Socket_Config), which include connection mode, remote device IP, port number, and link keeping time. Set the variable Config_exe to TRUE to trigger the execution of the Socket parameter configuration instruction, and if the variable Config_done is TRUE, the configuration is successful.

Step 2: Open the Socket port to establish a connection (Socket_Open). There is no need to set the connection mode under the UDP mode. Open_exe variable is set to TRUE to trigger the Socket_Open instruction to open the Socket port and establish a connection. Use the Socket_GetStatus instruction to get the status of the Socket. Check whether a Socket port is open by checking whether the Opening variable is TRUE. Establish a connection when both ends of the UDP are in the Opening state at the same time.

Step 3: Sending and receiving Socket data. The sending and receiving instructions can be executed only after the Socket port is opened and the connection is successfully established. In this example, the variable Send_Exe is set to TRUE to trigger Socket sending, (Socket_Send1.CyclicRun) is set to 1 to send cyclically, and the variable Receive1_Exe is set to TRUE to trigger Socket receiving.

Step 4: Close the Socket port to break the connection (Socket_Close). Set the variable Close_exe to TRUE to trigger the instruction for closing the Socket port, and if the variable Close_done is TRUE, it means that the closing is successful.

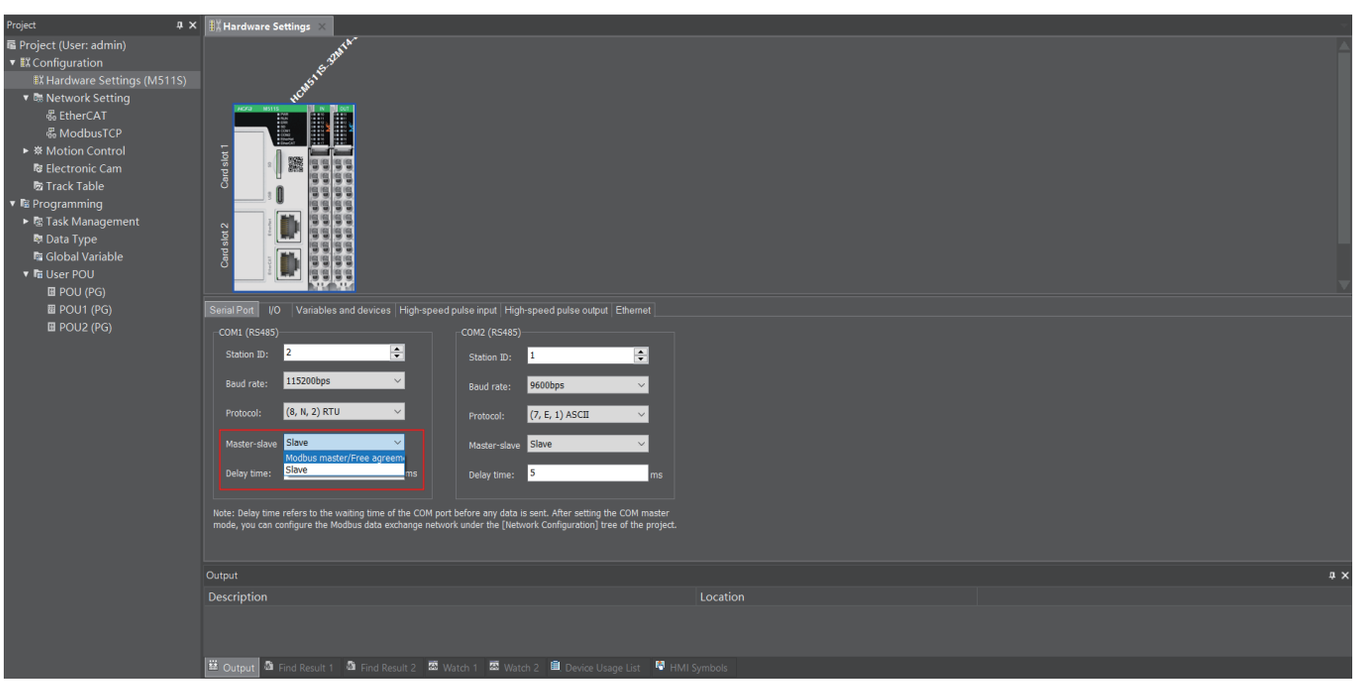
Chapter 2 Serial communication

2.1 Serial communication usage overview

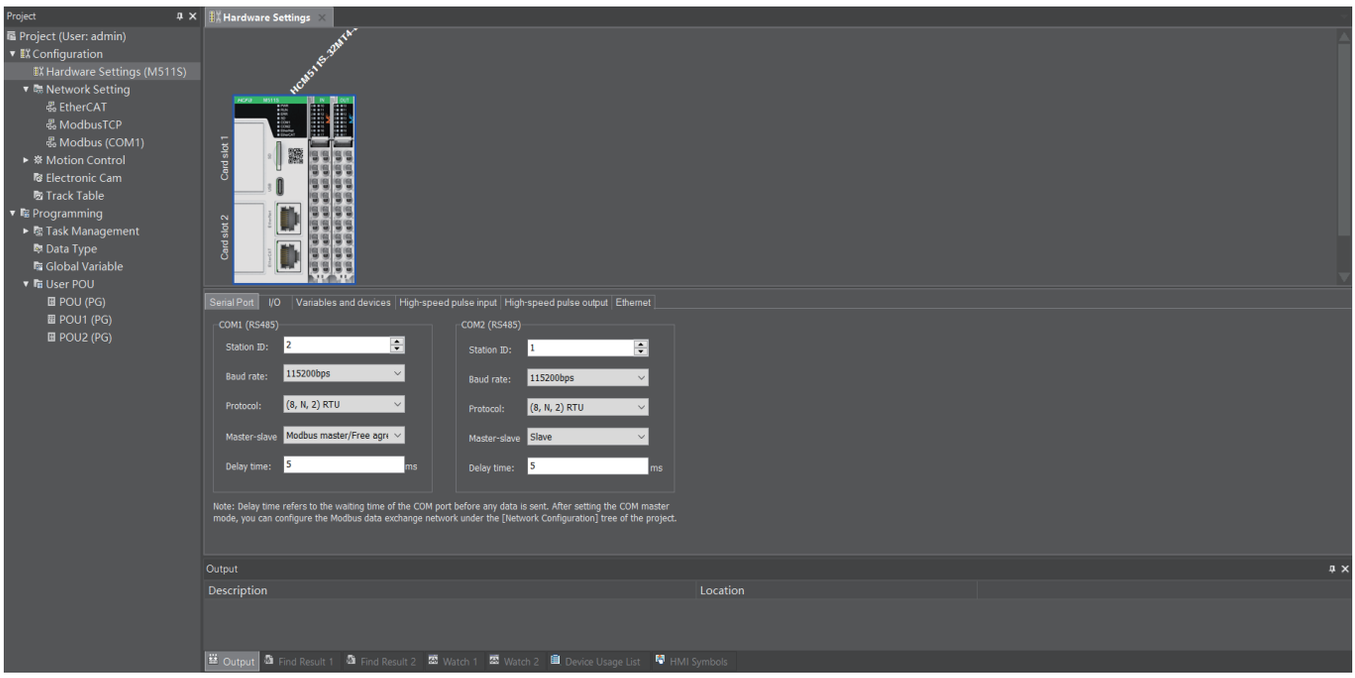
Software configuration using Modbus functionality:

Step	Item	Usage		Description
1	Set serial port master-slave mode	Enable by instructions	Serial_Manage	Set serial port master-slave mode
		Change the serial port master-slave mode to master/free protocol in the hardware configuration interface	Default slave	
2	Set communication protocol	Software configuration interface	Default (7,E,1) ASCII	Set communication protocol
3	Control master	Select “enable by instructions” in the software configuration interface	Modbus_MasterRun	Enable Modbus master function
		Select “enable by default” in the software configuration interface	Enable by default	Enable Modbus master function
		Modbus_MasterStop		Disable Modbus master function
		Modbus_Masterstatus		Get Modbus master status
4	Configure channels	Software configuration interface		Configure the specified data exchange channels
5	Control channels	Select “trigger by program” in the software configuration interface	Modbus_LinkRun	Open the specified data exchange channels
		Select “trigger by default” in the software configuration interface	Enable by default	Open the specified data exchange channels
		Modbus_LinkStop		Close the specified data exchange channels
		Modbus_GetLinkStatus		Get the status of the specified data exchange channels

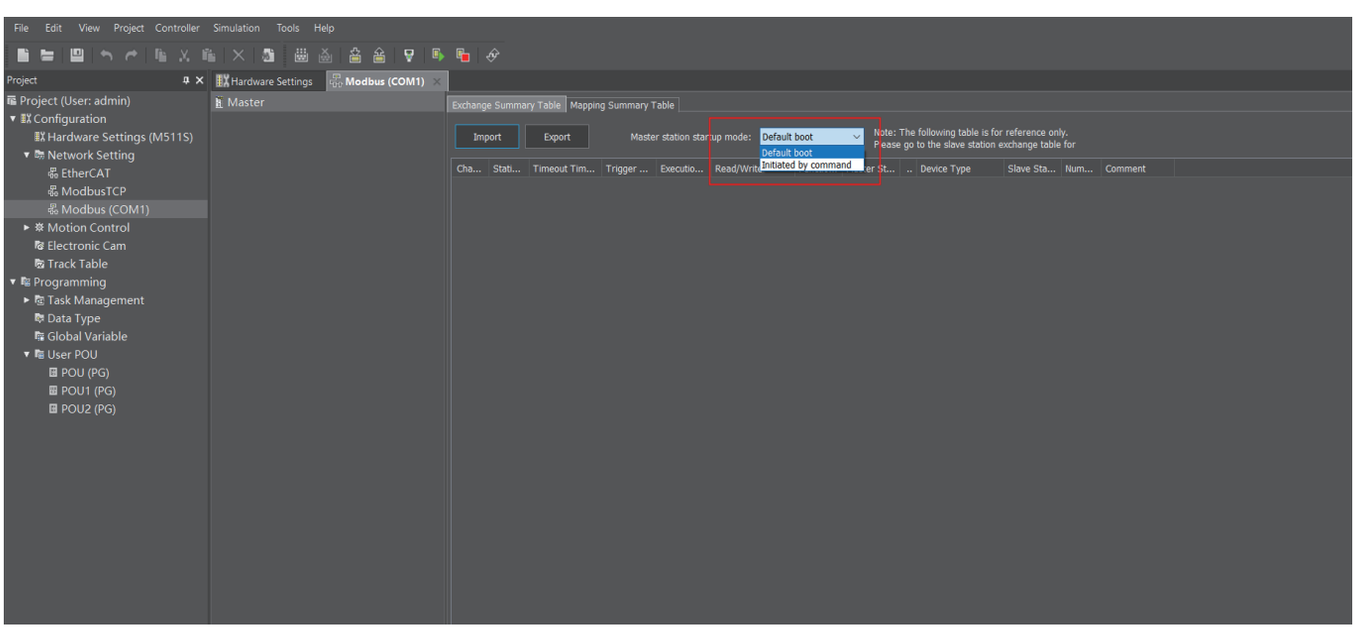
Step 1: Set the serial port master-slave mode of the controller. Set the master-slave mode to Master/Free Protocol in the hardware settings-serial port, or change the master-slave mode of the serial port by Serial_Manage instruction.



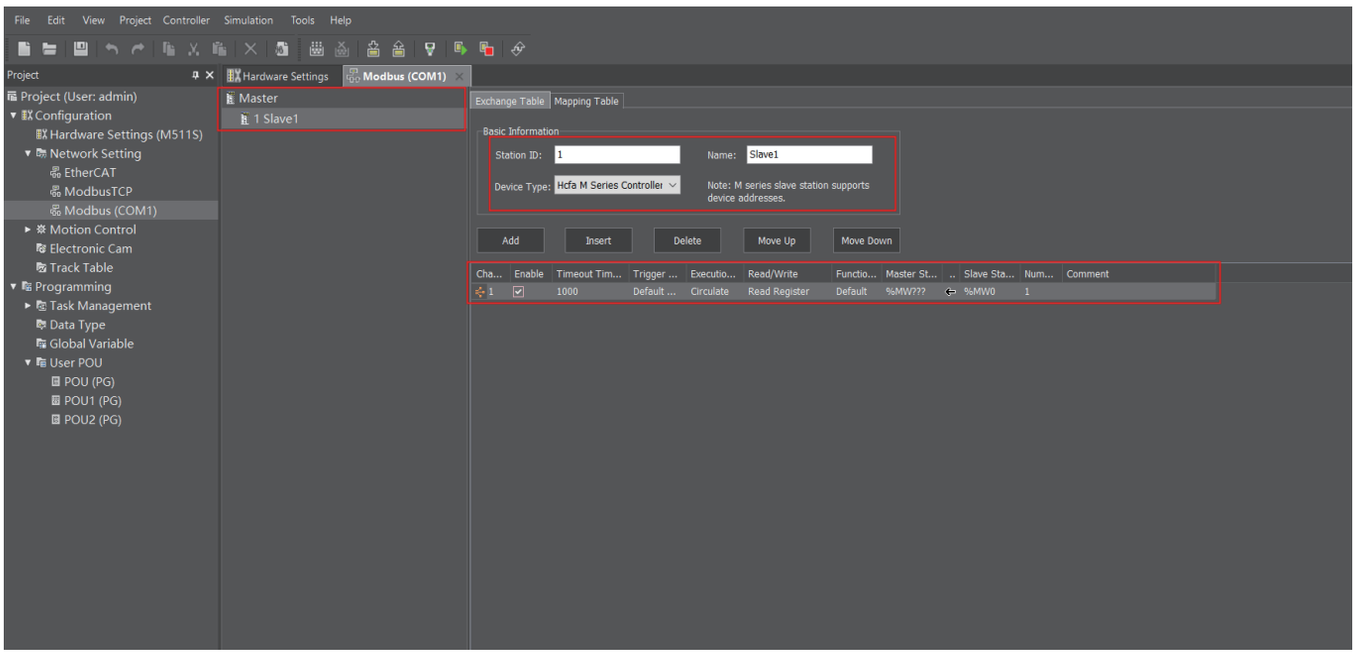
Step 2: Change the communication protocol to ensure that the baud rate and protocol of the master and slave are the same.



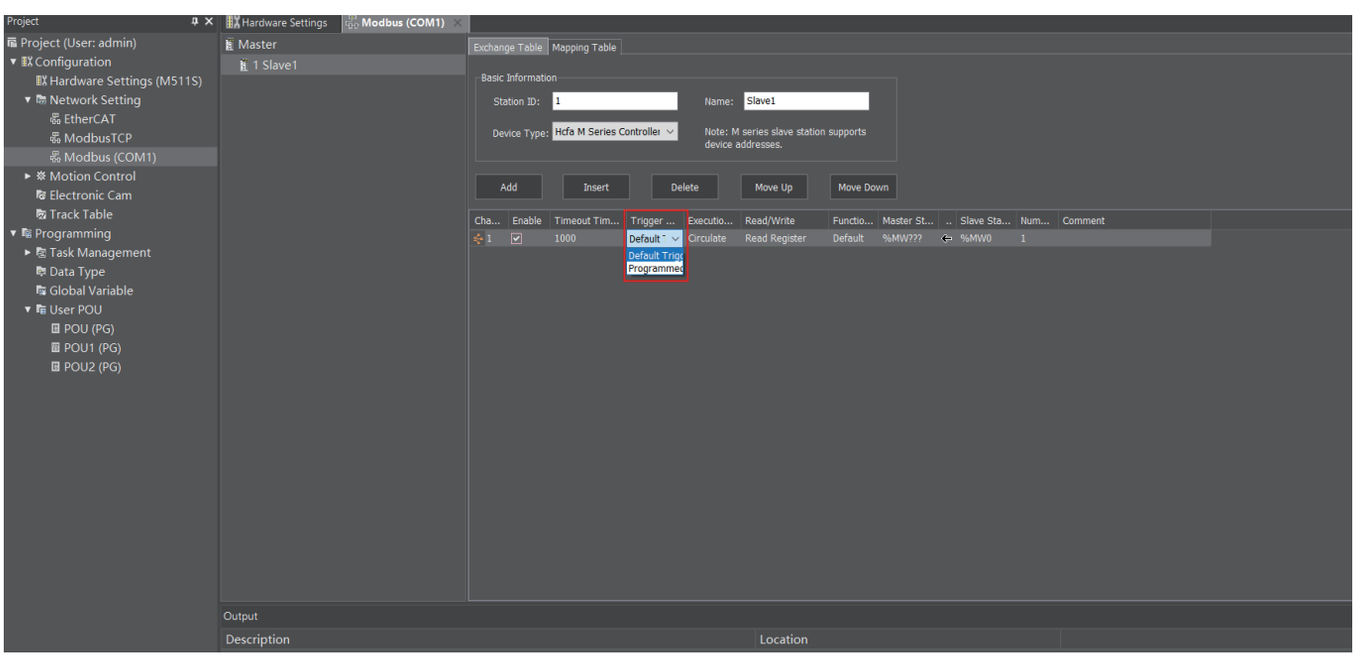
Step 3: Operation of the Modbus master function master switch. After turning it on, Modbus-related functions can be used. The software configuration can be enabled by default (run by default). Or it can be enabled by the Modbus_MasterRun instruction (executed after the instruction is triggered) and disabled by the Modbus_MasterStop instruction. Get the Modbus master status by Modbus_GetMasterstatus instruction.



Step 4: There are multiple data exchange channels in the Modbus master station, which are independent of each other. The channel parameters can be configured separately in the software configuration interface.



Step 5: There are multiple data exchange channels in the Modbus master station, which are independent of each other. The channels configured through software can be triggered by default (run by default) or be triggered by a program (executed after the Modbus_LinkRun instruction is triggered) to open the specified channel. Close via Modbus_LinkStop instruction, and get the status of the specified data exchange channels by the ModbusTCP_ GetLinkStatus instruction.



Instruction configuration using Modbus functionality:

Step	Item	Instruction	Description
1	Set master-slave mode	Serial_Manage	Set serial port master-slave mode
2	Control master	Modbus_MasterRun	Enable Modbus master function
		Modbus_MasterStop	Disable Modbus master function
		Modbus_GetMasterstatus	Get Modbus master status
3	Configure channels	Modbus_LinkConfig	Configure the specified data exchange channels
4	Control channels	Modbus_LinkRun	Open the specified data exchange channels
		Modbus_LinkStop	Close the specified data exchange channels
		Modbus_GetLinkStatus	Get the status of the specified data exchange channels

Step 1: Set the serial port master-slave mode of the controller, and change the serial port master-slave mode to “Master/Free Protocol” by Serial_Manage instruction.

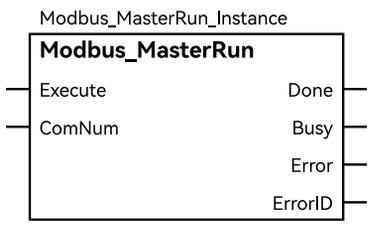
Step 2: Operation of the Modbus master function master switch. When turning the switch on, Modbus-related functions can be used. It can be enabled by Modbus_MasterRun instruction, disabled by Modbus_MasterStop instruction, and it gets Modbus master status by Modbus_GetMasterstatus instruction.

Step 3: There are multiple data exchange channels in the Modbus master station, which are independent of each other. The channel parameters are configured by the Modbus_LinkConfig instruction.

Step 4: There are multiple data exchange channels in the Modbus master station, which are independent of each other and can be enabled by the Modbus_LinkRun instruction. It can be disabled by the Modbus_LinkStop instruction, and get the status of the specified data exchange channels by the Modbus_GetLinkStatus instruction.

2.2 Modbus_MasterRun

This instruction is used to control the master to enable instructions. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Serial port Modbus management	Serial port Modbus management	FB		Modbus_MasterRun(Execute:=parameter, ComNum:=parameter, Done=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute the instruction when this parameter is TRUE
ComNum	Serial hardware interface number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

This instruction is used to enable serial Modbus cyclic data exchange. When the instruction is executed (Execute is TRUE), it can be enabled by the parameter Execute, if Execute is FALSE, the instruction is not executed.

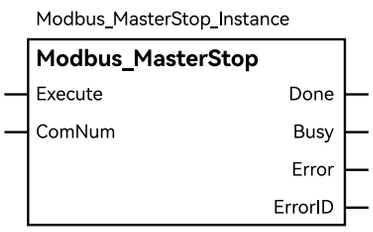
The parameters set by the Modbus_LinkConfig instruction take effect when Open changes from FALSE to TRUE, and the operation flow of Modbus periodic data exchange is as follows.

Step1 (Instruction configuration method): Modbus_LinkConfig instruction configures parameters.

Step2: The Modbus_Mange instruction is executed (Enable is TRUE), and the data exchange is enabled by the Modbus_MasterRun and Modbus_LinkRun parameters and disabled by the Modbus_MasterStop and Modbus_LinkStop parameters.

2.3 Modbus_MasterStop

Disable Modbus master function Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Modbus_MasterStop	Serial port Modbus management	FB		Modbus_MasterStop(Execute:= parameter, ComNum:= parameter, Done=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute the instruction when this parameter is TRUE
ComNum	Serial hardware interface number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

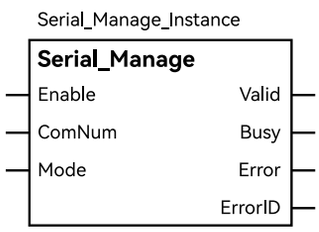
Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

Use the Modbus_MasterStop instruction during configuration. Execute is FALSE when the Modbus master is conducting periodic data exchange. When it is necessary to stop the Modbus data exchange, set Execute to TRUE to stop the data exchange between the master and the slave.

2.4 Serial_Manage

Set a specified serial port as a Modbus master or slave. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Serial_Manage	Serial port customized master-slave mode	FB		Serial_Manage_Instance(Enable:= parameter, ComNum:= parameter, Mode:= parameter, Valid=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	Execute when it is TRUE; Not to execute when it is FALSE
ComNum	Serial hardware interface number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.
Mode	Set the serial port master-slave mode	INT	0~1	0	Set the serial port to master mode or slave mode. 0: Serial port is slave mode 1: Serial port is master mode

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the instruction is executed normally
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Busy	When Enable changes from FALSE to TRUE	When Enable changes from TRUE to FALSE When Error changes from FALSE to TRUE
Valid	When Enable is TRUE	When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Enable changes from TRUE to FALSE

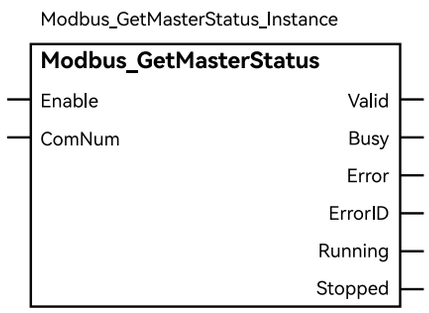
◆ Function description

This instruction is used to set a specified serial port as a Modbus master or slave. When the instruction is executed (Enable is TRUE), the master and slave modes are controlled by the parameter Mode (0: the serial port is set as the slave, 1: the seri-

al port is set as the master). If Enable is FALSE, the instruction will not be executed, and the parameter Mode is invalid at this time.

2.5 Modbus_GetMasterStatus

Get master status. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Modbus_Get-MasterStatus	Get Modbus master status	FB		Modbus_GetMasterStatus _Instance(Enable:= parameter, ComNum:= parameter, Valid=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter, Running=>parameter, Stopped=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	Execute when it is TRUE; Not to execute when it is FALSE
ComNum	Serial hard-ware inter-face number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	Output valid parameters	BOOL	TRUE / FALSE	TRUE when the instruction is executed normally
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Running	Instruction is running	BOOL	TRUE / FALSE	TRUE when master communication is running
Stopped	Instruction is stopped	BOOL	TRUE / FALSE	TRUE when master communication is stopped

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Busy	When Enable changes from FALSE to TRUE	When Enable changes from TRUE to FALSE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Enable changes from TRUE to FALSE
Running	When Enable is TRUE	When Enable changes from TRUE to FALSE

Stopped	When Enable changes from TRUE to FALSE When Error changes from FALSE to TRUE	When Enable is TRUE

◆ Function description

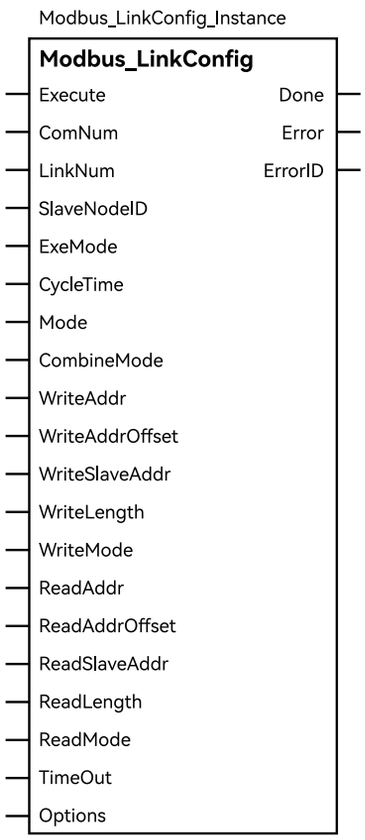
This instruction is used to get the status of the serial Modbus periodic data exchange.

If an error occurs during data exchange, the output variables Error, and ErrorID will output the corresponding error status and error code.

Note: The output variables Error and ErrorID indicate whether the instruction is executed incorrectly, which is independent of whether the data exchange reports an error.

2.6 Modbus_LinkConfig

Configure the parameters required for the serial port data exchange as a Modbus master. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Modbus_Link-Config	Parameter configuration for Modbus data exchange channels	FB		Modbus_LinkConfig (Execute:=parameter, ComNum:=parameter, LinkNum:= parameter, SlaveNodeID:= parameter, ExeMode:= parameter, CycleTime:= parameter, Mode:= parameter CombineMode:= parameter WriteAddr:= parameter, WriteAddrOffset:=parameter, WriteSlaveAddr:= parameter WriteLength:= parameter, WriteMode:= parameter, ReadAddr:= parameter, ReadAddrOffset:=parameter, ReadSlaveAddr:=parameter, ReadLength:= parameter, ReadMode:= parameter TimeOut:= parameter, Options:= parameter, Done=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	The output variable is valid	BOOL	TRUE / FALSE	FALSE	TRUE when the output variable is valid
ComNum	Serial hardware interface number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.
LinkNum	Data exchange channel number	UINT	1~24	0	Set the numbers of the serial Modbus data exchange channels. Up to 24 different data exchange channels can be supported at the same time and are distinguished by LinkNum.
SlaveNodeID	Modbus station number of the target device	USINT	0~255	0	Set the Modbus station number of the target device
ExeMode	Execution mode	USINT	0~1	0	Set the mode of execution 0: Cyclic sending 1: Single sending

CycleTime	Cycle time	USINT	0~65535	10	Set the interval between the last data sending and the next data sending in the cyclic sending mode. Unit: ms
Mode	Type of reading/ writing address	USINT	0~1	0	Set the reading/writing type of the slave address, and select reading/writing Word-type address or Bit-type address by this parameter. 0 : Word 1 : Bit
Comebine-Mode	Read/write integration mode selection	BOOL	TRUE / FALSE	FALSE	This parameter is used to set whether to enable the read/write integration function. 0: Disabled (read and write data separately) 1: Enable (use 0x17 function code to integrate read and write data into one message) Note: Valid only when Mode is 0
WriteAddr	Starting address of the write operation data cache	UINT	%MW0~%MW32767 %QW0~%QW63	Required field	Set the storage address in the controller for the data to be written to the target. For write operations, the controller writes the contents of this address (which is determined by both WriteAddr and Write-AddrOffset) to the target device. If the input corresponding to this parameter is a variable, the correct address must be specified when the variable is declared.
WriteAddrOff-set	Offset value of the write operation data cache	USINT	0~255	0	Set the offset value of the write operation data cache. The actual content to be written is obtained from the address based on WriteAdd offset by WriteAddrOffset. If it is a Word read/write, the offset unit is 1 Word. For example, if WriteAddr specifies the address as %MW0 and WriteAddrOffset is 1, it means the starting address is %MW1. If it is a Bit read/write, the offset unit is 1Bit. For example, if WriteAddr specifies the address as %MW1 and WriteAddrOffset is 1, it means the starting address is %MX2.1. 1, it means the starting address is %MX2.1.
WriteSlaveAd-dr	Destination address for write operation	UINT	16#0~16#FFFF	0	Set the starting address of the write operation, which is the Modbus address in the target device.
Writelength	Write operation length	UINT	Word device: 0~100 Bit device: 0~256	0	Set the length of the written data. If it is a Word read/write, the unit is Word; if it is a Bit read/write, the unit is Bit.
WriteMode	Write operation mode (function code)	USINT	16#0、16#06、16#10、16#05、16#0F	0	Set the function code to be used for write operation. If set to 0, the controller will select it automatically.
ReadAddr	Starting address of the read operation data cache	UINT	%MW0~%MW32767 %QW0~%QW63	Required field	Set the storage starting address of the data read from the target device, which is determined by both ReadAddr and ReadAddrOffset. If the input corresponding to this parameter is a variable, the correct address must be specified when the variable is declared.

ReadAddrOffset	Offset value of the read operation data cache	USINT	0~255	0	Set the offset value of the read operation data cache. The read content will be stored in the address based on ReadAddr and offset by ReadAddrOffset. If it is a Word read/write, the offset unit is 1 Word. For example, ReadAddr specifies the address as %MW100, and ReadAddrOffset is 1, which means the starting address is %MW101. If it is a Bit read/write, the offset unit is 1Bit. For example, ReadAddr specifies the address as %MW100, and ReadAddrOffset is 1, which means the starting address is %MX200.1.
ReadSlaveAddr	Destination address for read operation	UINT	16#0~16#FFFF	0	Set the starting address of the read operation, which is the Modbus address in the target device.
Readlength	Read operation length	UINT	Word device: 0~100 Bit device: 0~256	0	Set the length of the read data. If it is a Word read/write, the unit is Word; if it is a Bit read/write, the unit is Bit.
ReadMode	Read operation mode (function code)	USINT	16#0、16#03、16#04、16#01、16#02	0	Set the function code to be used for read operation. If set to 0, the controller will select it automatically.
TimeOut	Communication timeout	UINT	0~65535	0	Set the time to wait for a response from the target device. Unit: ms. Ms
Options	Reserved	Option_Modbus_Link	Reserved		Reserved

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	BOOL	0~65535	Output an error code when an instruction execution exception occurs. *1

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

• Basic function description

This instruction is used to configure the parameters related to Modbus data exchange, such as the IP address of the slave device, the station number of the slave device, the destination address for data reading and writing, the cache address, the data length, and the function code.

Note: This instruction is for parameter configuration only. For the configuration parameters to take effect at any time, the Modbus_MasterRun and Modbus_LinkRun instructions are required to start the data exchange.

• ComNum

This parameter is used to specify the serial port hardware interface number. Based on the hardware interface used, the corresponding number is set by this parameter. The serial port hardware interface numbers start with 1 and 2, and the exten-

sion port hardware interface numbers of the expansion card start with 11 and 12.

- **LinkNum**

There are several Modbus data exchange channels built into the controller, which are independent of each other and can be configured separately. The LinkNum parameter of this instruction is used to specify which channel is to be configured.

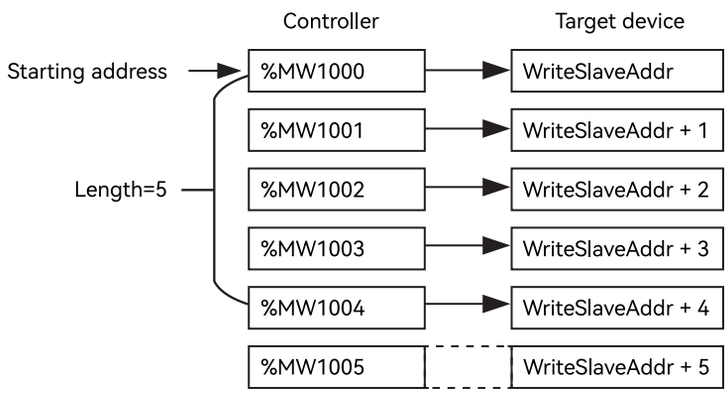
- **WriteAddr、WriteAddrOffset、WriteLength**

The data exchange configuration consists of write operation and read operation. Write operation is to write the data of specified length from the write operation cache address to the slave device, the write operation cache address is specified by the parameters WriteAddr, WriteAddrOffset, and the data length of the write operation is specified by WriteLength.

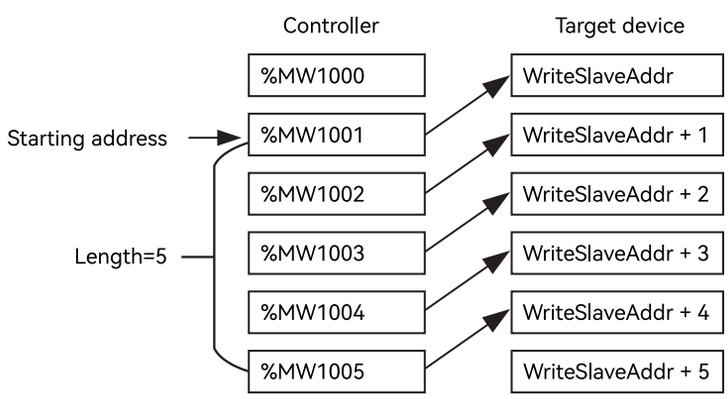
The parameters WriteAddr, WriteAddrOffset specify the starting address of the cache area, which is in the form of "base address+ offset". The base address is WriteAddr and the offset is WriteAddrOffset.

For Word read/write (Mode=0), the unit of WriteAddrOffset is Word, e.g., WriteAddr specifies the base address as %MW1000, and WriteLength specifies the length as 5:

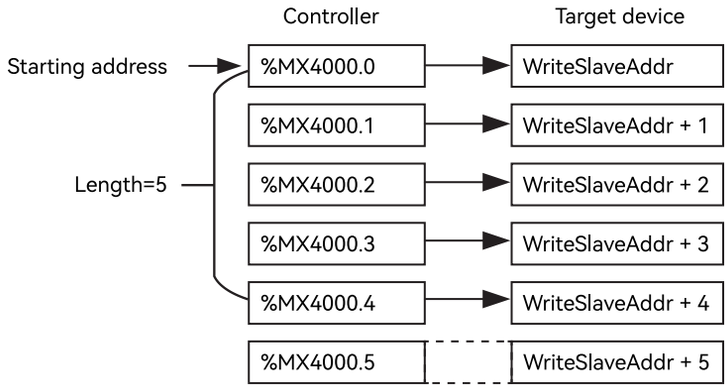
If WriteAddrOffset is 0, the cache starting address is %MW1000, as shown below:



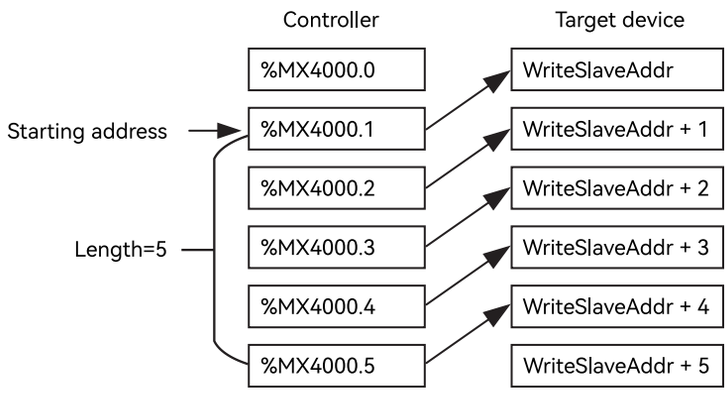
If WriteAddrOffset is 1, the cache starting address is %MW1001, as shown below:



For Bit read/write (Mode=1), the unit of WriteAddrOffset is Bit, e.g., the base address specified by WriteAddr is %MW2000 (%MX4000.0), and the length specified by WriteLength is 5. If WriteAddrOffset is 0, the cache starting address is %MX4000.0, as shown in the following figure.



If WriteAddrOffset is 1, the cache starting address is %MX4000.1 as shown below:

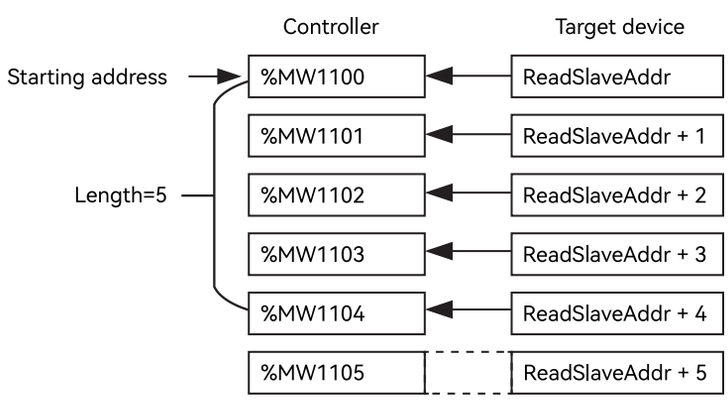


- **ReadAddr、ReadAddrOffset、ReadLength**

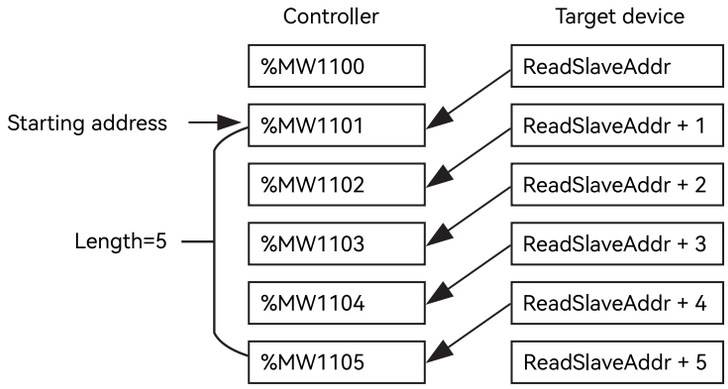
The data exchange configuration consists of Write operation and Read operation. Read operation reads data of a specified length starting at a specified address in the slave device to the controller and stores that data in a specified cache address. The read operation cache address is specified by the parameters ReadAddr and ReadAddrOffset, and the data length for the read operation is specified by ReadLength.

The parameters ReadAddr, and ReadAddrOffset specify the starting address of the cache area. The address is specified in the form of base address + offset, where the base address is ReadAddr and the offset is ReadAddrOffset.

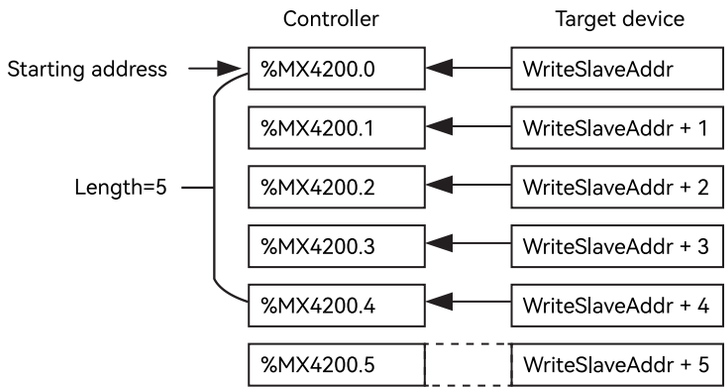
For Word read/write (Mode=0), the unit of ReadAddrOffset is Word. For example, ReadAddr specifies the base address as %MW1100 and ReadLength specifies the length as 5: If ReadAddrOffset is 0, the cache starting address is %MW1100, as shown below:



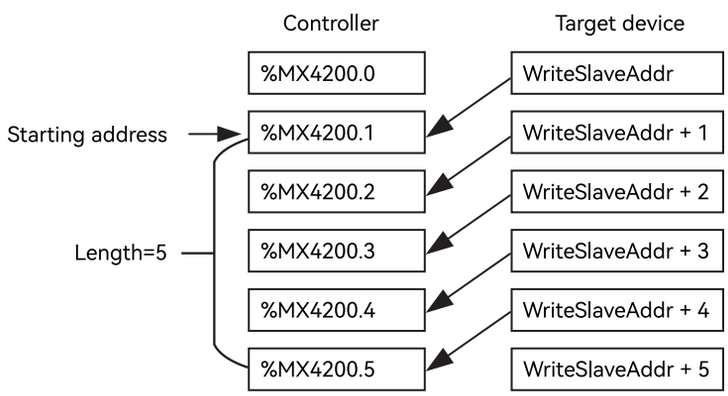
If ReadAddrOffset is 1, the cache starting address is %MW1101, as shown below:



For Bit read/write (Mode=1), the unit of ReadAddrOffset is Bit, e.g. ReadAddr specifies the base address as %MW2100 (%MX4200.0), and ReadLength specifies the length as 5: If ReadAddrOffset is 0, the cache starting address is %MX4200.0, as shown below:

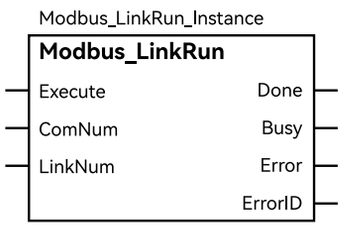


If ReadAddrOffset is 1, the cache starting address is %MX4200.1, as shown below:



2.7 Modbus_LinkRun

Enable the data exchange channels specified by serial Modbus. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Modbus_Link-Run	Serial port slave link management	FB		Modbus_LinkRun1(Execute:= parameter, ComNum:= parameter, LinkNum:= parameter, Done=>parameter, Busy=> parameter, Error=> parameter, ErrorID=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute the instruction when this parameter is TRUE
ComNum	Serial hardware interface number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.
LinkNum	Data exchange channel number	USINT	1-24	Required field	Set the numbers of the serial Modbus data exchange channels. Up to 24 different data exchange channels can be supported at the same time and are distinguished by LinkNum.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Effective	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

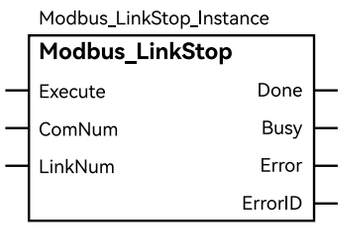
Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

The instruction is used to enable the data exchange channels specified by serial Modbus.

2.8 Modbus_LinkStop

Disable the data exchange channels specified by serial Modbus. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Modbus_Link-Stop	Serial port slave link management	FB		Modbus_LinkStop(Execute:= parameter, ComNum:= parameter, LinkNum:= parameter, Done=>parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute the instruction when this parameter is TRUE
ComNum	Serial hardware interface number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.
LinkNum	Data exchange channel number	USINT	1-24	Required field	Set the numbers of the serial Modbus data exchange channels. Up to 24 different data exchange channels can be supported at the same time and are distinguished by LinkNum.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Effective	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

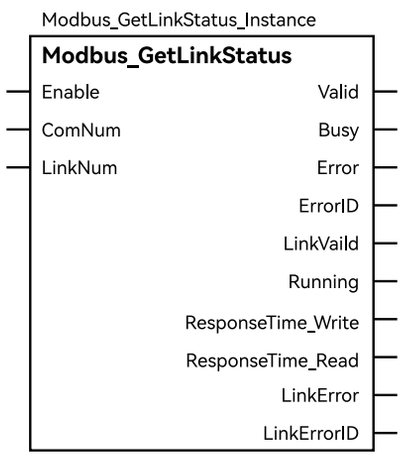
Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

The instruction is used to disable the data exchange channels specified by serial Modbus, that is, to stop the data exchange between the master and slave of the specified channels.

2.9 Modbus_GetLinkStatus

Get Modbus status. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Modbus_ GetLinkStatus	Get Modbus slave status	FB		Modbus_GetLinkStatus(Enable:=parameter , ComNum:=parameter , LinkNum:= parameter, Valid=> parameter, Busy=>parameter, Error=>parameter, ErrorID=>parameter, LinkVaild=> parameter, Running=>parameter, ResponseTime_Write=>parameter, ResponseTime_Read=>parameter, LinkError=>parameter, LinkErrorID=>parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	Execute when it is TRUE; Not to execute when it is FALSE
ComNum	Serial hardware interface number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.
LinkNum	Data exchange channel number	UINT	1~24	Required field	Set the numbers of the serial Modbus data exchange channels. Up to 24 different data exchange channels can be supported at the same time and are distinguished by LinkNum.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE/FALSE	TRUE when the instruction is executed normally
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Error	Error	BOOL	TRUE / FALSE	TRUE when an exception or parameter error occurs in the execution of the instruction
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction is executed abnormally or when a parameter error occurs. *1
LinkVaild	Valid channels	BOOL	TRUE / FALSE	Configuration of the channels specified by LinkNum is completed.
Running	Instruction is running	BOOL	TRUE/FALSE	TRUE when the specified channel data exchange is normal and FALSE when it is abnormal.
ResponseTime_ Write	Response time of write operation	TIME	0~65535	The time interval from when the master sends a write instruction to when the master receives the slave's response. Unit: ms
ResponseTime_ Read	Response time of read operation	TIME	0~65535	The time interval from when the master sends a read instruction to when the master receives the slave's response. Unit: ms

LinkError	Channel configuration error	BOOL	TRUE/FALSE	TRUE when the specified channel data exchange is abnormal and FALSE when it is normal. This bit changes to TRUE if the slave responds with abnormal data or timeout, and automatically changes to FALSE if the slave responds normally.
LinkErrorID	Channel configuration error code	WORD	0~65535	Refer to "instruction error code description" for the meaning of the output error code value when the specified channel data exchange exception occurs, i.e., when LinkError is TRUE.

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Busy	When Enable changes from FALSE to TRUE	When Enable changes from TRUE to FALSE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Enable changes from TRUE to FALSE When Error changes from FALSE to TRUE
LinkVaild	When Enable is TRUE, Error does not change to TRUE, and it is TRUE when the channel is opened normally	FALSE when the configuration of the LinkNum specified channels is invalid.
Running	When Enable is TRUE, the instruction Run Error does not change to TRUE	FALSE when the specified channel data exchange is abnormal
LinkError	Enable is TRUE, and TRUE when the channel status is abnormal	FALSE when the specified channel data exchange is normal

◆ Function description

The Modbus_GetLinkStatus instruction is used to detect the status of the configuration channel and the response time for data reading and writing.

The Enable instruction enables the parameters. ComNum is the serial port number used in the project. Linknum is the configured channel number. ResponseTime_Write is the time for the master to write data to the slave and other slave replies. ResponseTime_Read is the time for the master to read data to the slave and other slave responses. They are mainly used to detect whether the software configuration channel is opened normally, together with Modbus_GetMsterStatus instruction to detect the communication status of the master station.

2.10 Serial communication example

2.10.1 Serial data exchange example 1: Software configuration for Modbus data exchange

- **Target demand**

The two M-Series PLCs exchange data via the 0x10 function code and 0x03 function code of the serial protocol.

The controller program is as follows:

Item	Master controller	Item	Slave controller
Communication protocol	115200,(8,N,2),RTU	Communication protocol	115200,(8,N,2),RTU
Master-Slave mode	Master/Free Protocol	Master-Slave mode	Slave
Address	MW200	Address	MW300
Address	MW500	Address	MW400

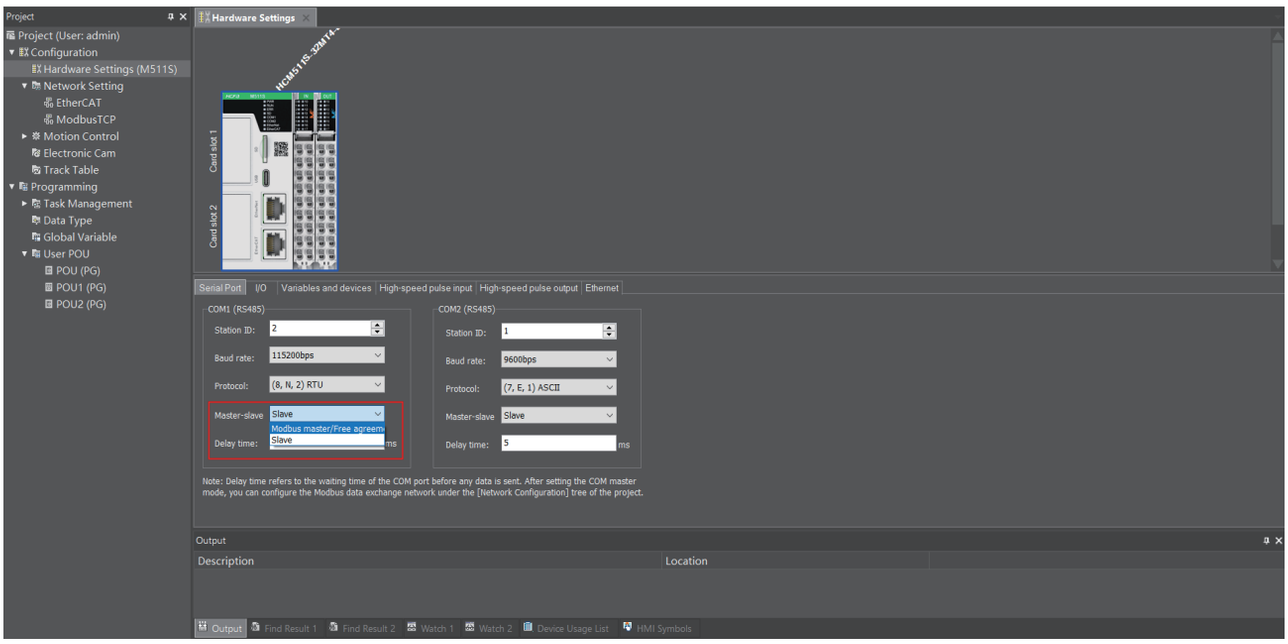
- **Demand analysis**

According to the demand, it is necessary for the master controller to use the 0x10 function code of channel 1 to send data to the slave controller and use the 0x03 function code to read data from the slave controller.

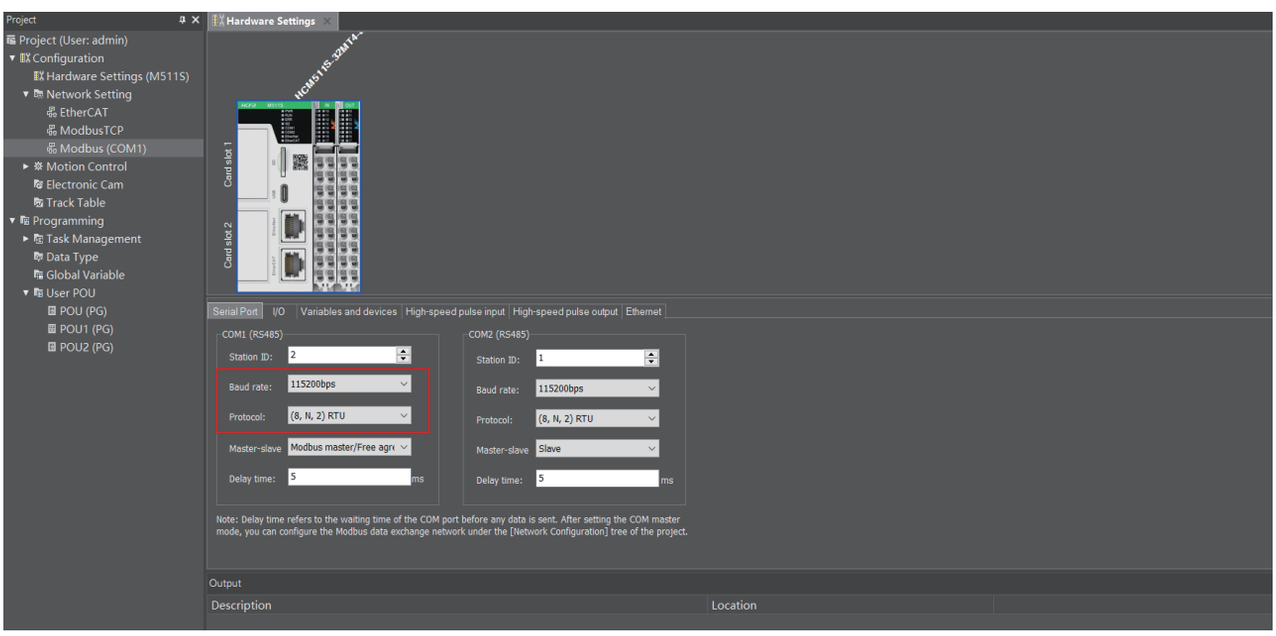
Step	Item	Usage		Description
1	Set serial port master-slave mode	Change the serial port master-slave mode to master/free protocol in the hardware configuration interface	Default slave	Set serial port master-slave mode
2	Set communication protocol	Software configuration interface	Default (7,E,1) ASCII	Set communication protocol
3	Control master	Select “enable by default” in the software configuration interface	Enable by default	Enable Modbus master function
		Modbus_MasterStop		Disable Modbus master function
		Modbus_GetMasterstatus		Get Modbus master status
4	Configure channels	Software configuration interface		Configure the specified data exchange channels
5	Control channels	Select “trigger by default” in the software configuration interface	Enable by default	Open the specified data exchange channels
		Modbus_LinkStop		Close the specified data exchange channels
		Modbus_GetLinkStatus		Get the status of the specified data exchange channels

- **Software configuration**

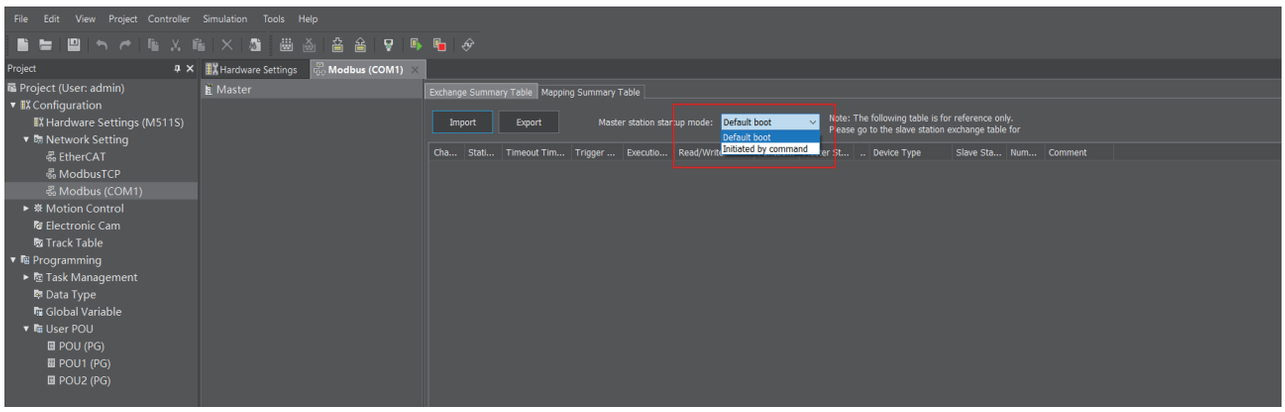
Step 1: Set the serial port master-slave mode of the controller to Master/Free Protocol in the hardware settings - serial port.



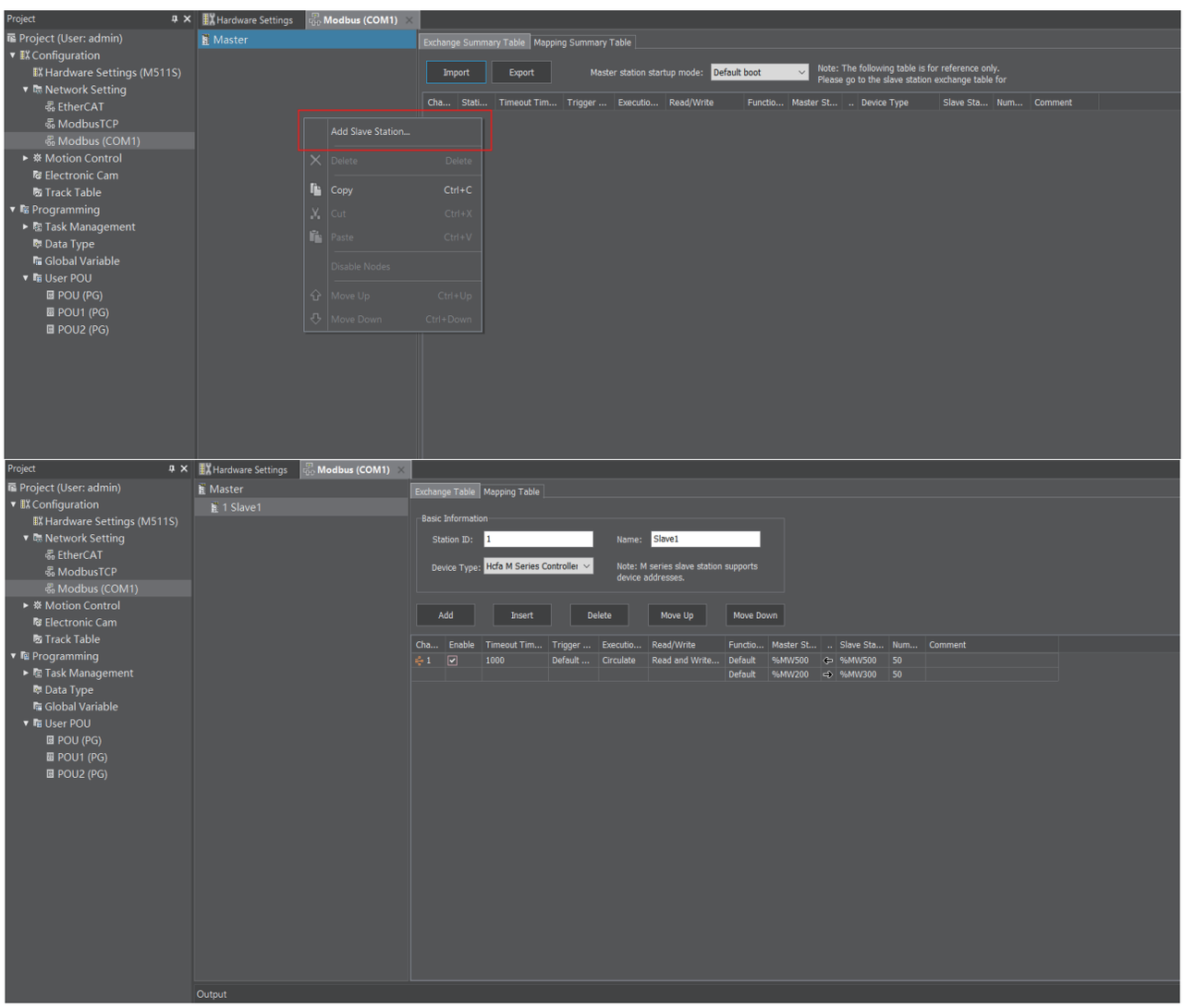
Step 2: To change the communication protocol, it is necessary to ensure that the baud rate and protocol of the master and slave are the same.



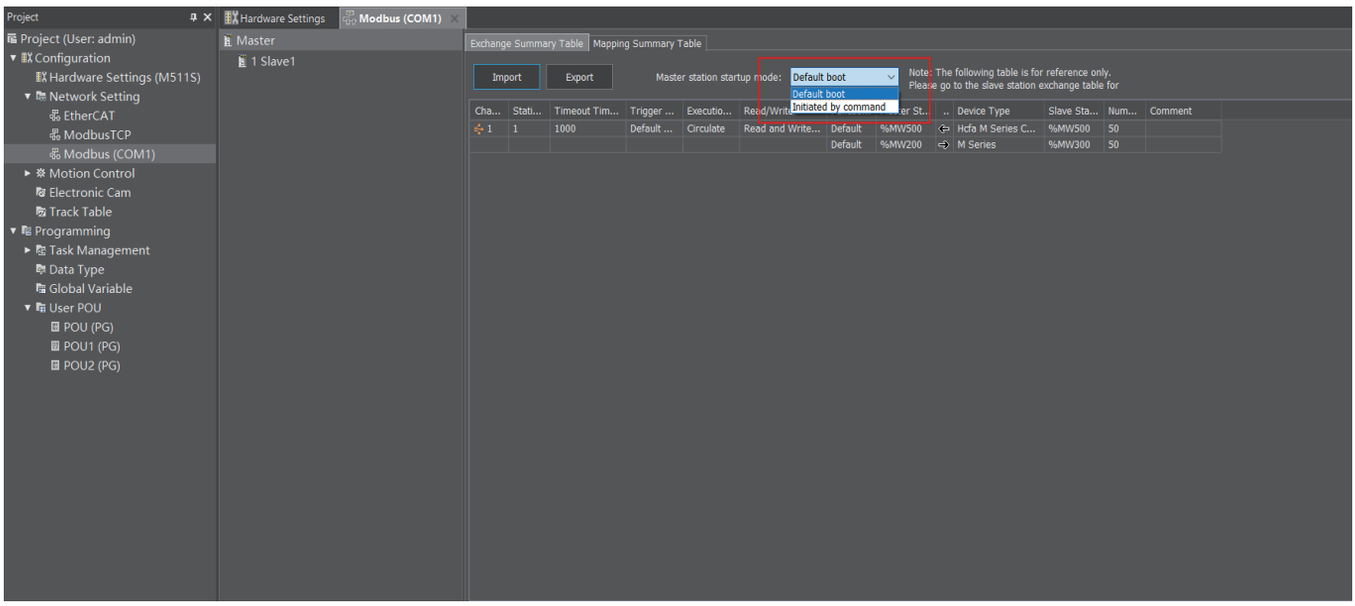
Step 3: Set the Modbus master startup mode to enable by default, and the Modbus master function will be enabled automatically after the download.



Step 4: Add a slave and set the data parameters for channel 1



Step 5: Set the trigger mode as trigger by default, after downloading, the data of channel 1 will be enabled automatically to start data exchange.



2.10.2 Modbus data exchange example 2: Software and instruction configuration for Modbus data exchange

- **Target demand**

The two M-Series PLCs exchange data via the 0x10 function code and 0x03 function code of the serial protocol.

The IP address and port of the controller are as follows:

Item	Master controller	Item	Slave controller
Communication protocol	115200,(8,N,2),RTU	Communication protocol	115200,(8,N,2),RTU
Master-Slave mode	Master/Free Protocol	Master-Slave mode	Slave
Address	MW200	Address	MW300
Address	MW500	Address	MW400

- **Demand analysis**

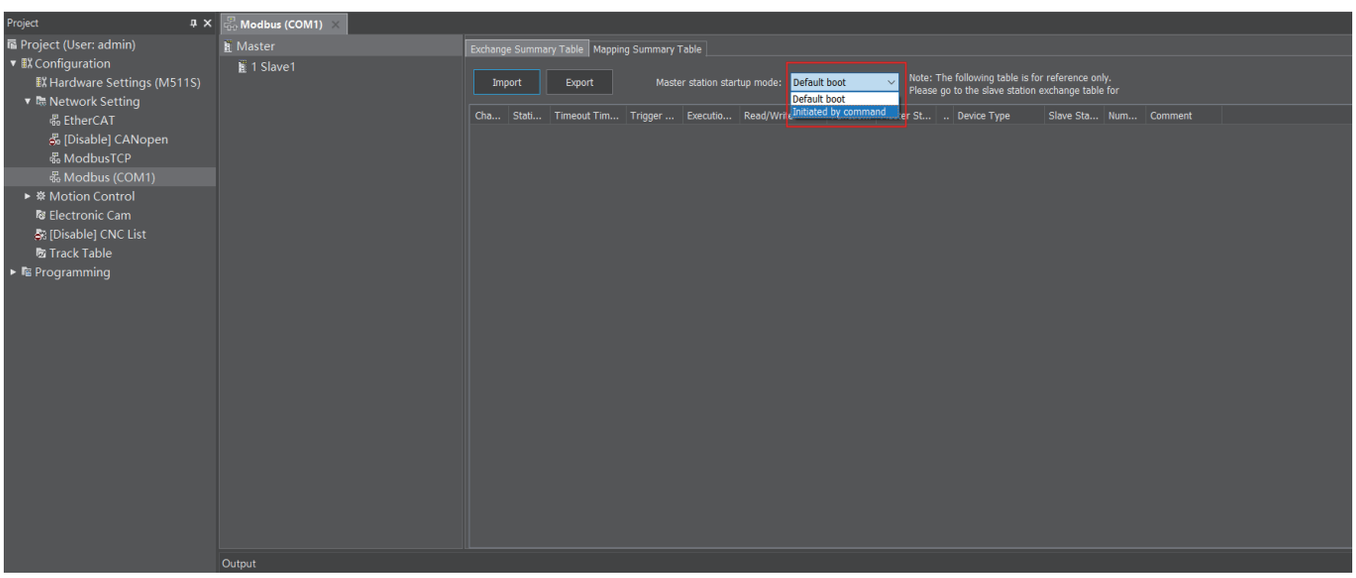
According to the demand, it is necessary for the master controller to use the 0x10 function code of channel 1 to send data to the slave controller and use the 0x03 function code to read data from the slave controller.

Step	Item	Usage		Description
1	Set serial port master-slave mode	Enable by instructions	Serial_Manage	Set serial port master-slave mode
		Change the serial port master-slave mode to master/free protocol in the hardware configuration interface	Default slave	
2	Set communication protocol	Software configuration interface	Default (7,E,1) ASCII	Set communication protocol

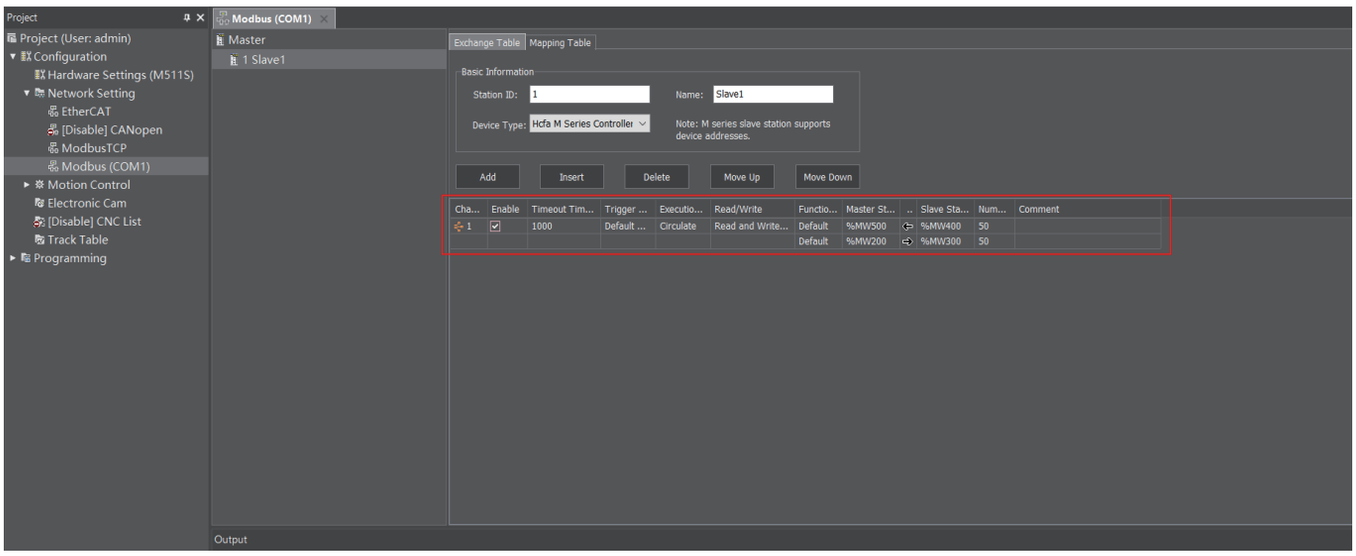
3	Control master	Select “enable by instructions” in the software configuration interface	Modbus_MasterRun	Enable Modbus master function
		Select “enable by default” in the software configuration interface	Enable by default	Enable Modbus master function
		Modbus_MasterStop		Disable Modbus master function
		Modbus_Masterstatus		Get Modbus master status
4	Configure channels	Software configuration interface		Configure the specified data exchange channels
5	Control channels	Select “trigger by program” in the software configuration interface	Modbus_LinkRun	Open the specified data exchange channels
		Select “trigger by default” in the software configuration interface	Enable by default	Open the specified data exchange channels
		Modbus_LinkStop		Close the specified data exchange channels
		Modbus_GetLinkStatus		Get the status of the specified data exchange channels

- **Software configuration**

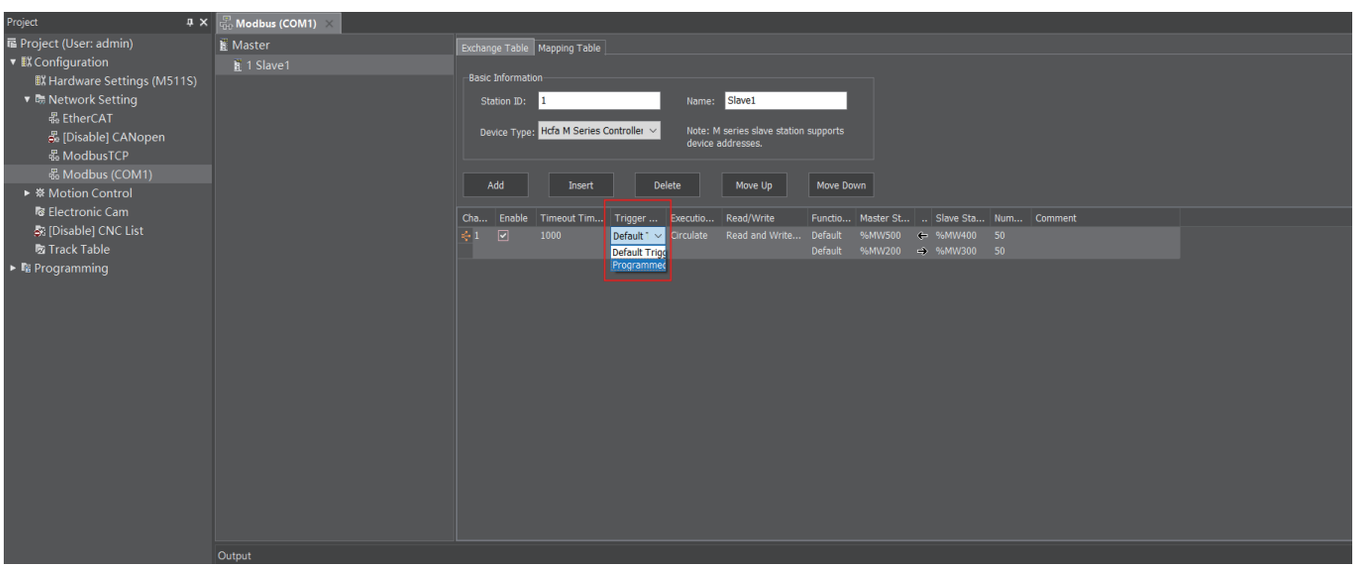
Step 1: Set the ModbusTCP master startup method to enable by instructions, and the ModbusTCP master function will be enabled automatically after the download.



Step 2: Add a slave and set the data parameters for channel 1.



Step 3: The trigger mode is set to trigger by program.



Category	Name	Assigned to	Data type	Initial value	Comment
VAR	MasterRun1Ex		BOOL		
VAR	MasterStop1EX		BOOL		
VAR	Modbus_MasterRun1		Modbus_MasterRun		
VAR	Modbus_MasterStop1		Modbus_MasterStop		
VAR	Link1RunEX		BOOL		
VAR	LinkStop1EX		BOOL		
VAR	Modbus_LinkRun1		Modbus_LinkRun		
VAR	Modbus_LinkStop1		Modbus_LinkStop		

LD

ST

```
Modbus_MasterRun1(Execute:= MasterRun1Ex,
    ComNum:=1 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
Modbus_MasterStop1(Execute:= MasterStop1EX,
    ComNum:=1 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
Modbus_LinkRun1(Execute:= Link1RunEX,
    ComNum:=1 ,
    LinkNum:=1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
Modbus_LinkStop1(Execute:=LinkStop1EX ,
    ComNum:= 1,
    LinkNum:= 1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
```

Step 1: The Modbus_MasterRun1 instruction will be triggered when the variable MasterRun1Ex is set to TRUE to enable the Modbus master function.

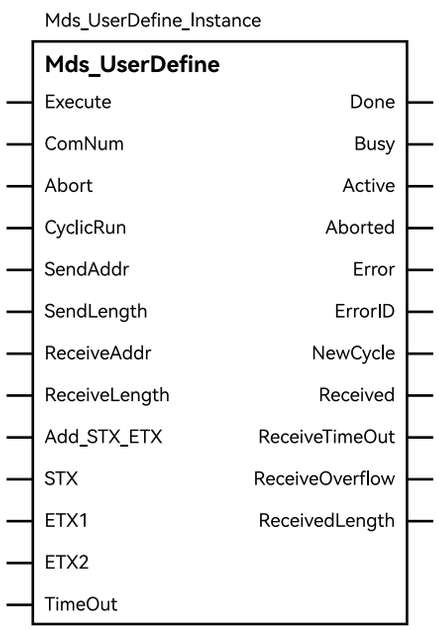
Step 2: The variable LinkRun1EX when set to TRUE triggers the Modbus_LinkRun1 instruction to enable the Modbus slave function.

Step 3: The variable LinkConfigEX is set to TRUE to trigger the Modbus_LinkConfig1 instruction to write the configured data parameters to channel 1.

Step 4: To stop data exchange, set the variable LinkStop1EX to TRUE to trigger the Modbus_LinkStop1 instruction to disable data exchange of channel 1. Alternatively, set the variable MasterStop1EX to TRUE to trigger the Modbus_MasterStop1 instruction to disable the Modbus function. Use the Modbus_LinkStop instruction to disable the data exchange of the specified channel, and other channels can still exchange data if the user configures other channels. If the Modbus function is disabled by the Modbus_MasterStop instruction, all data channel exchanges will be stopped.

2.11 Mds_UserDefine

Control the serial port to send and receive data by customized protocols. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
Mds_UserDefine	Control the serial port to send and receive data by customized protocols	FB		Mds_UserDefine_Instance(Execute:= parameter, ComNum:=parameter, Abort:= parameter, CyclicRun:= parameter, SendAddr:= parameter, SendLength:= parameter, ReceiveAddr:= parameter, ReceiveLength:= parameter, Add_STX_ETX:= parameter, STX:= parameter, ETX1:= parameter, ETX2:= parameter, TimeOut:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Aborted=> parameter, Error=> parameter, ErrorID=> parameter, NewCycle=> parameter, Received=> parameter, ReceiveTimeOut=> parameter, ReceiveOverflow=> parameter, ReceivedLength=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when it is TRUE; Not to execute when it is FALSE
ComNum	Serial hardware interface number	UINT	1~255	Required field	This parameter specifies the serial port hardware interface number to be used. The serial port hardware interface numbers start with 1 and 2. The expansion serial port hardware interface numbers of the expansion card start with 11 and 12.
Abort	Abort data sending and receiving	BOOL	TRUE / FALSE	FALSE	Abort when the rising edge of this parameter is detected
CyclicRun	Cyclic sending and receiving	BOOL	TRUE / FALSE	FALSE	Set whether to send or receive the data in the cache cyclically. TRUE: Cyclic sending and receiving FALSE: Single sending and receiving
SendAddr	Cache starting address of the data to be sent	USINT	%MB0~%MB65535	Required field	Set the starting address for storing the data to be sent
SendLength	Send data length	UINT	0~200	0	Set the length of the data to be sent (Unit: Byte)

ReceiveAddr	Cache starting address of the received data	USINT	%MB0~%MB65535	Required field	Set the starting address for storing the received data
ReceiveLength	Receive data length	UINT	0~200	0	Set the length of the data to be received (Unit: Byte)
Add_STX_ETX	Add header and footer codes	BOOL	TRUE / FALSE	FALSE	Set whether to add header and footer codes TRUE: Adding FALSE: No adding
STX	Header code	USINT	16#0~16#7F	16#3A	Set the header code to be added
ETX1	Footer code 1	USINT	16#0~16#7F	16#0D	Set the 1st footer code to be added
ETX2	Footer code 2	USINT	16#0~16#7F	16#0A	Set the 2nd footer code to be added
TimeOut	Timeout	UINT	0~32767	0	Set the timeout of data receiving. (Unit: ms)

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Execution completed (single sending only)	BOOL	TRUE / FALSE	In single sending mode (when CyclicRun is set to TRUE), it changes to TRUE when completing data sending, data receiving, and data sending and receiving.
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	The instruction is controlling the serial port	BOOL	TRUE / FALSE	TRUE when the instruction controls the serial port normally
Aborted	The instruction is aborted	BOOL	TRUE / FALSE	TRUE when the instruction is aborted
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
NewCycle	Starting of the new sending/receiving cycle	BOOL	TRUE / FALSE	A new sending/receiving cycle is about to start under the cyclic sending mode
Received	Receiving completed	BOOL	TRUE / FALSE	TRUE when data receiving is completed
ReceiveTimeOut	Receiving timeout	BOOL	TRUE / FALSE	TRUE when data receiving timeout occurs
ReceiveOverflow	Receiving data overflow	BOOL	TRUE / FALSE	TRUE when the data to be received exceeds the set length.
ReceivedLength	Actual received length	UINT	0~200	Actual received length (Unit: Byte)

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Execute changes from TRUE to FALSE
Busy	When Execute changes from FALSE to TRUE	When Done changes from FALSE to TRUE When Aborted changes from FALSE to TRUE When Error changes from FALSE to TRUE
Active	When the instruction starts controlling the serial port	When Done changes from FALSE to TRUE When Aborted changes from FALSE to TRUE When Error changes from FALSE to TRUE
Aborted	When the instruction is aborted	When Execute changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE

NewCycle	When the previous sending/receiving cycle ends under the cyclic sending mode	When Aborted changes from FALSE to TRUE When Error changes from FALSE to TRUE When a new sending / receiving cycle is about to start under the cyclic sending mode
Received	When data is received	When Aborted changes from FALSE to TRUE When Error changes from FALSE to TRUE When a new sending / receiving cycle is about to start under the cyclic sending mode
ReceiveTime-Out	Receiving timeout	When Execute changes from TRUE to FALSE
ReceiveOver-flow	When the length of data to be received exceeds the set value	When the actual received length does not exceed the set length

◆ Function description

• Basic function description

This instruction is used to control the serial port to send and receive customized data.

When the instruction triggers execution, it sends data of the length specified by SendLength in the sending cache area to the corresponding device and then stores the data to be received in the receiving cache area (the portion exceeding ReceiveLength will be discarded). The starting address of the sending cache area is specified by the parameter SendAddr, and the starting address of the receiving cache area is specified by the parameter ReceiveAddr. Single sending and receiving or cyclic sending and receiving can be specified by CyclicRun.

• SendLength (length of data to be sent)

When STX is FALSE (header and footer codes are not enabled), the instruction sends the data of the length specified by SendLength (length of data to be sent) in the sending cache area to the corresponding device. When STX is TRUE (header and footer codes are enabled), the instruction sends the data of the length specified by SendLength (length of data to be sent) in the sending cache area and the data of the header code and the footer code to the corresponding device. SendLength (length of data to be sent) does not count the data of the header code and the footer code.

• ReceiveLength （length of data to be received)

ReceiveLength is the length of all data in a received frame. When STX is TRUE (header and footer codes are enabled), ReceiveLength counts the data in the header and footer codes as well.

• Single sending and receiving

When the parameter CyclicRun is FALSE, it indicates single sending, single receiving or single sending and receiving.

When it is necessary to conduct a single sending and receiving, set CyclicRun to FALSE before executing the instruction (Execute changes from FALSE to TRUE).

Single sending: When SendLength (length of data to be sent) is not 0, ReceiveLength (length of data to be received) is 0, CyclicRun is FALSE, it is a single sending. After the instruction is executed, the data length specified by SendLength (length of data to be sent) is sent, Done becomes TRUE, and the instruction no longer sends data. If the data needs to be sent again, the instruction needs to be re-executed.

Single receiving: When SendLength (length of data to be sent) is 0, ReceiveLength (length of data to be received) is not 0 and CyclicRun is FALSE, it is single receiving within the time specified by TimeOut. After receiving data, Done becomes TRUE, and this instruction no longer receives data. If no data is received within the time specified by TimeOut, the instruction will report an error. If data needs to be received again, the instruction needs to be executed again.

Single sending and receiving: When SendLength (length of data to be sent) is not 0, ReceiveLength (length of data to be received) is not 0, and CyclicRun is FALSE, it is single sending and receiving. After sending and receiving the data, Done becomes TRUE, and the instruction no longer sends or receives the data. If the data needs to be sent or received again, the instruction needs to be re-executed.

- **Cyclic sending and receiving**

When the parameter CyclicRun is TRUE, it indicates cyclic sending, cyclic receiving, or cyclic sending and receiving.

When it is necessary to conduct cyclic sending and receiving, set CyclicRun to TRUE before executing the instructions (Execute changes from FALSE to TRUE).

Cyclic sending: When SendLength (length of data to be sent) is not 0 and ReceiveLength (length of data to be received) is 0, it sends cyclically.

Cyclic receiving: When SendLength (length of data to be sent) is 0 and ReceiveLength (length of data to be received) is not 0, it receives cyclically within the time specified by TimeOut. If no data is received within the time specified by TimeOut, the instruction will report an error. If the data needs to be received again, the instruction needs to be re-executed.

Cyclic sending and receiving: When SendLength (length of data to be sent) is not 0 and ReceiveLength (length of data to be received) is not 0, it sends and receives cyclically, and conducts the next data sending and receiving after that. If no data is received within the time specified by TimeOut, the instruction will report an error. If the data needs to be received again, the instruction needs to be re-executed.

When data is received, ReceivedLength is the length of the actual data received, Received becomes TRUE for one task cycle and then automatically becomes FALSE.

When the Abort bit changes from FALSE to TRUE during cyclic sending and receiving, the cyclic sending and receiving loop is aborted, and the output variable Aborted changes to TRUE at the same time.

- **Header and footer codes**

If the parameter Add_STX_ETX is TRUE, the set header and tail codes will be added automatically when sending data. When receiving data, the header and footer codes in the actual received data are compared to the set values to see if they match. If they do not match, the data to be received will be discarded and it is considered that no data was received.

Note: When the data to be sent and the received header code and footer codes are not the same, the add header code and footer code function cannot be enabled.

- **Precautions**

This instruction will not automatically add the checksum when sending data, it can be calculated and placed in the corresponding address of the sending area according to the check rule by itself.

2.12 Customized protocol example

- **Target demand**

The M511S master COM1 port sends data (sends standard Modbus message data through the serial port customization protocol sending and receiving instructions), and the M511S master COM2 port receives data.

COM1 communication port sends standard Modbus message data	16#[01,10,15,00,00,01,02,00,08, check digit]
COM2 communication port receives data sent from COM1	16#[01,10,15,00,00,01,02,00,08, check digit]

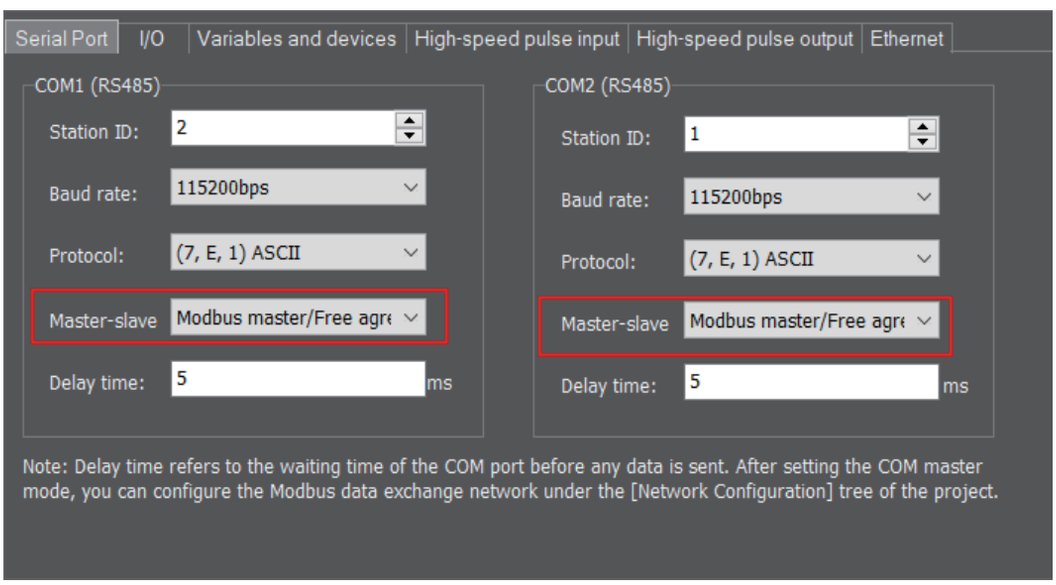
ASCII mode:

- **Demand analysis**

Depending on the demand, the header code is 16#3A and the footer code is 16#0D and 16#0A in ASCII mode. Header and footer codes are enabled in this example, and the order in which the data is requested is: 3A 30 31 31 30 31 35 30 30 30 30 30 30 31 30 32 30 30 30 38 43 46 0D 0A, and the above data are in the hexadecimal format. Before sending data, it is necessary to specify the cache starting address and the length of the data to be sent. The data except for the header code and the footer code will be written into the cache area in order, and the length is 20 (excluding the header code and the footer codes). Before receiving data, it is necessary to specify the cache starting address and the length of the data to be received, and the length is 23 (including the header and footer codes).

- **Software configuration**

When the serial port uses custom protocol-related instructions to send or receive data, it is necessary to set the corresponding serial port to "Master/Free Protocol", as shown in the red box in the following figure.



- **Program**

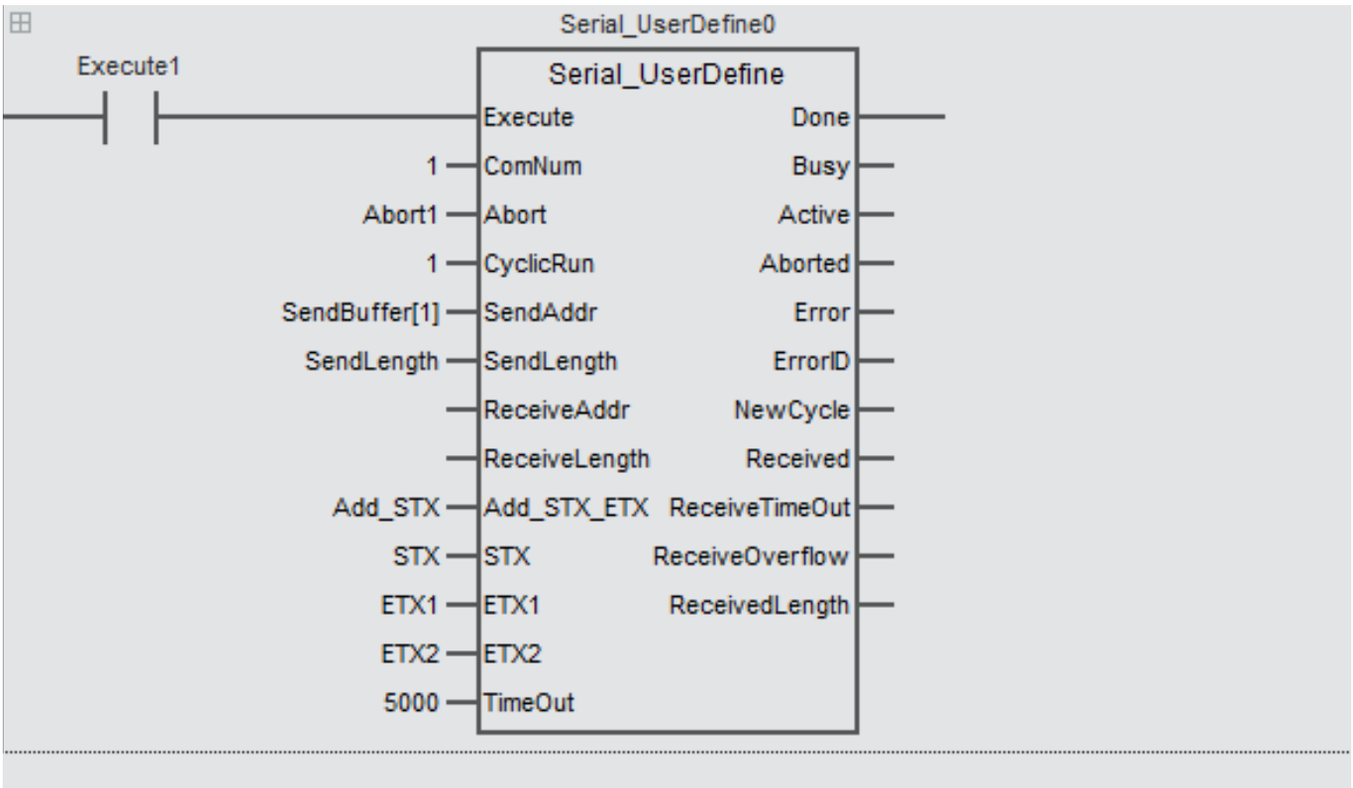
Write a program based on the demand analysis.

When sending data from the COM1 communication port, the program variables are programmed as shown in the table below:

Variable	Assigned to	Data type	Initial value	Description
Serial_UserDefine0		Serial_UserDefine		Customized protocols for sending and receiving instructions
Execute1		BOOL		Execution conditions for customized protocol sending/receiving instructions
Abort1		BOOL		Abort cyclic sending
SendBuffer	%MB0	ARRAY[1..20] OF USINT		Send data cache address
SendLength		UINT	20	Send data length
Add_STX		BOOL	TRUE	Add header and footer codes
STX		USINT	16#3A	Header code

ETX1		USINT	16#0D	Footer code 1
ETX2		USINT	16#0A	Footer code 2

The master station LD program is as follows:



The master station ST program is as follows:

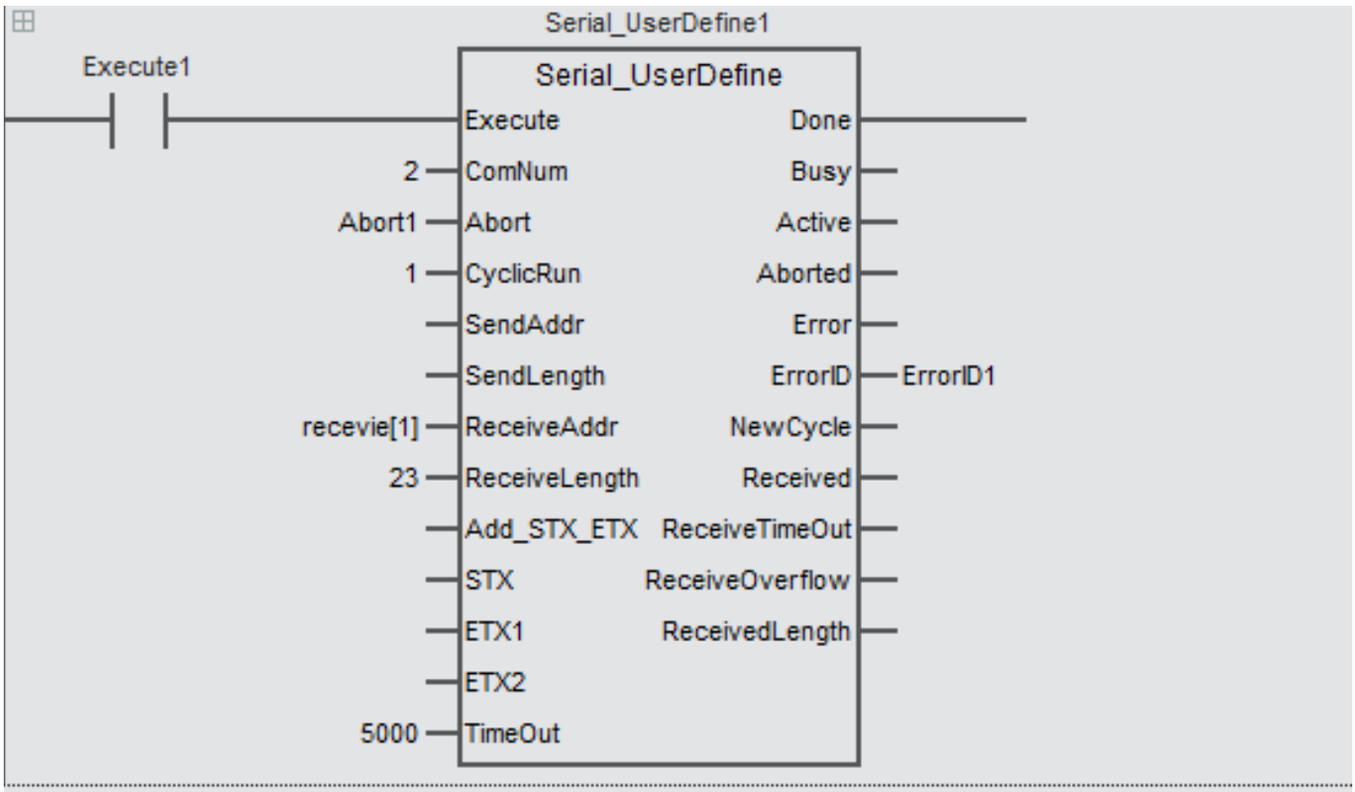
```
Serial_UserDefine0(Execute:= Execute1,
    ComNum:=1 ,
    Abort:= Abort1,
    CyclicRun:= 1,
    SendAddr:=SendBuffer[1] ,
    SendLength:= SendLength,
    ReceiveAddr:= ,
    ReceiveLength:= ,
    Add_STX_ETX:= ,
    STX:= ,
    ETX1:= ,
    ETX2:= ,
    TimeOut:=5000 ,);
```

When receiving data from the COM2 communication port, the program variables are programmed as shown in the table below:

Variable	Assigned to	Data type	Initial value	Description
Serial_UserDefine1		Serial_UserDefine		Customized protocols for sending and receiving instructions
Execute1		BOOL		Execution conditions for customized protocol sending/receiving instructions

Abort1		BOOL		Abort cyclic receiving
receive	%MB100	ARRAY[1..23] OF USINT		Receive data cache address
ErrorID1		WORD		Error code

The slave station LD program is as follows:



The slave station ST program is as follows:

```
Serial_UserDefine1(Execute:= Execute1,
    ComNum:= 2,
    Abort:= Abort1,
    CyclicRun:=1 ,
    SendAddr:= ,
    SendLength:= ,
    ReceiveAddr:= recevie[1],
    ReceiveLength:= 23,
    Add_STX_ETX:= ,
    STX:= ,
    ETX1:= ,
    ETX2:= ,
    TimeOut:= 5000,);
```

- **Program flow description:**

Step 1: Write the data to be sent to the cache area. In this example, it is necessary to write the hexadecimal numbers 30 31 31 30 31 35 30 30 30 30 30 30 31 30 32 30 30 30 30 38 43 46 to %MB0 to %MB20 in order (SendBuffer[1]~SendBuffer[20]), and the data to be received is stored to %MB100 to %MB122 (recevie [1]~ recevie[23]). Based on the sent and received data, the length of the data to be sent is 20 (excluding header code footer code) and the length of the data to be received is 23 (includ-

ing header code footer code). The array length specified by the input parameter SendBuffer (starting address of the data to be sent cache) should be equal to or greater than 20, and the array length specified by the input parameter receive (starting address of the received data cache) should be equal to or greater than 23.

Step 2: The input variable CyclicRun is set to TRUE (cyclic sending and receiving), and after the input parameter execution changes from FALSE to TRUE, the UserDefine1 and UserDefine2 instructions will be triggered. The UserDefine1 instruction appoints COM1 to send the data, the UserDefine2 instruction appoints COM2 to receive the data.

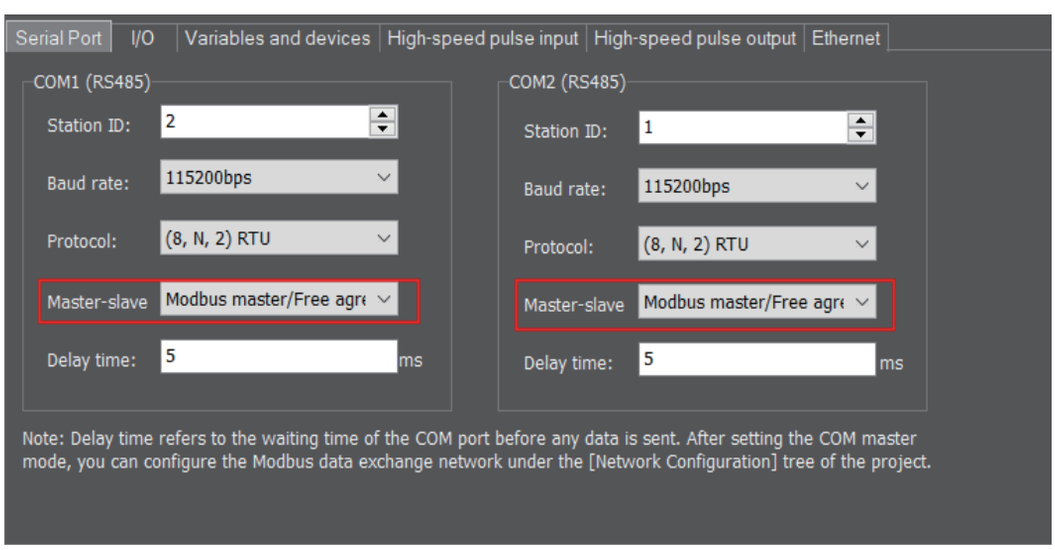
RTU mode:

- **Demand analysis**

According to the demand, the order of data requested in RTU mode is: 01 10 15 00 00 01 02 00 08 E3 57, and the above data are all in the hexadecimal format. Before sending data, it is necessary to specify the cache starting address and length of the data to be sent and write the data to be sent into the sending cache area in order, and the sending data length is 11. Before receiving data, it is necessary to specify the cache starting address and the length of the data to be received, and the receiving data length is 11.

- **Software configuration**

When the serial port uses custom protocol-related instructions to send or receive data, it is necessary to set the corresponding serial port to "Master/Free Protocol", as shown in the red box in the following figure.



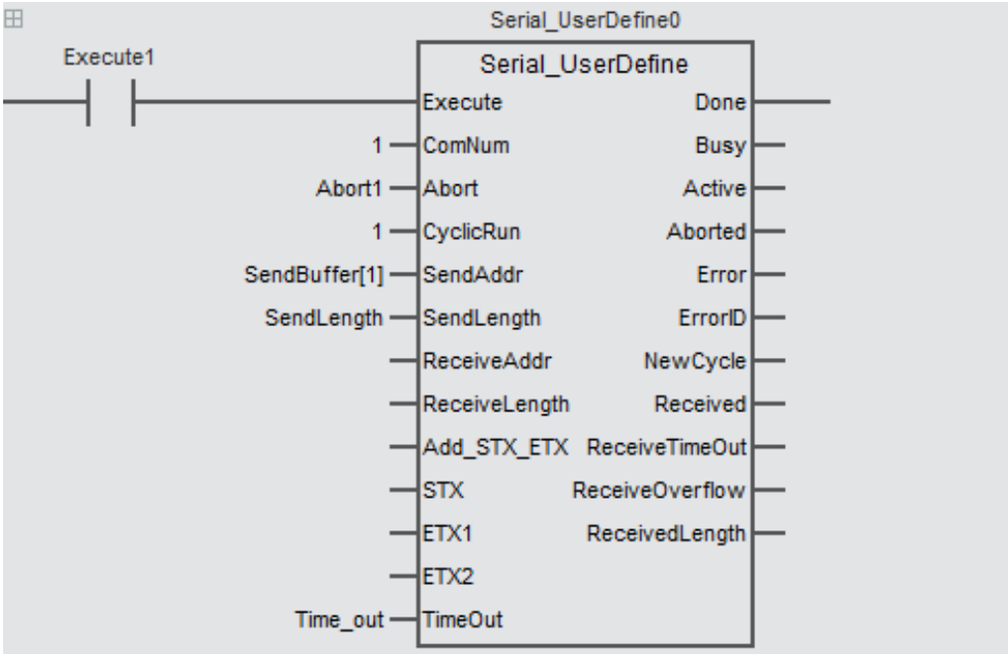
- **Program**

Write programs based on the demand analysis.

When sending data from the COM1 communication port, the program variables are programmed as shown in the table below:

Variable	Assigned to	Data type	Initial value	Description
Serial_UserDefine0		Serial_UserDefine		Customized protocols for sending and receiving instructions
Execute1		BOOL		Execution conditions for customized protocol sending/receiving instructions
Abort1		BOOL		Abort cyclic sending
sendbuffer	%MB0	ARRAY[1..11] OF USINT		Send data cache address
Time_out		UINT	5000	Set the timeout of data receiving

The master station LD program is as follows:



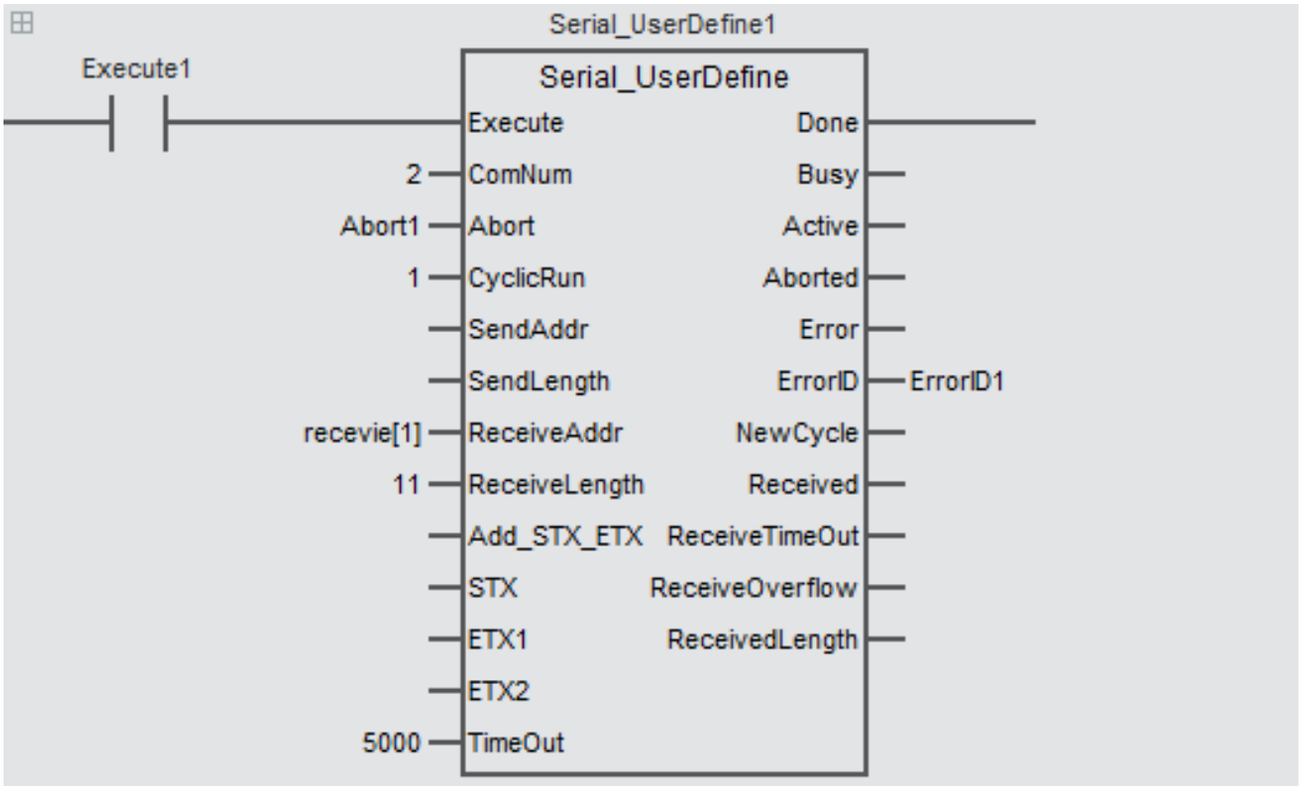
The master station ST program is as follows:

```
Serial_UserDefine0(Execute:= Execute1,
    ComNum:=1 ,
    Abort:= Abort1,
    CyclicRun:= 1,
    SendAddr:=SendBuffer[1] ,
    SendLength:= SendLength,
    ReceiveAddr:= ,
    ReceiveLength:= ,
    Add_STX_ETX:= ,
    STX:= ,
    ETX1:= ,
    ETX2:= ,
    TimeOut:=5000 ,);
```

When receiving data from the COM2 communication port, the program variables are programmed as shown in the table below:

Variable	Assigned to	Data type	Initial value	Description
Serial_UserDefine1		Serial_UserDefine		Customized protocols for sending and receiving instructions
Execute1		BOOL		Execution conditions for customized protocol sending/receiving instructions
Abort1		BOOL		Abort cyclic sending
receive	%MB100	ARRAY[1..11] OF USINT		Send data cache address

The slave station LD program is as follows:



The slave station ST program is as follows:

```
Serial_UserDefine1(Execute:= Execute1,
    ComNum:= 2,
    Abort:= Abort1,
    CyclicRun:=1 ,
    SendAddr:= ,
    SendLength:= ,
    ReceiveAddr:= recevie[1],
    ReceiveLength:= 11,
    Add_STX_ETX:= ,
    STX:= ,
    ETX1:= ,
    ETX2:= ,
    TimeOut:= 5000,);
```

- **Program flow description:**

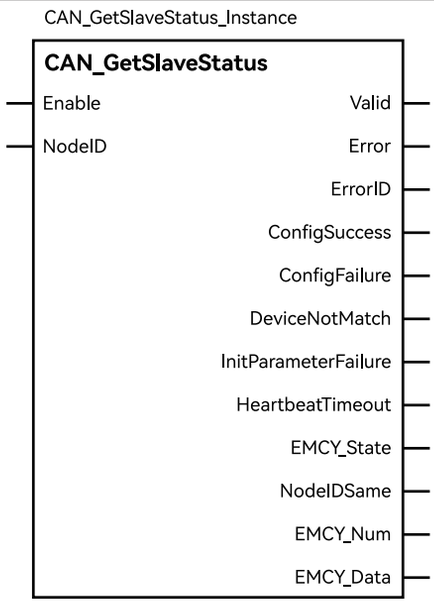
Step 1: Write the data to be sent to the sending cache area. In this example, the hexadecimal numbers 01 10 15 00 00 01 02 00 08 E3 57 need to be written to %MB0 to %MB10 in order (SendBuffer[1]~SendBuffer[11]), and the data to be received will be stored to %MB100 to %MB110 (receive [1]~ receive [11]). Depending on the data sent and received, the length of the data to be sent is 11 and the length of the data to be received is 11. The length of the array specified by the input parameter SendBuffer (starting address of the send data cache) is to be equal to or greater than 11, and the length of the array specified by the input parameter receive (starting address of the receive data cache) is to be equal to or greater than 11.

Step 2: The input variable CyclicRun is set to TRUE (cyclic sending and receiving), and after the input parameter execution changes from FALSE to TRUE, the UserDefine1 and UserDefine2 instructions will be triggered. The UserDefine1 instruction appoints COM1 to send the data, and the UserDefine2 instruction appoints COM2 to receive the data.

Chapter 3 CAN communication

3.1 CAN_GetSlaveStatus

Get CANopen slave status. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
CAN_Get-SlaveStatus	Get CANopen slave status	FB		CAN_GetSlaveStatus_Instance(Enable:= parameter, NodeID:= parameter, Valid=> parameter, Error=> parameter, ErrorID=> parameter, ConfigSuccess=> parameter, ConfigFailure=> parameter, DeviceNotMatch=> parameter, InitParameterFailure=> parameter, HeartbeatTimeout=> parameter, EMCY_State=> parameter, NodeIDSame=> parameter, EMCY_Num=> parameter, EMCY_Data=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	Execute when it is TRUE; Not to execute when it is FALSE
NodeID	Slave station number	USINT	Refer to communication instruction specifications	Required field	Specify the station number of the CANopen slave

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the output variable is valid
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
ConfigSuccess	Configure slave successfully	BOOL	TRUE / FALSE	TRUE:Configure slave successfully。
ConfigFailure	Configure slave unsuccessfully	BOOL	TRUE / FALSE	TRUE:Configure slave unsuccessfully 。
DeviceNot-Match	The actual connected slaves do not match the configured slaves	BOOL	TRUE / FALSE	TRUE:The actual connected slaves do not match the configured slaves。
InitParameter-Failure	Parameter initialization failure	BOOL	TRUE / FALSE	TRUE:Parameter initialization failure。
HeartbeatTim-eout	Slave timeout offline	BOOL	TRUE / FALSE	TRUE: Slave timeout offline。
EMCY_State	Slave sends an emergency message	BOOL	TRUE / FALSE	TRUE: TRUE when the slave sends emergency message
NodeIDSame	The slave station number is the same as the master station number	BOOL	TRUE / FALSE	TRUE: The slave station number is the same as the master station number

EMCY_Num	Number of emergency messages sent by the slave	USINT	0~255	Number of emergency messages sent by the slave For one emergency message sent by the slave, the value of this parameter is increased by one
EMCY_Data	Slave sends emergency message data	ARRAY[1..5] OF CAN_EMCY_Type		Slave sends emergency message data

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Enable changes from TRUE to FALSE When the error is eliminated
ConfigSuccess	When configure slave successfully	When Enable changes from TRUE to FALSE
ConfigFailure	When configure slave unsuccessfully	When Enable changes from TRUE to FALSE
DeviceNotMatch	When the actual connected slaves do not match the configured slaves	When Enable changes from TRUE to FALSE
InitParameterFailure	When the parameter initialization is failed	When Enable changes from TRUE to FALSE
HeartbeatTimeout	When the HeartBeat timeout occurs	When Enable changes from TRUE to FALSE
EMCY_State	When the slave station reports an alarm	When Enable changes from TRUE to FALSE
NodeIDSame	When the station number is repeated	When Enable changes from TRUE to FALSE

◆ Function description

Get the current status of the slave whose station number is NodeID in the CANopen network.

Note: This instruction is valid only when the CANopen port of the controller is configured in master mode.

3.2 CAN_GetSlaveStatus usage example

- **Target demand**

Get the CANopen slave status of the M500S Series by the CAN_GetSlaveStatus instruction.

- **Software configuration**

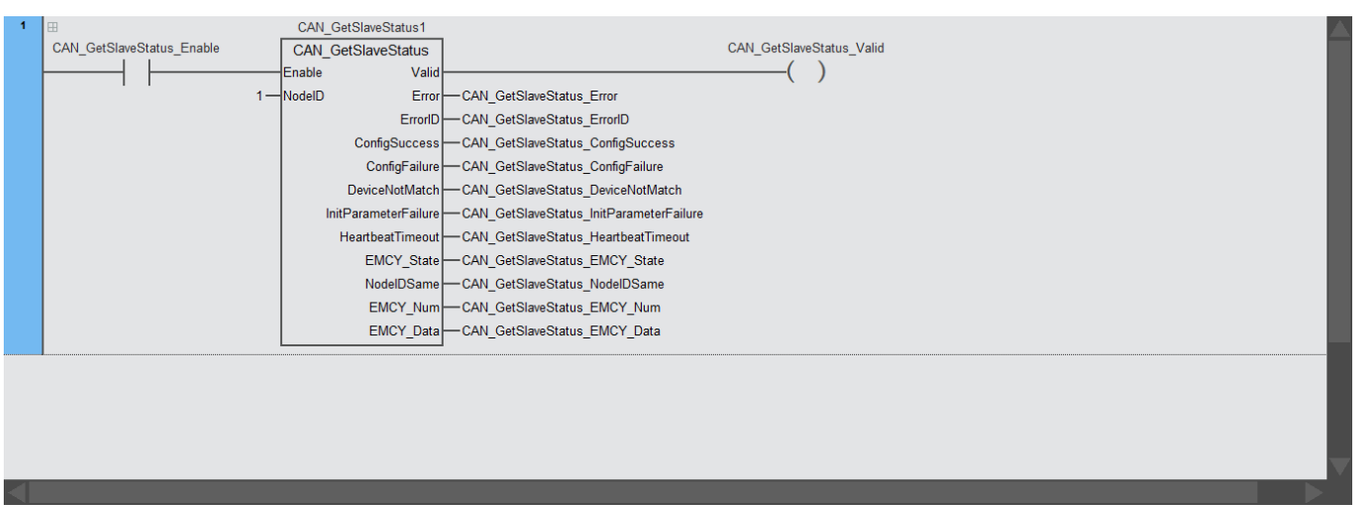
Refer to the "CANopen network settings" section for instructions.

- **Instruction configuration**

Variable table

Category	Name	Assigned to	Data type	Initial value	Comment
VAR	CAN_GetSlaveStatus1		CAN_GetSlaveStatus		
VAR	CAN_GetSlaveStatus_Enable		BOOL		
VAR	CAN_GetSlaveStatus_Valid		BOOL		
VAR	CAN_GetSlaveStatus_Error		BOOL		
VAR	CAN_GetSlaveStatus_ErrorID		WORD		
VAR	CAN_GetSlaveStatus_ConfigSuccess		BOOL		
VAR	CAN_GetSlaveStatus_ConfigFailure		BOOL		
VAR	CAN_GetSlaveStatus_DeviceNotMatch		BOOL		
VAR	CAN_GetSlaveStatus_InitParameterFailure		BOOL		
VAR	CAN_GetSlaveStatus_HeartbeatTimeout		BOOL		
VAR	CAN_GetSlaveStatus_EMCY_State		BOOL		
VAR	CAN_GetSlaveStatus_NodeIDSame		BOOL		
VAR	CAN_GetSlaveStatus_EMCY_Num		USINT		
VAR	CAN_GetSlaveStatus_EMCY_Data		ARRAY[1..5] OF CAN_EMCY_Type		

LD:



ST:

```
CAN_GetSlaveStatus1(Enable:=CAN_GetSlaveStatus_Enable ,
    NodeID:=1 ,
    Valid=>CAN_GetSlaveStatus_Valid ,
    Error=>CAN_GetSlaveStatus_Error ,
    ErrorID=>CAN_GetSlaveStatus_ErrorID ,
    ConfigSuccess=>CAN_GetSlaveStatus_ConfigSuccess ,
    ConfigFailure=>CAN_GetSlaveStatus_ConfigFailure ,
    DeviceNotMatch=>CAN_GetSlaveStatus_DeviceNotMatch ,
    InitParameterFailure=>CAN_GetSlaveStatus_InitParameterFailure ,
    HeartbeatTimeout=>CAN_GetSlaveStatus_HeartbeatTimeout ,
    EMCY_State=>CAN_GetSlaveStatus_EMCY_State ,
    NodeIDSame=>CAN_GetSlaveStatus_NodeIDSame ,
    EMCY_Num=>CAN_GetSlaveStatus_EMCY_Num ,
    EMCY_Data=>CAN_GetSlaveStatus_EMCY_Data
    );
```

- **Program description**

Step 1: NodeID writes the slave node number.

Step 2: Write TRUE to CAN_GetSlaveStatus_Enable and then check if the slave is successfully configured by CAN_GetSlaveStatus_ConfigSuccess.

Step 3: When CAN_GetSlaveStatus_Enable is TRUE, and CAN_GetSlaveStatus_Error changes from FALSE to TRUE, the error code can be viewed by the CAN_GetSlaveStatus_ErrorID.

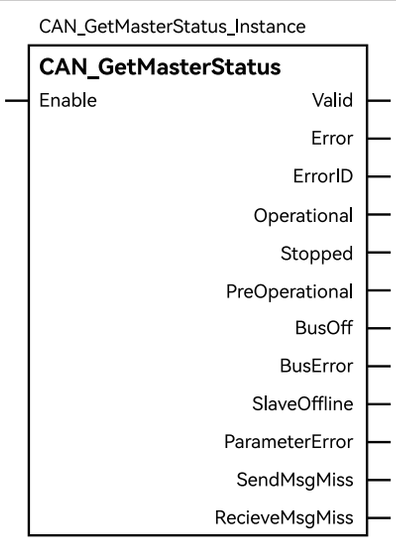
Step 4: When CAN_GetSlaveStatus_Enable is TRUE, and CAN_GetSlaveStatus_DeviceNotMatch changes from FALSE to TRUE, check whether the configured slave station is the same as the actual one and whether the station number is the same.

Step 5: When CAN_GetSlaveStatus_Enable is TRUE, and CAN_GetSlaveStatus_EMCY_State changes from FALSE to TRUE, the configured slave station number is checked for conflict.

Step 6: When CAN_GetSlaveStatus_Enable is TRUE, the number of slave alarms can be viewed by CAN_GetSlaveStatus_EMCY_Num, and CAN_GetSlaveStatus_EMCY_Data is the content of slave alarms.

3.3 CAN_GetMasterStatus

Get CANopen master status. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
CAN_GetMas-terStatus	Get CANopen master status	FB		CAN_GetMasterStatus_Instance(Enable:= parameter, Valid=> parameter, Error=> parameter, ErrorID=> parameter, Operational=> parameter, Stopped=> parameter, PreOperational=> parameter, BusOff=> parameter, BusError=> parameter, SlaveOffline=> parameter, ParameterError=> parameter, SendMsgMiss=> parameter, RecieveMsgMiss=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	Execute the instruction when this parameter is TRUE

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the output variable is valid
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Operational	Operating	BOOL	TRUE / FALSE	The status of the master in the CANopen network TRUE: The master station is in operation
Stopped	Stopped	BOOL	TRUE / FALSE	The status of the master in the CANopen network TRUE: The master station is in the stopped status
PreOperational	Preoperational	BOOL	TRUE / FALSE	The status of the master in the CANopen network TRUE: The master station is in the pre-operational status
BusOff	The bus communi-cation is off	BOOL	TRUE / FALSE	TRUE: The bus communication is off When this variable is TRUE, it is necessary to detect whether the field hardware is wired correctly, whether there is interference in the field environment, and whether the equipment is grounded.
BusError	Bus abnormality	BOOL	TRUE / FALSE	TRUE: Bus is abnormal. This may occur when the bus data is sent several times before it is successfully sent. For example, when the field environment is poor or there is interference, there may occur bus error
SlaveOffline	Slave offline	BOOL	TRUE / FALSE	TRUE: When any of the slaves configured within the master is offline after establishing a connection with the master

ParameterError	Configuration error when configuring slave parameters	BOOL	TRUE / FALSE	TRUE: An error occurs when the master configures the slave parameters, i.e., the master cannot establish a connection with the slave. For example, the error occurs when the slave parameters configured by the software are not supported by the slave.
SendMsgMiss	Data sending error	BOOL	TRUE / FALSE	TRUE: when the master sends data with an error. For example, when the send cache is full due to excessively fast sending of data.
RecieveMsgMiss	Data receiving error	BOOL	TRUE / FALSE	TRUE: when the master receives data with an error, such as when the receiving cache is full due to excessively fast receiving of data.

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Enable changes from TRUE to FALSE When the error is eliminated
Operational	The master station is in operation	When Enable changes from TRUE to FALSE When the master station is not in operation
Stopped	The master station is in the stopped status	When Enable changes from TRUE to FALSE When the master station is not in the stopped status
PreOperational	The master station is in the pre-operational status	When Enable changes from TRUE to FALSE When the master station is not in the pre-operational status
BusOff	The bus communication is off	When Enable changes from TRUE to FALSE When the bus is repaired
BusError	When the bus is abnormal	When Enable changes from TRUE to FALSE When the bus abnormality is eliminated
SlaveOffline	When the slave is offline	When Enable changes from TRUE to FALSE
ParameterError	When there is an error in configuring the slave parameters	When Enable changes from TRUE to FALSE
SendMsgMiss	When there is a data sending error	When Enable changes from TRUE to FALSE
RecieveMsgMiss	When there is a data receiving error	When Enable changes from TRUE to FALSE

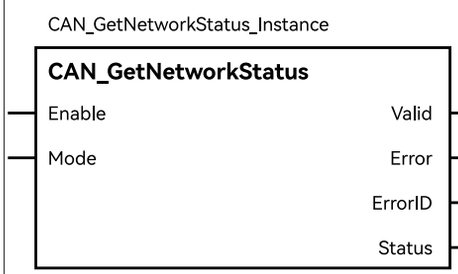
◆ Function description

This instruction is used to get the current status of the master in the CANopen network.

Note: This instruction is valid only when the CANopen port of the controller is configured in master mode.

3.4 CAN_GetNetworkStatus

Get the status of all slaves in a CANopen network. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
CAN_GetNetworkStatus	Get the status of all slaves in a CANopen network.	FB		CAN_GetNetworkStatus_Instance(Enable:= parameter, Mode:= parameter, Valid=> parameter, Error=> parameter, ErrorID=> parameter, Status=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	Execute when it is TRUE; Not to execute when it is FALSE
Mode	Status type	USINT	1-3	Required field	Set the category to get the status of all slaves in the CANopen network 1: Whether the slave is configured; 2: Whether the slave has established a connection with the master; 3: Whether the slave sends an emergency message

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the output variable is valid
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Status	Status	ARRAY[1..32] OF BOOL		Status of all slaves. Array element 1 corresponds to the status of the slave with station number 1 and element 2 corresponds to the status of the slave with station number 2. TRUE indicates consistency with the category set by the input variable Mode; FALSE indicates inconsistency. Refer to the Function description section of this instruction for details.

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Enable changes from TRUE to FALSE When the error is eliminated

◆ Function description

This instruction gets the status of all slaves in the CANopen network. The status of each slave is determined by the input variable "Mode" and the result is returned by the parameter "Status". For instance, if Mode is set to 2, Status[1] indicates wheth-

er communication between the master and slave with station number 1 is normal or not. TRUE indicates normal communication, while False indicates abnormal communication. Similarly, Status[2] indicates whether communication between the master and slave with station number 2 is normal or not, and so on. TRUE indicates normal communication, while False indicates abnormal communication. If Mode is set to 1, Status[1] indicates whether slave station 1 is configured. TRUE indicates successful configuration, while False indicates no configuration. Status[2] indicates whether the slave with station number 2 has been configured. TRUE indicates successful configuration, while False indicates no configuration, and so on.

Note: This instruction is valid only when the CANopen port of the controller is configured in master mode.

3.5 CAN_GetNetworkStatus usage example

- **Target demand**

Get the status of all slaves in the CANopen network with the CAN_GetNetworkStatus instruction.

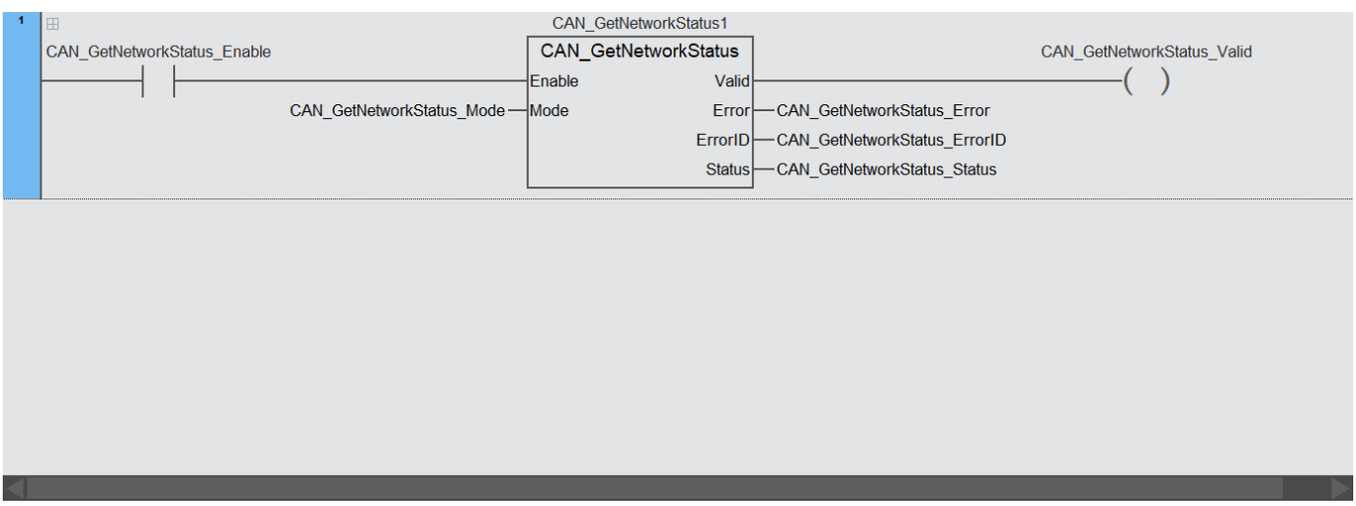
- **Software configuration**

Refer to the "CANopen network settings" section for instructions.

Variable table

Category	Name	Assigned to	Data type	Initial value	Comment
VAR	CAN_GetNetworkStatus1		CAN_GetNetworkStatus		
VAR	CAN_GetNetworkStatus_Enable		BOOL		
VAR	CAN_GetNetworkStatus_Mode		USINT		
VAR	CAN_GetNetworkStatus_Valid		BOOL		
VAR	CAN_GetNetworkStatus_Error		BOOL		
VAR	CAN_GetNetworkStatus_ErrorID		WORD		
VAR	CAN_GetNetworkStatus_Status		ARRAY[1..32] OF BOOL		

LD:



ST:

```
CAN_GetNetworkStatus1(Enable:=CAN_GetNetworkStatus_Enable ,
    Mode:=CAN_GetNetworkStatus_Mode ,
    Valid=>CAN_GetNetworkStatus_Valid ,
    Error=>CAN_GetNetworkStatus_Error ,
    ErrorID=>CAN_GetNetworkStatus_ErrorID ,
    Status=>CAN_GetNetworkStatus_Status
    );
```

- **Program description**

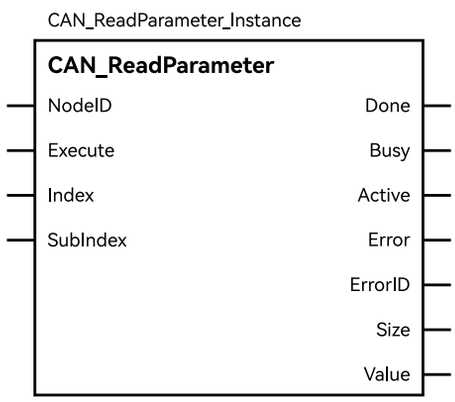
Step 1: Write 1 to CAN_GetNetworkStatus_Mode to check if the slave is configured by CAN_GetNetworkStatus_Status, write 2 to CAN_GetNetworkStatus_Mode to check if connection with the slave is established by CAN_GetNetworkStatus_Status, and write 3 to CAN_GetNetworkStatus_Mode to check if the slave reports an alarm.

Step 2: CAN_GetNetworkStatus_Mode triggers CAN_GetNetworkStatus_Enable after writing the value, and then CAN_GetNetworkStatus_Status detects the slave status.

Step 3: When CAN_GetNetworkStatus_Enable is TRUE and CAN_GetNetworkStatus_Error changes from FALSE to TRUE, the error code can be viewed by CAN_GetNetworkStatus_ErrorID.

3.6 CAN_ReadParameter

Read the parameters in the CANopen slave. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
CAN_ReadParameter	Read the parameters in the CANopen slave	FB		CAN_ReadParameter_Instance(NodeID:= parameter, Execute:= parameter, Index:= parameter, SubIndex:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Error=> parameter, ErrorID=> parameter, Size=> parameter, Value=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
NodeID	Slave station number	USINT	1~63	Required field	Specify the slave station number
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
Index	Index of the parameters	UINT	0~65535	0	Set the index of the parameter to be read
SubIndex	Subindex of parameters	USINT	0~255	0	Set the sub-index of the parameter to be read

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	Instruction in control	BOOL	TRUE / FALSE	TRUE when the instruction is in control
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Size	Data type of parameters	USINT		The type of the parameter to be read. 1: Byte (1 byte) 2: Word (2 bytes) 4: DWord (4 bytes)
Value	The value of the parameter that has been read	UDINT		The value of the parameter that has been read

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE

Active	The instruction controls the read parameters.	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

• Basic function description

This instruction is used to read the parameter of the CANopen slave, the parameter to be read is specified by Index and SubIndex. The Index and SubIndex of the slave parameters can be obtained from the slave-related information.

This instruction reads the slave's parameters according to the value set by the input variable when Execute changes from FALSE to TRUE.

When this instruction is executed, the value of the specified slave parameter is read into the output variable Value. The data type of the output variable Value is UDINT. When the data type of Value and the data type of the read slave parameter do not match, the data type of the Value variable can be converted to the data type of the slave parameter by the data type conversion instruction. If the data type of the read slave parameter is DINT, the value of Value can be converted to the variable of DINT type by UDINT_TO_DINT instruction; if the data type of the read slave parameter is INT, the value of Value can be converted to the variable of INT type by UDINT_TO_INT instruction.

• Instruction completion timing

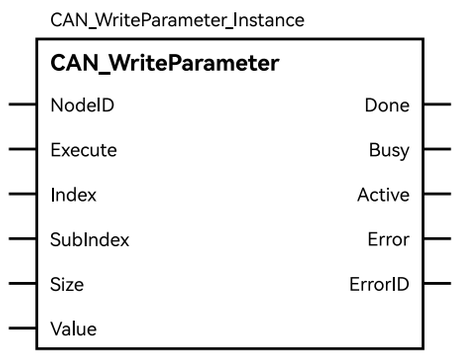
When the value of the slave's parameter is read, the instruction is completed and the Done bit changes from FALSE to TRUE.

• Re-execute the instruction

When an instruction is executed and Execute changes from FALSE to TRUE again, the instruction can be re-executed; When the instruction is being executed and Execute changes from FALSE to TRUE again, there is no effect on the execution of the instruction, and the instruction still executes the instruction in accordance with the input variables that have not been executed to completion.

3.7 CAN_WriteParameter

Set the parameters in the CANopen slave. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
CAN_WritePa-rameter	Set the pa-rameters in the CANopen slave	FB		CAN_WriteParameter_Instance(NodeID:= parameter, Execute:= parameter, Index:= parameter, SubIndex:= parameter, Size:= parameter, Value:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Error=> parameter, ErrorID=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
NodeID	Slave station number	USINT	1~63	Required field	Specify the slave station number
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
Index	Index of the parameters	UINT	0~65535	0	Set the index of the parameter to be set
SubIndex	Subindex of parameters	USINT	0~255	0	Set the sub-index of the parameter to be set.
Size	Data type of parameters	USINT	1~4	Required field	Set the type of parameter to be set. 1: Byte (1 byte) 2: Word (2 bytes) 4: DWord (4 bytes)
Value	Set value	UDINT	0 ~ 4294967295	0	Set value

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	Instruction in control	BOOL	TRUE / FALSE	TRUE when the instruction is in control
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

*1 Refer to "instruction error code description" for the meaning of the output error code value.

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE

Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Active	The instruction controls the write parameters.	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

• Basic function description

This instruction is used to set the parameter values of the CANopen slave, which are specified by Index and SubIndex. The Index and SubIndex of the slave parameters can be obtained from the slave data. The parameter value is the value of the input variable Value.

This instruction sets the slave's parameters according to the value set by the input variable when Execute changes from FALSE to TRUE.

When this instruction is executed, the value in the input variable Value is set to the specified slave parameter. The data type of the input variable Value is UDINT. When the data type of Value and the data type of the slave parameter do not match, the data type of the Value variable can be converted to the data type of the slave parameter by the data type conversion instruction. If the data type of the slave parameter is DINT, users can use the UDINT_TO_DINT instruction to convert the value of Value to a variable of DINT type; if the data type of the slave parameter is INT, the value of Value can be converted to a variable of INT type by using the UDINT_TO_INT instruction.

• Instruction completion timing

When the instruction to write the parameter value is sent, the instruction is completed and the Done bit changes from FALSE to TRUE.

• Re-execute the instruction

When an instruction is executed and Execute changes from FALSE to TRUE again, the instruction can be re-executed. When the instruction is being executed and Execute changes from FALSE to TRUE again, there is no effect on the execution of the instruction, and the instruction still executes the instruction in accordance with the input variables that have not been executed to completion.

3.8 Example of CAN parameter reading and writing

- **Target demand**

The M511S CANopen master reads and writes the M511S CANopen slave parameter 16#2000:01 (index: subindex) by the CAN_ReadParameter instruction and CAN_WriteParameter instruction.

- **Software configuration**

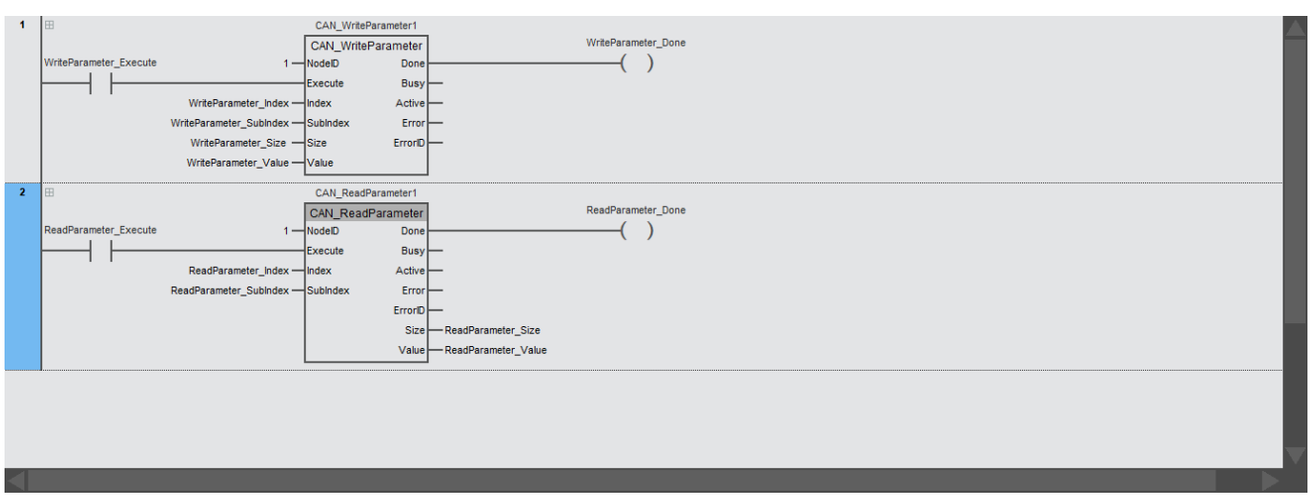
Refer to the "CANopen network settings" section for instructions.

- **Instruction configuration**

Variable table

Category	Name	Assigned to	Data type	Initial value	Comment
VAR	CAN_WriteParameter1		CAN_WriteParameter		
VAR	CAN_ReadParameter1		CAN_ReadParameter		
VAR	WriteParameter_Execute		BOOL		
VAR	ReadParameter_Execute		BOOL		
VAR	WriteParameter_Done		BOOL		
VAR	ReadParameter_Done		BOOL		
VAR	WriteParameter_Value		udint		
VAR	ReadParameter _Value		udint		
VAR	WriteParameter_Index		uint		
VAR	ReadParameter _Index		uint		
VAR	WriteParameter_SubIndex		usint		
VAR	ReadParameter_SubIndex		usint		
VAR	WriteParameter_Size		usint		
VAR	ReadParameter_Size		usint		

LD:



ST:

```
CAN_WriteParameter1
(NodeID:=1 ,
    Execute:=WriteParameter_Execute ,
    Index:=WriteParameter_Index ,
    SubIndex:=WriteParameter_SubIndex ,
    Size:=WriteParameter_Size ,
    Value:=WriteParameter_Value,
    Done=>WriteParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=>
    );
CAN_ReadParameter1
(NodeID:=1 ,
    Execute:=ReadParameter_Execute ,
    Index:=ReadParameter_Index ,
    SubIndex:=ReadParameter_SubIndex ,
    Done=>ReadParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    Size=>ReadParameter_Size ,
    Value=>ReadParameter_Value
    );
```

- **Program description**

Step 1: Set the value of the input variable NodeID to 1, which indicates that the write parameter is written to the slave with station number 1.

Step 2: Set the value of the input variable WriteParameter_Index to 16#2000, the value of the input variable WriteParameter_SubIndex to 1, the value of the input variable WriteParameter_Size to 2, and the value of the input variable WriteParameter_Value to 1.

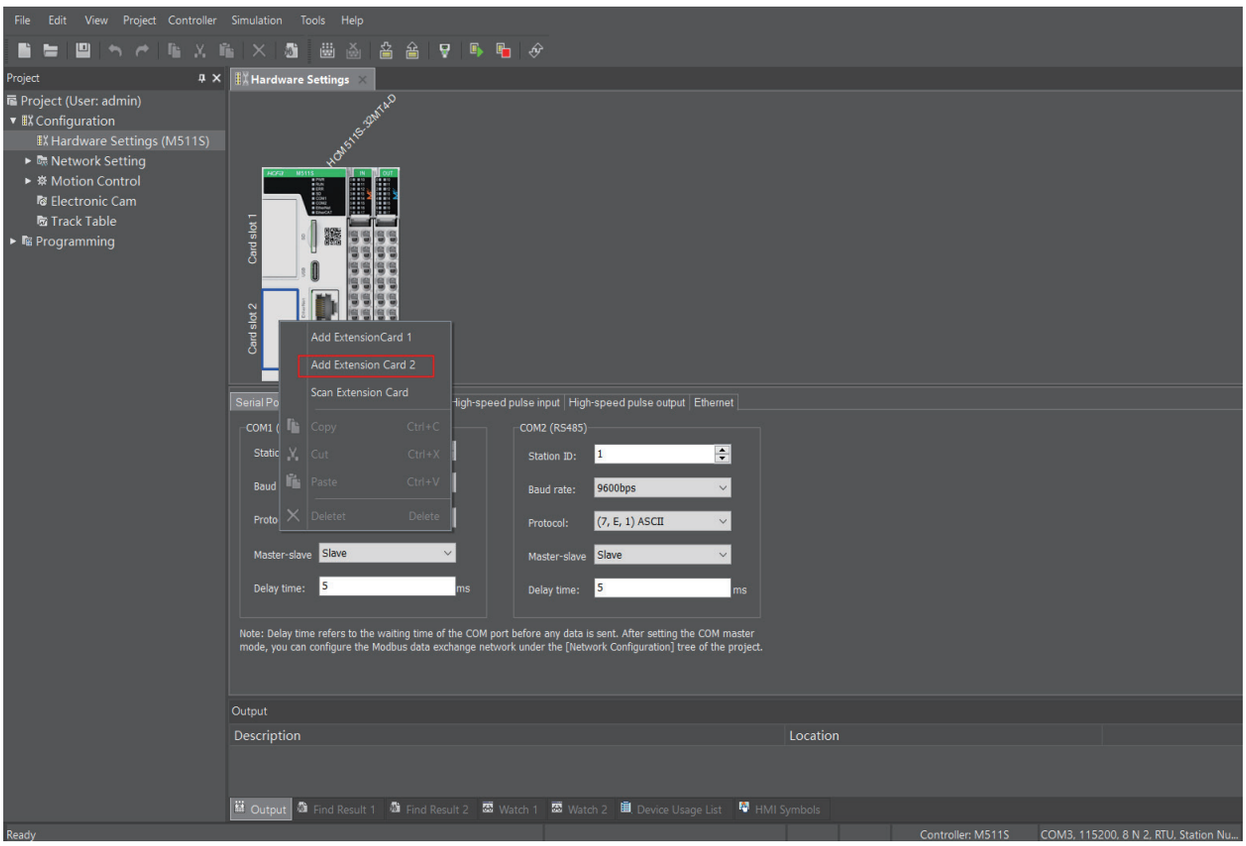
Step 3: When the input variable WriteParameter_Execute changes from FALSE to TRUE, it triggers the CAN_WriteParameter instruction to be executed, and when the WriteParameter_Done bit changes to TRUE, it indicates that the parameter write is completed.

Step 4: Set the value of the input variable ReadParameter_Index to 16#2000 and the value of the input variable ReadParameter_SubIndex to 1.

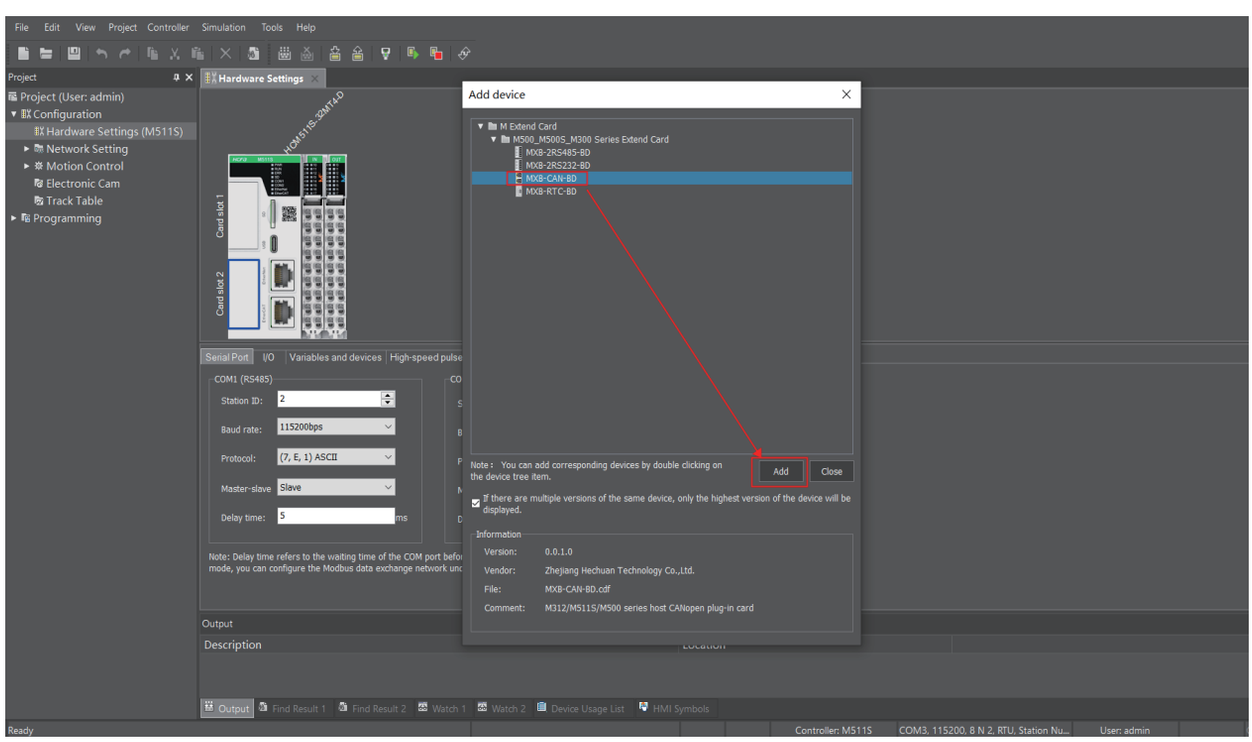
Step 5: When the input variable ReadParameter_Execute changes from FALSE to TRUE, it triggers the execution of the CAN_ReadParameter instruction. When the ReadParameter_Done bit changes to TRUE, it indicates that the parameter reading is completed. After that, the ReadParameter_Size and the ReadParameter_Size and ReadParameter_Value are 2 and 1 respectively, and the value of ReadParameter_Value is the value of the parameter that has been read.

3.9 CANopen network settings

Step 1: Double-click on hardware settings - click on the card slot and then right-click on Add Expansion Card. Double-click on "Hardware settings" and click on the slot corresponding to the controller and right-click on it, and finally click on "Add expansion card 2".

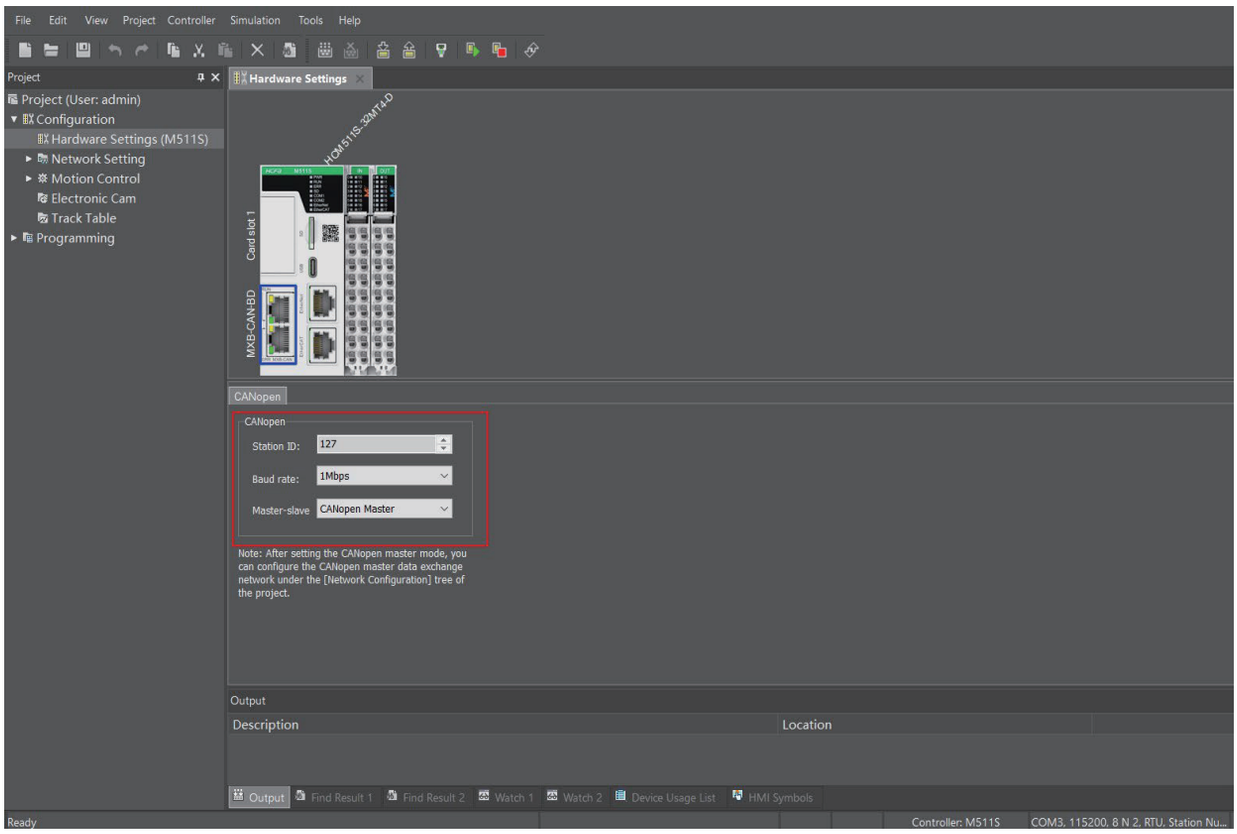


Step 2: Double-click on "MXB-CAN-BD" or click on the "MXB-CAN-BD" and then click on the "Add" button at the red box in the figure below to add a slave.

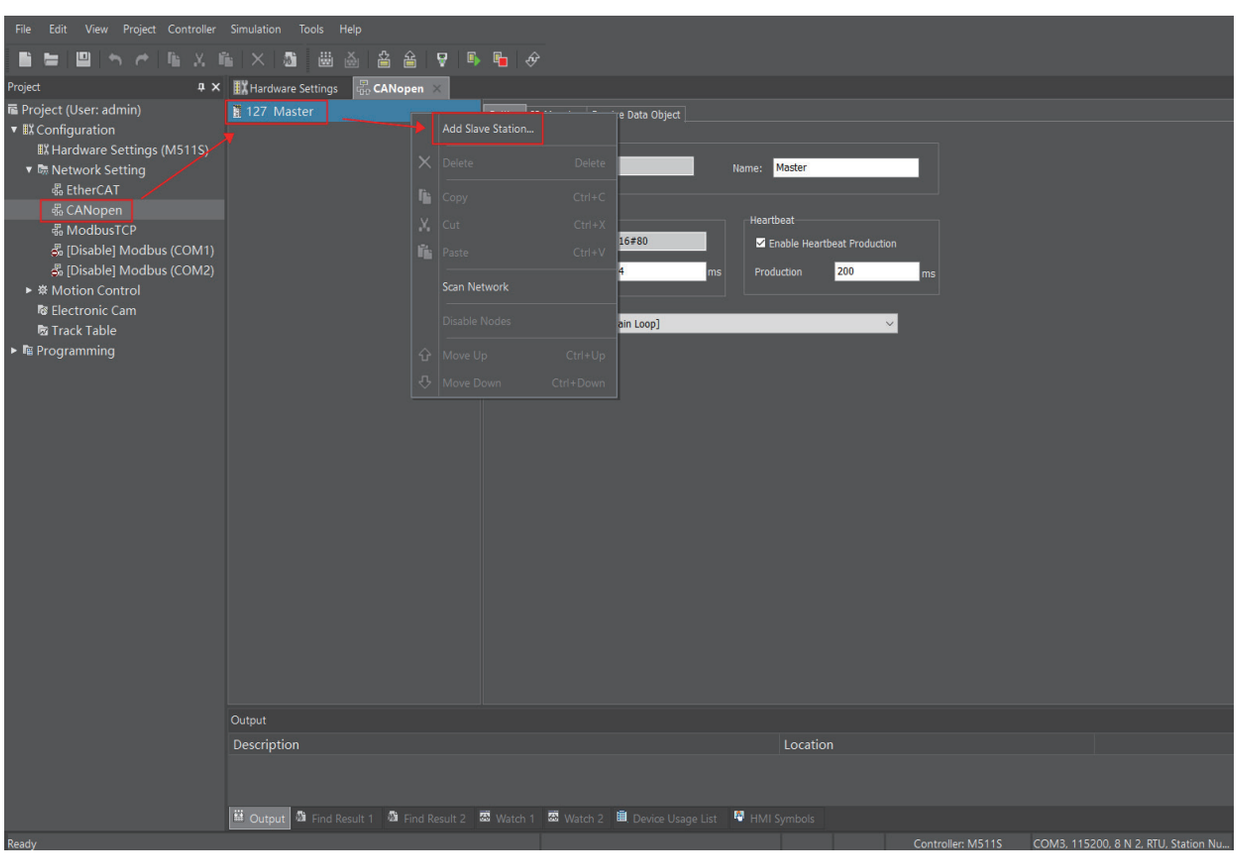


Step 3: Set the master station number, baud rate and master-slave mode (the master and slave station numbers should

not be repeated and the baud rate should be the same).

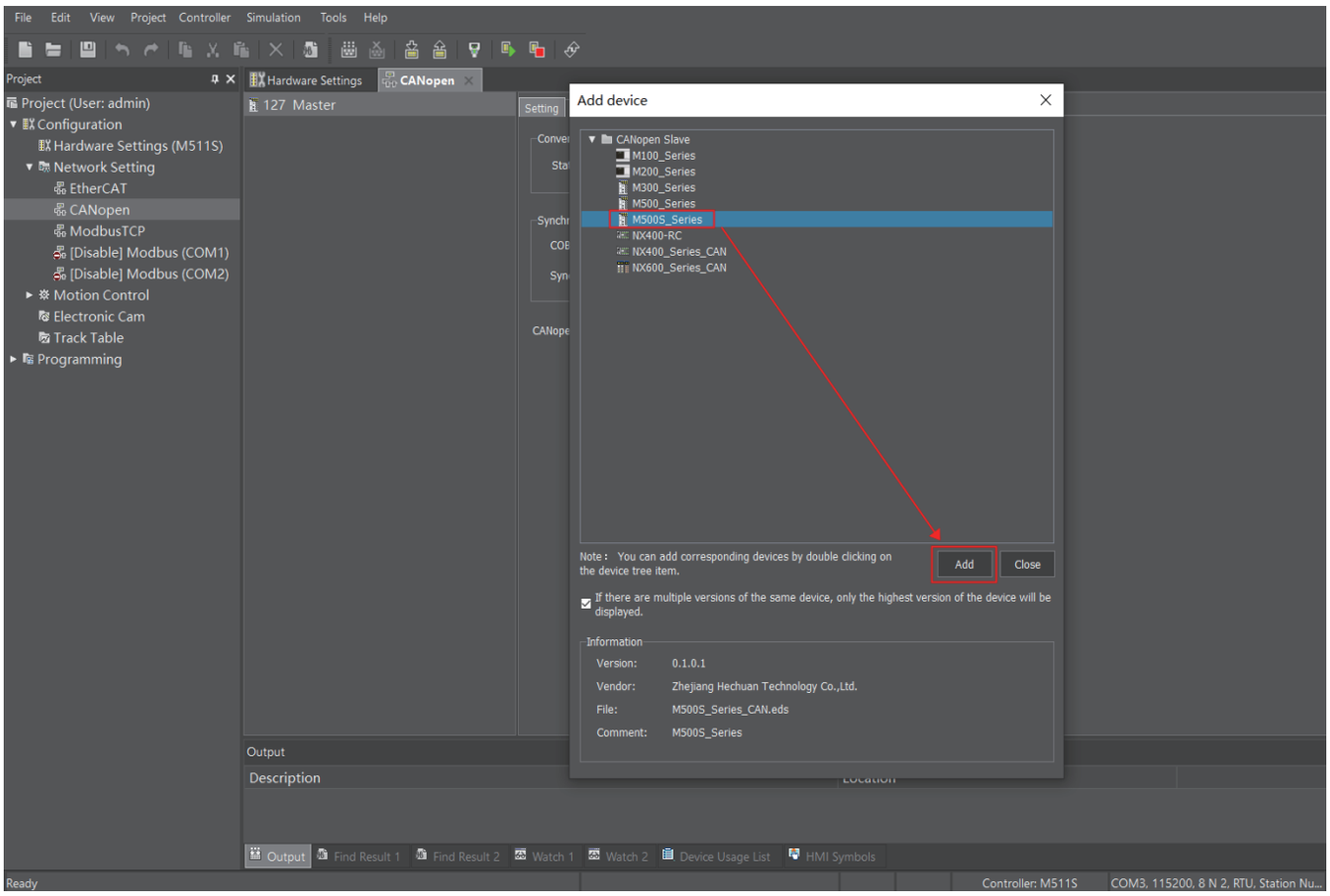


Step 4: Click on "Network Settings", then double-click on "CANopen", click on "127 Master", then right-click on it. Finally, click on “Add Slave”.

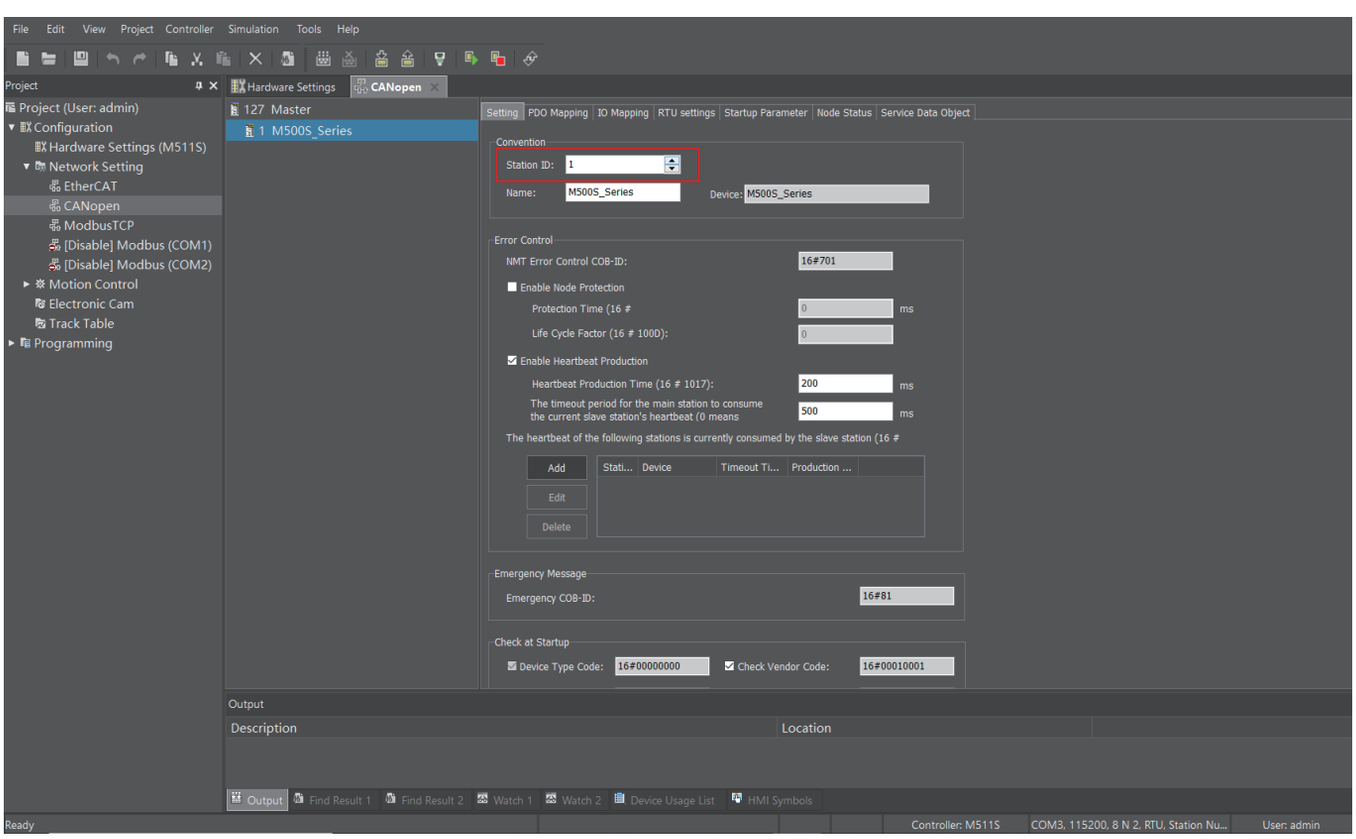


Step 5: Double-click on the M500S Series to add a slave. Or select M500S Series and then click on the "Add" button in the

red box below to add a slave.



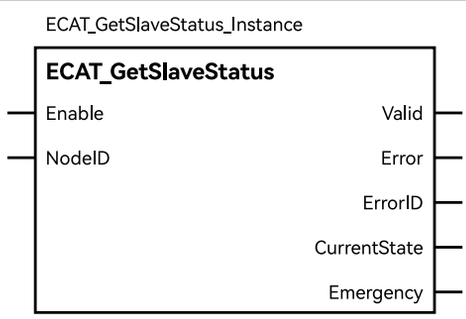
Step 6: Specify the slave station number (the slave station number specified here must be the same as the actual connected slave station number).



Chapter 4 EtherCAT communication

4.1 ECAT_GetSlaveStatus

Get EtherCAT slave status. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ECAT_Get-SlaveStatus	Get EtherCAT slave status	FB		ECAT_GetSlaveStatus_Instance(Enable:= parameter, NodeID:= parameter, Valid=> parameter, Error=> parameter, ErrorID=> parameter, CurrentState=> parameter, Emergency=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Enable	Effective	BOOL	TRUE / FALSE	FALSE	Execute when it is TRUE; Not to execute when it is FALSE
NodeID	Slave EtherCAT address	UINT	Refer to communication instruction specifications	Required field	Specify the address of the EtherCAT slave. For example, the address of the 1st slave station is 1001, the address of the 2nd slave station is 1002, and so on.

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Valid	The output variable is valid	BOOL	TRUE / FALSE	TRUE when the output variable is valid
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
CurrentState	Current status of the slave	ECAT_State_Type		Current status of the EtherCAT slave 0: Unknown; 1: Init; 2: PreOP ; 3: SafeOP ; 4: OP
Emergency	Error code value in slave emergency message	UINT	0~65535	Error code value in slave emergency message

***1 Refer to "instruction error code description" for the meaning of the output error code value.**

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Valid	When Enable is TRUE	When Enable changes from TRUE to FALSE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Enable changes from TRUE to FALSE When the error is eliminated

◆ Function description

This instruction is used to get the current status of the specified slave in the EtherCAT network, which is specified by the input variable NodeID.

When the master (controller) establishes a connection with an EtherCAT slave, the normal slave state changes as follows: Init => PreOP => SafeOP => OP. When the slave is in OP status, it means that the communication connection between the master and the slave has been established, i.e. the communication between the master and the slave is normal.

If the EtherCAT slave is to be controlled or parameters are to be read, this instruction can be used to judge that the slave status is in the OP (Operational) status before carrying out the relevant operation.

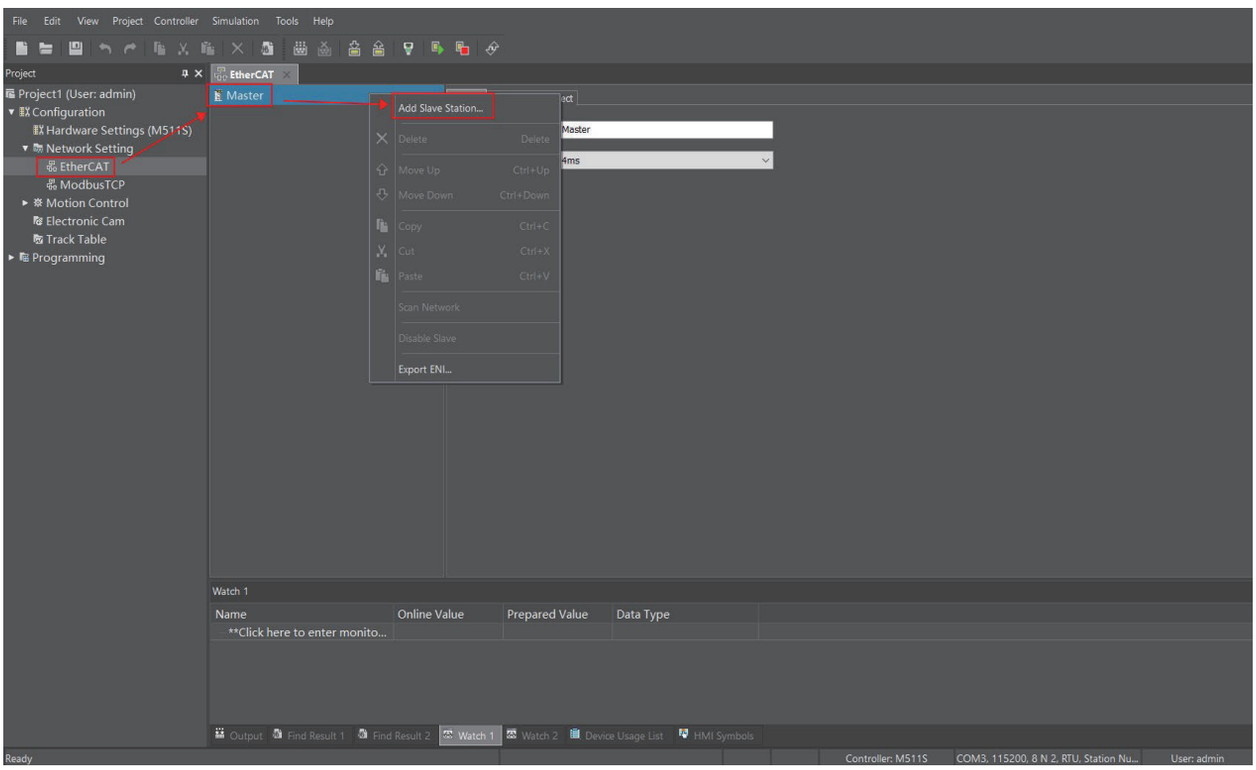
4.2 ECAT_GetSlaveStatus usage example

- **Target demand**

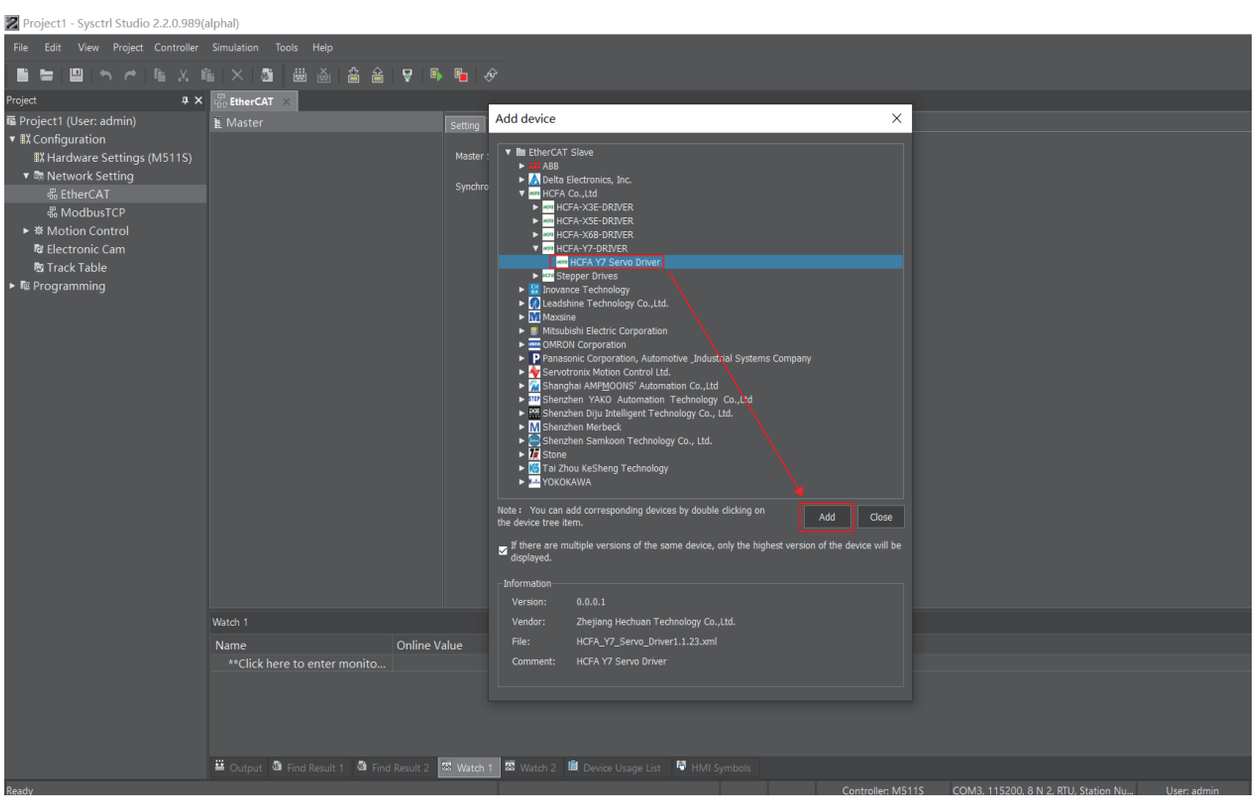
Get the EtherCAT slave status of HCFA Y7 series servo drive by ECAT_GetSlaveStatus instruction.

- **Software configuration**

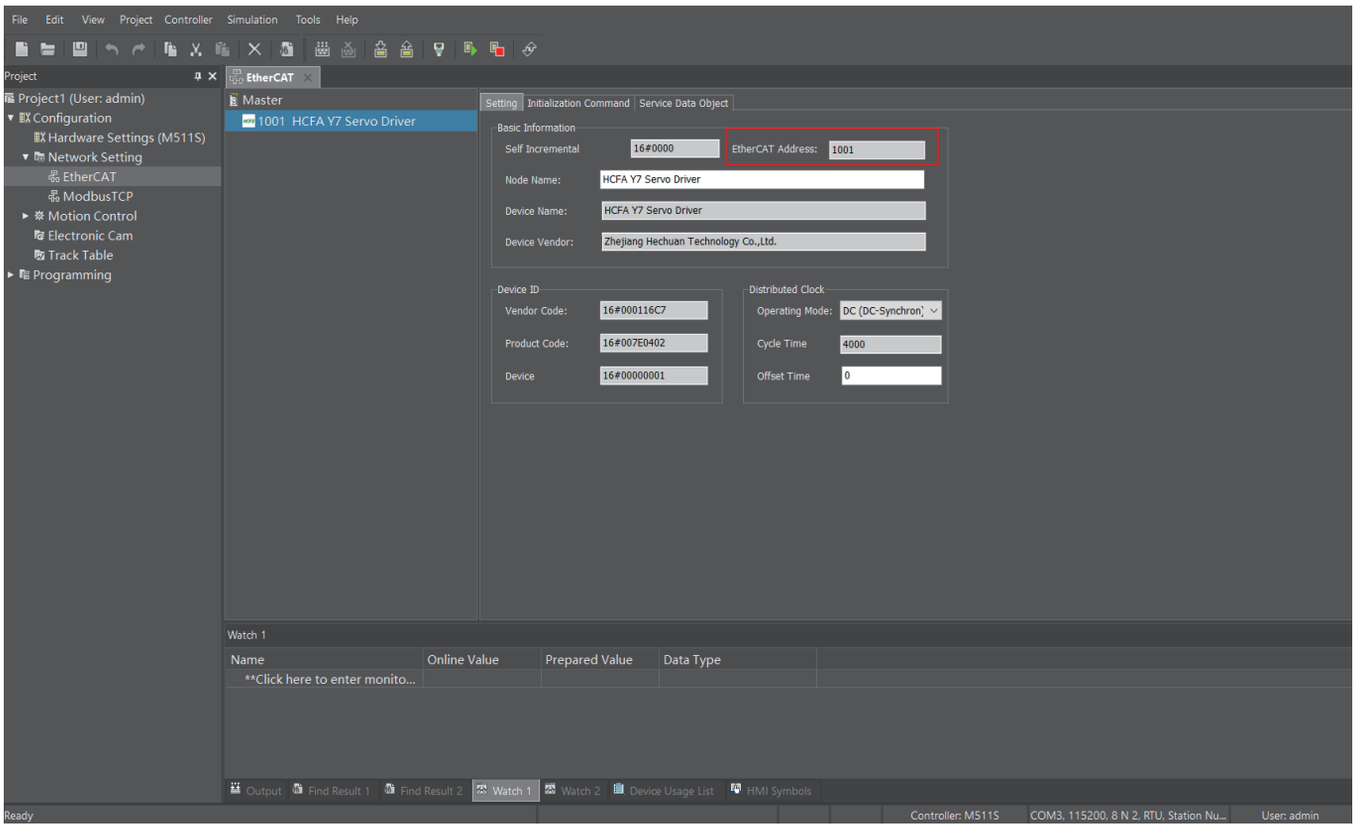
Step 1: Click on "Network settings", and then double-click on "EtherCAT", right-click on Master and click on Add Slave.



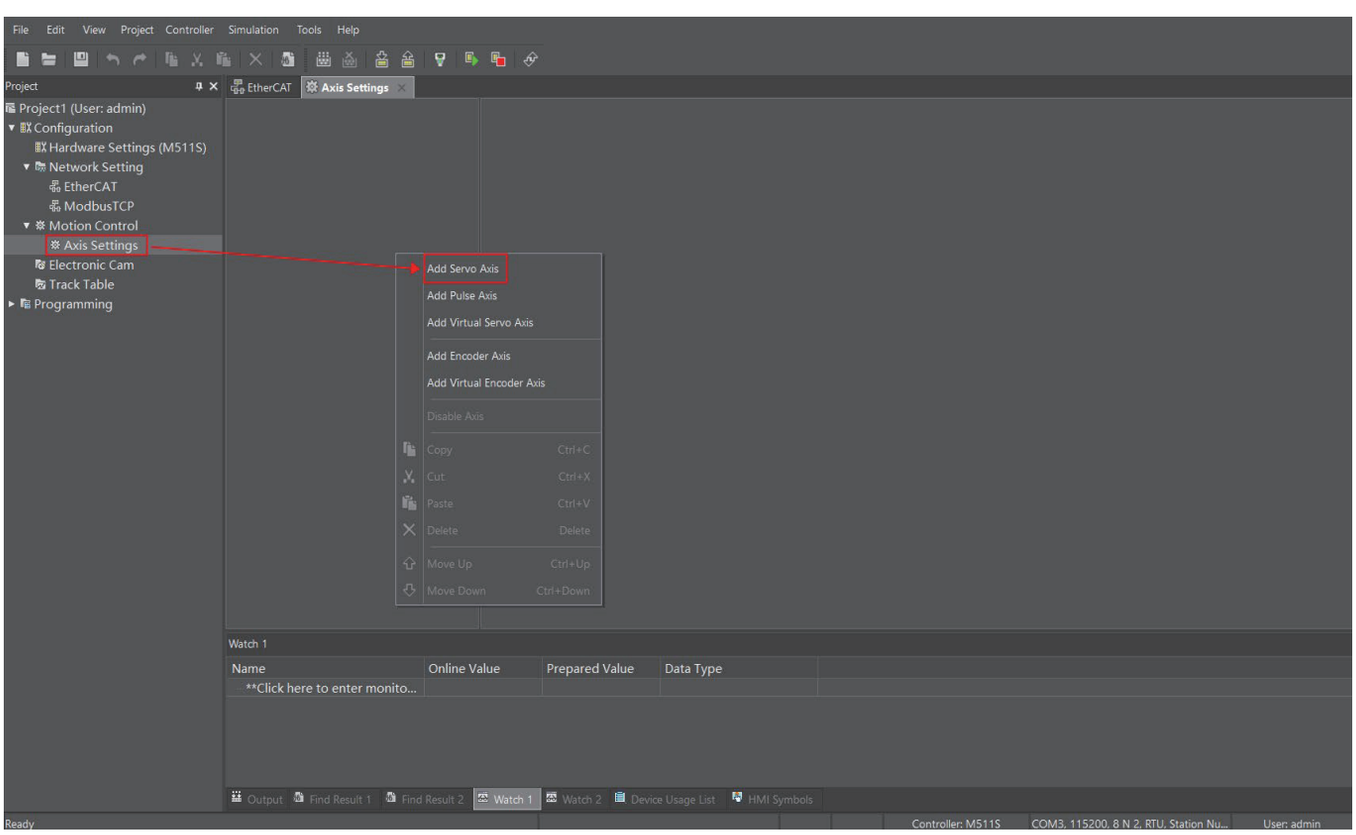
Step 2: Double-click on the HCFA Y7 servo drive to add a slave. Or select HCFA Y7 servo drive and then click on the "Add" button in the red box below to add a slave.



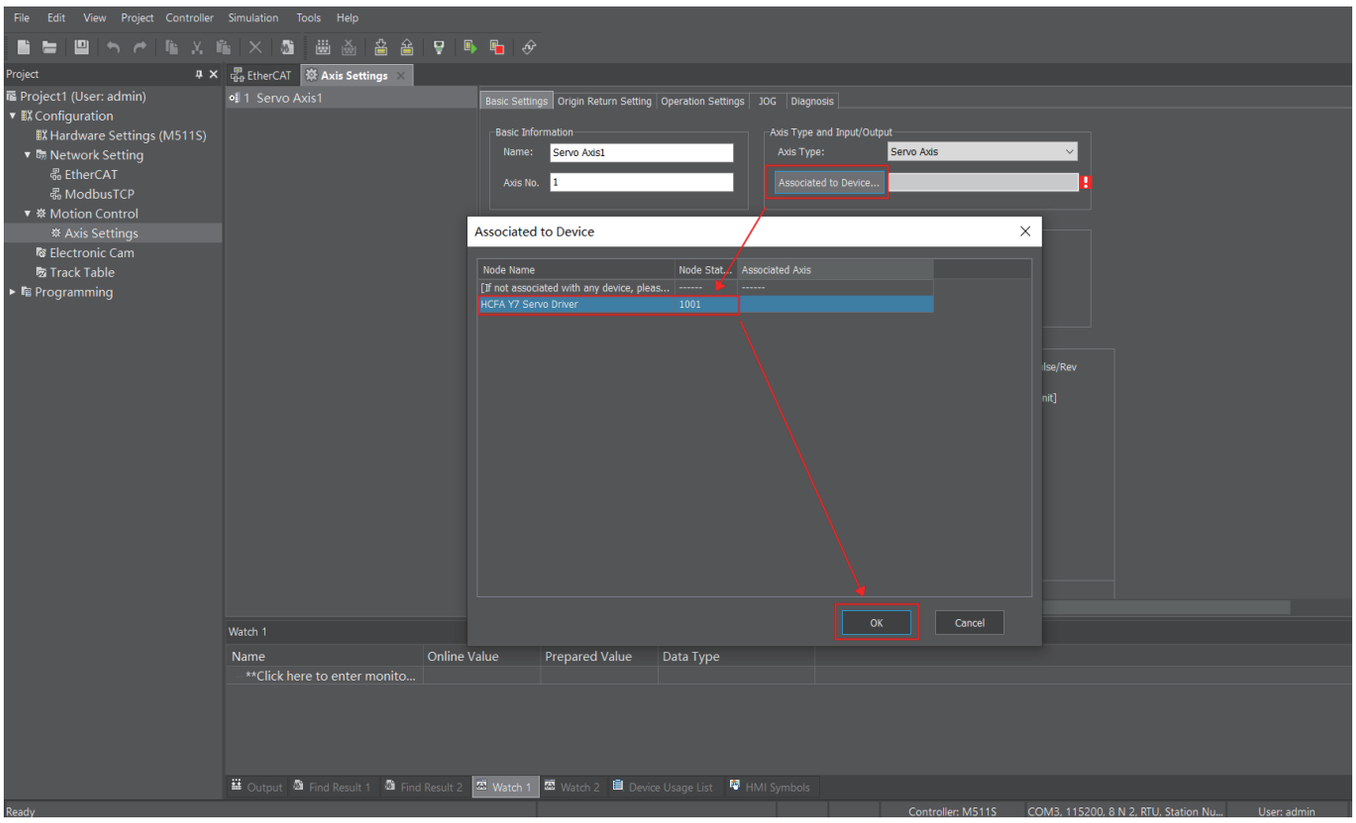
Step 3: The EtherCAT address of the 1st EtherCAT slave is 1001, the EtherCAT address of the 2nd EtherCAT slave is 1002 and so on.



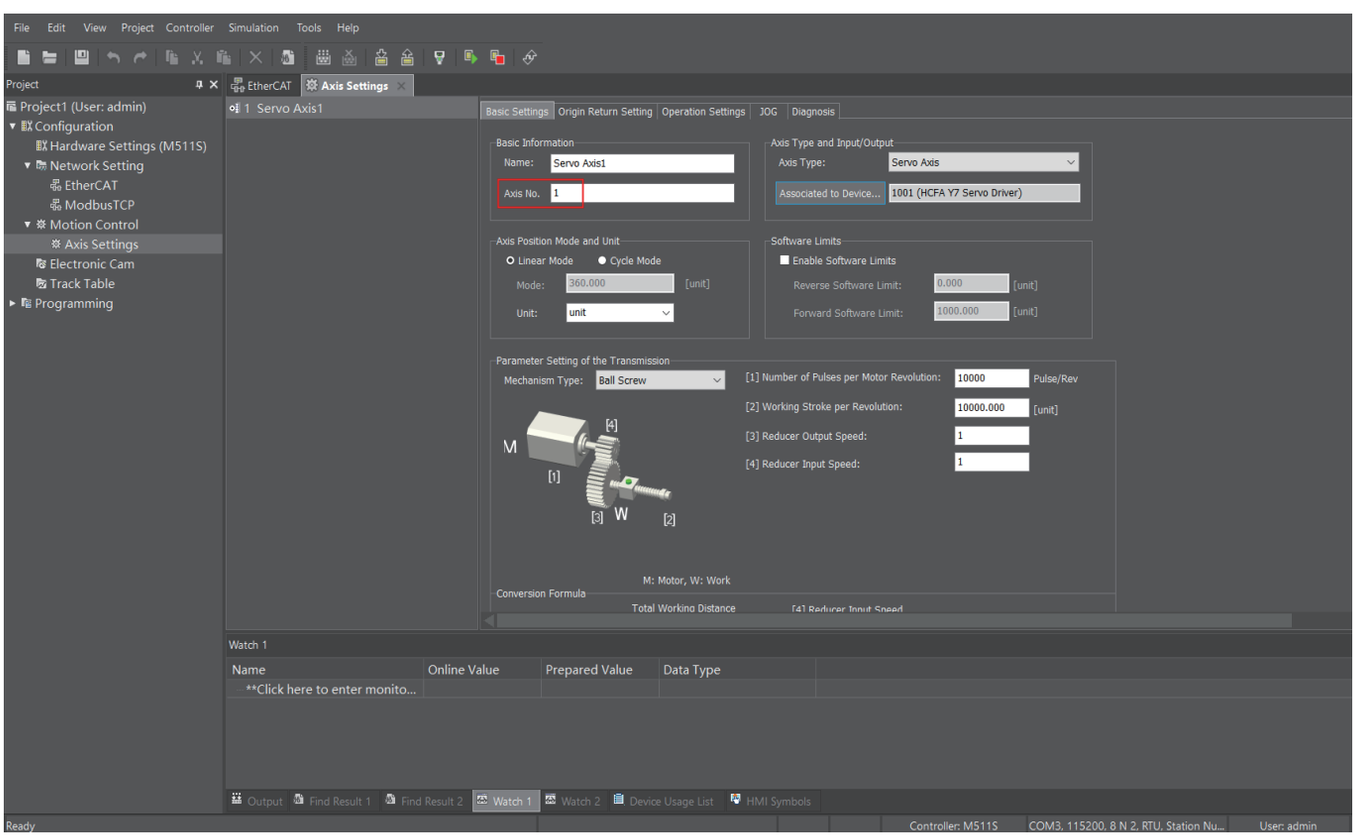
Step 4: Double-click on Axis setting - right-click on the middle blank space - click on Add servo axis.



Step 5: Click on “Link to device” -> Click on “HCFA Y7 servo drive” -> Click on the “OK” button. With this step, a one-to-one relationship is established between the axis and the EtherCAT slave.



Step 6: Set the axis number at the red box in the figure below.



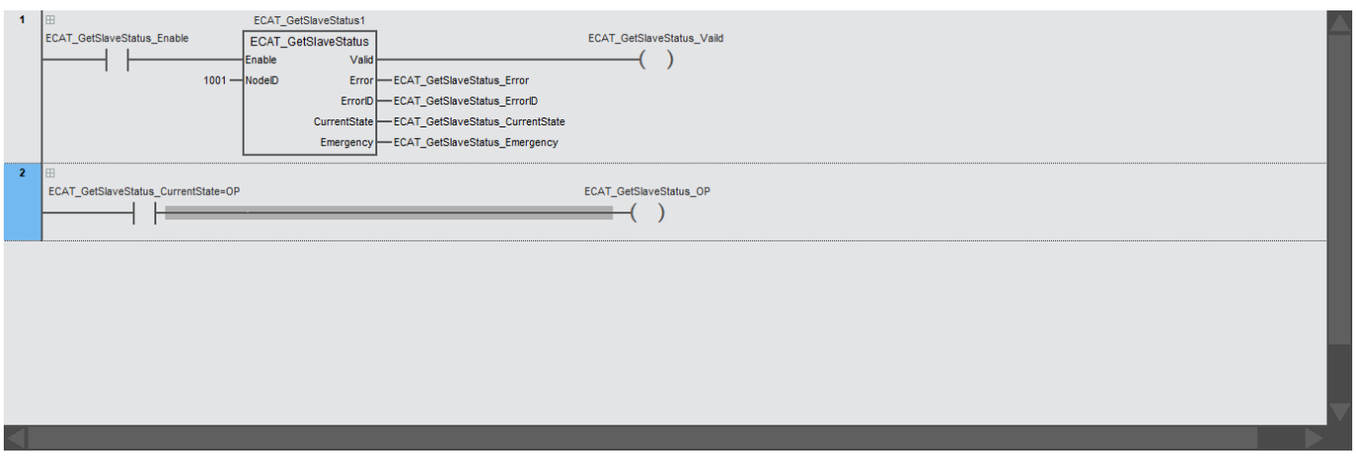
- **Instruction configuration**

Variable table

Category	Name	Assigned to	Data type	Initial value	Com-ment
VAR	ECAT_GetSlaveStatus1		ECAT_GetSlaveStatus		

VAR	ECAT_GetSlaveStatus_Enable		BOOL		
VAR	ECAT_GetSlaveStatus_Vaild		BOOL		
VAR	ECAT_GetSlaveStatus_Error		BOOL		
VAR	ECAT_GetSlaveStatus_ErrorID		WORD		
VAR	ECAT_GetSlaveStatus_CurrentState		ECAT_State_Type		
VAR	ECAT_GetSlaveStatus_Emergency		UINT		
VAR	ECAT_GetSlaveStatus_OP		BOOL		

LD:



ST:

```
ECAT_GetSlaveStatus1(Enable:=ECAT_GetSlaveStatus_Enable ,
    NodeID:=1001 ,
    Valid=>ECAT_GetSlaveStatus_Vaild ,
    Error=>ECAT_GetSlaveStatus_Error ,
    ErrorID=>ECAT_GetSlaveStatus_ErrorID ,
    CurrentState=>ECAT_GetSlaveStatus_CurrentState ,
    Emergency=>ECAT_GetSlaveStatus_Emergency
    );

IF ECAT_GetSlaveStatus_CurrentState=OP THEN
    ECAT_GetSlaveStatus_OP:=TRUE;
ELSE
    ECAT_GetSlaveStatus_OP:=FALSE;
END_IF;
```

- **Program description**

Step 1: Set the value of the input variable NodeID to 1001 to read the status of the 1st EtherCAT slave.

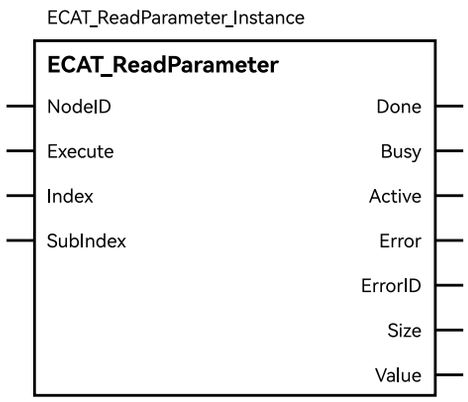
Step 2: When the input variable ECAT_GetSlaveStatus_Enable is TRUE and ECAT_GetSlaveStatus_Vaild becomes TRUE, the current state of the slave can be viewed by ECAT_GetSlaveStatus_CurrentState. When the value of ECAT_GetSlaveStatus_CurrentState is OP, it means that the slave status is in OP (operating) state, which indicates that the establishment of the communication connection between master and slave is completed, that is, the communication between master and slave is normal.

Step 4: After the input variable ECAT_GetSlaveStatus_Enable is TRUE, ECAT_GetSlaveStatus_Vaild becomes TRUE, and the value of ECAT_GetSlaveStatus_Emergency is not 0, the error code of the emergency message sent by the slave can be viewed by ECAT_ GetSlaveStatus_Emergency.

Step 5: When the input variable ECAT_GetSlaveStatus_Enable is TRUE, and ECAT_GetSlaveStatus_Error changes from False to TRUE, the error code can be viewed by ECAT_GetSlaveStatus_ErrorID.

4.3 ECAT_ReadParameter

Read EtherCAT slave parameter. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ECAT_ReadParameter	Read EtherCAT slave parameters	FB		ECAT_ReadParameter_Instance(NodeID:= parameter, Execute:= parameter, Index:= parameter, SubIndex:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Error=> parameter, ErrorID=> parameter, Size=> parameter, Value=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
NodeID	Slave EtherCAT address	UINT	Refer to communication instruction specifications (0)	Required field	Specify the address of the EtherCAT slave For example, the address of the 1st slave station is 1001, the address of the 2nd slave station is 1002, and so on.
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
Index	Index of the parameters	UINT	0~65535	0	Set the index of the parameter to be read
SubIndex	Subindex of parameters	USINT	0~255	0	Set the sub-index of the parameter to be read

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	Instruction in control	BOOL	TRUE / FALSE	TRUE when the instruction is in control
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Size	Data type of parameters	USINT	1~4	The type of the parameter to be read. 1: Byte (1 byte) 2: Word (2 bytes) 4: DWord (4 bytes)
Value	The value of the parameter that has been read	UDINT		The value of the parameter that has been read

*1 Refer to "instruction error code description" for the meaning of the output error code value.

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Active	The instruction controls the read parameters.	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

• Basic function description

This instruction is used to read the parameter values of the EtherCAT slave. The parameter to be read is specified by Index and SubIndex. The Index and SubIndex of the slave parameters can be obtained from the slave-related information.

This instruction reads the slave's parameters according to the value set by the input variable when Execute changes from FALSE to TRUE.

When this instruction is executed, the value of the specified slave parameter is read into the output variable Value. The data type of the output variable Value is UDINT. When the data type of Value and the data type of the read slave parameter do not match, the data type of the Value variable can be converted to the data type of the slave parameter by the data type conversion instruction. If the data type of the read slave parameter is DINT, the value of Value can be converted to the variable of DINT type by UDINT_TO_DINT instruction; if the data type of the read slave parameter is INT, the value of Value can be converted to the variable of INT type by UDINT_TO_INT instruction.

• Instruction completion timing

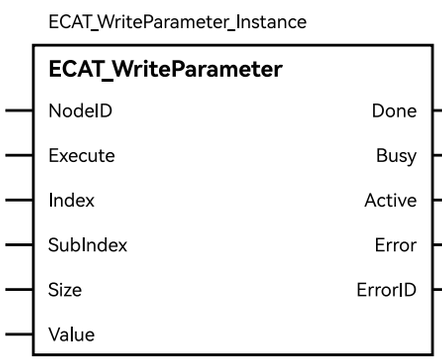
When the value of the slave's parameter is read, the instruction is completed and the Done bit changes from FALSE to TRUE.

• Re-execute the instruction

When an instruction is executed and Execute changes from FALSE to TRUE again, the instruction can be re-executed. When the instruction is being executed and Execute changes from FALSE to TRUE again, there is no effect on the execution of the instruction, and the instruction still executes the instruction in accordance with the input variables that have not been executed to completion.

4.4 ECAT_WriteParameter

Set the parameters in the EtherCAT slave. Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
ECAT_WritePa-rameter	Set the parame-ters in the Ether-CAT slave	FB		ECAT_WriteParameter_Instance(NodeID:= parameter, Execute:= parameter, Index:= parameter, SubIndex:= parameter, Size:= parameter, Value:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Error=> parameter, ErrorID=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
NodeID	Slave EtherCAT address	UINT	Refer to communi-cation instruction specifications	Required field	Specify the address of the EtherCAT slave For example, the address of the 1st slave station is 1001, the address of the 2nd slave station is 1002, and so on.
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
Index	Index of the parameters	UINT	0~65535	0	Set the index of the parameter to be set
SubIndex	Subindex of parameters	USINT	0~255	0	Set the sub-index of the parameter to be set.
Size	Data type of parameters	USINT	1~4	Required field	Set the type of parameter to be set. 1: Byte (1 byte) 2: Word (2 bytes) 4: DWord (4 bytes)
Value	Set value	UDINT	0 ~ 4294967295	0	Set value

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	Instruction in control	BOOL	TRUE / FALSE	TRUE when the instruction is in control
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

*1 Refer to "instruction error code description" for the meaning of the output error code value.

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE

Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Active	The instruction controls the write parameters.	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

• Basic function description

This instruction is used to set the parameter values of the EtherCAT slave, which are specified via Index and SubIndex. The Index and SubIndex of the slave parameters can be obtained from the slave-related information. The value of the parameter set is the value of the input variable Value.

This instruction sets the slave's parameters according to the value set in the input variable when Execute changes from FALSE to TRUE.

When this instruction is executed, the value in the input variable Value is set to the specified slave parameter. The data type of the input variable Value is UDINT. When the data type of Value and the data type of the slave parameter do not match, the data type of the Value variable can be converted to the data type of the slave parameter by the data type conversion instruction. If the data type of the slave parameter is DINT, users can use the UDINT_TO_DINT instruction to convert the value of Value to a variable of DINT type; if the data type of the slave parameter is INT, the value of Value can be converted to a variable of INT type by using the UDINT_TO_INT instruction.

• Instruction completion timing

When the instruction to write the parameter value is sent, the instruction is completed and the Done bit changes from FALSE to TRUE.

• Re-execute the instruction

When an instruction is executed and Execute changes from FALSE to TRUE again, the instruction can be re-executed. When the instruction is being executed and Execute changes from FALSE to TRUE again, there is no effect on the execution of the instruction, and the instruction still executes the instruction in accordance with the input variables that have not been executed to completion.

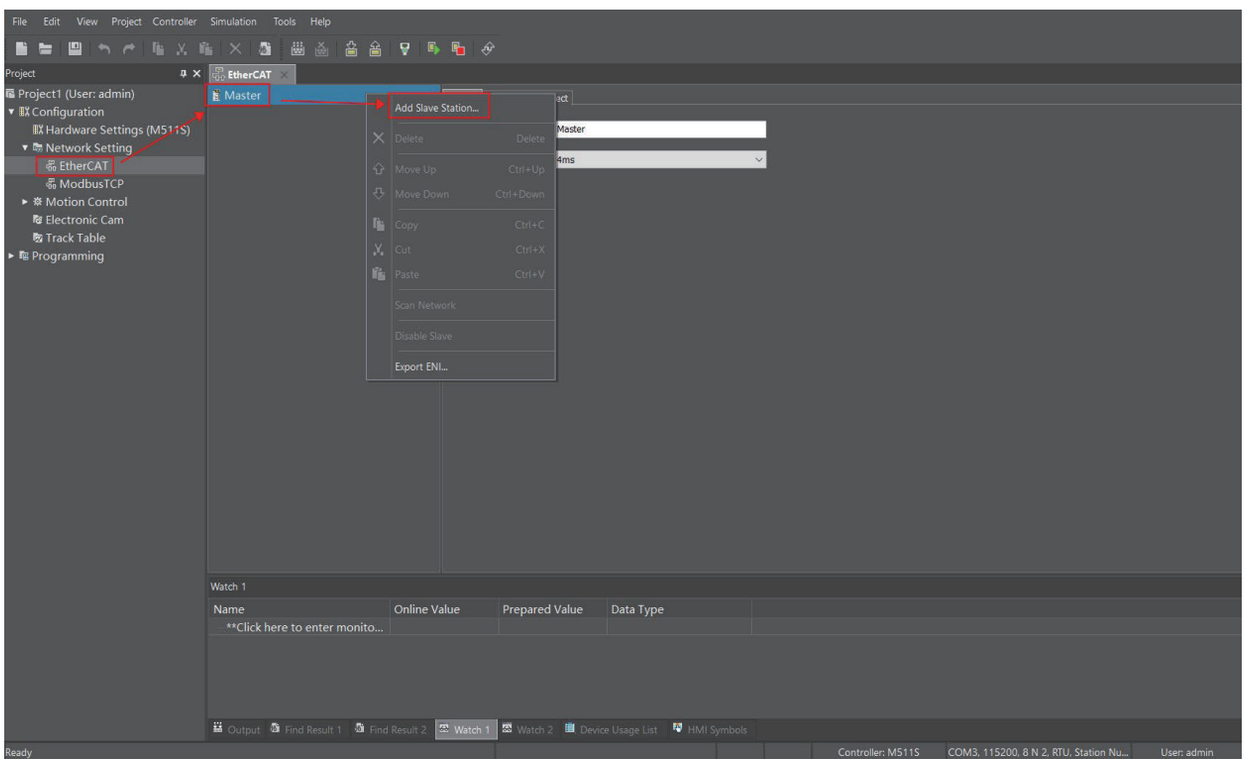
4.5 Example of ECAT parameter reading and writing

- **Target demand**

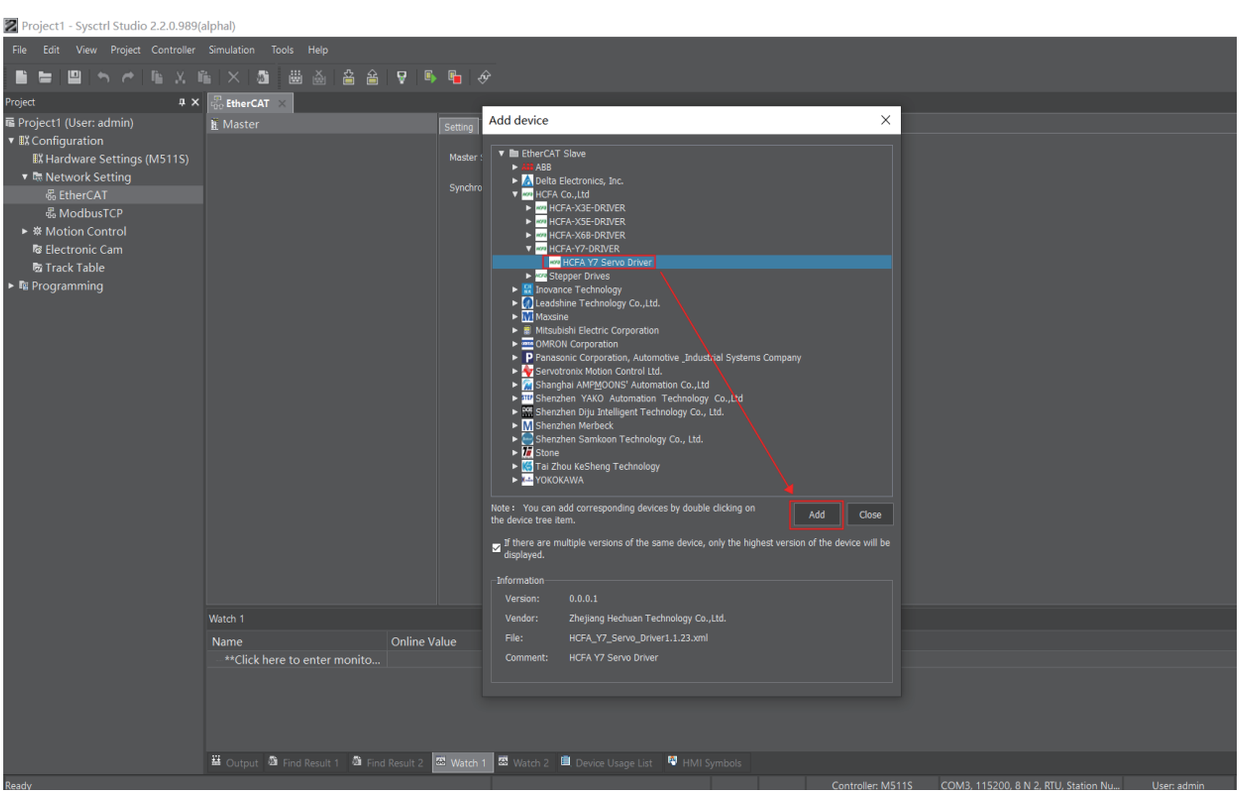
Read and write HCFA Y7 series servo drive parameter 16#607D:02 (index: subindex) by ECAT_ReadParameter instruction and ECAT_WriteParameter instruction.

- **Software configuration**

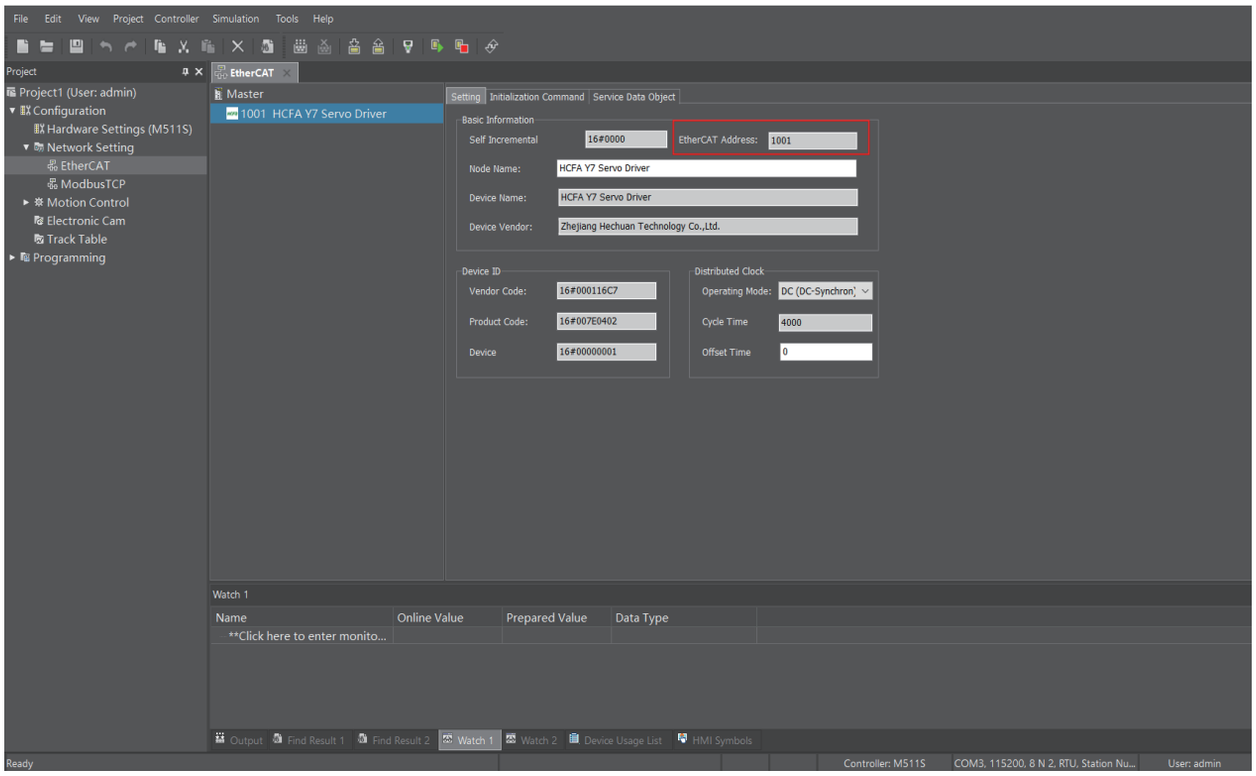
Step 1: Click on "Network settings", and then double-click on "EtherCAT", right-click on Master, and click on Add Slave.



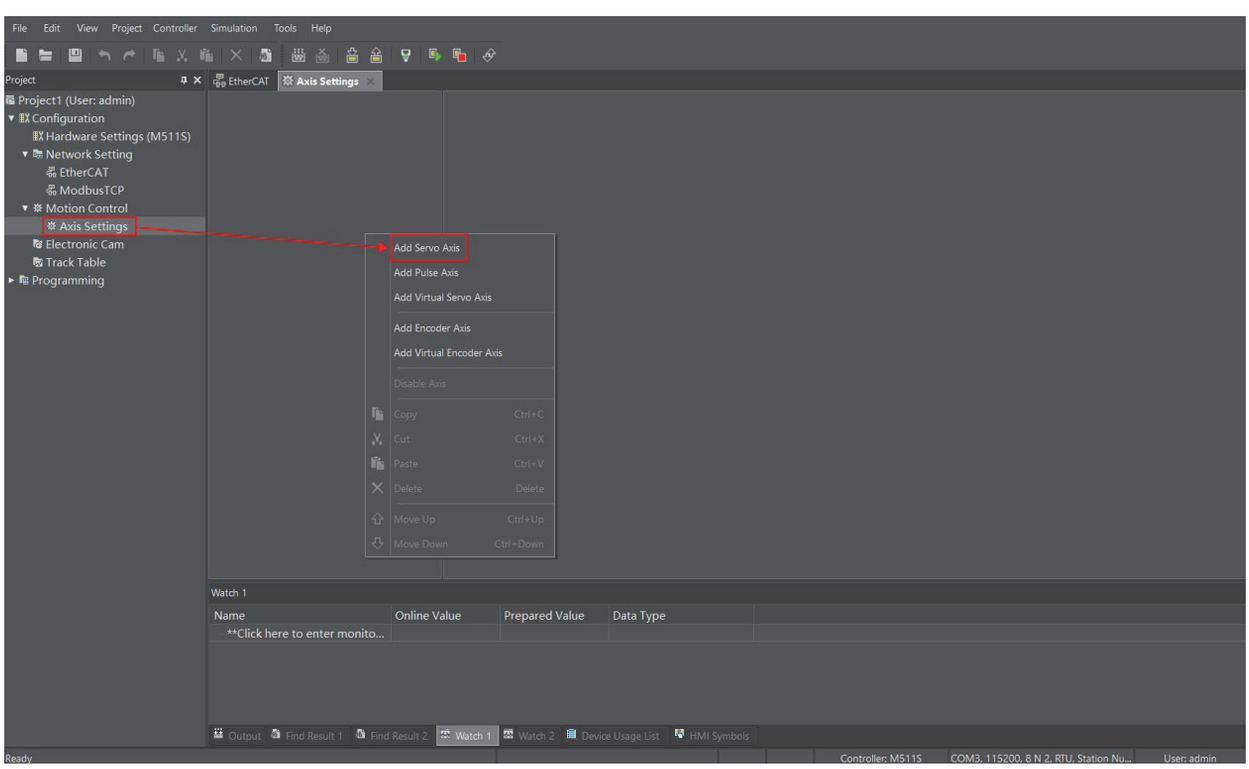
Step 2: Double-click on the HCFA Y7 servo drive to add a slave. Or select HCFA Y7 servo drive and then click on the "Add" button in the red box below to add a slave.



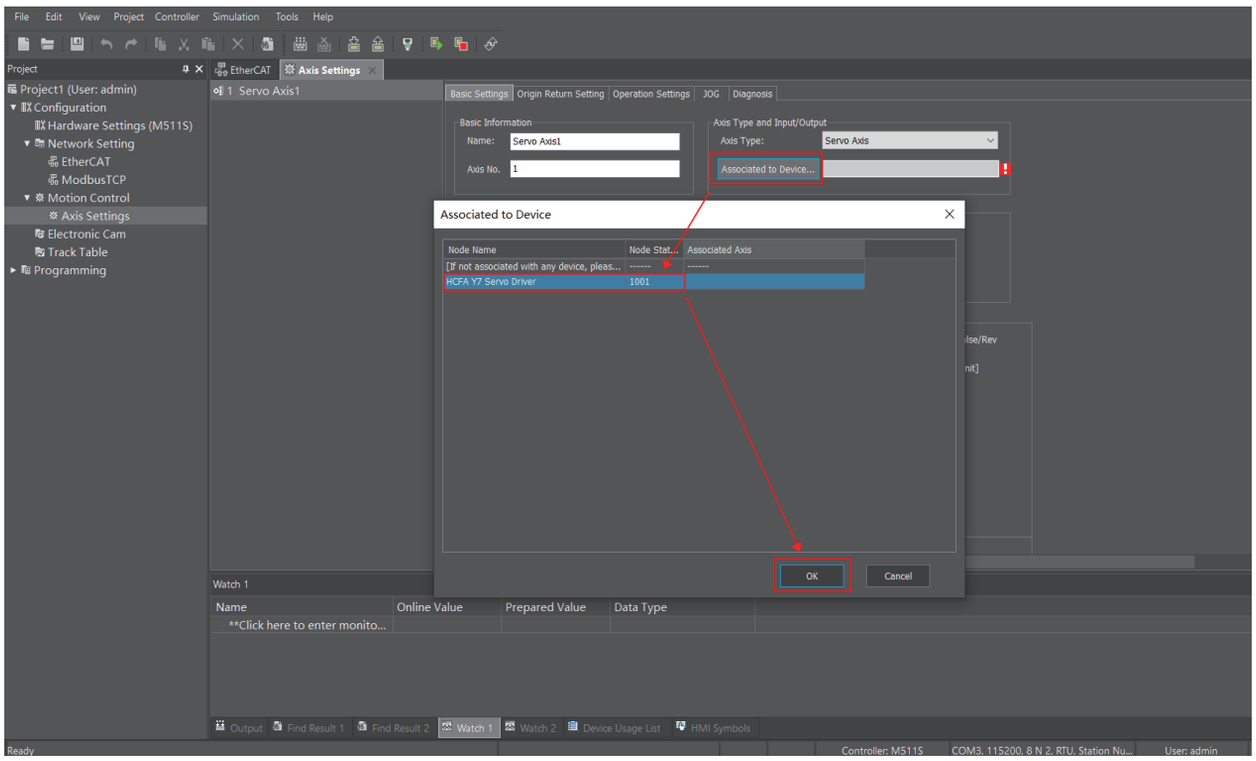
Step 3: The EtherCAT address of the 1st EtherCAT slave is 1001, the EtherCAT address of the 2nd EtherCAT slave is 1002 and so on.



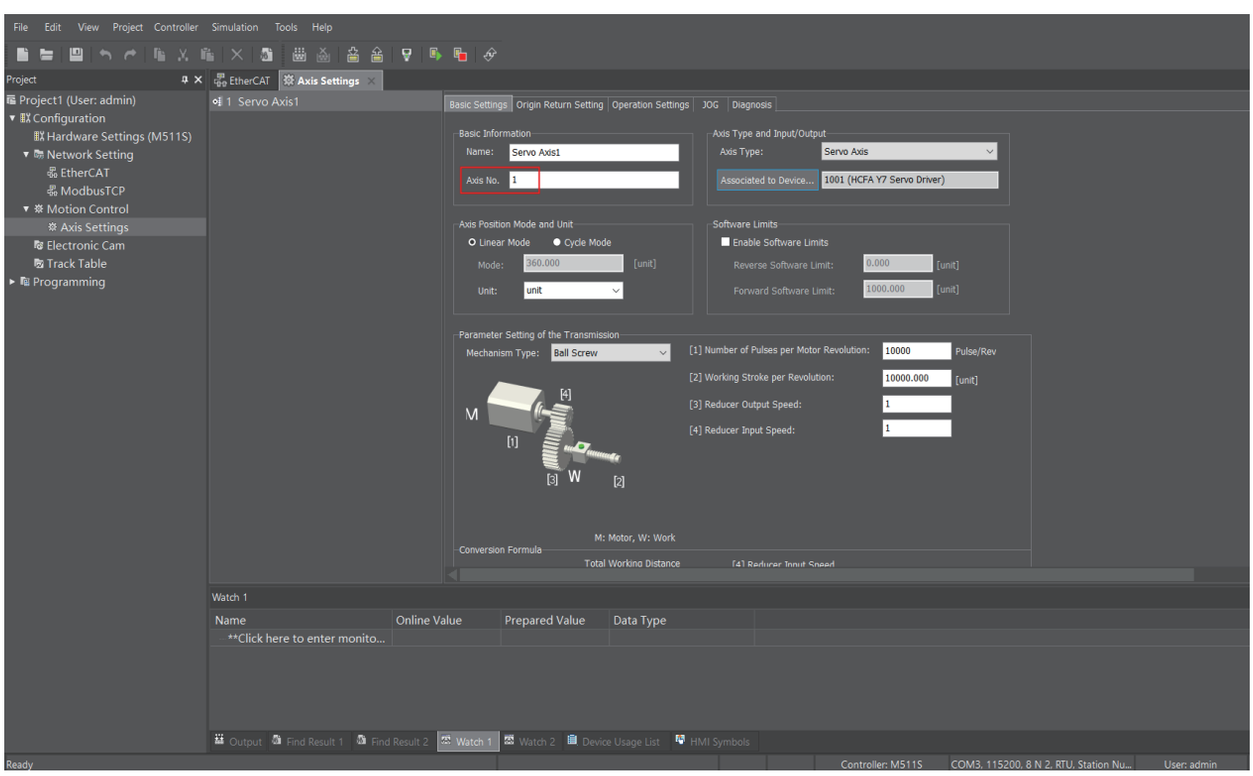
Step 4: Double-click on Axis settings - right-click on the middle blank space - click on Add servo axis



Step 5: Click on " Link to device " -> Click on " HCFA Y7 servo drive " -> Click on the " OK " button. With this step, a one-to-one relationship is established between the axis and the EtherCAT slave.



Step 6: Set the axis number at the red box in the figure below.



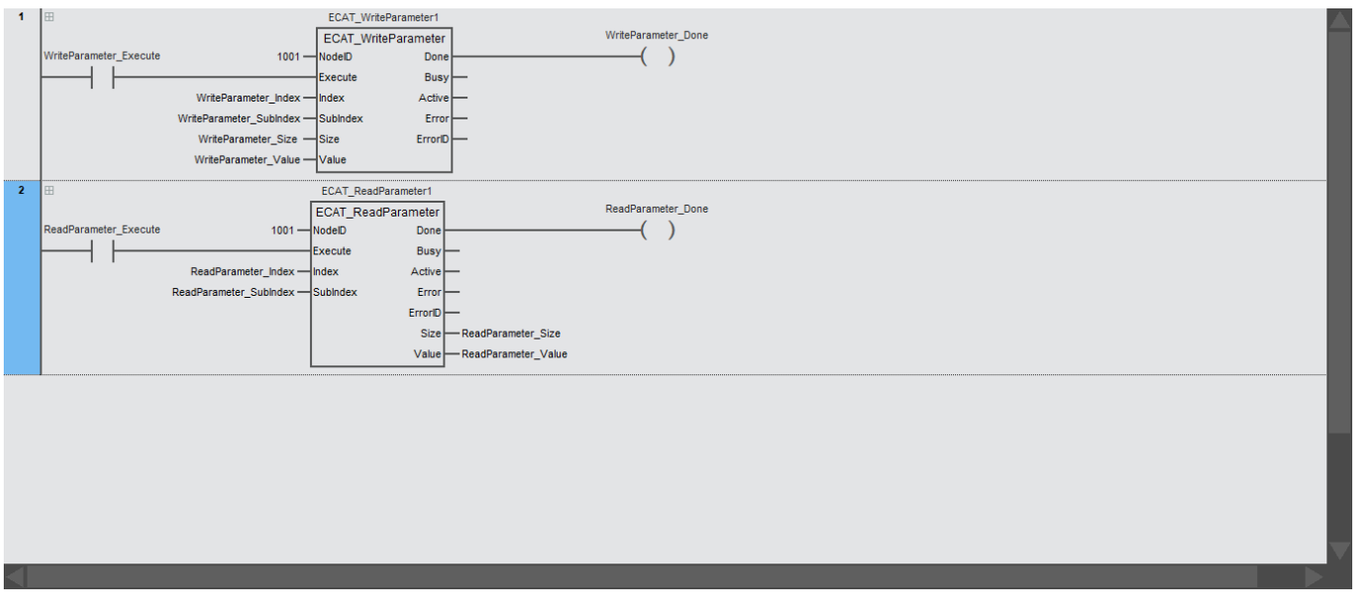
- **Instruction configuration**

Variable table

Category	Name	Assigned to	Data type	Initial value	Comment
VAR	ECAT_WriteParameter1		ECAT _WriteParameter		
VAR	ECAT _ReadParameter1		ECAT _ReadParameter		
VAR	WriteParameter_Execute		BOOL		
VAR	ReadParameter_Execute		BOOL		

VAR	WriteParameter_Done		BOOL		
VAR	ReadParameter_Done		BOOL		
VAR	WriteParameter_Value		UDINT		
VAR	ReadParameter _Value		UDINT		
VAR	WriteParameter_Index		UINT	16#607D	
VAR	WriteParameter_SubIndex		UINT	2	
VAR	ReadParameter _Index		USINT	16#607D	
VAR	ReadParameter_SubIndex		USINT	2	
VAR	WriteParameter_Size		USINT		
VAR	ReadParameter_Size		USINT		

LD:



ST:

```
ECAT_WriteParameter1
(NodeID:=1001 ,
    Execute:=WriteParameter_Execute ,
    Index:=WriteParameter_Index ,
    SubIndex:=WriteParameter_SubIndex ,
    Size:=WriteParameter_Size ,
    Value:=WriteParameter_Value,
    Done=>WriteParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=>
    );

ECAT_ReadParameter1
(NodeID:=1001 ,
    Execute:=ReadParameter_Execute ,
    Index:=ReadParameter_Index ,
    SubIndex:=ReadParameter_SubIndex ,
    Done=>ReadParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    Size=>ReadParameter_Size ,
    Value=>ReadParameter_Value
    );
```

- **Program description**

Step 1: Set the value of the input variable NodeID to 1001 to write the parameter for the 1st EtherCAT slave connected to the master.

Step 2: Set the value of the input variable WriteParameter_Index to 16#607D, the value of the input variable WriteParameter_SubIndex to 2, the value of the input variable WriteParameter_Size to 4, and the value of the input variable WriteParameter_Value to 2147483647.

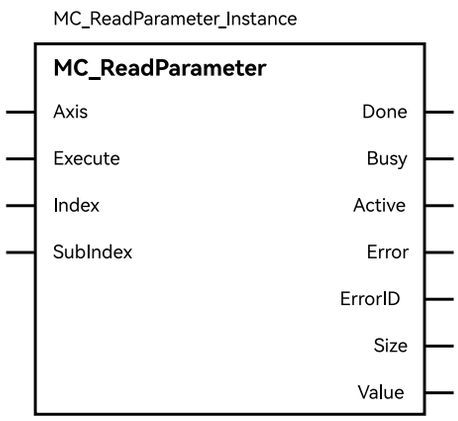
Step 3: The input variable WriteParameter_Execute changes from FALSE to TRUE, which triggers the execution of the ECAT_WriteParameter instruction. WriteParameter_Done bit changes to TRUE, which indicates that the parameter write is completed.

Step 4: Set the value of the input variable ReadParameter_Index to 16#607D and the value of the input variable ReadParameter_SubIndex to 2.

Step 5: After the input variable ReadParameter_Execute changes from FALSE to TRUE, it triggers the execution of the ECAT_ReadParameter instruction. ReadParameter_Done bit changes to TRUE, which indicates that the parameter reading is completed. After the reading is completed, the values of ReadParameter_Size and ReadParameter_Value are 4 and 2147483647 respectively, and the value of ReadParameter_Value is the value of the parameter that has been read.

4.6 MC_ ReadParameter

The parameter values of the servo axes are read via SDO (Service Data Object). Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
MC_ReadPar-ameter	Read EtherCAT slave parameters	FB		MC_ReadParameter_Instance(Axis:= parameter, Execute:= parameter, Index:= parameter, SubIndex:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Error=> parameter, ErrorID=> parameter, Size=> parameter, Value=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Axis	Axis number	USINT	Refer to communi-cation instruction specifications	Required field	Specify the axis number of the control axis
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
Index	Index of the parameters	UINT	0~65535	0	Set the index of the parameter to be read
SubIndex	Subindex of parameters	USINT	0~255	0	Set the sub-index of the parameter to be read

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	Instruction in control	BOOL	TRUE / FALSE	TRUE when the instruction is in control
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1
Size	Data type of parameters	USINT		The type of the parameter to be read. 1: Byte (1 byte) 2: Word (2 bytes) 4: DWord (4 bytes)
Value	The value of the parameter that has been read	UDINT		The value of the parameter that has been read。

*1 Refer to "instruction error code description" for the meaning of the output error code value.

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE
Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Active	The instruction controls the read parameters.	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

• Basic function description

This instruction is used to read the parameter value of the specified servo axis, the parameter to be read is specified by Index and SubIndex. The Index and SubIndex of the slave parameters can be obtained from the slave-related information.

This instruction reads the slave's parameters according to the value set by the input variable when Execute changes from FALSE to TRUE.

When this instruction is executed, the value of the specified slave parameter is read into the output variable Value. The data type of the output variable Value is UDINT. When the data type of Value and the data type of the read slave parameter do not match, the data type of the Value variable can be converted to the data type of the slave parameter by the data type conversion instruction. If the data type of the read slave parameter is DINT, the value of Value can be converted to the variable of DINT type by UDINT_TO_DINT instruction; if the data type of the read slave parameter is INT, the value of Value can be converted to the variable of INT type by UDINT_TO_INT instruction.

• Instruction completion timing

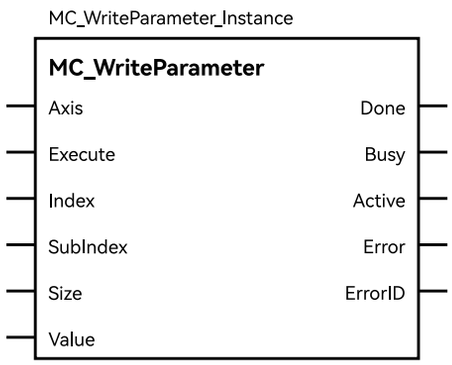
When the value of the slave's parameter is read, the instruction is completed and the Done bit changes from FALSE to TRUE.

• Re-execute the instruction

When an instruction is executed and Execute changes from FALSE to TRUE again, the instruction can be re-executed. When the instruction is being executed and Execute changes from FALSE to TRUE again, there is no effect on the execution of the instruction, and the instruction still executes the instruction in accordance with the input variables that have not been executed to completion.

4.7 MC_ WriteParameter

Set the parameter values of the servo axes via SDO (Service Data Object). Library: Communications

Instruction	Name	FB/FUN	Graphic expression	ST expression
MC_WritePa-rameter	Set the parame-ters in the Ether-CAT slave	FB		MC_WriteParameter_Instance(Axis:= parameter, Execute:= parameter, Index:= parameter, SubIndex:= parameter, Size:= parameter, Value:= parameter, Done=> parameter, Busy=> parameter, Active=> parameter, Error=> parameter, ErrorID=> parameter);

◆ Input variable

Name	Meaning	Data type	Valid range	Default value	Description
Axis	Axis number	USINT	Refer to communi-cation instruction specifications	Required field	Specify the axis number of the control axis
Execute	Effective	BOOL	TRUE / FALSE	FALSE	Execute when the rising edge of this parameter is detected
Index	Index of the parameters	UINT	0~65535	0	Set the index of the parameter to be set
SubIndex	Subindex of parameters	USINT	0~255	0	Set the sub-index of the parameter to be set.
Size	Data type of parameters	USINT	1~4	Required field	Set the type of parameter to be set. 1: Byte (1 byte) 2: Word (2 bytes) 4: DWord (4 bytes)
Value	Set value	UDINT	0 ~ 4294967295	0	Set value

◆ Output variable

Name	Meaning	Data type	Valid range	Description
Done	Completed	BOOL	TRUE / FALSE	TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE / FALSE	TRUE when the instruction is being executed normally
Active	Instruction in control	BOOL	TRUE / FALSE	TRUE when the instruction is in control
Error	Error	BOOL	TRUE / FALSE	TRUE when there is an error
ErrorID	Error code	WORD	0~65535	Output an error code when an instruction execution exception occurs. *1

*1 Refer to "instruction error code description" for the meaning of the output error code value.

◆ Output variable refreshing timing

Name	Whether or not to become TRUE	Whether or not to become FALSE

Done	TRUE when the instruction is completed	When Done is TRUE and Execute changes from TRUE to FALSE When the instruction is executed and the Execute is FALSE, Done becomes TRUE and after one period, it becomes FALSE
Busy	When the rising edge of Execute is detected	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Active	The instruction controls the write parameters.	When Done changes from FALSE to TRUE When Error changes from FALSE to TRUE
Error	The input variable value of the instruction is not within the allowed range, does not meet the execution conditions of the instruction, or encounters an exception during the instruction execution process.	When Execute changes from TRUE to FALSE The Instruction has been executed and the Execute has become FALSE. When an exception is encountered during the execution, the Error becomes TRUE. After one period, it becomes FALSE.

◆ Function description

• Basic function description

This instruction is used to set the parameter value of the specified servo axis. The parameters of the slave are specified by Index and SubIndex. The Index and SubIndex of the slave parameters can be obtained from the slave-related information. The value of the parameter set is the value of the input variable Value.

This instruction sets the slave's parameters according to the value set by the input variable when Execute changes from FALSE to TRUE.

When this instruction is executed, the value in the input variable Value is set to the specified slave parameter. The data type of the input variable Value is UDINT. When the data type of Value and the data type of the slave parameter do not match, the data type of the Value variable can be converted to the data type of the slave parameter by the data type conversion instruction. If the data type of the slave parameter is DINT, users can use the UDINT_TO_DINT instruction to convert the value of Value to a variable of DINT type; if the data type of the slave parameter is INT, the value of Value can be converted to a variable of INT type by using the UDINT_TO_INT instruction.

• Instruction completion timing

When the instruction to write the parameter value is sent, the instruction is completed and the Done bit changes from FALSE to TRUE.

• Re-execute the instruction

When an instruction is executed and Execute changes from FALSE to TRUE again, the instruction can be re-executed. When the instruction is being executed and Execute changes from FALSE to TRUE again, there is no effect on the execution of the instruction, and the instruction still executes the instruction in accordance with the input variables that have not been executed to completion.

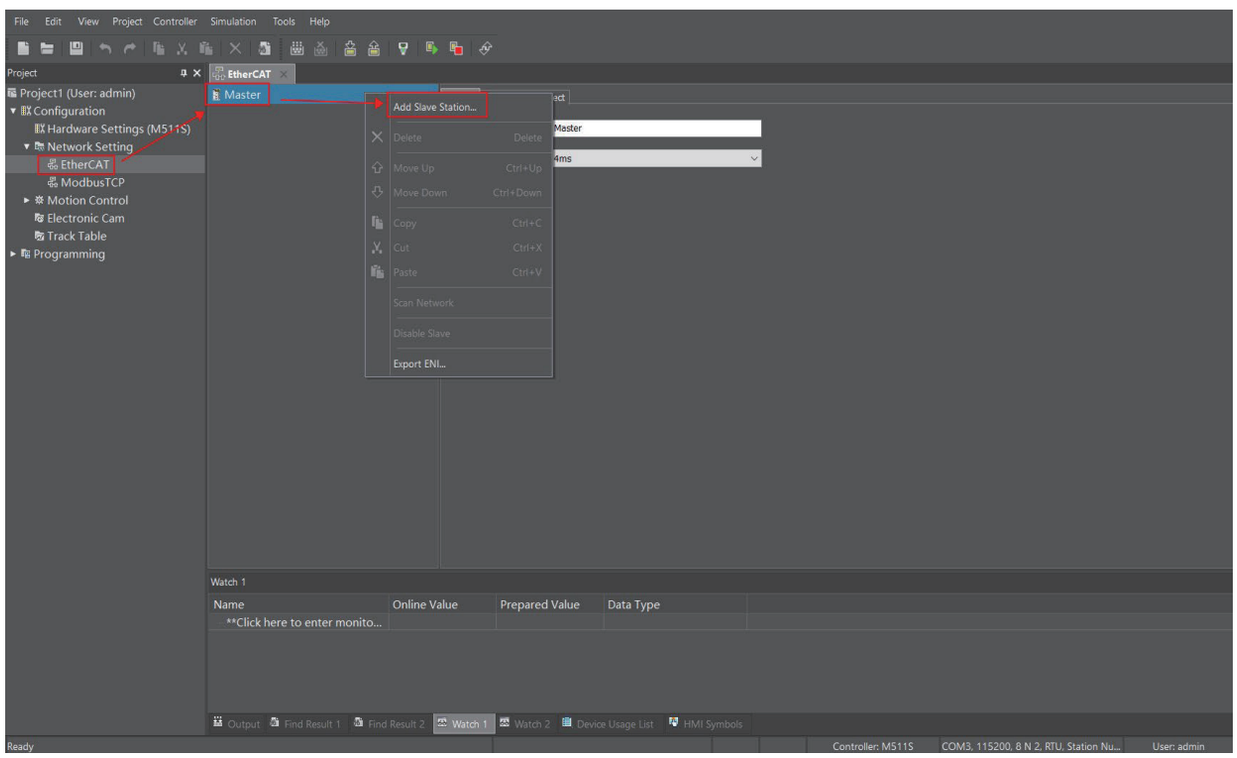
4.8 Example of MC parameter reading and writing

- **Target demand**

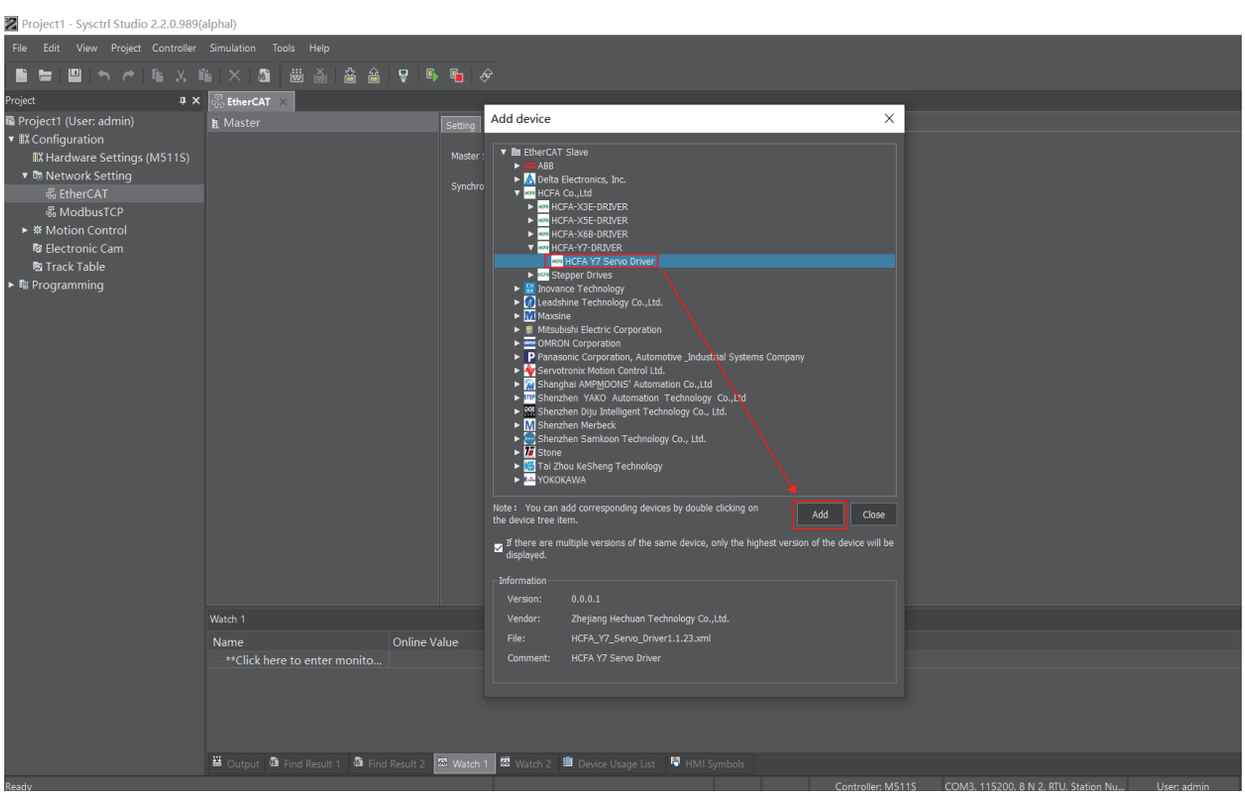
The HCFA Y7 series servo drive parameter 16#607D:02 (index: subindex) is read and written by the MC_ ReadParameter instruction and MC_ WriteParameter instruction.

- **Software configuration**

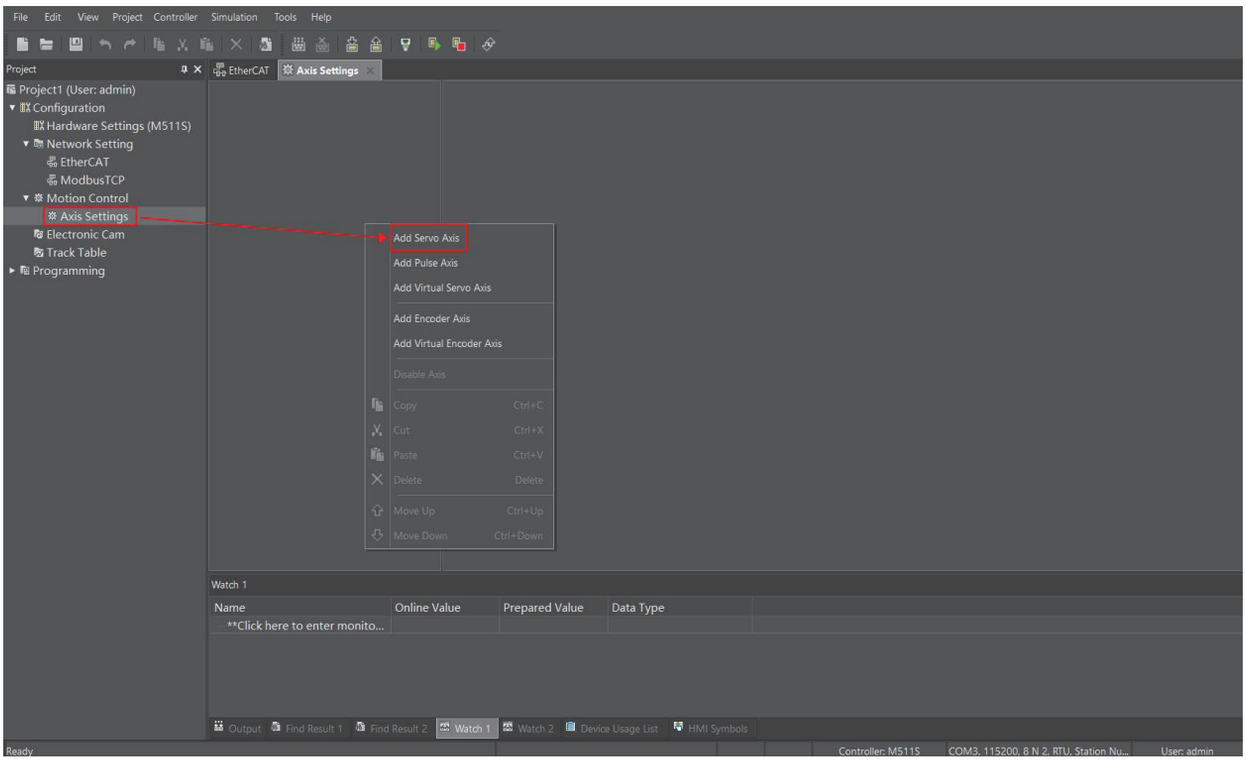
Step 1: Click on "Network settings", and then double-click on "EtherCAT", right-click on Master, and click on Add Slave.



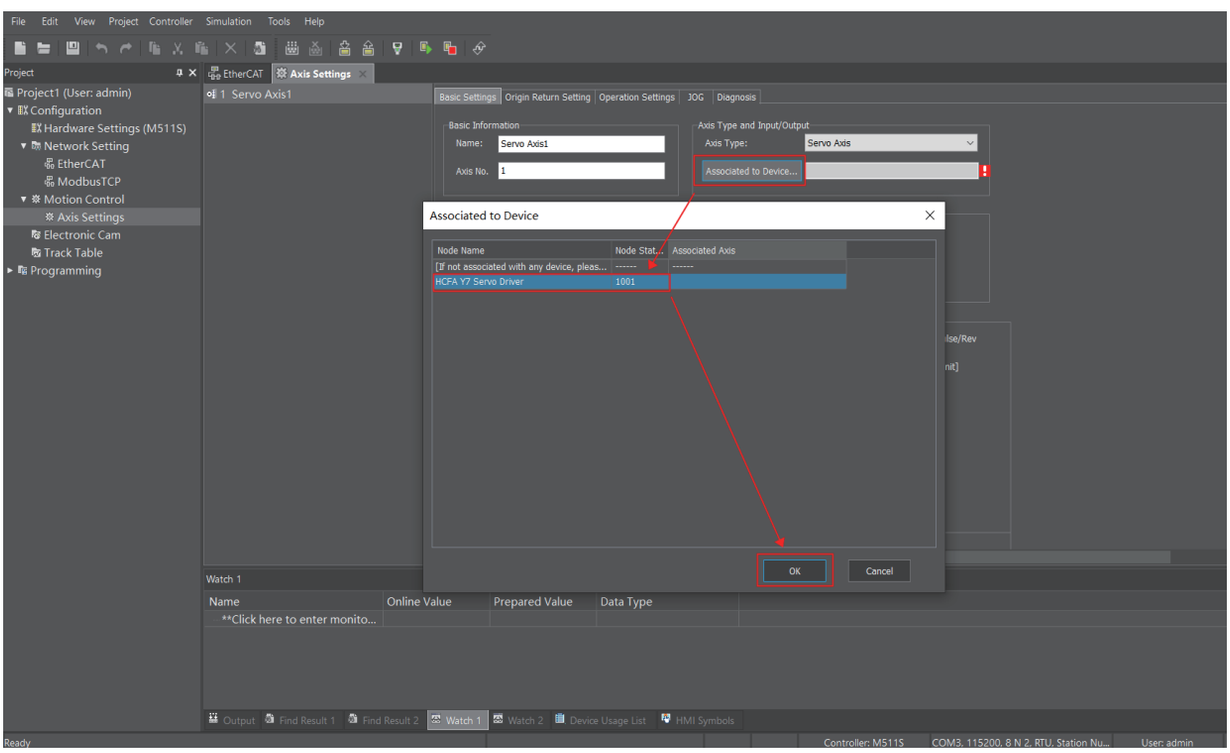
Step 2: Double-click on the HCFA Y7 servo drive to add a slave. Or select HCFA Y7 servo drive and then click on the "Add" button in the red box below to add a slave.



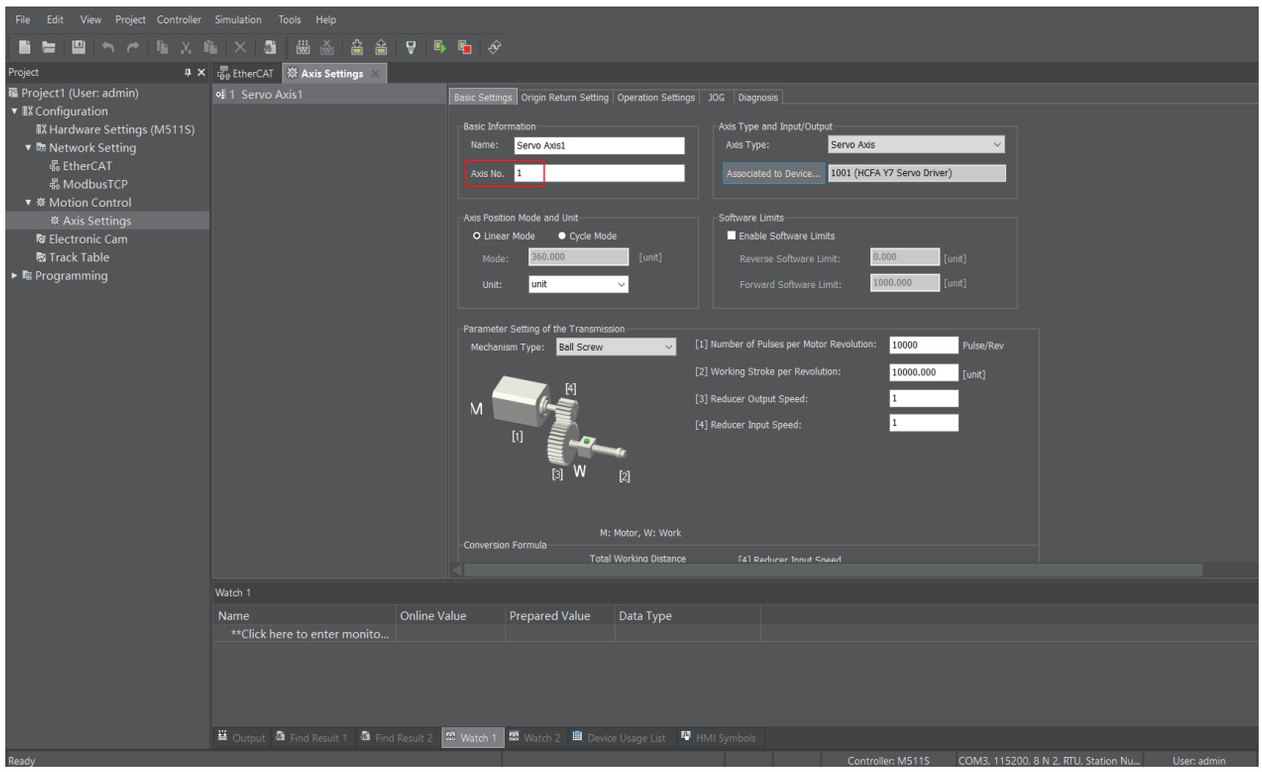
Step 3: Double-click on Axis settings - right-click on the middle blank space - click on Add servo axis



Step 4: Click on " Link to device " -> Click on " HCFA Y7 servo drive " -> Click on the " OK " button. With this step, a one-to-one relationship is established between the axis and the EtherCAT slave.



Step 5: Set the axis number at the red box in the figure below.

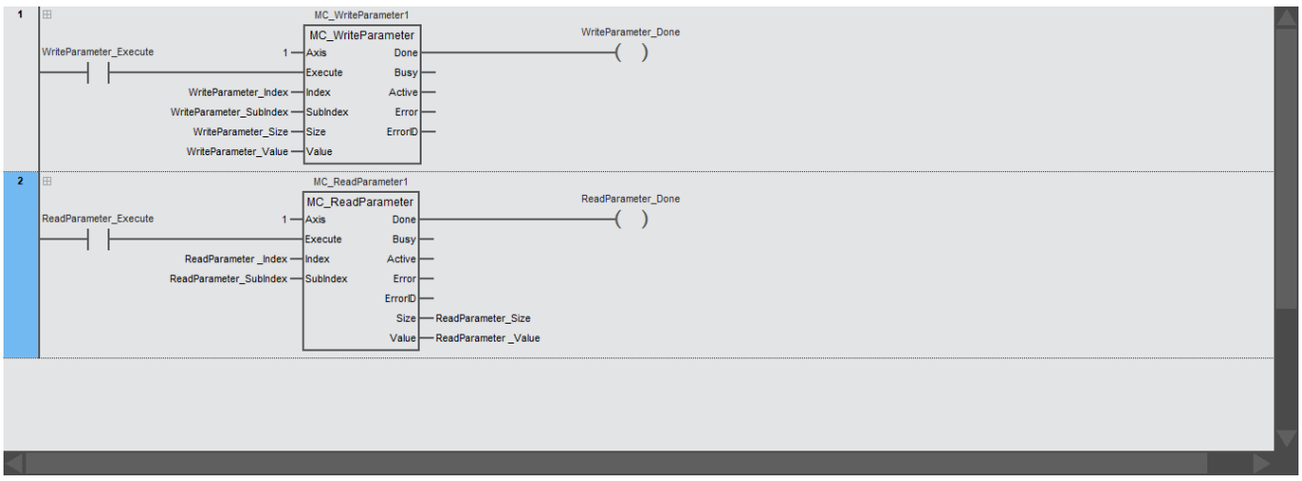


- **Instruction configuration**

Variable table

Category	Name	Assigned to	Data type	Initial value	Comment
VAR	MC_WriteParameter1		MC_WriteParameter		
VAR	MC_ReadParameter1		MC_ReadParameter		
VAR	WriteParameter_Execute		BOOL		
VAR	ReadParameter_Execute		BOOL		
VAR	WriteParameter_Done		BOOL		
VAR	ReadParameter_Done		BOOL		
VAR	WriteParameter_Value		UDINT		
VAR	ReadParameter _Value		UDINT		
VAR	WriteParameter_Index		UINT	16#607D	
VAR	WriteParameter_SubIndex		UINT	2	
VAR	ReadParameter _Index		USINT	16#607D	
VAR	ReadParameter_SubIndex		USINT	2	
VAR	WriteParameter_Size		USINT		
VAR	ReadParameter_Size		USINT		

LD:



ST:

```
MC_WriteParametel(Axis:=1 ,
    Execute:=WriteParameter_Execute ,
    Index:=WriteParameter_Index ,
    SubIndex:=WriteParameter_SubIndex ,
    Size:=WriteParameter_Size ,
    Value:=WriteParameter_Value ,
    Done=>WriteParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=>
    );

MC_ReadParameter1(Axis:=1 ,
    Execute:=ReadParameter_Execute ,
    Index:=ReadParameter_Index ,
    SubIndex:=ReadParameter_SubIndex ,
    Done=>ReadParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    Size=>ReadParameter_Size ,
    Value=>ReadParameter_Value
    );
```

- **Program description**

Step 1: Set the value of the input variable Axis to 1 to write parameter for the axis with the axis number 1.

Step 2: Set the value of the input variable WriteParameter_Index to 16#607D, the value of the input variable WriteParameter_SubIndex to 2, the value of the input variable WriteParameter_Size to 4, and the input variable WriteParameter_Value written to 2147483647.

Step 3: After the input variable WriteParameter_Execute changes from FALSE to TRUE, it triggers the execution of the WriteParameter instruction. When the WriteParameter_Done bit changes to TRUE, it indicates that parameter writing is completed.

Step 4: Set the value of the input variable ReadParameter_Index to 16#607D and the value of the input variable ReadParameter_SubIndex to 2.

Step 5: After the input variable ReadParameter_Execute changes from FALSE to TRUE, it triggers the execution of the ReadParameter instruction. When the ReadParameter_Done bit changes to TRUE, it indicates that the parameter reading is completed. After the reading is completed, the ReadParameter_Size and the ReadParameter_Size and ReadParameter_Value are 4 and 2147483647 respectively, and the value of ReadParameter_Value is the value of the parameter that has been read.

4.9 Communication instruction specifications

Model name	Data exchange channel number range (Linknum)	Socket number range（Socket-Num）	CANopen slave station number range (NodeID)	EtherCAT slave station number range (NodeID)	Axis number range
HCM511S-32MT4-D	1~4	1~4	1~16	1001~1008	1~16
HCM501S-16MT4-D	Unsupported	Unsupported	Unsupported	1001~1008	1~16
HCM301-16MT4-D	Unsupported	Unsupported	Unsupported	Unsupported	1~16
HCM302-16MT4-D	Unsupported	Unsupported	Unsupported	Unsupported	1~16
HCM310-16MT4-D	1~16	1~8	Unsupported	Unsupported	1~16
HCM311-16MT4-D	1~16	1~8	Unsupported	Unsupported	1~16
HCM312-32MT6-D	1~16	1~8	1~32	Unsupported	1~16
HCM511-32MT4-D	1~16	1~8	1~32	1001~1016	1~64
HCM512-32MT4-D	1~16	1~8	1~32	1001~1032	1~64
HCM513-32MT4-D	1~16	1~8	1~32	1001~1064	1~64
HCM514-32MT4-D	1~16	1~8	1~32	1001~1128	1~128

Chapter 5 Related to Modbus and Modbus TCP communication

5.1 Supported Modbus function codes

The function codes supported by the Ethernet, RS485, RS232, and USB communication ports of the M-Series controller are shown in the table below:

Category	Function code	Description	Whether broadcastable or not (Y/N: Yes/No)	Maximum read/write value	Operable device
Bit device	0x01	Meaning: Read the value of the bit device. The values of the M-Series controller bit devices can all be read using the 01 function code	N	256	%IX,%QX
	0x02	Meaning: Read the value of the input bit device. The values of the M-Series controller bit devices can all be read using the 02 function code	N	256	%IX,%QX
	0x05	Write the value of a single bit device	Y	1	%QX
	0x0F	Write the value of multiple bit devices	Y	256	%QX
Word device	0x03	Reads the value of a single device or multiple word devices	N	100	%MW,%QW,%IW
	0x04	Meaning: Reads the value of a single or multiple input word devices The values of all M-Series controller word devices can be read using the 04 function code	N	100	%MW,%QW,%IW
	0x06	Write the value of a single word device	Y	1	%MW,%QW
	0x10	Write values of multiple word devices	Y	100	%MW,%QW
	0x17	Read or write the value of a single device or multiple word devices	Y	100	%MW,%QW, % IW (read-only)

5.2 Modbus exception response codes

The exception response codes supported by the Ethernet, RS485, RS232, and USB communication ports of the M-Series controller are shown in the table below:

Exception response code	Meaning	Handling measure
1	The slave does not support the function codes specified by the master	Specify the function codes supported by the slave
2	The slave does not support the Modbus address specified by the master	Specify the Modbus address supported by the slave
3	Read or write the specified data length is out of range	When the controller is a slave, the maximum length that can be read or written at one time is 100 WORDs when operating a word (WORD) device, and the maximum length that can be read or written at one time is 256 bits when operating a bit (bit) device. If the above specifications are exceeded, the controller replies with the exception response code.
7	Master and slave calculate different check digits	Make sure the baud rate and communication format of master and slave are the same Check the bus attachment for interference source Check whether the bus is shielded Check whether both master and slave are grounded

5.3 List of Modbus addresses for the device

The following table shows the Modbus addresses supported by M series controllers, which can be accessed through Ethernet, RS232, and RS485 communication, or through the corresponding function codes, including 03, 04, 06, 16#10, 16#17, 01, 02, 05, 16#0F. If the user needs the HMI to read or write the bit device of the controller, the bit device of the output device can be used as the intermediate bit device, e.g., %QX50.0~%QX127.7 can be used as the intermediate bit device, and the output device that does not have a controlling output channel can be used as the intermediate bit device.

Name	Type	Number	Address	Property
I (Input device)	Bit device	%IX0.0~%IX0.7	6000 ~ 6007	Read-only
		%IX1.0~%IX1.7	6008 ~ 600F	Read-only
		.....	.....	Read-only
		%IX127.0~%IX127.7	63F8 ~ 63FF	Read-only
	Word device	%IW0~%IW63	8000 ~ 803F	Read-only
Q (Output device)	Bit device	%QX0.0~%QX0.7	A000 ~ A007	Read/Write
		%QX1.0~%QX1.7	A008 ~ A00F	Read/Write
		.....	.....	Read/Write
		%QX127.0~%QX127.7	A3F8 ~ A3FF	Read/Write
	Word device	%QW0~%QW63	A000 ~ A03F	Read/Write
M (Middle device)	Word device	%MW0~%MW32767	0000 ~ 7FFF	Read/Write

Note: The %MW0~%MW999 addresses are defaulted to power outage persistency without binding variables.

The conversion of the MODBUS address for the bit devices related to QX or IX is done as follows.

For QXA.B, the conversion rule is A*8 + B converted to Hexadecimal +0xA000.

If the Modbus address corresponding to %QX50.1 is 0xA191, the calculation method is 50*8+1=401=0x191

0x191+0xA000=0xA191

Chapter 6 Modbus communication protocol description

6.1 ASCII mode message structure

The basic format of a Modbus protocol frame message in RTU mode is shown in the table below.

Starting character	Slave station number	Function code	Device first address	Data	LRC Checksum	Ending character
1 character	2 characters	2 characters	4 characters	n characters	2 characters	2 characters

Data structure of a frame of data

Content	Description
Starting character	The starting character is a colon " : ", and the corresponding ASCII code is 16#3A.
Slave station number	Specify the station number of the slave station to be accessed. Range: 1~255（16#1~26#FF）
Function code	Specify read, write, or read/write to the slave. Supported function codes: 03, 04, 06, 16#10, 16#17, 01, 02, 05, 16#0F .
Device first address	Specify the first address of the device to read or write to the slave.
Data	Specify the data associated with reading, writing, or reading and writing to the slave.
LRC Checksum	Start from "slave station number" to the last byte of "data", and calculate LRC checksum according to LRC checksum rule, please refer to other books for LRC checksum calculation method.
Ending character	Ending character 1 (CR) = 16#0D Ending character 0 (LF) = 16#0A

The correspondence between characters and ASCII codes is shown in the table below:

Character	“0”	“1”	“2”	“3”	“4”	“5”	“6”	“7”
ASCII code	16#30	16#31	16#32	16#33	16#34	16#35	16#36	16#37
Character	“8”	“9”	“A”	“B”	“C”	“D”	“E”	“F”
ASCII code	16#38	16#39	16#41	16#42	16#43	16#44	16#45	16#46

◆ Function codes and data

The format of the data depends on the function code. For example, when reading data from two consecutive addresses of the controller with 16#1000 as the starting address, the communication station number of the controller is 1. The Modbus address corresponding to the controller %MW4096 unit is 16#1000. The meaning of bus data is explained below:

Master → Slave: 3A 30 31 30 33 31 30 30 30 30 30 30 32 45 41 0D 0A

Slave → Master: 3A 30 31 30 33 30 34 30 30 30 31 30 30 30 32 46 35 0D 0A

- **Request information:**

Content	Character	ASCII code
Starting character	“: ”	3A
Communication station number: 01	“0”	30
	“1”	31
Function code：03	“0”	30
	“3”	33
Starting device address: 16#1000	“1”	31
	“0”	30
	“0”	30
	“0”	30
Number of data (word): 2	“0”	30
	“0”	30
	“0”	30
	“2”	32
LRC Check digit: 16#EA	“E”	45
	“A”	41
Ending character 1	CR	0D
Ending character 0	LF	0A

- **Response information:**

Content	Character	ASCII code
Starting character	“: ”	3A
Communication station number: 01	“0”	30
	“1”	31
Function code：03	“0”	30
	“3”	33
Number of data (Byte):	“0”	30
	“4”	34
Value of address 16#1000	“0”	30
	“0”	30
	“0”	30
	“1”	31
Value of address 16#1001	“0”	30
	“0”	30
	“0”	30
	“2”	32
LRC Check digit: 16#F5	“F”	46
	“5”	35
Ending character 1	CR	0D
Ending character 0	LF	0A

6.2 RTU mode message structure

The basic format of a frame of Modbus protocol message in RTU mode is shown in the table below.

Starting character	Slave station number	Function code	Device first address	Data	LRC Checksum	Ending character
1 character	2 characters	2 characters	4 characters	n characters	2 characters	2 characters

Content	Description
Starting	Keep the bus no data for ≥ 10ms.
Slave station number	Specify the station number of the slave station to be accessed. Range: 1~255（16#1~26#FF）.
Function code	Specify read, write, or read/write to the slave. Supported function codes:03, 04, 06, 16#10, 16#17, 01, 02, 05, 16#0F .
Device first address	Specify the first address of the device to read or write to the slave.
Data	Specify the data associated with reading, writing, or reading and writing to the slave.
CRC Checksum	Start from "slave station number" to the last byte of "data", and calculate CRC checksum according to CRC checksum rule, please refer to other books for CRC checksum calculation method.
Ending	Keep the bus no data for ≥ 10ms.

◆ Function codes and data

The format of the data depends on the function code. For example, when read the data of 2 consecutive addresses with 16#2000 as the starting address of the controller, the communication station number of the controller is 1. The Modbus address corresponding to the controller %MW8192 device is 16#2000. The meaning of bus data is explained below:

Master → Slave: 01 03 20 00 00 02 CF CB

Slave → Master: 01 03 04 00 01 00 02 2A 32

- **Request information:**

Starting	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	High 8 bits word device number	Low 8 bits word device number	CRC checksum high 8 bits	CRC checksum lower 8 bits	Ending
	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	
Keep the bus no data for ≥ 10ms	01	03	20	00	00	02	CF	CB	No input data ≥ 10ms

- **Response information:**

Starting	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	High 8 bits word device number	Low 8 bits word device number	CRC checksum high 8 bits	CRC checksum lower 8 bits	Ending
	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	
Keep the bus no data for ≥ 10ms	01	03	20	00	00	02	CF	CB	No input data ≥ 10ms

6.3 An introduction to Modbus function codes

◆ Function code: 03, Read the value of a single register or multiple word device registers

• Request information data structure:

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	High 8 bits word device number	Low 8 bits word device number	CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

Note: The unit of the register number is Word.

• Response information data structure:

Data meaning	Slave station number	Function code	Number of data	Data value		Data value	Data value		CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	……	Byte n	Byte n+1	Byte n+2	Byte n+3

Note: The unit of the data number is Byte.

• Exception response information data structure:

Data meaning	Slave station number	16#80+Function code	Exception response code	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Read the contents of controller 16#A000, 16#A001 address by 03 function code, 16#A000, 16#A001 is the Modbus address of %QW0 and %QW1 inside the controller. Assume the value of %QW0 is 16#1234, and the value of %QW1 is 16#5678.

• Request information:

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	High 8 bits word device number	Low 8 bits word device number	CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	03	A0	00	00	02	E6	0B

• Response information:

Data meaning	Slave station number	Function code	Number of data	Data value		Data value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte 9	Byte 10
Hexadecimal number	01	03	04	12	34	56	78	81	07

◆ Function code: 04, Read the value of a single register or multiple word device registers

• Request information data structure:

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	High 8 bits word device number	Low 8 bits word device number	CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

Note: The unit of the register number is Word.

• Response information data structure:

Data meaning	Slave station number	Function code	Number of data	Data value		Data value	Data value		CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	……	Byte n	Byte n+1	Byte n+2	Byte n+3

Note: The unit of the data number is Byte.

• Exception response information data structure:

Data meaning	Slave station number	16#80+Function code	Exception response code	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Read the contents of controller 16#8000 and 16#8001 addresses by 04 function code, 16#8000 and 16#8001 are the Modbus addresses of %IW0 and %IW1 inside the controller. Assume that the value of %IW0 is 16#1234 and the value of %IW1 is 16#5678.

• Request information:

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	High 8 bits word device number	Low 8 bits word device number	CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	04	80	00	00	02	58	0B

• Response information:

Data meaning	Slave station number	Function code	Number of data	Data value		Data value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte 9	Byte 10
Hexadecimal number	01	04	04	12	34	56	78	80	B0

◆ **Function code: 06, Write the value of a single word device register**

• **Request information data structure:**

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	Data value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

• **Response information data structure:**

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	Data value		CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

• **Exception response information data structure:**

Data meaning	Slave station number	16#80+Function code	Exception response code	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

The value 16#1234 is written into the controller 16#A000 address by the 06 function code.

• **Request information:**

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	Data value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	06	A0	00	12	34	A6	BD

• **Response information:**

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	Data value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	06	A0	00	12	34	A6	BD

◆ **Function code: 0x10, Write the value of multiple word device registers**

• **Request information data structure:**

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	Number of data（Word）		Number of data (Byte)	Data value		Data value	Data value		CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	……	Byte n	Byte n+1	Byte n+2	Byte n+3

- **Response information data structure:**

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	Number of data (Word)	CRC checksum high 8 bits		CRC checksum lower 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte7

- **Exception response information data structure:**

Data meaning	Slave station number	16#80+Function code	Exception response code	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

The 16#10 function code writes 16#1234 and 16#5678 to the controller's 16#A000 and 16#A001 addresses, and 16#A000 and 16#A001 are the Modbus addresses of %QW0 and %QW1 within the controller.

- **Request information:**

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	Number of data (Word)		Number of data (Byte)	Data value		Data value		CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte 9	Byte 10	Byte 11	Byte 12
Hexadecimal number	01	10	A0	00	00	02	04	12	34	56	78	70	9C

- **Response information:**

Data meaning	Slave station number	Function code	High 8 bits of the first address of the word device	Low 8 bits of the first address of the word device	Number of data（Word）		CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	10	A0	00	00	02	63	C8

◆ Function code: 0x17, Read and write the value of a single word device or multiple word devices

• Request information data structure:

Slave station number	Function code	First address of the read device	Number of read data (Word)	First address of the write device	Number of write data (Word)		Number of write data (Byte)	Write data value		Write data value	Write data value	CRC Checksum
1Byte	1Byte	2Byte	2Byte	2Byte	2Byte	1Byte	2Byte	……	2Byte	2Byte	Byte n Byte n+1 Byte n+2	Byte n+3

• Response information data structure:

Slave station number	Function code	Number of read data	Read data value	Read data value	Read data value	CRC Checksum
1Byte	1Byte	1Byte	2Byte	……	2Byte	2Byte

• Exception response information data structure:

Slave station number	16#80+Function code	Exception response code	CRC Checksum
1Byte	1Byte	1Byte	2Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Write 16#1234 and 16#0064 to the controller 16#0000 and 16#0001 addresses by 16#17 function code, 16#0000 and 16#0001 are the Modbus address of %MW0 and %MW1 inside the controller, and read the value of 16#8000 and 16#8001 addresses from the slave. 16#8000 and 16#8001 are the Modbus address of %IW0 and %IW1 inside the slave controller. 8001 are the Modbus addresses of %IW0 and %IW1 inside the slave controller.

• Request information:

Slave station number	Function code	First address of the read device	Number of read data (Word)	First address of the write device	Number of write data (Word)	Number of write data (Byte)	Write data value	Write data value	CRC Checksum
1Byte	1Byte	2Byte	2Byte	2Byte	2Byte	1Byte	2Byte	2Byte	2Byte
0x01	0x17	0x8000	0x0002	0x0000	0x0002	0x04	0x000C	0x0064	0x7DDC

• Response information:

Slave station number	Function code	Number of read data	Read data value	Read data value	CRC Checksum
1Byte	1Byte	1Byte	2Byte	2Byte	2Byte
0x01	0x17	0x04	0x0000	0x0000	0XF927

◆ Function code: 01, Read bit device register value

• Request information data structure:

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Number of bit device		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

Note: The unit of the number of data is Bit.

• **Response information data structure:**

Data meaning	Slave station number	Function code	Number of data（Byte）	Bit device value	Bit device value	Bit device value	CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

Notes: The value of the data numbers (Byte2) in the response message is decided by the value of bit device numbers (Byte4 and Byte5) in the response information. If the number of bit devices read in the request information is m and the quotient of m divided by 8 is n, the value of the number of bit devices (Byte4 and Byte5) in the response information is n if it is integrable; on the contrary, the value of the number of bit devices (Byte4 and Byte5) in the response information will be n + 1, please refer to the following example for details.

• **Exception response information data structure:**

Data meaning	Slave station number	16#80+Function code	Exception response code	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Read the status value of controller %QX0.0~%QX1.6 by 01 function code, the address of %QX0.0 is 16#A000. Assume %QX0.7~%QX0.0= 1000 0001, %QX1.6~%QX1.0=101 0001.

• **Request information:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Number of bit device		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	01	A0	00	00	0F	5E	0E

• **Response information:**

Data meaning	Slave station number	Function code	Number of data (Byte)	Bit device value	Bit device value	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
Hexadecimal number	01	01	02	81	51	18	50

◆ Function code: 02, Read bit device register value

• **Request information data structure:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Number of bit device		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

• **Response information data structure:**

Data meaning	Slave station number	Function code	Number of data（Byte）	Bit device value	Bit device value	Bit device value	CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

Notes: The value of the data numbers (Byte2) in the response message is decided by the value of bit device numbers (Byte4 and

Byte5) in the response information. If the number of bit devices read in the request information is m and the quotient of m divided by 8 is n, the value of the number of bit devices (Byte4 and Byte5) in the response information is n if it is integrable; on the contrary, the value of the number of bit devices (Byte4 and Byte5) in the response information will be n + 1, please refer to the following example for details.

- **Exception response information data structure:**

Data meaning	Slave station number	16#80+Function code	Exception response code	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Read the status value of controller %IX0.0~%IX1.6 by 02 function code, the address of %IX0.0 is 16#6000. Assume %IX-0.7~%IX0.0= 1000 0001, %IX1.6~%IX1.0=101 0001.

- **Request information:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Number of bit device		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	02	60	00	00	0F	26	0E

- **Response information:**

Data meaning	Slave station number	Function code	Number of data (Byte)	Bit device value	Bit device value	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
Hexadecimal number	01	02	02	81	51	18	14

◆ Function code: 05, Set the value of a single bit device register

- **Request information data structure:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Bit device value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

Notes: A bit device value of 0 indicates that FALSE is written to the bit device, and a bit device value of 16#FF00 indicates that TRUE is written to the bit device.

- **Response information data structure:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Bit device value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

- **Exception response information data structure:**

Data meaning	Slave station number	16#80+Function code	Exception response code	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

The value of the controller %QX0.7 is set to TRUE by the 05 function code and the address of %QX0.7 is 16#A007.

- **Request information:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Bit device value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	05	A0	07	FF	00	1F	FB

- **Response information:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Bit device value		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	05	A0	07	FF	00	1F	FB

◆ Function code: 0x0F, Write the values of multiple bit device registers

- **Request information data structure:**

Data meaning	Slave station number	Func-tion code	High 8 bits of the first	Low 8 bits of the first	Number of bit device（Bit）		Number of data (Byte)	Bit device value	Bit device value	Bit device value	CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	……	Byten	Byte n+1	Byte n+2

Notes: The number of Bytes of data in the request information is decided by the number of bit values to be written in the request information.

- **Response information data structure:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Number of bit device（Bit）		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

- **Exception response information data structure:**

Data meaning	Slave station number	16#80+Function code	Exception response code	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Set the address of controller %QX0.7~%QX0.0=1000 0001, %QX1.7~%QX1.0=1010 0011,%QX0.0 to 16#A000 by 0F function code.

- **Request information:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Number of bit device (Bit)		Number of data (Byte)	Bit device value	Bit device value	CRC checksum high 8 bits	CRC checksum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte9	Byte 10	Byte 11
Hexadecimal number	01	0F	A0	00	00	10	02	81	A3	62	03

- **Response information:**

Data meaning	Slave station number	Function code	High 8 bits of the first	Low 8 bits of the first	Number of bit device（Bit）		CRC check-sum high 8 bits	CRC check-sum lower 8 bits
Data order	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Hexadecimal number	01	0F	A0	00	00	10	76	07

Chapter 7 Modbus TCP communication protocol description

Modbus TCP is the Modbus message transfer protocol running on TCP/IP. The port number for Modbus TCP is 502. The basic format of a frame of ModbusTCP protocol message is shown in the table below.

Transaction identifier	Protocol Identifier	Data length	Slave station number	Function code	Device address	Data content
2Byte	2Byte	2Byte	1Byte	1Byte	2Byte	n Byte

Content	Description
Transaction identifier	Modbus request/response transaction processing identifier, which can be understood as the sequence number of the message, generally every time after the communication should add 1 to distinguish between different communication data messages.
Protocol Identifier	00 00 indicates Modbus protocol.
Data length	The number of bytes starts from the "slave station number" to the last byte of "data", and the length of data is measured in bytes.
Slave station number	Specify the station number of the slave station to be accessed. Range: 1~255（16#1~26#FF）.
Function code	Specify read, write, or read/write to the slave. Supported function codes: 03, 04, 06, 16#10, 16#17, 01, 02, 05, 16#0F .
Device address	Specify the first address of the device to read or write to the slave.
Data content	Specify the first address of the device to read or write to the slave.

7.1 An introduction to Modbus function codes

◆ Function code: 03, Read the value of a single register or multiple word device registers

• Request information data structure:

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Number of data (Word)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

• Response information data structure:

Transaction and protocol identifier	Data length	Slave station number	Function code	Number of data（Byte）	Data value	Data value	Data value
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

• Exception response information data structure:

Transaction and protocol identifier	Data length	Slave station number	16#80+Function code	Exception response code
4Byte	2Byte	1Byte	1Byte	1Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Read the contents of the controller 0x1000, 0x1001 address through the 03 function code, and 0x1000, 0x1001 are the Modbus addresses of the controller internal %MW4096, %MW4097. Assume that the value of %MW4096 is 0x1234, and %MW4097 value is 0x5678.

• Request information:

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Number of data (Word)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x03	0x1000	0x0002

• Response information:

Transaction and protocol identifier	Data length	Slave station number	Function code	Number of data (Byte)	Data value	Data value
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0007	0x01	0x03	0x04	0x1234	0x5678

◆ Function code: 04, Read the value of a single register or multiple word device registers

• Request information data structure:

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Number of data (Word)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

• Response information data structure:

Transaction and protocol identifier	Data length	Slave station number	Function code	Number of data（Byte）	Data value	Data value	Data value

4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

- **Exception response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	16#80+Function code	Exception response code
4Byte	2Byte	1Byte	1Byte	1Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Read the contents of controller 16#8000 and 16#8001 addresses by 04 function code, 16#8000 and 16#8001 are the Modbus addresses of %IW0 and %IW1 inside the controller. Assume that the value of %IW0 is 16#1234 and the value of %IW1 is 16#5678.

- **Request information:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Number of data (Word)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x04	0x8000	0x0002

- **Response information:**

Transaction and protocol identifier	Data length	Slave station number	Function code	Number of data (Byte)	Data value	Data value
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0009	0x01	0x04	0x04	0x1234	0x5678

◆ Function code: 06, Write the value of a single word device register

- **Request information data structure:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

- **Response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Data value	Data value	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	……	2Byte

- **Exception response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	16#80+Function code	Exception response code
4Byte	2Byte	1Byte	1Byte	1Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

The value 16#1234 is written into the controller 16#2000 address by the 06 function code.

- **Request information:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x06	0x2000	0x1234

- **Response information:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x06	0x2000	0x1234

◆ Function code: 0x10, Write the value of multiple word device registers

- **Request information data structure:**

Transaction/ Protocol	Data length	Slave station number	Function code	First address of the Word device	Number of data (Word)	Number of data (Byte)	Data value	Data value	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	1Byte	2Byte	……	2Byte

- **Response information data structure:**

Transaction/ Protocol	Data length	Slave station number	Function code	First address of the Word device	Number of data（Word）	Data value	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	……	2Byte

- **Exception response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	16#80+Function code	Exception response code
4Byte	2Byte	1Byte	1Byte	1Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

The 16#10 function code writes 16#0020 and 16#0021 to the controller's 16#1234 and 16#5678 addresses, and 16#010 and 16#0011 are the Modbus addresses of %QW16 and %QW17 within the controller.

- **Request information:**

Transaction/ Protocol	Data length	Slave station number	Function code	First address of the Word device	Number of data (Word)	Number of data（Byte）	Data value	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	1Byte	2Byte	2Byte
0x00000000	0x000B	0x01	0x10	0x0010	0x0002	0x04	0x0020	0x0021

- **Response information:**

Transaction/Protocol	Data length	Slave station number	Function code	First address of the Word device	Number of data (Word)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x10	0x1234	0x5678

◆ Function code: 01, Read bit device register value.

- **Request information data structure:**

Transaction/Protocol	Data length	Slave station number	Function code	First address of the bit device	Number of data (Bit)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

- **Response information data structure:**

Transaction/ Protocol	Data length	Slave station number	Function code	Number of data (Byte)	Data value	Data value	Data value
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

Notes: The value of the data numbers (Byte2) in the response message is decided by the value of bit device numbers (Byte4 and Byte5) in the response information. If the number of bit devices read in the request information is m and the quotient of m divided by 8 is n, the value of the number of bit devices (Byte4 and Byte5) in the response information is n if it is integrable; on the contrary, the value of the number of bit devices (Byte4 and Byte5) in the response information will be n + 1, please refer to the following example for details.

- **Exception response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	16#80+Function code	Exception response code
4Byte	2Byte	1Byte	1Byte	1Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Read the status value of controller %QX0.0~%QX1.6 by 01 function code, %QX0.0 address is 16#A000. Assume %QX-0.7~%QX0.0= 1000 0001, %QX1.6~%QX1.0=101 0001.

- **Request information:**

Transaction/Protocol	Data length	Slave station number	Function code	First address of the bit device	Number of data (Bit)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x01	0xA000	0x000F

- **Response information:**

Transaction/Pro-tocol	Data length	Slave station number	Function code	Number of data (Byte)	Data value	Data value
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x01	0x02	81	51

◆ Function code: 02, Read bit device register value

- **Request information data structure:**

Transaction/Protocol	Data length	Slave station number	Function code	First address of the bit device	Number of data (Bit)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

- **Response information data structure:**

Transaction/ Protocol	Data length	Slave station number	Function code	Number of data (Byte)	Data value	Data value	Data value
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

Notes: The value of the data numbers (Byte2) in the response message is decided by the value of bit device numbers (Byte4 and Byte5) in the response information. If the number of bit devices read in the request information is m and the quotient of m divided by 8 is n, the value of the number of bit devices (Byte4 and Byte5) in the response information is n if it is integrable; on the contrary, the value of the number of bit devices (Byte4 and Byte5) in the response information will be n + 1, please refer to the following example for details.

- **Exception response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	16#80+Function code	Exception response code
4Byte	2Byte	1Byte	1Byte	1Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Read the status value of controller %IX0.0~%IX1.6 by 02 function code, the address of %IX0.0 is 16#6000. Assume %IX0.7~%IX0.0= 1000 0001, %IX1.6~%IX1.0=101 0001.

- **Request information:**

Transaction/Protocol	Data length	Slave station number	Function code	First address of the bit device	Number of data (Bit)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x02	0x6000	0x000F

- **Response information:**

Transaction/Protocol	Data length	Slave station number	Function code	Number of data (Byte)	Data value	Data value
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x02	0x02	81	51

◆ Function code: 05, Set the value of a single bit device register

- **Request information data structure:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

Notes: The write value of 16#0000 indicates that FALSE is written to the bit device, and 16#FF00 indicates that TRUE is written to the bit device.

- **Response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

- **Exception response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	16#80+Function code	Exception response code
4Byte	2Byte	1Byte	1Byte	1Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

The value of the controller %QX0.7 is set to TRUE by the 05 function code and the address of %QX0.7 is 16#A007.

- **Request information:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x05	0xA007	0XFF00

- **Response information:**

Transaction and protocol identifier	Data length	Slave station number	Function code	First address of the Word device	Data value	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x05	0xA007	0XFF00	51

◆ Function code: 0x0F, Write the values of multiple bit device registers

- **Request information data structure:**

Transaction/ Protocol	Data length	Slave station number	Function code	First address of the bit device	Number of data（Bit）	Number of data (Byte)	Data value	Data value	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	1Byte	1Byte	……	1Byte

Notes: The number of Bytes of data in the request information is decided by the number of bit values to be written in the request information.

- **Response information data structure:**

Transaction/Protocol	Data length	Slave station number	Function code	First address of the Word device	Number of data (Bit)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

- **Exception response information data structure:**

Transaction and protocol identifier	Data length	Slave station number	16#80+Function code	Exception response code
4Byte	2Byte	1Byte	1Byte	1Byte

Notes: The exception response code is described in the section "Modbus and Modbus TCP Communication".

Example:

Set controller %QX0.7~%QX0.0=1000 0001, %QX1.7~%QX1.0=1010 0011, the address of %QX0.0 is 16#A000, and the address of QX0=1.0 is 16#A008 by 0F function code.

- **Request information:**

Transaction/ Protocol	Data length	Slave station number	Function code	First address of the bit device	Number of data（Bit）	Number of data（Byte）	Data value	Data value
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	1Byte	1Byte	1Byte
0x00000000	0x0009	0x01	0x0F	0xA000	0x0010	0x02	0x81	0xA3

- **Response information:**

Transaction/Protocol	Data length	Slave station number	Function code	First address of the Word device	Number of data (Bit)
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x0F	0xA000	0x0010

◆ Function code: 0x17, Read and write the value of a single word device or multiple word devices

- **Request information data structure:**

Transaction/Protocol	Slave station number	Function code	First address of the read device	Number of read data (Word)	First address of the write device	Number of write data (Word)	Number of write data (Byte)	Write data value	Write data value	Write data value
4Byte	1Byte	1Byte	2Byte	2Byte	2Byte	2Byte	1Byte	2Byte	……	2Byte

Notes: The number of Bytes of data in the request information is decided by the number of bit values to be written in the request information.

- **Response information data structure:**

Transaction/Protocol	Slave station number	Function code	Number of read data	Read data value	Read data value	Read data value
4Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

- **Exception response information data structure:**

Transaction and protocol identifier	Slave station number	16#80+Function code	Exception response code
4Byte	1Byte	1Byte	1Byte

Notes: Exception response codes are described in section [B.5 Modbus exception response codes].

Example:

Write 16#000C and 16#0064 to 16#0000 and 16#0001 by 16#17 function code, 16#0000 and 16#0001 are the Modbus address of %MW0 and %MW1 inside the controller, and read the value of 16#8000 and 16#8001 from 16#8000 and 16#8001, which are the Modbus addresses of %IW0 and %IW1 inside the slave controller. 8001 are the Modbus addresses of %IW0 and %IW1 inside the slave controller.

- **Request information:**

Transaction/ Protocol	Slave station number	Function code	First address of the read device	Number of read data (Word)	First address of the write device	Number of write data (Word)	Number of write data (Byte)	Write data value	Write data value
4Byte	1Byte	1Byte	2Byte	2Byte	2Byte	2Byte	1Byte	2Byte	2Byte
0x00000000	0xFF	0x17	0x8000	0x0002	0x0000	0x0002	0x04	0x000C	0x0064

- **Response information:**

Transaction/Protocol	Slave station number	Function code	Number of read data	Read data value	Read data value
4Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0xFF	0x17	0x04	0x0000	0x0000

Chapter 8 Communication instruction error code description

8.1 Communication instruction error code description

Error Code		Meaning	Handling measure
Hexadecimal number	Decimal number		
1004	4100	CAN slave station number exceeds the allowed range.	Check that the CAN slave station number is within the allowed range.
1301	4865	The CANopen master writes or reads parameters to the slave via the data service (SDO), but the slave does not respond.	Check that the hardware is connected correctly. Check that the slave station operated in the instructions exists Check that all stations in the network have the same baud rate. Check that the cable is a shielded twisted pair. Check that there are no sources of interference in the vicinity.
1302	4866	CANopen master writes or reads parameters to slave via data service (SDO), the slave does not respond properly.	Check that the specified parameter exists. Check that the value of the written parameter is within the range allowed by the parameter.
1303	4867	The CANopen master writes or reads parameters to the slave via the data service (SDO), but the slave replies with an error code.	Check that the slave supports data services (SDO) Check that the slave is working properly.
4500	17664	CAN slave station number exceeds the allowed range.	Check that the CAN slave station number is within the allowed range.
4501	17665	The value of the input variable Mode is outside the allowed range.	Modify and ensure that the value of the input variable Mode is within the allowed range.
4502	17666	Diagnosed slave is not configured in the software.	Perform diagnostics on the slave configured in the software.
4503	17667	CAN communication port is not set to master mode	Set the CAN communication port to master mode.
4510	17680	CAN slave station number exceeds the allowed range.	Check that the CAN slave station number is within the allowed range.
4511	17681	The diagnosed slave does not exist (not configured).	Perform diagnostics on the configured slaves.
6000	24576	Ethernet ModbusTCP data exchange channel number (LinkNum) setting value is not within the allowed range.	Modify and ensure that the ModbusTCP data exchange channel number (LinkNum) is within the allowed range.
6001	24577	The write operation length configured for the Ethernet ModbusTCP data switching channel exceeds the maximum permissible value.	Modify and ensure that the configured write operation length is within the allowed range.
6002	24578	The read operation length of the Ethernet ModbusTCP data exchange channel configuration exceeds the maximum allowed value.	Modify and ensure that the configured write operation lengths are within the allowed range.
6003	24579	Ethernet physical connection is abnormal.	Check and ensure that the Ethernet hardware connection is normal.
6004	24580	Ethernet Socket number is set incorrectly.	Modify and ensure that the parameter SocketNum is within the allowed range.
6005	24581	The transmit length configured by the Ethernet Socket function exceeds the upper limit.	Modify and ensure that the send length is within the allowed range.
6006	24582	The transmit length configured by the Ethernet Socket function exceeds the upper limit.	Modify and ensure that the send length is within the allowed range.
6007	24583	Setting error of communication timeout (TimeOut) for Ethernet ModbusTCP Link function	Modify and ensure that the configured communication timeout is within the allowed range.

Error Code		Meaning	Handling measure
Hexadecimal number	Decimal number		
6008	24584	The sending and receiving lengths configured for the Ethernet Socket function are both 0.	Modify and ensure that the sending and receiving lengths are not both 0 at the same time.
6010	24592	Setting error of Number (LinkNum) of the serial Modbus data exchange channel parameter configuration	Modify and ensure that the parameter LinkNum is within the allowed range.
6011	24593	The write operation length of the serial Modbus data exchange channel parameter exceeds the upper limit.	Modify and ensure that the configured write operation length is within the allowed range.
6012	24594	The read operation length of the serial Modbus data exchange channel parameter exceeds the upper limit.	Modify and ensure that the configured read operation length is within the allowed range.
6013	24595	The sending data length configured in the Serial Modbus data exchange channel parameter exceeds the upper limit.	Modify and ensure that the sending data length is within the allowed range.
6014	24596	The receiving data length of the serial Modbus data exchange channel parameter exceeds the upper limit.	Modify and ensure that the receiving data length is within the allowed range.
601A	24602	Communication timeout for serial Modbus data exchange channel parameter configuration	Modify and ensure that the configured communication timeout is within the allowed range
601B	24603	The read operation length and write operation length of the serial Modbus data exchange channel parameter configuration are both 0.	Modify and ensure that the read operation length and write operation length are not both 0 at the same time.
601C	24604	The sending data length and receiving data length configured by the serial port custom protocol data send/receive instruction function are both 0.	Modify and ensure that the sending data length and receiving data length are not both 0.
6020	24608	Insufficient cache space at the start of the WORD write operation data cache (WriteAddr) in serial Modbus data exchange channel parameter configuration	Modify the starting address of the local write operation data cache and ensure that there is enough space to to meet the specified write operation length.
6021	24609	The local cache starting address (WriteAddr) specified by the WORD write operation in the serial Modbus data exchange channel parameter configuration is not within the allowed range.	Modify and ensure that the starting address of the local write data cache is within the allowed area.
6022	24610	The starting place (WriteAddr) of the data cache for the WORD write operation in the serial Modbus data exchange channel parameter configuration is within the allowed area but does not satisfy the word address alignment.	Modify the offset of the starting address of the local cache specified for a write operation or the starting address of the data cache for a write operation.
6023	24611	Insufficient cache space at the start of the WORD read operation data cache (ReadAddr) in serial Modbus data exchange channel parameter configuration	Modify the starting address of the local read operation data cache and ensure that there is enough space for the specified read operation length.
6024	24612	The local cache starting address (ReadAddr) specified by the WORD read operation in the serial Modbus data exchange channel parameter configuration is not within the allowed range.	Modify and ensure that the starting address of the local read operation data cache is within the allowed area.
6025	24613	The starting place (ReadAddr) of the data cache for the WORD read operation in the serial Modbus data exchange channel parameter configuration is within the allowed area, but does not satisfy the word address alignment.	Modify the local cache starting address specified for a read operation or the offset of the data cache starting address for a read operation.

Error Code		Meaning	Handling measure
Hexadecimal number	Decimal number		
6026	24614	Insufficient cache space for the starting address (WriteAddr) of the Bit write operation data cache in the serial Modbus data exchange channel parameter configuration.	Modify the starting address of the local write data cache and ensure that there is enough space for the specified write operation length.
6027	24615	The local cache starting address (WriteAddr) specified in the Bit Write operation in the serial Modbus data exchange channel parameter configuration is not within the allowed range.	Modify and ensure that the starting address of the local write data cache is within the allowed area.
6028	24616	Insufficient cache space for the starting address (ReadAddr) of the Bit read operation data cache in the serial Modbus data exchange channel parameter configuration.	Modify the starting address of the local read operation data cache and ensure that there is enough space for the specified read operation length.
6029	24617	The starting address (ReadAddr) of the Bit read operation data cache in the serial Modbus data exchange channel parameter configuration is not within the allowed range.	Modify and ensure that the starting address of the local read operation data cache is within the allowed area.
6030	24624	The type (Mode) of the read/write address parameter in the parameter configuration of the serial Modbus data exchange channel is incorrectly set.	Modify and ensure that the parameters are within the allowed range.
6031	24625	Error in the WriteMode specified for the read operation in the parameter configuration of the serial Modbus data exchange channel.	Modify and ensure that the parameters are within the allowed range.
6040	24640	Insufficient local cache space specified for the send operation	Modify the starting address of the local cache or the length of the data to be sent.
6042	24642	The starting address of the local cache specified for sending operation is out of the allowed range.	Modify and ensure that the starting address of the local cache is within the allowed range.
6043	24643	Insufficient local cache space specified for receiving operations	Modify the starting address of the local cache or the length of the data to be sent.
6045	24645	The starting address of the local cache specified for the receive operation is out of the allowed range.	Modify and ensure that the starting address of the local cache is within the allowed range.
6050	24656	The communication port number is out of range.	Modify and ensure that the communication port numbers are within the allowed range.
6051	24657	The specified port does not exist.	Ensure that the specified port is in a normal state. For example, if the specified port is a port on an expansion card, but the corresponding expansion card is not installed, this error will be reported.
6100	24832	Ethernet Socket function parameter setting error	Modify and ensure that the parameters related to the Socket function are within the allowed range.
6101	24833	Ethernet physical connection error	Check the Ethernet physical connection is correct.
6102	24834	Ethernet TCP remote IP address error	Modify and ensure that the remote IP address is correct
6103	24835	Ethernet TCP port error	Modify and ensure that the remote port is correct
6105	24837	Ethernet TCP sending cache address error	Modify and ensure that the address of the sending cache is correct
6106	24838	Ethernet TCP/UDP receive action has been triggered	Wait until receiving is completed before executing
6107	24839	Ethernet TCP receiving cache address error	Modify and make sure the address of the receiving cache is correct

Error Code		Meaning	Handling measure
Hexadecimal number	Decimal number		
6108	24840	As a TCP server, the actual received data length is longer than the set length	Modify and ensure that the received length is greater than or equal to the length of the first data received (in bytes).
6109	24841	Ethernet UDP actual received data length exceeds the set length	Modify and ensure that the received length is greater than or equal to the length of the first data received. (Unit: Byte)
610A	24842	Ethernet UDP remote IP address error	Modify and ensure that the remote IP address is correct
610B	24843	Ethernet UDP port error	Modify and ensure that both the local and remote ports cannot be 0.
610C	24844	Ethernet UDP sending cache address error	Modify and ensure that the address of the sending cache is correct.
610D	24845	Ethernet UDP receiving cache address error	Modify and ensure that the address of the receiving cache is correct.
610E	24846	Ethernet TCP connection timeout	Check that the Socket configuration is correct or that the remote device is working properly.
610F	24847	As a TCP client, the actual received data length exceeds the set length.	Modify and ensure that the received length is greater than or equal to the length of the first data received (in Byte)
6110	24848	Ethernet TCP connection is rejected by the remote device	Re-establish the connection after ensuring that the remote device is OK.
6111	24849	Ethernet TCP/UDP connection has not been opened.	Check that the connection is open.
6112	24850	Ethernet TCP/UDP connection has been triggered	Ensure that the connection is not triggered repeatedly during the connection establishment process.
6113	24851	Ethernet TCP/UDP data transmission has been triggered.	Wait for the sending is completed before triggering.
6114	24852	Ethernet TCP/UDP connection is established.	Ensure that connection establishment is not triggered repeatedly when the connection has already been established.
6115	24853	Ethernet TCP/UDP connection is closing.	Ensure that shutdown is not triggered repeatedly when the connection is being closed.
6116	24854	Ethernet TCP/UDP Connection not closed.	Ensure that Socket parameters are configured when the connection is closed.
6117	24855	Ethernet TCP/UDP specified sending length error	Modify and ensure that the sending length is within the allowed range.
6118	24856	Ethernet TCP/UDP specified receiving length error	Modify and ensure that the receiving length is within the allowed range.
6120	24864	ModbusTCP master is not started.	Ensure that the ModbusTCP master is active at the time of executing this instruction, the ModbusTCP master function can be activated by an instruction.
6121	24865	ModbusTCP data exchange is not configured.	Ensure that the data exchange channel to be operated has been configured via software or instruction.
6122	24866	ModbusTCP data exchange mode error	Check the range of the parameter ExeMode to make sure that it is within the allowed range (0 to 1).
6123	24867	ModbusTCP data exchange LinkKeepTime is set incorrectly	Check the parameter LinkKeepTime to make sure that it is within the allowed range, LinkKeepTime must be greater than 0.
6124	24868	TCP connection is closed.	Ensure that the TCP connection is open

Error Code		Meaning	Handling measure
Hexadecimal number	Decimal number		
6125	24869	TCP connection timeout	A TCP connection was not established within the specified time.
6126	24870	ModbusTCP message response timeout	A ModbusTCP message was not responded to within the specified time.
6129	24873	Transaction identifier error in ModbusTCP message (message header)	The transaction identifier in the ModbusTCP message does not conform to the rules.
612A	24874	Error in protocol identifier in ModbusTCP message (message header)	Ensure that the protocol identifier in the ModbusTCP message is not 0.
612B	24875	ModbusTCP message length error (message header)	Check ModbusTCP message length
612D	24877	The TCP connection is not closed, but the connection is established.	Ensure that the TCP connection is closed before establishing the connection.
6130	24880	Incorrect setting of the type (Mode) of the serial port read/write address.	Check the range of the parameter Mode to make sure that it is within the allowed range (0 or 1).
6131	24881	Modbus master function is not activated.	Start the Modbus master function first
6132	24882	Modbus data exchange is not configured.	Ensure that the data exchange channel to be operated has been configured by software or instruction.
6133	24883	The serial port is in slave mode, Modbus master startup is not allowed.	Ensure that the serial port is working in master mode before executing the serial port master enable instruction.
6134	24884	Modbus data exchange mode (ExeMode) error	Check the range of the ExeMode parameter to make sure it is within the allowed range (0~1).
6135	24885	Unrecognizable function code	Check whether the function code in the message data from the slave to the master is correct.
6136	24886	Modbus slave replies with an address different from the configured address	Check the correctness of the message data from the slave to the master.
6137	24887	Modbus received data length in slave reply data does not match the configured length.	Check whether the message data from the slave to the master is correct.
6138	24888	Modbus receiving timeout	Check that the hardware connections are correct. Check that all stations in the network have the same baud rate. Check that the cable is a shielded twisted pair. Check that there are no interference sources in the vicinity.
6139	24889	Modbus Checksum error	
613B	24891	Modbus Actual receive length exceeds maximum receive length.	Check that the message data replied by the slave station to the master station is correct.
7000	28672	The current device does not support this function.	Check that the current device supports the currently used function

Innovation Integrity Service

HCFA

HCFA_ATC

Zhejiang Hechuan Technology Co., Ltd.
No.9, Fucai Road, Longyou Industrial Zone, Quzhou City, Zhejiang Province, P.R. China

R&D Center (Hangzhou)
No. 299, Lixin Road, Qingshanhu Road, Lin'an District, Hangzhou City, Zhejiang Province, P.R. China

400 TEL - 400-012-6969

HCFA Official Website - www.hcfa.cn