CTH300 H5X-10 motion controller

User Manual

❍ Version: V1.00

❍ Release date: 04/2022

Copyright Notice

Copyright Notice

Shenzhen COTRUST Co., Ltd. All rights reserved, and reserves the right of final interpretation and modification of this manual and this statement. Without the written permission of Shenzhen COTRUST Co., Ltd, no unit or individual shall copy, extract, backup, modify, disseminate, compile, translate into other languages, or translate any part of this manual in any way or form. All or part of it is used for commercial purposes.

Trademark statement

 ®、TrustPLC®、MagicWorks®、COTRUST®、MagiCampus®、COMOTION®、  ® are registered trademarks of Shenzhen COTRUST Co., Ltd. or its affiliates, and are protected by law, and infringement must be investigated. Without the written permission of Shenzhen COTRUST Automation Technology Co., Ltd. or its affiliates, no unit or individual may use, copy, permanently download, modify, disseminate, transcribe or interact with any part of the trademark in any way or for any reason. Products are sold in bundles, or registered as domain names or wireless URL names, or the main part of domain names or wireless URLs.

Disclaimer

This manual is based on existing information, and its content is subject to change without notice. Shenzhen COTRUST Co., Ltd. has tried its best to ensure that the contents are accurate and reliable when compiling this manual. However, Shenzhen COTRUST Automation Technology Co., Ltd. shall not be responsible for the loss and damage caused by omissions, inaccuracies or printing errors in this manual.

Preface

Thank you for purchasing the CTH300 H56-10 and H52-10 products of Shenzhen COTRUST Co., Ltd.!

Before using CTH300 H56-10 and H52-10 products, please read this manual carefully so that you can grasp the product's characteristics more clearly, apply it more safely, and make full use of the rich functions of this product.

Product Features

The CTH300 H56-10 and H52-10 motion controllers can cooperate with a variety of expansion modules or units to form a powerful programmable logic controller system. The system meets the control needs of high-end OEM equipment and small and medium-sized engineering projects. Features such as precision, intelligence and ease of use, strong communication interconnection capabilities, and medium I/O scale. Compared with the previous CTH300-H PLCs, H56-10 and H52-10 motion controllers inherited their advantages while adding digital input, high-speed counters and other functions, greatly improving the performance of COTRUST's medium-sized PLCs.

Suitable

This manual is designed for engineers, installers, maintenance personnel and electricians with common sense of automation. Provide installation and debugging information about CTH300 H56-10 and H52-10 motion controllers.

Service support

Shenzhen COTRUST Co., Ltd. has established an after-sales maintenance service center and provides telephone hotline services. When customers encounter problems during the use of the product, they can contact the service support at any time. Please go to http://www.co-trust.com/Company/Contact/index.html to obtain the service support hotlines in various regions. In addition, customers can also keep up to date with the latest product developments and download required technical documents through the website of Shenzhen COTRUST Co., Ltd.http://www.co-trust.com or official WeChat.

Safety Precautions

Only qualified person are allowed to install, operate and maintenance on CTH300-H5X PLC. Qualified persons are defined as persons who are authorized to commission, ground, tag circuits, equipment, and systems in accordance with established safety practices and standards.

COTRUST has no responsibility for any consequence caused by using this product.

This manual contains notices that you should observe to ensure your personal safety, as well as to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows according to the level of danger:

Warning

“Warning” indicates operation not as required may result in severe personal injuries or even death.

Caution

“Caution” indicates operation not as required may result in personal injuries or device damage

Notice

“Notice” indicates necessary supplements or explanations

Revision History

Release date	Version	Revised content
April 2022	V1.00	First edition release

Content

Product Overview

1

This chapter mainly provides an overview of H56-10 and H52-10 application systems, as follows:

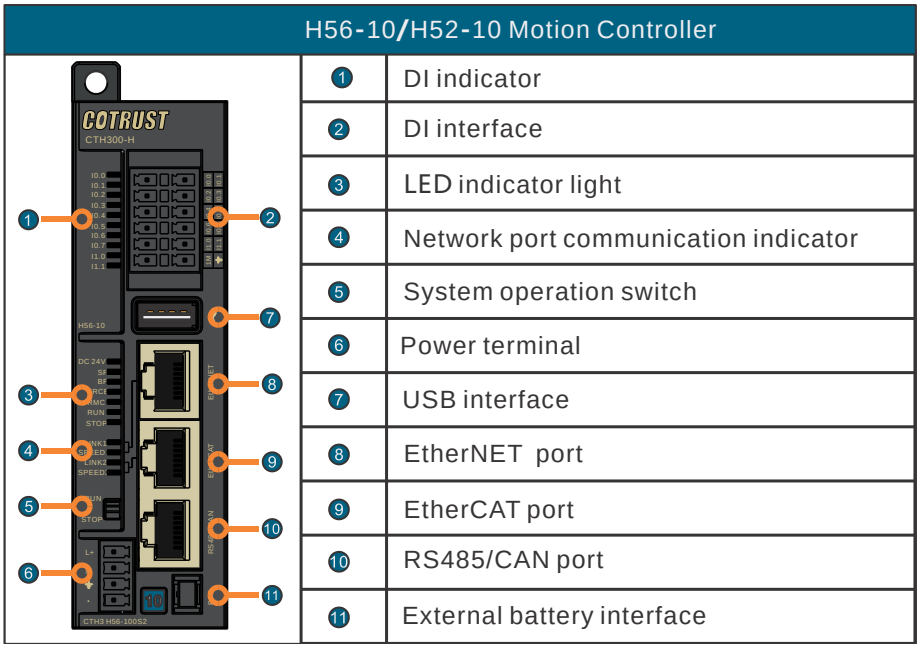
1.1 Product Description

H56-10/H52-10 motion controller can be combined with a variety of expansion modules to form a powerful PROGRAMMABLE logic controller system. The system can meet the control requirements of high-end OEM equipment and small and medium-sized engineering projects, and has the characteristics of high speed, high precision, intelligent and easy to use, strong communication and interconnection capability, and medium I/O scale. H56-10/H52-10 motion controller added digital input, high speed counter, greatly improve the performance of COTRUST medium PLC.

1.1.1 CPU Introduction

The CTH300 H56-10/H52-10 motion controller is the central processing unit of the PLC application system. As the core control device of the system, it handles various operations and the operation of user programs.

The following is a schematic diagram of the appearance of CTH300 H56-10 and H52-10:



The characteristics of CTH300 H56-10 and H52-10 are as follows:

◆ **Fast calculation speed**

The execution speed of the logic instructions of the CPU is 0.086μs, and the floating-point operation speed is 1.4μs. the CPU adopts the Cortex-A8 core processor, the main frequency is up to 1GHz.

◆ **High-speed transmission**

The high-speed Bus between the CPU and the expansion module adopts M-LVDS technology, and the data transmission rate reaches 55Mbps.

The CPU natively integrates 6 high-speed counters and 10 digital inputs.

◆ **Powerful system expansion capability**

The maximum expandable digital I/O is 1024, and the maximum expandable analog I/O is 256.

◆ **Support multiple communication protocols**

The CPU natively supports communication protocols such as EtherNET, EtherCAT, CANopen, PPI/MPI, ModBus, etc., and can also support CANopen high-speed fieldBus through expansion modules.

◆ **Easy and fast programming**

Program through MagicWorks PLC programming software (V2.19 or later). C programming language is supported.

◆ **Support remote function**

H56-10 and H52-10 support remote Ethernet communication, remote monitoring and debugging of programs, and remote firmware update.

◆ **Perfect operation control function**

Support single-axis motion function, multi-axis interpolation function, electronic cam and electronic gear function.

◆ **Support external battery**

H56-10 and H52-10 are equipped with an external battery port to support an external battery, extending the retention time of the super capacitor to more than one year.

Note: The external battery needs to be purchased separately from COTRUST (order number: CTH3-BAT-000S1). For the installation of the external battery, please refer to 3.4 Installation Method

1.1.2 CTH300 Expansion Modules

H56-10 and H52-10 motion controllers belong to the CTH300 of medium-sized PLC family, which support CTH300 expansion modules, and their module types include: digital input and output, analog input and output, temperature acquisition, high-speed counting, CAN communication module, etc., Please refer to the table below for specific modules and models:

Table 1-1 CTH300 expansion modules

	DI module CTH3 DIT-080S1 CTH3 DIT-160S1 CTH3 DIT-320S1	DO module CTH3 DQT-080S1 CTH3 DQT-160S1 CTH3 DQT-320S1 CTH3 DQR-080S1 CTH3 DQR-160S1	
	AI module CTH3 AIS-040S1 CTH3 AIV-080S1 CTH3 AIC-080S1	AO module CTH3 AQS-040S1 CTH3 AQS-080S1 AI/AO module CTH3 AMS-060S1	Temperature module CTH3 AIT-040S1 CTH3 AIT-080S1 CTH3 AIR-040S1 CTH3 AIR-080S1

	High-speed counting module CTH3 HSC-020S1		Pulse output module CTH3 HSP-040S1		CAN master module CTH3 CAN-1M0S1
	Power module CTH3 PWR-020S1		Intermediate expansion module CTH3 INT-000S1		ECT slave module CTH3 ECT-000S1
	Profinet slave module CTH3 PNT-000S1				

1.1.3 Programming Software MagicWorks PLC

The programming software MagicWorks PLC, which is developed by COTRUST, provides a good programming environment for users to develop, edit and monitor their own applications, so that you can develop your applications quickly and efficiently.

Software features

- Support IEC61131 programming language
- programing language:LAD, STL, C program
- Support Chinese / English programming
- Intelligent help
- Rich instruction

Computer configuration requirements

Operating system: Windows 7 and above.

Hardware Configuration: Compatible machines above IBM 486, with more than 128MB of RAM, and at least 500MB of free hard disk space.

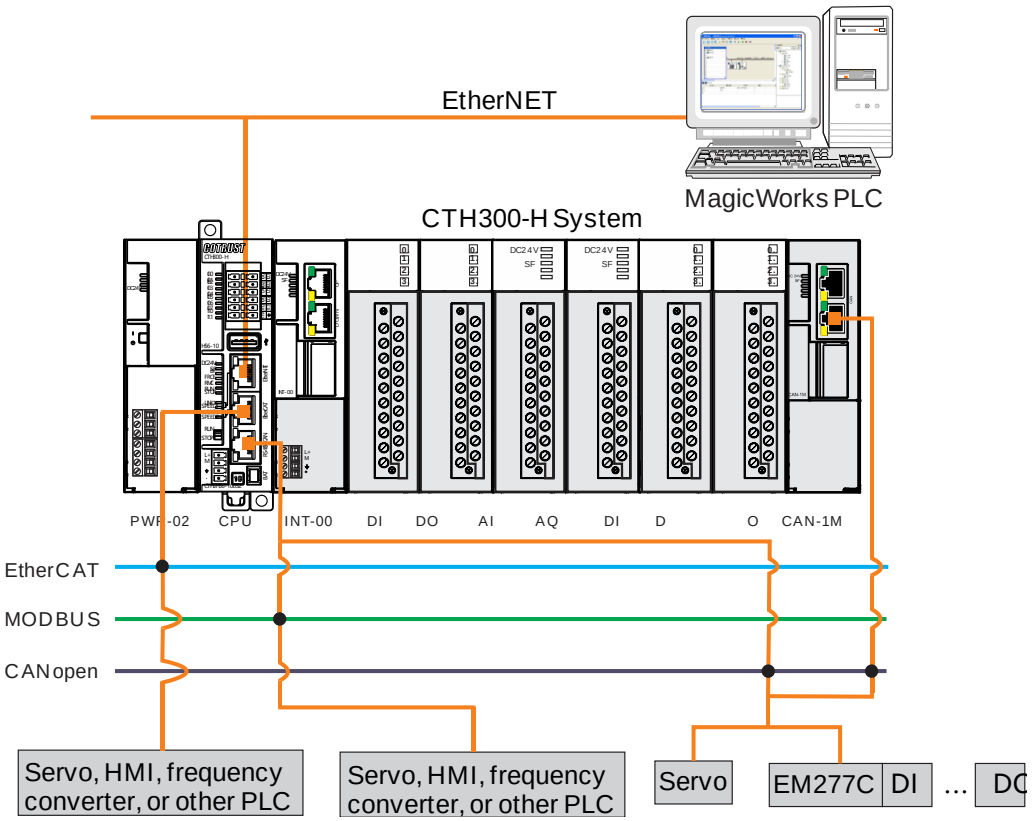
Tips

For specific usage, please refer to the online help of MagicWorks PLC software.

The software can be downloaded for free from the official website of COTRUST, download address: http://www.co-trust.com/Download/index.html

1.2 System Structure

In the application system, the I/O expansion module is connected to the rear of the CTH300 H56-10/H52-10 or the Intermediate extension module INT-00 through the bus interface. The I/O extension module sends various collected data and diagnostic information to H56-10/H52-10 via the bus, and H56-10/H52-10 processes it according to the user program and related information. The typical system architecture of H56-10 and H52-10 application systems is shown in the figure below:



- H56-10, H52-10 can communicate with the host computer through the EtherNET communication port (using a standard network cable) or the RS485 communication port.
- The H56-10 application system consists of a central rack and three expansion racks at most, and a maximum of 32 expansion units can be installed. H52-10 supports local rack expansion modules.
- H56-10 and H52-10 support multiple communication methods such as RS485, EtherCAT, EtherNET, CANopen, etc., and realize CANopen communication through the Bus expansion CAN-1M module or the CAN communication port of the CPU (as shown in the figure above).

1.3 Electrical Specifications and Environmental Conditions

Electromagnetic compatibility (EMC) refers to the ability of electrical equipment to operate normally in its electromagnetic environment without disturbing the environment. Table 1-2 and

table 1-3 describe the electrical specifications and environmental conditions standards that H56-10 / H52-10 should follow. Programmable logic controller standard: IEC61131-2,GB15969.

Table 1-2 Electrical specifications

Electromagnetic compatibility-immunity	
Electrostatic discharge IEC61000-4-2	Contact discharge: ±4kV Air discharge: ±8kV
Electrical fast transient burst IEC61000-4-4	power cable: 2kV, 5kHz Signal cable: 2kV, 5kHz(I/O coupling clamp) 1kV, 5kHz (Communication Coupling Clip)
SurgeIEC61000-4-5	power cable: 2kV(asymmetric),1kV(symmetry)
Radio frequency electromagnetic field radiation IEC61000-4-3	80MHz~1GHz, 10V/m, 80%AM(1kHz)
RF field induced conducted interference IEC61000-4-6	0.15MHz~80MHz, 10V/m, 80%AM(1kHz)
Short-term interruption and voltage change of DC power input port IEC61000-4-29	Short interruption: 10ms Voltage change: 80%~120%,100ms
Environmental test	
High temperature operation IEC60068-2 Low temperature operation IEC60068-2	140°F 16 hours 14°F 16 hours
High temperature start IEC60068-2 Low temperature start IEC60068-2	140°F 2 hours 14°F 2 hours
High and low temperature cycle operation IEC60068-2	-14°F~140°F Residence time 3 hours, Temperature rise rate 33.8°F /min, 2 cycles
High temperature storage IEC60068-2 Low temperature storage IEC60068-2	158°F 72 hours -40°F 72 hours
Thermal shock IEC60068-2	-40°F~158°F Residence time 3 hours, temperature change time <1min, 5 cycles
High temperature and humidity IEC60068-2	104°F 48 hours
Alternating damp heat IEC60068-2	77°F ~131°F 95% 2 cycles
Sinusoidal vibration (bare metal) IEC60068-2	5~150Hz, 0.05G2/Hz 150Hz~500Hz -3dB/oct,1 hour/axis, X, Y, Z total 3 axes
Impact (bare metal) IEC60068-2	15G,11ms pulse, 3 times/direction
Flowing mixed gas corrosion test IEC60068-2-60	H2S: 0.1ppm, NO2: 0.2ppm, CL2: 0.02ppm,temperature: 86°F, humidity: 75%, Cycle: 4 days
High voltage insulation test	
Between 24V/5V nominal circuit 110V/220V circuit to ground 110V/220V circuit to 110V/220V circuit 110V/220V circuit connected to 24V/5V circuit	500 VAC 1500 VAC 1500 VAC 1500 VAC

Table 1-3 Environmental conditions for the H56-10/H52 motion controller

Environmental conditions - transportation and storage		
Temperature		-40~158°F
Atmospheric pressure		1080 hPa~660 hPa (The corresponding height is -1000m~+3500m)
Relative humidity		10%~95%, non-condensing
Fall		1m, 110 times, transport package
Environmental conditions-work		
Temperature	Horizontal installation position	0~140°F
	Vertical installation position	0~104°F
Atmospheric pressure		1080 hPa~795 hPa (The corresponding height is -1000m~+2000m)
Relative humidity		10%~95%, non-condensing
Harsh environment Concentration of pollutants		Low salt mist, humidity, dust mist and other environmentsSO2<0.5ppm Relative humidity <60%, non-condensing H2S<0.1ppm, Relative humidity <60%, non-condensing

Getting Started

2

This chapter introduces the introduction and use of H56-10 and H52-10 based on examples, as follows:

Based on specific examples, this chapter introduces how to create a simple project containing PLC program, download the program to the target equipment, run and monitor the program.

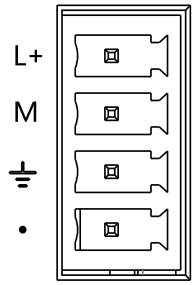
Write an example program in LAD language: when the power is on for the first time, the timer T33 starts timing and stops timing after 1 second. Timer T37 starts timing after T33 stops, and stops timing after 1s. After that, timer T33 and timer T37 perform alternating timing operation in a 1-second interval.

2.1 Connecting the Main Control Module

Connect the programming equipment with PLC through communication cable (such as standard network cable), and then supply power to PLC.

❑ **Supply power to PLC**

Schematic diagram of POWER terminal of PLC:



Warning

Before installing or removing H56-10 and H52-10 motion controllers, corresponding safety protection specifications must be observed and their power supply must be disconnected.

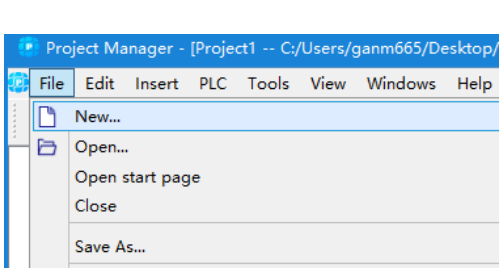
❑ **Connecting cable**

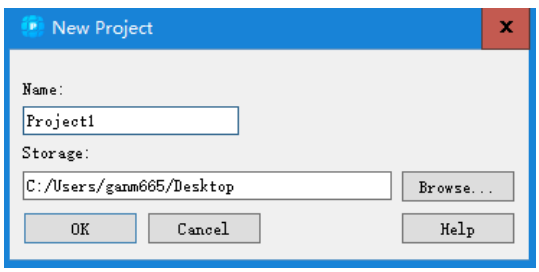
Take H56-10 as an example, connect the Ethernet communication port between the programming device and H56-10 with a standard network cable, or connect the RS485 communication port between the programming device and H56-10 with a programming cable.

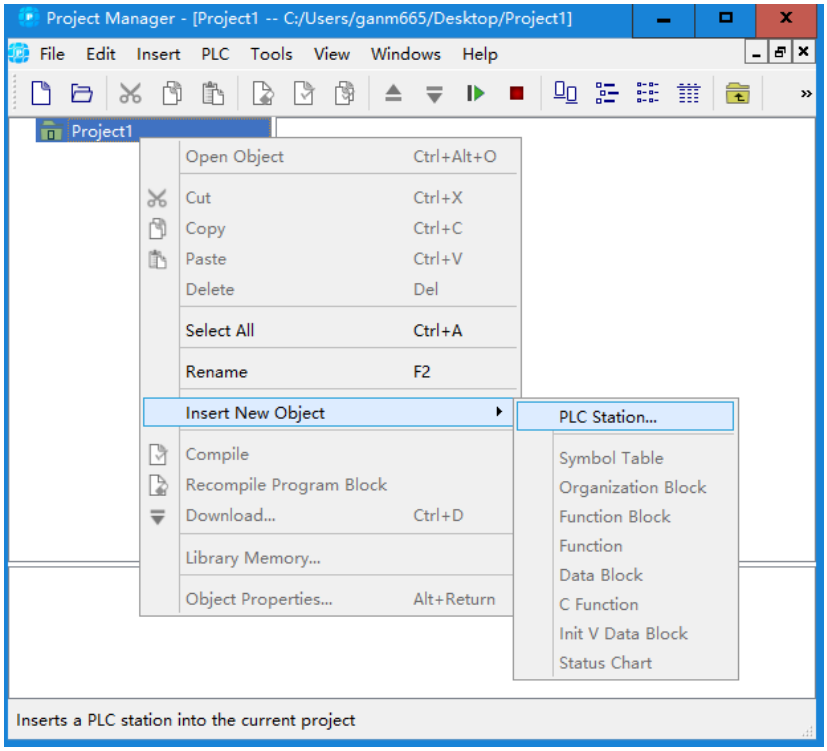
Note: Please refer to chapter 5.3.3 Make Standard Network Cables making standard network cable.

2.2 Communicate with H56-10/52-10

Open MagicWorks PLC software, select the menu item "file" → "New" to create a new project, and then right-click the new project and choose "Insert new object" → "PLC" to insert a H56-10 in the project.

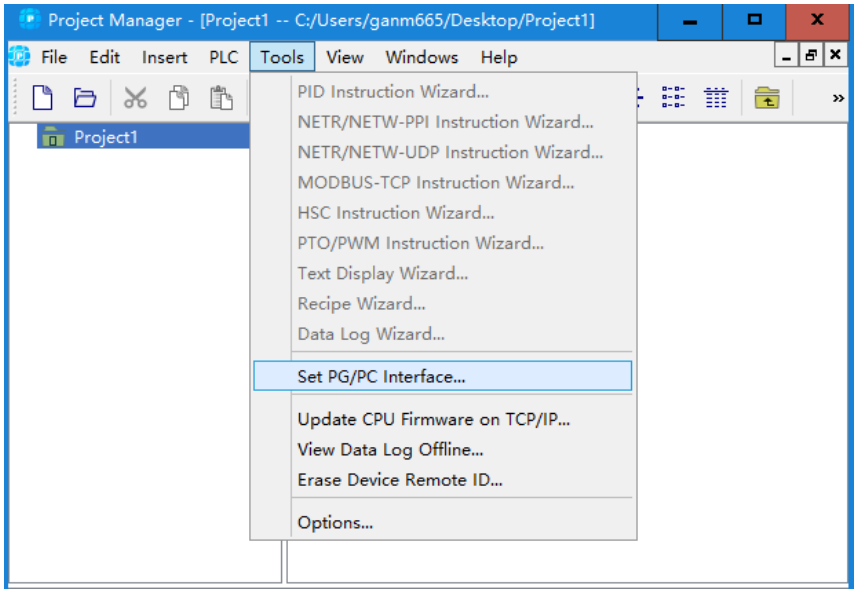


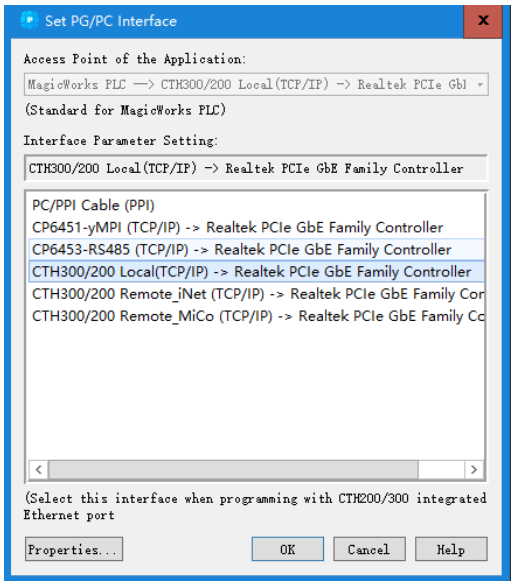




1. Set the PG/PC interface

Select the menu item "Tools" → "Set PG/PC Interface". If you use a serial port connection, select "PC/PPI Cable (PPI)". if you use a standard network cable connection, select "CTH300/200 Local (TCP/IP) → Realtek PCIe GBE Family Controller", and then click "OK".



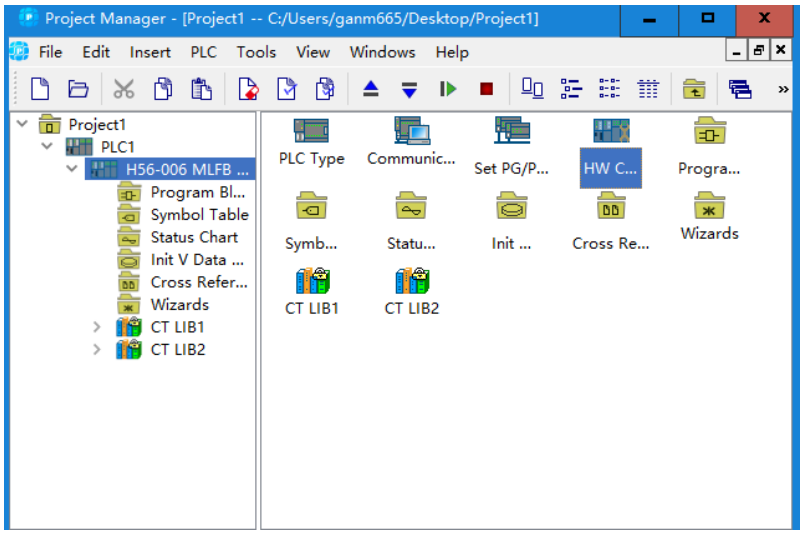


2. Set the IP of the programming device and the IP of H56-10 to the same LAN

Search the H56-10 connected to the programming equipment through magicworks PLC, the IP address of the H56-10 can be recorded, and the IP address of the local connection currently used by the programming equipment and the IP address of the H56-10 can be set to the same network segment.

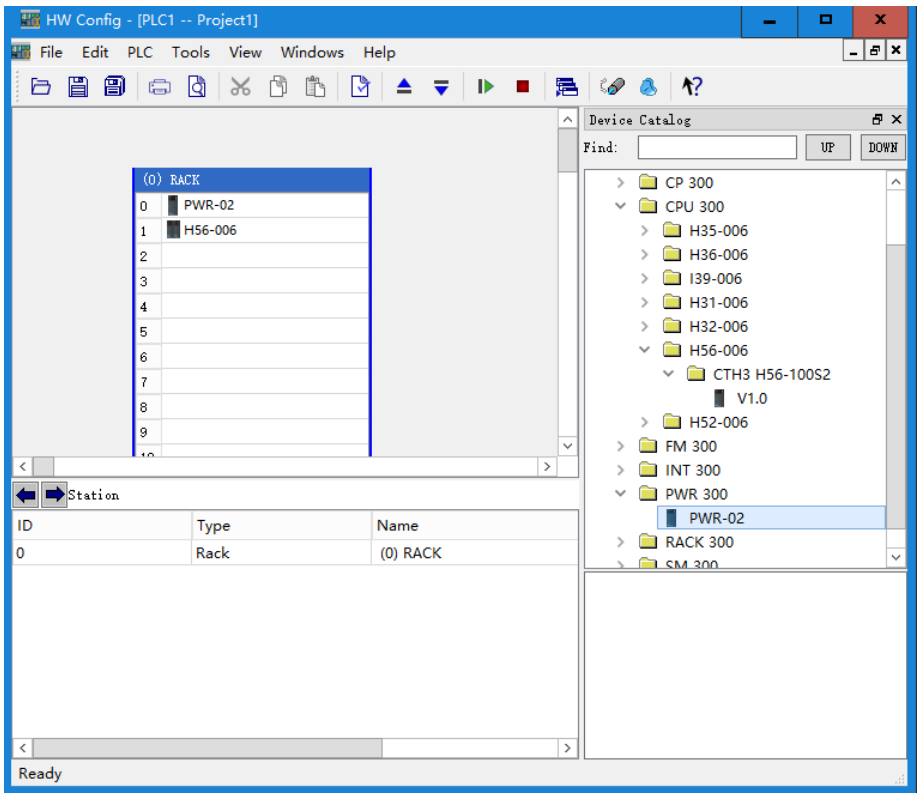
2.3 HW Config

Select "HW Config" in the "project manager" interface of CPU site to configure the hardware.

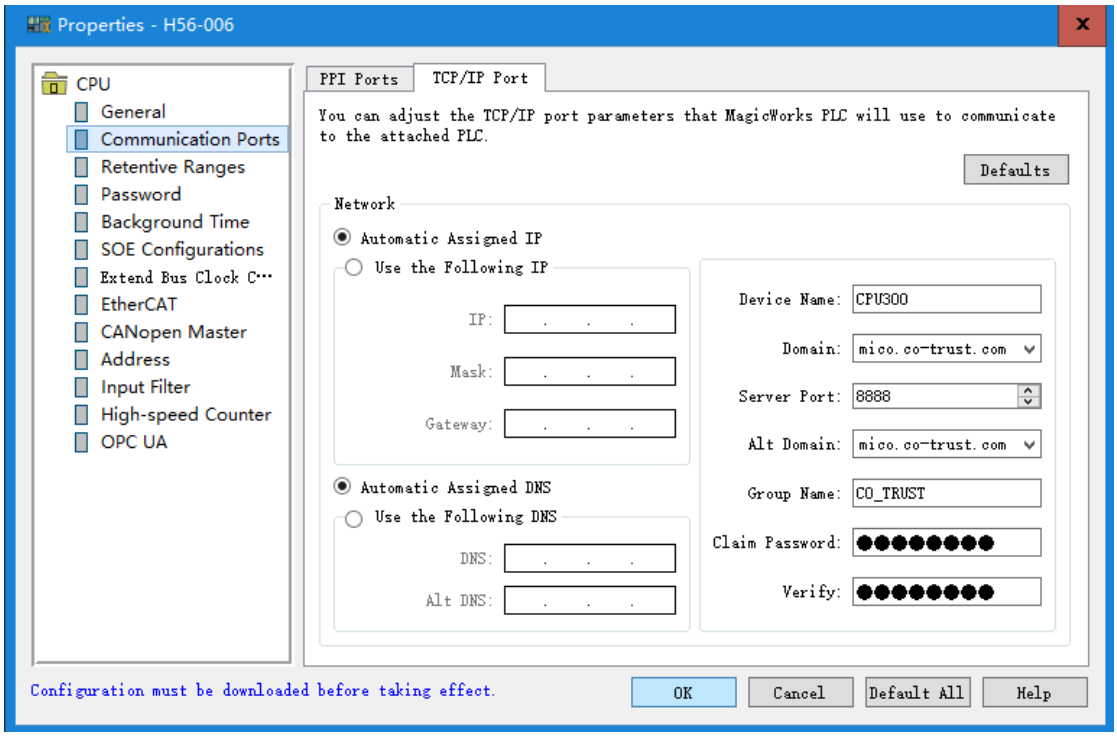


Step 1: add rack, power module and CPU

In the hardware configuration interface, expand the "PWR 300" and "CPU 300" folders in sequence under the "co-trust 300" item tree, and add the power module and CPU. The power module can only be placed in slot 0 of the rack, and the CPU can only be placed in slot 1 of the rack, as shown in the figure below.

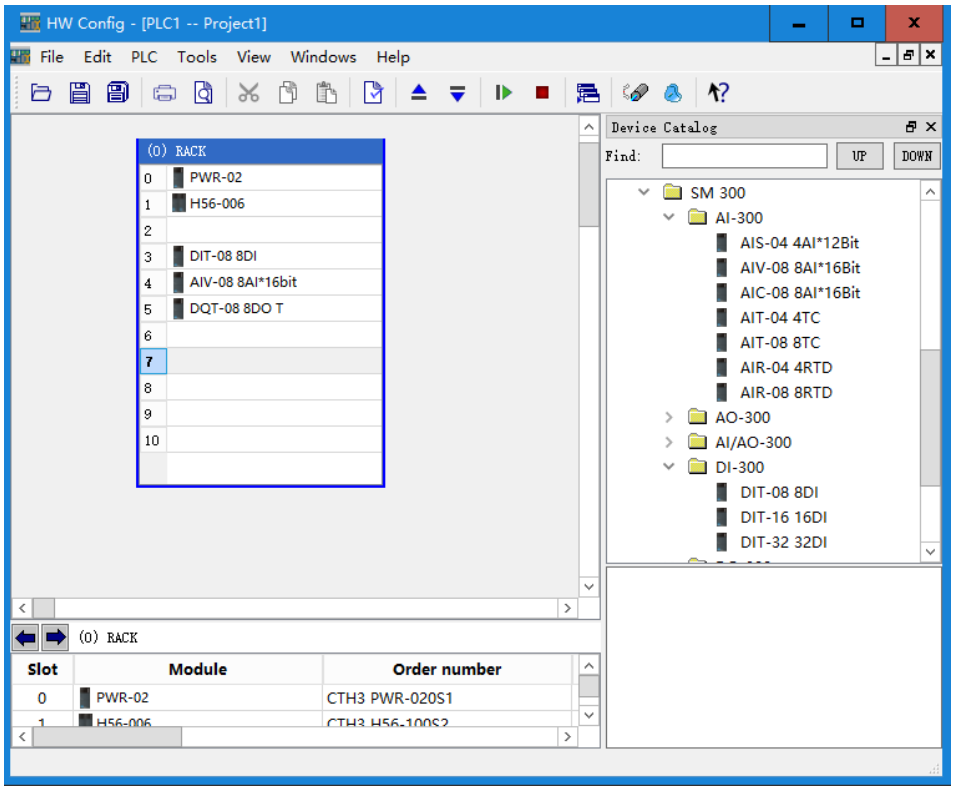


Double click "H56" on the rack to configure CPU properties, and select "Communication Ports" in the properties interface to set communication port properties.

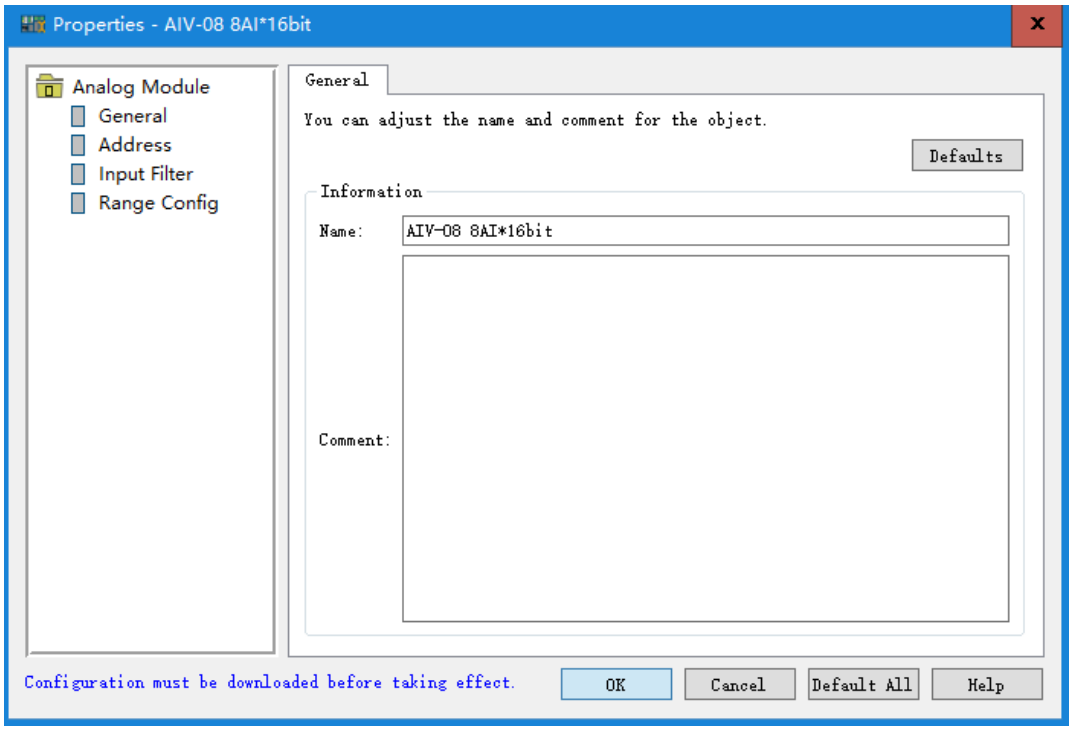


Step 2: add functional modules

Expand the "SM 300" item tree and select the required module to add to the rack, as shown in the following figure.

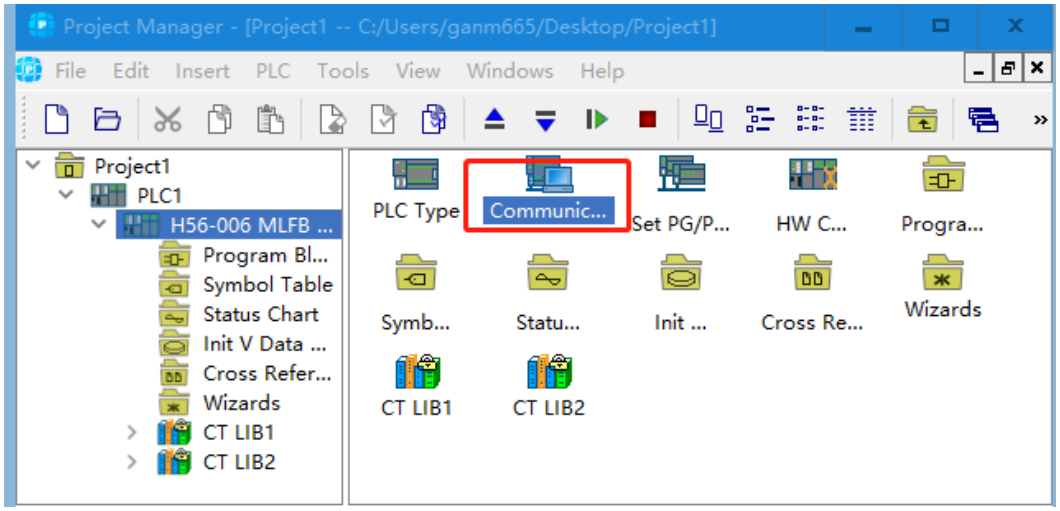


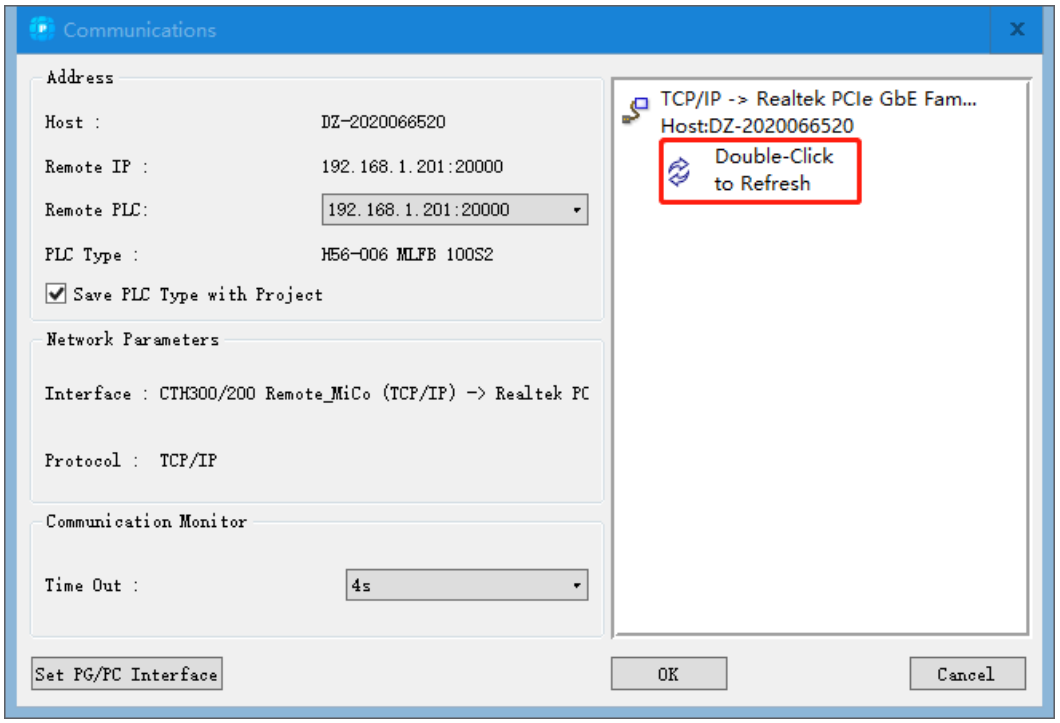
Double click the module on the rack to configure the module properties



Establish communication with CPU

Double click the “communication“, then double-click the “Double-click to Refresh“ to search CPU, and the CPU with successful search will be displayed in the communication dialog box.



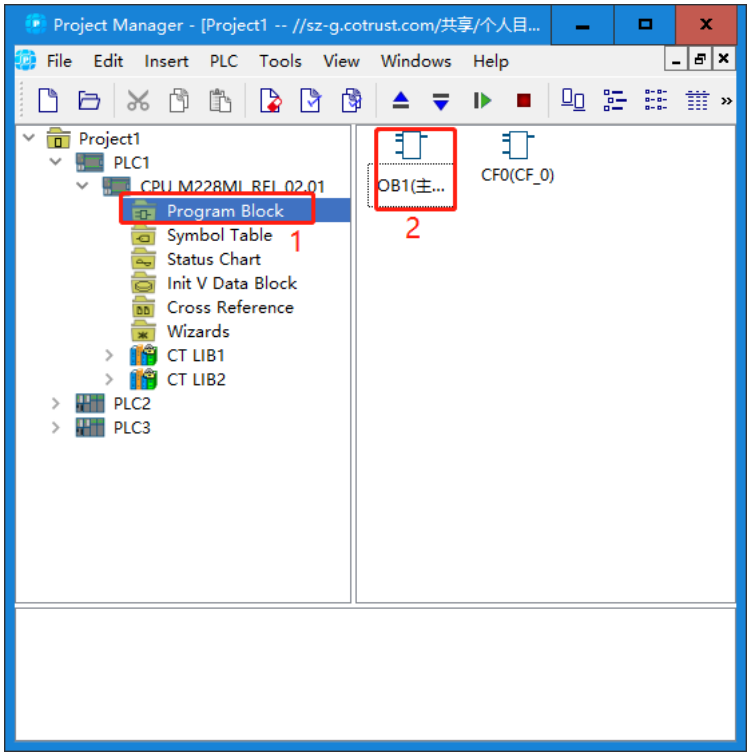


After setting, you can compile, download and run the CPU through the programming device.

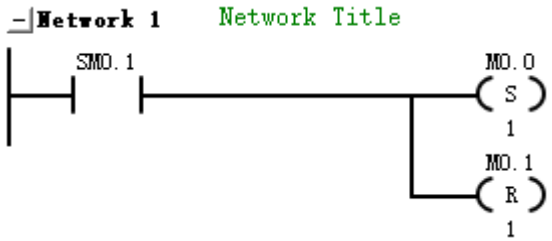
2.4 Programming

This example program achieves the purpose: Timer T33 and timer T37 alternately count in 1 second intervals.

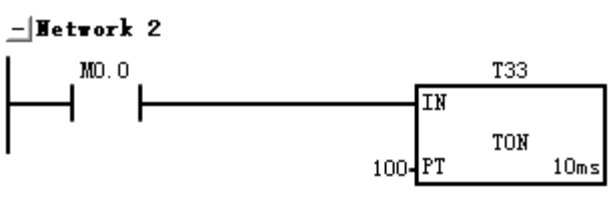
1) Expand "H56-10" in the project tree, then select the sub-item "Program Block", and double-click OB1 in the right window to enter the program editing window.



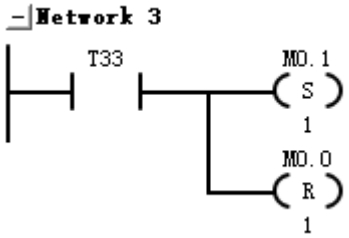
2) Using LAD language programming, the final program is as shown in the figure below:
When power on for the first time, m0.1 is set to 1, m0.1 is reset.



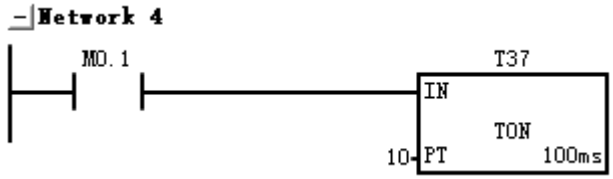
After M0.0 is set, execute T33.



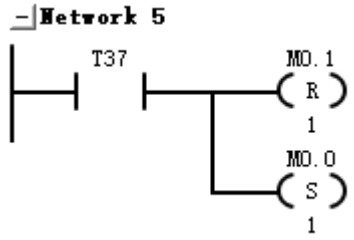
After T33 arrives, M0.1 is set and M0.0 is reset.



After M0.1 is set, start to execute T37.



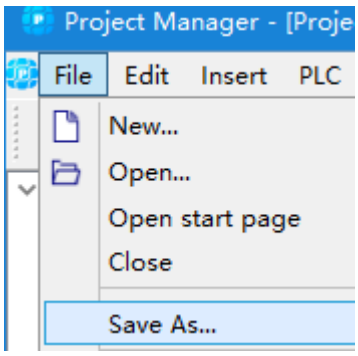
After T37 timing arrives, reset M0.1 and M0.0 is set.

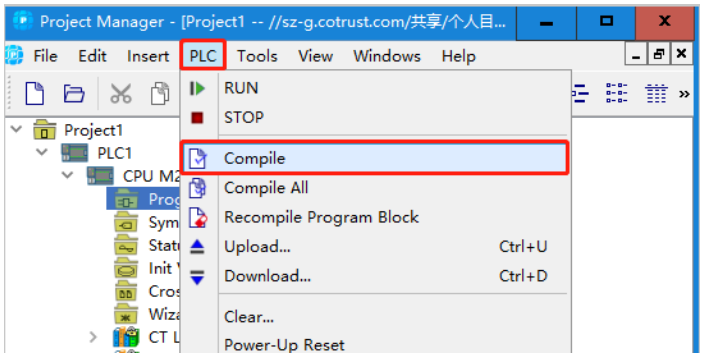


2.5 Compile and Download Program

1. Save and compile the configuration

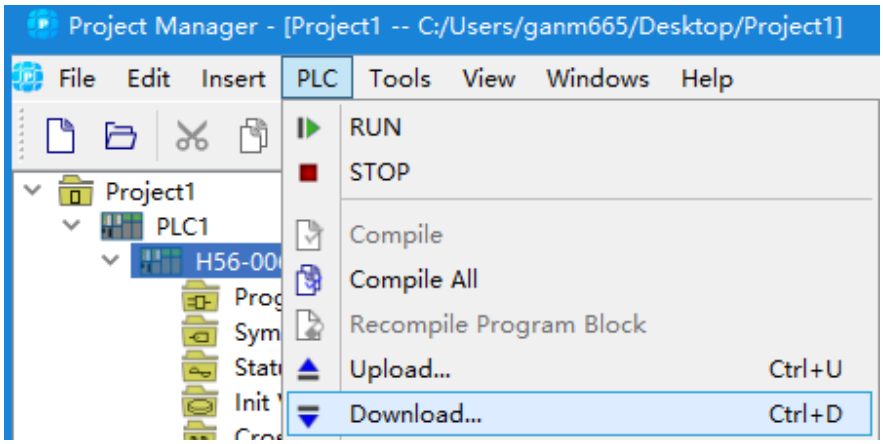
Select "File" → "Save as" to save the current configuration, and then select "PLC" → "Compile" to compile the current project. if the compilation is successful, the download operation can be carried out.





2. Download the program to the CPU

Select "PLC" → "Download" to download the program blocks and hardware configuration to the CPU. If your CPU is in running mode, a dialog box will pop up on the download interface to prompt you to put the CPU in STOP mode.



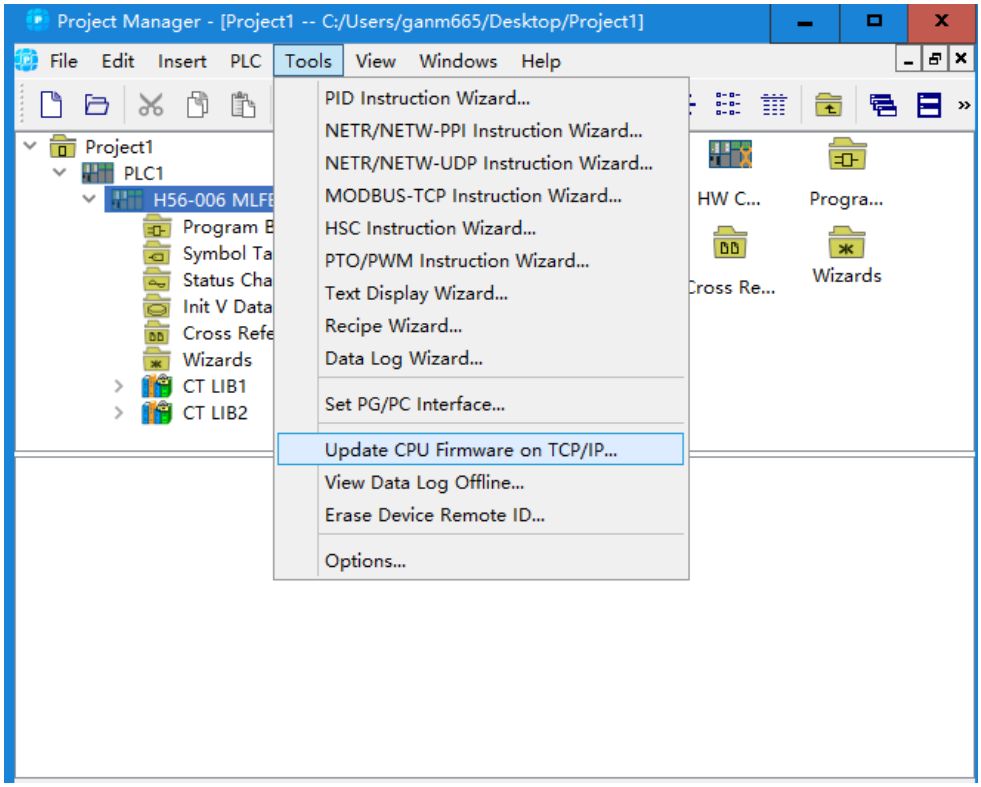
2.6 Run the Program

After the program is successfully downloaded to the CPU, turn the system operation switch of the CPU to RUN, and then you can monitor the operation of the program through the status table.

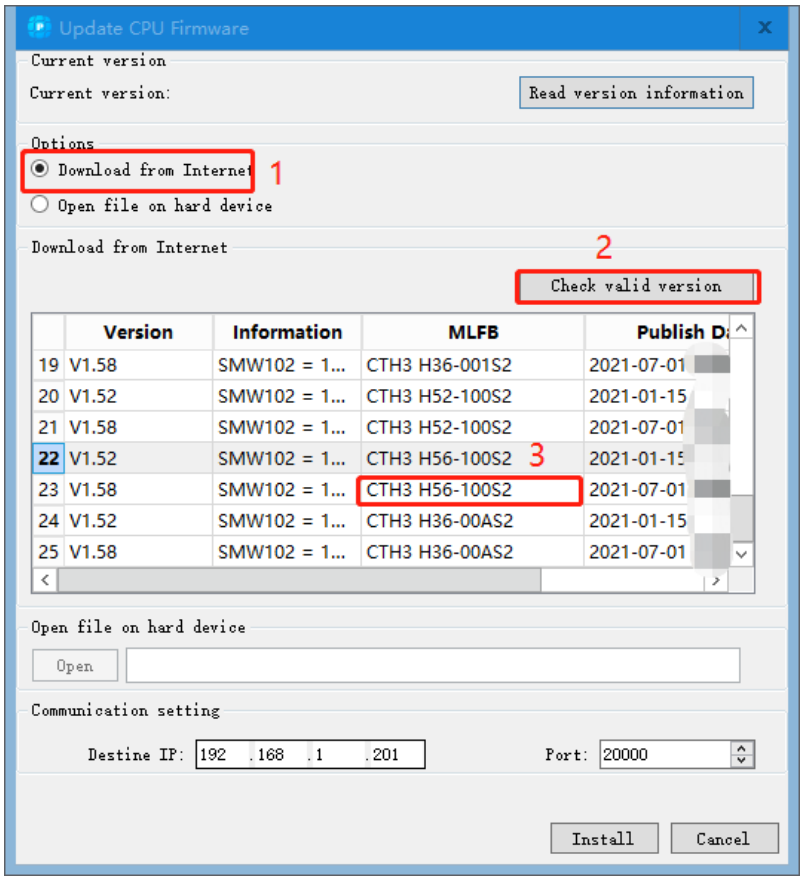
<Remarks> When the system fails, please refer to chapter 5.8 PU Fault Diagnosis to obtain the fault diagnosis method of the motion controller.

2.7 Online Firmware Upgrade

1. CTH300-H PLC supports online firmware update. Before performing online firmware update, ensure that the CPU is connected to the network. In the programming software project manager interface, select "Tools" → "Update CPU Firmware (TCP/IP)"



2. In the update CPU firmware interface, select "Download from Internet", click "Check valid version", the information window will display the current available PLC firmware, select the firmware to be updated and click "Install", the firmware will be automatically downloaded from the Internet.



3. Set PLC to STOP mode

4. After loading, you need to wait for the PLC to restart automatically, and click "OK". Do not operate the PLC or disconnect the power supply before the PLC firmware is loaded!

5. After downloading, the PLC needs to be powered off and restarted.

Installation

3

The installation methods of H56-10 and H52-10 motion controllers are introduced as follows:

3.1 Safety Guidelines

You can use the mounting holes to fix the H56-10/H52-10 motion controller on the back panel of the control cabinet, or use the DIN clip of the device to fix the module on a standard (DIN) rail.

Installation precautions of H56-10/H52-10 motion controller:

❑ Insulate PLC from heating device, high voltage and electronic noise

When installing equipment components, separate the equipment generating high voltage and high electronic noise from the low-voltage electronic equipment of the H56-10/H52-10 motion controller.

When installing the H56-10/H52-10 motion controller on the back panel of the control cabinet, you should consider arranging the electronic components in the lower temperature area of the control cabinet, because the long-term operation of the electronic components in a high temperature environment will shorten the Time to failure.

❑ Leave proper space for heat dissipation and wiring

The controller adopts natural convection heat dissipation, and there must be at least 60mm space above and below the module to facilitate normal heat dissipation.

Notice

The maximum temperature allowed in the vertical installation is 10°C lower than that in the horizontal installation, and the CPU should be installed under all expansion modules.

When installing the controller, sufficient space is required for wiring and connecting communication cables.

Figure 3-1 shows the CPU installed on multiple racks, which shows the distance between each rack and adjacent components, cable troughs, and cabinets. When wiring the module through the cable trough, the minimum distance between the bottom of the shield connection element and the cable trough is 60mm.

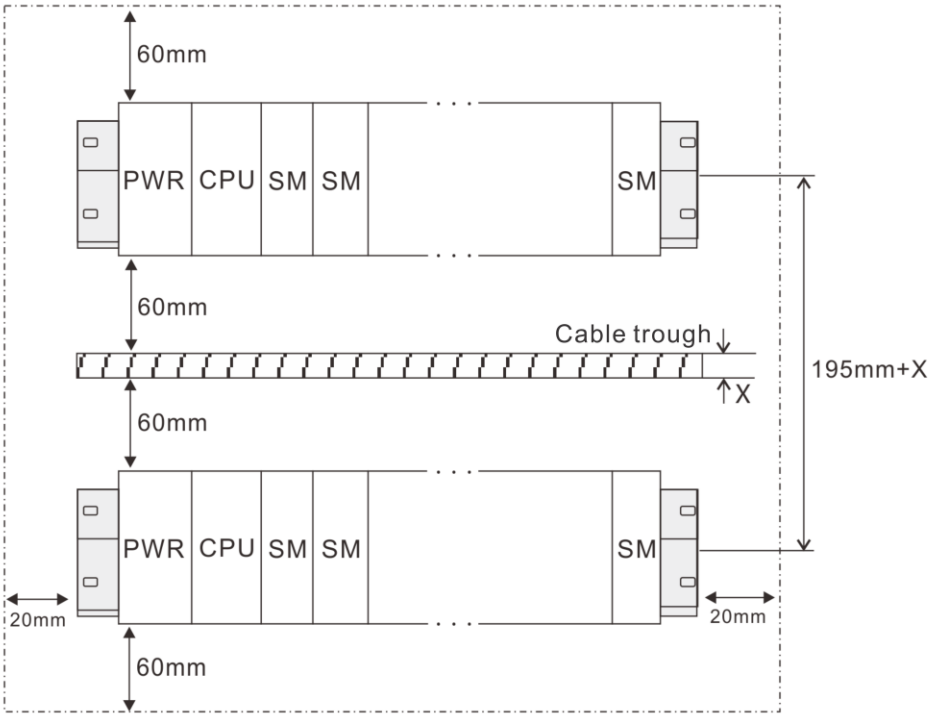


Figure 3-1 Installation diagram

❑ Power budget

After confirming the CPU, power supply module, relay module and expansion module of each rack, it is also necessary to confirm whether the current consumption and power consumption of the system Bus meet the following conditions:

Condition 1: Confirmation of Bus current consumption

The internal bus voltage is 5VDC, and the current is provided by CPU. The sum of bus current consumption of expansion module on each rack shall not exceed the maximum bus current allowed by CPU or relay module.

Condition 2: Power consumption confirmation

When using power modules, the sum of the power consumption of other modules in each rack cannot exceed the maximum power consumption allowed by the power module.

When using an external power supply, select the appropriate power model according to the sum of the connected power.

<Remarks> For the budget of the Bus power current of the controller, please refer to A Power budget.

❑ Device power off

When installing and disassembling the controller and related equipment, appropriate safety measures must be taken in advance and the power supply of the controller must be cut off.

Warning

Installing or disassembling the H56-10/H52-10 motion controller and related equipment under power-on conditions may cause electric shock or equipment malfunction, further causing serious personal injury or even death and equipment damage!

When replacing or installing the H56-10/H52-10 motion controller, be sure to use the correct or equivalent module. When replacing the H56-10/H52-10 motion controller, in addition to using the same module, make sure that the installation direction and position are correct

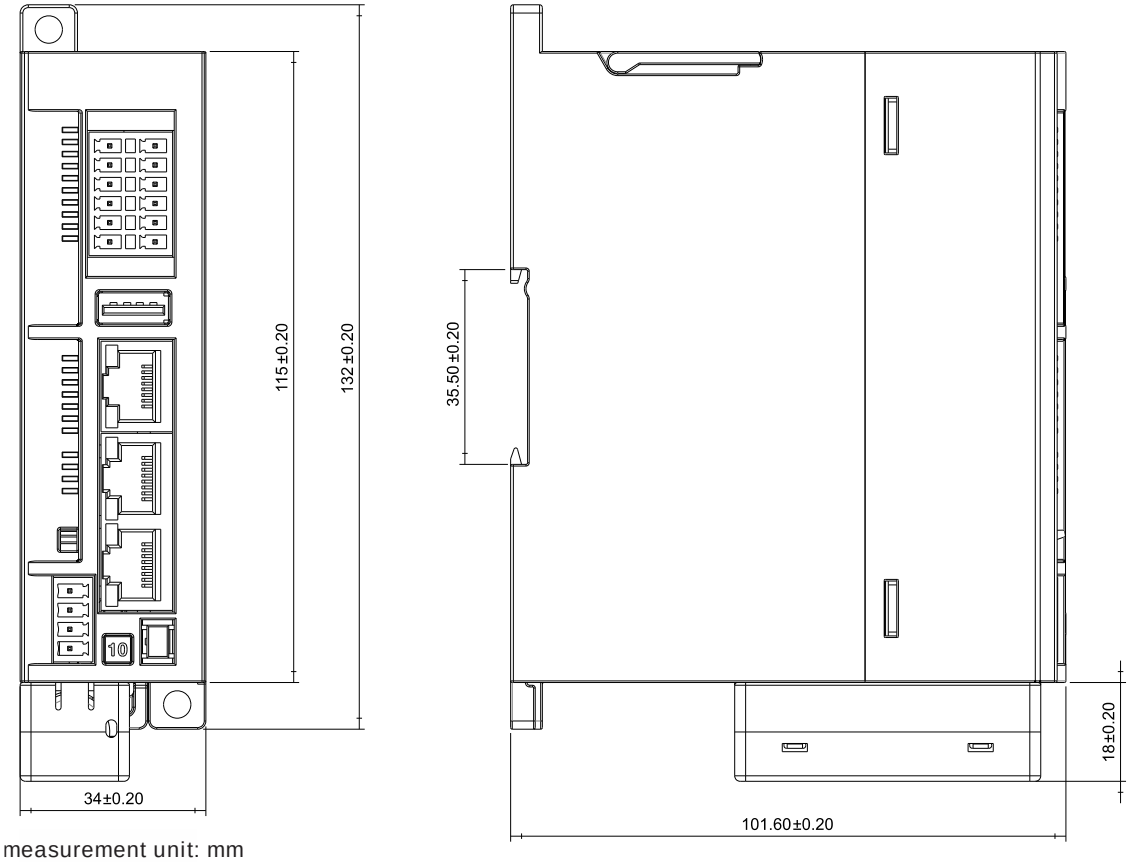
Notice

● If you install an incorrect module, the program of the H56-10/H52-10 motion controller may produce incorrect functions.

● Failure to use the same module to replace the H56-10/H52-10 motion controller in the same direction and sequence may cause serious personal injury and equipment damage.

3.2 Installation Dimension

H56-10/H52-10 motion controllers have mounting holes, which can be easily installed on the back panel. The installation dimensions are shown in figure 3-2.



measurement unit: mm

Figure 3-2 H56-10/H52-10 motion controller installation dimensions

3.3 Use of the Rack

❑ Central rack (Rack0) and expansion rack (Rack1\Rack2\Rack3)

The H56-10/H52-10 application system consists of a central unit and one or more expansion modules. The rack containing the CPU is the central unit. Equipped with modules and connected

to the central rack to form the expansion rack of the system.

❑ Use of expansion racks

If all slots of the central rack are used, you can use an expansion rack.

When using expansion racks, in addition to additional racks and relay modules (INT), more power supply modules may be required. When using a intermediate extension module, it must be compatible with the extension station.

<Remarks> H56-10 supports expansion racks, H52-10 only supports central racks

❑ Module layout on the rack

The rack is an assembly rail. This guide rail can be used to install the module to which the H56-10/H52-10 application system belongs.

① Module layout on a rack

If you want to install the module on a rack, please note:

- The number of modules installed on the right of the CPU does not exceed eight (SM, FM, CP).
- The cumulative power consumption of the modules on the rack on the H56-10/H52-10 backplane bus must not exceed 1.6A.

The following figure shows the layout of eight signal modules in the H56-10/H52-10 system.

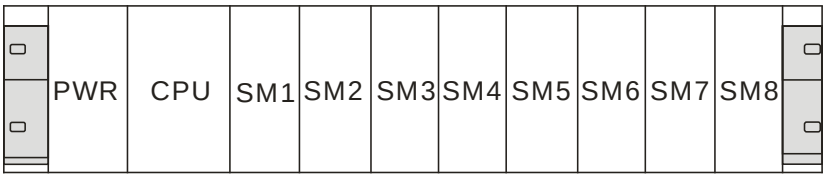


Figure 3-3 H56-10/H52-10 application system layout

② Complete assembly of four racks

If you plan to perform assembly on multiple racks, you need to use a intermediate expansion module (INT), and the different racks are connected through the network port of the relay module.

If you want to arrange modules on multiple racks, please pay attention to the following:

- INT module always uses slot 3 (slot 1: PWR. slot 2: CPU, slot 3: INT)
- It is always on the left before inserting the first signal module.
- A maximum of 8 modules (SM, FM, CP) can be installed on each rack.
- The number of modules (SM, FM, CP) is limited by the allowable current consumption on the H56-10/H52-10 bus. The cumulative power consumption of each rack must not exceed 1600 mA.

The following figure shows the arrangement of each module in the H56-10 system on 4 racks.

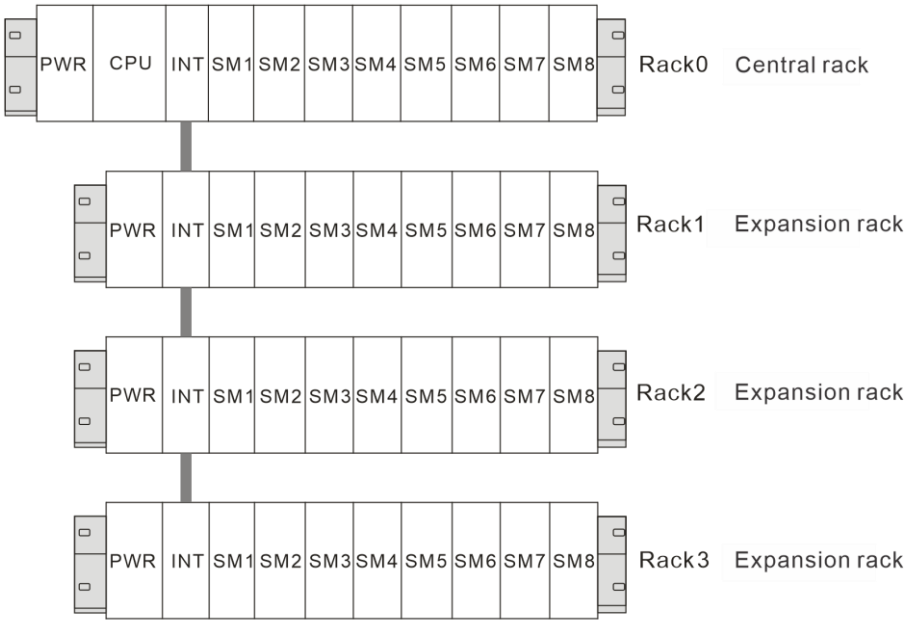


Figure 3-4 H56-10 application system module layout with 4 racks

Tips

When a multi-rack system uses Intermediate expansion module, only the Intermediate expansion module in the first rack need to be connected to the CPU through the bus, and other Intermediate expansion module need to be connected through network ports. When connecting Intermediate expansion module through network ports, pay attention to the IN/OUT port sequence. That is, the OUT port on the previous Intermediate expansion module is connected to the IN port on the next Intermediate expansion module.

3.4 Installation Method

Installation mode

H56-10 /H52-10 motion controller can be installed on the back plate of control cabinet or standard DIN rail. It can be installed horizontally or vertically.

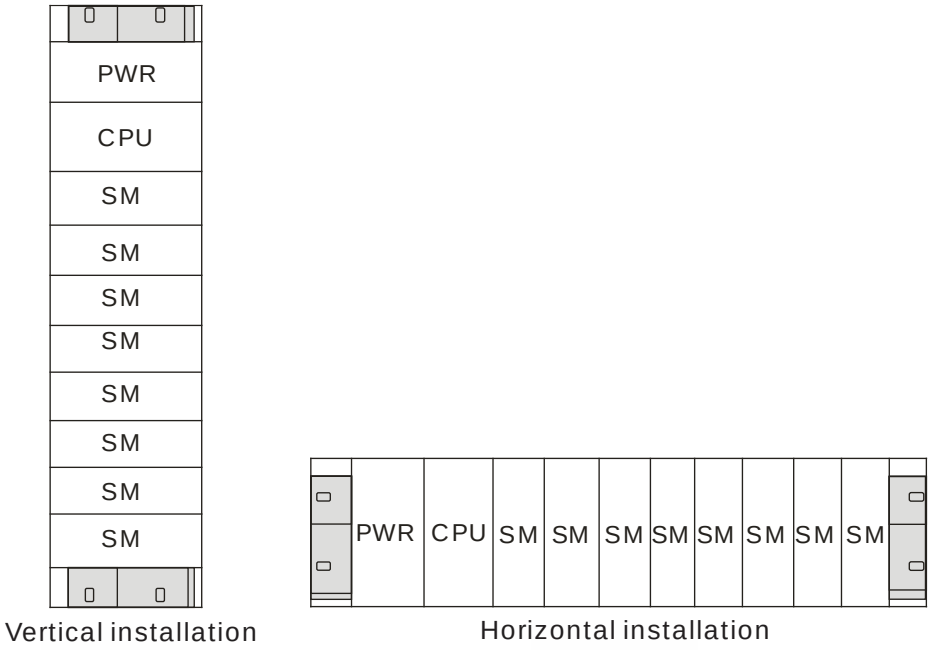


Figure 3-5 CPU and module installation method

Installation and removal operations

Please install or remove the H56-10 / H52-10 motion controller as follows.

- **Install the panel**

1) Position and punch holes according to size requirements.

2) Fix the module on the backplane with appropriate screws.

3) If an expansion module is used, connect the flat cable of the expansion module to the expansion port under the front cover.

- **DIN rail mounting**

1) Fix the guide rail on the back plate.

2) Open the din clip at the bottom of the module and clamp the back of the module on the DIN rail.

3) If an expansion module is used, connect the flat cable of the expansion module to the expansion port under the front cover.

4) Rotate the module close to the DIN rail and close the din clip.

5) Carefully check whether the din clip and DIN rail on the module are tightly fixed.

6) To avoid damage to the module, do not directly press the front of the module, but press the part of the mounting hole.

Be careful

When H56-10 / H52-10 motion controller is used in the service environment with large vibration or vertical installation, DIN rail stop shall be used.If the system is in a high vibration environment, a higher vibration protection.

- **Remove the CPU or expansion module**

1）Remove the power supply of H56-10 / H52-10 motion controller.

2）Remove all wires and cables from the module.

3）If other expansion modules are connected to the module, open the front cover and unplug the expansion flat cable of the adjacent module.

4）Remove the mounting screw or open the din clip.

5）Remove the module, removing and installing the terminal strip.

- **Installation of terminal strip**

1）Ensure that the pin on the module is aligned with the small hole at the edge of the terminal block.

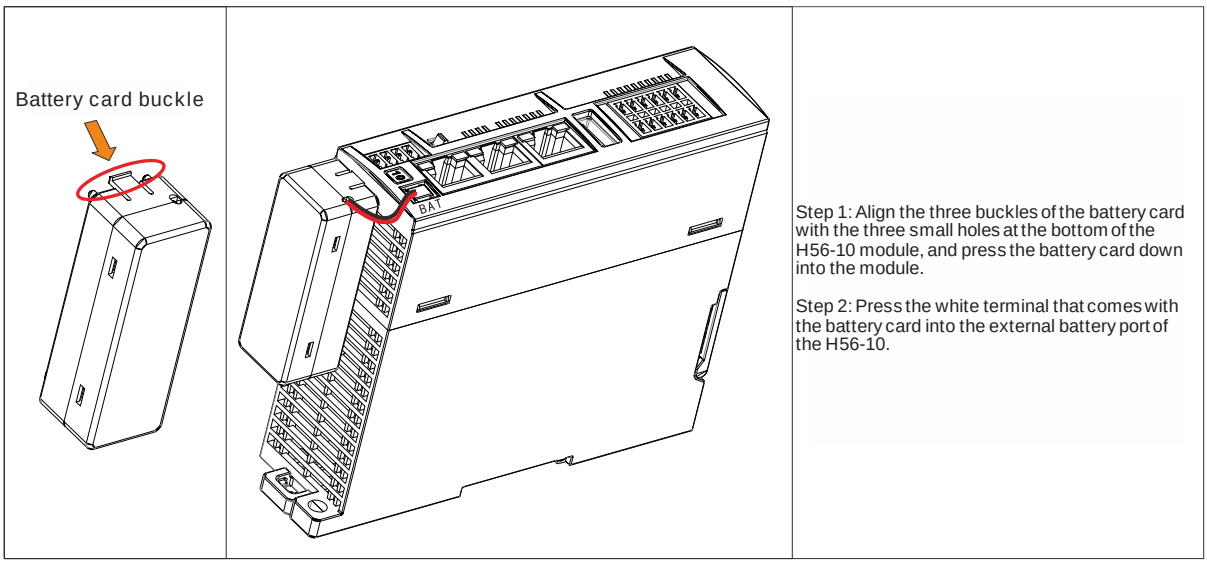
2）Press the terminal block down into the module to ensure that the terminal block is aligned and locked.

- **Removal of terminal strip**

Hold the terminal block and pull it up.

- **Installation of battery card**

Align the three snaps of the battery card with the three small holes at the bottom of the H56-10 / H52-10 module, ensure that the three snaps of the battery card are aligned with the position, and then press the battery card down into the module.Finally, press the white terminal on the battery card into the external battery port of H56-10 / H52-10.



Wiring

Wiring precautions:

Be careful

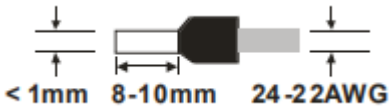
- When installing or wiring, ensure that all external power supplies are turned off.Failure to turn off all power may cause electric shock to the user or damage to the product.
- After installation or wiring, when starting the power supply or operating the PLC, confirm whether the wiring terminal block is correctly inserted into the corresponding terminal base of the PLC. Otherwise, it may cause electric shock or work error.

• When wiring the PLC, check the rated voltage and terminal configuration defined in the product specification to ensure correct and safe wiring.Connecting a power supply that does not match the rated value or incorrect product safety wiring may cause dangerous conditions such as fire or damage.
• For external wiring configuration, special tools shall be used for flanging, pressure welding and correct welding.Poor wiring configuration may cause short circuit, fire, or incorrect operation.
• It must be ensured that there are no foreign matters such as scrap iron or wiring residues in each module.These foreign objects may cause fire, damage, or incorrect operation.

Wiring Description:

1) Terminal blocks with crimped insulating sleeves shall not be used.It is recommended to wrap the crimp terminal lugs with sleeves containing labels or insulating materials.

2) Please use 24~22AWG single core wire or multi core wire for wiring connecting the terminal block. It is recommended to match the pin terminal with aperture less than 1mm for wiring. The specifications are shown in the following figure:



3.5 Grounding and Wiring

❑ H56-10 / H52-10 motion controller grounding and wiring Guide

Reasonable grounding and wiring are very important for all electrical equipment. It can ensure that your system has the best operating characteristics and provide better electronic noise protection for your system.

The wiring of H56-10 / H52-10 motion controller and its related equipment shall comply with all effective electrical coding rules.All equipment installed and operated shall comply with all valid national or regional standards.Keep in touch with authoritative representatives of the region to determine standards that meet special needs.

Safety factors must be considered when designing the grounding and wiring of H56-10 / H52-10 system, otherwise it may cause misoperation of equipment.Therefore, you should implement all safety regulations to avoid personal injury and equipment damage.

Warning

1. Attempting to conduct grounding or wiring under live conditions may cause death or serious personal injury and equipment damage.
2. The control equipment may cause the misoperation of the equipment it controls, resulting in death or serious personal injury and equipment damage.Therefore, H56-10 / H52-10 system must have emergency stop function, electromechanical interlock or other redundant safety facilities independent of H56-10 / H52-10 motion controller.

3.6 Suppression Circuit

When using inductive load, a suppression circuit should be added to limit the rise of voltage when the output is turned off.The suppression circuit can protect the output point from premature damage due to high inductive reactance switching current.In addition, the suppression circuit can also limit the electronic noise generated by inductive load switching.

Be careful

The effectiveness of the suppression circuit depends on the application, and you should adjust its parameters to suit your special application.Ensure that all device parameters are consistent with the actual application.

❑ Transistor output and relay output for controlling DC load

The transistor output has internal protection and can be adapted to a variety of applications.Since the relay type output can be connected to both DC load and AC load, there is no internal protection.

Figure 3-6 shows an example of a DC load suppression circuit. In most applications, an additional diode A is sufficient, but if your application requires faster shutdown speed, it is recommended that you add zener diode B.Ensure that the zener diode can meet the current requirements of the output circuit.

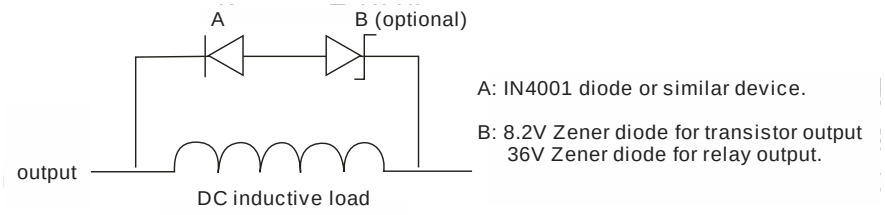


Figure 3-6 suppression circuit of DC load

❑ AC output and relay output controlling AC load

The AC output has internal protection for most applications because the relay can be used for DC or AC loads, so it does not provide internal protection.

Figure 3-7 shows an example of an AC load suppression circuit. In most applications, an additional metal oxide variable resistor (MOV) can limit the peak voltage to protect the internal circuit of the H56-10 / H52-10 motion controller.Ensure that the operating voltage of MOV is at least 20% higher than the normal line voltage.

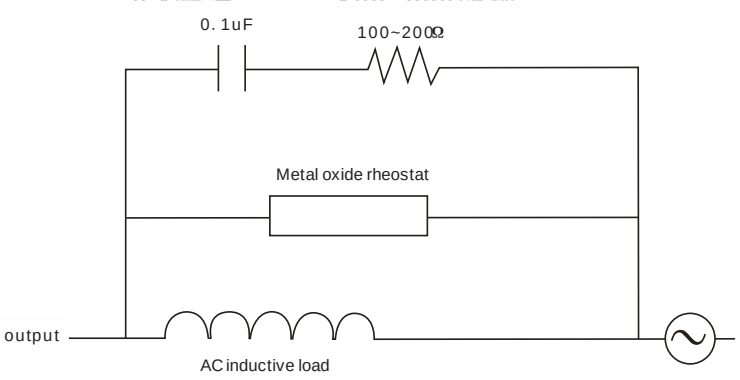


Figure 3-7 suppression circuit of AC load

Power Supply Module

4

Introduce the power module of H56-10 / H52-10 application system, as follows:

4.1 Technical Specifications

The PWR-02 power supply module can provide 24V DC for H56-10/H52-10 and expansion modules (except digital modules). Please select a power supply module for each rack, and select other power supplies for digital input and output and sensors.

Table 4-1 Basic properties of PWR-02

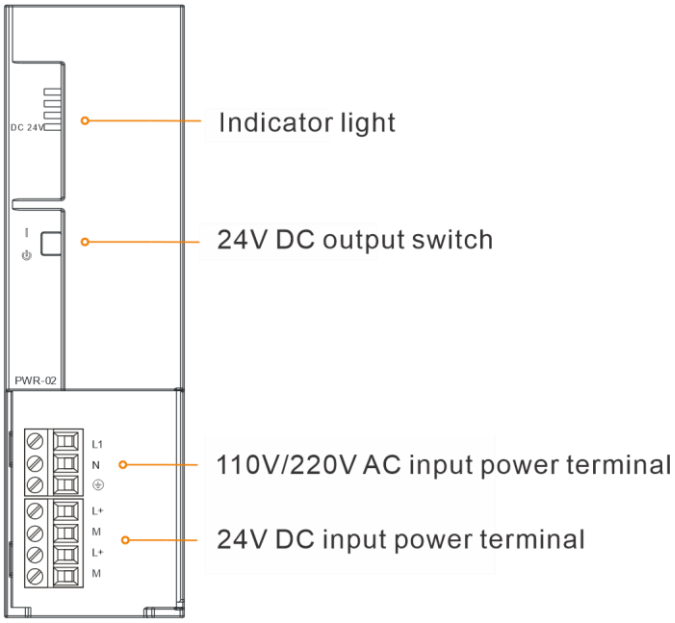
Name	Descreption	Order no.
PWR-02	Input: 85~264V AC output:24V DC/2A	CTH3 PWR-020S1

Table 4-2 Specification of PWR-02

Physical characteristics	
dimensions(W×H×D)	34×115×101.6 mm
LED indicator	
24VPOWER(green)	ON: with 24V DC output, OFF: without 24V DC output
Switching	
24V DC power control switch	ON: with 24V DC output, OFF: without 24V DC output
Input voltage	
Voltage range	85~264VAC, wide voltage input
Rated frequency	50Hz/60Hz
Frequency Range	47Hz~63Hz
Efficient	75%
Alternating current	0.9A/110V, 0.5A/220V
Inrush current (25℃ max)	≤20A/110V, ≤35A/220V
Leakage current	≤5mA/220VAC
Output voltage	
DC voltage / rated current	24VDC/2A
Rated power	48W
Ripple and noise (maximum)	150mVp-p
Voltage output range	±5%
Start / rise / hold time	≤2.5s/≤50ms/≥20ms
Isolation (power input and output)	Isolation between 110V / 220V AC and 24V DC
Protection function	
Overload protection	105%~130% of the rated output power, cut off the output, and automatically recover after troubleshooting.
Overvoltage protection	115%~135%Ue. Protection mode: hiccup mode, automatic recovery after troubleshooting.
Surge protection	The power supply terminal provides surge absorption function
Overcurrent protection	The power output provides overcurrent protection.
Safety electromagnetic compatibility	
Withstand voltage	Input ~ output: 1.5kvdc, input PE: 1.5kVDC, output-PE: 500VDC
Isolation resistance	Input ~ output, input-PE, output-PE: 100M Ω / 500VDC.
Standard basis	Refer to UL60950 and UL1950 for safety and EN55022 for electromagnetic compatibility.

4.2 Wiring

4.2.1 Interface Diagram



4.2.2 Interface Definition

Table 4-4 Definition of the 240V AC input power port of the PWR-02

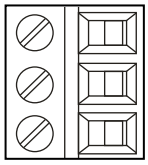
Removable terminal	Signal	Definition
	L	FireWire
	N	Zero line
	⏚	Grounded

Table 4-5 Definition of the 24V DC output power port of the PWR-02

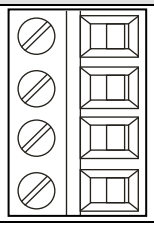
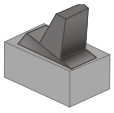
Removable terminal	Signal	Definition
	L+	24V+
	M	24V-
	L+	24V+
	M	24V-

Table 4-6 DIP switch definition of PWR-02

Switch	Status	Direction	Definition
	ON	UP	with 24V DC output
	OFF	DOWN	without 24V DC output

Main Control Module

5

Introduce the performance specifications and main functions of H56-10 / H52-10 motion controller, as follows:

The H56-10 and H52-10 main control module is the central processing unit of the application system, as the core control device of the system, used to process various operations and the operation of user programs.

Table 5-1 Basic characteristics of the main control module

Name	Specification Description	Order No.
H56-10 motion controller	H56-10 motion controller, 256KB+64KB program space, 1MB data space, 24V DC power supply, 10 digital inputs, 6*500kHz high-speed counter, 1 EtherNET interface, 1 EtherCAT interface, 1 CAN interface, 1 RS485 interface, 1 USB interface. Supports 64 EtherCAT slave , supports up to 4 racks, supports single-axis motion control, interpolation, electronic cams and electronic gears, and supports C language programming.	CTH3 H56-100S2
H52-10 motion controller	H52-10 motion controller, 256KB+64KB program space, 1MB data space, 32K data block space, power-down retention. 24V DC power supply, 10 digital inputs, 6*500kHz high-speed counter, 1 EtherNET interface, 1 EtherCAT Interface, 1 CAN interface, 1 RS485 interface, 1 USB interface. Support single-axis control functions (such as positioning, speed and return to the original), support linear/arc interpolation function, support continuous interpolation function, support electronic cams and electronic gears support C language programming.	CTH3 H52-100S2

5.1 Basic Performance Parameters

Table 5-2 Regular feature

Physical characteristics	
Dimension(W×H×D)	34×115×100 mm
Power loss	19.2W
Power characteristics	
Rated input voltage	24V DC
Input voltage range	20.4V~28.8V DC
Input Current	0.8A
Reverse polarity protection	YES
Bus power supply voltage	+5V DC
Bus power current	1.6A
LED	
24VPOWER(green)	ON: with 24VDC. OFF: without 24VDC
SF(red)	ON: system error. OFF: No error
BF(red)	ON: BUS error. OFF: No error

FRCE (yellow)	ON: some items are forced. OFF: no item is forced (can also be controlled by the LED command)
RMC (green)	ON: The CPU communicates with the remote server successfully (EtherNET port parameters are correctly configured). OFF: CPUCommunication with the remote server failed or was disabled
RUN (green)	ON: system running. OFF: The system stops running
STOP (yellow)	ON: The system stops running. OFF: system running
Network port communication indicator	
Link1 (green)	ON: connect. OFF: not connect
SPEED1 (yellow)	ON: 100Mbps. OFF: 10Mbps
Link2 (green)	ON: connect. OFF: not connect
SPEED2 (yellow)	ON: 100Mbps. OFF: 10Mbps
I0.0~I1.1 (green)	ON: Signal input. OFF: No signal input
Extended I / O	
1 CPU supports the number of INT-00 racks	Up to 4
Number of modules supported per rack	Except for power supply, CPU, and intermediate expansion module, each rack supports up to 8 modules
Number of ECT-00 slave supported by one CPU	Up to 64
Number of modules supported per ECT-00	Up to 8 modules are supported
Power down hold	
Clock power down holding time	The supercapacitor lasts for 112 hours. After H56/H52 is connected to an external battery, the power-off retention time can reach at least 1 year, and the typical value is 3 years (for the order information of the external battery, see N Product Oder Info.
Instruction performance	
Bit instruction execution speed	0.086μs/step
Floating point instruction execution speed	1.4μs/step
Memory	
User program space	256KB+64KB(confidential space)
User data space	1MB
Power down holding space	32KB

5.2 CPU Input Function

5.2.1 Digital Input

The CTH300 H56-10/H52-10 motion controller integrates 10 digital inputs. The input characteristics are shown in the table below.

Table 5-3 H56-10/H52-10 motion controller digital input characteristics

Digital input characteristics		
Native integrated IO points		10
Input type		Sink/source
Rated voltage		24V DC
Input voltage range		20.4~28.8V DC
Surge voltage		35V DC, continue 0.5s
Logic 1 signal (minimum) Logic 0 signal (maximum)		15 VDC, 2.5mA 5 VDC, 1mA
Connect 2-wire proximity switch sensor (BERO)		1mA(Maximum allowable leakage current)
Input filtering		Configurable, support 0.2us, 0.4us, 0.8us, 1.6us, 3.2us, 6.4us, 12.8us, 0.2ms, 0.4ms, 0.8ms, 1.6ms, 3.2ms, 6.4ms, 12.8ms, the default is 6.4ms
Isolation (field and logic) Isolation group		500V AC, 1 minute See wiring diagram
Simultaneously connected inputs		10
Maximum cable length shield Unshielded		500m(Standard input) 50m(Standard input) 300m(Standard input)
Pulse catch input		10
High counter	total	6
	single phase	6×500kHz
	two-phase	4×250kHz

5.2.2 High-Speed Counting Input

Table 5-4 H56-10/H52-10 motion controller high-speed counter input points

mode	**description**	**input**			
	HSC0	I0.0	I0.1	I0.2	
	HSC1	I0.3	I0.4	I0.5	
	HSC2	I0.6	I0.7		
	HSC3	I1.0	I1.1		

	HSC4	I0.2			
	HSC5	I0.5			
0	Single-phase counter with internal direction control	clock			
1		clock		reset	
2					
3	Single-phase counter with external direction control	clock	direction		
4		clock	direction	reset	
5					
6	Two-phase counter with up and down counting clock	Incremental clock	Minus clock		
7		Incremental clock	Minus clock	reset	
8					
9	A/B phase quadrature counter	clock A	clock B		
10		clock A	clock B	reset	
11					

5.3 Communication Function

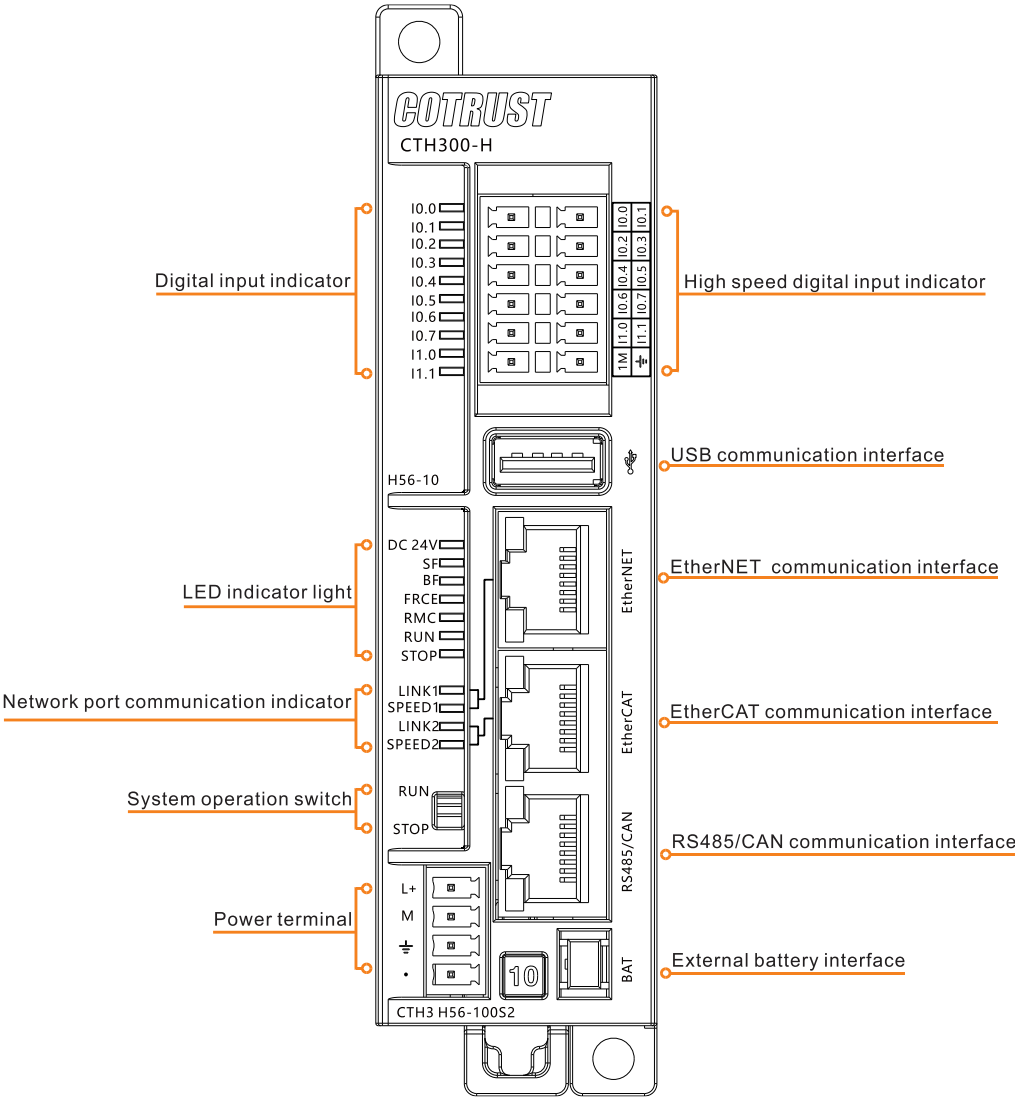
5.3.1 Communication Port Specification

Table 5-5 Communication port specifications

RS485 communication port	
Communication interface and protocol	RS485 communication port (switchable between PPI and free port protocol)
Baud rate	PPI/MPI: 9.6kbps, 19.2kbps, 187.5kbps
	Free port: 1.2kbps~115.2kbps, built-in ModBus master/slave function
Maximum cable length per section	Isolation repeater: 1000m (187.5kbps)/1200m (38.4kbps)
Maximum number of stations	32 stations per segment, 126 stations per network
Point to point	Yes (NETR/NETW), each instruction can read/write at most 200 bytes, and a maximum of 8 read and write instructions are allowed at the same time.
(PPI master mode)	One MPI slave can connect up to 8 MPI master
MPI connection	Communication port isolation
EtherNET communication port	
Communication interface	1 standard Ethernet port
Baud rate	10/100Mbps adaptive
Protocol type	UDP_PPI protocol, ModBus_TCP/IP master-slave protocol, Support S7 protocol
Maximum cable	100m

length per section							
Maximum connections per site	UDP_PPI supports up to 8 connections, ModBus_TCP supports up to 32 connections S7 protocol supports up to 8 connections, and regardless of master and slave						
Number of user data	UDP_PPI supports 200 bytes ModBus_TCP/IP supports 240 bytes The S7 protocol supports 200 bytes						
DHCP function	support						
Remote monitoring / programming function	support						
Isolation	Communication port isolation						
EtherCAT communication port							
Communication interface	1 EtherCAT communication master interface						
Baud rate	100Mbps						
Protocol type	EtherCAT Protocol						
Maximum number of secondary sites	H56: Each master supports up to 64 EtherCAT slave H52: Each master supports up to 8 EtherCAT slave						
Maximum cable length for each section	100m						
Supported functions	Configure the distributed clock, configure the startup parameters, configure the PDO parameters and mapping, configure the Bus cycle, configure the startup check the supplier ID and product ID						
Isolation	Communication port isolation						
CANopen communication port							
Communication Interface	1 CAN communication master interface						
CAN extension maximum number of master	8						
Maximum number of slave	Up to 32 slave can be connected behind a master						
agreement type	CANopen DS301 standard protocol						
Support function	Automatically start CANopen manager, optional slave polling, start slave, NMT, synchronous production, synchronous consumption, heartbeat generation, create activation time						
Transmission rate	1000	800	500	250	125	50	20
The maximum length	25	50	100	250	500	1000	2500
isolation	Communication port isolation						

5.3.2 Schematic Diagram and Definition of CPU External Interface



Schematic diagram of external interface

Table 5-6 Definition of power interface

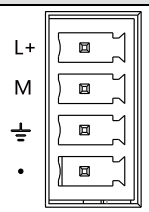
4 detachable terminals (spacing 3.50mm)	Symbol	Description
	L+	24V power positive
	M	24V power negative
	⏚	24V power ground
	--	--

Table 5-7 USB interface definition

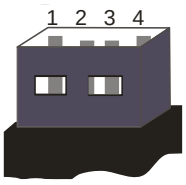
USB interface	NO.	Symbol	Description
	1	V_BUS	+5V power supply
	2	Data-	Data negative
	3	Data+	Data positive
	4	GND	Ground

Table 5-8 EtherNET/EtherCAT interface definition

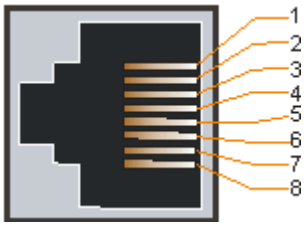
Dual network port interface	NO.	Symbol	Description
	1	TX+	Data send+
	2	TX-	Data send -
	3	RX+	Data receive+
	4	TERM	-
	5	TERM	-
	6	RX-	Data receive-
	7	TERM	-
	8	TERM	-
	shell	PE	Chassis ground

Table 5-9 RS485/CAN communication interface definition

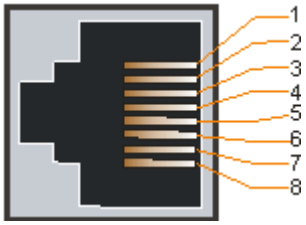
Single network port interface	NO.	Symbol	Description
	1	CAN_H	Data send+
	2	CAN_L	Data send -
	3	--	-
	4	A0	RS485 signal A/-
	5	B0	RS485 signal B/+
	6	--	-
	7	CAN_GND	CAN/RS485 signal ground
	8	--	-
	shell	PE	Chassis ground

Table 5-10 Expansion BUS interface definition

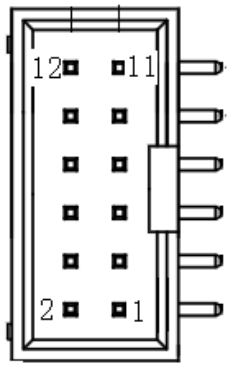
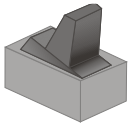
Expansion BUS	N0.	signal	description
	1	BUS_+5V	Bus 5V+
	2	GND	Bus 5V-
	3	BUS_CLK_A	Bus differential clock pair
	4	BUS_CLK_B	
	5	GND	Bus 5V-
	6	BUS_DAT_A	Bus differential data pair
	7	BUS_DAT_B	
	8	GND	Bus 5V-
	9	BUS_+5V	Bus 5V+
	10	BUS_+5V	Bus 5V+
	11	BUS_INT_B	Interrupt
	12	BUS_ADDR_B	Address

Table 5-11 Definition of DIP switches for system operation

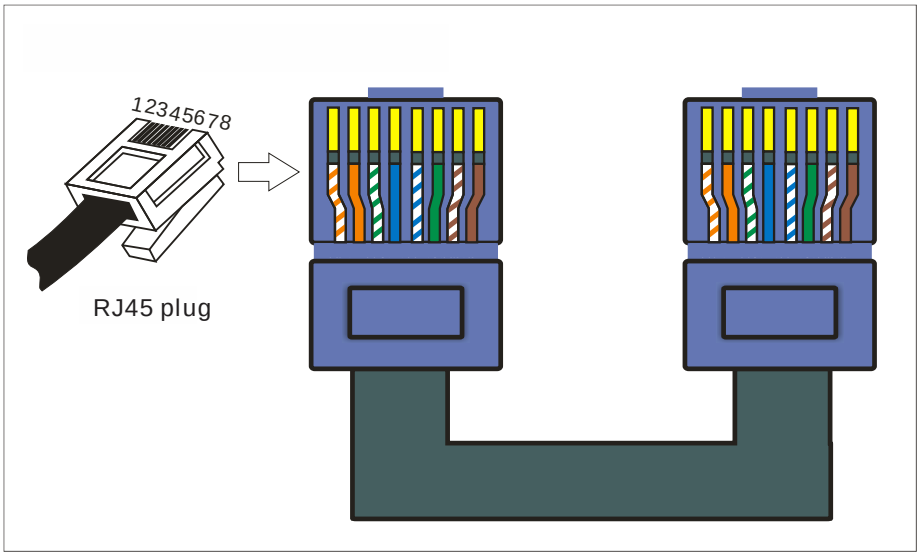
System operation switch	Symbol	Direction	Condition
	RUN	UP	system running
	STOP	DOWN	system stop
	Quickly flick 8 times in 3 seconds		Reset IP configuration (from RUN->STOP as once, and from STOP->RUN as once)

5.3.3 Make Standard Network Cables

When the H56-1/H52-100 motion controller performs the following communication, it is recommended to use a standard network cable:

- The main control module communicates with the host computer through EtherNET
- Main control module for EtherCAT communication
- Main control module for CANopen communication
- CAN-1M master module communicates with CAN slave such as E10/H1A/H2A/EM277C

Refer to the following diagram to make a standard network cable:



5.4 Data Storage Specifications

The CPU allows a specified value to be forced to one or all inputs or outputs (I and Q bits). In addition, you can also forcibly change the first 128 internal memory data M areas. The M memory variables can be changed by byte, word or double word.

Table 5-13 Data storage area specifications

Digital input/output image area (I/Q)		
	Digital input image area (I) Digital output image area (Q)	**Digital input image area (I) Digital output image area (Q)**
Bit address range	I0.0~I255.7	Q0.0~Q255.7
Byte address range	IB0~IB255	QB0~QB255
Word address range	IW0~IW254	QW0~QW254
Double word address range	ID0~ID252	QD0~QD252
Access attributes	Support immediate access/direct access/indirect access	
Keep attributes	Power down hold is not supported	
Analog input / output mapping area（AI/AQ）		
Word length per channel	16 bits	
Parameter address range	Analog input image area (AI): AIW0~AIW1022 (read only)	Analog output image area (AQ): AQW0~AQW1022 (write only)
Access attributes	Support immediate access/direct access/indirect access	
Keep attributes	Power down hold is not supported	
Variable memory area (V)		
Capacity (bytes)	65536 bytes	
Bit address range	V0.0~V65535.7	Read/write
Byte address range	VB0~VB65535	Read/write
Word address range	VW0~VW65534	Read/write
Double word address range	VD0~VD65532	Read/write
Access attributes	Support direct access/indirect access	
Keep attributes	All support power-down retention	
Special memory area (SM)		
Capacity	2048 bytes	
Bit address range	SM0.0~SM2047.7	SMB0~29 read only, read/write later
Byte address range	SMB0~SMB2047	SMB0~29 read only, read/write later
Word address range	SMW0~SMW2046	SMB0~29 read only, read/write later
Double word address range	SMD0~SMD2044	SMB0~29 read only, read/write later
Access attributes	Support direct access/indirect access	
Special register SM function description, please refer to Appendix J Special Function Register		
Internal memory area (M)		

Capacity (bytes)	32 bytes
Bit address range	M0.0~M31.7
Byte address range	MB0~MB31
Word address range	MW0~MW30
Double word address range	MD0~MD28
Access attributes	Support direct access/indirect access
Keep attributes	Configurable as full or partial power-down retention
Local variable area (L)	
Capacity (bytes)	128 bytes
Bit address range	L0.0~L127.7
Byte address range	LB0~LB127
Word address range	LW0~LW126
Double word address range	LD0~LD124
Access attributes	Support direct access
Keep attributes	The subroutine in the same calling position can be retained from power-on to shutdown, but cannot be retained after power-off
Accumulator register (AC)）	
capacity	4
Bit address range	not support
Byte address range	AC0~AC3
Word address range	AC0~AC3
Doubleword address range	AC0~AC3
Access properties	Support direct access
Keep properties	Power down hold is not supported
Status register(S)	
Capacity (bytes)	32
Bit address range	S0.0~S31.7
Byte address range	SB0~S31
Word address range	SW0~SW30
Double word address range	SD0~SD28
Access attributes	Support direct access/indirect access
Keep attributes	Power down hold is not supported
Timer (T)	

Type resolution	Type resolution	amount	NO.	Maximum timing value
With memory on-delay timer TONR	1ms	2	T0,T64	32.767s
	10ms	8	T1~T4,T65~T68	327.67s
	100ms	54	T5~T31,T69~T95	3276.7s
On-delay timer TON/Off-delay timer TOF	1ms	34	T32,T96,T256~T287	32.767s
	10ms	744	T33~T36,T97~T100,T288~T1023	327.67s
	100ms	1206	T37~T63,T101~T255,T1024~T2047	3276.7s
Access attributes	The timing register can be directly/indirectly accessed, and the status bit can only be directly accessed			
Keep attributes	The power-down retention attribute of the current timing value can be configured, and the status bit cannot be retained			
Counter(C)				
Number	2048			
Counting method	Count up/count down/count up and down			
Maximum count value	32767			
Access properties	The counting register can be directly/indirectly accessed, and the status bit can only be directly accessed			
Keep properties	Configurable power-down retention properties of the current count value, the status bit cannot be retained			
Rising and falling edges				
Sum of maximum available quantities	4096			

Table 5-14 Data types supported by H56-10/H52-10

type of data	Capacity	instruction	Ranges
BOOL	1 Bit	Boolean value	0~1
BYTE	8 Bit	Unsigned byte	0~255
WORD	16 Bit	Unsigned int	0~65535
INT	16 Bit	Signed int	-32768~+32767
DWORD	32 Bit	Unsigned double int	0~4294967295
Double int	32 Bit	Signed double int	-2147483648~+2147483647
Real	32 Bit	IEEE 32-bit float	+1.175495E-38~+3.402823E+38 -1.175495E-38~-3.402823E+38
String	1~255 byte	The ASCII string is stored in the PLC memory as it is, in the form of: 1 byte string length + ASCII characters	without

5.5 Password Level and Authority Control

Table 5-15 CPU password level and authority control

Operating instructions	Permission level 1	Permission level 2	Permission level 3	Permission level 4
Read and write user data	admit	admit	admit	admit
Run/stop/power-on reset	admit	admit	admit	admit
Read and write real-time clock	admit	admit	admit	admit
Write Q point in STOP mode	admit	Password verification required	Password verification required	Password verification required
Mandatory data	admit	Password verification required	Password verification required	Password verification required
Upload program block/data block/hardware configuration block	admit	admit	Password verification required	Not allowed
Download program block/data block/hardware configuration block	admit	Password verification required	Password verification required	Password verification required (downloading of hardware configuration blocks is not allowed)
Clear program block/data block/hardware configuration block	admit	Password verification required	Password verification required	Need to verify the password (the hardware configuration block is not allowed to be deleted, and the three blocks are allowed to be deleted together)
Runtime editing	admit	Password verification required	Password verification required	Not allowed
First or multiple scans	admit	Password verification required	Password verification required	Password verification required
Refresh scan cycle	admit	Password verification required	Password verification required	Password verification required
Project comparison	admit	admit	Password verification required	Not allowed
Program status monitoring (time	admit	admit	admit	admit

stamp comparison pass)				
Program status monitoring (time stamp comparison fails)	admit	Password verification required	Password verification required	Not allowed

5.6 Real Time Clock

Table 5-16 Real-time clock functions of the CPU

Factory settings	Not set, fixed value Sunday, January 1, 2090 00:00:00	
Power-down hold time	Supercapacitor 112 hours, H56/H52 connected to an external battery, the power-off retention time can reach at least 1 year, the typical value is 3 years.	
Precision	Monthly deviation <60 seconds	
Read clock function	Read by command TODR/TODRX or by software	
Set the clock function	Set by command TODW/TODWX or by software	
Conventional clock format (8 bytes)		
T byte	**describe**	**byte data**
0	year（0-99）	Current year (BCD value)
1	month（1-12）	Current month (BCD value)
2	data（1-31）	Current date (BCD value)
3	hour（0-23）	Current hour (BCD value)
4	minute（0-59）	Current minute (BCD value)
5	second（0-59）	Current second (BCD value)
6	0	Reserved, always set to 00
7	week（1-7）	The current day of the week, 1=Sunday (BCD value)
Extended clock format (19 bytes)		
T byte	**describe**	**byte data**
0	year（0-99）	Current year (BCD value)
1	month（1-12）	Current month (BCD value)
2	data（1-31）	Current date (BCD value)
3	hour（0-23）	Current hour (BCD value)
4	minute（0-59）	Current minute (BCD value)
5	secind（0-59）	Current second (BCD value)
6	0	Reserved, always set to 00
7	week（1-7）	The current day of the week, 1=Sunday (BCD value)
8	Time zone	00H-03H, 08H,10H-13H, FFH
9	Correction hours (0-23)	Correction amount, hour (BCD value)
10	Corrected minutes (0-59)	Correction amount, minute (BCD value)

11	Starting month (1-12)	Start month of daylight saving time (BCD value)
12	Start date (1-31)	Daylight saving time start date (BCD value)
13	Start hour (0-23)	Start hour of daylight saving time (BCD value)
14	Start minute (0-59)	Start minute of daylight saving time (BCD value)
15	End month (1-12)	End month of daylight saving time (BCD value)
16	End date (1-31)	End date of daylight saving time (BCD value)
17	End hour (0-23)	End hour of daylight saving time (BCD value)
18	End minute (0-59)	End minute of daylight saving time (BCD value)

5.7 Interrupt Event

Table 5-17 Interrupt events

Priority group	Interrupt event number	Priority	Describe
Communication and diagnostic events (highest priority)	8	0	Port0: receive character
	9	0	Port0: transfer complete
	23	0	Port 0: receiving information completed
	24	0	Port 1: receiving information completed
	25	0	Port 1: receive character
	26	0	Port 1: transmission completed
	34	0	100us time base timing interrupt 0
	35	0	100us time base timing interrupt 1
	36	0	Module diagnostic event interrupt (not implemented)
IO interrupt and high-speed counter interrupt	0	1	Rising edge, I0.0
	2	1	Rising edge, I0.1
	4	1	Rising edge, I0.2
	6	1	Rising edge, I0.3
	48	1	Rising edge, I0.4
	50	1	Rising edge, I0.5
	52	1	Rising edge, I0.6
	54	1	Rising edge, I0.7
	56	1	Rising edge, I1.0
	58	1	Rising edge, I1.1
	1	1	Falling edge, I0.0
	3	1	Falling edge, I0.1
	5	1	Falling edge, I0.2
	7	1	Falling edge, I0.3
	49	1	Falling edge, I0.4
	51	1	Falling edge, I0.5
	53	1	Falling edge, I0.6
	55	1	Falling edge, I0.7
	57	1	Falling edge, I1.0
	59	1	Falling edge, I1.1
	12	1	HSC0 CV=PV
	27	1	HSC0 direction change
	28	1	HSC0 external recovery / Zphase
	13	1	HSC1 CV=PV
	14	1	HSC1 direction change
	15	1	HSC1 external recovery / Zphase
	16	1	HSC2 CV=PV

	17	1	HSC2 direction change
	18	1	HSC2 external recovery /Zphase(not support)
	32	1	HSC3 CV=PV
	38	1	HSC3 direction change
	40	1	HSC3 external recovery /Zphase(not support)
	29	1	HSC4 CV=PV
	30	1	HSC4 direction change(not support)
	31	1	HSC4 external recovery /Zphase(not support)
	33	1	HSC5 CV=PV
	39	1	HSC5 direction change(not support)
	41	1	HSC5 external recovery /Zphase(not support)
PTO interrupt	19	1	PTO 0 completion interrupt (not supported)
	20	1	PTO 1 completion interrupt (not supported)
Timing (lowest priority)	10	2	Timing interrupt 0
	11	2	Timing interrupt 1
	21	2	Timer T32 CT = PT interrupt
	22	2	Timer T96 CT = PT interrupt

5.8 CPU Fault Diagnosis

When the system fails, please first check the following conditions:

1) Whether the H56-10/H52-10 motion controller and expansion modules are normally powered (Note: The main control module and expansion modules (except digital modules) are recommended to use PWR-02 independent power supply. Unstable power supply will cause shutdown).

2) Check whether the screws and connectors of the H56-10/H52-10 motion controller and expansion module I/O terminals are not loose.

3) Check the connection of the communication cable to make sure it is correct.

4) The PLC cannot be searched, please check the communication settings, such as changing the baud rate, connecting the serial port or IP, etc. and search again.
After the above conditions are met, the PLC fault can be read through MagicWorks PLC to read the diagnostic information, or through the LED indicator status of the PLC to check whether the PLC itself and the outside are abnormal.

5.8.1 Diagnose through MagicWorks PLC

Reading method of diagnostic information: select "PLC" → "get diagnostic information" in the menu item of magicworks PLC project manager interface to open the diagnostic window.

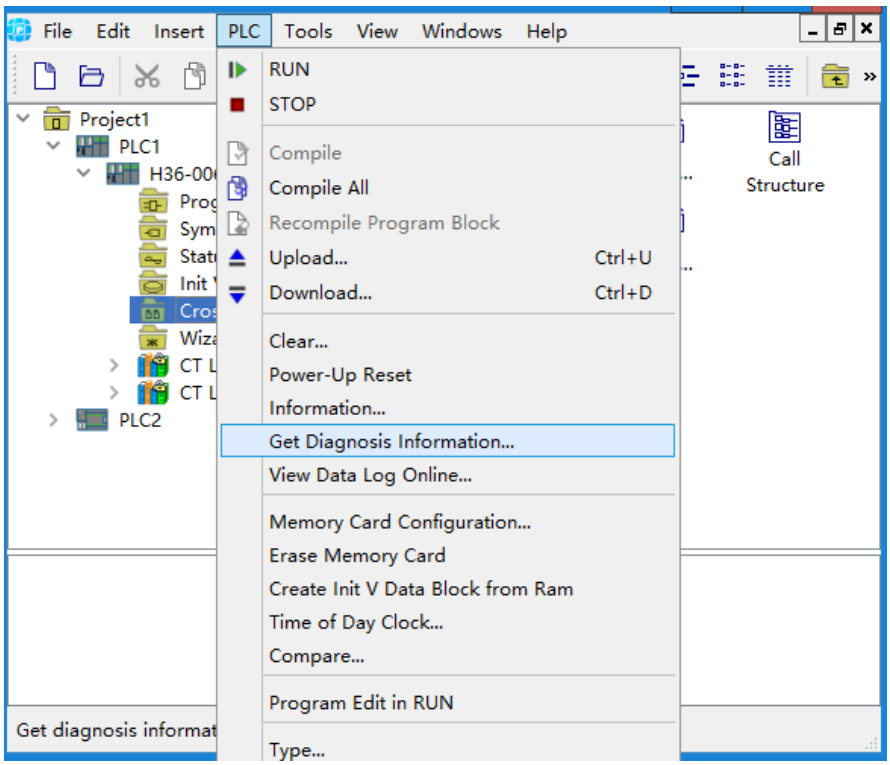


Table 5-18 description of diagnostic information window

Item	Description	Remark
Date	The date when the diagnosis event occurred, in the format: year.month.day	--
Time	The time when the diagnostic event occurred, in the format: hour: minute: second: millisecond	--
Type	Diagnose the type of event	--
Process variable	When the SOE event occurs, the value of the process variable (only SOE events have process variables)	Up to 16 SOE events can be configured. the type and display format of process variables can be configured in the CPU "Properties" → "SOE Configuration".

Table 5-19 Diagnostic function specifications

Supported event types	Encode type	Specific event coding and description	
		Code	Description
Operation mode conversion synchronization error	0x5	0x01	Power-up event
		0x02	Boot event
		0x03	Downtime
Operation mode conversion Synchronization error	0x6	0x00	No error (reserved)
		0x01	The used DB block does not exist or the used DB is out of bounds (reserved)

CPU non fatal error	0x7	0x00	No error
		0x01	Enable HSC before executing HDEF box (reserved)
		0x02	Input interrupt assignment conflict, assigned to the point assigned to HSC (reserved)
		0x03	Input assignment conflict, assigned to HSC that has been assigned to input interrupt or other HSC (reserved)
		0x04	Try to execute ENI, DISI, or HDEF instructions in the interrupt routine
		0x05	Attempted execution of a second HSC/PLS with the same number before completing the first (HSC/PLS in an interrupt routine conflicts with HSC/PLS in main program)
		0x06	Indirect addressing error
		0x07	TODW (Daily Write Time) or TODR (Daily Read Time) data error
		0x08	Exceeded maximum user subroutine nesting level
		0x09	Simultaneous execution of XMT/RCV instructions in port 0
		0x0A	Attempt to redefine the HSC by executing another HDEF instruction for the same HSC (reserved)
		0x0B	Simultaneous execution of XMT/RCV instructions in port 1
		0x0C	Reserve
		0x0D	Reserve
		0x0E	Reserve
		0x0F	Illegal numeric value in compare contact instruction
		0x10	Reserve
		0x11	Reserve
		0x12	Reserve
		0x13	Illegal PID loop table
		0x80	The program is too large, the CPU cannot generate executable code. please reduce the program
		0x81	Reserve
		0x82	Illegal instruction, check instruction mnemonic 0
		0x83	MEND is missing, or instructions are not allowed in the main program. Add MEND command or remove incorrect command
		0x85	Missing FOR, add FOR instruction or delete NEXT instruction
		0x86	Missing NEXT, add NEXT command or delete FOR command
		0x87	Missing labels (LBL, INT, subroutine), add appropriate labels
		0x88	Missing RET or instruction not allowed in subroutine, add RET at end of subroutine or remove incorrect instruction
		0x89	Missing RETI or instruction not allowed in an interrupt routine: add RETI to the end of the interrupt routine or remove incorrect instruction.

		0x8B	Illegal JMP to or from an SCR segment
		0x8C	Duplicate label (LBL, INT, SBR), rename one of the labels.
		0x8D	llegal label (LBL, INT, SBR), ensure the number of labels allowed was not exceeded.
		0x90	Illegal parameter, verify the allowed parameters for the instruction.
		0x91	Range error (with address information). check the operand ranges.
		0x92	Error in the count field of an instruction (with count information). verify the maximum count
		0x93	More than the number of FOR/NEXT nesting levels
		0x94	Use address information to write range error to non-volatile memory
		0x95	Missing LSCR instruction (load SCR)
		0x96	Missing SCRE instruction (SCR End) or disallowed instruction before the SCRE instruction
		0x97	User program contains both unnumbered and numbered EU/ED instructions
		0x98	Illegal edit in RUN mode (edit attempted on program with unnumbered EU/ED instructions)
		0x99	Too many hidden blocks
		0x9A	Attempt to switch to free port mode during user interrupt
		0x9B	Illegal index (string operation, a starting position value 0 in the operation has been specified)
CPU fatal error	0x8	0x00	No serious errors
		0x01	Reserve
		0x02	Reserve
		0x03	Scan watchdog timeout error
		0x04	Reserve
		0x05	Reserve
		0x06	Reserve
		0x07	Reserve
		0x08	Reserve
		0x09	Reserve
		0x0A	Reserve
		0x0B	Reserve
		0x0C	Reserve
		0x0D	Reserve
		0x0E	Reserve
		0x0F	Reserve
		0x10	Internal software error
		0x11	Indirect addressing error of comparison contact
		0x12	Illegal floating point value error in comparison node
		0x13	Reserve
		0x14	Compare node range error

Communication error	0xA	Reserve	
SOE event	0xB	BOOL variable state change event: (1) DI, DQ status ON→OFF, OFF→ON (2) BOOL type ON→OFF, OFF→ON	Monitoring event at the end of scan cycle, up to 16 SOE events can be configured, encoded in sequence 1-16, process variables can be configured (support signed integer, unsigned integer, signed double integer, unsigned double integer, floating point measurement of solid state), Configure the corresponding event text and display it in the diagnostic information.
SOE monitoring and recording method		Monitor and record once per scan cycle	
EtherCAT register SMB400~SMB465		For the description of each status bit, refer to chapter J Special Function Register	
Module diagnostic event SMB500～SMB531	0xC	0x00	No faults in the module
		0x01	Module is busy
		0x02	Module timed out and did not respond
		0x03	Module type does not match
		0x04	Module version does not match
		0x05	Software error
		0x06	The module has timed out waiting for the flag
		0x07	Bus response error
		0x08	Bus CRC check error
		0x10	Memory offset is out of range
		0x11	Module is not ready
		0x12	Module configuration error
		0x13	Module does not support this instruction
		0x15	Module internal diagnosis
		0x16	Module has no power

Note: For the diagnostic functions of the special storage areas SMB400~SMB465, SMB500~SMB531 and SMB550~SMB2047, please refer to chapter J Special Memory Description.

5.8.2 Diagnose Via LED Status Indicator

Table 5-23 H56-10/H52-10 LED status indicator function description

Indicator light	Condition	description
24V POWER	green	ON: with 24VDC, OFF: without 24VDC
SF	red	ON: system error, OFF: no error
BF	red	ON: BUS error, OFF: no error
FRCE	yellow	ON: Some items are forced, OFF: No items are forced

		(can also be controlled by DLED instructions)
RMC	green	ON: The communication between the CPU and the remote server is successful (EtherNET communication port parameters have been correctly configured)
RUN	green	ON: system running, OFF: system stops,
STOP	yellow	ON: system stops, OFF: system running,
LINK1	green	ON: connected, OFF: not connected
SPEED1	yellow	ON: 100Mbps, OFF: 10Mbps
LINK2	green	ON: connected, OFF: not connected
SPEED2	yellow	ON: 100Mbps, OFF: 10Mbps
I0.0~I1.1	green	ON: with signal input , OFF: without signal input

Digital Module

6

This chapter mainly introduces the digital modules of the H56-10/H52-10 application system, as follows:

The digital quantity module is used to output digital signals to field devices or to receive digital signals input by field devices.

This chapter mainly introduces the following contents of CTH300-H digital module:

- ❑ Digital input module technical specifications and wiring specifications
- ❑ Digital input module technical specifications and wiring specifications

Table 6-1 Basic attributes of digital modules

name	Specification Description	Order number
DIT-080S1	Digital input 8 x 24VDC	CTH3 DIT-080S1
DIT-160S1	Digital input 16 x 24VDC	CTH3 DIT-160S1
DIT-320S1	Digital input 32 x 24VDC	CTH3 DIT-320S1
DQT-080S1	Digital output 8 x 24VDC	CTH3 DQT-080S1
DQT-160S1	Digital output 16 x 24VDC	CTH3 DQT-160S1
DQT-320S1	Digital output 32 x 24VDC	CTH3 DQT-320S1
DQR-080S1	Digital output 8 x Relay	CTH3 DQR-080S1
DQR-160S1	Digital output 16 x Relay	CTH3 DQR-160S1

6.1 Digital Input Module

6.1.1 Specification

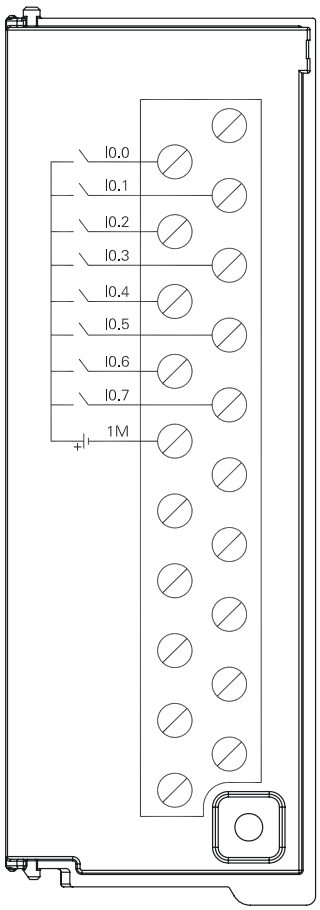
Table 6-2 Specifications of Digital Input Module

Item		DIT-080S1	DIT-160S1	DIT-320S1
Dimension（W×H×D）		34×115×100 mm		
Input		8	16	32
Current consumption	24V DC	4mA/ channel		
	+5VBus	60mA	80mA	130mA
Input type		Drain/Source （IEC 1 Drain）		
Input rated voltage		24V DC, 6mA		
Input voltage range		20.4~28.8V DC		
Surge voltage		35V DC, continue 0.5s		
Logic 1 (minimum)		15V DC, 2.5mA, Flip level: 10.5V DC±15%		
Logic 0 (maximum)		5V DC, 1mA		
Connect 2-wire proximity switch sensor (BERO) Allowable leakage current (maximum)		1mA		
Input filtering		Configurable, supports 0.2ms, 0.4ms, 0.8ms, 1.6ms, 3.2ms, 6.4ms (default), 12.8ms		
Input frequency		1.5kHz, 50% duty cycle		
input resistance		6.6kΩ		
isolation		500V AC for 1 minute		

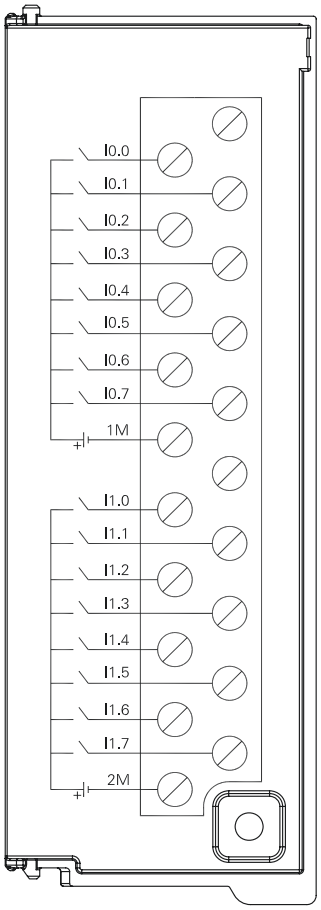
Number of isolation points per group		8		
Simultaneous ON points		8	16	32
Cable length	Shield	500m		
	Unshielded	300m		

6.1.2 Wiring

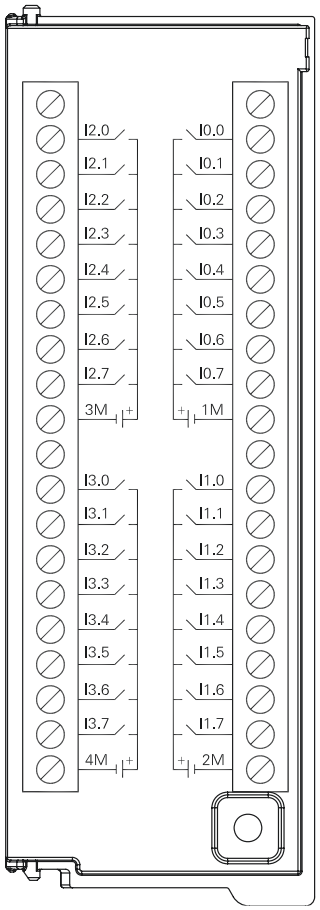
◆ DIT-080S1



◆ DIT-160S1



◆ DIT-320S1



6.2 Digital Output Module

6.2.1 Specifications

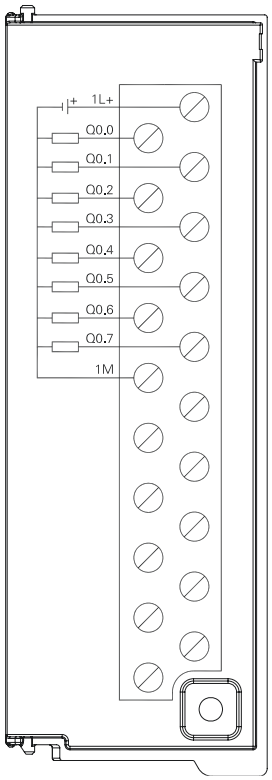
Table 6-3 Specifications of Digital Output Module

Item		DQT-080S1	DQT-160S1	DQT-320S1	DQR-080S1	DQR-160S1
Dimension（W×H×D）		34×115×100 mm				
Output		8	16	32	8	16
Current consumption	24V DC	50mA	95mA	180mA	64mA	130mA
	+5VBus	70mA	120mA	210mA	45mA	60mA
Input type		Solid state-MOSFET, source type			Relay-dry contact	
Input rated voltage		24V DC			DC: 24V,AC: 110V/220V	

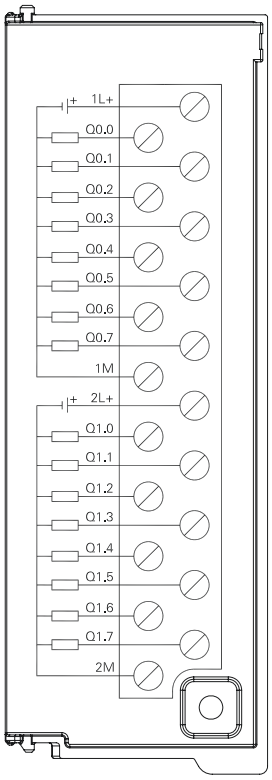
Output voltage range		20.4~28.8V DC				DC: 5~30V,AC: 5~250V	
Logic 1 (min)		20V DC				-	
Logical 0 (max)		0.1V DC,10kΩ load				-	
Maximum output current		0.5A				2A	
Current at each common terminal		Maximum 4A				Maximum 16A	
Maximum output leakage current		15uA				-	
Surge current		8A,100ms				5A, 4s@10% duty cycle	
Lamp load		5W				DC: 30W /AC: 200W	
contact resistance		0.3Ω, Maximum 0.6Ω				The maximum for new devices is 0.2Ω	
Output delay		Off to on (max): 50uS On to off (max): 200us				Maximum 10ms	
Maximum output frequency		1kHz				Resistive load: 10Hz Lamp load: 1Hz Inductive load: 0.5Hz	
Mechanical life (no load)		-				10,000,000	
Contact life (rated load)		-				100,000	
Quarantine	Field to logic	500V AC,Lasting for 1min					
	Coil to contact	-				1500V AC, lasts 1min	
Isolation points per group		8					
Simultaneous on points		8	16	32		8	16
Two outputs are connected in parallel		Support parallel connection of two outputs in the same group					
Cable length	Shield	500m					
	Unshielded	150m					

6.2.2 Wiring

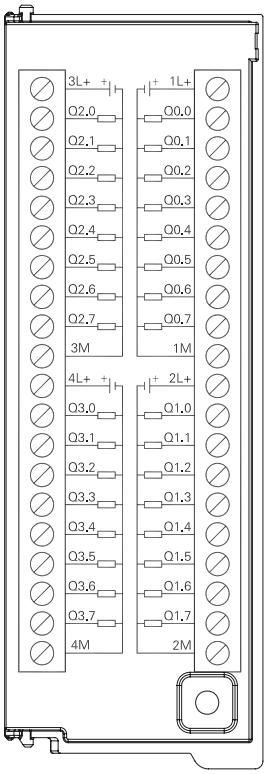
◆ DQT-080S1



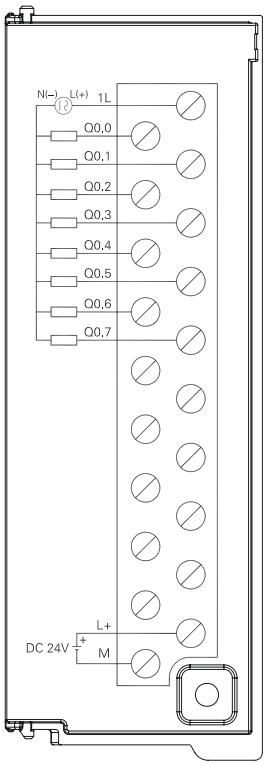
◆ DQT-160S1



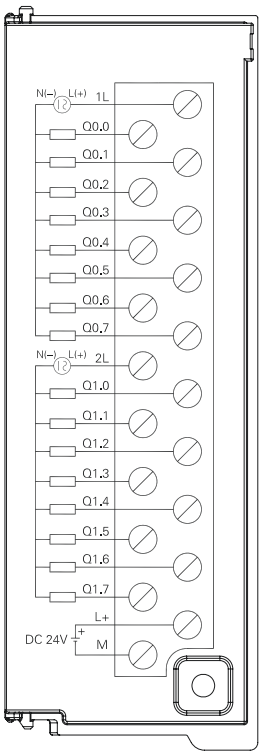
◆ DQT-320S1



◆ DQR-080S1



◆ DQR-160S1



Analog Module

7

The analog quantity module of H56-10 / H52-10 application system is introduced as follows:

The analog module is used to output analog signals to field devices or to receive analog signals input by field devices.

This chapter mainly introduces the following contents of CTH300-H analog modules:

- ❑ General characteristics of analog modules
- ❑ Technical specifications and wiring of analog input modules
- ❑ Analog output module technical specifications and wiring
- ❑ Technical specifications and wiring of analog input/output modules

Table 7-1 Basic attributes of analog modules

Name	Specification Description	Order number
AIS-04	Analog voltage and current input, 4AI x 12 bits	CTH3 AIS-040S1
AMS-06	Analog voltage and current input and output, 4AIx12 bits, 2AQx12 bits	CTH3 AMS-060S1
AIV-08	Analog voltage input, 8AI x 16 bits	CTH3 AIV-080S1
AIC-08	Analog current input, 8AI x 16 bits	CTH3 AIC-080S1
AQS-04	Analog voltage and current output, 4AQ x 12 bits	CTH3 AQS-040S1
AQS-08	Analog voltage and current output, 8AQ x 12 bits	CTH3 AQS-080S1

7.1 General Features

7.1.1 Conventional Characteristics

Table 7-2 General characteristics of analog modules

Item	AIS-040S1	AMS-060S1	AIV-080S1/ AIC-080S1	AQS-040S1	AQS-080S1
Physical					
Dimension（W×H×D）	34×115×100 mm				
Power					
Rated input voltage	24V DC				
Input voltage range	20.4V~28.8V DC				
Input Current	65mA	110mA	50mA	110mA	200mA
Reverse polarity protection	YES				
Bus power supply voltage	+5V DC				
Bus power current	50mA		30mA	40mA	
LED indicator					
24V power indicator	ON: 24VDC power supply. OFF: No 24VDC power supply				
SF indicator	ON: Module failure. OFF: No error Flashing: Input current signal exceeds limit alarm (only for 4-20mA)				

7.1.2 Functional Characteristics

Table 7-3 Functional characteristics of analog modules

Item	Description
Extensions	Provide Bus expansion function
Signal isolation	Isolation between field and Bus
Power protection	The power supply terminal provides reverse connection protection and surge absorption
Filter function	Using a combination of hardware filtering and software filtering
Power function	The module uses 24V DC power supply

7.2 Analog Input Module

7.2.1 Specifications

Table 7-4 Analog input module specifications

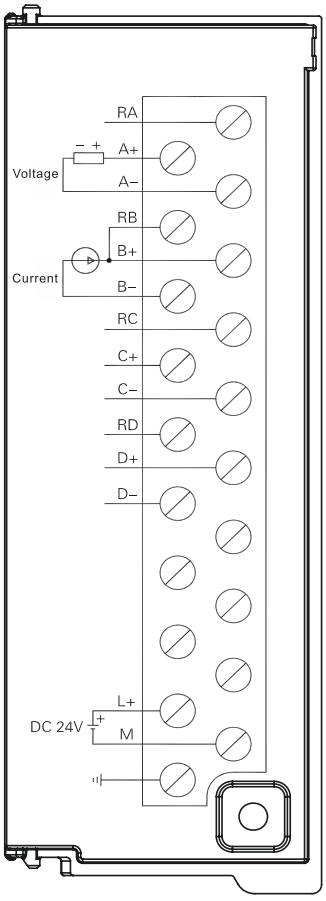
Item		AIS-040S1	AIV-080S1	AIC-080S1
Input type		Voltage or current (differential input)	Voltage input	Current input
Input		4	8	
Input range	Voltage	Unipolar: 0 ~ 5V, 0 ~ 10V, bipolar: ± 2.5V, ± 5V		
	Current	0~20mA, 4~20mA		
Maximum input	Voltage	30V DC		
	Current	40mA		
input resistance	Voltage	≥2MΩ		
	Current	250Ω		
Data Format	Voltage	0~+32000(unipolar), -32000~+32000(bipolar)		
	Current	0~+32000		
Input step response		4 channels 5ms (fastest)	8 channels 50ms (fastest)	
Module update frequency (all channels)		4 channels support 200Hz, 100Hz, 50Hz, 20Hz and 10Hz configurations Default: 50Hz all channels	8 channels support 50Hz, 20Hz, 10Hz, 5Hz and 2Hz configurations Default: 10Hz all channels (50Hz only meets 4 channels)	
Common mode suppression		>40dB		
Channel crosstalk		>60dB		
Common mode voltage		-12V≤signal voltage+common mode voltage≤+12V		
Resolution	Unipolar	12 bits	16 bits	
	Bipolar	11 bits + Sign bit	15 bits + Sign bit	
Measuring principle		Successive approximation	Sigma-delta（Σ-△）	
Measurement error		0.5%(maximum)	0.1%(maximum)	
Wire break detection (only for 4~20mA)		Broken wire calibration: -32768, 32767 two values are optional		
Isolation	Field to logic	500V AC		
	24VDC to logic			

Diagnosis	Ultra-negative range	Unipolar: 0, bipolar: -32768	Unipolar: 0, bipolar: -32768
	Over positive range	Unipolar: 32760, bipolar: 32752	Unipolar: 32767, Bipolar: 32767
	No power	32736	32766

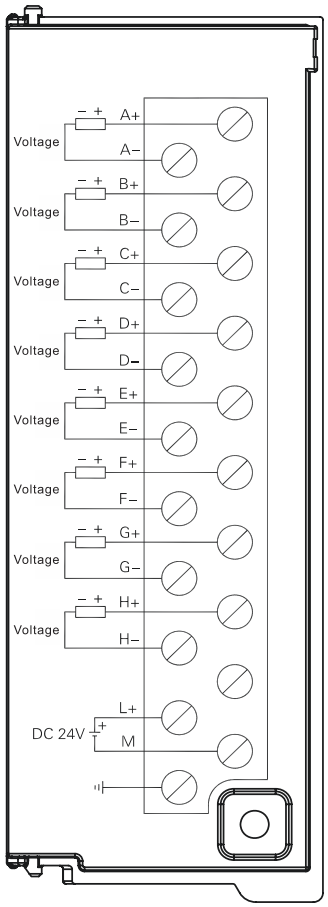
Remarks: The input range of CTH3 AIC-080S1 is 4~20mA. When 1.2mA≤input current<4mA, the value will change linearly from 0 to -32767. When it is less than 1.2mA, the value will display -32767.

7.2.2 Wiring

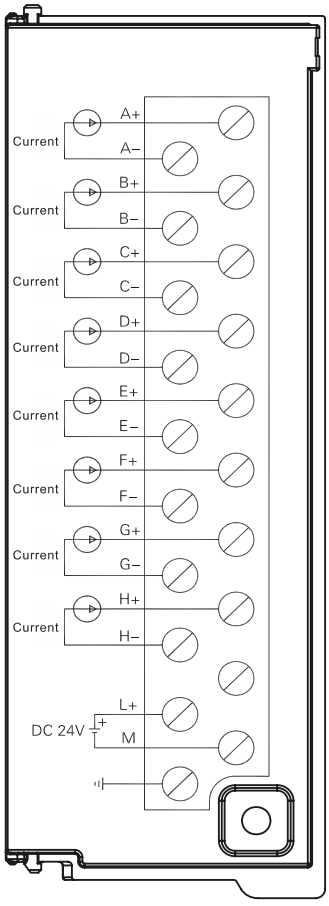
◆ AIS-040S1



◆ AIV-080S1



◆ AIC-080S1



7.3 Analog Output Module

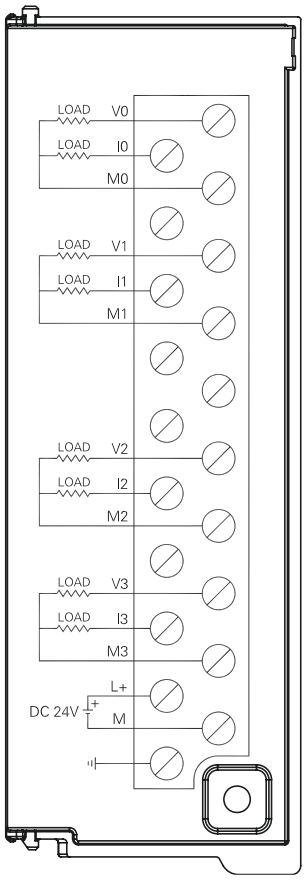
7.3.1 Specifications

Table 7-5 specifications of analog output module

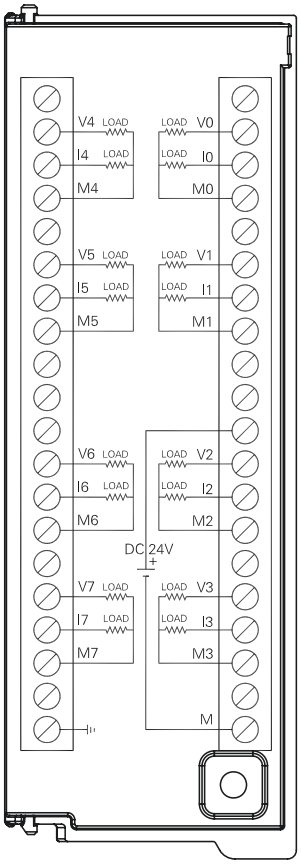
Item		AQS-040S1	AQS-080S1
Output type		Voltage or current	
Input		4	8
Output range	Voltage	±10V	
	Current	0~20mA, 4mA~20mA	
Protect	Output misconnection voltage	Max. 30V DC	
	Voltage short circuit protection	YES	
Data format	Voltage	-32000~+32000(at full scale)	
	Electric current	0~+32000(at full scale)	
Establish time	Voltage output	100us	
	Current output	2ms	
Load impedance	Voltage output	5000Ω(min)	
	Current output	500Ω(max)	
Resolution	Unipolarity	12 bits	
	Bipolar	11 BIST + sign bit	
Accuracy	Voltage	Typical value: ± 0.5% of full scale. worst case: ± 2% of full scale	
	electric current	Typical value: ± 0.6% of full scale. worst case: ± 2% of full scale	
Quarantine	Power isolation	500V AC	
	Field to logic		

7.3.2 Wiring Specifications

◆ AQS-040S1



◆ AQS-080S1



7.4 Analog Input and Output Module

7.4.1 Specifications

Table 7-6 Specifications of analog input and output modules

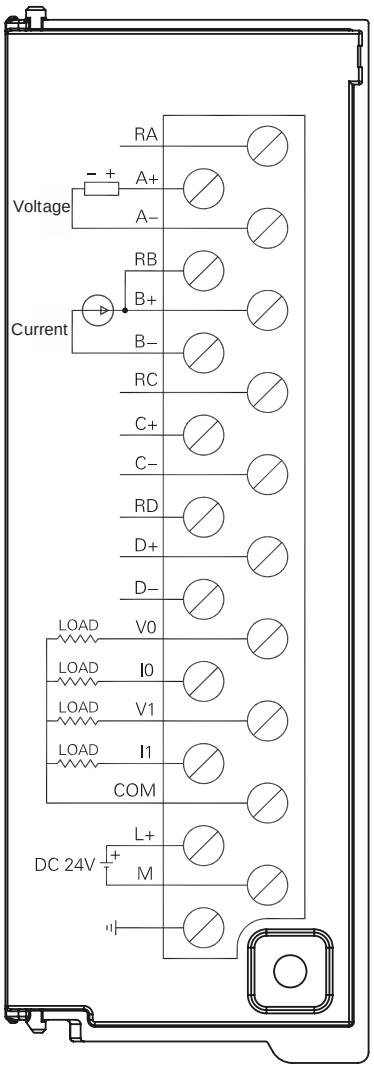
Input characteristics		AMS-060S1
Input type		Voltage or current (differential input)
Input		4
Input range	Voltage	Unipolar: 0~5V, 0~10V. Bipolar: ±2.5V, ±5V
	Current	0~20mA,4~20mA
Maximum input	Voltage	30VDC
	Current	40mA
Input resistance	Voltage	≥2MΩ
	Current	250Ω
Data Format	Voltage	0~+32000(Unipolar), -32000~+32000(Bipolar)
	Current	0~+32000
Input step response 4 channels		Input step response 4 channels 5ms (fastest)

5ms (fastest)			
Module update frequency (all channels)		4 channels support 200Hz, 100Hz, 50Hz, 20Hz, 10Hz configuration Default: 50Hz all channels	8 channels support 50Hz, 20Hz, 10Hz, 5Hz, 2Hz configuration Default: 10Hz all channels
Common mode		>40dB	
Channel crosstalk >60dB		>60dB	
Common mode voltage		-12V≤signal voltage+common mode voltage≤+12V	
Resolution	Unipolar	12 bits	
	Bipolar	11 bits+sign bit	
Measuring principle		Successive approximation	
Measurement error		0.5%(max.)	
Disconnection detection (only for 4 ~ 20mA)		Disconnection calibration: -32768,32767 two values are optional,	
Isolation (field to logic) /24VDC to logic)		500V AC	
Diagnosis	Over negative range	Unipolar: 0, bipolar: - 32768	
	Over positive range	Unipolar: 32760, bipolar: 32752	
	No power supply	32736	
Output characteristics		AMS-060S1	
Output type		Voltage or current	
Output points		2	
Output range	Voltage	±10V	
	Current	0~20mA, 4mA~20mA	
Protect	Output misconnection voltage	Max. 30V DC	
	Voltage short circuit protection	YES	
Data Format	Voltage	-32000~+32000(at full scale)	
	Current	0~+32000(at full scale)	
Establish time	Voltage output	100us	
	Current output	2ms	
Load impedance	Voltage output	5000Ω(min)	
	Current output	500Ω(max)	
Resolution	Unipolar	12bit	
	Bipolar	11bit + sign bit	
Precision	Voltage	Typical value: ± 0.5% of full scale. worst case: ± 2% of full scale	
	Current	Typical value: ± 0.6% of full scale. worst case: ± 2% of full scale	

Isolate	Power isolation	500V AC
	Field to logic	

7.4.2 Wiring

◆ AMS-060S1



Temperature Module

8

Introduce the temperature module of H56-10 / H52-10 application system, as follows:

The temperature module is used for thermocouple and thermal resistance temperature measurement. It integrates 4 or 8 input points. The user can configure the type of temperature sensor to meet the user's multi-point temperature measurement needs.

This chapter mainly introduces the following contents of the CTH300-H temperature module:

- ❑ Technical specifications
- ❑ Wiring specifications

Table 8-1 Basic attributes of the temperature module

Name	Specification Description	Order number
AIT-04	Thermocouple input module, 4 TC, isolated type with 16 bits accuracy	CTH3 AIT-040S1
AIT-08	Thermocouple input module, 8 TC, isolated type with 16 bits accuracy	CTH3 AIT-080S1
AIR-04	Thermal resistance input module, 4 RTD, isolated type with 16 bits accuracy	CTH3 AIR-040S1
AIR-08	Thermal resistance input module, 8 RTD, isolated type with 16 bits accuracy	CTH3 AIR-080S1

8.1 Specifications

8.1.1 Conventional Characteristics

Table 8-2 General characteristics of CTH300-H temperature modules

Item	AIT-040S1	AIT-080S1	AIR-040S1	AIR-080S1
Physical characteristics				
dimension(W×H×D)	34×115×100 mm			
Power characteristics				
Rated input voltage	24V DC			
Input voltage range	20.4V~28.8V DC			
Input Current	50mA		60mA	80mA
Reverse polarity protection	YES			
Bus power supply voltage	+5V DC			
Bus power current	50mA			
LED indicator				
24V	ON: 24VDC power. OFF: No 24VDC power supply			
SF	ON: The module is faulty. OFF: No error. Flashing: There is channel disconnection or overrange			

8.1.2 Features

Table 8-3 Functional characteristics of CTH300-H temperature modules

Function category	Function name	Description
I/O	I/O interface	Provide 4 channels/8 channels temperature sensor input interface
Extensions	extensions	Provide Bus expansion function
Isolation	Signal isolation	Isolation between field and Bus
	Power isolation	Isolation between external power supply and system power supply
Protective function	Power protection	The power supply terminal provides reverse connection protection and surge absorption
	Broken wire detection	Input provides disconnection detection function
Filter function	Filter function	Using a combination of hardware filtering and software filtering
Power function	Power function	The module uses 24VDC power supply

8.1.3 Input Specification

Table 8-4 Input specifications of CTH300-H temperature modules

Item	AIT-040S1	AIT-080S1	AIR-040S1	AIR-080S1
Input type	Suspended thermocouple		Module reference ground resistance	
Number of input	4	8	4	8
Wiring	-		Support 2-wire system, 3-wire system and 4-wire system. Default: 3-wire system	
Input range	Thermocouple type (select one):S, T, R, E, N, K, J voltage range: ±80mV default: K		Thermocouple type (select one): Pt-100Ω,200Ω, 500Ω, 1000Ω（α=3850ppm, 3920ppm, 3850.55ppm, 3916ppm, 3902ppm） Pt-10000Ω（α=3850ppm） Cu-9.035Ω（α=4720ppm） Ni-100Ω, 120Ω, 1000Ω（α=6720ppm, 6178ppm） R-150Ω, 300Ω, 600ΩFS default: Pt-100Ω（α=3850ppm）	
Temperature measurement range (℃)	S: -50~1768 T: -270~400 R: -50~1768 E: -270~1000 N: -270~1300 K: -270~1372 J: -210~1200 <Remarks> An error will be reported if the measurement exceeds the range. Refer to		Pt-100Ω, 200Ω, 500Ω, 1000Ω: -200~850 Pt-10000Ω: -200~600 Cu-9.035Ω: -200~260 Ni-0.00672: -80~260 Ni-0.006178: -60~300 < remarks > if the measurement exceeds the range, an error will be reported. Refer to table 8-5 for diagnosis details.	

	Table 8-5 for diagnostic details. For the overflow range refer to Section 8.1.4 Thermocouple Characteristics	

Characteristic		AIT-040S1	AIT-080S1	AIR-040S1	AIR-080S1
Isolate	Field to logic	500VAC			
	Field to 24VDC				
	24VDC to logic				
Common mode rejection		＞100dB@120VAC			
Input resolution	Temperature	0.1℃ / 0.1℉			
	Voltage	15 bits + sign bit		-	
	Resistance	-		15 bits + sign bit	
Measuring principle		Sigma-Delta			
Module update frequency (all channels)		4 channels support 8Hz, 4Hz, 2Hz, 1Hz configuration, default: 2Hz all channels			
		8 channels support 4Hz, 2Hz, 1Hz, 0.5Hz configuration, default: 1Hz all channels			
Length of wire to sensor		Up to 100m			
Wire loop resistance		100Ω		20Ω, Cu type 2.7Ω	
Noise suppression		85dB@50Hz/60Hz/400Hz			
Data word format		Voltage: -27648~+27648			
Input resistance		＞10MΩ			
Maximum input voltage		The input terminal can support misconnection with a maximum voltage of 30V DC			
Resolution		15 bits + sign bit			
Input filter attenuation		-3dB@21kHz		-3dB@3.6kHz	
Basic error		0.1%Fs(Voltage)		0.1%Fs(resistance)	
Repeatability		0.05%Fs			
Cold junction compensation		Configurable, with cold junction compensation by default		-	
Cold end error		±1.5℃		-	
Temperature unit		℃ / ℉ Configurable, default ℉			
Broken wire detection		Thermocouple: configurable, with wire break detection by default		Thermal resistance: there is always disconnection detection, which cannot be configured	
		Support both positive and negative directions, the default is positive			
PID control function		None		-	

Table 8-5 Diagnostic information of cth300-h temperature module

Error Type	Channel data	SF indicator	24V indicator	Range status bits	24V power failure
No module power supply	32766	OFF	OFF	0	1
Disconnection	32767(positive) -32768(negative)	FLASH	ON	1	0
Out of temperature range	32767(positive) -32768(negative)	FLASH	ON	1	0

<Remarks> The range status bit is the bit in the module error register byte. You can check SMB500~531 according to the module sequence to query the error code of the corresponding module. For the description of the status bits of SMB500~531, please refer to chapter 5.8.1 Diagnose through MagicWorks PLC

8.1.4 Thermocouple Characteristics

Temperature range (°C) and accuracy for each type of thermocouple

data word 1bit=0.1℃		TypeJ	Type K	Type T	Type E	Type R, S	Type N	±80mV	
decimal	hexadecimal								
32767	7FFF	>1200.0°C	>1372.0°C	>400.0°C	>1000.0°C	>1768.0°C	>1300.0°C	>94.071mV	OF
↑	↑							↑	↑
32511	7EFF							97.071mV	OR
:	:								
27649	6C01							80.0029mV	
27648	6C00							80mV	
:	:					↑			
17680	4510		↑			1768.0°C			
:	:						↑		
13720	3598		1372.0°C out of range						
:	:								
13000	32C8	↑	1300.0°C				1300.0°C		NR
:	:								
12000	2EE0	1200.0°C			↑				
:	:								
10000	2710			↑	1000.0°C				
:	:								
4000	0FA0			-400.0°C		400.0°C			
:	:								
1	0001	0.1°C	0.1°C	0.1°C	0.1°C	0.1°C	0.1°C	0.0029mV	
0	0000	0.0°C	0.0°C	0.0°C	0.0°C	0.0°C	0.0°C	0.0mV	
-1	FFFF	-0.1°C	-0.1°C	-0.1°C	-0.1°C	-0.1°C	-0.1°C	-0.0029mV	
:	:					below range			
-500	FE0C					-50.0°C			
-1500	FA24	-150.0°C				↓			
:	:								
-2000	F830	below range	-200.0°C						
:	:								
-2100	F7CC	-210.0°C							
:	:		below range						
-2550	F60A			-255.0°C	-255.0°C				
:	:			below range	below range				
-2700	F574	↓	-270.0°C	-270.0°C	-270.0°C		-270.0°C		
:	:								
-27648	9400		↓	↓	↓		↓	-80mV	
-27649	93FF							-80.0029mV	UR
:	:								
-32512	8100							-94.071mV	
#	#							↓	↓
-32768	8000	<-210.0°C	<-270.0°C	<-270.0°C	<-270.0°C	<-50.0°C	<-270.0°C	<-94.07mV	UF
Full Scale Range Accuracy		S0.1%	S0.3%	S0.6%	S0.1%	S0.6%	S0.1%	S0.1%	
Accuracy (Rated range without cold-junction compensation)		S1.5°C	S1.7°C	S1.4°C	S1.3°C	S3.7°C	S1.6°C	S0.10°C	
cold-junction error		S1.5°C	S1.5°C	S1.5°C	S1.5°C	S1.5°C	S1.5°C	NA	

↑ Indicates that all analog quantities greater than this value but less than the short-line threshold are reported as overflow values, 32767 (0X7FFF)

↓ Indicates that all analog quantities less than this value but greater than the short-line threshold are reported as underflow values, -32768 (0X8000)

8.1.5 Thermal resistance characteristics

Temperature Range (°C) and Accuracy for RTD Types

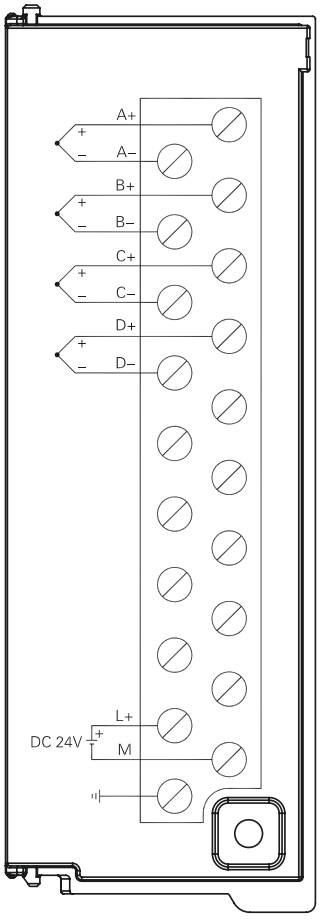
system word 1bit=0.1℃		Pt10000	Pt100 Pt200 Pt500 Pt1000	Ni100 Ni120 Ni1000	Cu9.035	0-150Ω	0-300Ω	0-600Ω	
decimal	hexadecimal								
32767	7FFF								
32766	7FFE					↑	↑	↑	
32511	7EFF					176.383Ω	352.767Ω	705.534Ω	
29649	6C01					150.005Ω	300.011Ω	600.022Ω	
27648	6C00					150.000Ω	300.000Ω	600.000Ω	
25000	61AB								↑
18000	4650								OR
15000	3A98								
13000	32C8	↑	↑						
10000	2710	1000.0°C	1000.0°C						
8500	2134		850.0°C						
6000	1770	600.0°C			↑				
3120	0C30			↑	312.0°C				
2950	0B86			295.0°C					
2600	0A28				260.0°C				
2500	09C4			250.0°C					
1	0001	0.1°C	0.1°C	0.1°C	0.1°C	0.005Ω	0.011Ω	0.022Ω	
0	0000	0.0°C	0.0°C	0.0°C	0.0°C	0.000Ω	0.000Ω	0.000Ω	
-1	FFFF	-0.1°C	-0.1°C	-0.1°C	-0.1°C		(no negative values)		
						↓	↓	↓	
-600	FDA8			-60.0°C					
-1050	FBE6			-105.0°C					
				↓					
-2000	F830	-200.0°C	-200.0°C		-200.0°C				
-2400	F6A0				-240.0°C				
-2430	F682	-243.0°C	-243.0°C		↓				
		↓	↓						
-5000	EC78								
-6000	E890								UR
-10500	D6FC								↓
-12000	D120								
-20000	4E20								
-32767	8001								
-32768	8000								
Full Scale Range Accuracy		±0.4%	±0.1%	±0.2%	±0.5%	±0.1%	±0.1%	±0.1%	
Rated range accuracy		±4°C	±1°C	±0.6°C	±2.8°C	±0.15°C	±0.3°C	±0.6°C	
*OF=Overflow, OR=Out of Range, NR=Rated Range, UR=Under Range, UF=Underflow									

↑ ↓ Indicates that all analog values that exceed or fall below this limit are reported as the selected fuse calibration value, 32767 (0X7FFF) or -32768 (0X8000)

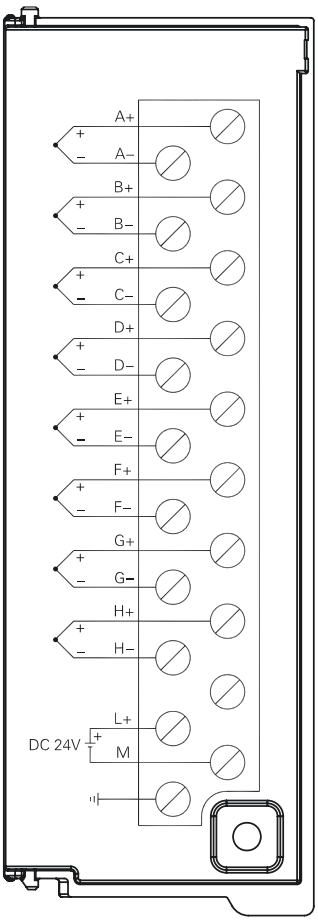
8.2 Wiring

8.2.1 Thermocouple wiring

◆ AIT-040S1

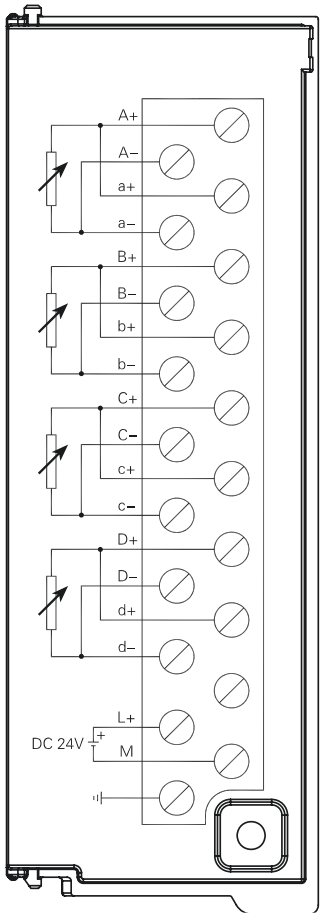


◆ AIT-080S1

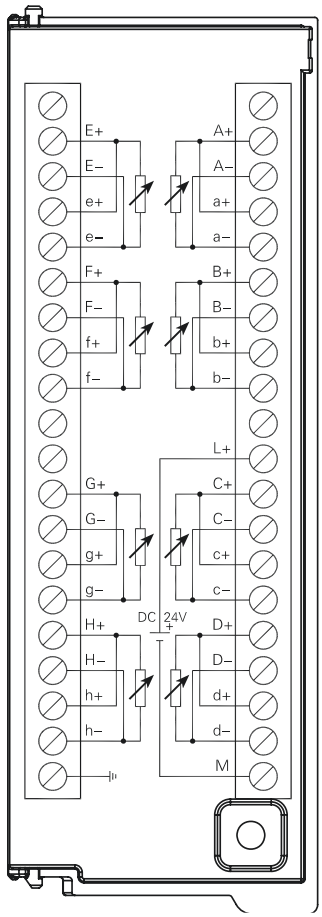


8.2.2 Thermal Resistance Wiring

◆ AIR-040S1



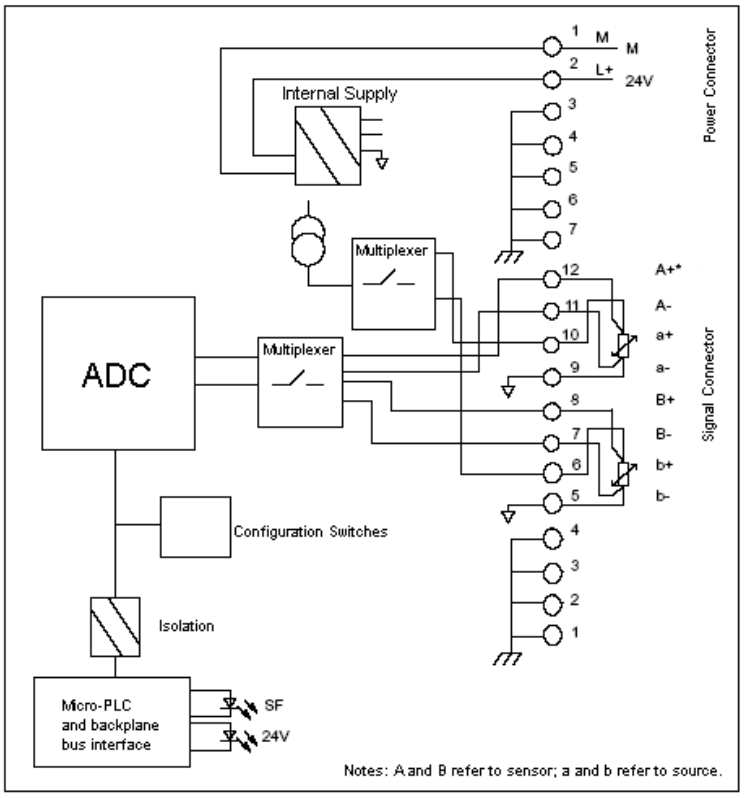
◆ AIR-080S1

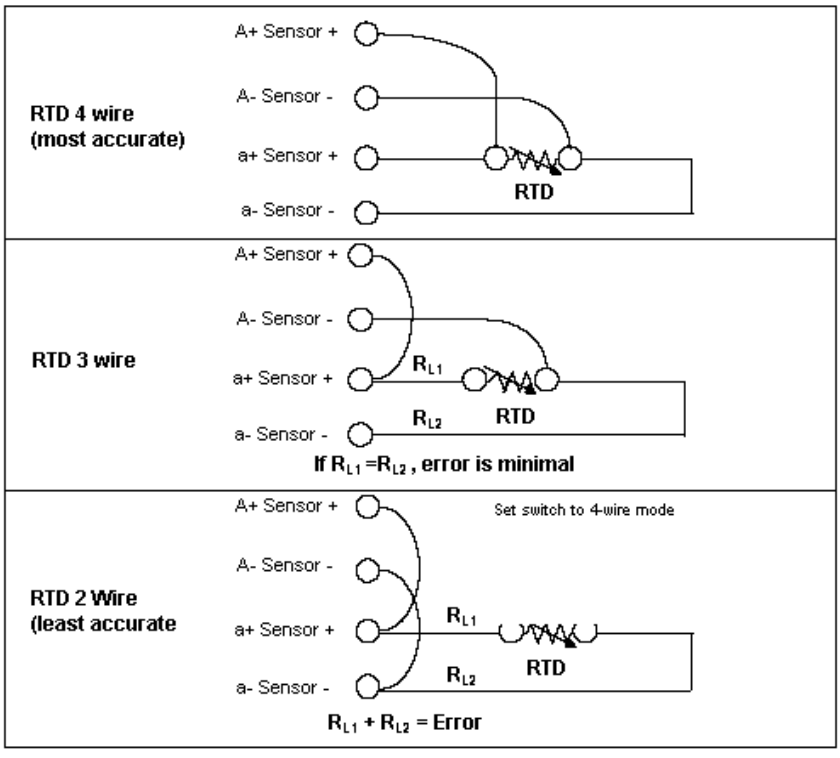


Wiring between RTD and sensor

As shown in the figure below, the user can connect the thermal resistance sensor to the thermal resistance module by direct wiring or expansion line. The best anti irritability can be achieved by using shielded wire. If shielded wire is used, the shielded wire shall be connected to the 1-4 pin grounding point of the signal connector. This ground point is grounded with pins 3 to 7 of the power connector. If some thermal resistance input channels are not used, the user shall connect a resistor with the unused input channel to prevent the error caused by the floating input signal from affecting the false display of the effective channel.

The user needs to connect the power supply to pins 1 and 2 of the power connector, and pin 3 of the power connector must be connected to the nearby chassis ground. One of the following three wiring modes (2-wire system, 3-wire system and 4-wire system) can be selected to connect the thermal resistance module with the sensor. The 4-wire connection method has the highest accuracy and the 2-wire connection method has the lowest accuracy. It is recommended to use the 2-wire connection method only when the conductor error is not concerned in the application.


Internal Supply
Multiplexer
Multiplexer
ADC
Configuration Switches
Isolation
Micro-PLC
and backplane
bus interface
SF
24V
1 M
2 L+
M
24V
Power Connector
3
4
5
6
7
12
11
10
9
8
7
6
5
4
3
2
1
A+*
A-
a+
a-
B+
B-
b+
b-
Signal Connector
Notes: A and B refer to sensor; a and b refer to source.


RTD 4 wire
(most accurate)
A+ Sensor +
A- Sensor -
a+ Sensor +
a- Sensor -
RTD
RTD 3 wire
A+ Sensor +
A- Sensor -
a+ Sensor +
a- Sensor -
RL1
RL2
RTD
If RL1 =RL2 , error is minimal
RTD 2 Wire
(least accurate
A+ Sensor +
A- Sensor -
a+ Sensor +
a- Sensor -
Set switch to 4-wire mode
RL1
RL2
RTD
RL1 + RL2 = Error
Notes: A refers to sensor; a refers to source

High-Speed Counting Module

9

This chapter mainly introduces the high-speed counting module of H56-10/H52-10 application system, as follows:

This chapter mainly introduces the following contents of CTH300-H HSC-02 high-speed counting module:

- ❑ Technical specifications
- ❑ Wiring specification
- ❑ Software configuration specification

Table 9-1 basic properties of high speed counting module hsc-02

Specification	Order No
High speed counting module HSC-02, 2-way differential / single ended signal input	CTH3 HSC-020S1

9.1 Technical Specifications

Table 9-2 general characteristics of HSC-02

physical characteristics		
Dimension（W×H×D）		34×115×100 mm
Power characteristics		
Bus supply voltage		+5V DC
Bus supply current		130mA
LED indicator characteristics		
Signal indicator		ON: there is an input signal. OFF: no input signal
Sensor connection		
Number of input channels		2
Signal type	Differential input	Signal voltage: 5VDC Maximum input frequency: 2MHz
	Single ended input	Signal voltage: 24VDC Maximum input frequency: 500KHz Allowable range of signal duty cycle: 40% - 60%
Maximum protection voltage of signal input		30VDC
Input filtering		Configurable, 125kHz / 250kHz / 500kHz / 1MHz / 2MHz
Positive transaction code		1, 2 and 4 frequency doubling
Counter format		32position
Counter reset function		Support, Z signal
Counter capture function		Support, Z signal
Multi counter synchronous counting function		Support, int signal
Int signal voltage		24VDC
Maximum input frequency of int signal		500kHz
Int signal input filtering		Configurable, 125kHz / 250kHz / 500kHz
Photoelectric isolation		500VAC,1min

9.2 Wiring

The wiring of the high-speed counting module is shown on the right. The module can adopt two modes: single ended signal output and differential signal output. When connecting different drivers, the specific connection methods are as follows:

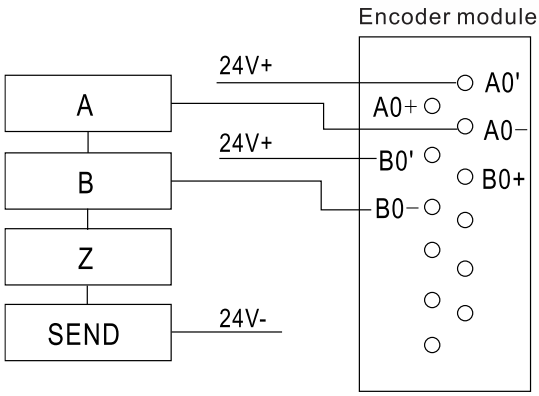


Figure 9-1 E10 servo single ended signal output connection encoder module

Servo feedback pulse output point

A+	21 A phase output+
A-	21 A phase output-
B+	48 B phase output+
B-	49 B phase output-
Z+	23 Z phase output+
Z-	24 Z phase output-
VSS	25 Digital signal ground
CZ	19 Encoder Z phase output

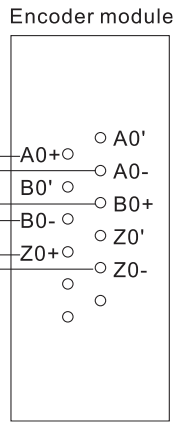
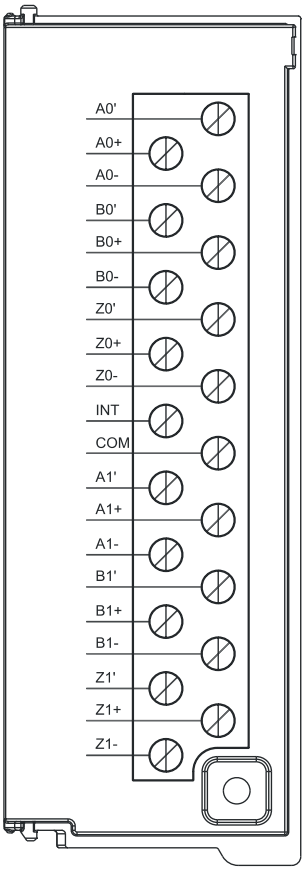


Figure 9-2 H1A servo differential signal output connection encoder module



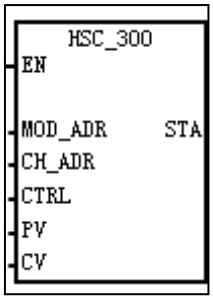
9.3 Software Configuration

High speed counting module HSC-02 supports instruction library HSC_300_Lib, the following sections are instructions for the library.

Table 9-3 interrupt events

Counter	Interrupt event number	Priority	Interrupt description
HSC0	12	1	CV = PV interrupt
	27	1	Count direction change interrupt
	28	1	External reset Z signal interrupt
HSC1	13	1	CV = PV interrupt
	14	1	Count direction change interrupt
	15	1	External reset Z signal interrupt

9.3.1 HSC_300(Configure the counter)



Function: configure counter parameters.

Table 9-4 HSC_300 parameter description

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address Bit0 ~ Bit3: slot number Bit4 ~ Bit7: rack number	BYTE		Modules on Y (3 ~ 10) of rackx (0 ~ 3) For example, rack16 is module 6 of the first rack.
CH_ADR	IN	channel 0: HSC0. 1: HSC1	BYTE	0~1	
CTRL	IN	Control word	BYTE		See table 9-5
PV	IN	The preset value	DINT		
CV	IN	The current value	DINT		
STA	OUT	Return status	BYTE	0~255	0: OK 1: The parameter is incorrect 2: The access module fails

Tips

MOD_ADR module address describes the value range of rack number and slot number. This parameter is common in multiple HSC library instructions, so the relevant settings are the same.

Example

In the figure, two HSCs are hung at the 4th and 5th module positions of the INT-00, while MOD_ ADR depends on the rack number of the module in the hardware configuration and the corresponding slot number (16#16 and 16#17), rather than the module position (16#14 and 16#15).

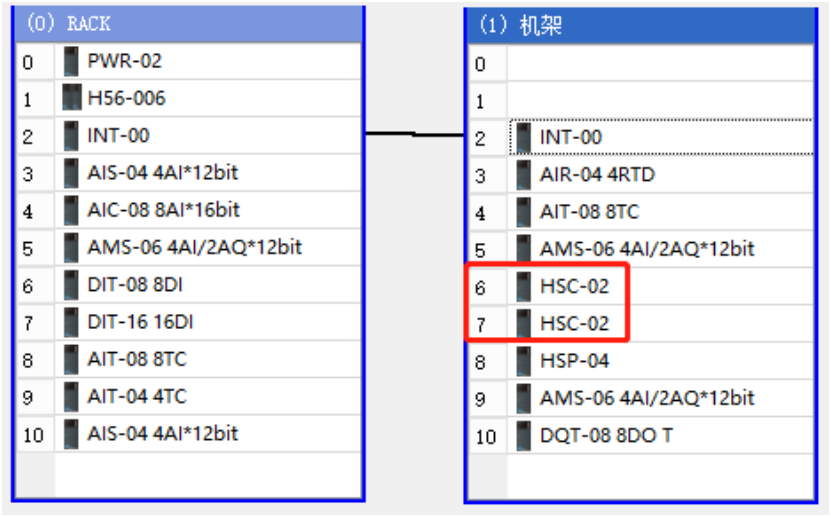


Table 9-5 control word (R / W)

7	6	5	4	3	2	1	0
hsc_en	hsc_cv_update	hsc_pv_update	hsc_dir_update	hsc_dir	quad_rate		reset_level

reset_Level: reset level, 1 -- high level reset, 0 -- low level reset.

quad_rate[1:0]: Quadrature counting selection, 00--4x multiple, 01--2x multiple, 10--1x multiple.

hsc_dir: counting direction, 0-- count down, 1-- count up.

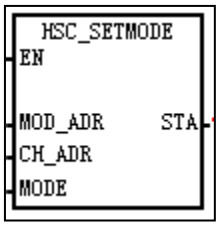
hsc_dir_update: count direction update, 0-- not update, 1—update.

hsc_pv_update: default value update, 0-- not update, 1—update.

hsc_cv_update: the current value is updated, 0--not update, 1—update.

hsc_en: count enabled, 0-- not enabled, 1—enabled.

9.3.2 HSC_SETMODE(Set mode command)



Function: set counter mode.

Table 9-6 HSC_ SETMODE parameter description

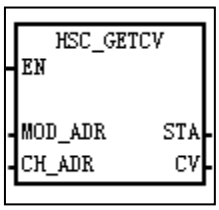
Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address Bit 0~ Bit 3: slot number Bit 4~ Bit 7: rack number	BYTE		Modules on Y (3 ~ 10) of rackx (0 ~ 3) For example, rack16 is module 6 of the first rack.
CH_ADR	IN	Channel	BYTE	0~1	
MODE	IN	Counting mode	BYTE		Bit0~Bit3: HSC

					counting mode (see table 9-7) Bit4: Z signal latch function, 0: latch, 1: not latch Bit5: Z signal zeroing function, 0: zeroing, 1: not zeroing Bit6: reserved Bit7: Latch value reset 0: invalid, 1: valid.
STA	OUT	Return status	BYTE	0~255	0: OK 1: The parameter is incorrect 2: The access module fails

Table 9-7 counter mode

Mode	Describe	Input			Software control
	HSC0	A0	B0	Z0	INT
	HSC1	A1	B1	Z1	
0	Single phase counter with internal direction control	Clock	-	-	-
1			-	Reset	-
2			-		Start (external synchronization)
3	Single phase counter with external direction control	Clock	direction	-	-
4				Reset	-
5					Start (external synchronization)）
6	Dual phase counter with 2 clock inputs	Up clock	Down clock	-	-
7				Reset	-
8					Start (external synchronization)）
9	A/B phase quadrature counter	Clock A	Clock B	-	-
10				Reset	-
11					Start (external synchronization)

9.3.3 HSC_GETCV(Get current count instruction)



Function: get current count value

Table 9-8 HSC_ GETCV parameter description

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address Bit0 ~ Bit3: slot number Bit4 ~ Bit7: rack number	BYTE		Modules on Y (3 ~ 10) of rackx (0 ~ 3) For example, rack16 is module 6 of the first rack.
CH_ADR	IN	Channel	BYTE	0~1	
CV	OUT	Current count value	DINT		Current count value
STA	OUT	Return status	BYTE	0~255	0: OK 1: The parameter is incorrect 2: The access module fails

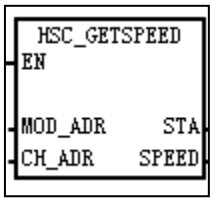
9.3.4 HSC_GETSTA(Get current count status instruction)

Function: get the current count status.

Table 9-9 HSC_ GETSTA parameter description

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address Bit0 ~ Bit3: slot number Bit4 ~ Bit7: rack number	BYTE		Modules on Y (3 ~ 10) of rackx (0 ~ 3) For example, rack16 is module 6 of the first rack.
CH_ADR	IN	Channel	BYTE	0~1	
HSC_STA	OUT	Counter status	BYTE		Bit0~Bit3:current mode. Bit4: reserved. Bit5: HSC0 current counting direction: 1 = count up. Bit6 = 1: the current value = preset value. Bit7 = 1: the current > the preset value.
STA	OUT	Return status	BYTE	0~255	0: OK 1: The parameter is incorrect 2: The access module fails

9.3.5 HSC_GETSPEED(Get current speed command)

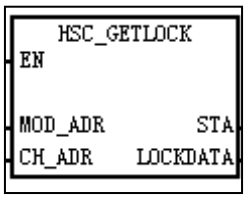


Function: get the current speed.

Table 9-10 HSC_Parameter description of getspeed instruction

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address Bit0 ~ Bit3: slot number Bit4 ~ Bit7: rack number	BYTE		Modules on Y (3 ~ 10) of rackx (0 ~ 3) For example, rack16 is module 6 of the first rack.
CH_ADR	IN	channel	BYTE	0~1	
SPEED	OUT	The current speed	DWORD		Hz
STA	OUT	Return status	BYTE	0~255	0: OK 1: The parameter is incorrect

9.3.6 HSC_GETLOCK(Get current Latch value instruction)

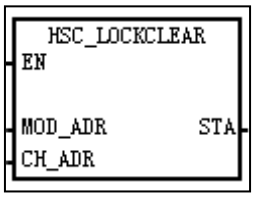


Function: get the current Latch value.

Table 9-11 HSC_ GETLOCK parameter description

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address Bit0 ~ Bit3: slot number Bit4 ~ Bit7: rack number	BYTE		Modules on Y (3 ~ 10) of rackx (0 ~ 3) For example, rack16 is module 6 of the first rack.
CH_ADR	IN	channel	BYTE	0~1	
LOCKDATA	OUT	Current lock value	DINT		Current lock value
STA	OUT	Return status	BYTE	0~255	0: OK 1: The parameter is incorrect

9.3.7 HSC_CLEARLOCK(Clear Latch value command)



Function: clear the latch value.

Table 9-12 HSC_ clearlock parameter description

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address Bit 0 ~ bit 3: slot number Bit 4 ~ bit 7: rack number	BYTE		Modules on Y (3 ~ 10) of rackx (0 ~ 3) For example, rack16 is module 6 of the first rack.
CH_ADR	IN	channel	BYTE	0~1	
STA	OUT	Return status	BYTE	0~255	0: OK 1: The parameter is incorrect

Pulse Output Module

10

Introduce the pulse output module of H56-10 / H52-10 application system, as follows:

CTH300-H HSP-04 pulse output modules are used for multi-axis motion control and support bus expansion. Up to 8 HSP modules can be hung behind each CPU or EtherCAT slave (4 racks can be hung behind a CPU, and a total of 8 HSP modules can be hung in the 4 racks).

This chapter mainly introduces the following contents:

- ❑ General Features
- ❑ Wiring Specifications
- ❑ Software Configuration Specifications

Table 10-1 Basic properties of pulse output module HSP-04

Name	Specification Description	Order number
Pulse output module	4 differential/single-ended signal outputs	CTH3 HSP-040S1

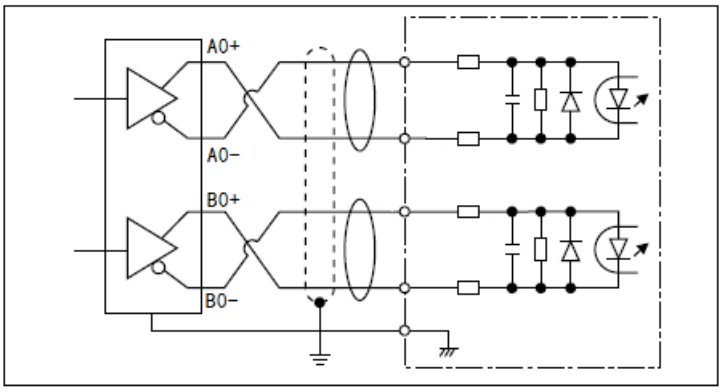
10.1 Specifications

Table 10-2 General characteristics of HSP-04

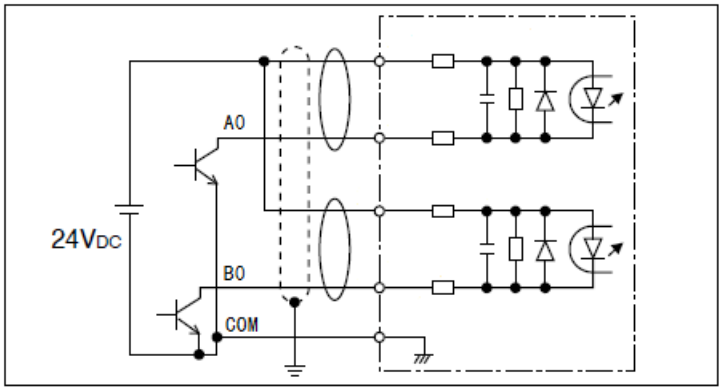
Physical properties		
Dimension（W×H×D）	34×115×101.6 mm	
Power characteristics		
Rated input voltage	24V DC	
Input voltage range	20.4V~28.8V DC	
Input Current	100mA	
Reverse polarity protection	YES	
bus supply voltage	+5V DC	
Bus supply current	100mA	
LED Indicator		
Signal indicator	ON: with input signal. OFF: without input signal	
Output		
Number of output channels	4	
Output type	differential signal	Single-ended (NPN) signaling
Maximum output frequency	4MHz	500KHz
Output signal duty cycle	- -	50%
Rated output voltage	5VDC	5~24VDC
Output voltage range	0~5.5VDC	5~28.8VDC
Output signal logic "1"	3.8V(Min)	0.5V(Max)
Output signal logic "0"	0.3V（Max）	Vcc-0.5V(Min)
Inrush current	8A for 100ms	
Current per point (max)	20mA	
Maximum current per common	No	160mA
Leakage current (max)	10μA	
Isolate	500VAC for 1min	

10.2 Wiring

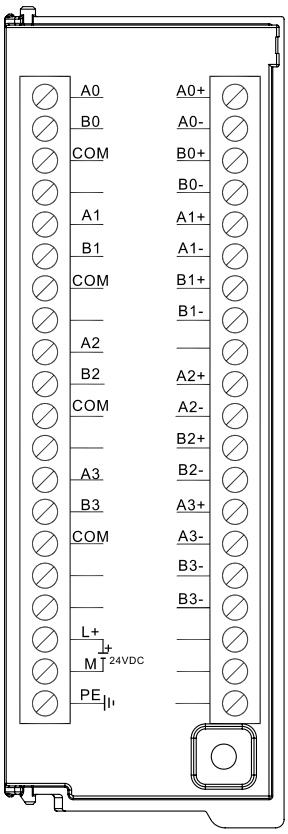
1. The right half of the wiring diagram is the differential output, there are 4 axes in total, and each axis has two pairs of differential outputs, which are pulse and direction respectively (such as A0+, A0- for pulse, B0+, B0- for direction), output frequency up to 4MHz.
2. The left half of the wiring diagram is for single-ended output and 24VDC power supply. The single-ended output has a total of 4 axes. Each axis has two sets of outputs, which are pulse and direction respectively (for example, A0 is pulse, B0 is direction, and COM is common terminal). The output frequency is up to 500kHz. The wiring methods are as follows:



The differential output



Single-ended output



10.3 Software Configuration Specifications

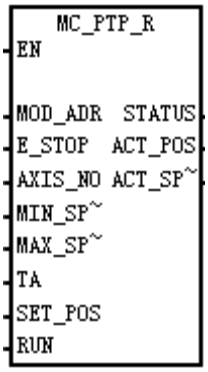
The HSP-04 pulse output module is used in conjunction with the pulse output instruction library Hsp_libv1.4 in the MagicWorks PLC programming software, and contains the following control instructions:

Table 10-3 Pulse output control commands

Name	Function
MC_INIT_DIR	Configure the motor direction command
MC_PTP_R	Single axis relative motion command
MC_PTP_A	Single axis absolute position command
MC_SPEED_CTRL	Speed control command

MC_SET_POS	Set the current absolute position
MC_READPOS	Read the current absolute coordinates
MC_SET_MODE	Set output mode command

10.3.1 MC_PTP_R(Single axis relative motion command)



Function: used for single-axis point-to-point control (single-axis fixed-length drive). A fixed pulse can be output when called once. By setting the maximum, minimum speed and acceleration/deceleration time, the output pulse will gradually accelerate to the maximum speed when starting. When the number of pulses is about to run out, the pulse frequency will automatically decrease , In order to prevent the inertia of the machine when starting or stopping is too large to cause vibration or jamming.

Table 10-4 Instruction parameter description

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address	Byte	High 4 bits 0~3 Low 4 bits 3~10	Bit4~Bit7: rack number Bit3~Bit0: slot number
E_STOP	IN	Emergency stop position. 1: Effective 0: invalid	Bool	0/1	1. Only run when Run is 1 and E_Stop is 0. 2. When E_STOP is 1, RUN is reset internally.
AXIS_NO	IN	Set the axis number	Byte	0~3	This parameter cannot be modified during operation.
MIN_SPEED	IN	The minimum speed is the speed when starting or stopping. Unit: HZ	Dword	100~4000000	1. The minimum speed setting should be less than the maximum speed. 2. This parameter can be modified during operation. 3. The maximum speed of single axis is set to 500K, and the maximum
MAX_SPEED	IN	The maximum speed is the maximum speed in operation. Unit: HZ	Dword	100~4000000	

					speed of difference is 4M.
TA	IN	Acceleration/deceleration time. Unit: ms	Dword	0~10000	This parameter can be modified during operation
SET_POS	IN	The number of output pulses is divided into positive and negative. The number of positive pulses indicates the positive direction along the X axis, and the number of negative pulses indicates the negative direction along the X axis.	Dint	-2147483648 ~ +2147483647	This parameter can be modified during operation. When the new set value is greater than the number of pulses that have been output, the last output pulse will be subject to the new set value. When the new set value is less than the number of output pulses, the pulse output will be stopped immediately.
RUN	IN/OUT	Run enable bit. 1: Effective 0: invalid	Bool	0/1	1. It runs only when RUN is 1 and E_STOP is 0. 2. When the operation is completed, RUN resets internally. 3. When E_STOP is 1, RUN resets internally. 4. When the instruction is in the running state, set RUN to 0 to realize the software stop function.
STATUS	OUT	Output status byte: 7 6 5 4 3 2 Bit0: Parameter configuration error flag 1—Parameter configuration error 0—Parameter configuration is normal Bit1: Run sign 1—It is running,	Byte	0~255	Bit0: 1. Only judge the axis parameters and TA and MOD_ADR configuration errors. 2. MIN_SPEED/ Parameters such as MAX_SPEED will not be reported as an error, and will be automatically set to the closest reasonable value.

		the instruction is outputting pulses, and the instruction has not been executed. 0—No operation, because the common resource is occupied by other instructions, the instruction has not been executed yet. or the instruction has been executed. Bit2: Finish flag 1—Completed, the instruction is executed. 0—Incomplete, the instruction has not been executed or the instruction is being executed but not completed. Bit3: Busy sign 1—The Busy flag is valid and the axis is being occupied by other commands. 0—The Busy flag is invalid, the instruction is being executed or the execution has completed. Bit4: Module status. 1-- Module access error. 0-- There is no error in module access.			

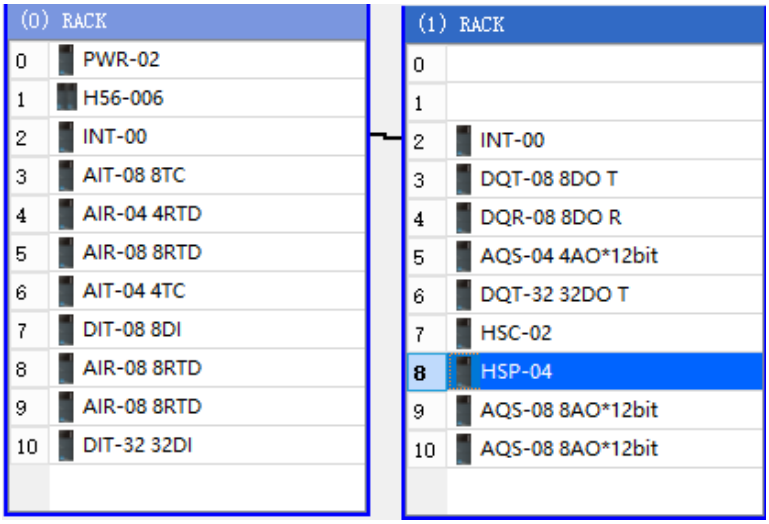
		Bit6: Aborted by other instructions			
ACT_POS	OUT	The current relative coordinate or the number of pulses output by this command.	Dint	-2147483648~ +2147483647	
ACT_SPEED	OUT	The current actual running speed.	Dword	100~4000000	

Tips

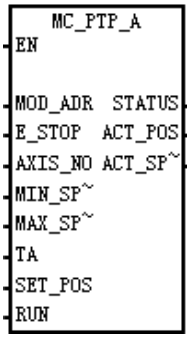
The MOD_ADR module address indicates the value range of the rack number and slot number. This parameter is common in multiple HSP library instructions, so the relevant settings are the same.

Example

HSP is hung at the sixth module position of the second relay, and MOD_ADR depends on the rack number and corresponding slot number (16#18) of the module in the hardware configuration, not the position of the module (16#) 16).



10.3.2 MC_PTP_A(Single axis absolute motion command)



Function: used for single axis point-to-point control (not fixed length, but fixed point). Once called, the pulse can be output to the specified pulse number based on the original pulse number. Through the setting of maximum and minimum speed and acceleration and deceleration time, the output pulse will gradually accelerate to the maximum speed when starting. When the pulse number is about to run out, the pulse frequency will be automatically reduced to prevent vibration or jamming caused by too large inertia of the machine when starting or stopping.

Table 10-5 Instruction parameter description table:

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address	Byte	High 4 bits 0~3 Low 4 bits 3~10	Bit4~Bit7: rack number Bit3~Bit0: slot number
E_STOP	IN	Emergency stop position. 1: Effective 0: invalid	Bool	0/1	1. Only run is 1 and E_ Run when stop is 0. 2. When e_ When stop is 1, run resets internally.
AXIS_NO	IN	Set the axis number	Byte	0~3	This parameter cannot be modified during operation.
MIN_SPEED	IN	The minimum speed is the speed when starting or stopping. Unit: Hz	Dword	100~4000000	1. The minimum speed setting should be less than the maximum speed. 2. This parameter can be modified during operation. 3. The maximum single-axis speed is set to 500K, and the maximum differential speed is 4M.
MAX_SPEED	IN	Maximum speed, unit: Hz	Dword	100~4000000	
TA	IN	Acceleration/deceleration time. Unit: ms	Dword	0~10000	This parameter can be modified during operation
SET_POS	IN	The number of output pulses is divided into positive and negative. The	Dint	-2147483648 ~ +2147483647	This parameter can be modified during operation. When the new set value is greater

		number of positive pulses indicates the positive direction along the X axis, and the number of negative pulses indicates the negative direction along the X axis.			than the number of output pulses, the last output pulse will be subject to the new set value. when the new set value is less than the number of output pulses, the pulse output will be stopped immediately.
RUN	IN/OUT	Run enable bit. 1: Effective 0: invalid	Bool	0/1	1. Only RUN ==1 and E_STOP ==0 can run. 2. When the operation is completed, RUN is reset internally. 3. When E_STOP is 1, RUN is reset internally. 4. When the command is in the running state, set RUN to 0 to realize the soft stop function.
STATUS	OUT	Output status bytes: 7 6 5 4 3 2 1 0 Bit0: parameter configuration error flag 1- Parameter configuration error 0 - parameter configuration is normal Bit1: Running flag 1- Running, the command is outputting pulses, and the command has not been	Byte	0~255	Bit 0: 1. Only judge the axis parameters, Ta / TD and MOD_ ADR configuration error. 2.MIN_ SPEED/SET_ Parameters such as speed will not report errors and will be automatically set to the nearest reasonable value

		executed. 0-Not running, because the common resources are occupied by other instructions, so the instruction is not running. or the instruction has been executed. Bit2: Completion flag 1 - Done, the instruction is executed. 0 - Not completed, the instruction is not executed or the instruction is executing but not completed. Bit3: busy flag 1 - The busy flag is valid, the axis is being occupied by other commands. 0 - The busy flag is invalid, the instruction is executing or this execution has completed.Bit4: Module status. 1 - Module access error. 0 - No errors occurred in module access. Bit6: Aborted by other instructions			
ACT_POS	OUT	The current absolute coordinates	Dint	-2147483648~+2147483647	
ACT_SPEED	OUT	Actual running speed.	Dword	100~4000000	

10.3.3 MC_SPEED_CTL(Speed control command)

Function: control the frequency of single axis output pulse, and change the frequency (speed) of output pulse at any time. When receiving the soft stop command, it will automatically decelerate and stop. When the emergency stop command is received, the pulse output will be stopped immediately without deceleration.

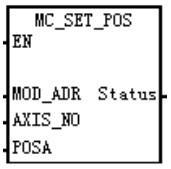
Table 10-6 Instruction parameter description table:

Parameter	Input/output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address	Byte	High 4 bits 0~3 Low 4 bits 3~10	Bit4~Bit7: rack number Bit3~Bit0: slot number
RUN	IN	Run enable bit. 1: effective, 0: invalid.	Bool	0/1	1. Only run is 1 and E_ Stop = 0 and soft_ It can only run when stop is 0. 2. When the operation is completed, run resets internally.
E_STOP	IN	Emergency stop position. 1: Valid, 0: invalid. When an effective emergency stop command is received, the output pulse will stop immediately without deceleration.	Bool	0/1	It can only run when RUN is 1 and E_Stop is 0 and Soft_Stop is 0.
SOFT_STOP	IN	Soft stop bit. 1: Valid, 0: invalid. When a valid soft stop command is received, the output pulse will	Bool	0/1	It can only run when RUN is 1 and E_Stop is 0 and SOFT_STOP is 0.

<table>
<tr><td></td><td></td><td>decelerate and stop.</td><td></td><td></td><td></td></tr>
<tr><td>DIR</td><td>IN</td><td>Pulse direction</td><td>Bool</td><td>0/1</td><td rowspan="2">This parameter cannot be modified during operation.</td></tr>
<tr><td>AXIS_NO</td><td>IN</td><td>Set the axis number</td><td>Byte</td><td>0~3</td></tr>
<tr><td>MIN_SPEED</td><td>IN</td><td>The minimum speed is the speed when starting or stopping.
Unit: Hz</td><td>Dword</td><td>0~4000000</td><td rowspan="2">1. When the set speed is 0, there is no pulse output. When the minimum speed is set to be non-0 and less than 100, the module defaults to 100. If the set speed is less than the minimum speed, the module modifies the minimum speed value to the set speed value by default.
2. SET_ Speed can be modified during operation. MIN_ Speed can be modified during operation.</td></tr>
<tr><td>SET_SPEED</td><td>IN</td><td>Set the speed. Before receiving the stop command, the output pulse will accelerate or decelerate to this speed.</td><td>Dword</td><td>0~4000000</td></tr>
<tr><td>TA</td><td>IN</td><td>Acceleration time, the acceleration time from the minimum speed to the set speed.
Unit: ms</td><td>Dword</td><td>0~10000</td><td rowspan="2">This parameter can be modified during operation.</td></tr>
<tr><td>TD</td><td>IN</td><td>Deceleration time, the deceleration time from the set speed to the minimum speed.
Unit: ms</td><td>Dword</td><td>0~10000</td></tr>
<tr><td>STATUS</td><td>OUT</td><td>Output status byte:
<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>Bit0: parameter configuration error flag
1-parameter configuration error
0-parameter configuration is normal
Bit1: Operation flag
1-running, the</td><td>BYTE</td><td>0~255</td><td>Bit 0:
1. Only judge the shaft parameters, Ta / TD and mod_ ADR configuration error.
2.MIN_ SPEED/SET_ Parameters such</td></tr>
</table>

		instruction is outputting pulses, and the instruction is not finished. 0-not running, because the public resources are occupied by other instructions, the instructions have not been run. Or the command has finished running. Bit2: completion flag 1-complete, instruction execution completed. 0-incomplete, instruction not executed or instruction in execution but not completed. Bit3: busy flag 1-the busy flag is valid, and the axis is occupied by other commands. 0-the busy flag is invalid, the instruction is executing or the execution has completed. Bit5 ~ bit6: reserved Bit7: command communication status flag 1-Communication timeout. 0-no timeout			as speed will not report errors and will be automatically set to the nearest reasonable value.
ACT _SPEED	OUT	Current speed (frequency) output.	Dword	100~400000	The value may deviate slightly from the actual value.

10.3.4 MC_SET_POS(Set the current absolute coordinates)

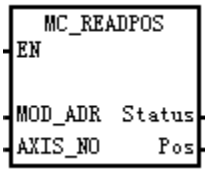


Function: Set the current absolute coordinates. This command can only be used when the axis is stopped.

Table 10-7 Instruction parameter

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address	Byte	High 4 bits 0~3 Low 4 bits 3~10	Bit4~Bit7: rack number Bit3~Bit0: slot number
AXIS_NO	IN	Set the axis number	BYTE	0~3	
POSA	IN	Set coordinates		-2147483648~ +2147483647	-
Status	OUT	Status	BYTE	0~255	0: OK 1: Parameter error 2: Error accessing the module

10.3.5 MC_READPOS(Get the current absolute position command)



Function: Read the absolute coordinate value command of the current position.

Table 10-8 Instruction parameter

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address	Byte	High 4 bits 0~3 Low 4 bits 3~10	Bit4~Bit7: rack number Bit3~Bit0: slot number
AXIS_NO	IN	Set the axis number	Byte	0~3	This parameter cannot be modified during running.
STATUS	OUT	Status	Byte	0~255	0: OK 1: Parameter error 2: Error accessing the module
POS	OUT	Current absolute	Dint	-2147483648~+ 2147483647	

		coordinates			

10.3.6 MC_INIT_DIR(Set the motor direction command)

Function: Configure the direction of the motor.

Tips
This command is executed only once during the first CPU scanning period.

Table 10-9 Instruction parameter description table:

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address	Byte	High 4 bits 0~3 Low 4 bits 3~10	Bit4~Bit7: rack number Bit3~Bit0: slot number
AXIS_NO	IN	Set the axis number	Byte	0~3	This parameter cannot be modified during operation.
DIR	IN	Configure the effective level when the direction signal is positive When DIR is 1, when the corresponding direction axis output is set to "1", the motor is rotating in the forward direction. When DIR is 0, when setting the corresponding direction axis output "0", the motor rotates in the reverse direction.	Bool	0~1	Default value: 0
STATUS	OUT	Status	Byte	0~255	0: OK 1: Parameter error 2: Error accessing the module

10.3.7 MC_SET_MODE(Set axis output mode)

Function: Set the axis output mode

Table 10-10 Instruction parameter description table:

Parameter	Input/ output	Description	Type data	Range	Remark
MOD_ADR	IN	Module address	Byte	High 4 bits 0~3 Low 4 bits 3~10	Bit4~Bit7: rack number Bit3~Bit0: slot number
AXIS_NO	IN	Set the axis number	Byte	0~3	
MODE	IN	Set mode	Byte	0, 1, 2	0: Direction+pulse 1: Positive and negative pulse 2: AB phase mode
Status	OUT	Status	Byte	0~255	0: OK 1: Parameter error 2: Error accessing the module

CAN Master Module

11

Introduce the can master module of H56-10 / H52-10 application system, as follows:

The CAN master communication module CAN-1M is a member of the CTH300-H system expansion module. It belongs to the CANopen master module and is used for CANopen communication with other slave modules.

This chapter mainly introduces the following contents of the CTH300-H CAN-1M master module:

- ❑ Technical specifications
- ❑ Wiring specifications

Table 11-1 Basic characteristics of CAN-1M master module

Name	Description	Order No
CAN-1M master module	CANopen master communication module, 1 CAN communication port	CTH3 CAN-1M0S1

11.1 Specifications

11.1.1 General Characteristics

Table 11-2 General characteristics of CAN-1M

Physical characteristics	
Dimension(W×H×D)	34×115×101.6 mm
Power consumption	2.5W
Power characteristics	
Rated input voltage	24V DC
Input voltage range	20.4V~28.8V DC
Input Current	100mA
Reverse polarity protection	YES
Bus power supply voltage	+5V DC
Bus power current	100mA
LED indicator	
24V	ON: with 24VDC, OFF: without 24VDC
SF	ON: The master is not initialized. Flashing: Initializing. OFF: loading finished
BF	ON: The slave has a diagnosis. OFF: No diagnosis from the station
DX	ON: There is an error in the configuration data. If the number of slave exceeds 32, the number of 20K baud rate slave exceeds 8

11.1.2 Features

Table 11-3 Features of CAN-1M

Function category	Function item	Description
Extensions	Extensions	Provide Bus expansion function
Communication function	CAN interface	Provide 1 CANopen communication interface
Isolation function	Power isolation	Isolation between external power supply and system power supply
	Communication isolation	CAN communication port isolation
Protective function	Power protection	The power supply terminal provides reverse connection protection and surge absorption
	Interface protection	Lightning protection for communication ports

<Remarks> When the CAN_1M module is used as the master, it can be configured to write 300 bytes and read 300 bytes of data at most.

11.1.3 Communication Port Specification

Table 11-4 CAN-1M communication port specifications

CANopen communication							
Communication Interface	1						
Maximum number of primary sites	Up to 8 master can be connected behind a CPU						
Maximum number of slave	Up to 32 slave can be connected behind a master						
Transmission rate (kbit/s)	1000	800	500	250	125	50	20
Maximum length (m)	25	50	100	250	500	1000	2500
isolation	Communication port isolation						

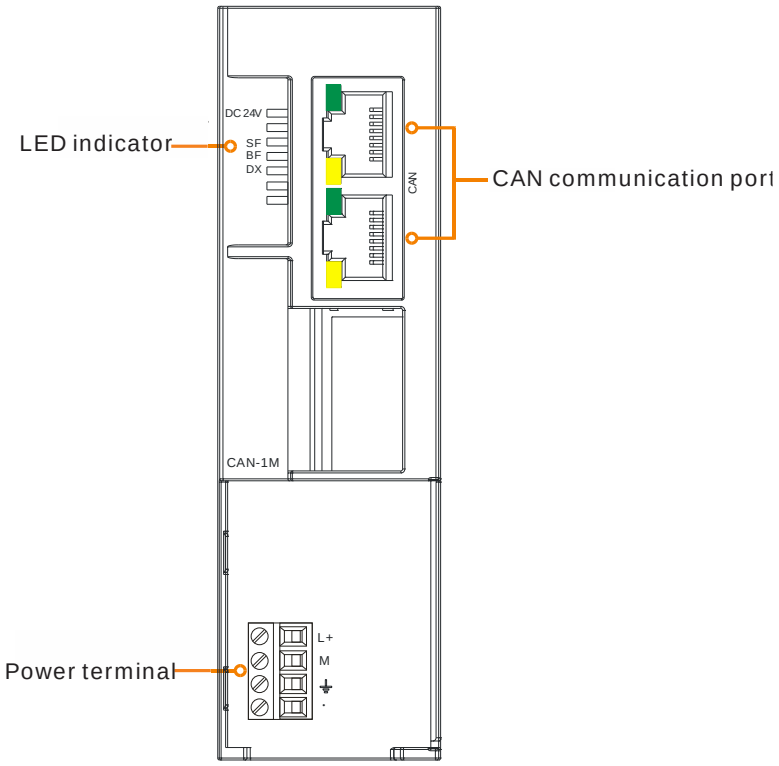
The CAN communication port uses a shielded network cable as the communication cable. The available network cable types are 22AWG~25AWG. The specifications and standards are as follows. The resistance value is the DC resistance value of a single wire. It is recommended to use fully shielded Category 5 cable or fully shielded Category 5e cable, 24AWG**.**

Table 11-5 Communication port wire specifications

AWG	Outer Diameter Metric/ mm	outside diameter imperial/ inch	Cross-sectional area/mm^2	resistance W/km
22	0.643	0.0253	0.3247	54.3
23	0.574	0.0226	0.2588	48.5
24	0.511	0.0201	0.2047	89.4
25	0.44	0.0179	0.1624	79.6

11.2 Wiring

11.2.1 Interface diagram

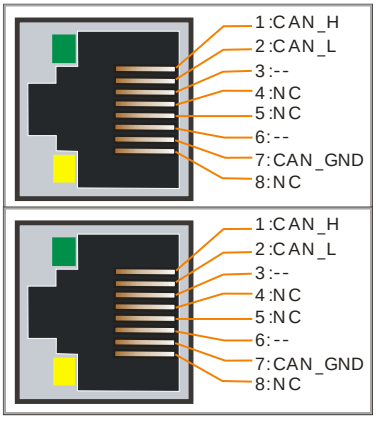


11.2.2 Communication Port Specification

Table 11-6 Power interface definition of CAN-1M

removable terminal	No.	Signal	Signal definition
L+ M ⏚ •	1	L+	24V power supply+
	2	M	24V power supply-
	3	⏚	The earth
	4	--	--

Table 11-7 CAN communication port definition of CAN-1M



No.	signal	description
1	CAN_H	CAN data sending+
2	CAN_L	CAN data sending-
3	--	--
4	--	--
5	--	--
6	--	--
7	CAN_GND	CAN ground
8	--	--
Connector shell	PE	Chassis ground

EtherCAT Slave Module

12

This chapter mainly introduces the EtherCAT slave module of H56-10 / H52-10 application system, as follows:

12.1 Technical Specifications

Table 12-1 Basic characteristics of EtherCAT slave module

Name	Description	Order No.
EtherCAT slave module	EtherCAT slave module, up to 8 IO modules can be expanded, supporting third-party EtherCAT master	CTH3 ECT-000S1

For the specific application of this module, please visit the official website of CORUST to download the "CTH300 EtherCAT Slave Module User Manual" at http://www.co-trust.com

12.1.1 General Characteristics

Table 12-2 general characteristics

Physical characteristics	
Dimension(W×H×D)	34×115×100 mm
Power consumption	2.5W
Power characteristics	
Rated input voltage	24V DC
Input voltage range	20.4V~28.8V DC
Input Current	0.8A
Reverse polarity protection	YES
Bus power supply voltage	+5V DC
Bus power current	1.6A
LED indicator	
24V (green)	ON: with 24VDC, OFF: without 24VDC
SF (red)	ON: EtherCAT slave module failure, OFF: no error
BF (red)	ON: Expansion bus failure, OFF: No error
LINK (green) (Slave status indicator)	ON: normal operation (8), flashing: pre-operation (2), safe operation (4) (see note 1), OFF: no connection (0), initialization (1)
RJ45 (green)	ON: Connect with other EtherCAT interface OFF: No connection with other EtherCAT interfaces Flashing: communicating with other EtherCAT interfaces

Note 1: When the slave expansion bus has no output, the communication between the slave and the master is disconnected, and the slave will not switch to safe operation.

12.1.2 Features

Table 12-3 Features

Function category	Function item	Description
Hardware configuration	Magicworks PLC or other third-party configuration	The added ECT-00 module supports the expansion of 8 slots
Extensions	extensions	Allows to expand 8 I/O modules, supports expanded digital quantity module, analog quantity module, temperature module, HSC module, HSP module, does not support expanded CAN module.
Communication function	Bus interface	Provide expansion module interface, support CTH300 PLC custom 55MHz bus protocol
	EtherCAT interface	EtherCAT interface supports CANopen over EtherCAT (CoE)
Isolation function	Power isolation	Isolate the external power supply from the system power supply
Protective function	Power protection	The power supply terminal provides reverse connection protection and surge absorption

12.1.3 Communication Interface Specification

Table 12-4 Communication port specifications

EtherCAT communication	
Communication interface	1 double RJ45 port
Baud rate	100Mbps
Protocol type	CANopen over EtherCAT (CoE)
	Supports the PDO service
	Support for SDO services
	Supports the EtherCAT state machine command
	Support for third-party EtherCAT master
The communication distance between slave is the longest	100m (100BASE-TX)
isolation	Communication port isolation

The EtherCAT communication port uses a shielded network cable as the communication cable. The available network cable types are 22AWG~25AWG. The specifications and standards are as follows. The resistance value is the DC resistance value of a single wire. It is recommended to use fully shielded Category 5 cable or fully shielded Category 5e cable, 24AWG.

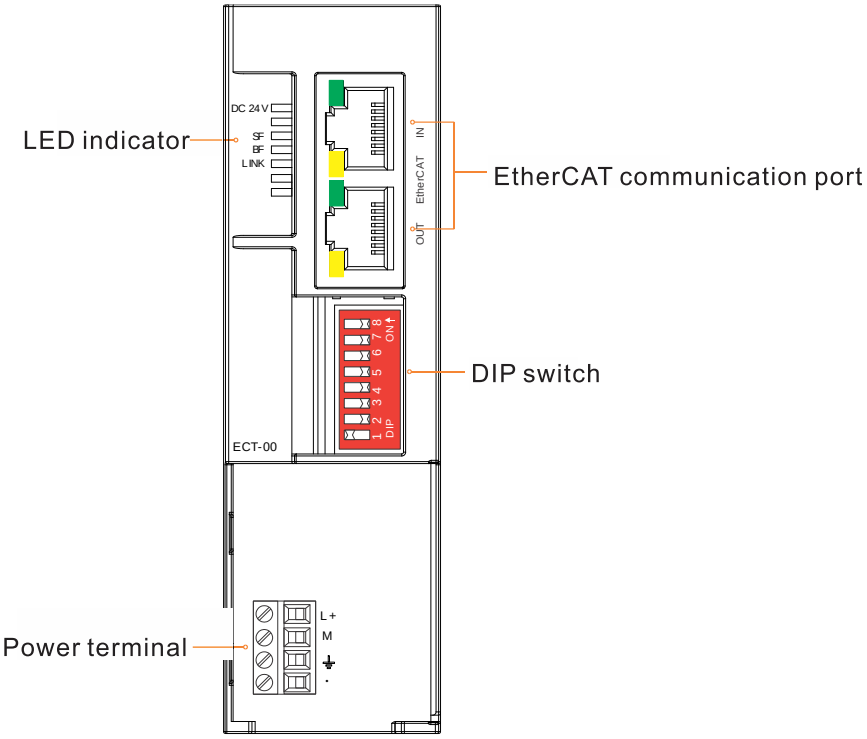
Table 12-5 Communication port wire specifications

AWG	Outer diameter Metric/ mm	Outside diameter imperial/ inch	Cross-sectional area/mm²	Resistance W/km
22	0.643	0.0253	0.3247	54.3
23	0.574	0.0226	0.2588	48.5
24	0.511	0.0201	0.2047	89.4
25	0.44	0.0179	0.1624	79.6

It is recommended to use the super five types of shielded crystal head, as shown in the figure below:

12.2 Wiring

12.2.1 Interface diagram

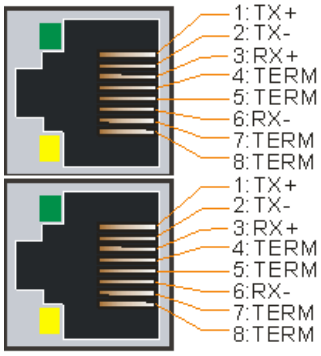


12.2.2 Interface definition

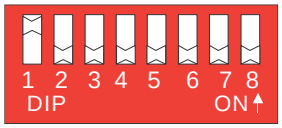
Table 12-6 Power port definitions

Removable Terminals	No.	Signal	Signal definition
L+ M ⏚ •	1	L+	24V power supply+
	2	M	24V power supply -
	3	⏚	ground
	4	--	--

Table 12-7 Definitions of dual RJ45 ports

Dual RJ45 network ports	No.	Signal	Signal definition
	1	TX+	Data sending+
	2	TX-	Data sending-
	3	RX+	Data receiving+
	4	--	--
	5	--	--
	6	RX-	Data receiving-
	7	--	--
	8	--	--
	Connector shell	PE	Chassis ground

DIP switch definition: Ethercat slave module address, dial to the top is ON, and the bottom is OFF.



DIP1	DIP2	DIP3	DIP4	DIP5	DIP6	DIP7	DIP8	address value
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	2
1	1	0	0	0	0	0	0	3
0	0	1	0	0	0	0	0	4
1	0	1	0	0	0	0	0	5
….								….
1	1	1	1	1	1	1	1	255

Remarks: When communicating with Omron CPU, the configured device node address needs to be consistent with the actual device address (DIP switch address) in order to successfully communicate

Intermediate Expansion Module

13

This chapter mainly introduces the rIntermediate Expansion module of h56-10 / h52-10 application system, as follows:

INT-00 is an important part of CTH300-H system. More modules can be extended by this module.

13.1 Specifications

Table 13-1 Basic attributes of intermediate expansion module INT-00

Name	Description	Order No.
Intermediate expansion module	Intermediate expansion module	CTH3 INT-000S1

13.1.1 General Characteristics

Table 13-2 General characteristics of INT-00

Physical characteristics	
Dimension(W×H×D)	34×115×101.6 mm
Power consumption	19.5W
Power characteristics	
Rated input voltage	24V DC
Input voltage range	20.4V~28.8V DC
Input Current	0.8A
Reverse polarity protection	YES
Bus power supply voltage	+5V DC
Bus power current	1.6A
LED indicator	
24V	ON: with 24VDC, OFF: without 24VDC
SF	ON: The module is faulty. OFF: No error

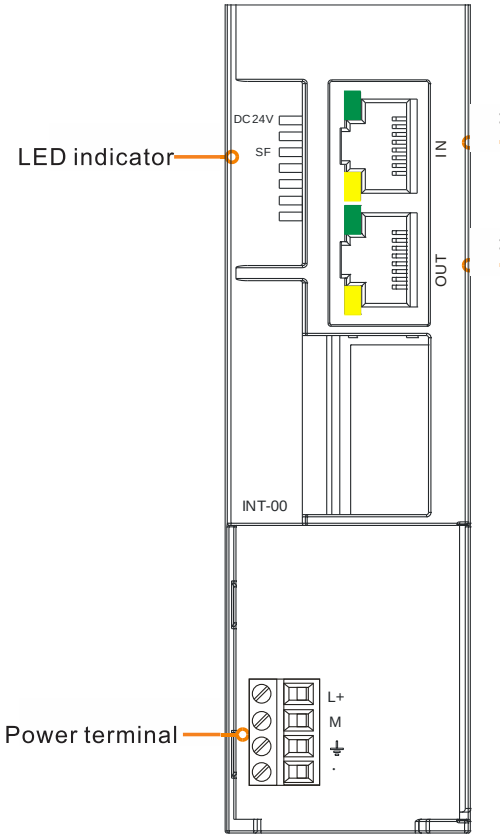
13.1.2 Functions and Features

Table 13-3 Functional characteristics of INT-00

Function category	Function item	Describe
Extensions	Extended bus interface	Provide bus expansion function
Communication function	Inter- extended Interface	Provide communication interface between Intermediate extension modules
Extensions	Extended function	Provides 8 I / O modules that allow expansion
Isolation function	Power isolation	Isolation between external power supply and system power supply
Protective function	Power protection	The power supply terminal provides reverse connection protection function and surge absorption function

13.2 Wiring

13.2.1 Interface Diagram



Follow the instructions to connect the relay module.

IN: the interface connected to the previous relay module (if it is the first Intermediate expansion module, this port is not connected).

OUT: the interface for connecting to the next relay module (if it is the last Intermediate expansion module, this port is not connected).

Note: The first relay module is connected to the CPU through the bus.

13.2.2 Interface Definition

Table 13-4 Definition of power interface of INT-00

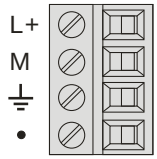
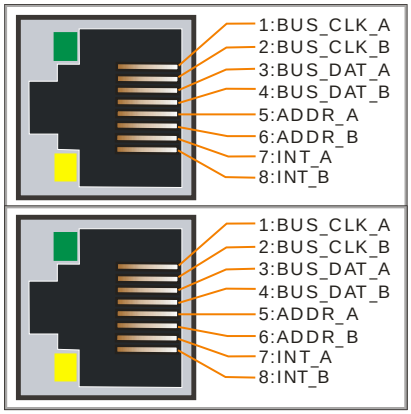
Removable Terminals	No.	Signal	Signal definition
	1	L+	24V power supply positive
	2	M	24V power supply negative
	3	⏚	the earth
	4	--	--

Table 12-5 Definition of dual RJ45 ports of INT-00



No.	Signal	Signal definition
1	BUS_CLK_A	bus clock
2	BUS_CLK_B	
3	BUS_DAT_A	bus data
4	BUS_DAT_B	
5	ADDR_A	configure address
6	ADDR_B	
7	INT_A	interrupt
8	INT_B	
connector housing	PE	chassis ground

13.2.3 Network Cables of Intermediate Expansion Module

The network cables used by CTH300 PLC relay modules are non-standard network cables. When making network cables, make sure that the signal cables in the network cables are combined in pairs, namely: white orange-orange, white green-green, white blue-blue, white brown- Brown, and the order between the four can be combined at will.

Profinet Slave Module

14

This chapter introduces the communication functions of H56-10/H52-10 motion controller as follows:

Profinet slave module CTH3 PNT-000S1 is an important part of CTH300 PLC system, each slave module allows to expand 8 CTH300 I/O modules (digital module, analog module, temperature module and HSC/HSP module), together with the Siemens Profinet master to form a network to realize the Profinet remote IO expansion communication function.

14.1 Technical Specifications

Table 14-1 Environmental standards and specifications

environmental conditions	
Temperature	Shipping and Storage : -40℃~+70℃ Horizontal installation : 0℃~60℃ vertical installation 0 : ℃~40℃
Relative humidity	10%~95%, non-condensing
Electromagnetic Compatibility - Immunity	
Electrostatic discharge IEC61000-4-2	Contact discharge: ±4kV Air discharge: ±8kV
Electrical fast transient burst IEC61000-4-4	Power line: 2kV, 5kHz Signal line: 2kV, 5kHz (I/O coupling clip) 1kV, 5kHz (communication coupling clip)
Surge IEC61000-4-5	Power cord: 2kv (asymmetric), 1kv (symmetric)
Radio frequency electromagnetic field radiation IEC61000-4-3	80MHz~1GHz, 10V/m, 80%AM (1kHz)
RF Field Induced Conducted Disturbance IEC61000-4-6	0.15MHz~80MHz, 10V/m, 80%AM (1kHz)
DC power input port for short time interruptions and voltage changes IEC61000-4-29	Short interruption: 10ms
High temperature operation IEC60068-2 Low temperature operation IEC60068-2	65℃, dwell time 24h, temperature change rate: ≤1℃/min, normal temperature recovery time: ≥1h -20℃, dwell time 24h, temperature change rate: ≤1℃/min, normal temperature recovery time: ≥1h
High temperature startup IEC60068-2 Low temperature start IEC60068-2	65℃, 2 hours, after the temperature is stable, turn it on 3 times discontinuously -20℃, 2 hours, after the temperature is stable, turn it on 3 times discontinuously
High and low temperature cycle operation IEC60068-2	-10℃~60℃ , dwell time 3h, temperature change rate 1℃/min, 5 cycles, normal temperature recovery time: 1h
High temperature	75℃, storage time: 72h, temperature change rate: ≤1℃/min, normal

storage IEC60068-2 Low temperature storage IEC60068-2	temperature recovery time: ≥1h -40℃, storage time: 72h, temperature change rate: ≤1℃/min, normal temperature recovery time: ≥1h
Thermal shock IEC60068-2	-40℃~75℃, dwell time 3h, temperature change time: 2min, 3 cycles
Constant high humidity IEC60068-2	Storage temperature: 40℃, storage time: 48h, storage humidity: 93%
Alternating Damp Heat IEC60068-2	25℃~55℃, humidity: 95%, single cycle: 24h, 2 cycles
Sinusoidal vibration (bare metal) IEC60068-2	5~150Hz, 0.05G2/Hz 150Hz~500Hz, -3dB/oct, 1h/axis, X, Y, Z total 3 axes
Mechanical shock (bare metal) IEC60068-2	15G, 11ms pulse, 3 times/direction
Fall	1m, 10 times, shipping package

Table14-2 General and functional characteristics of CTH3 PNT-000S1 module

physical properties	
Dimensions (W×H×D)	34×115×100mm
Power consumption	
Rated input voltage	24V DC
Input voltage range	20.4V~28.8V DC
Input current	0.8A
Reverse polarity protection	Have
Bus supply voltage	+5V DC
Bus supply current	1.6A
LED Indicator	
24V power indicator (green)	ON = 24VDC power supply is normal. OFF = no 24VDC power supply
SF indicator (red)	ON = Expansion I/O bus failure or PROFINET module failure OFF = no error
BF indicator (red)	ON = PROFINET bus communication failure (no switch connected, no network detected) Blinking = configuration configuration is inconsistent. OFF = no error
MT indicator (yellow) (maintenance indicator)	Reserve
RJ45 port indicator (green)	ON = there is a connection to the switch/PN master
	OFF = no connection to switch/PN master
RJ45 port indicator (yellow)	ON = there is data transceiving to the switch/PN master
	OFF = No data transmission/reception to switch/PN master
Expanded I/O	
Maximum number of supported modules per	8 (digital module, analog module, temperature module, HSP module, HSC module)

slave	
Protocol type	CTH300 PLC custom 55MHZ bus protocol
Maximum configuration of IO per slave	The maximum configuration of analog IO can reach 64AI/64AQ The maximum configuration of digital IO can reach 256DI/256DQ
PROFINET communication port	
Communication Interface	1 dual RJ45 port
Data transfer rate	Transmission rate for Ethernet services: 10 Mbps
	Transmission rate for PROFINET: 100 Mbps, full duplex
Supported Ethernet Services	ping, arp, network diagnostics (SNMP)/MIB-2, LLDP
Send cycle	250us~4ms
Third-party PN master	Support S7-300/400, SMART CPU, S7-1200/1500
The longest communication distance between slave stations	100m (100BASE-TX)
Topology	Support star, tree, line, ring topology
Isolation	Communication port isolation
Hardware configuration function	
File to import	PROFINET GSD file XML format
PN slave	The added CTH3 PNT-000S1 module supports 8-slot expansion
	Expansion modules can add digital modules, analog modules, temperature modules, HSP modules, HSC modules
Isolation and Protection	
Power isolation	Isolation between external power and system power
Interface isolation	PROFINET communication interface isolation
Power protection	The power supply end provides reverse polarity protection and surge absorption

14.2 Product structure

The structure of the CTH3 PNT-000S1 module is shown as follows:

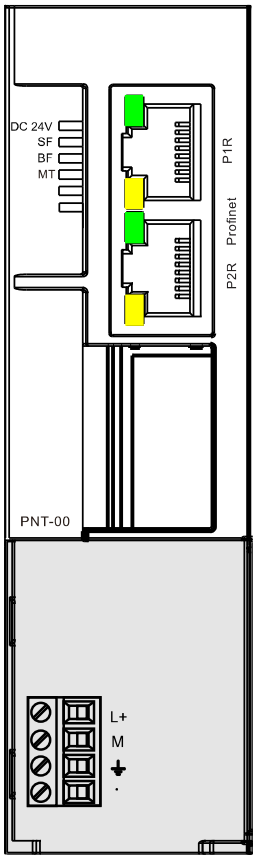


Table 14- 3 PROFINET communication port definition

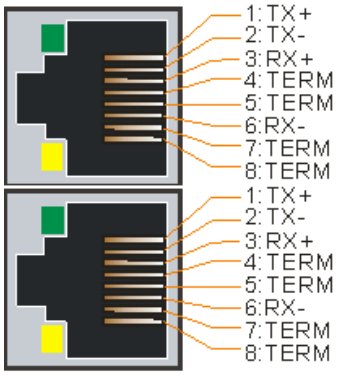
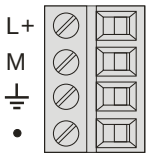
Dual RJ45 network ports (P1R/P2R)	NO.	Signal	Signal definition
	1	TX+	data sending +
	2	TX-	Data sending -
	3	RX+	Data receiving +
	4	--	--
	5	--	--
	6	RX-	Data receiving -
	7	--	--
	8	--	--
	connector housing	PE	Chassis ground

Table 14-4 definition of external power interface

Terminal	symbol	Signal definition
	L+	24V power supply +
	M	24V power supply -
	⏚	the earth
	--	--

Communication Application

15

This chapter introduces the communication functions of H56-10/H52-10 motion controller as follows:

15.1 Communication Application Example

H56/H52 motion controller supports multiple communication methods such as Bus, CANopen, EtherCAT, EtherNET, etc. This section will introduce the communication methods supported by H56/H52 with specific examples of H56.

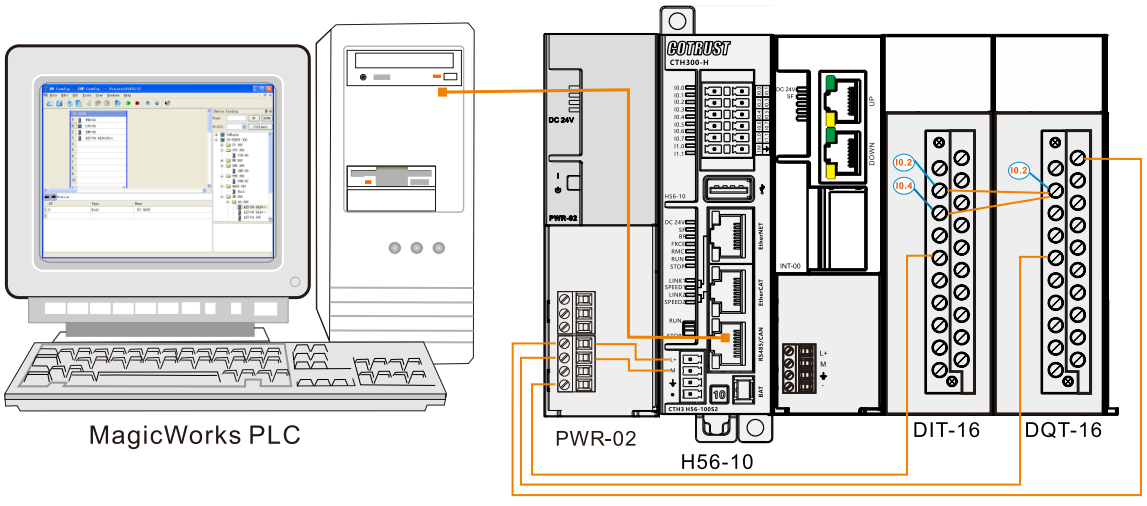
15.1.1 Bus Communication

1. Preparation before communication

Table 15-1 Components for Bus communication

Component	Function
Programming device PG\PC	MagicWorks PLC software (V2.19 or higher) is installed to configure, program and debug H56-10.
Assembly rail	H56-10 rack, used to fix the modules in the system.
Power module PWR-02	Power H56-10 motion controller and its 24 VDC load circuit.
H56-10 main control module	The CPU executes the user program, provides 5V voltage to the H56-10 backplane Bus, and performs PPI communication with other modules through the RS485 interface.
Expansion module	SM converts different process signal levels to match H56-10.
Standard network cable	This example uses digital input module DIT-16, digital output module DQT-16.

Bus communication connection:



Prerequisites

- A CTH300 system composed of power modules and H56-10. Correctly connect the programming device and H56-10 through the RS485 port.
- The MagicWorks PLC software (V2.19 or higher) has been correctly installed on the programming device.
- The connection device for serial data transmission has been prepared, and the necessary cables have been used for connection.
- The CPU has been correctly connected to the power supply module.

Realize the function

The output Q0.2 is lit, and the input I2.2 and I2.4 are lit at the same time.

2. Operation steps

Step 1: Install rails and modules

1) Install the modules in the order of the above network connection diagram

2) Installation and grounding of fixed rails

Use screws to install the fixed rail on the bottom plate, so that the upper and lower edges of the fixed rail leave a gap of at least 40mm. Then, connect the fixed rail to the protective conductor.

3) Install the module on the rail

Hook the power module to the rail, slide it to the grounding screw of the fixed rail, and then fix it with the screw. Hang the H56-10, digital modules DIT-16, and DQT-16 on the guide rail in turn, slide them until they touch the module on the left, then push it in firmly at the bottom, and finally fix all the modules with screws.

Step 2: Wiring

Open the front panel of H56-10, power module, digital input (DIT-16) and output module (DQT-16), and then refer to the communication connection in this chapter to wire them. The specific operations are as follows:

1) Use standard network cable to connect PC and H56-10

2) DIT-16 is connected to H56-10 via Bus

3) DIQ-16 is connected to DQT-16 via Bus

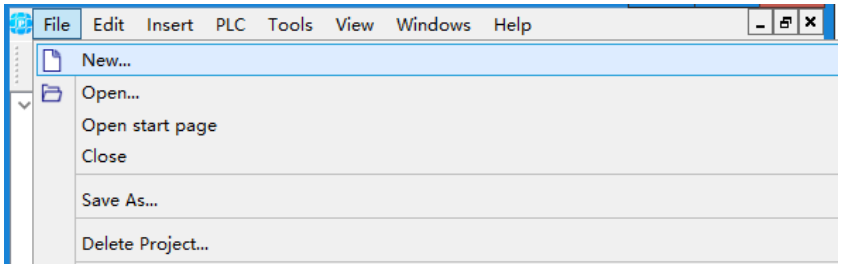
Warning

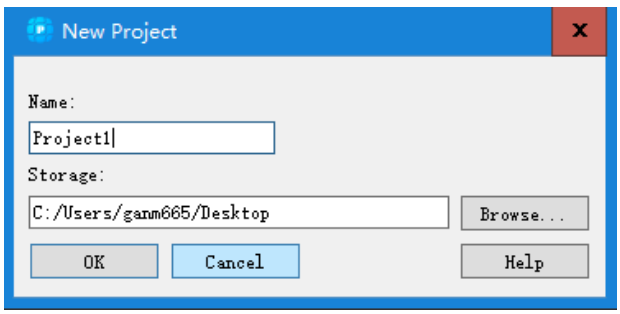
Do not touch the energized cable after the power module is powered on or after the power cable is connected to the main power supply. Any wiring work must be carried out under the condition of power failure!

Step 3: set up communication

1) Create a new project

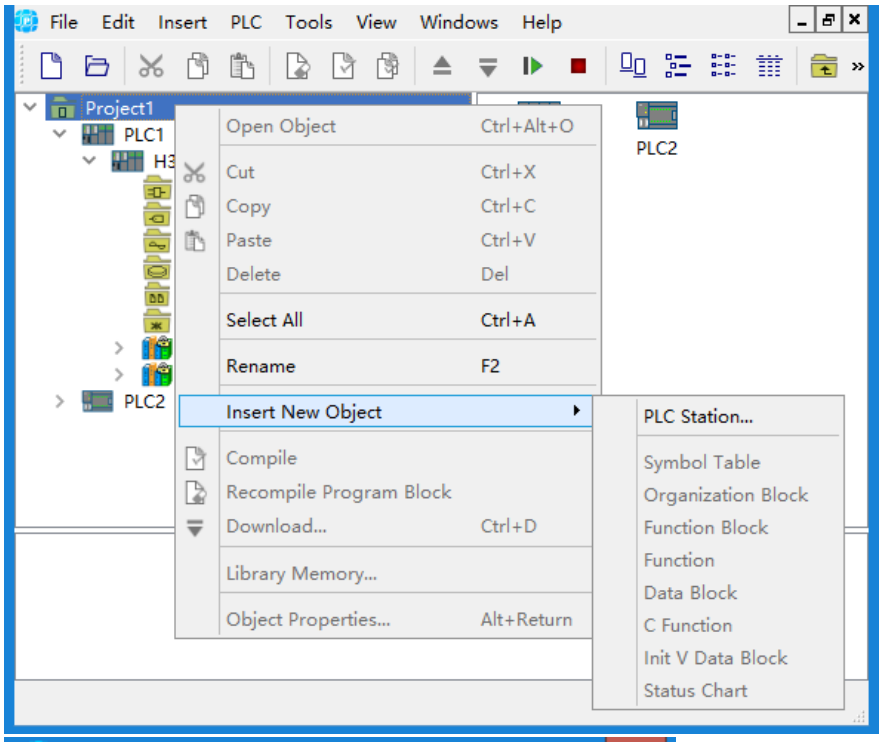
Open the MagicWorks PLC software, and then select the menu item "File" → "New" → enter the project name → click the "OK" button to complete the creation of the new project.

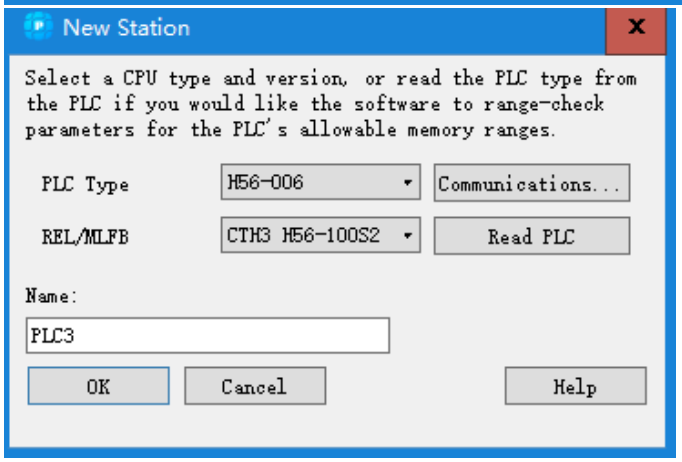




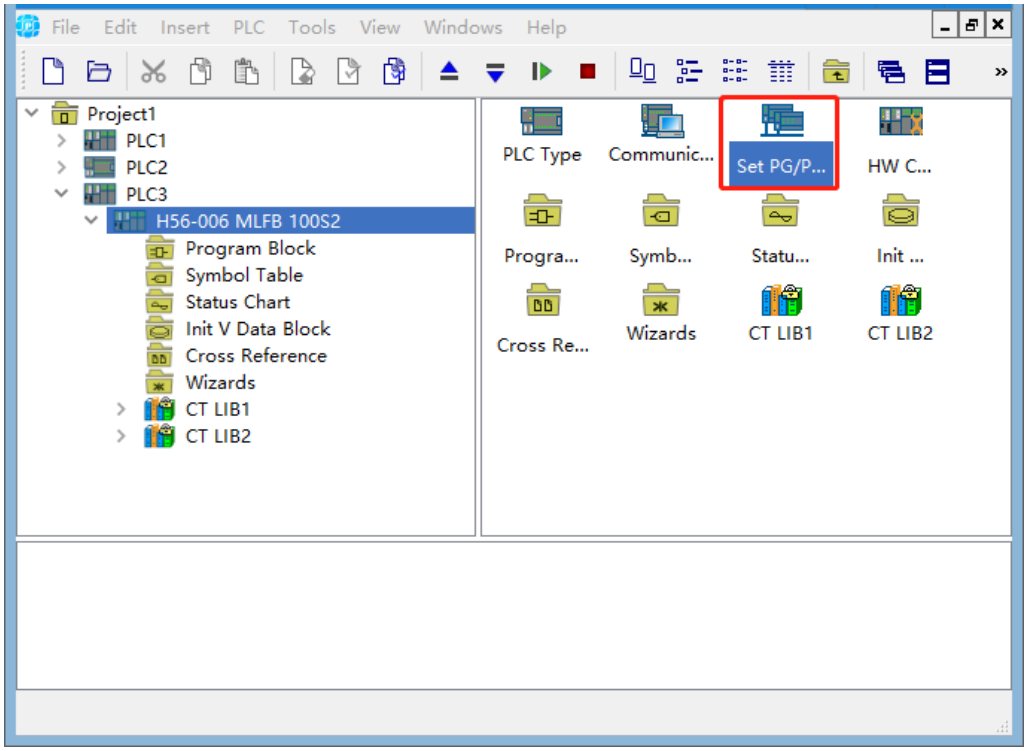
2) Insert the PLC station

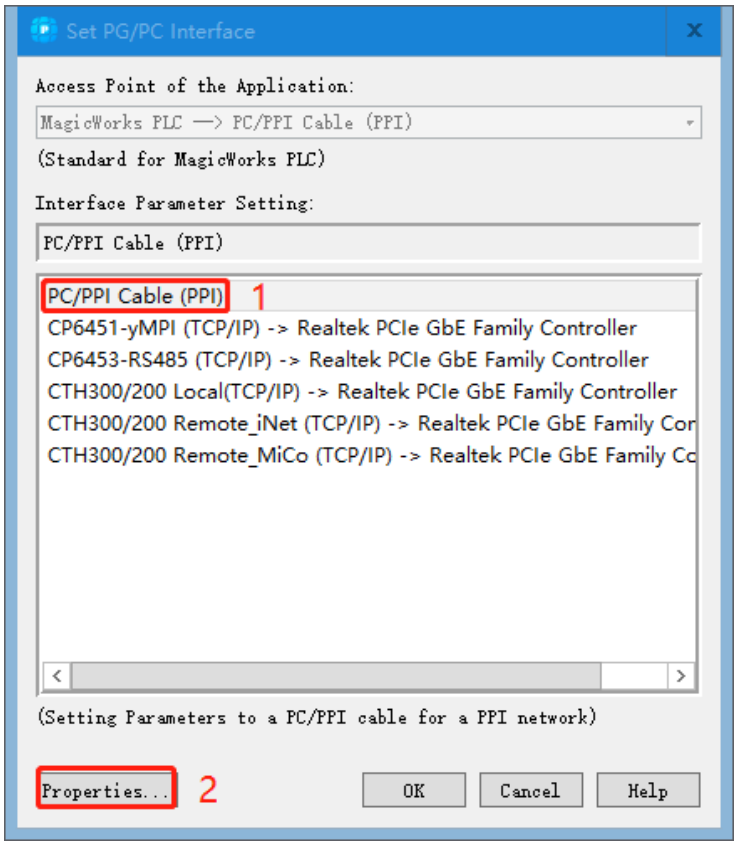
Right-click the “project” and select "Insert New Object" → "PLC Station" → "H56-006".



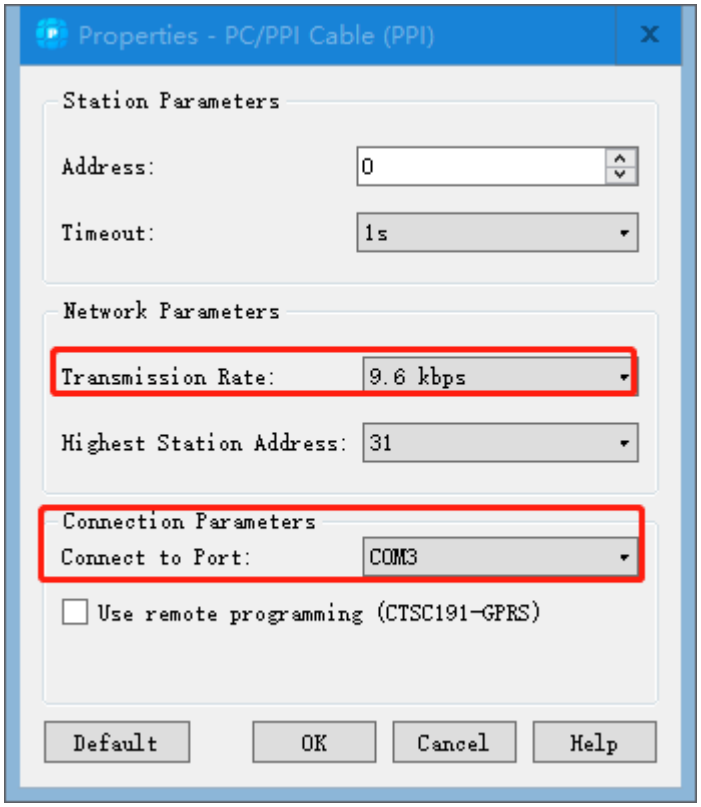


3) In the main interface of MagicWorks PLC, double-click "Set PG/PC Interface" or select the menu item "Tools" → "Set PG/PC Interface", then select "PC/PPI Cable(PPI)", and then click "Properties".

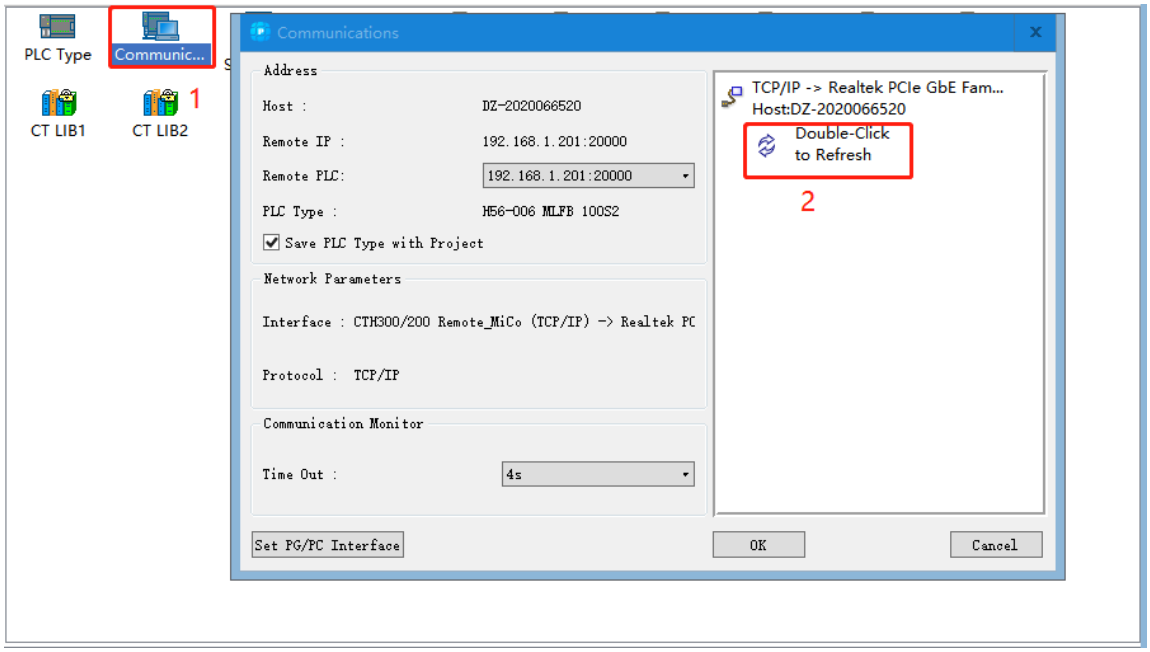




In the "Properties" configuration interface, configure the parameters in the above window according to the PPI port in the hardware configuration. After the configuration is completed, click "OK" to complete the configuration.



4) Click Communication , double-click the icon to search device, when the searched PLC is displayed in the communication dialog box, which means the search is successful.



Step 4: Hardware configuration

1) Enter the hardware configuration interface

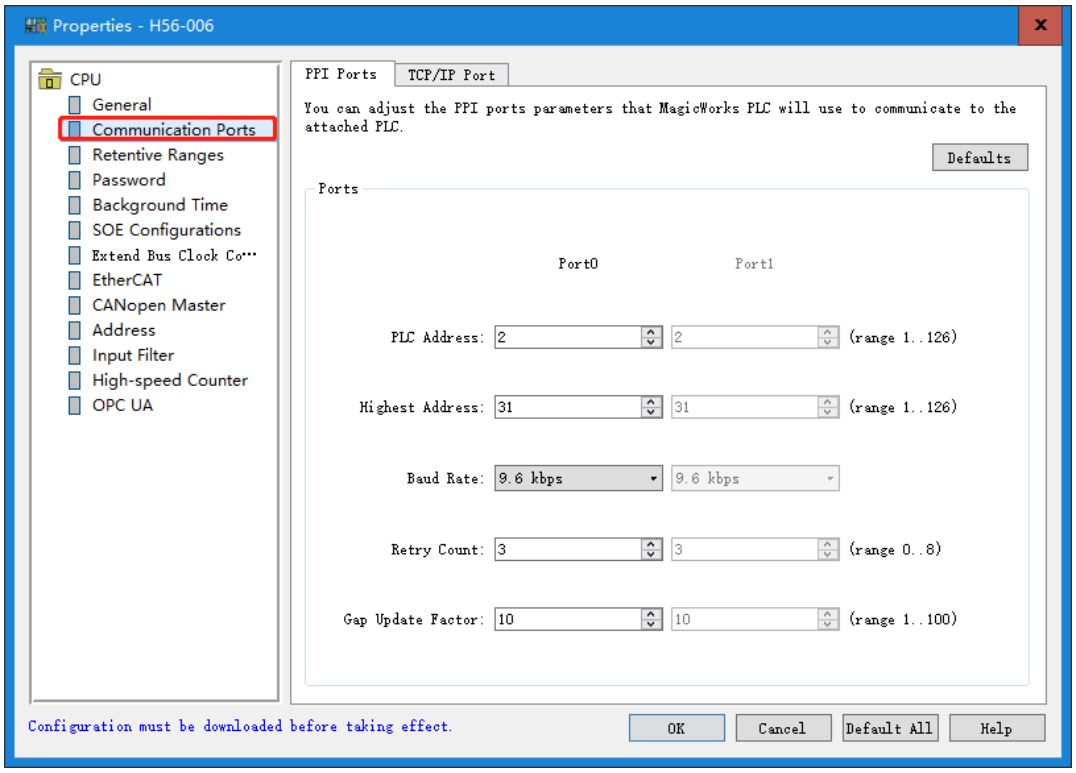
Select the H56-10 in the project tree of the project manager, and then double-click the hardware configuration icon to enter the hardware configuration interface.

2) Add power module

In the hardware configuration interface, select Rack 0, then click the CO-TRUST 300 node, select PWR-300 → PWR-02, and drag it to slot 0 of the Rack 0.

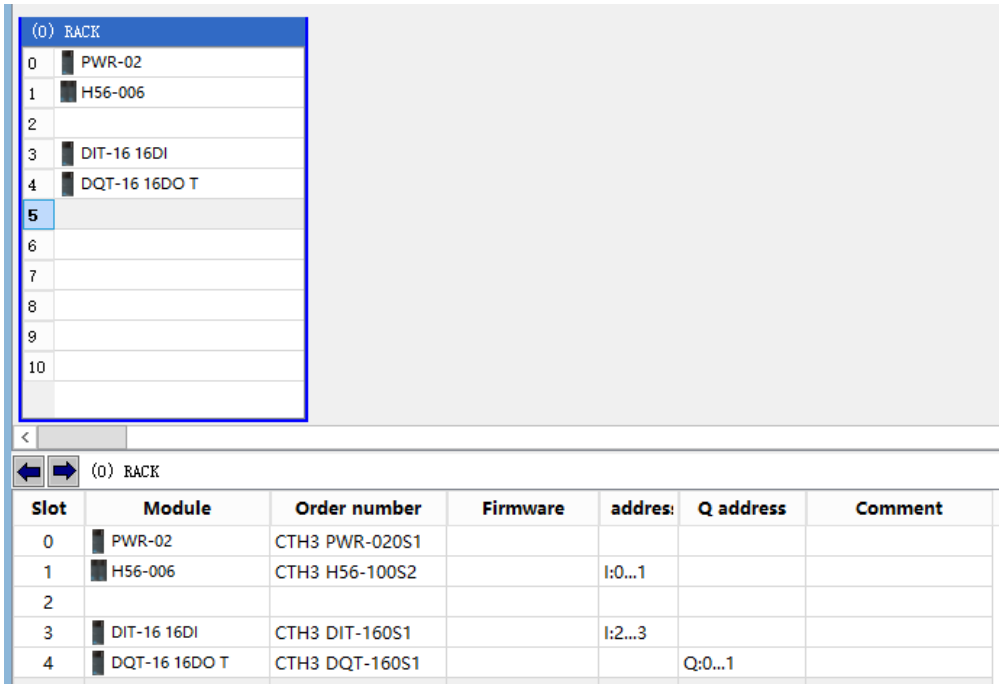
3) Add CPU and configure PPI communication for it

Expand CPU300→H56-10 from the device catalog and drag it to slot 1 of the rail. Then double-click H56-10 in slot 1, and the following window will pop up. Set the port on the "PPI Port" interface:



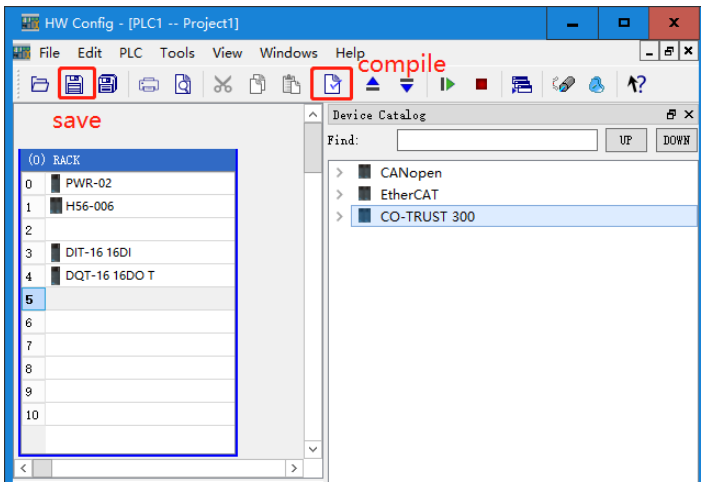
4) Add expansion module

Expand "SM 300" → "DI-300" from the hardware catalog and insert the DIT-16 into rack slot 3. Then unfold the DO-300 and insert the DQT-16 into slot 4 of the rack.



5) Save and compile the configuration

Select menu item "file" → "Save" to save the current configuration, and then select menu item "PLC" → "Compile" to compile the current project.

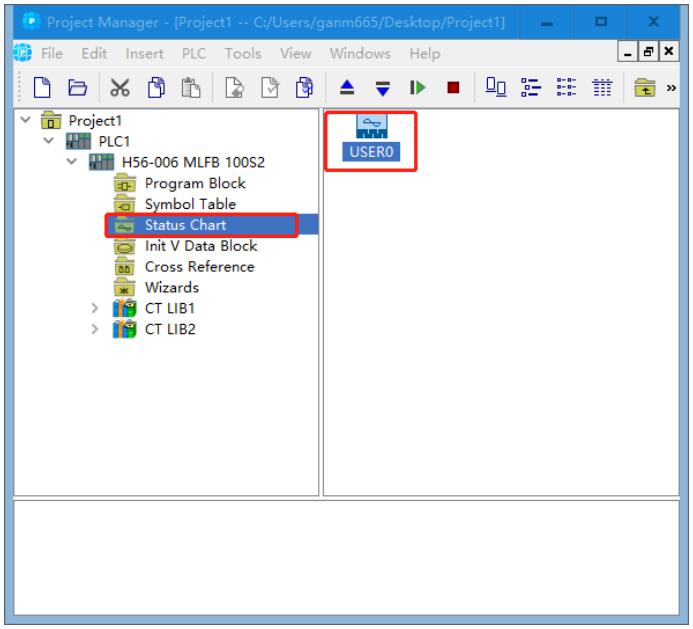


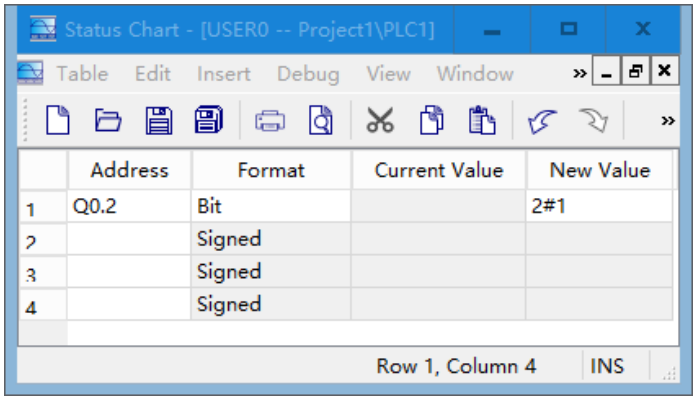
Step 5: Debug

1) Choose "PLC" → "Download" to download the program block and hardware configuration into H56-10. If H56-10 is in running mode, a dialog box will pop up on the download interface asking you to put H56-10 in STOP mode. Click "OK" to put H56-10 in STOP mode.

2) After downloading, turn the system operation switch of H56-10 to RUN mode.

3) Open the Status Char, fill in Q0.2 in the address of the Status Char, and write 1 in its new value, then Q0.2 of the digital module DQT will be lit. At the same time, the input points I2.2 and I2.4 are lit.





<Remarks> For the diagnosis method of H56-10 motion controller and expansion module, please refer to 5.8 CPU Fault Diagnosis

15.1.2 PPI/MPI Communication

This section will demonstrate the PPI/MPI communication of the H56-10 motion controller through an example.

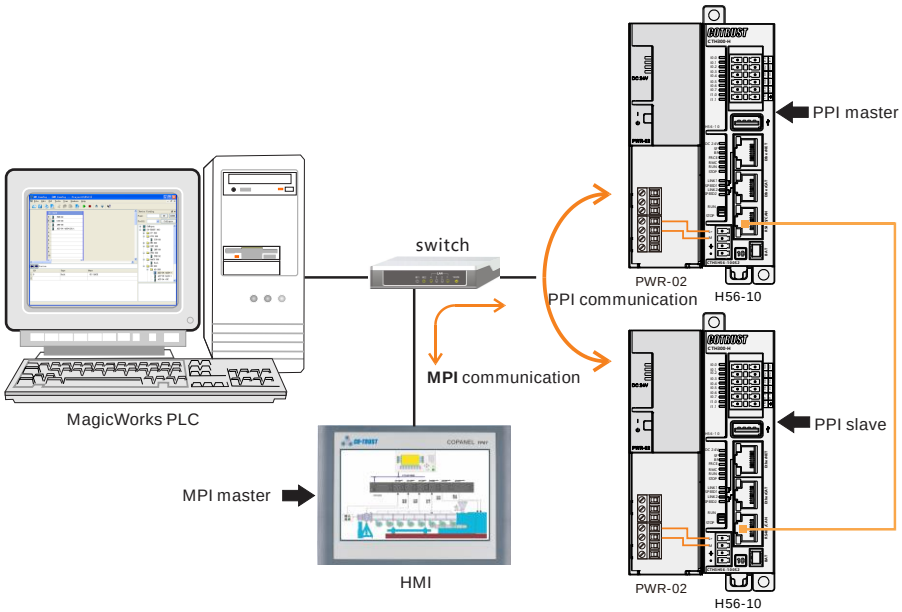
1. Preparation before communication

Table15-2 Components of PPI/MPI communication

Component	Function
Programming device PG\PC	MagicWorks PLC software (V2.19 or higher) is installed to configure, program and debug H56-10.
Power module PWR-02	Power H56-10 motion controller and its 24 VDC load circuit.
H56-10 main control module	This example requires two H56-10s, one as the PPI master and the other as the PPI slave. They communicate with each other through the RS485 interface.
Copanel HMI	As the MPI master, connect with H56-10 for MPI communication.
Standard network cable	Connect H56-10 to the programming device. Connect Copanel HMI to the PC to execute the download task of HMI. Connect H56-10 to Copanel HMI. <Remarks> Please refer to 5.3.3 Make Standard Network Cables to make a standard network cable.

◆ PPI communication: Two H56-10s in the network read and write data to each other through commands NETR/NETW.

◆ MPI communication: Copanel HMI is used as MPI master to monitor H56-10 (MPI slave).



2. Operation steps

Step 1: Connect to the PPI master (H56-10) and supply power to it.

1) Open the front panel of H56-10 and PWR-02 and connect cables to them.

2) Use standard network cables to connect each communication device as shown in the figure above.

Step 2: Program the PPI master (H56-10**)**

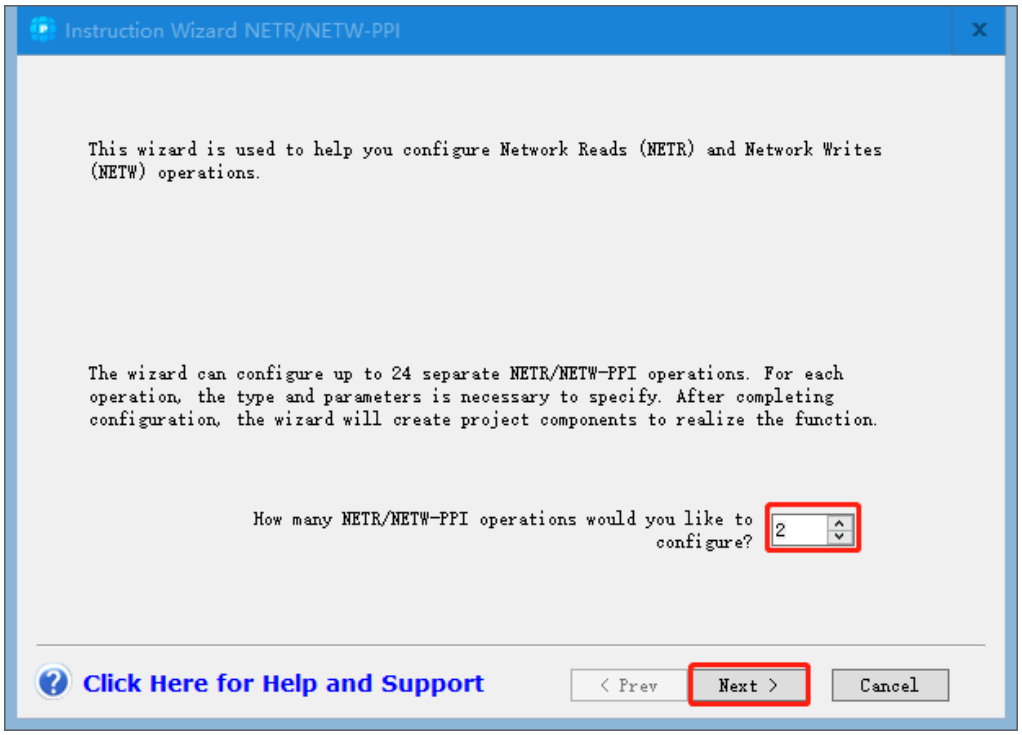
In this example, taking the NETR /NETE Instruction Wizard as an example to program the H56-10 of the PPI master and download the program to the PPI master (H56-10).

1) Create a new project in MagicWorks PLC, add H56-10 to the project, and then refer to 2.2 Set Up Communication to connect H56-10 with PC.

2) Select "Tools" → "NETR/W-PPI Instruction Wizard" or click the wizard node in the project tree, and double-click the NETR/W-PPI Wizard icon in the working area on the right to start the NETR/W-PPI instruction wizard. And then open this wizard or an existing wizard for configuration.

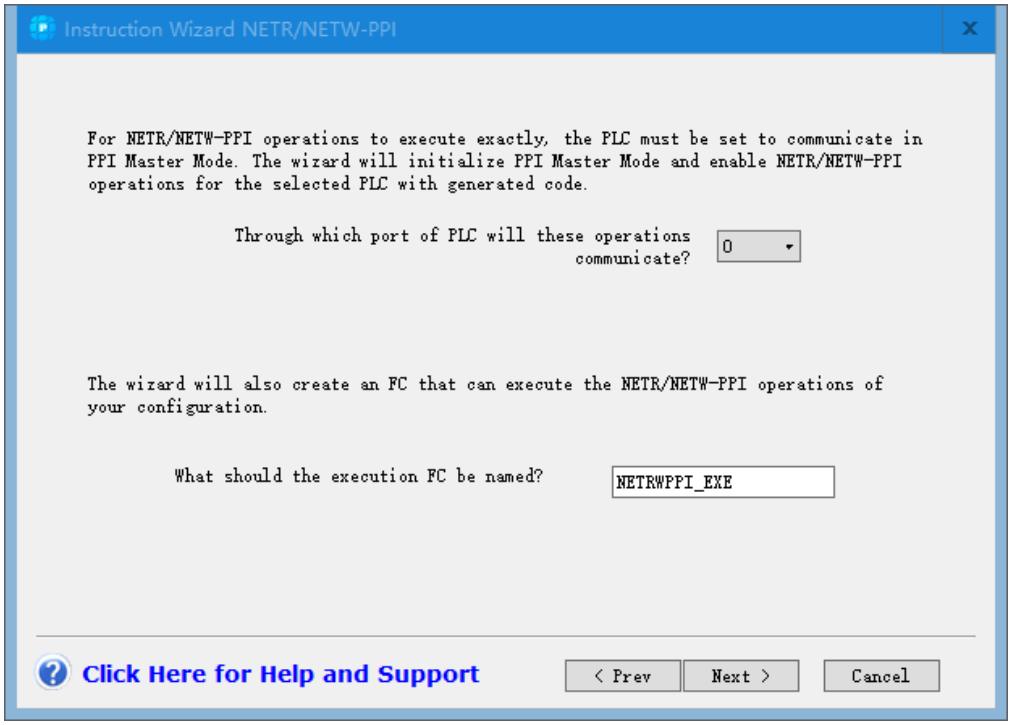
The steps of NETR/W-PPI wizard configuration are as follows:

<Step 1> Select the PPI communication port for network read and write operations, and specify the number of network operations to be configured:



In the dialog box "How many NETR/NETW - PPI operations do you like to configure?", set the network operands to be configured. The currently allowed operand range is 1~24.

<Step 2> Specify the communication port used and the generated function name:

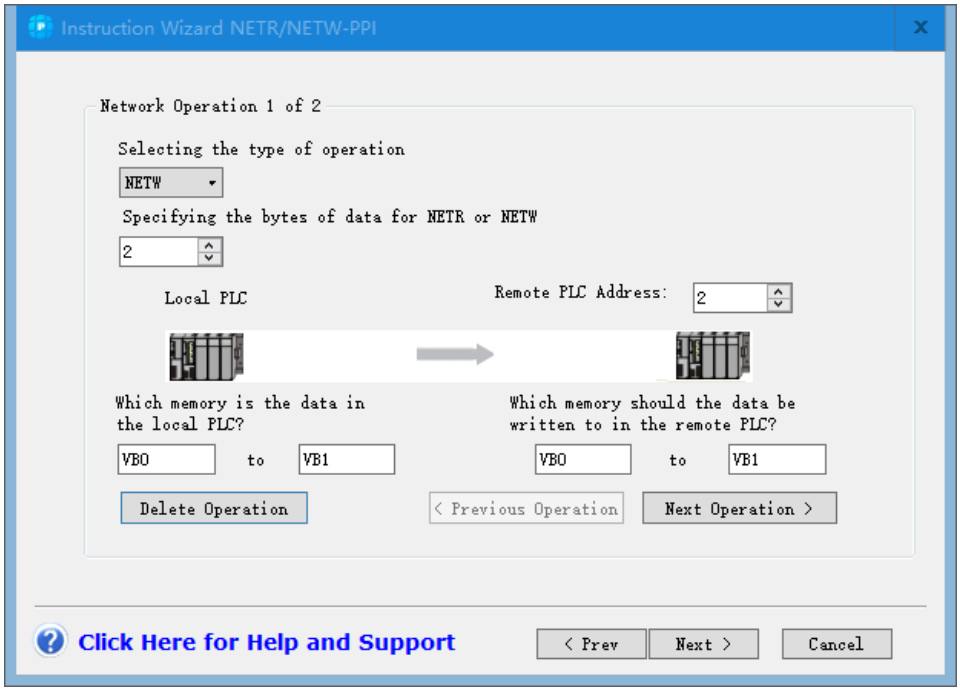


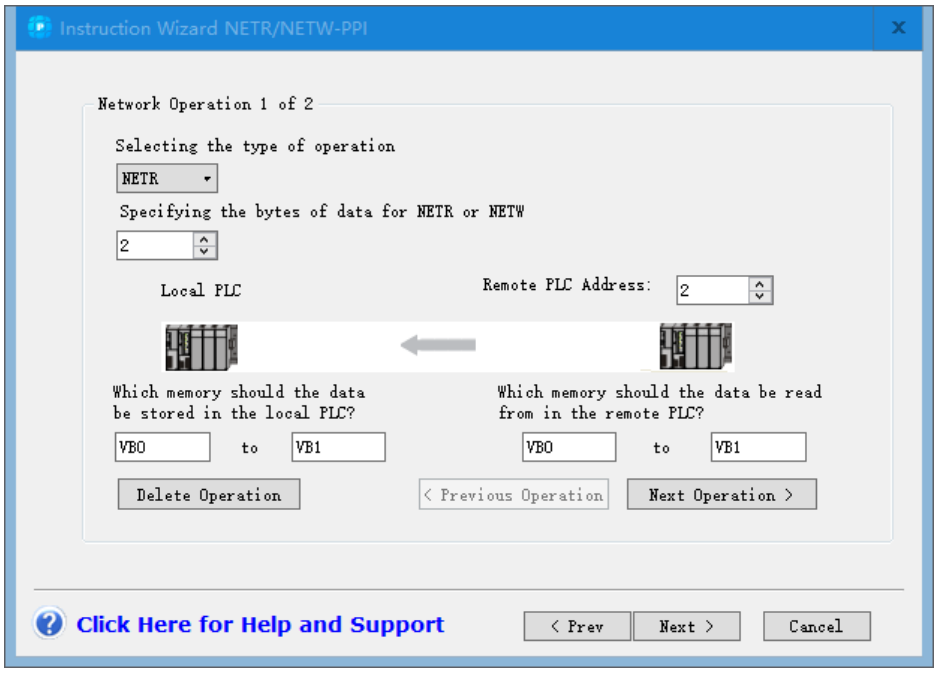
On this page you can change the following information:

➢ Which communication port (PPI communication port) is used by the PLC as the PPI master to communicate with the PPI slave.

➢ Edit the function symbol name generated by the wizard. The wizard will create a parameterized function under this name for performing specific network operations.

<Step 3> Configure network operation





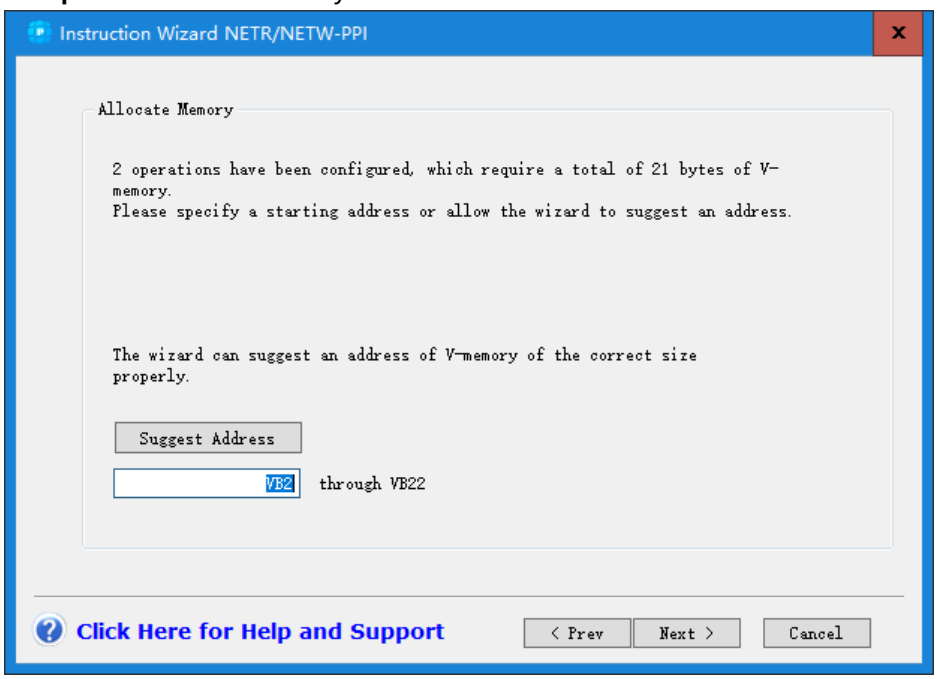
The information to be configured for network operation includes:

- Is this operation a network read or write.
- How many bytes are read or written from the slave.
- PLC slave address to be interacted.
- Local PLC address map (V memory only), that is, the location in the local PLC V memory where the data read back or data to be written is stored.

There are three buttons on this interface:

- Delete operation: Deletes the current operation. This operation reduces the number of configured operations by one.
- Previous Operation: Switch the interface to the previous operation.
- Next operation: switch the interface to the next operation.

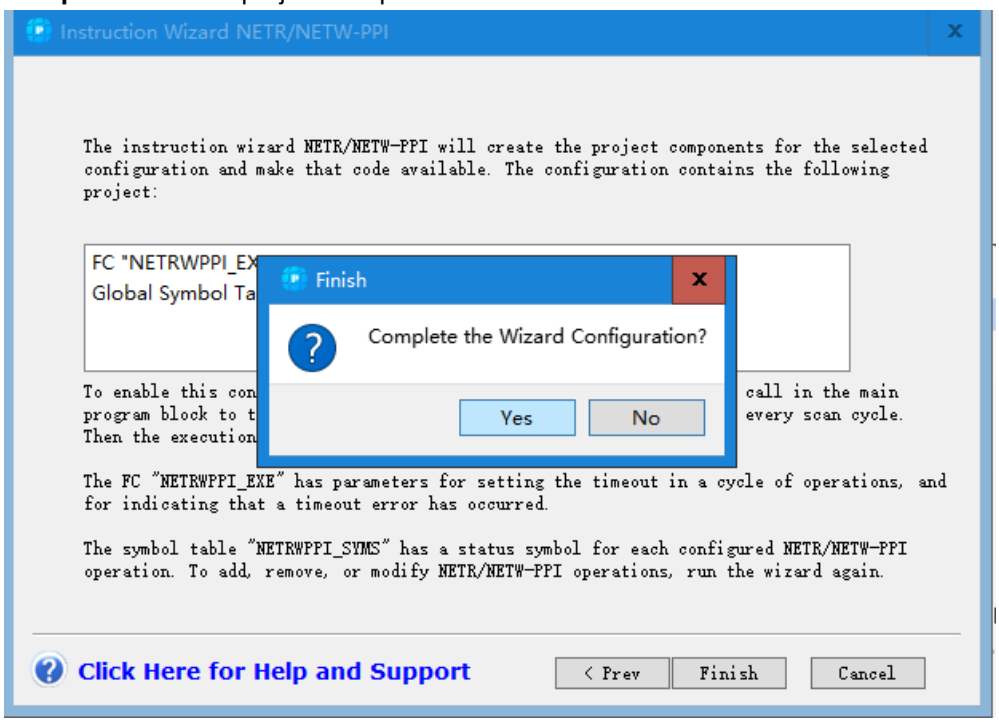
<Step 4> Allocate V memory area



For each network operation you configure, a 12-byte V storage area is required. On this page,

you can select the V memory address where you want to store the configuration block. If you want the wizard to suggest the correct size of the unused memory block address, click the "Suggest Address" button.

<Step 5> Generate project components

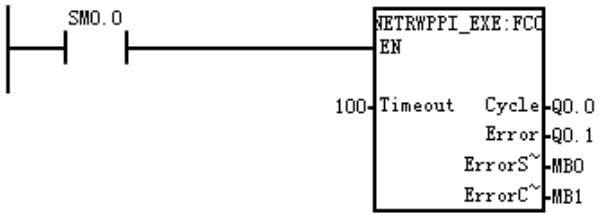


Click the "Finish" and click "Yes" on the pop-up interface to complete the NETR/W_PPI Instruction Wizard configuration. In the last step, the instruction wizard will generate related program components for the configuration you specify, including:

- Network read and write symbol table, containing the state symbol addresses for each network read and write operation for each operation in the configuration.
- The function NET_EXE contains the specific code of the operation. To enable network communication in the program, the execution function (NET_EXE) needs to be called in the main program block. The function is called using SM0.0 every scan cycle. It starts the configure network operation execution. Data processing functions established for each network operation are automatically invoked at the appropriate time.

3) Call the function block generated by the wizard in the program block

To perform network operations configured by the PPI wizard in your program, you need to call the function (NETRWPPI_EXE) with SM0.0 in the main program block. An example of calling this function in the programming software is shown in the following figure:



For the functions generated by the configuration in this description, the related PPI communication operations are shown in the following table.

Table 15-3 PPI communication operation instructions

Operate	Local controller data table location	Storage location of remote device operation status	Data length
Write	VB0-VB1 -> VB0-VB1	NETR1_Status:VB48	2 Bytes
Read	VB2–VB3 <- VB0-VB1	NETR2_Status:VB57	2 Bytes

When viewing the execution status of each configured operation through the status monitoring table, please refer to the following table.

Table 15-4 UDP network read and write operation status byte definition

Bit	Definition	Describe
1	D	Done (function complete) bit, 0: not completed, 1: completed.
2	A	Active bit, 1: The instruction is being executed. 0. not in execution.
3	E	Error status bit, 0: no error, 1: error.
4	n4	Reserved, always 0.
5~8	Error code	For detailed error code description, see the table below

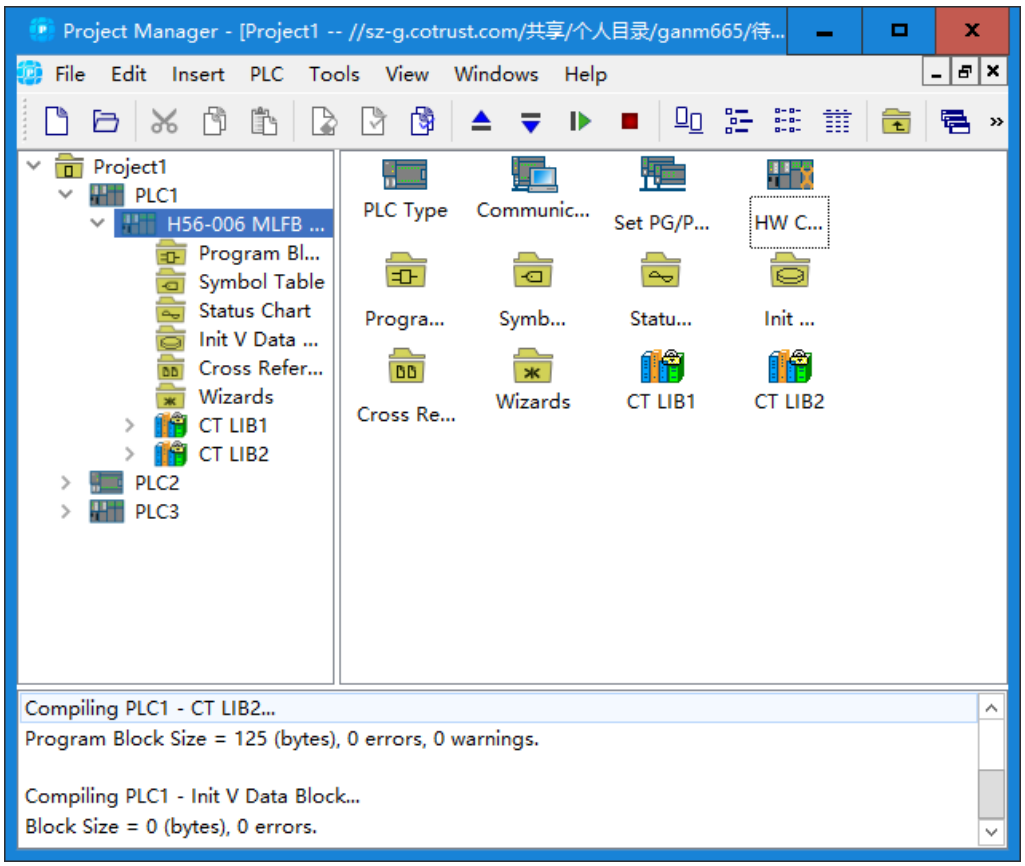
Table 15-5 Error codes

Error code	Definition
0	No error
1	Timeout error, the remote station does not respond.
2	Receive error, there is a check, frame or error in the response.
3	Offline error, duplicate station address or conflict caused by faulty hardware.
4	Team overflow error, more than 8 netw / netr instructions are activated.
5	Protocol violation, attempt NETW/NETR without enabling SMB30 and setting the port to PPI master mode.
6	Illegal parameter, the NETW/NETR table contains an illegal or invalid value.
7	No resources, the remote station is busy (uploading or downloading sequence).
8	Layer 7 error, application protocol violation
9	The information is wrong, the data address or data length is incorrect

<note> PPI communication adopts common network reading and writing. When reading and writing, a variety of registers can be configured, and the slave address is mapped to the direct mapping mode.

4) After programming, save the current program and compile it. Then click the download  on the toolbar or select "PLC" → "Download" to download the current program to the controller.

<Note> To ensure that the program blocks and function blocks are downloaded at the same time, be sure to return to the main interface of the project manager to perform the download operation.



Step 3: Write the value in the designated address of the PPI slave (H56-10)

Write a new value for the designated address of the PPI slave in the status table.

Step 4: Perform PPI master-slave communication

The function realized by PPI communication in this example: the PPI master reads the value in the specified address in the memory of the PPI slave.

After the system is powered on, turn the system operation switch of the PPI master to RUN. Then the RS485 communication port indicator lights of the PPI master and slave start to flash, which means that the PPI communication is proceeding normally.

Step 5: Perform MPI master-slave communication

1) Configure the project for Copanel HMI through the upper computer configuration software MagicWorks HMI.

First, establish a connection (communication protocol: CO-TRUST CTH 300) in MagicWorks HMI, and select the baud rate, protocol, and station address that match H56-10. Then configure a project and download the project to Copanel HMI.

2) After the system is powered on, Copanel HMI and H56-10 can perform MPI communication.

<Remarks> When the system fails, please refer to chapter 5.8 CPU Fault Diagnosis to obtain the diagnosis method of H56-10 motion controller and expansion modules.

15.1.3 ModBus RTU Communication

Through an example to guide you to build an application, through this example, you will be familiar with the ModBus RTU communication function of H56-10. For ModBus RTU communication, you need to use the CT_ModBus library file to configure the communication.

The library file includes 4 libraries, namely the master-slave library of PORT0 and the master-slave library of PORT1. This example uses the ModBus RTU library file of PORT0.

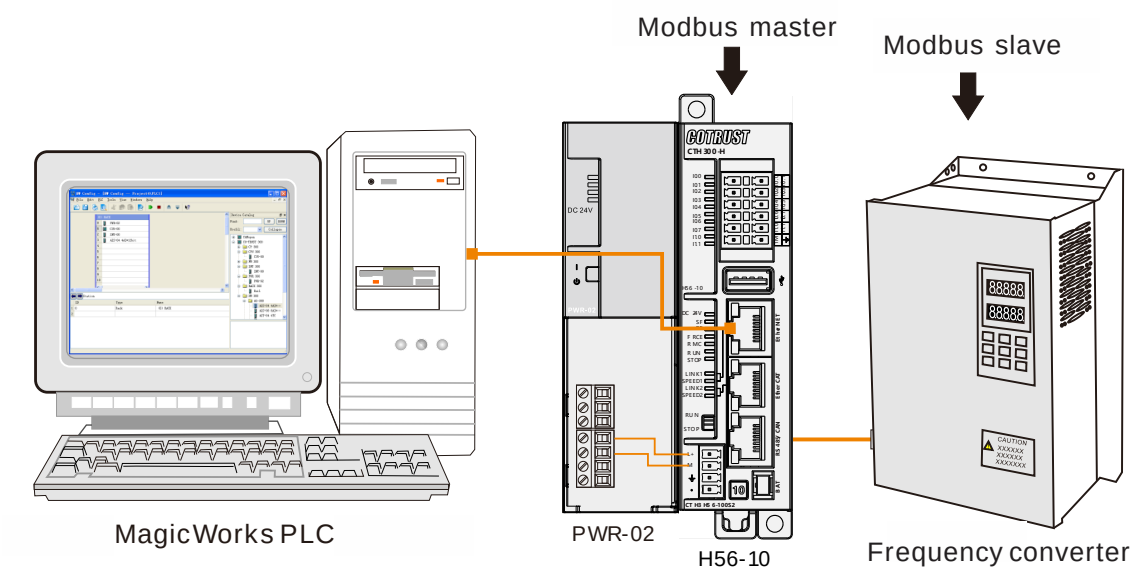
<Remarks> Please refer to Appendix C for the relevant instructions of the CT_ModBus library file.

1. Preparation before communication

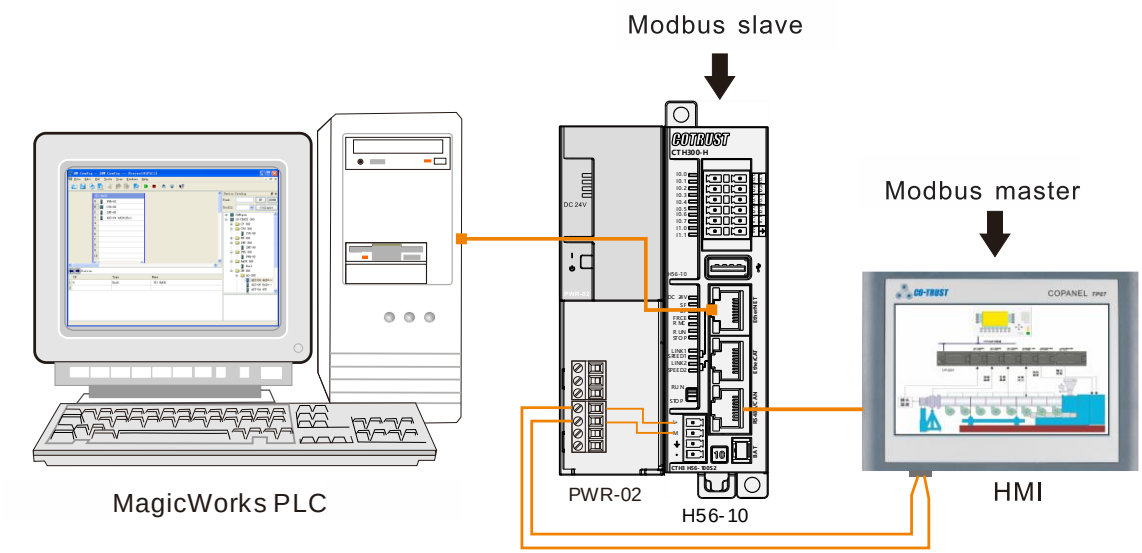
Table 15-6 ModBus RTU communication components

Component	Function
Programming device PG\PC	MagicWorks PLC software (V2.19 or higher) is installed to configure, program and debug H56-10.
Power module PWR-02	Power H56-10 motion controller and its 24 VDC load circuit.
H56-10 main control module	The CPU executes the program, provides 5V voltage to the CTH300 system Bus, and communicates with other modules through the Ethernet interface.
Copanel HMI	Copanel HMI, used as ModBus master to communicate with H56-10
Frequency converter	Inverter supporting ModBus protocol, as a ModBus slave to communicate with H56-10
Programming cable	Connect H56-10 to the programming device: PLC programming cable + programming adapter cable

◆H56-10 is used as ModBus RTU master to communicate with inverter.



◆ H56-10 is used as ModBus RTU slave to communicate with Copanel HMI.



2. Steps

Step 1: Wiring H56-10, power module, HMI and inverter

Refer to the above network connection diagram for H56-10, power module PWR-02, Copanel HMI and inverter wiring.

1) Use the programming cable to connect the PC and H56-10.

2) When H56-10 is used as ModBus RTU master to communicate with the inverter, use the communication line to connect the RS485 communication interface of H56-10 to the ModBus communication port of the inverter.

3) When H56-10 is used as ModBus RTU slave to communicate with Copanel HMI, use the communication line to connect the RS485 communication port of H56-10 to the ModBus communication port of HMI.

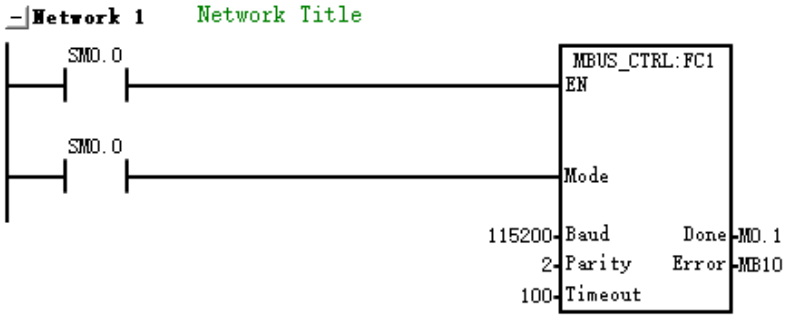
Step 2: Set up communication

Create a new project in MagicWorks PLC, add H56-10 to the project, and then refer to 2.2 Set Up Communication to connect H56-10 with PC.

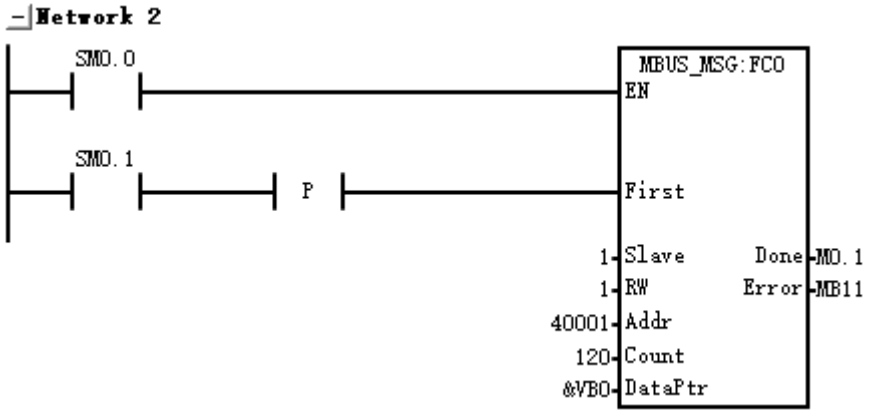
Step 3: When H56-10 as a ModBus master to communicate with the inverter

1) Programming for ModBus master (H56-10) is as follows:

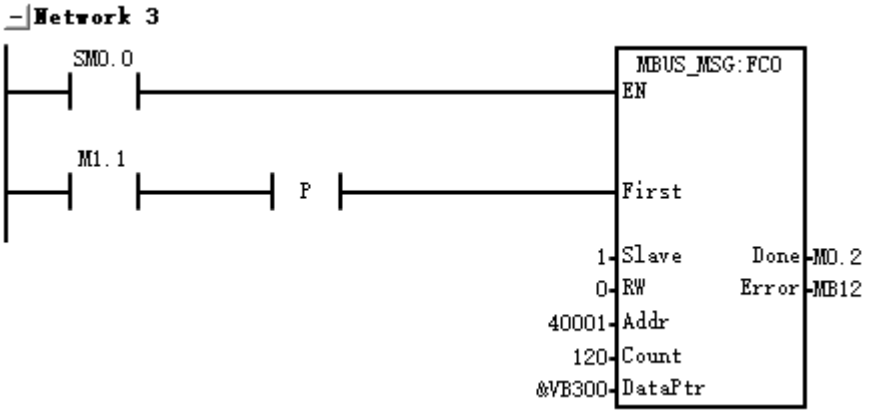
Master initialization: the baud rate is set to 115200bps, even parity, the timeout period is 100ms.



Write the 120 words starting from VB0 of the master to the address starting from 40001 of the slave (address 1).



Read the 120 bytes starting from the station (address 1) 40001 to the address starting from VB300.



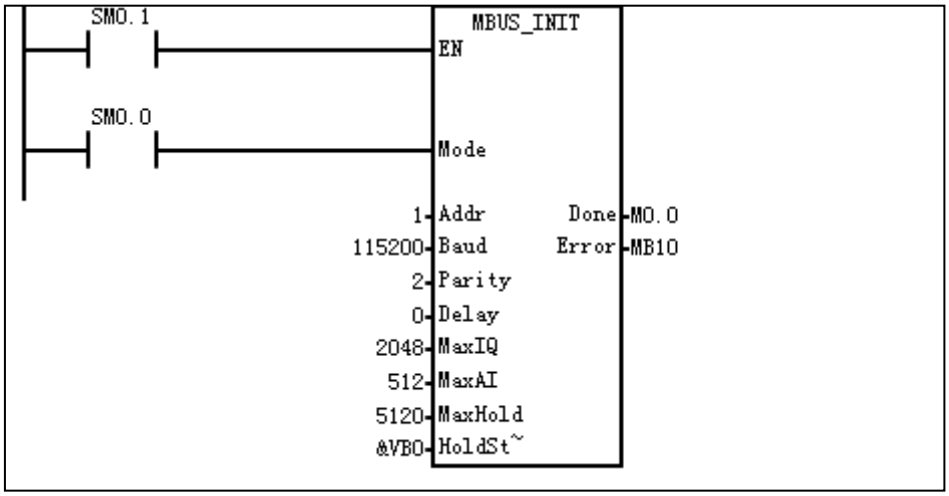
2) After programming, save the current program and compile it. Select "PLC" → "Download" to download the program blocks and hardware configuration from the programming device to the ModBus master (H56-10). If your H56-10 is in running mode, a dialog box will pop up on the download interface to prompt you to put H56-10 in STOP mode, click "OK" to put H56-10 in STOP mode.

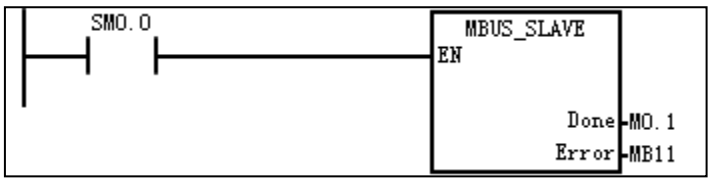
<Note> To ensure that the program blocks and function blocks are downloaded at the same time, be sure to return to the main interface of the project manager to perform the download operation.

3) The ModBus master (H56-10) starts ModBus RTU communication with the inverter.

Step 4: When H56-10 is used as ModBus slave to communicate with Copanel HMI

1) Program the ModBus slave (H56-10) as follows:





After programming, save the current program and compile it. Turn the system operation switch of the ModBus slave (H56-10) to STOP, then return to the MagicWorks PLC main interface, select the menu item "PLC" → "Download" to download the current program to the ModBus slave.
<Note> To ensure that the program blocks and function blocks are downloaded at the same time, be sure to return to the main interface of the project manager to perform the download operation.

3) ModBus slave (H56-10) starts ModBus RTU communication with Copanel HMI.

15.1.4 ModBus TCP Communication

This section will demonstrate the ModBus TCP communication function of the H56-10 motion controller through an example.

In ModBus TCP communication, when the CPU is slave, the communication is independent of the entire cycle. When the CPU is the master, its sending and receiving is controlled by the user program.

When using H56-10's EtherNET communication port for ModBus TCP communication, without any configuration, H56-10 can be used as a ModBus TCP slave to realize ModBus TCP communication. The default port number for MODBUS-TCP communication is 502.

When H56-10 is used as a ModBus TCP master to communicate with other slave, there are three ways to configure ModBus TCP communication, ModBus TCP single read/write instruction (ct_mBus_master_tcp_single), MBTCPSND instruction and ModBus TCP wizard.

Tips

Please refer to Appendix C.4 of this document to obtain the relevant instructions for the ModBus TCP single read and write command library file.
Download related manuals and library files from the official website of COTRUST, URL: http://www.co-trust.com/Download/index.html

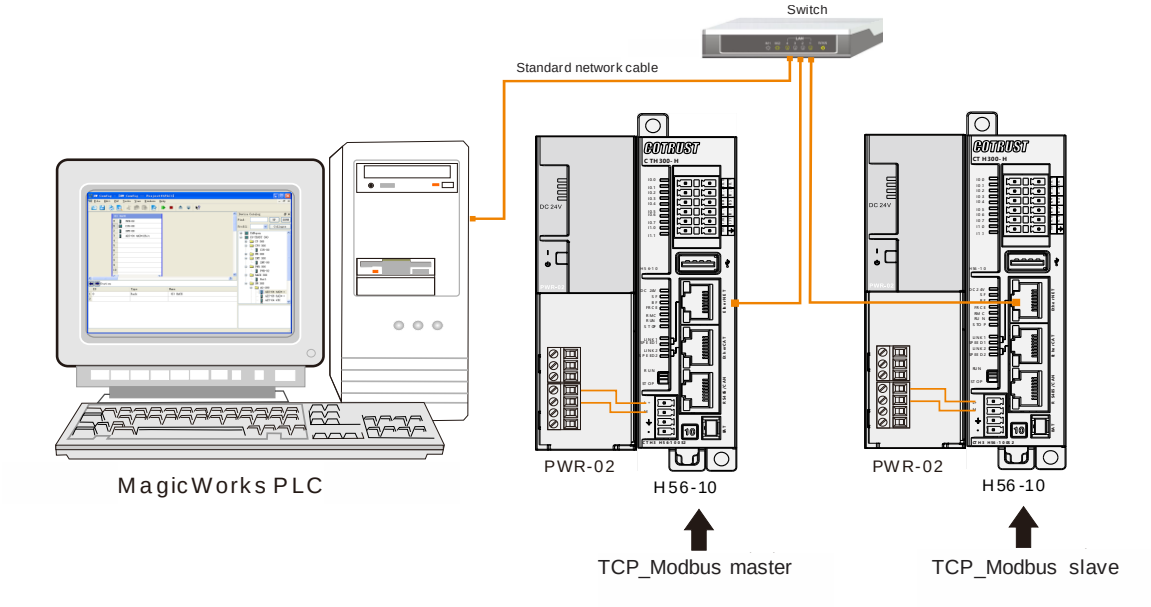
1. Preparation before communication

Table 15-7 ModBus TCP communication components

Component	Function
Programming device PG\PC	MagicWorks PLC software (V2.19 or higher) is installed to configure, program and debug H56-10.
Power module PWR-02	Power H56-10 motion controller and its 24 VDC load circuit.
H56-10 main control module	Connect the programming device with ModBus_TCP master and ModBus_TCP slave through standard network cables and switches, as shown in the figure below. Call CT_ModBus_TCP library programming, download the program to the ModBus master and monitor it. During ModBus TCP communication, the master will write the data in the specified address to the slave, and then read the data in the

	corresponding address through the slave.
Standard network	Connect H56-10 and programming device. Connect ModBus master (H56-10) and ModBus slave (H56-10).

Standard network cables and switches connect the programming device with ModBus_TCP master and ModBus_TCP slave, as shown in the figure below. Call the CT_ModBus_TCP library for programming, download the program to the ModBus master and monitor it. During ModBus TCP communication, the master writes the data at the specified address to the slave, and then reads the data in the corresponding address through the slave.



2. Steps

Step 1: Wiring the H56-10 and the power module

Open the front panel of the H56-10 and the power supply module PWR-02, and then wire them according to the above network connection diagram.

Step 2: Connect the device

Use standard network cables to connect programming equipment, switches and ModBus master and slave .

Step 3: Set up communication

Create a new project, add H56-10 to the project, and then refer to Chapter 2.2 Set Up Communicationto connect H56-10 with PC.

Step 4: Program the ModBus master (H56-10)

1) Set up ModBus TCP communication through the ModBus TCP library, add library instructions to the program blocks of MagicWorks PLC, the following instructions are the "MBTCPS_EXE" library instructions in the "Ct_MBus_master_tcp_single" library file:

The ModBus_TCP master writes 120 data (starting from VB0) into the ModBus_TCP slave (IP: 192.168.1.100), in a section of memory with the starting address 40001

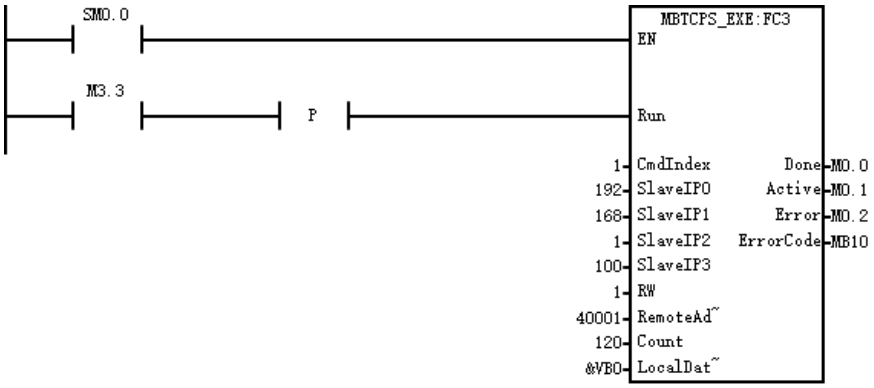


Table 15-8 "MBTCPS_EXE" instruction definition

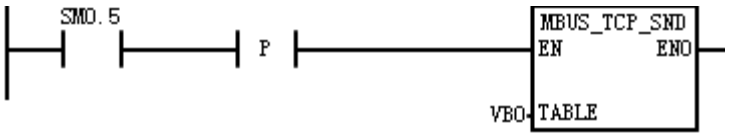
Parameter	Input/ output	Data type	Describe
EN	IN	BOOL	Enable
RUN	IN	BOOL	Communication enable bit, please use edge trigger
CmdIndex	IN	BYTE	The serial number of MBTCPS_EXE must be unique. The serial number ranges from 1 to 255.
SlaveIP0	IN	BYTE	The first byte of the slave IP
SlaveIP1	IN	BYTE	The second byte of the slave IP
SlaveIP2	IN	BYTE	The third byte of the slave IP
SlaveIP3	IN	BYTE	The fourth byte of the slave IP
RW	IN	BYTE	Read and write flag, 0: read. 1: write
RemoteAddr	IN	DWORD	Remote data address (such as 40001)
Count	IN	WORD	Number of elements (1~120 words or 1~1920 bits)
LocalDataPtr	IN	WORD	Local data pointer (such as &VB0)
Done	OUT	BOOL	Completion flag (0: not completed. 1: completed)
Active	OUT	BOOL	Active bit (0: activated. 1: not activated)
Error	OUT	BYTE	Error bit, 0: no error, 1: error
Errorcode	OUT	BYTE	Error code, valid when the completion bit is 1

Table 15-9 Definition of error codes at the output

Error code	Describe
0	No error
1	The number of connections reaches the maximum
2	Establishing connection
3	Receiving timeout, no response from slave
4	The requested parameter is wrong
5	Command is not enabled
6	The connection is processing other requests (such as activating multiple MBTCPS_EXE at the same time)

2) Call ModBus_TCP network read and write instructions for ModBus TCP communication settings. The library instructions are defined as follows:

Ethernet MODBUS network read and write instructions MBUS_TCP_SND
Instruction list: MBUS_TCP_SND EN, TABLE, ENO
LAD:



Parameter	Data type	Describe
EN		Enable bit of MBUS_TCP_SND
TABLE	BYTE	Including status bit, IP address, port number, read/write type, pointer, data length, data byte. For specific description, please refer to the TABLE parameter description in the following table.

Table 15-10 TABLE parameter description

Byte offset 7 0

D	A	E	n4	error code
The first byte of the remote station IP				
The second byte of the remote station IP				
The 3rd byte of the remote station IP				
The 4th byte of the remote station IP				
High byte of port number				
Low byte of port number				
Types of message reading and writing 0: Read 1: Write				
The first byte of the read and write address (high byte)				
The second byte of the read and write address				
The 3rd byte of the read and write address				
The 4th byte of the read and write address (low byte)				
Reserve				
Unit ID				
Data length high byte				
Low byte of data length				
Data byte 0				
...				
Data byte 239				

D: Read and write done bit, 0=not completed, 1=completed

A: Active bit, 1=instruction is being executed, 0=instruction is not being executed

E: Error status bit, 0=no error. 1=error

n4: reserved, always 0

Table 15-11 TABLE parameter error codes (error codes are valid only when the done bit is 1)

Error code	Describe
0	No error
1	Response check error
2	Unused

3	Receiving timeout (no response from slave)
4	Request parameter error(slave address,ModBus address,count,RW)
5	ModBus/Freeport is not enabled
6	ModBus is busy with other requests
7	Response error (response is not the requested operation)
8	Response to CRC checksum error
101	The slave does not support the requested function
102	The Slave does not support data address
103	The slave does not support this data type
104	The Slave equipment failure
105	The slave received the message, but the response was delayed
106	The slave is busy and rejected the message
107	The slave rejected the message
108	The Slave memory parity error

Notes:

• There can be multiple slaves to read and write messages at the same time, but each slave can only have one message to read and write at the same time.

• MBUS_ TCP_SND Done opens after receiving the response or when there is an error.

•TCP MODBUS supports up to 32 connections, and each message can read and write up to 240 bytes at a time.

Application examples of MODBUS_TCP network read and write instructions:

The rising edge of SM0.5 will write 120 words starting from VB16 to the remote IP (192.168.1.202). The port number is 502, and the CPU memory starts from VWO.

```
LD        SM0.1
MOVB      192, VB1
MOVB      168, VB2
MOVB      1, VB3
MOVB      202, VB4
MOVW      502, VW5
MOVB      1, VB7
MOVD      40001, VD8
MOVB      1, VB12
MOVW      120, VW14
```

```
LD        SM0.5
EU
MBTCPSND  VB0
```

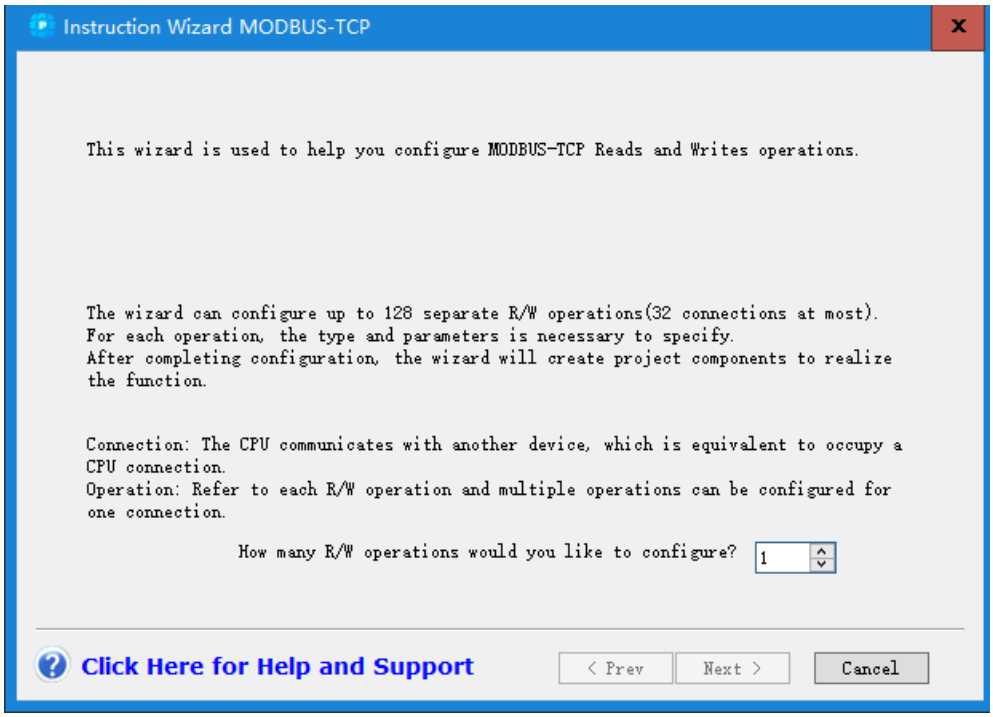
3) Program the ModBus_TCP master through the ModBus TCP command wizard

Select "Tools" → "MODBUS-TCP Wizard" or select the wizard node in the project tree and double-click the MODBUS-TCP wizard icon in the right working window to start the MODBUS-TCP wizard for wizard configuration.

When the wizard configuration is completed, a FC will be created in your project, which contains the information and address you specified in the MODBUS-TCP wizard. The parameters and information for MODBUS TCP communication are stored in the V storage area. If you want to view the configured parameter blocks and information after completion, you can open the wizard editor and select the corresponding wizard to check.

The operation steps are as follows:

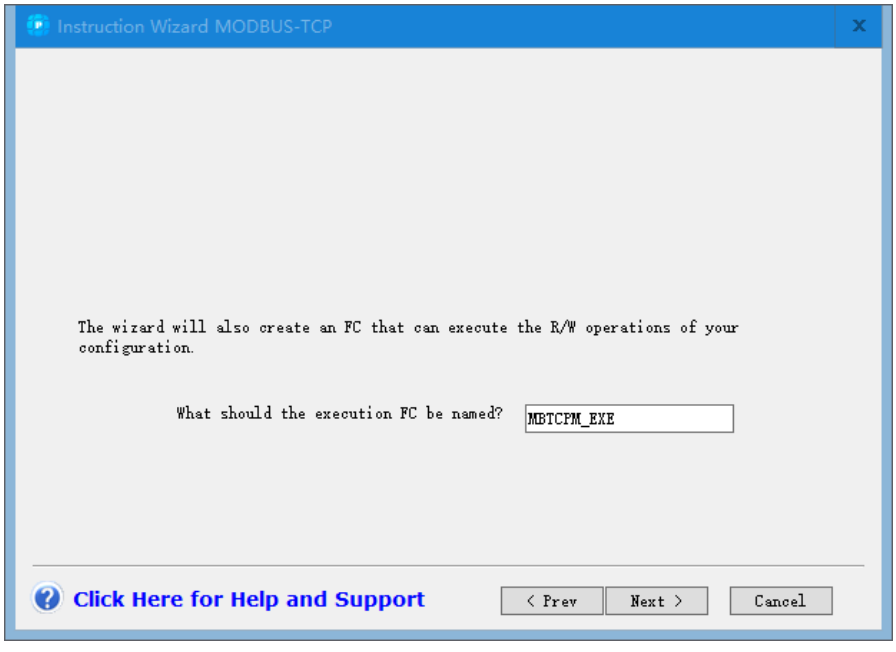
<Step 1> Specify the number of network operations to be configured.



In the dialog box "How many MODBUS-TCP read/write operations do you want to configure?", set the number of network operations to be configured. Currently, the allowed number of operations is 1~32.

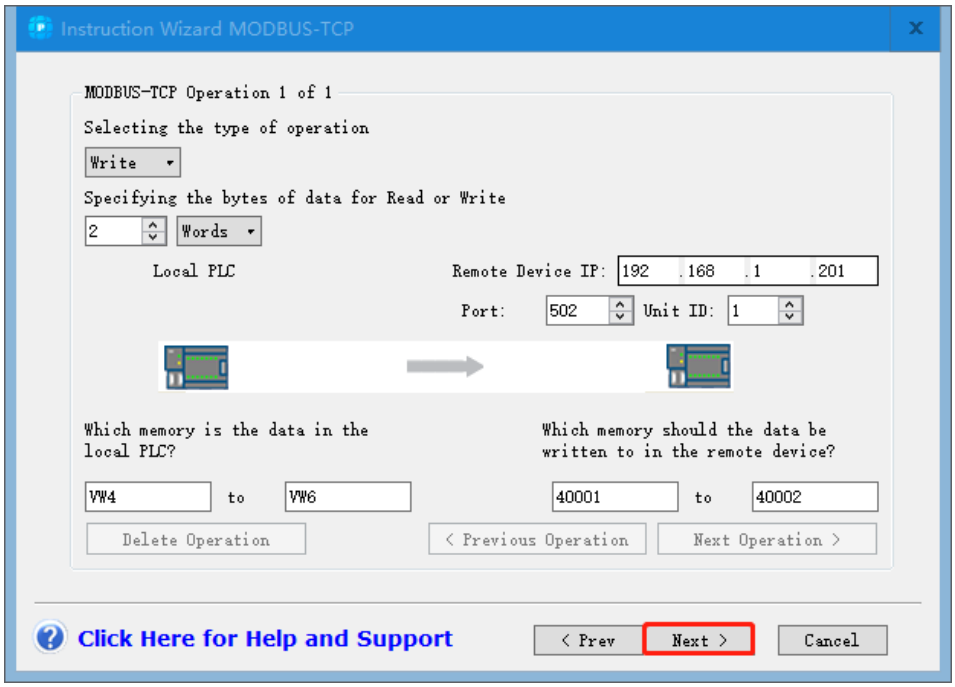
If the project contains the configured MODBUS-TCP command wizard, there will be a selection box "Remove this MODBUS-TCP configuration" to delete the old configuration on this page, you can choose to delete the existing configuration or click Next to edit it. If editing the current configuration, all previous configuration parameters will be loaded automatically when the configuration page is opened.

<Step 2> Specify the name of the generated function



On this page you can change the following information:
Edit the wizard-generated function symbol name. After the configuration is complete, the wizard creates a parameterized function under this name for performing specific network operations.

<Step 3> Configure network operation



The information to be configured for network operation including:

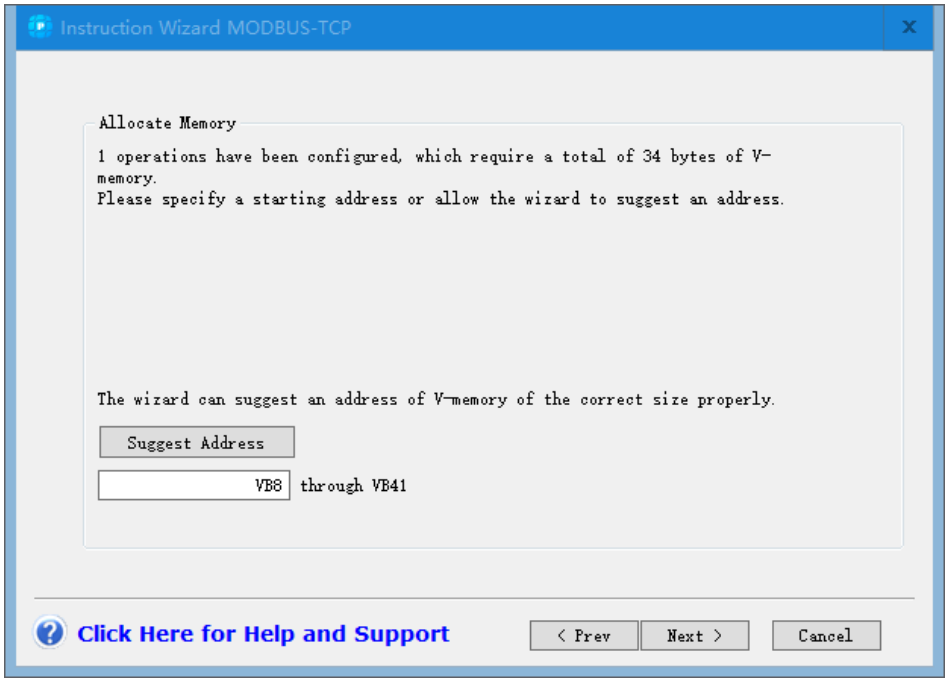
- Operation types: Read or Write
- Address, Port No.(When the slave is a controller, this value defaults to 502) and Unit ID for PLC slave which used for remote communication.
- Local PLC address map to V memory, that is, the location for reading/writing data store in local PLC V memory.
- Remote PLC address map to V memory, that is, the location for reading/ writing data store in remote Vmemory.

Buttons on the interface:

- Delete operation: Delete the current operation, which resulting the configured operations decreased by one.
- Previous operation: switch the interface to previous read/write operation.
- Next operation: switch the interface to next read/write operation.

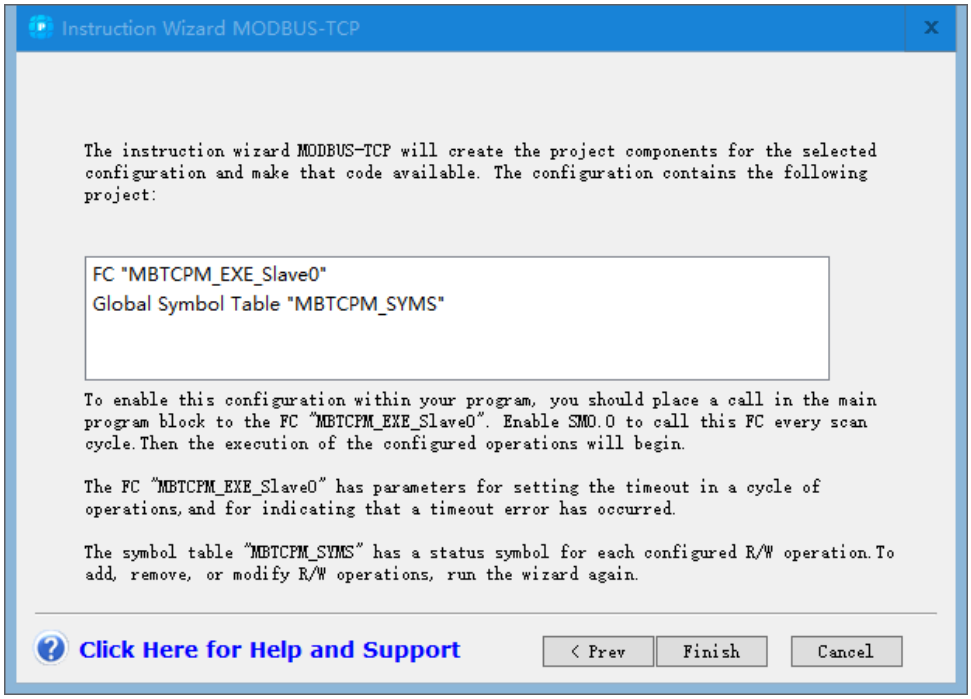
If you need to configure multiple operations, click "Next Operation" on the configured operation interface. After configuring the last operation, click "Next".

<Step 4> Allocating V Memory



For configuring each network operation, a V storage area of 34 bytes is required. You can choose the V memory address where you want to store the configuration block. If you want the wizard to suggest the correct size of the unused memory block address, click the "Suggest Address".

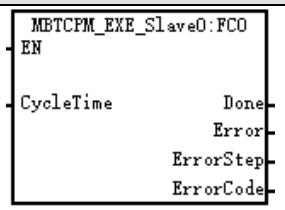
<Step 5> Generating project components



The last step for this wizard would create related program components, including:

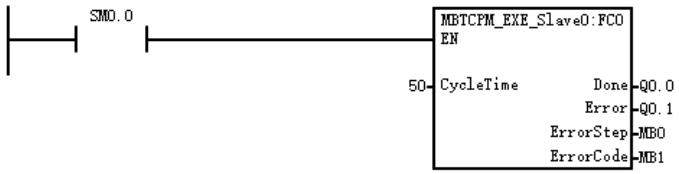
- Network R/W symbol table, which contain the status symbol address for each network R/W operation in the configuration.
- Function MBTCPM_EXE contains the specific operation code, it needs to be called in the main block to enable the network communication in the program. Using SM0.0 to call this function during each scanning cycle to initiate the implementation of the configured network operations. The data handling for each network operation would be called when necessary.

Table 14-12 MBTCPM_EXE command description

ModBus_TCP communication command			
Command name		**Function block diagram**	
MBTCPM_EXE			
Parameter	**Input/ output**	**Data type**	**Describe**
EN	IN	BOOL	Call through SM0.0
Cycletime	OUT	DWORD	This parameter is set after processing all communication operations in each cycle.
Done	OUT	BOOL	Complete the flag. 1: Completed, 0: Not completed.
Error	OUT	BYTE	0: No error. 1: There is an error (please check the error byte for details)
ErrorStep	OUT	BYTE	The operation step where the error occurred.
ErrorCode	OUT	BYTE	Error details. 0: no error. 1: The number of connections reaches the maximum value. 2: The connection is being established. 3: Timeout error. 4: The requested parameter is wrong. 5: The command is not enabled. 6: The connection is busy processing other requests. 7: Response error. 8: Failed to create connection. 9: The connection already exists. 10: The connection does not exist.

To implement the network operation configured in ModBus_TCP wizard, MBTCPM_EXE needs to be called using SM0.0 in the main block. For instance:

Initialization status bit



For the configuration used in above description, the related ModBus_TCP communication operations are implemented as following:

Table 15-13 ModBus_TCP communication description

Operation	**Local PLC data table location**	**Storage location of remote device operation status**	**Data length**
Write	VW0-VW2 -> 40001-40002	MBTCPR1_Status:VB64	2 Words
read	VW4-VW6 <- 40001-40002	MBTCPR2_Status:VB84	2 Words

To view the execution status of each configuration operation through the status monitoring table, please refer to the following table:

Table 15-14 ModBus_TCP network read and write operation status byte definition

bit	definition	Description
1	D	Done(Function completed), 0 = Not Done. 1 = Done.
2	A	Activate, 0 = not executing. 1 = executing.
3	E	Error state, 0 = No error. 1 = Error.
4	n4	Reserve, always 0.
5~8	error code	Refer the instruction comments for details.

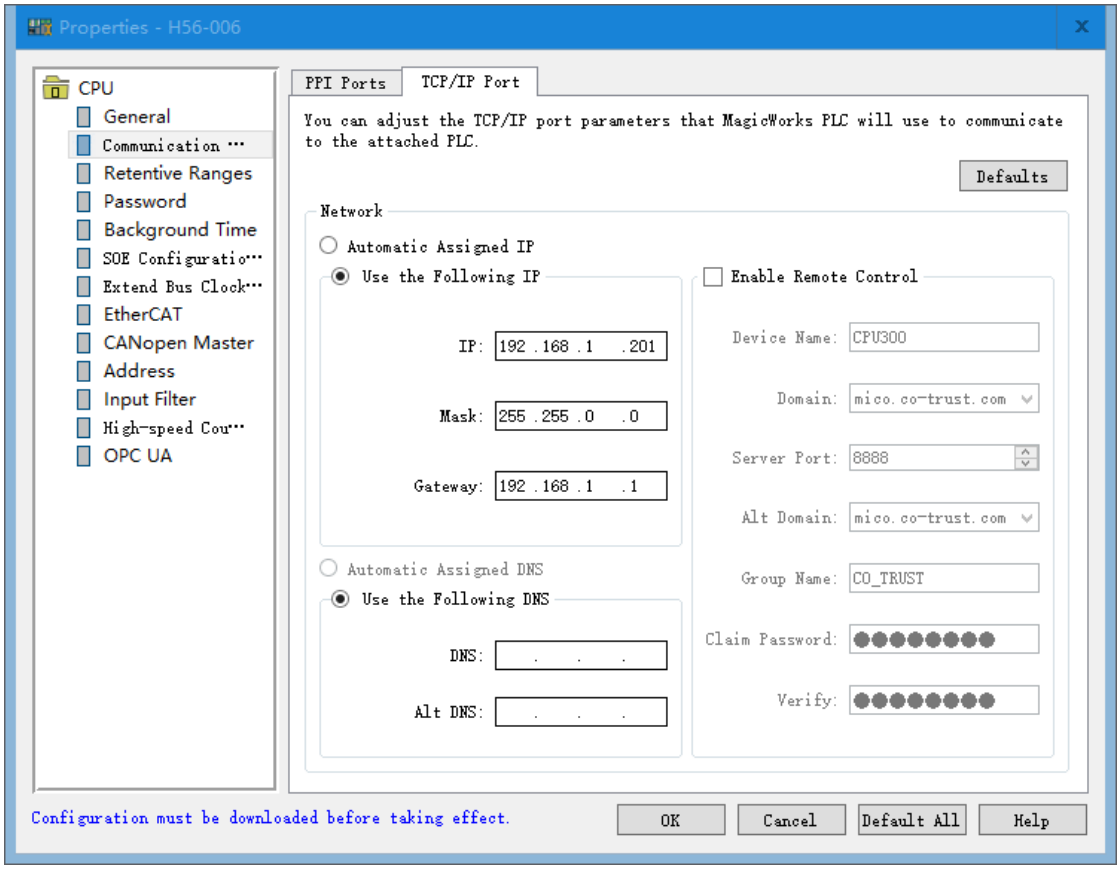
The ModBus address is a 5 or 6 character value, including the data type and offset. The first one or two characters determine the data type, and the next few characters are a corresponding value that conforms to the current data type.

Note: Up to 32 servers can be configured.

Configure the ModBus TCP instruction wizard in MagicWorks PLC, and call the FC generated by the wizard through SM0.0.

Step 5: Communication between ModBus TCP master and ModBus TCP slave

When configuring ModBus TCP slave, the slave IP must correspond to the IP set by the master. If you need to change the slave IP, please double-click the CPU module in the hardware configuration to open the properties and modify its IP address, as shown in the figure below. In addition, during ModBus TCP communication between PLCs, the slave port number should be 1024 or more, which is the default requirement of the system.



Step 6: Debug

The standard network cable connects the ModBus master and the slave, and then reads the 120 data of the ModBus_TCP slave starting with 40001 through the MagicWorks PLC status table. If

the data is consistent with the data in the corresponding memory in the ModBus_TCP master, it means ModBus TCP communication is successful.

3. ModBus TCP slave address mapping

The ModBus address is a 5 or 6 character value, including the data type and offset. The first one or two characters determine the data type, and the next few characters are a corresponding value that conforms to the current data type. The ModBus-TCP slave supports the following addresses:

Table15-15 Slave address map for TCP protocol

ModBus Slave address	Controller address
000001	Q0.0
000002	Q0.1
000003	Q0.2
...	...
002047	Q255.6
002048	Q255.7
010001	I0.0
010002	I0.1
010003	I0.2
...	...
012047	I255.6
012048	I255.7
030001	AIW0
030002	AIW2
030003	AIW4
...	...
030512	AIW1022
040001	VW0
040002	VW0+2
...	...
04xxxx	VW0+2 x (xxxx-1)

15.1.5 UDP_PPI Communication

This section shows the UDP_PPI communication (Ethernet communication port) of the H56-10 controller through an example.

In UDP PPI communication, when the CPU is used as a slave, the communication is independent of the entire cycle and can be received and returned immediately. when the CPU is used as the master, its sending and receiving are controlled by the user program.

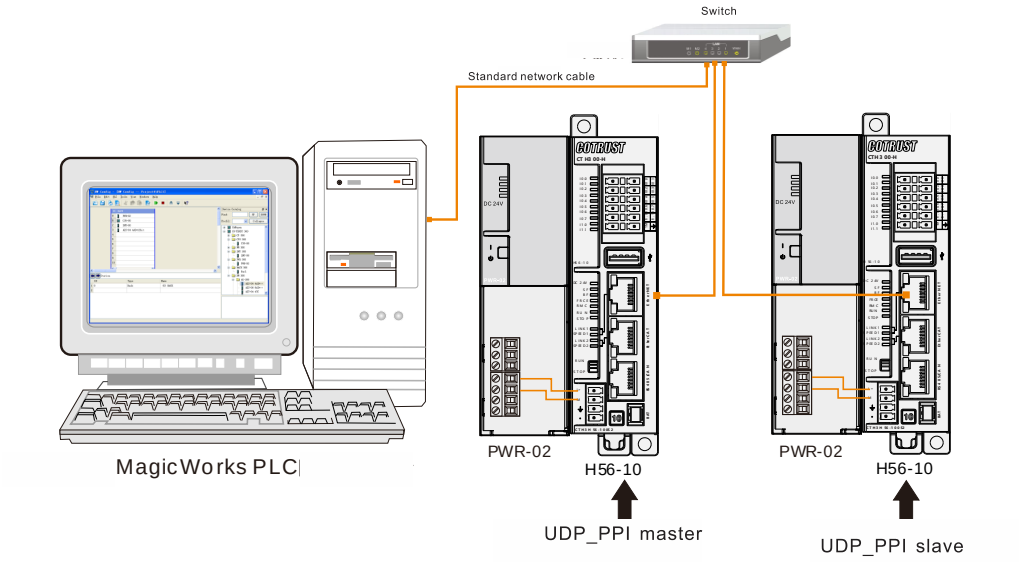
H56-10, as the master or slave, uses the network read and write commands UDP_NETR/UDP_NETW or NETW/NETR command wizard to communicate with other devices in the LAN for UDP PPI communication.

1. Preparation before communication

Table 15-16 Components of UDP PPI communication

Component	Function
Programming device PG\PC	MagicWorks PLC software (V2.19 or higher) is installed to configure, program and debug H56-10.
Power module PWR-02	Power H56-10 motion controller and its 24 VDC load circuit.
H56-10 main control module	One serves as the UDP_PPI master and the other serves as the slave. They communicate with UDP_PPI via the EtherNET interface.
Standard network cable	Connect H56-10 and programming device. Connect UDP_PPI master (H56-10) and UDP_PPI slave (H56-10).

Standard network cables and switches connect the programming device with the UDP_PP master and slave, and download the program to the UDP_PPI master through MagicWorks PLC software for monitoring. During UDP_PPI communication, the UDP_PPI master writes the data in the specified address to the UDP_PPI slave, and then reads the data in the corresponding address through the UDP_PPI slave.



2. Steps

Step 1: Wiring the H56-10 and the power module

Open the front panel of the H56-10 and the power supply module PWR-02, and then wire them according to the above network connection diagram.

Step 2: Connect the device

Standard network cables connect programming equipment, switches and UDP_PPI master and slave s (H56-10).

Step 3: Set up communication

Create a new project, add H56-10 to the project, and then refer to Chapter 2.2 Set Up Communicationto connect H56-10 with PC.

Step 4: Program the UDP_PPI master (H56-10)

Take UDP network read and write as an example, you can read and write to the UDP_PPI master network by one of the following two methods.

Read and write instructions UDP_NETR/UDP_NETW via UDP network to program the UDP_PPI master

Table 15-17 "UDP_NETR/UDP_NETW" instruction TABLE parameter table

D	A	E	0	error code	0
The first byte of IP					1
The second byte of IP					2
The 3rd byte of IP					3
4th byte of IP					4
Port number high byte					5
Port number low byte					6
Pointer to the first byte of the remote station pointer <I, Q, M, V, DB> (4 bytes in total)					7
Point to the second byte of the remote station pointer					8
Point to the third byte of the remote station pointer					9
Point to the 4th byte of the remote station pointer					10
Data length					11
Data byte 0					12
Data byte 1					13
...					...
Data byte 199					211

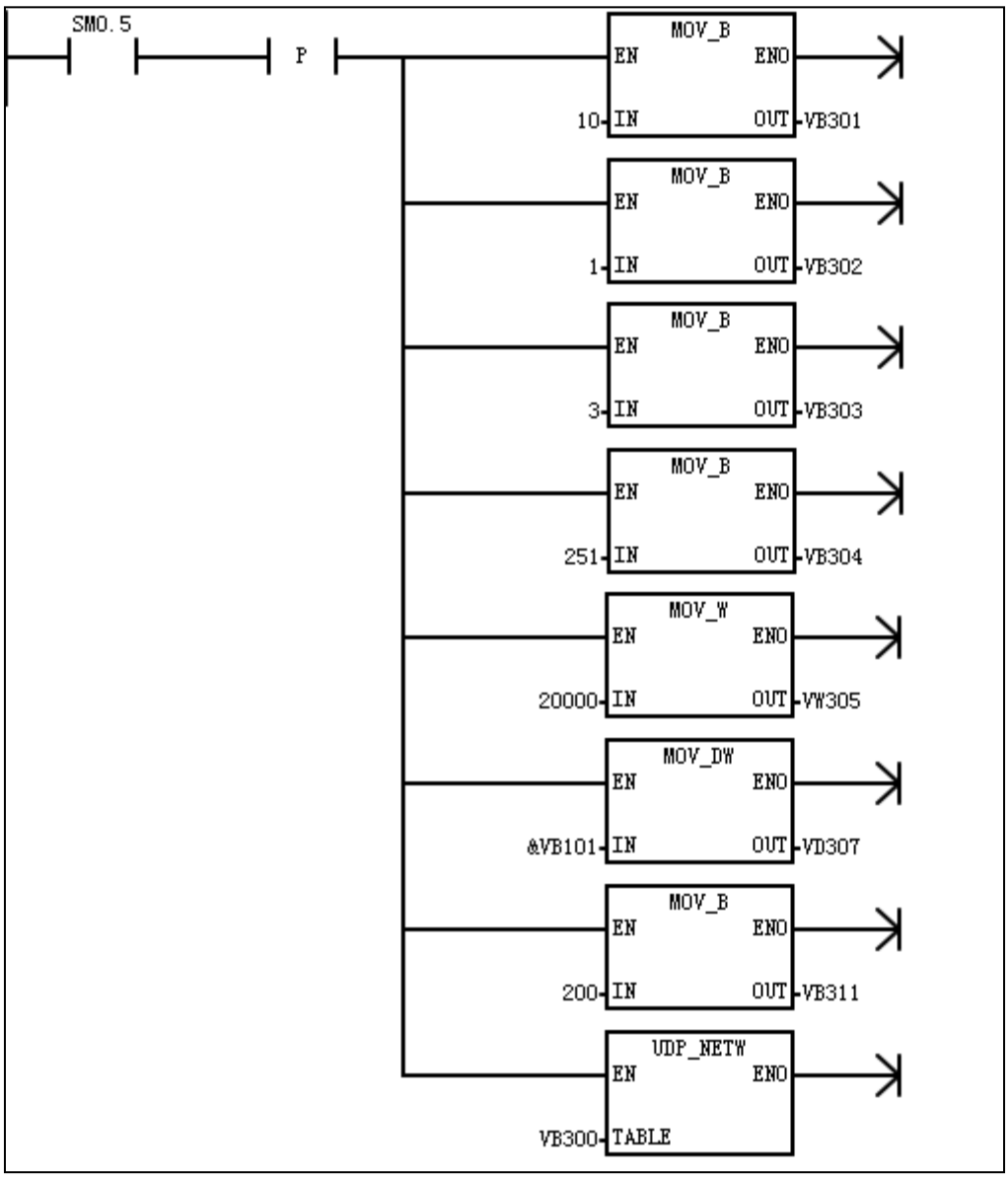
D: Completed (function completed), 0 = not completed, 1 = completed

A: Active (function joining the team), 0 = inactive, 1 = active

E: Error, 0 = no error, 1 = error

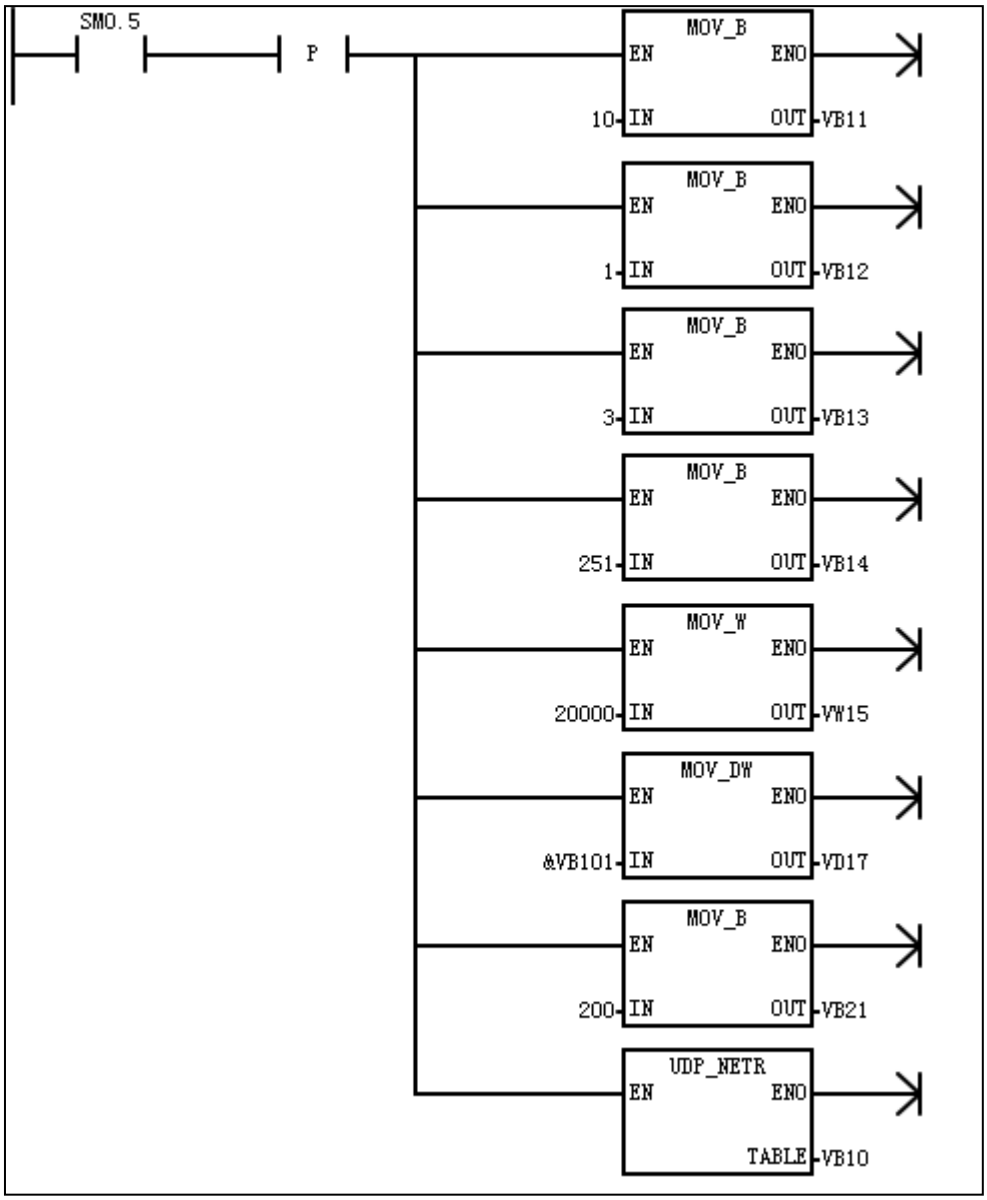
Network 1: Write 200 bytes from the UDP_PPI master (starting address: VB312) to the UDP_PPI slave (IP: 10.1.3.251, starting address: VB101).

```
PROGRAM COMMENTS: UDP PPI communication in LAN
IP: 10.1.3.251 (start address: VB301) PORT:20000   Write data length:200 bytes
```



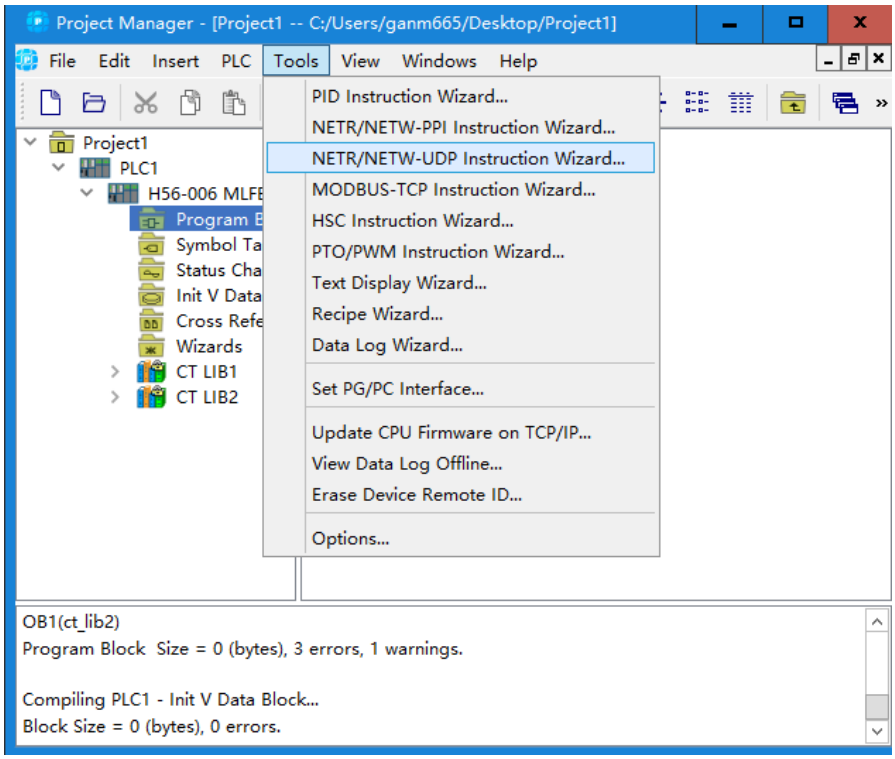
Network 2: UDP_PPI master (starting address: VB22) reads 200 bytes in UDP_PPI slave (IP: 10.1.3.251, starting address: VB101).

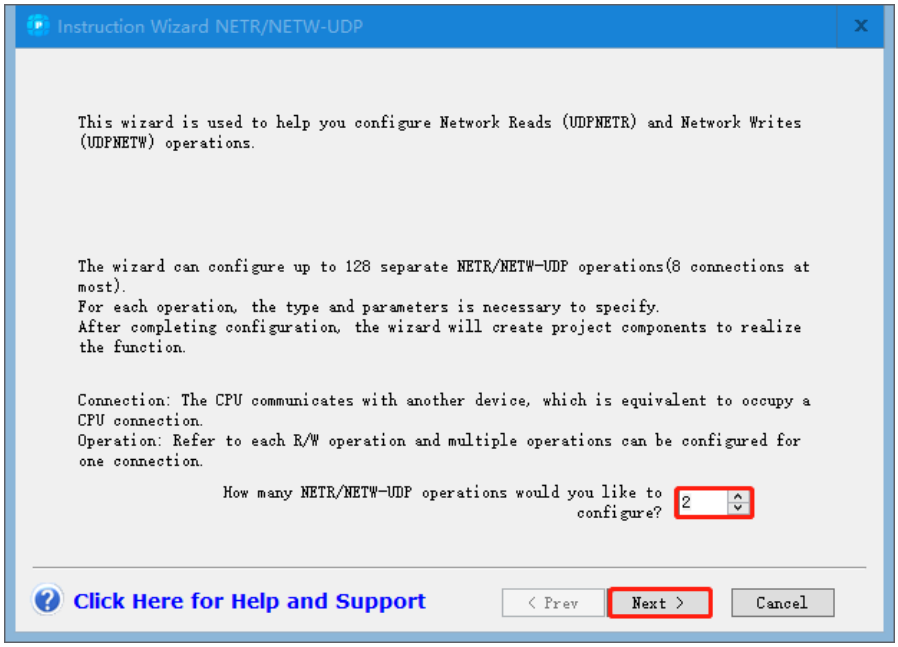
```
PROGRAM COMMENTS: UDP PPI communication in LAN
IP: 10.1.3.251 (start address: VB301) PORT:20000   reading data length:200 bytes
```

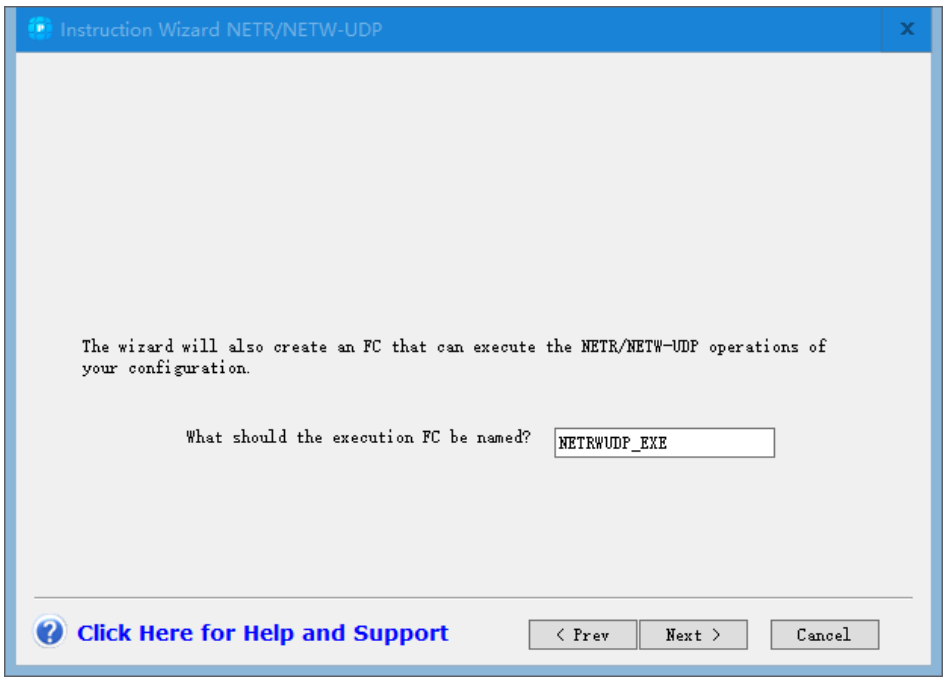


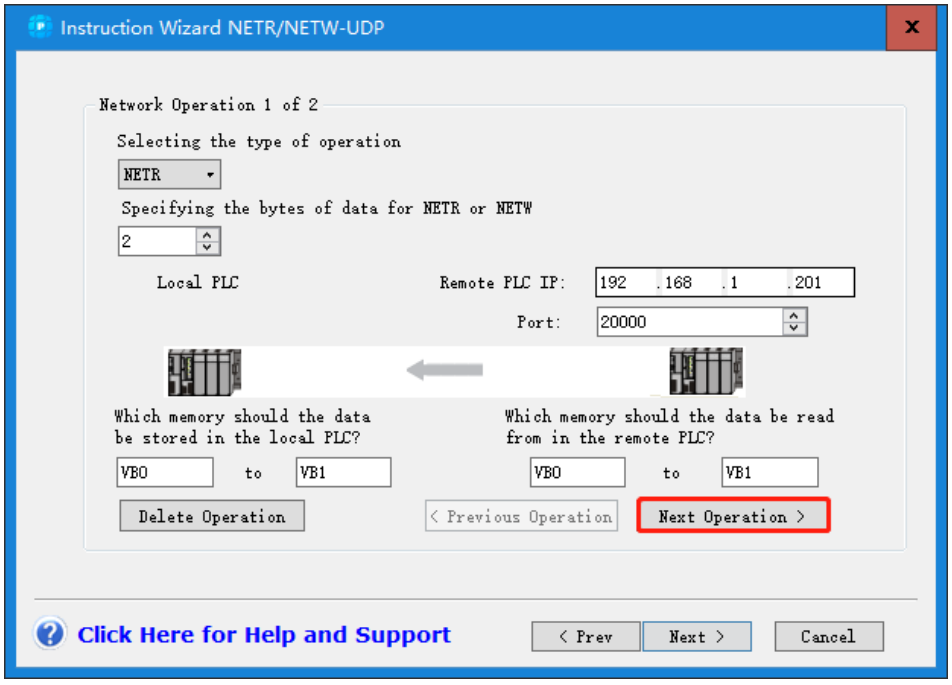
2) Program the UDP_PPI master through the NETR/NETW instruction wizard

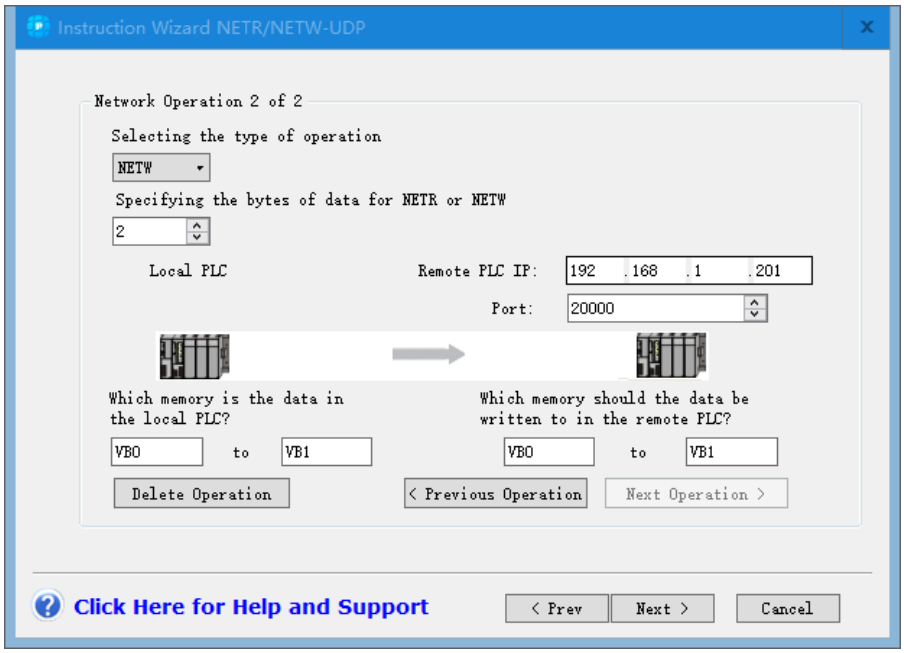
In the project manager interface menu item under the H56-10 site, select "Tools" → "NETR/NETW-UDP Command Wizard" to open the wizard, configure step by step, and finally click "Finish".

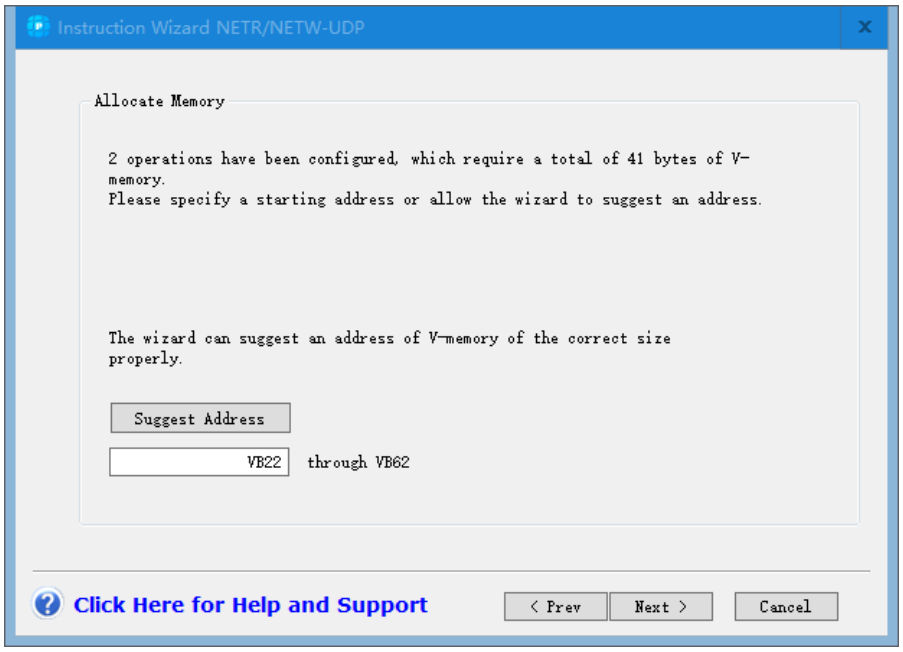

Project Manager - [Project1 -- C:/Users/ganm665/Desktop/Project1]
File Edit Insert PLC Tools View Windows Help
PID Instruction Wizard...
NETR/NETW-PPI Instruction Wizard...
NETR/NETW-UDP Instruction Wizard...
MODBUS-TCP Instruction Wizard...
HSC Instruction Wizard...
PTO/PWM Instruction Wizard...
Text Display Wizard...
Recipe Wizard...
Data Log Wizard...
Set PG/PC Interface...
Update CPU Firmware on TCP/IP...
View Data Log Offline...
Erase Device Remote ID...
Options...
Project1
PLC1
Wizards
CT LIB1
CT LIB2
OB1(ct_lib2)
Program Block Size = 0 (bytes), 3 errors, 1 warnings.
Compiling PLC1 - Init V Data Block...
Block Size = 0 (bytes), 0 errors.


Instruction Wizard NETR/NETW-UDP
This wizard is used to help you configure Network Reads (UDPNETR) and Network Writes (UDPNETW) operations.
The wizard can configure up to 128 separate NETR/NETW-UDP operations(8 connections at most).
For each operation, the type and parameters is necessary to specify.
After completing configuration, the wizard will create project components to realize the function.
Connection: The CPU communicates with another device, which is equivalent to occupy a CPU connection.
Operation: Refer to each R/W operation and multiple operations can be configured for one connection.
How many NETR/NETW-UDP operations would you like to configure?
2
Click Here for Help and Support
< Prev
Next >
Cancel


Instruction Wizard NETR/NETW-UDP
The wizard will also create an FC that can execute the NETR/NETW-UDP operations of your configuration.
What should the execution FC be named?
NETRWUDP_EXE
Click Here for Help and Support
< Prev
Next >
Cancel


Instruction Wizard NETR/NETW-UDP
Network Operation 1 of 2
Selecting the type of operation
NETR
Specifying the bytes of data for NETR or NETW
2
Local PLC
Remote PLC IP:
192 .168 .1 .201
Port:
20000
Which memory should the data be stored in the local PLC?
VB0 to VB1
Which memory should the data be read from in the remote PLC?
VB0 to VB1
Delete Operation
< Previous Operation
Next Operation >
Click Here for Help and Support
< Prev
Next >
Cancel





After completing the above operations, the FC can be called through SM0.0 in the OB1 network

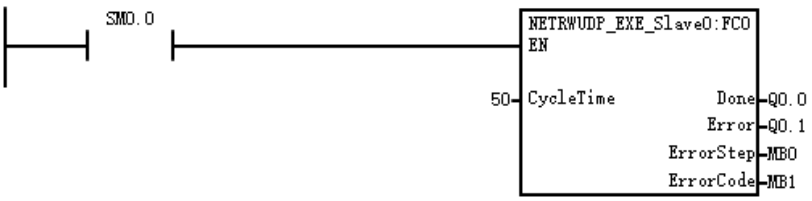


Table 15-18 UDP_PPI communication network operation description

Operation	Local PLC data table location	Remote device oprating state storage location	Data length
write	VB0-VB1 -> VB0-VB1	MBTCPR1_Status: VB4	2 Bytes
Read	VB2-VB3 <- VB0-VB1	MBTCPR2_Status: VB18	2 Bytes

3) After completing the program editing according to the above method 1) or 2), compile the

program and download it to the UDP_PPI master device.

Step 5: UDP_PPI master and UDP_PPI slave communicate with each other

After the above steps are completed, power on the UDP_PPI master and slave to perform UDP_PPI communication.

3. UDP_PPI communication slave address mapping

UDP_PPI communication adopts common network read and write, multiple registers can be configured when performing read and write operations, and the slave address is mapped in direct mapping mode.

15.1.6 Remote EtherNET Communication

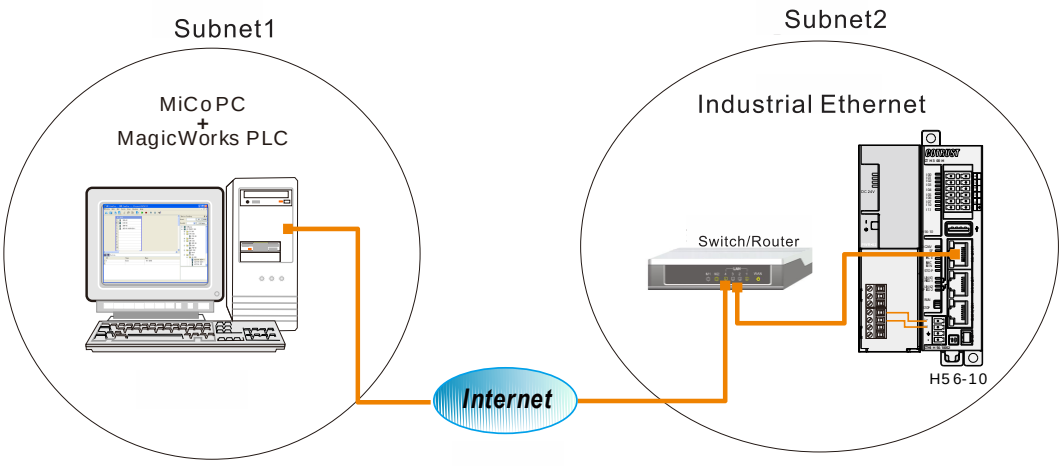
This section will demonstrate the remote Ethernet communication function of H56-10 through a concrete example.

When selecting remote Ethernet communication, MiCo needs to establish communication with the remote H56-10 first, and then the remote H56-10 can be programmed and monitored through the MagicWorks PLC programming software.

1. Preparation before communication

Table 15-19 Example components for remote EtherNET communication

Component	Function
Programming device PG \ PC	MagicWorks PLC software (V2.19 or higher version) is installed to configure, program and debug the H56-10 motion controller. Installed with MiCo to communicate with H56-10 motion controller.
Assembly rail	The CTH300 system rack is used to fix each module in the system.
Power Module PWR-02	Powers the H56-10 motion controller and its 24 VDC load circuit.
H56-10 main control module	CPU executes the user program, provides 5V voltage to the CTH300 system backplane bus, and communicates with other modules through the Ethernet interface.
router	Connect the devices in Subnet 1 and Subnet 2 so that the devices in Subnet 1 and Subnet 2 can read and write each other's data.
2 standard network cables	Connect PC and router, connect H56-10 and router



Subnet 1 contains a remote programming device (installed with software MiCo and MagicWorks PLC (V2.18 and above)), which can be used to communicate and program the remote CPU. Subnet 2 contains remote programming objects (H56-10), which is connected to the router or switch through the EtherNET communication port. Connect the subnets 1 and 2 to the WAN, configure the CPU and download the hardware configuration block, you can automatically log in to the COTRUST MICO server and wait for the remote programming device to operate

< Remarks > You can also install MiCo on your mobile phone or Pad, and then open the remote programming port, you can use MagicWorks PLC to operate the remote CPU on the PC in the same local area network.

2. Operation steps

Step 1: Wiring

Open the front panel of the H56-10 and the power module PWR-02 and wire them.

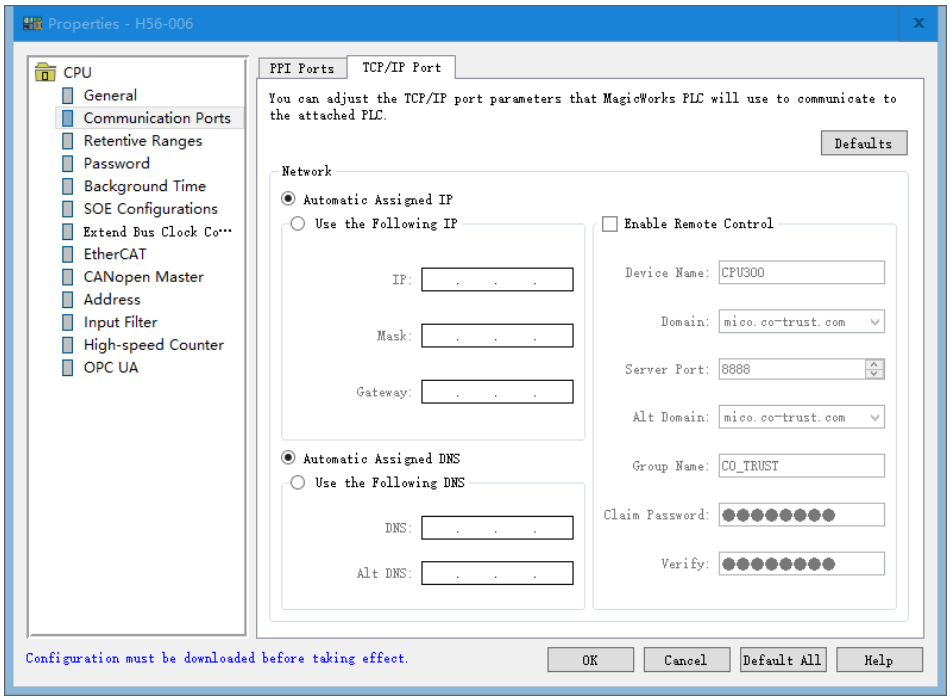
Step 2: Connect the Cables

Refer to the above network connection diagram to wire each communication device.

Step 3: CPU Properties Configuration

1) Double-click  under H56-10 to enter the "Hardware Configuration". For detailed steps of hardware configuration, please refer to Chapter 2.3 HW Config.

2) In the "Hardware Configuration", double-click H56-10 on the rack to configure the CPU properties. In the property configuration interface, select "Communication Port" → "TCP/IP Port" to configure IP and remote functions. Select "Automatic Assigned IP" and "Automatic Assigned DNS server address", the premise is to ensure that the controller and the programming device are in the same network segment.



Step 4: Download the hardware configuration block to the PLC

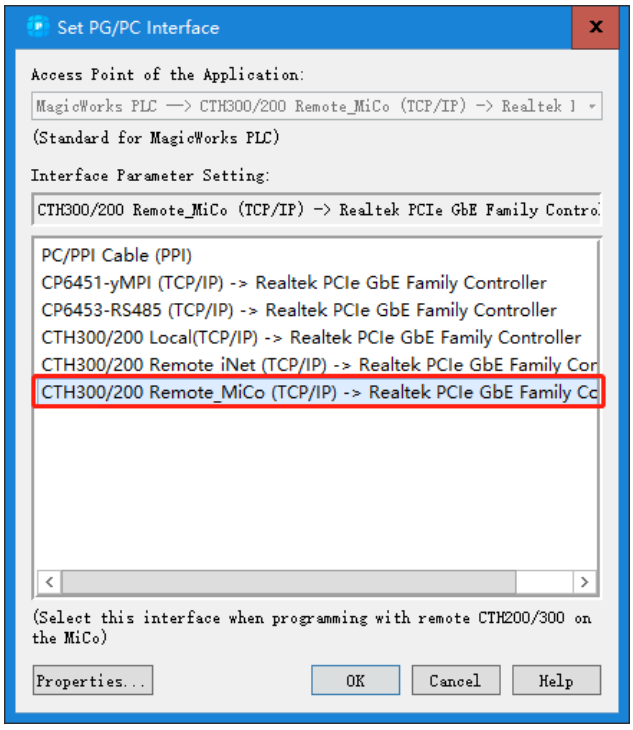
Step 5: MiCo side configuration

Claim the device on the MiCo client, add the CPU to my device. And start remote programming.

After completing the above steps, start the remote control, if the RMC indicator of the PLC is on, it means that the remote communication is successful and the device has been connected to the MiCo server.

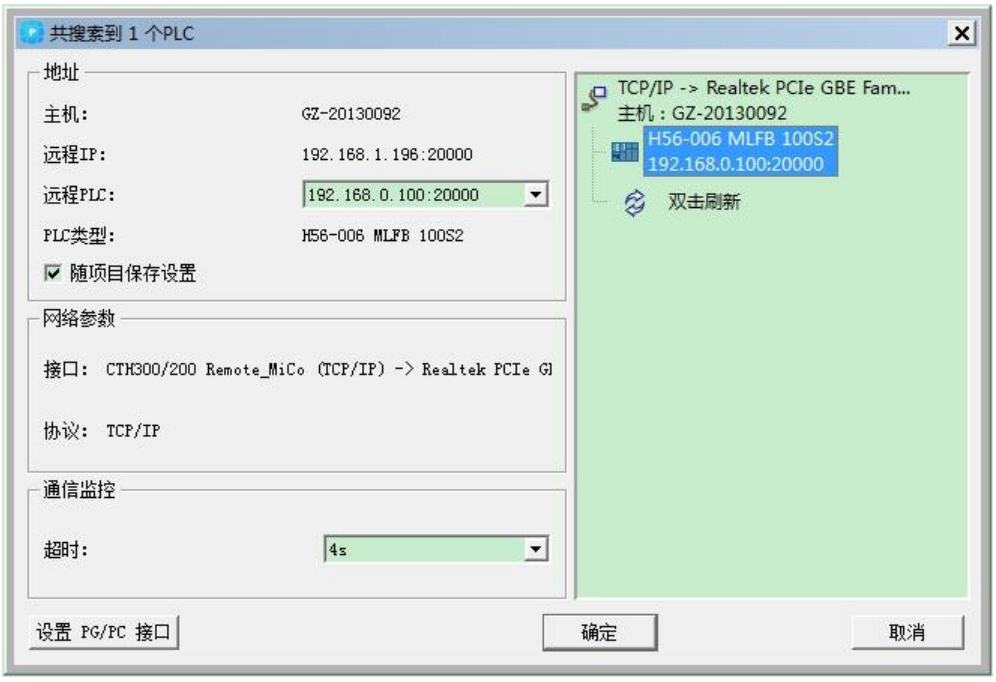
Step 6: Set up the communication connection

1) In the MagicWorks PLC project manager interface, select " Tools "→" Set PG/PC Interface " or double-click " Set PG/PC Interface " to set PG/PC, select "CTH300/200 Remote_Mico(TCP/IP) → Realtek Pcle GBE Family Controller", then click " OK " to complete the configuration.



2) Double-click the "communication" in the MagicWorks PLC project manager interface, and then

double-click the icon to search for the CPU. The successfully connected controller will be displayed in the communication interface.



3) After the PLC is successfully searched in the communication interface , it means that the communication is successful, and operations such as remote programming or monitoring can be performed.

<Remarks> When the system fails, please refer to 5.8 CPU Fault Diagnosis

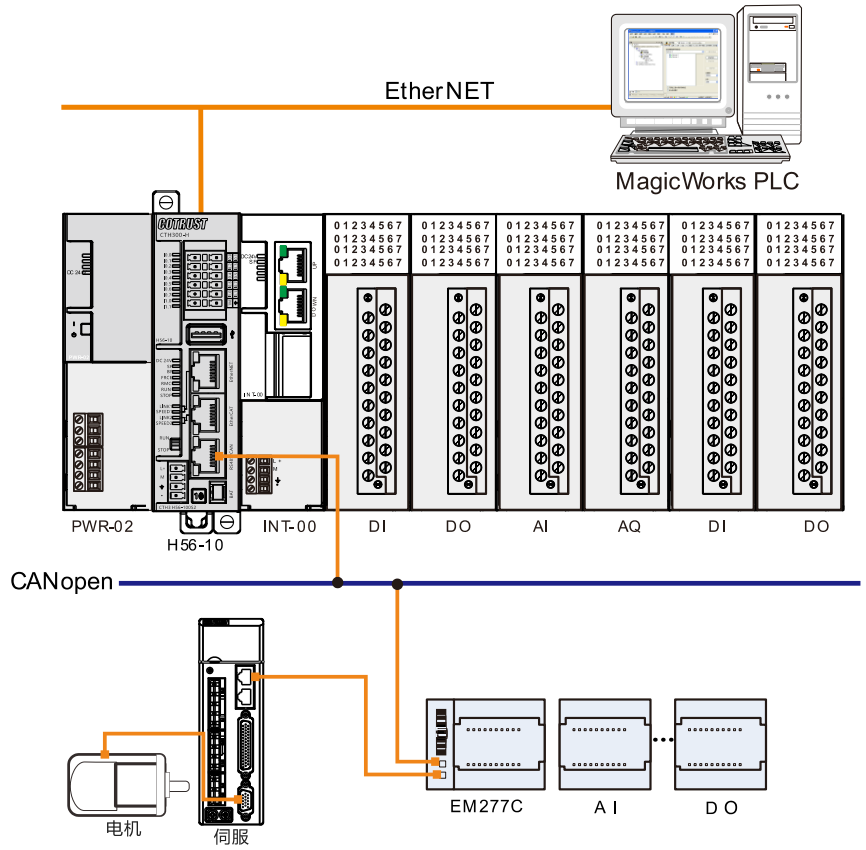
15.1.7 CANopen Communication

1. Preparation before communication

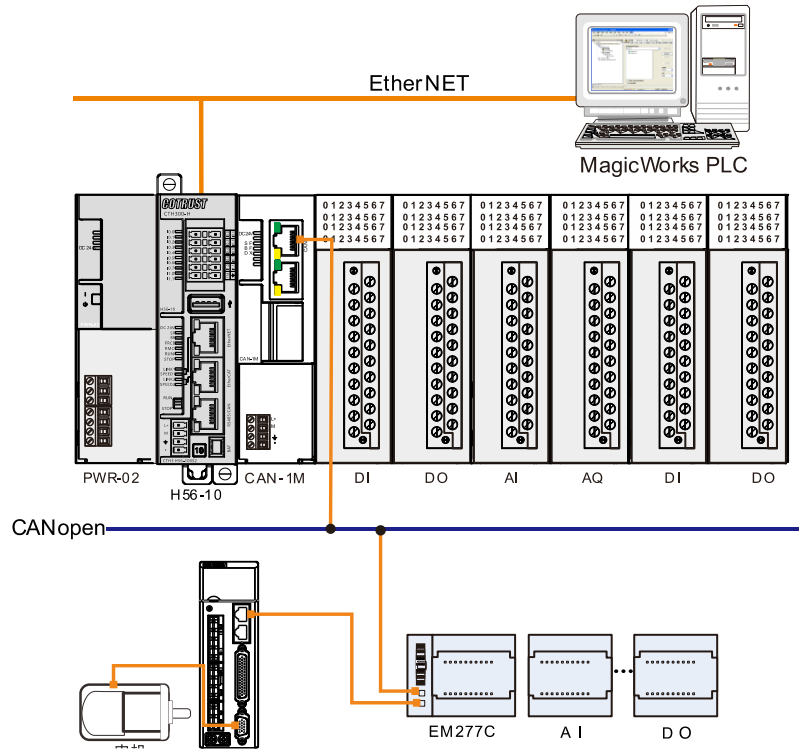
Table 15-20 Example components for CANopen communication

Component	Function
Programming device PG\PC	MagicWorks PLC software (V2.19 or higher) is installed to configure, program and debug H56-10.
Power Module PWR-02	Powers the H56-10 motion controller and its 24 VDC load circuit
H56-10 main control module	CPU executes the user program, provides 5V voltage to the CTH300 system backplane bus, and communicates with other nodes in the Ethernet network through the Ethernet interface.
CAN-1M master module (optional)	In CANopen communication, as CAN master, communicate with CPU and CAN slave.
CAN slave device	EM277C, servo driver.
Expansion module for EM277C	EM277C expansion module EM231 AI4, EM222 16DO.
E10 Servo Motor	Connect with E10 servo driver.
3 standard network cables	● Connect H56-10 with programming device ● Connect CAN-1M with EM 277C ● Connect CAN-1M and servo drive
Encoder cable	Connect the Servo Drive to the Motor

CANopen communication connection (through the CAN communication port of the CPU):



CANopen communication connection (through CAN-1M module):



2. Operation steps (take the connection CAN-1M module as an example)

Step 1: Wiring

Open the front panel of H56-10, power supply module and CAN-1M, and then connect them according to the above network connection diagram. The specific steps are as follows:

1) Use standard network cable to connect PC and H56-10.

2) H56-10 and CAN master module are connected through bus.

3) Use standard network cable to connect CAN-1M module and EM277C.

4) Use standard network cable to connect EM277C and servo.

5) The expansion module is connected to the EM277C through the bus.

Step2: Set up communication

Create a new project in MagicWorks PLC, add the H56-10 in the project, and then refer to chapter 2.2 Set Up Communication to connect the H56-10 to the PC for communication.

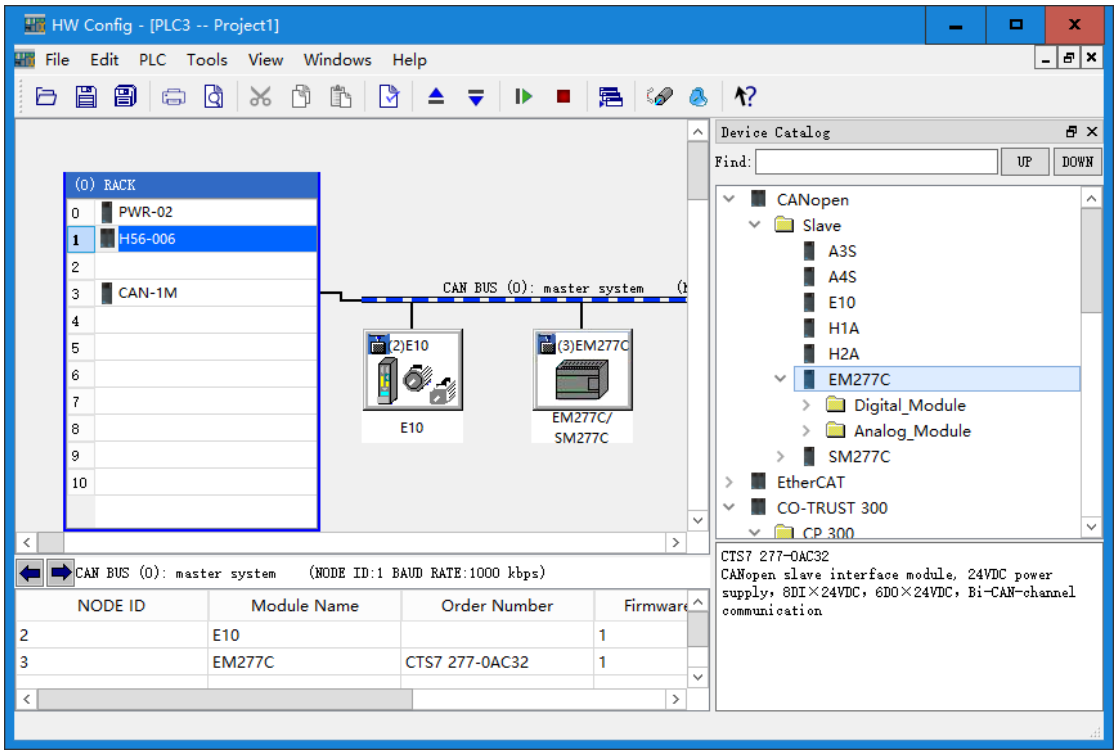
Step 3: Hardware configuration in MagicWorks PLC

in the MagicWorks PLC project, double-click the hardware configuration icon  to enter the hardware configuration interface.

1 Add power supply, CPU, CAN-1M master module, CAN slave module and slave extension module.

In the hardware configuration interface, select rack Rack 0, then expand the CO-TRUST 300 node in the device catalog, first select the power supply module PWR-02, drag and drop it to slot 0 of the rack (Rack 0), and then Select H56-10, drag and drop it to rail slot 1, and finally select CAN-1M, drag and drop it to rail slot 3.

Expand " CANopen "→"Slave" in the device catalog, and place EM277C and E10 on the CAN bus of the CAN-1M master module by dragging or double-clicking. Then add expansion modules EM231 AI4 and EM222 16DO for EM277C.

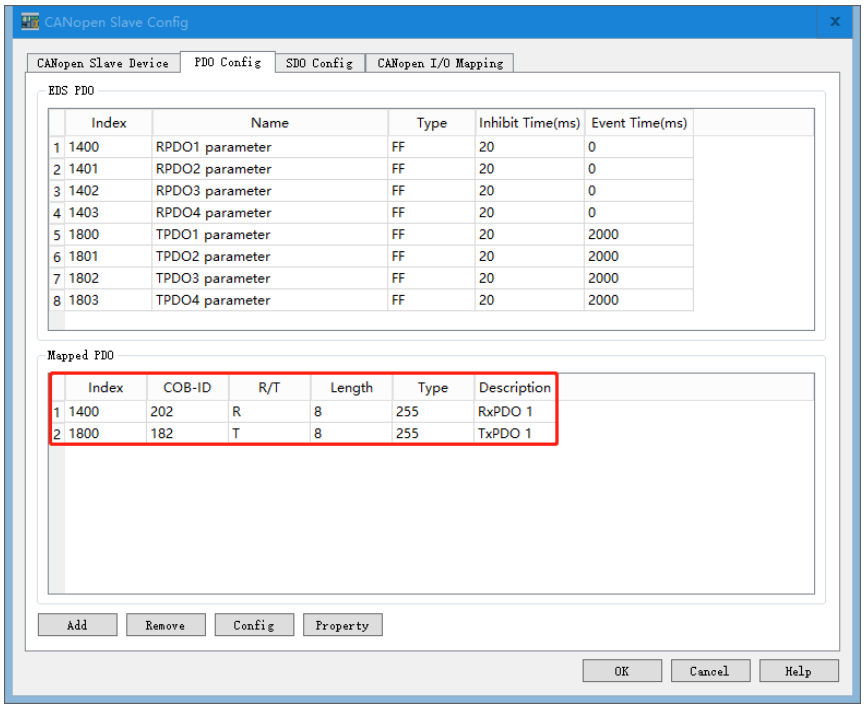


2 Configure parameters for CAN slave (E10)

Double-click the slave E10 on the CAN bus to configurate slave, then double-click the index 1400 in the "mapped PDO" to open the corresponding PDO mapping (as shown in the figure below), and then in the corresponding PDO Double-click P97 (set the running speed of the motor)

and P290 (set the given position size) in the parameters of the EDS file respectively. P97 and P290 are then added to the mapped parameter list.

<Note> Double-click or right-click a parameter in "Mapped Parameters" to configure the parameter or modify its properties.

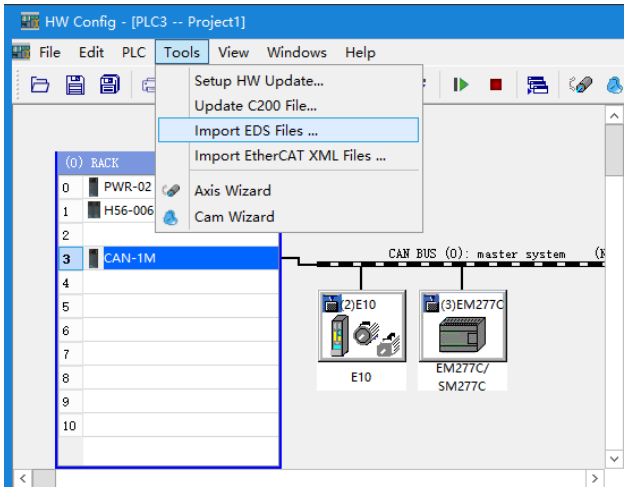


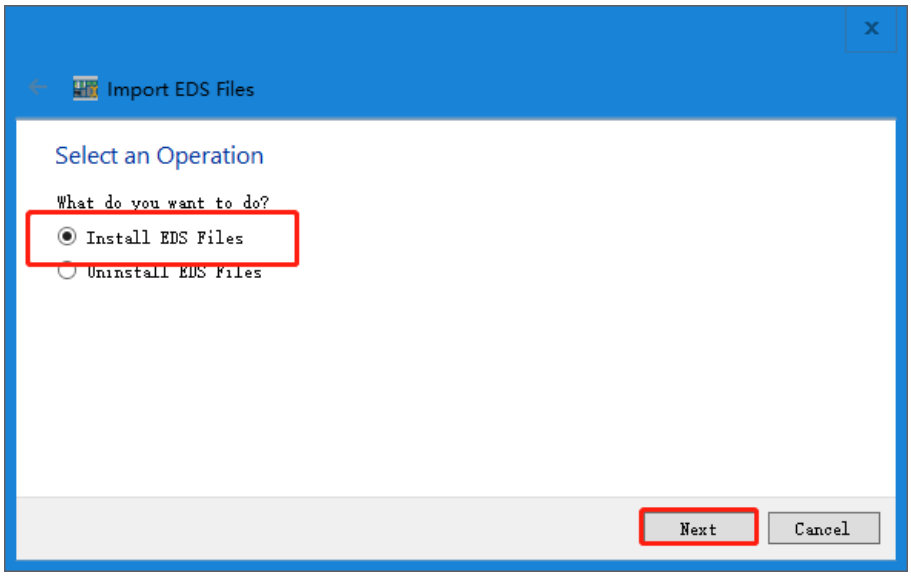
3. Save and compile the configuration

Select "File" → "Save" to save the current configuration, and then select "PLC" → "Compile" to compile the current project. If the compilation is correct, you can perform the following debugging steps.

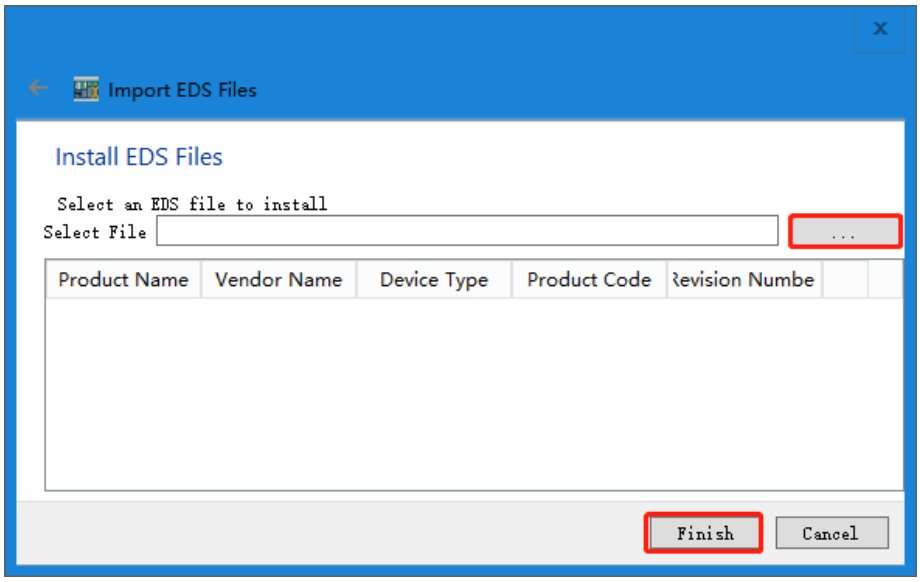
<Note> If you choose a third-party CAN slave, you need to import the corresponding EDS file first. Take Schneider Lexium 15 series servo driver as an example, please follow the steps to import its EDS file:

1) Open "Hardware Configuration" → "Tools" → "Import EDS File (E)" under the configuration project to open the following window, select "Install EDS file" and click "Next", as shown in the figure below.





2) Click the "..." to open the location of the EDS file, as shown in the figure below.



3) After selecting the storage path and adding it, the modified EDS file will be displayed in the window, as shown in the figure below, click "Finish" to complete the importing of the EDS file.

4) After completing the above operations, the file is listed in the list of CANopen slaves, and the user can configure and call it according to actual needs.

Step 4: Debug

1. Debug EM 277C and its expansion modules

Write 1 for all output of EM222 16DO (Q7.0~Q7.7, Q8.0~Q8.7) of the EM277C expansion module , and successfully light all output of EM222 16DO means that the debugging is successful.

2. Debug the servo drive and motor

1) Refer to the "*A4S Series AC Servo Drive Instruction Manual*" to configure the parameters of the current servo slave: P01=1 (communication position control mode), P11=1 (CANopen baud rate is set to 1000Kbps), P0=3 (communication address Set to 3), then power off and restart the servo drive.

2) Refer to the detailed explanation of parameters in "*A4S Series AC Servo Drive Instruction Manual*", and read and write P97 (QW4) and P290 (QW6) in the status table. Select P97=500, P290=10000, that is, in QW4 Write a new value of 500, and write a new value of 10000 to QW6.

3) Select the MagicWokrs PLC menu item "PLC" → "Download" to download the current configuration to H56-10; then, select "PLC" → "Run" to make the application program in H56-10 start to run, then E10 The motor of the servo drive starts to run, and stops after executing 10,000 given positions.

<Remark> When the system fails, please refer to Chapter 5.8 CPU Fault Diagnosis.

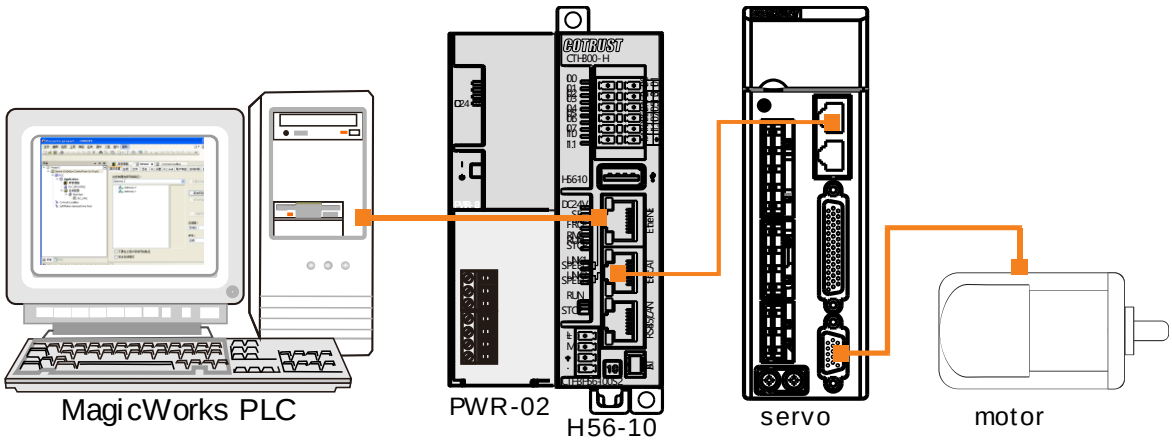
15.1.8 EtherCAT Communication

This section will demonstrate the EtherCAT communication function of H56-10 through a specific example .

1. Preparation before communication

Table 15-21 Example components for EtherCAT communication

Component	Function
Programming device PG\PC	MagicWorks PLC software (V2.19 or higher) is installed to configure, program and debug H56-10.
Assembly rail	The CTH300 system rack is used to fix each module in the system.
Power Module PWR-02	the H56-10 motion controller and its 24VDC load circuit
H56-10 main control module	CPU executes the user program, supplies 5 V to the CTH300 system backplane bus, and communicates with other nodes in the Ethernet network through the Ethernet interface.
EtherCAT slave device	A4N Servo Driver.
A4N Servo Motor	Connect with A4N servo driver
2 standard network cables	● Connect H56-10 with programming device ● Connect H56-10 and A4N servo drive
Encoder cable	Connect A4N Servo Drive and Motor



<Remarks> Please use the OUT port of the EtherCAT communication port for communication, and the IN port is unavailable in the case of non-redundancy .

2. Operation steps

Step 1: Wiring

Open the front panel of H56-10 and power supply module, and then refer to the above network connection diagram for wiring. The specific wiring steps are as follows:

1) Use standard network cable to connect PC and H56-10

2) Use standard network cable to connect H56-10 and A4N servo drive

3) Use the encoder cable to connect the A4N servo drive and the motor

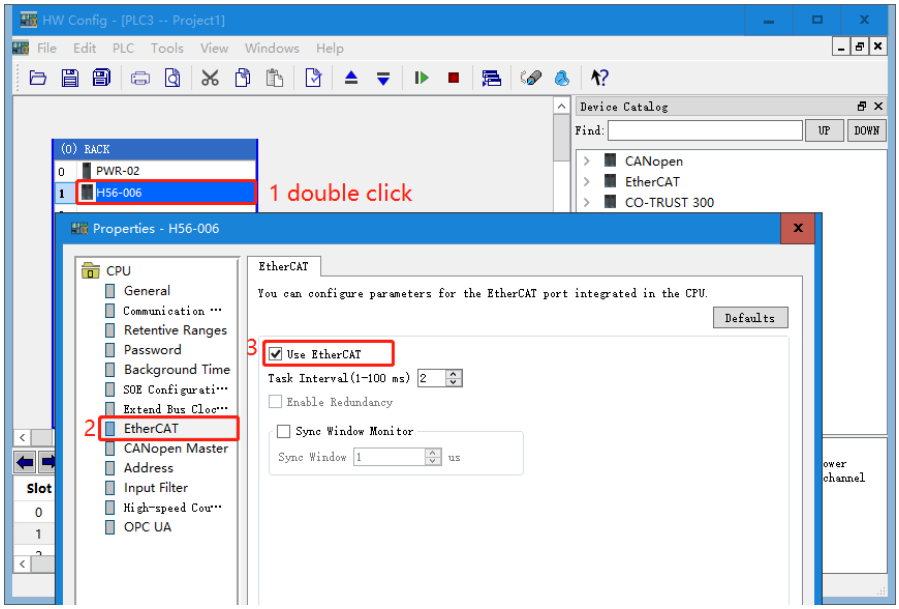
Step 2: Set up communication

Create a new project in MagicWorks PLC, add the H56-10 in the project, and then refer to chapter 2.2 Set Up Communication to connect the H56-10 to the PC for communication.

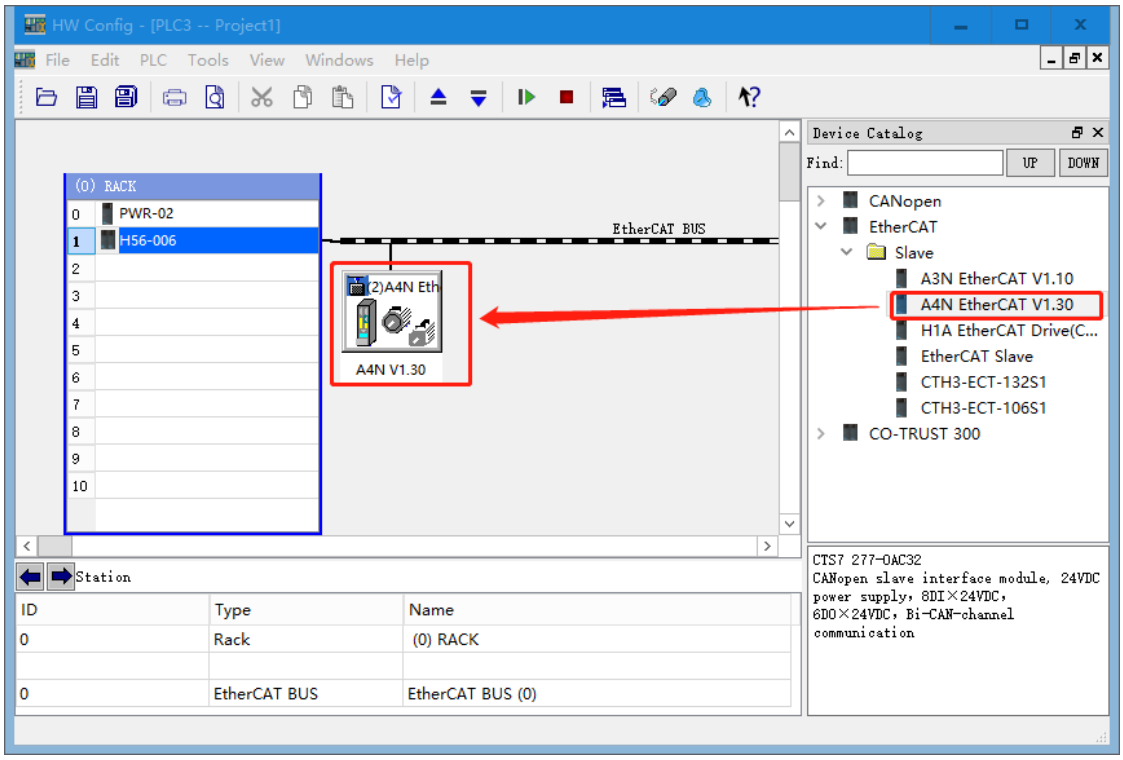
Step 3: Hardware configuration in MagicWorks PLC

in the MagicWorks PLC project, double-click the hardware configuration icon  to enter the hardware configuration interface.

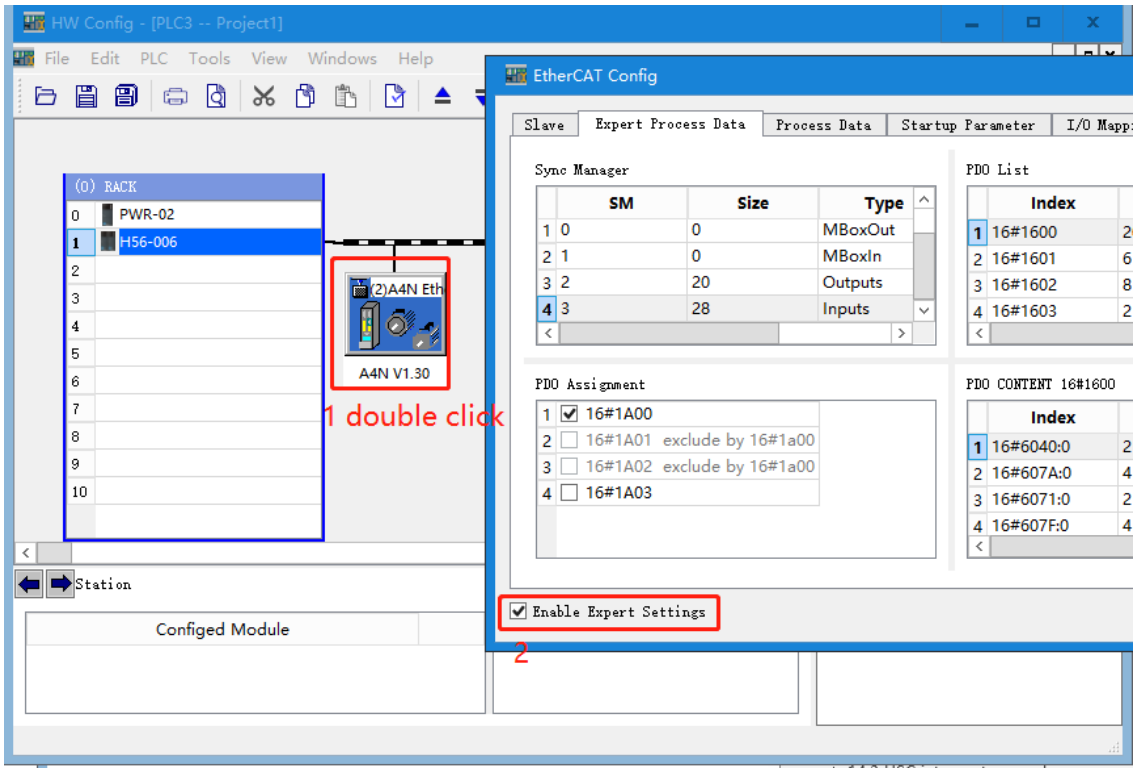
1) In the hardware configuration interface, add the power supply and CPU to the rack through the device catalog, then double-click the inserted H56-006 to open its properties, select the "EtherCAT" tab in the left column, and check " Enable EtherCAT Master Function" to enable the EtherCAT function.



2) From the EtherCAT node of the device directory tree, open the slave node Slave, select the slave device model A4N that matches the current actual device, select and drag the mouse to enter the EtherCAT bus area of the configuration interface and put it down, the slave is successfully added.

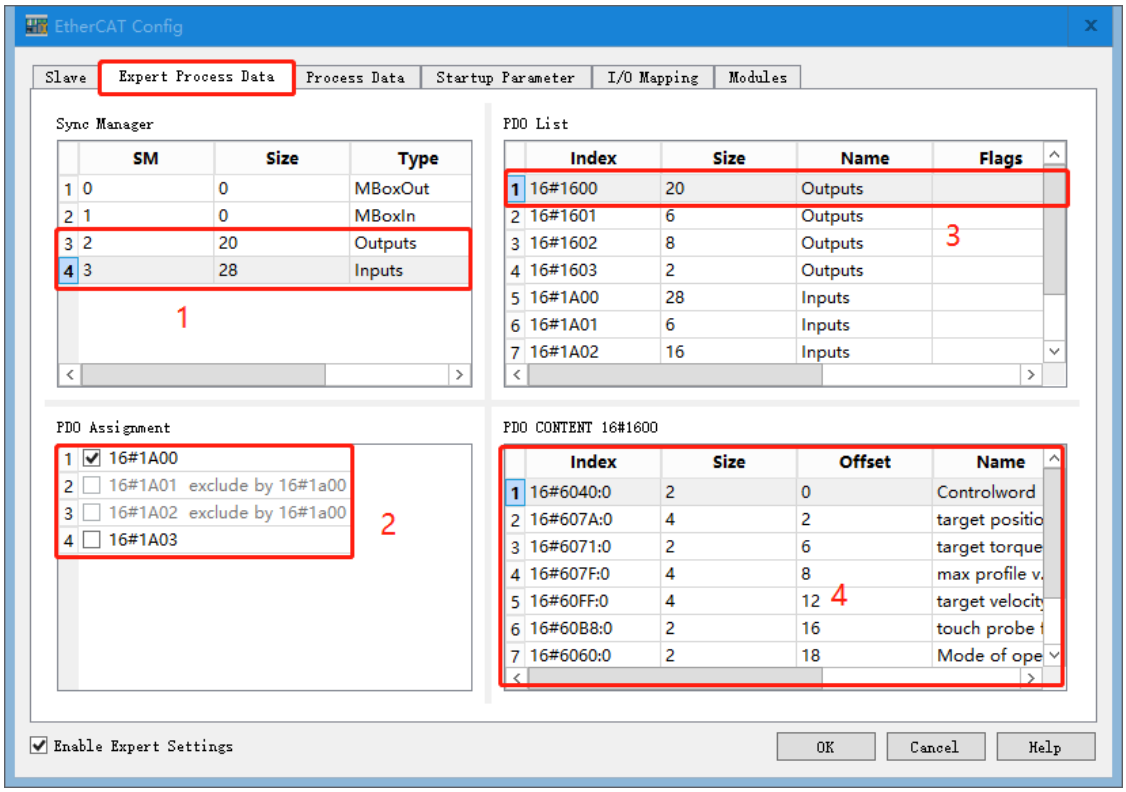


3) Double-click the A4N slave on the EtherCAT bus, For the specific parameter configuration, refer to the following description. Check "Enable Expert Settings" to configure the expert process data.

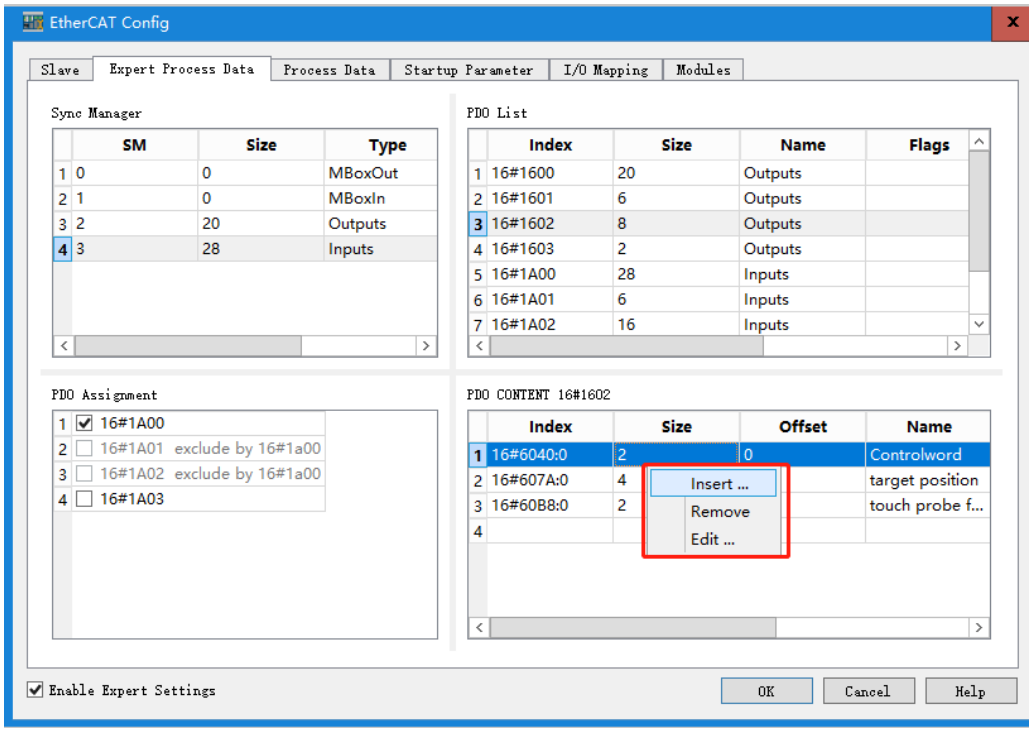


◆ "Expert process data"

Select input/output through Sync manager, then check the input/ output group in PDO Assignment, select the corresponding group in the PDO List, and finally add or delete parameters for this group in PDO Content.

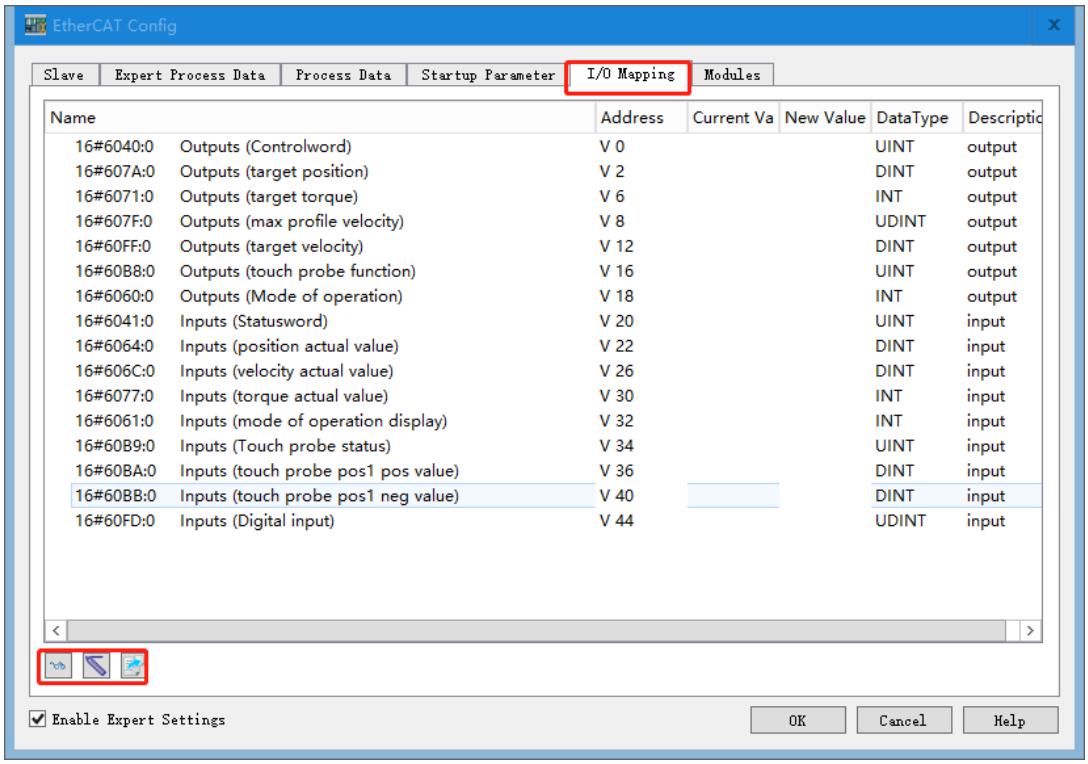


To add a variable for a specific PDO parameter, please select the parameter, right-click in the PDO Content area and select "Insert", the setting window for adding variables will be displayed, select the variable to be added, and click "OK", as shown in the following figure:



◆ I/O Mapping

The successfully configured I/O parameters will be displayed in the I/O map. Double-click the address to edit its address.

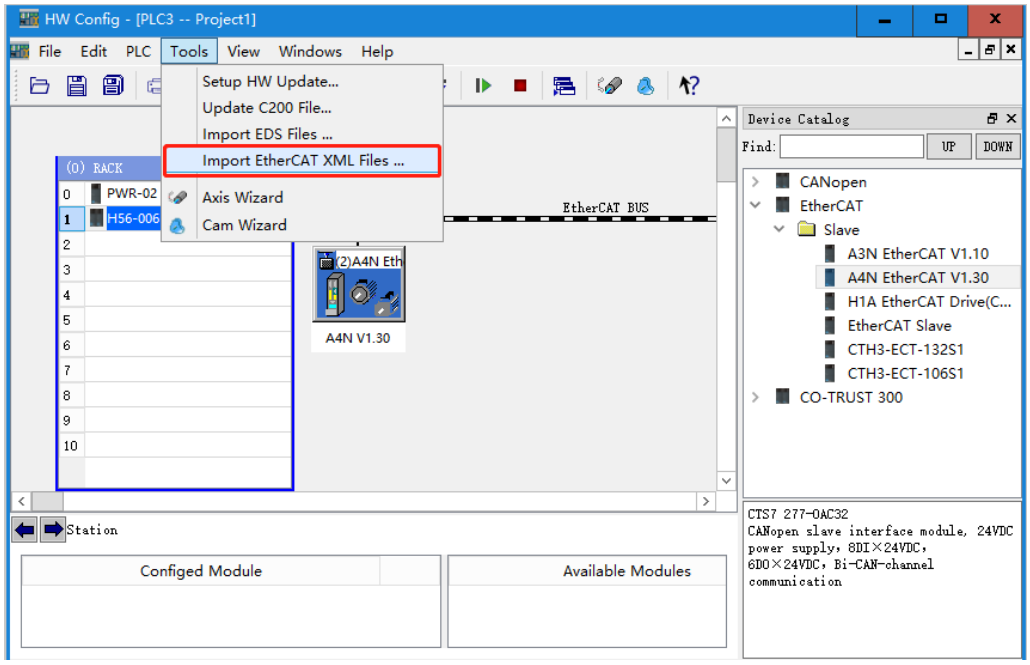


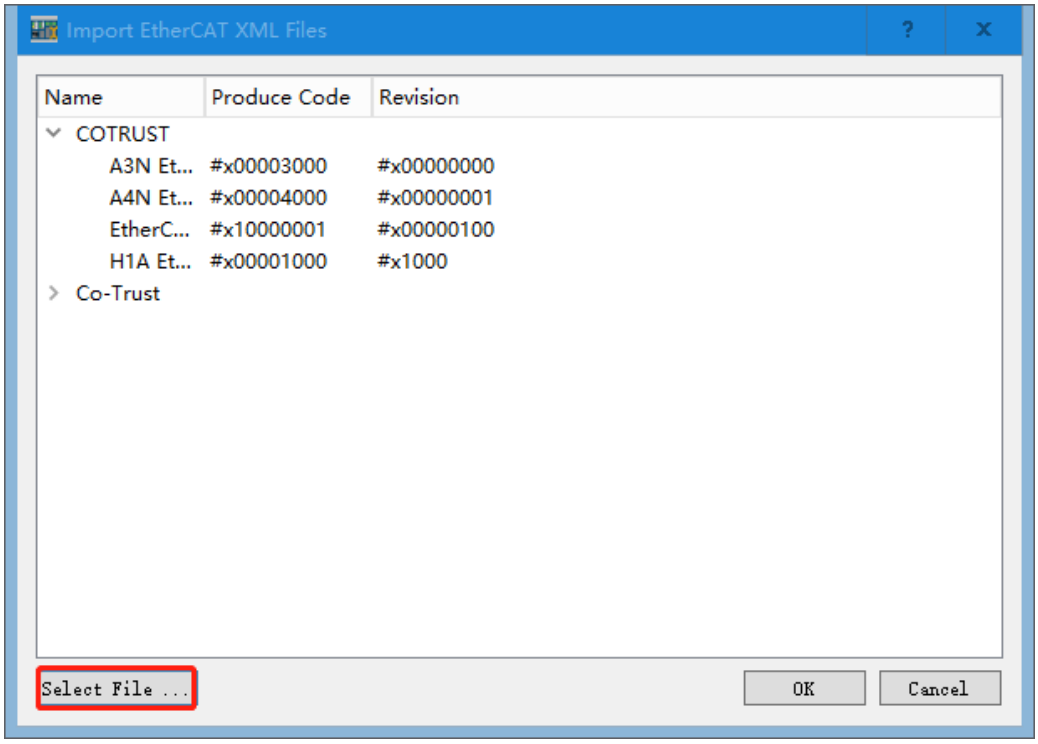
4) Click the Monitor in the "I/O Mapping" tab of the EtherCAT configuration dialog to start monitoring the configured parameters, then enter the parameter value in the "New Value", and finally click the Write button to successfully write the new value. value.

Import third-party slaves

If you choose a third-party EtherCAT slave, you need to import the corresponding XML device description file first. Please follow the steps to import its XML file:

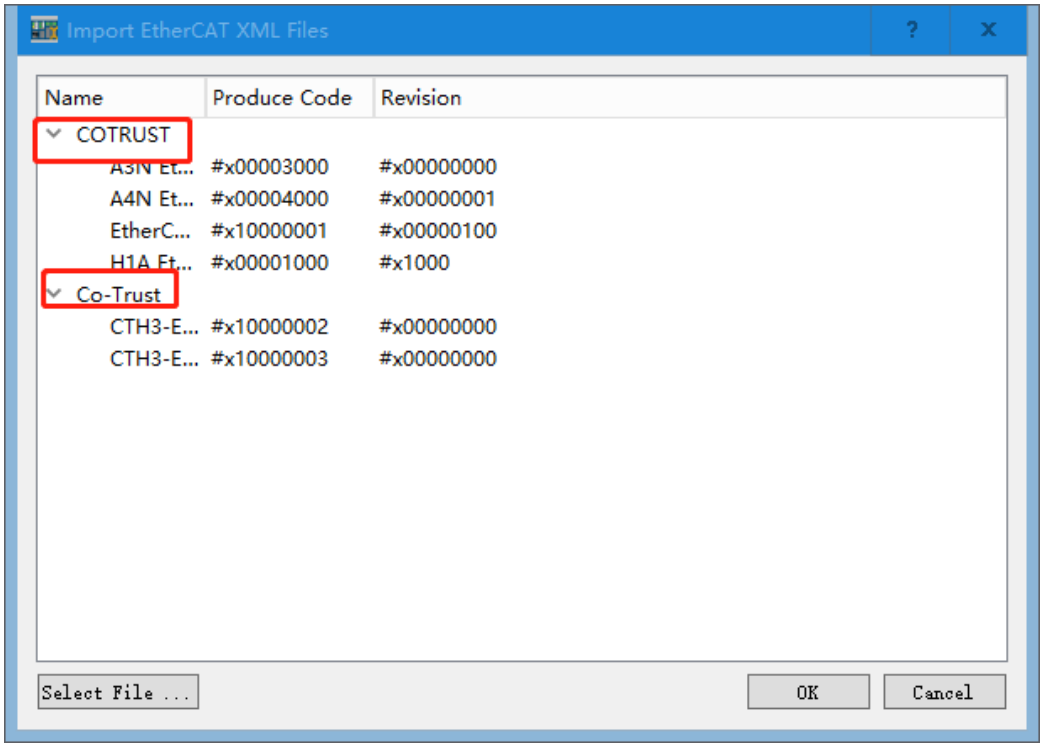
◆ Open "Hardware Configuration" → "Tools" → "Import EtherCAT XML File (X)" under the configuration project, and the following interface is displayed:



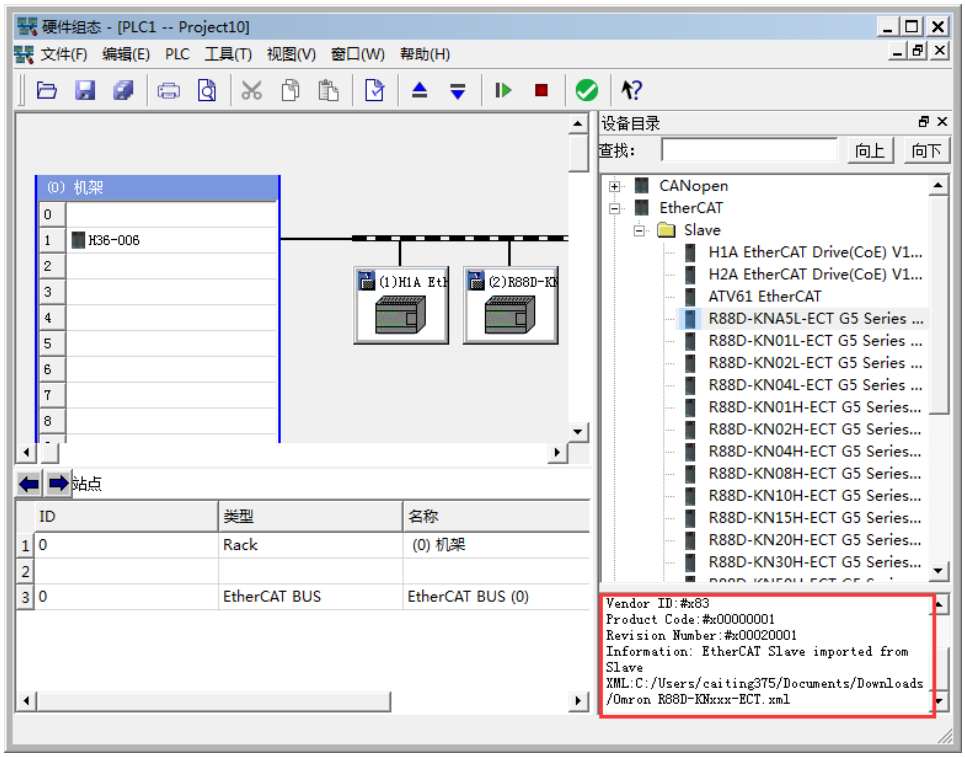


◆ Click "Select File" in the opened window, and select the XML file under the corresponding storage path.

◆ After clicking "OK", the added XML file and its manufacturer will be listed in the previous interface:



◆ After clicking "OK", the relevant servo will be listed in the EtherCAT slave list, and the manufacturer and relevant information of the product will be displayed in the window. Click "Finish" to end the import operation, and then the user can carry out the third-party slave, as shown in the following figure.



Tips

When the system fails, please refer to Chapter 5.8 CPU Fault Diagnosis to obtain the diagnosis method of H56-10 motion controller (EtherCAT register: SMB400~SMB465).

15.1.9 Siemens S7 protocol communication example

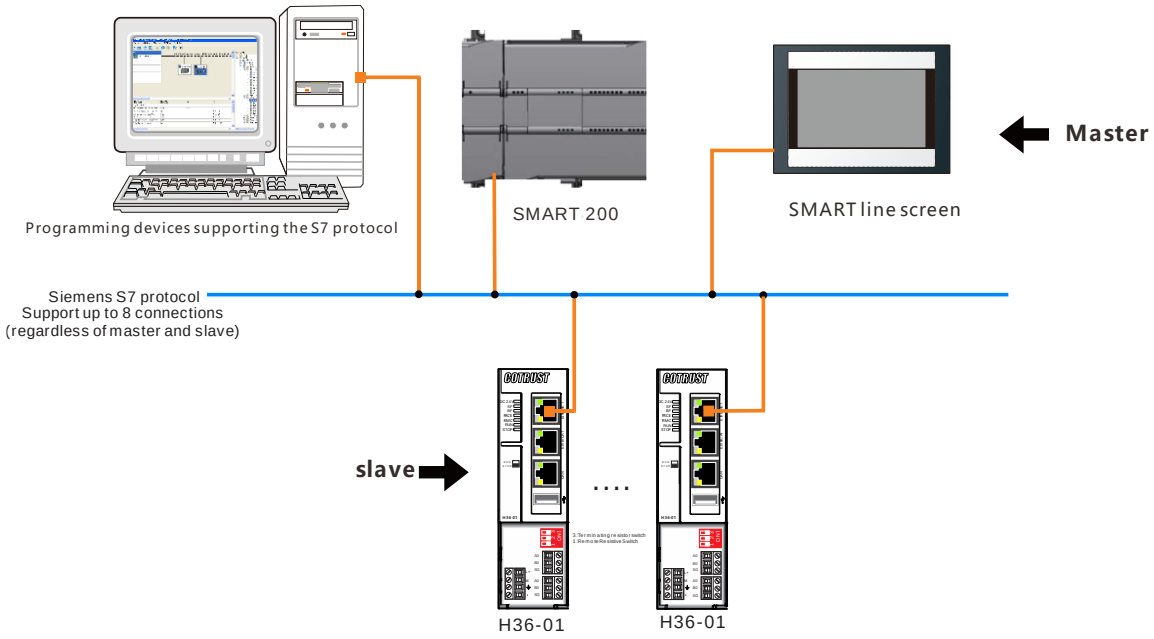
H56-10/H52-10 supports Siemens S7 protocol master- slave function, supports up to 8 connections (regardless of master-slave). Siemens PLC or screen or host computer supporting S7 protocol as the master. H56-10/H52-10 as the slave. In addition, H56-10/H52-10 can also be used as a master. This section will demonstrate the parameter configuration operation of communication between H52-10 and Siemens screen and PLC and third-party screen that supports Siemens S7 protocol and the host computer through examples.

Table 15-22 Components

Components	Features
programming equipment	STEP 7-MicroWin SMART software is installed to configure, program and debug Siemens SMART200 PLC .
COTRUST H52-10	As an S7 protocol slave, it communicates with the S7 protocol master.
Siemens SMART200 PLC	As an S7 protocol master, it communicates with S7 protocol slaves.
Kingview	The software version V6.55, as the S7 protocol master, communicates with the S7 protocol slave.
MCGS screen	As an S7 protocol master, it communicates with S7 protocol slaves.
Siemens screen	SMART 700 IE V3, as S7 protocol master, communicates with S7 protocol slaves.
WeinView screen	As an S7 protocol master, it communicates with S7 protocol slaves

2. Network connection

The following is a typical connection of H52-10 with Siemens and other equipment or host computer software for S7 protocol communication:



15.1.9.1 Ethernet port communication between Siemens products and CTH300-H products

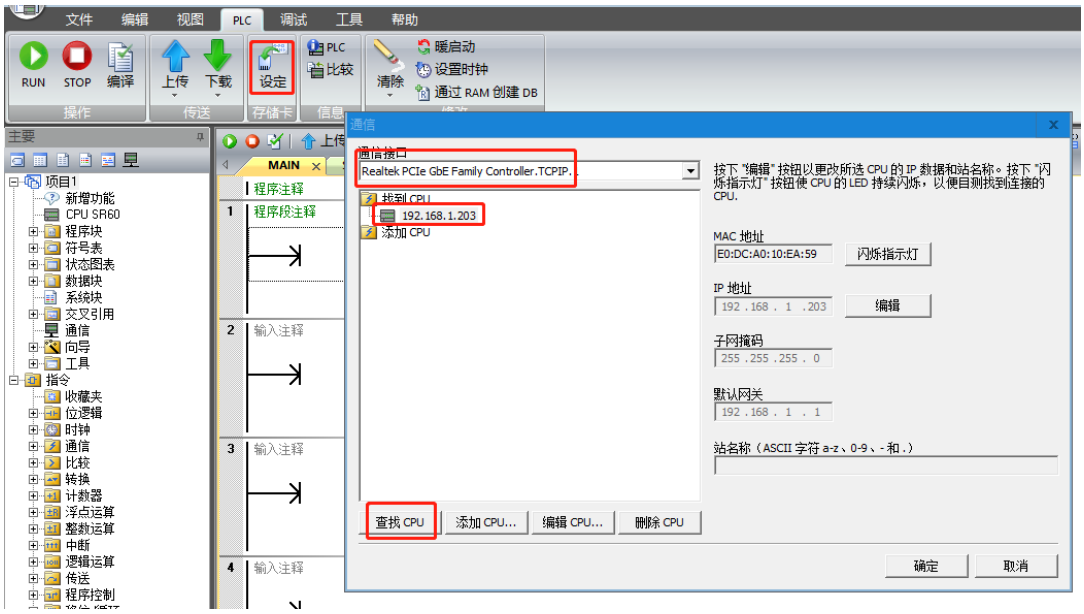
Example 1: S7 protocol communication between SMART200 and H52-10

In this example, the Siemens SMART200 is the master, and the H52-10 is the slave.

<Remarks>: Siemens screen can also be used as the master to communicate with H52-10 in S7 protocol .

1. Communication settings between SMART200 PLC and STEP 7-MicroWin SMART programming software.

Open STEP 7-MicroWin SMART software, connect the SMART200 PLC to the computer with RJ45 network cable (the computer and PLC are kept in the same network segment), open the SMART software setting interface, select the local network corresponding to the computer, and then click “Find CPU” to find PLC.

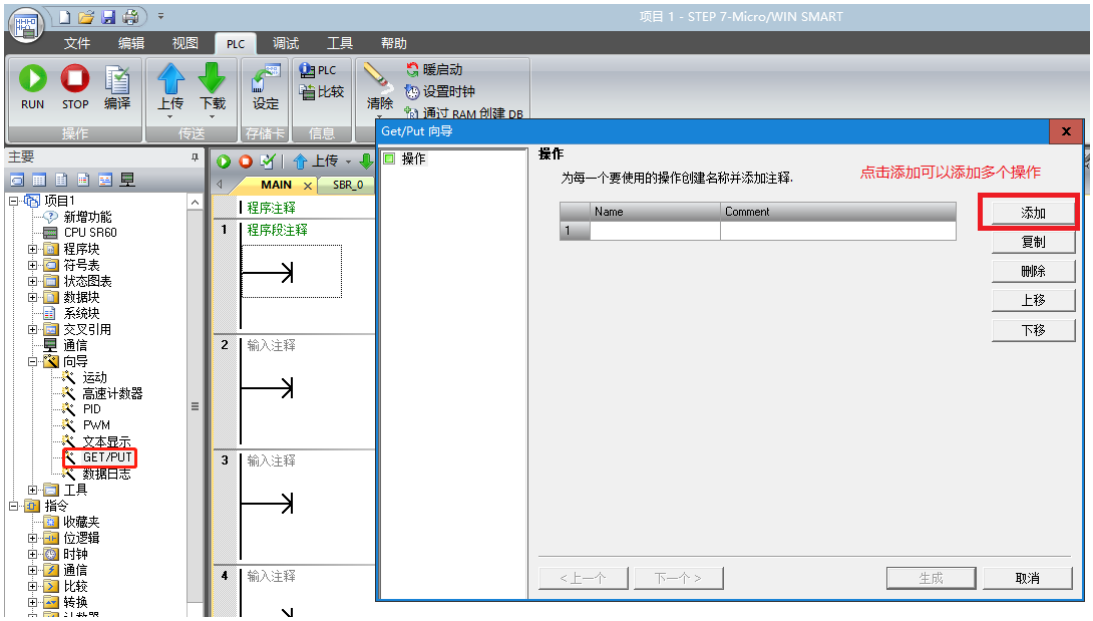


2. SMART200 PLC is the master, and CTH300-H PLC(h52-10) is the slave program setting.

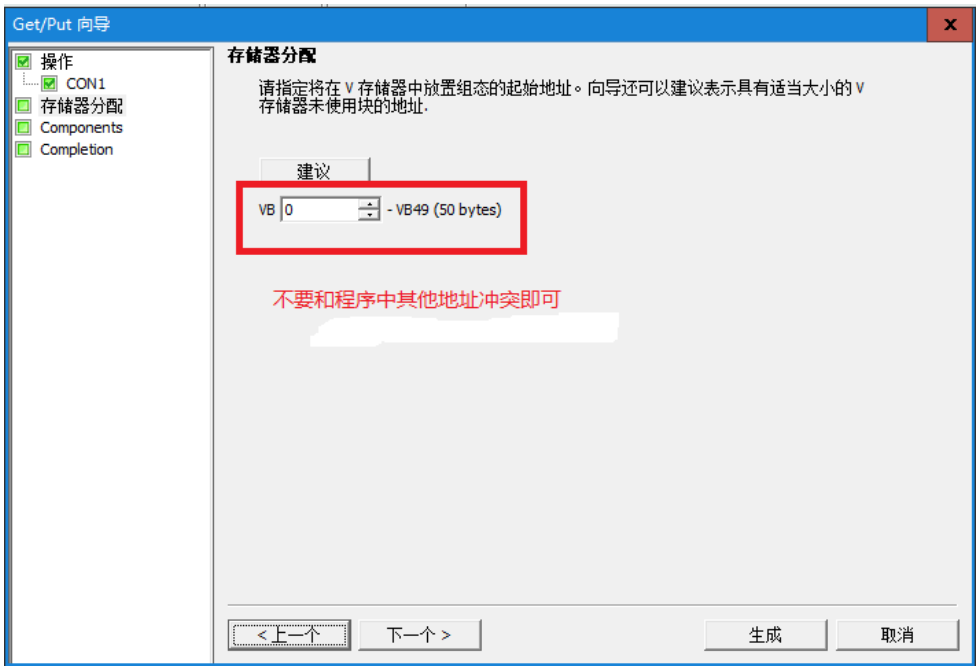
Master configuration

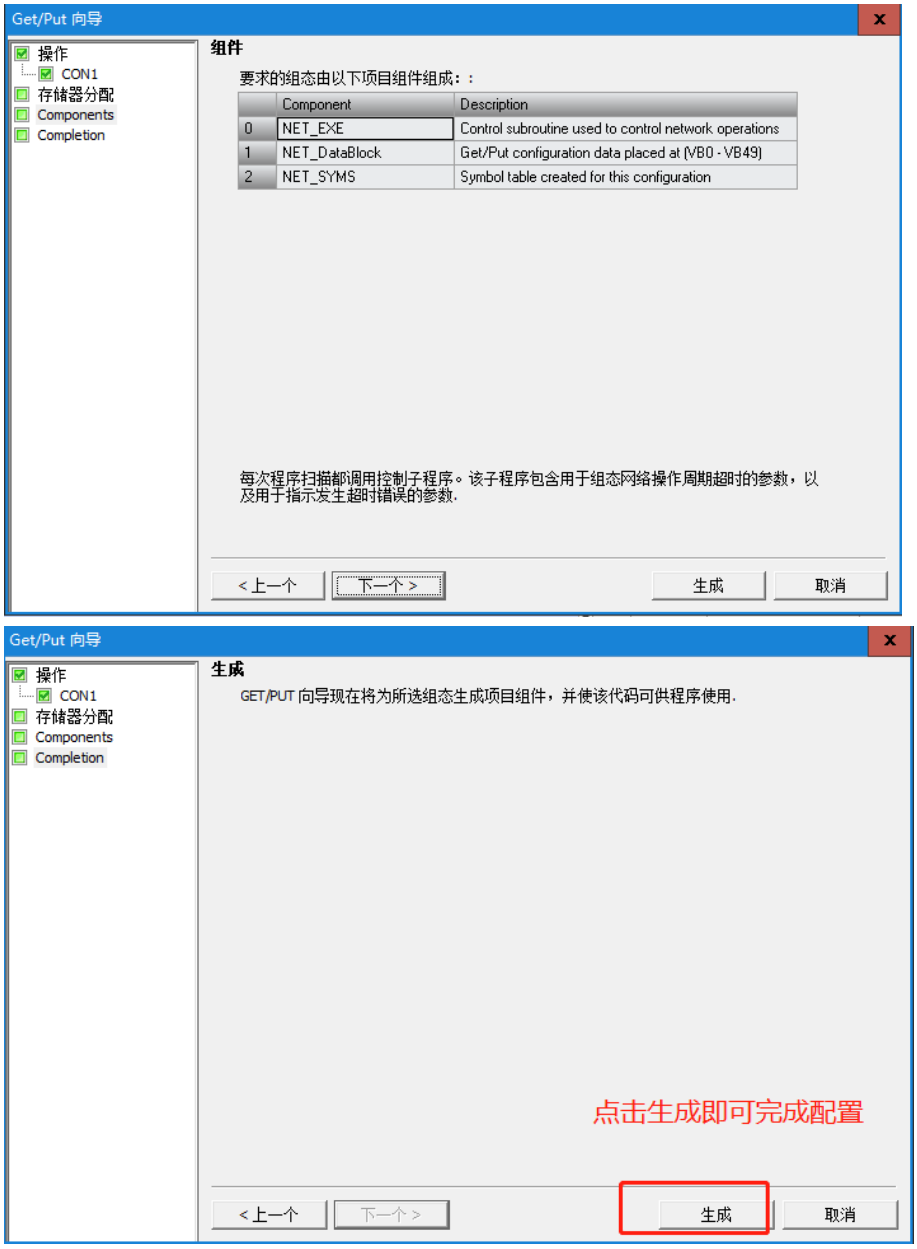
1) Build GET/PUT wizard

Open the GET/PUT wizard in STEP 7-MicroWin SMART software and make the following settings:

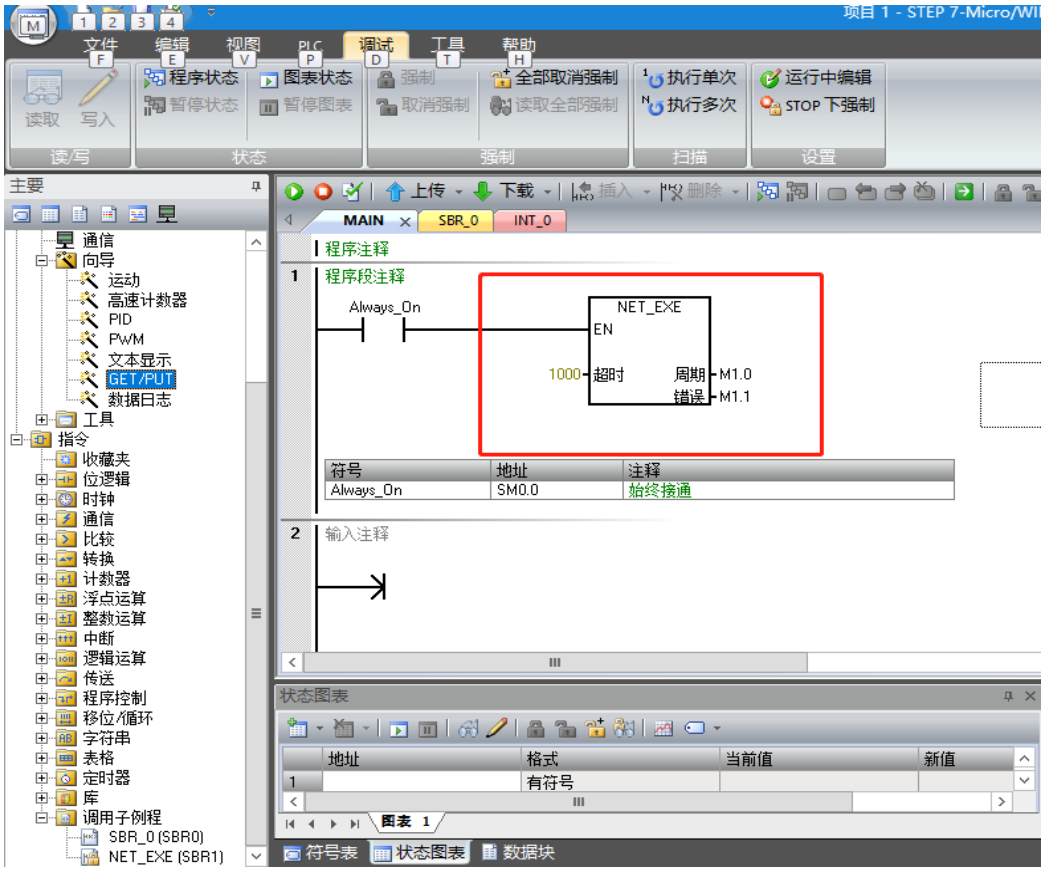


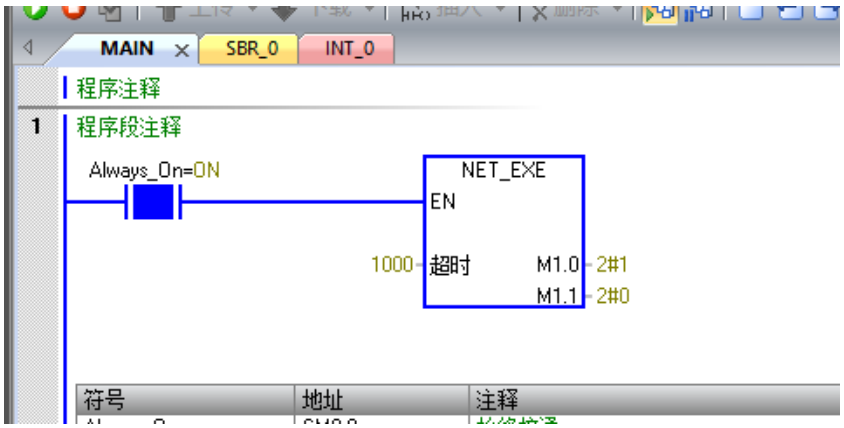

Get/Put 向导
操作
CON1
存储器分配
Components
Completion
CON1
类型
Get
选择发送还是接收
传送大小（字节）
0001
设置传输字节个数
远程PLC的IP地址
本地 CPU
远程 CPU
远程 IP
192 . 168 . 1 . 22
Write to: VB0 - VB0
Read from: VB0 - VB0
本地地址
VB
0000
远程地址
VB
0000
本地数据地址
远程数据地址
< 上一个
下一个 >
生成
取消


Get/Put 向导
操作
CON1
存储器分配
Components
Completion
存储器分配
请指定将在 V 存储器中放置组态的起始地址。向导还可以建议表示具有适当大小的 V 存储器未使用块的地址.
建议
VB 0 - VB49 (50 bytes)
不要和程序中其他地址冲突即可
< 上一个
下一个 >
生成
取消



2) In the SMART software program, use SM0.0 to call the block which generated by the wizard.

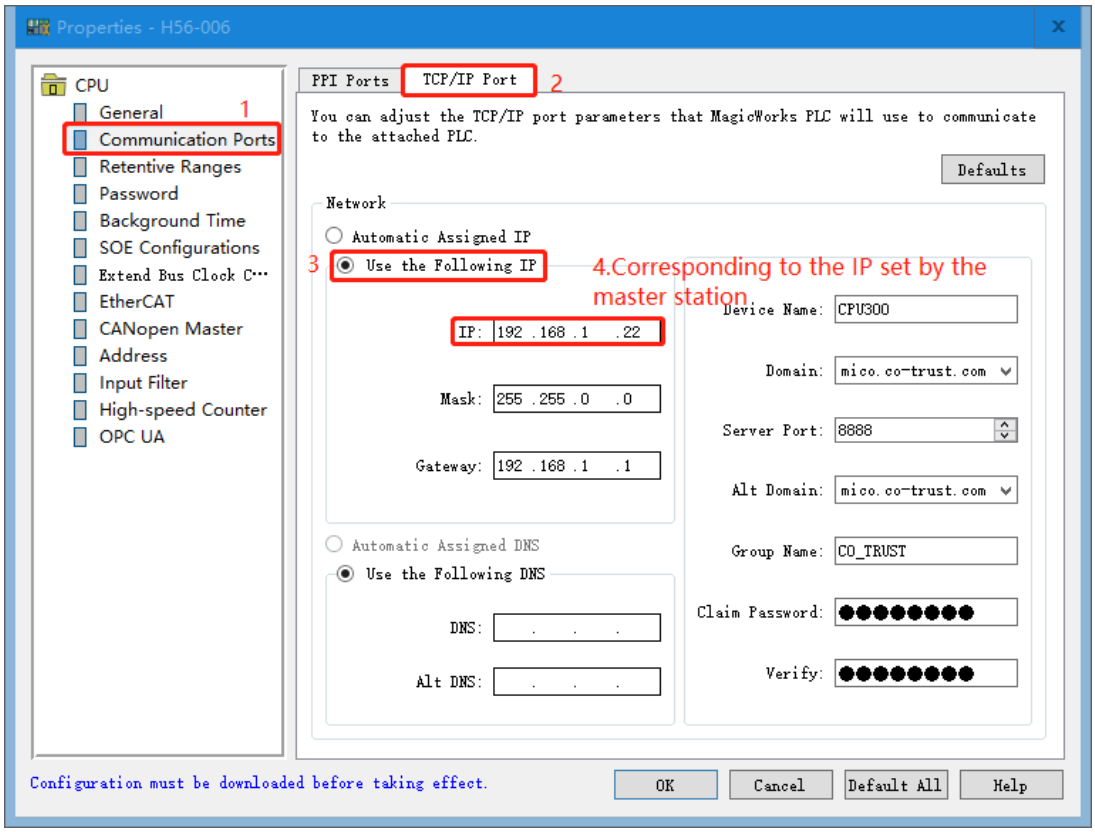




Slave configuration

Open the Magicworks PLC software, configure the slave IP address in the hardware configuration of the CTH300-H PLC. The operation instructions are as follows:

Add an H52-10 CPU to the rack, double-click the module and perform the following operations.



After completing the setting of the S7 protocol master and slave through the above steps, the S7 protocol communication between Siemens SMART200 and H52-10 can be realized.

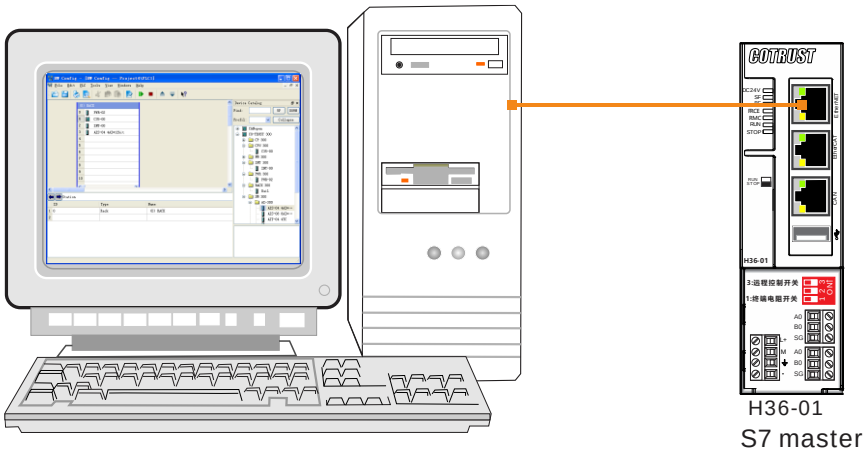
Example 2: S7 protocol communication between CTH300-H CPU and SMART200

This section takes CTH300-H H52-10 as the master and Siemens SMART200 as the slave station, and shows the parameter configuration operation of the communication between the master H52-10 PLC and the slave Siemens SMART200 PLC through specific examples .

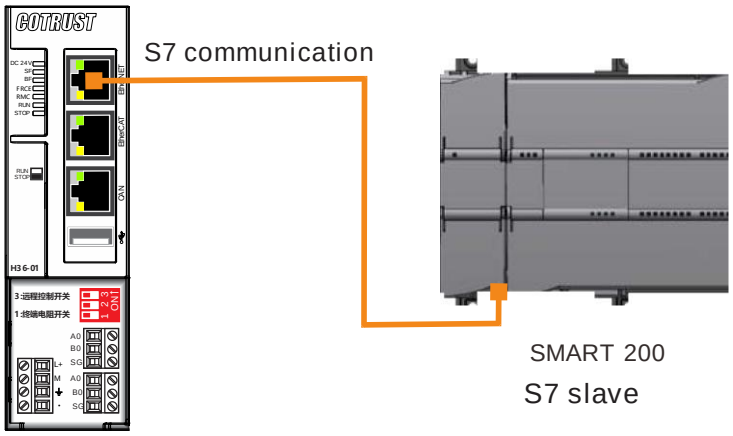
Table 15-23 Components

Components	Features
Programming equipment	Magicworks PLC is installed to configure, program and debug the CTH300-H H52-10.
H 52-10 CPU	As an S7 protocol master, it communicates with S7 protocol slaves.
Siemens SMART200 PLC	As an S7 protocol slave, it communicates with the S7 protocol master.
Standard network cable	• Connect H52-10 PLC and programming device. • Connect H52-10 PLC (S7 master) and Siemens SMART200 PLC (S7 slave).

Use a standard network cable to connect the programming device with the H52-10 PLC, and program the S7 master (H52-10 PLC) through the programming device :

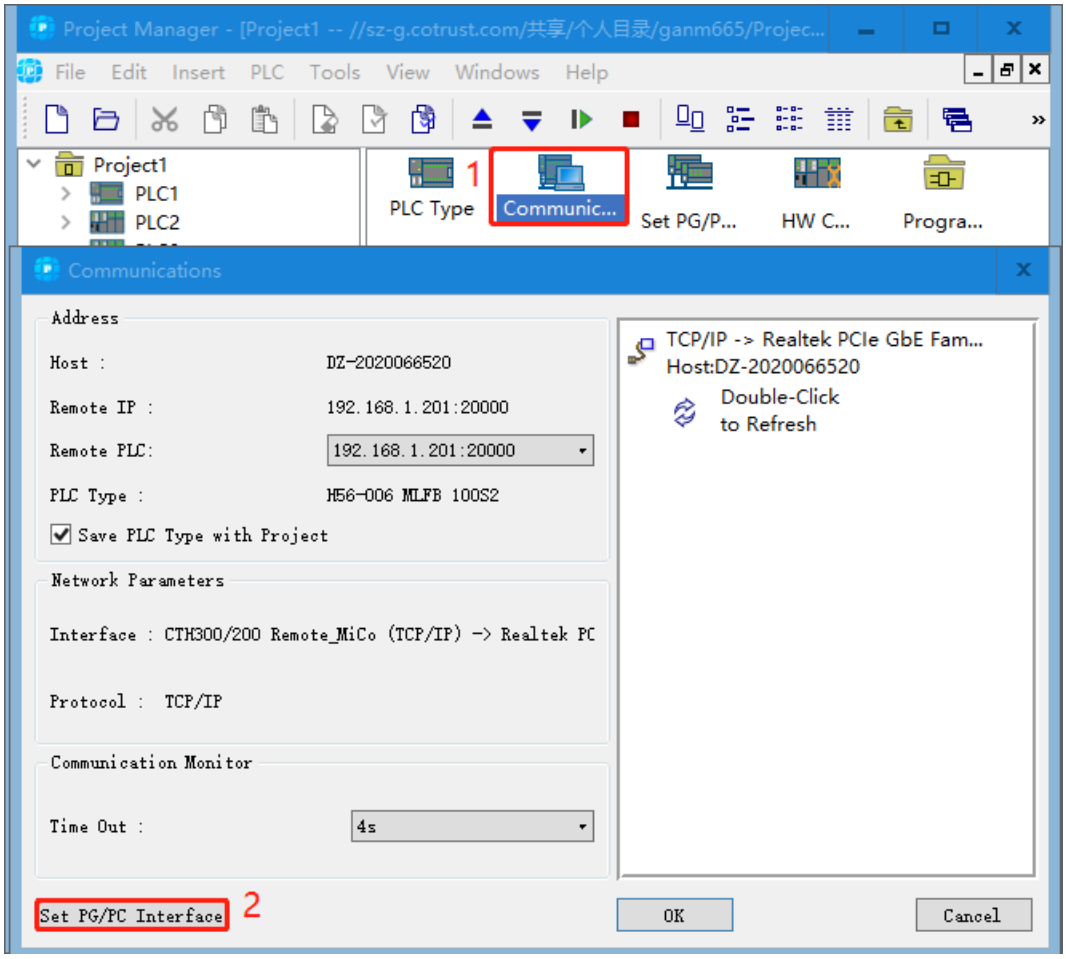


Use a standard network cable to connect the S7 master and the S7 slave, the S7 master will perform a write operation, write the data in the specified address to the S7 slave, and then read the data in the corresponding address through the S7 slave, and then realize S7 communication:

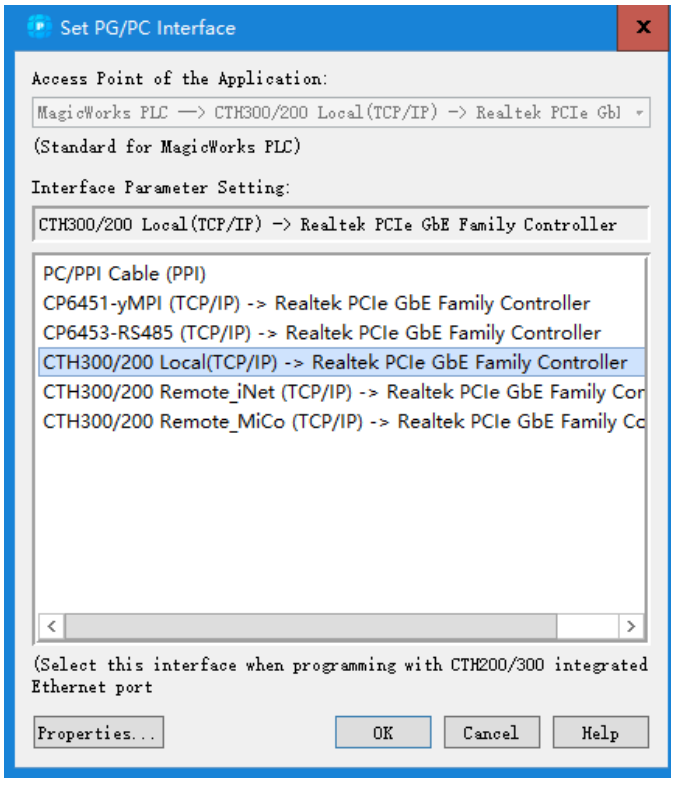


1. Communication settings between the master H52-10 PLC and Magicworks PLC programming software

Open the Magicworks PLC software, connect the H52-10 PLC to the computer with RJ45 network cable, keep the computer and PLC in the same network segment, open the communication interface, and click to set the PG/PC interface.



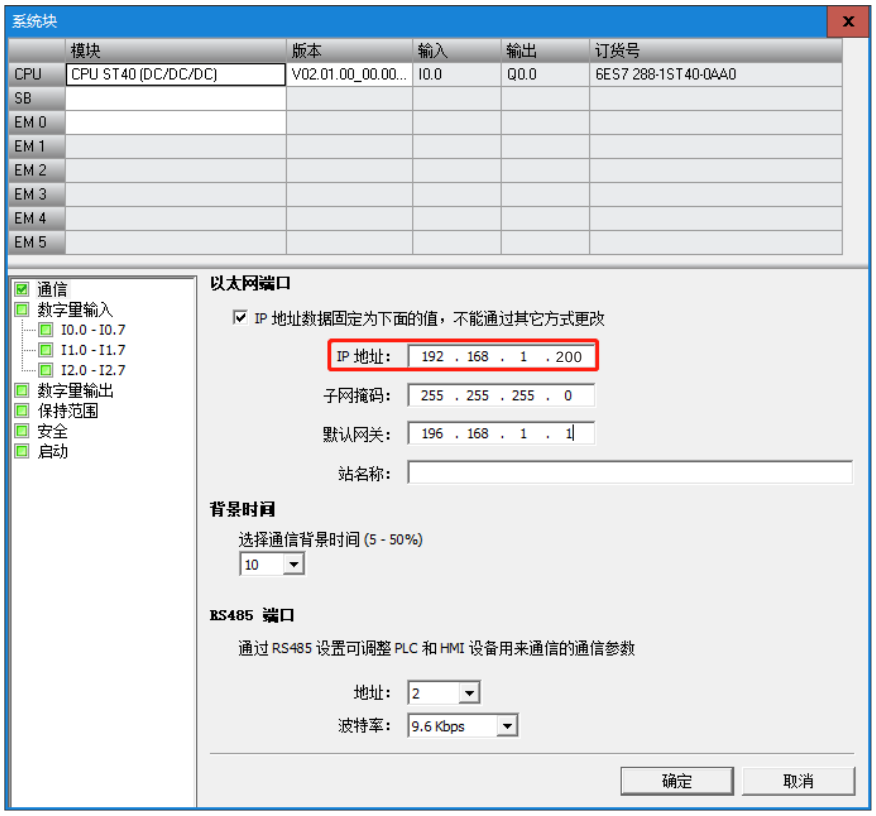
Select "CTH300/200Local (TCP/IP) -> Realtek PLCe GbE Family Controller".



Double-click to refresh on the communication interface, and the CPU H52-10 will appear. Click on the CPU H52-10 to confirm.

Slave configuration

Open STEP7-MicroWin SMART, click "System block" under the project tree, set the slave IP in the system block, and the IP address should correspond to the master.



The S7 master is programmed thou the S7 network read and write commands S7_read/S7_write.

2. Program the S7 master (CPU H52-10)

Taking S7 network read and write as an example , you can perform network read and write operations on the S7 master through the following methods.

master is programmed through the S7 network read and write commands S7_read/S7_write.

Table 15-24 TABLE parameter of the S7_read/S7_write command:

D	A	E	0	error code	0
Remote station IP first byte					1
Remote station IP second byte					2
Remote station IP third byte					3
The fourth byte of the remote station IP					4
reserved (must be set to 0)					5
reserved (must be set to 0)					6
1st byte of the pointer to the remote station target area					7
2nd byte of the pointer to the remote station target area					8
3rd byte of the pointer to the remote station target area					9
4th byte of the pointer to the remote station target area					10
length (not more than 200)					11
1st byte of the pointer to the target area of the local station					12
2nd byte of the pointer to the target area of the local station					13
3rd byte of the pointer to the target area of the local station					14

4th byte of the pointer to the target area of the local station	15

D: Done bit, 0-not done, 1-done

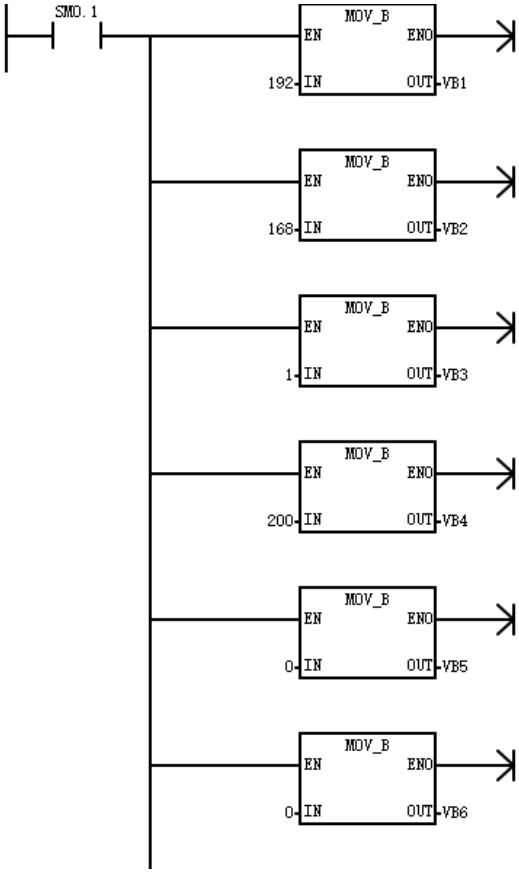
A: Active bit, 1-when the instruction is being executed, 0- it is not executed

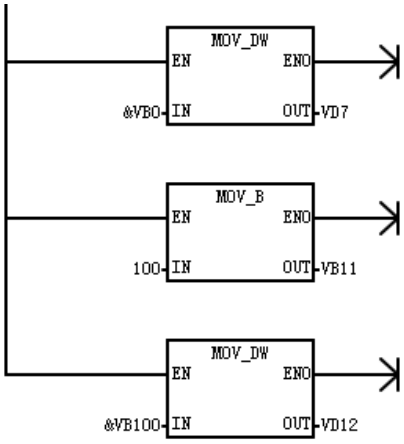
E : bit, 0 no error, 1 error

Table 15-25 Error code table

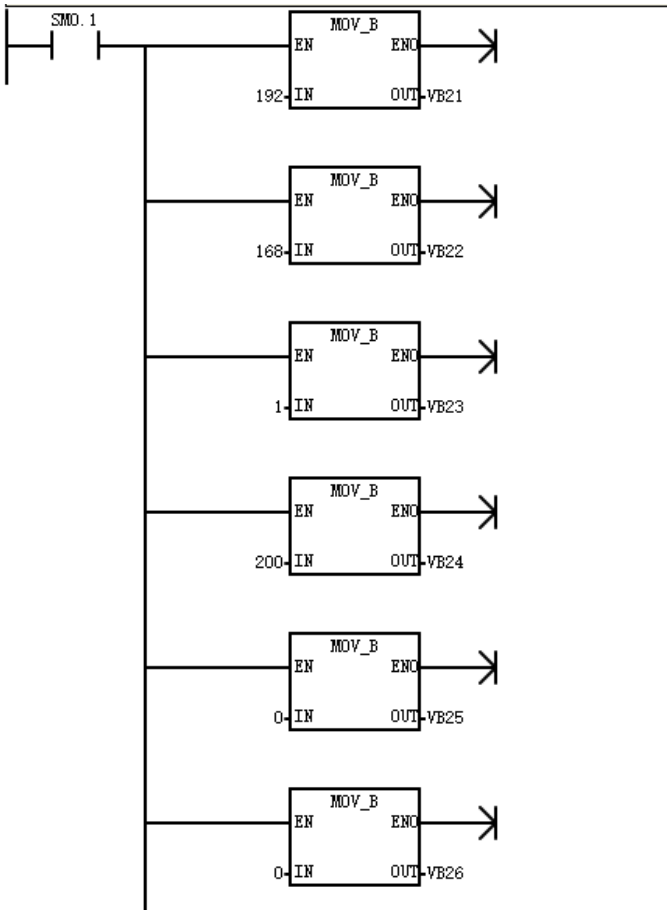
code	Illustrate
0	No error
1	Timeout error: remote station does not answer
2	reserve
3	Offline error: Conflict due to duplicate site or faulty hardware
4	Queue overflow error: 8 S7_write/S7_read blocks activated
5	Receive error: Received reply with error
6	Illegal parameter: The S7_write/S7_read table contains an illegal or invalid value
7	reserve
8	reserve
9	Information error: Incorrect data length
10	More than 8 connections
11	network cable not plugged in

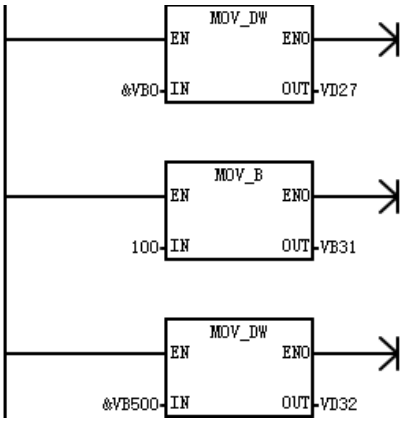
Network 1: According to the TABLE parameter of the S7_read /S7_write command, set the IP of the remote station to 192.168.1.200, set the target area pointer of the remote station to &VB0 (must use a pointer), set the read length to 100 bytes, set the target area pointer of the local station to &VB100 (must use a pointer), and the parameter table is configured.



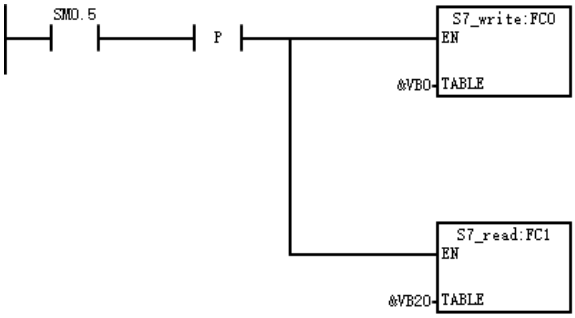


Network 2: Configure the parameter according to the TABLE parameter of the S7_read/S7_write command. Set the IP of the remote station to 192.168.1.200, the target area pointer of the remote station to &VB0 (must use a pointer), the write length to 100 bytes, and the local station target area pointer set to &VB500 (must use a pointer).





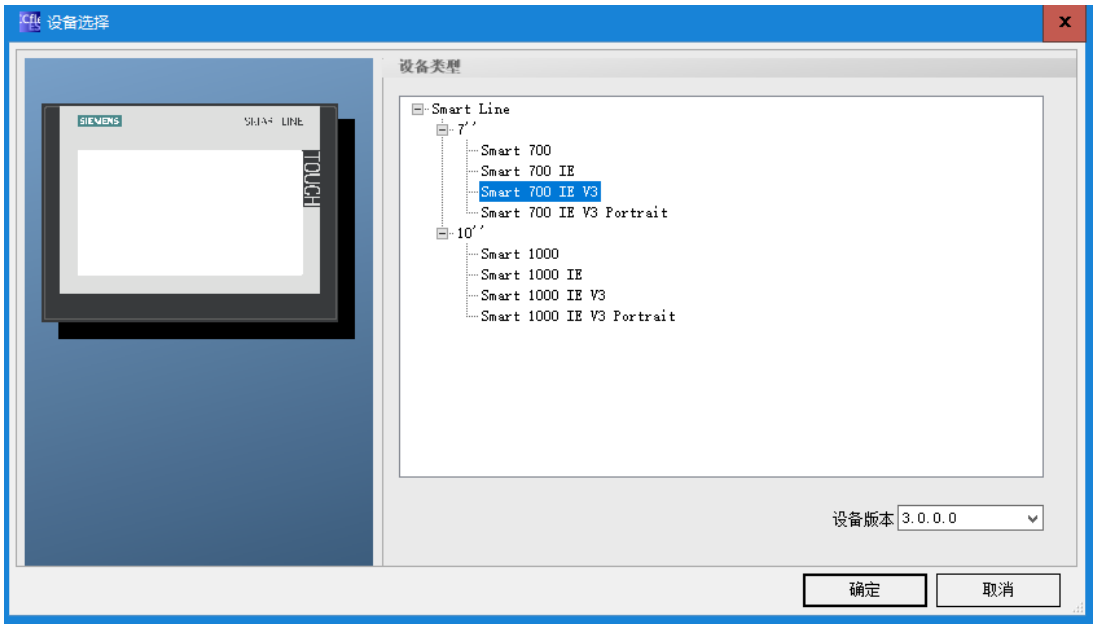
Network 3: Call the S7_read/S7_write command to read and write data through the parameters of the parameter table. The TABLE parameter of the command must use a pointer.

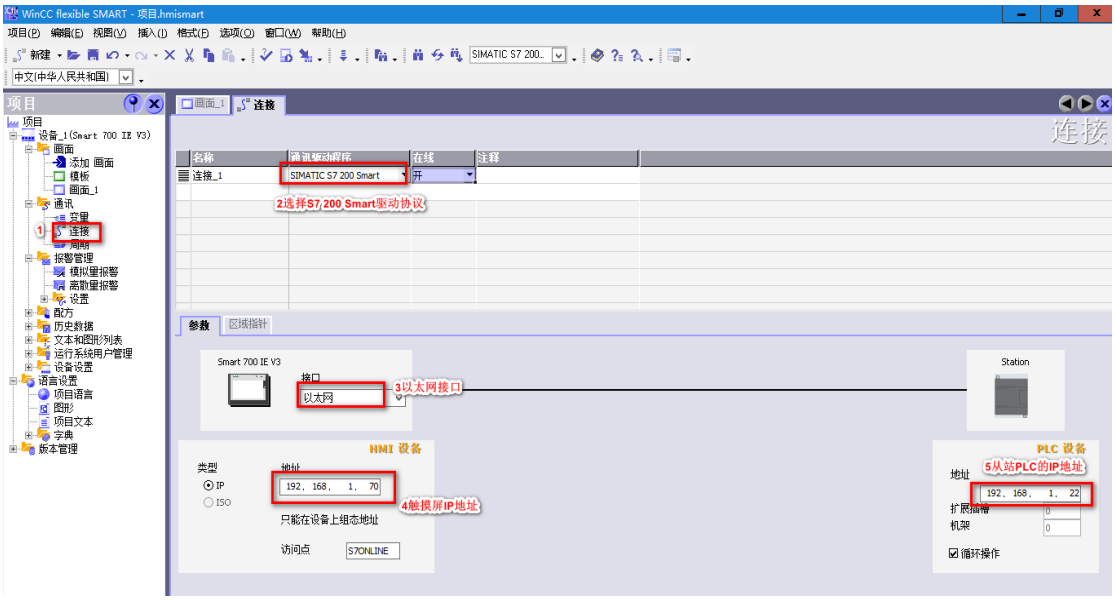


Example 3:Communication between Siemens touch screen and H52-10

In this example, the Siemens screen is the master, and the CTH300-H PLC is the slave to communicate with the S7 protocol.

Master configuration

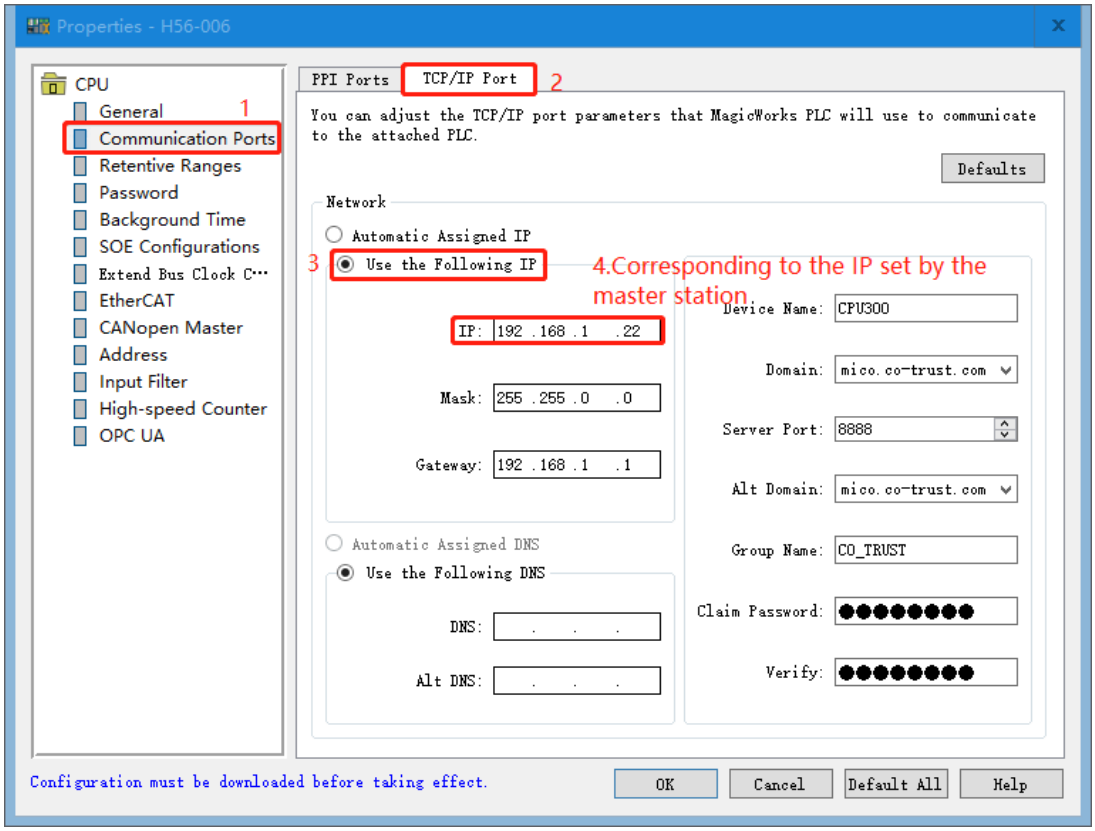




Slave configuration

configure the slave IP address in the hardware configuration. The operation instructions are as follows:

Add an H52-10 CPU to the rack, double-click the module and perform the following operations.



Example 4: S7 protocol communication between S7-1200/1500 and CTH300-H PLC

S7 protocol communication between S7-1200/1500 CPU and CTH300-H PLC (S7-1200/1500 as the master), COTRUST CTH300-H PLC (H52-10) supports Siemens S7 protocol slave function, Siemens PLC Or screen or the host computer software supporting S7 protocol as the master, CTH300-H PLC (H52-10) as the slave station, can carry out S7 protocol communication.

Hardware and software requirements and communication tasks performed	
hardware	● S 7-1200 CPU hardware version V2.0 or higher ● H52-10 ● PC (with Ethernet card) ● TP ethernet cable ● Industrial switch
software	● TIA portal V15 ● MagicWorks PLC

Communication tasks:

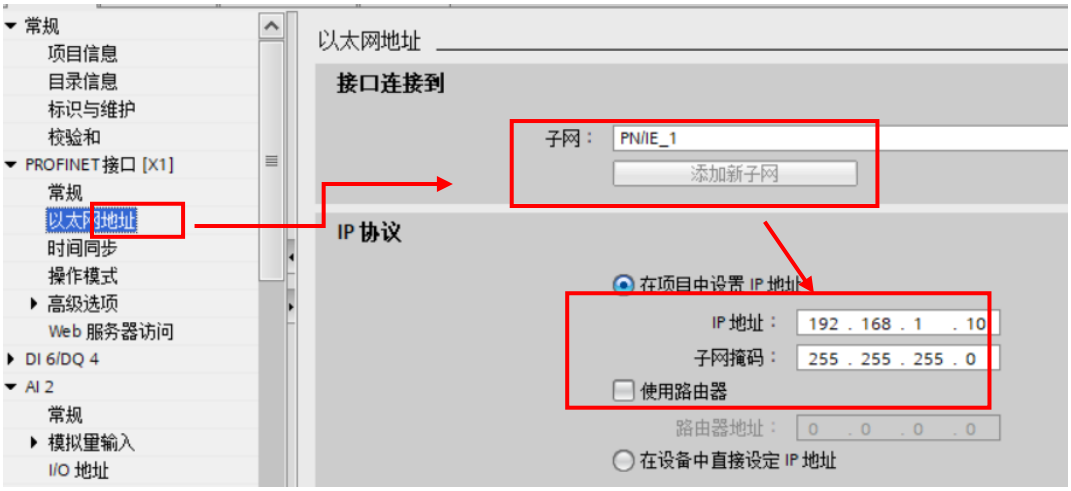
The S7-1200 sends 100 bytes in the communication data area DB1 to the VB data area of the H52-10, and the S7-1200 reads the VB data area in the H52-10 and stores it in the data area DB1 data area of the S7-1200.

MB0-MB9 in the M area of S7-1200 sends 10 consecutive bytes to MB0-MB9 in the M area of H52-10, S7-1200 reads MB10-MB19 in the M area of H52-10 and sends 10 consecutive bytes to the S7 -1200 MB10-MB19 in the M area.

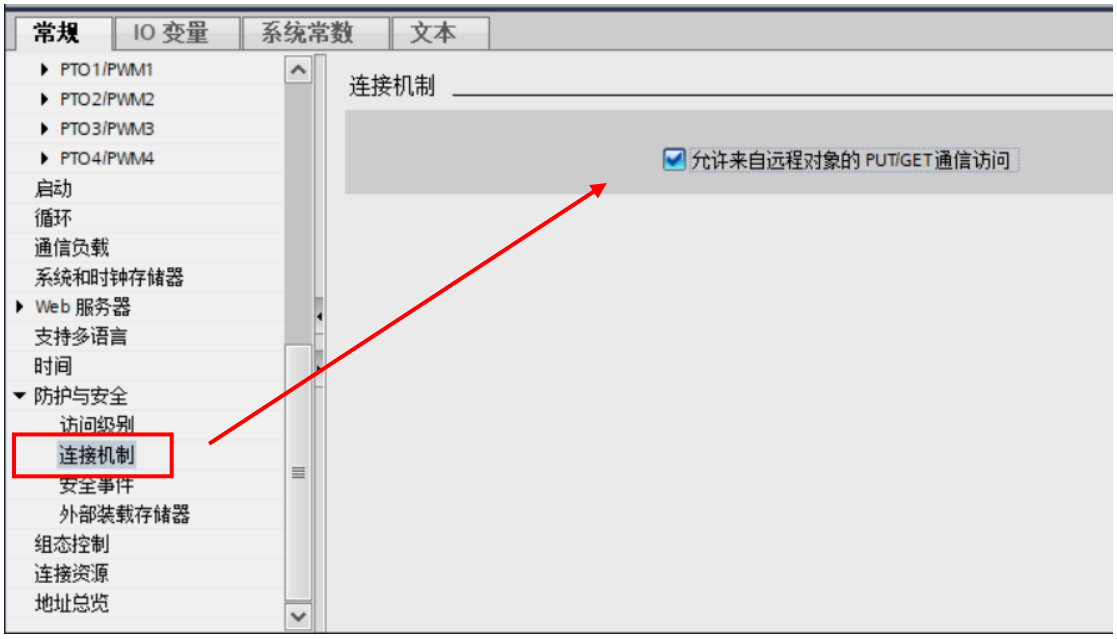
Configuration programming of the S7-1200 master

Step 1: Use TIA portal V15 software to create a new project and complete hardware configuration and network configuration

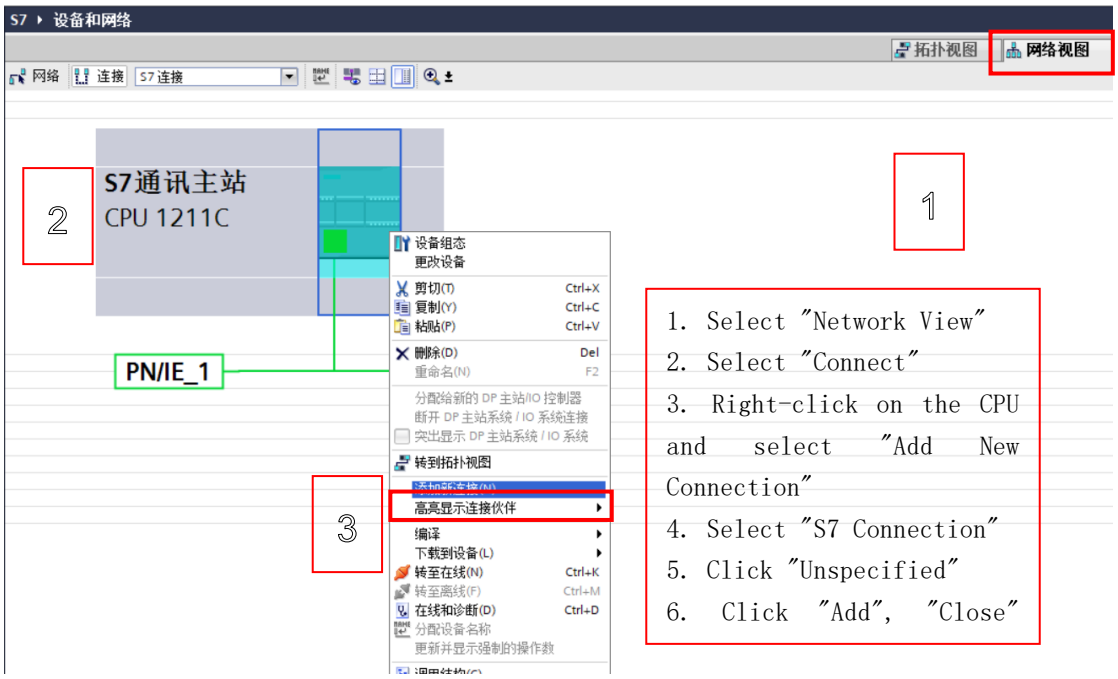
1) Create a new subnet PN/IE_1, and modify the IP address to be in the same network segment as PC and H52-10

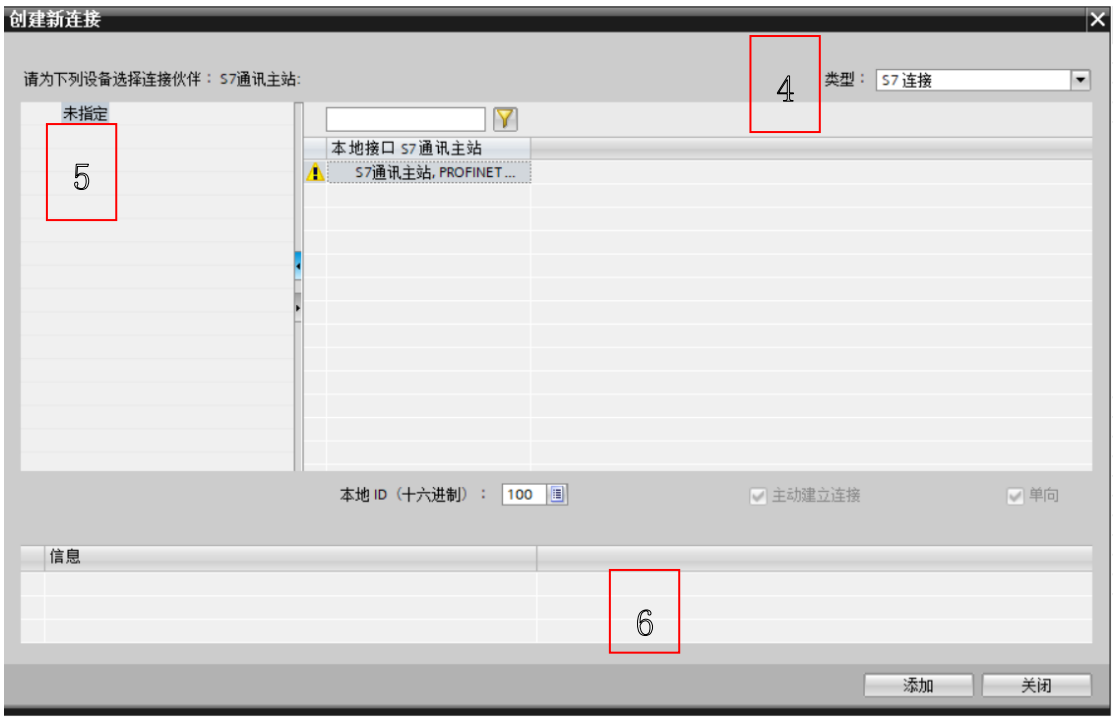


2) Check "Protection and Security" → "Connection Mechanism" → "Allow PUT/GET Communication Access from Remote Objects"

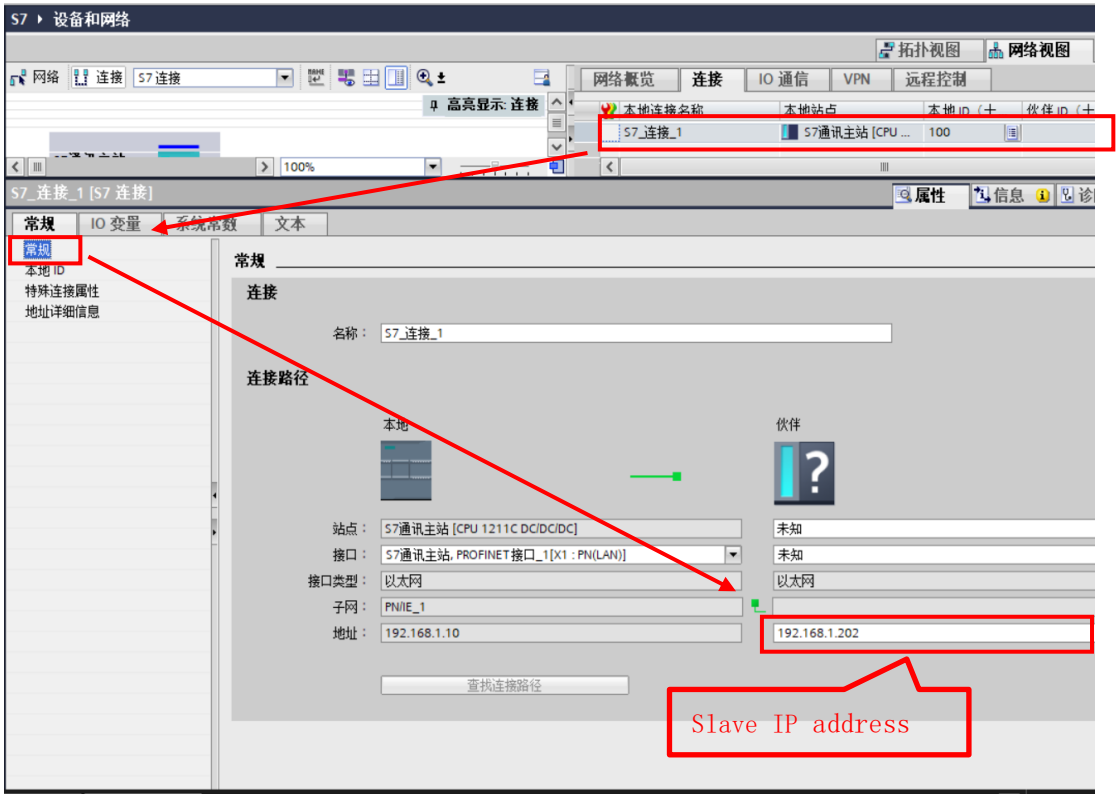


3) Under "Project Tree"→"Device Configuration"→"Network View", follow the steps 1~6 in the following picture to establish the S7 connection

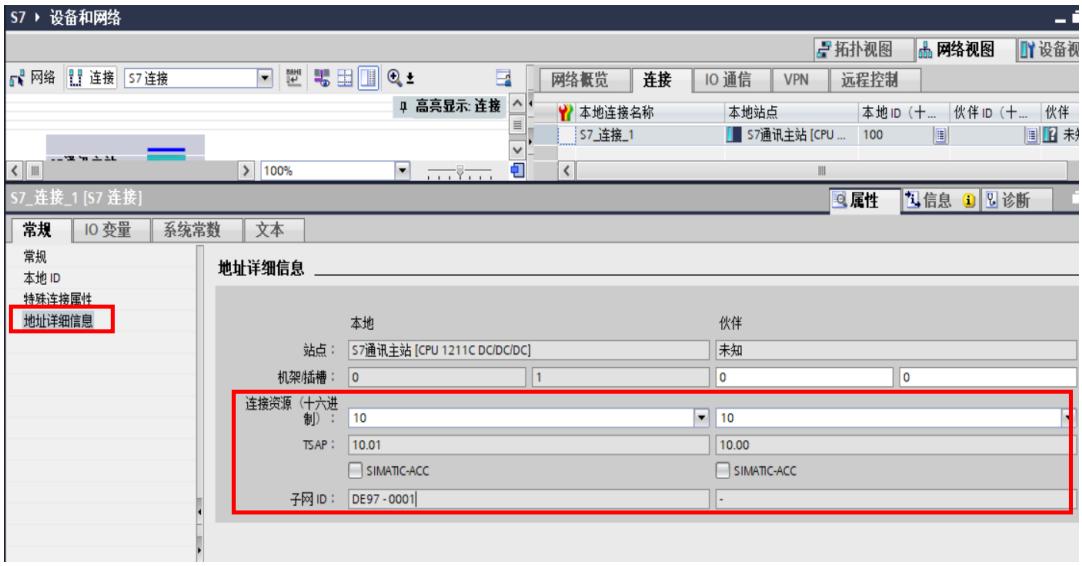




4) Fill in the connection parameters according to the steps in the picture below

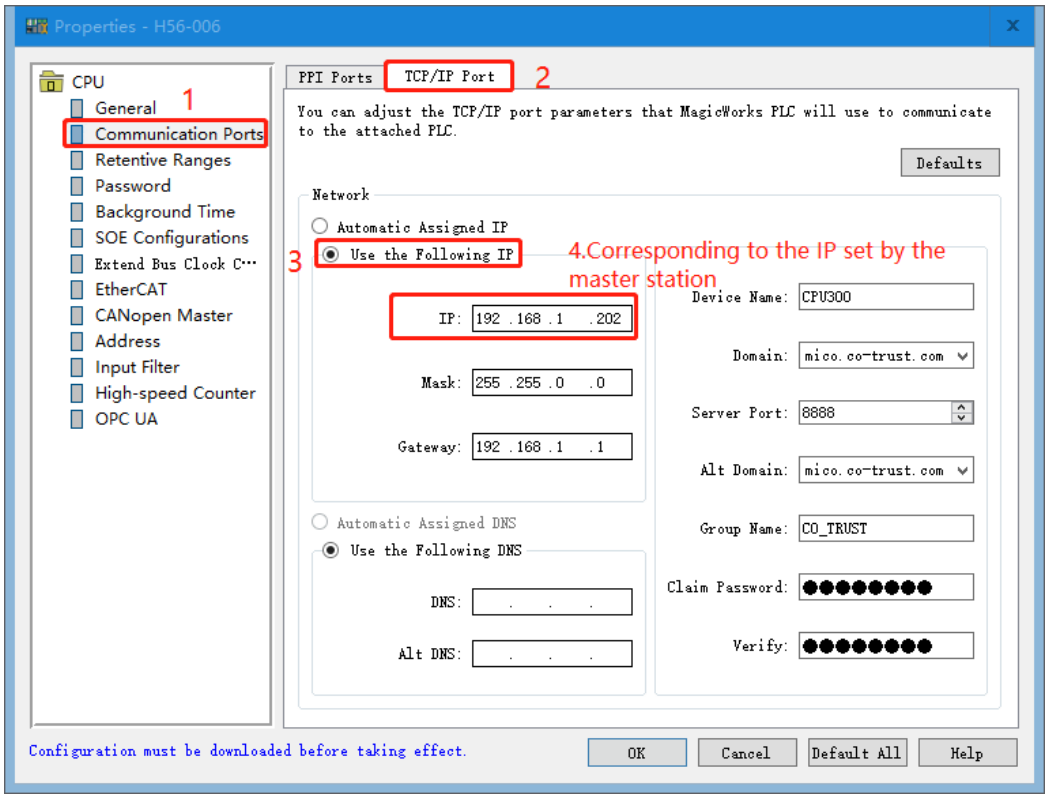


5) Set the TSAP address of the in "Address Details"



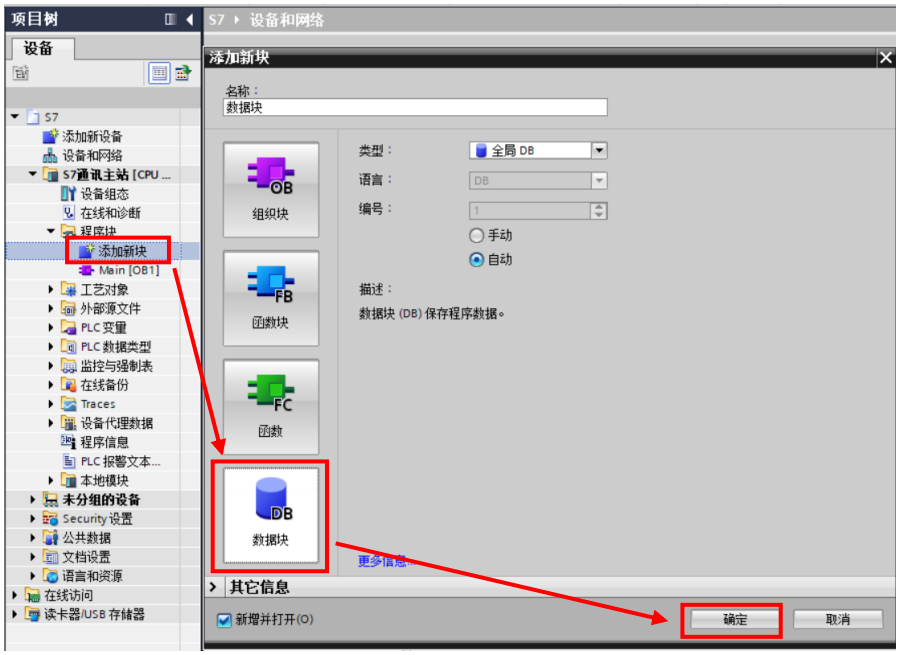
6) Configure the slave IP address in the hardware configuration of the CTH300-H PLC. The operation instructions are as follows:

Add a H52-10 CPU to the rack, and double-click the module to perform an operation.

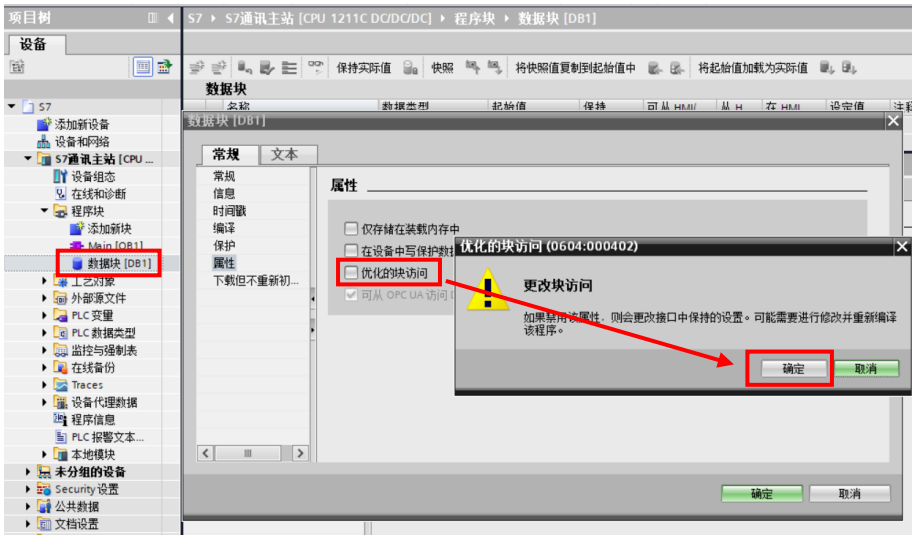


Step 2: Software Programming

1) Create a data block DB1 and define two groups as arrays of 100 bytes

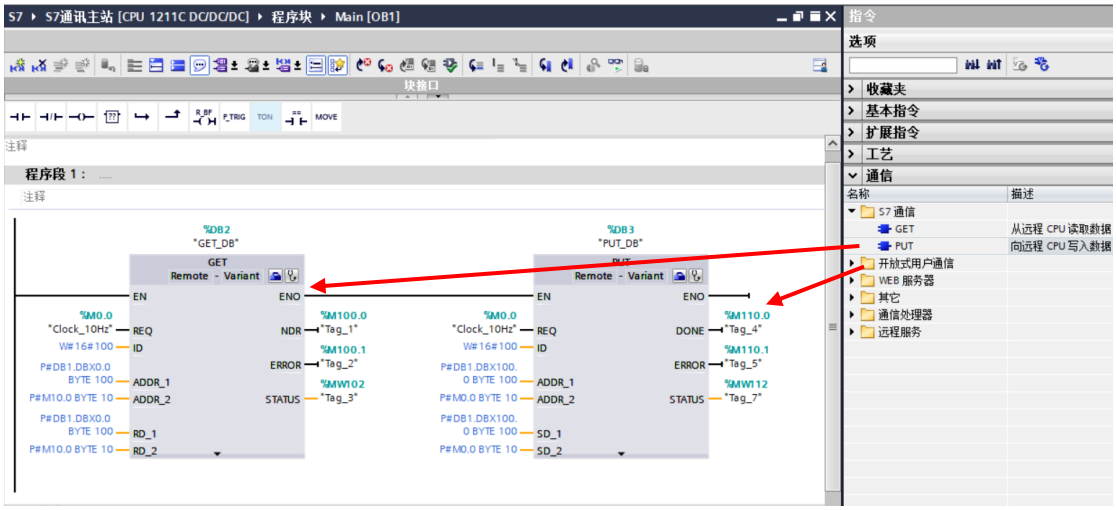


2) Under "Project Tree"→ "Data Blocks" → "Properties", uncheck Optimized Block Access.



3) Create buffers for write data and read data of data block DB1, and define two groups as arrays of 100 bytes

4) In OB1, from "Instructions"→"S7 Communication", call the Get and Put communication instructions. The program is written as follows:



CALL “GET” , %DB2 // call PUT, use DB block: DB2

REQ : =%M0.0 // system clock 0.1s pulse

ID : =W#16#0100 // the connection number when the connection is created

DONE : =%M100.0 // When it is 1, the sending is completed

ERROR : =%M100.1 // When it is 1, there is a fault

STATUS : =%MW102 // Status code

ADDR_1 : =P#DB1.DBX0.0 BYTE 100 //Address to read data from communication partner data area

ADDR_2 : =P#M10.0 BYTE 10 //Address to read data from communication partner data area

SD_1 : =P#DB1.DBX0.0 BYTE 100 // Local send data area

SD_2 : = P#M10.0 BYTE 10 // Local send data area

CALL “PUT” , %DB3 // call PUT, use DB block: DB2

REQ : =%M0.0 // system clock 0.1s pulse

ID : =W#16#0100 // The connection number when the connection was created

NDR : =%M110.0 // When it is 1, there is a fault

ERROR : =%M110.1 // When it is 1, there is a fault

STATUS : =%MW112 // Status code

ADDR_1 : =P#DB1.DBX100.0 BYTE 100 // Address sent to the data area of the communication partner

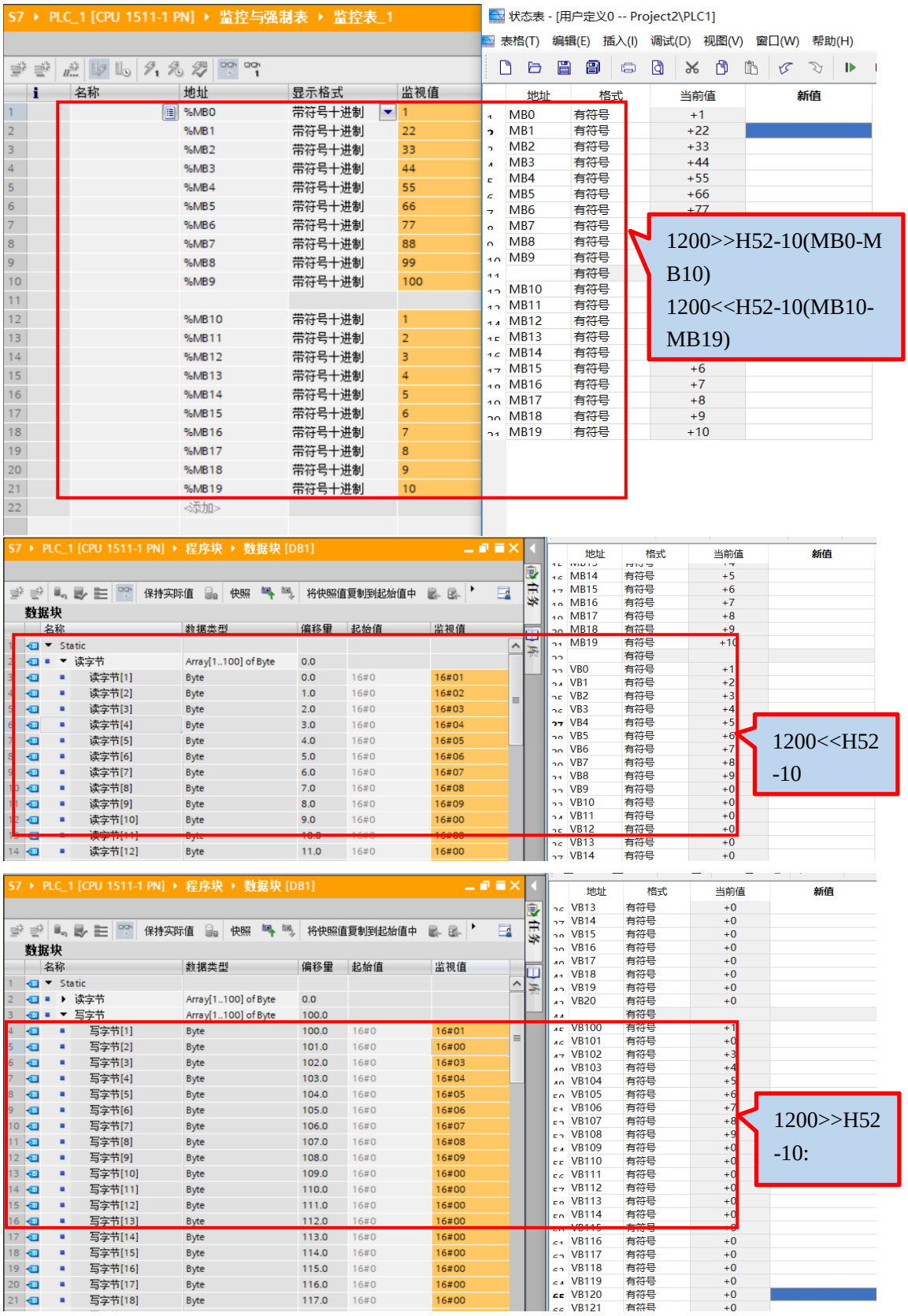
ADDR_1 : =P#M0.0 BYTE 10 // Address sent to the data area of the communication partner

RD_1 : =P#DB1.DBX100.0 BYTE 100 //Local receive data area

RD_1 : =P#M0.0 BYTE 10 // Local receive data area

Step 3: Monitor Results

Through S7 communication by programming on the S7-1200, data exchange between the two CPUs is realized, and the monitoring results are as follows:



Note: V area in H52-10 corresponds to DB1, that is, the communication partner data area ADDR_1=P#DB1.DBX0.0 BYTE 100 used in PUT instruction corresponds to VB0~VB99 in

H52-10. That is, the communication partner data area ADDR_1=P#DB1.DBX100.0 BYTE 100 used in the GET command corresponds to VB100~VB199 in H52-10.

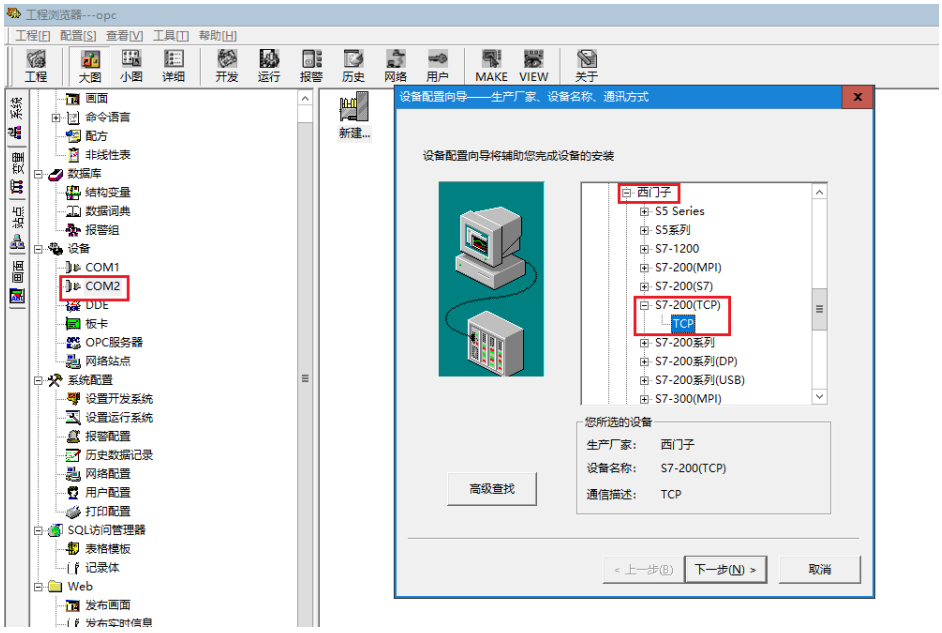
15.1.9.2 Network port communication between other products and CTH300-H PLC

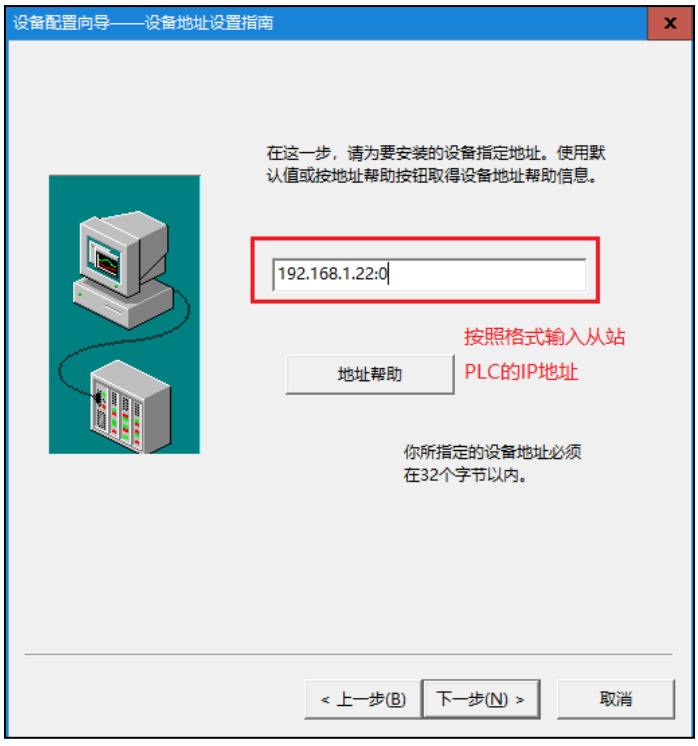
Example 1:S7 protocol communication between Kingview and CTH300-H PLC

In this example, Kingview is used as the master, and CTH300-H PLC is the slave to communicate with S7 protocol.

Master configuration

Select the S7 communication protocol in the serial device configuration wizard of the Kingview software :

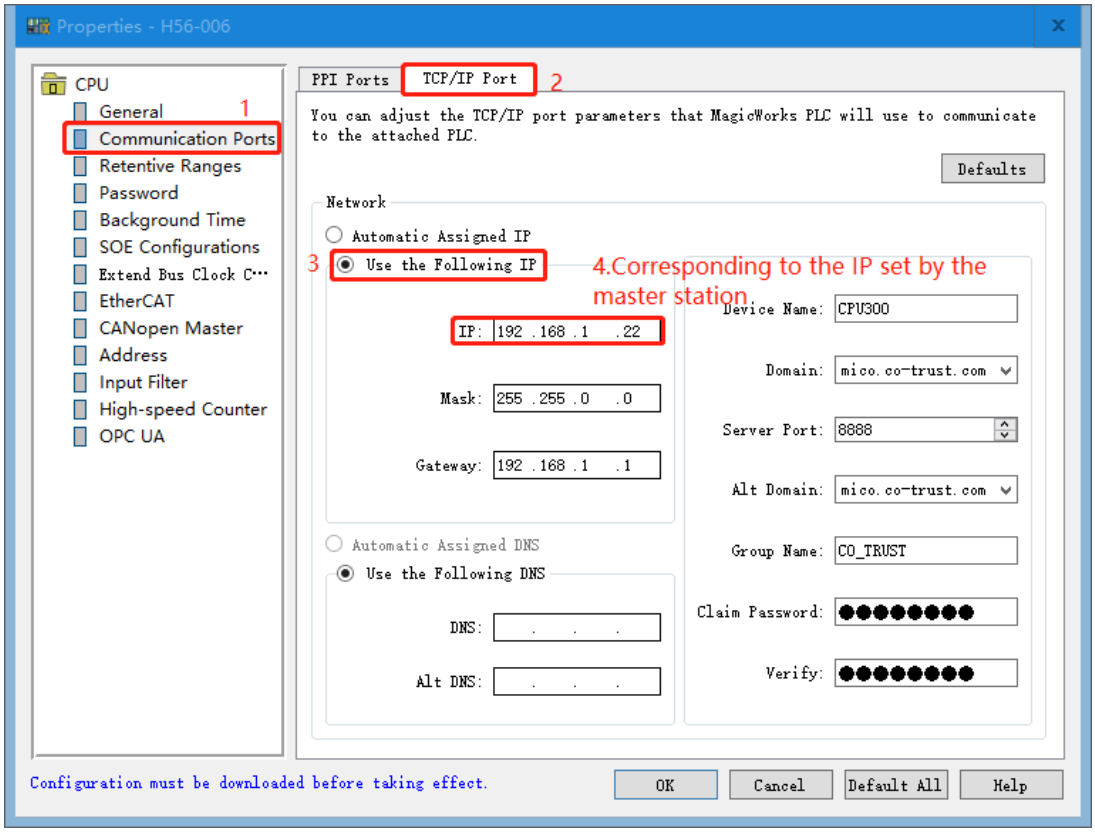




Slave configuration

Configure the slave IP address in the hardware configuration of the CTH300-H PLC. The operation instructions are as follows:

Add an H52-10 CPU to the rack, double-click the module and perform the following operations.

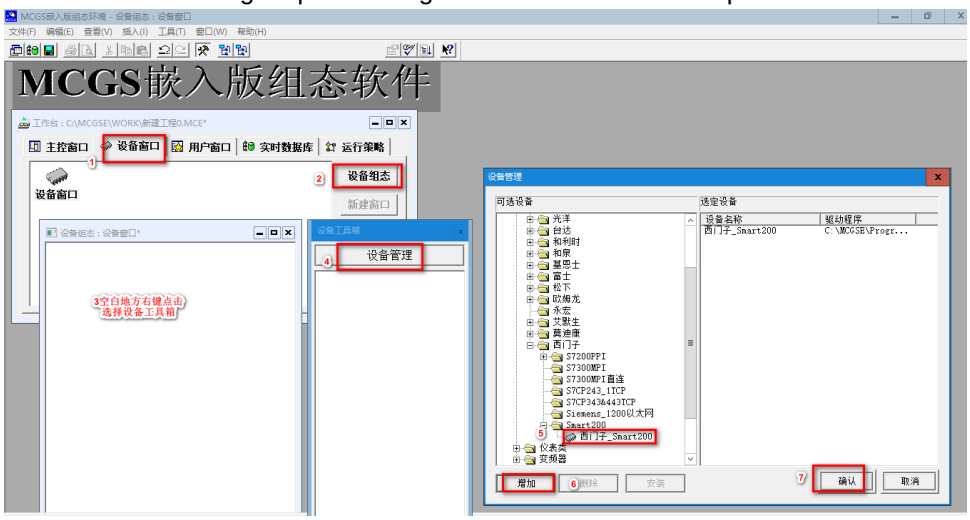


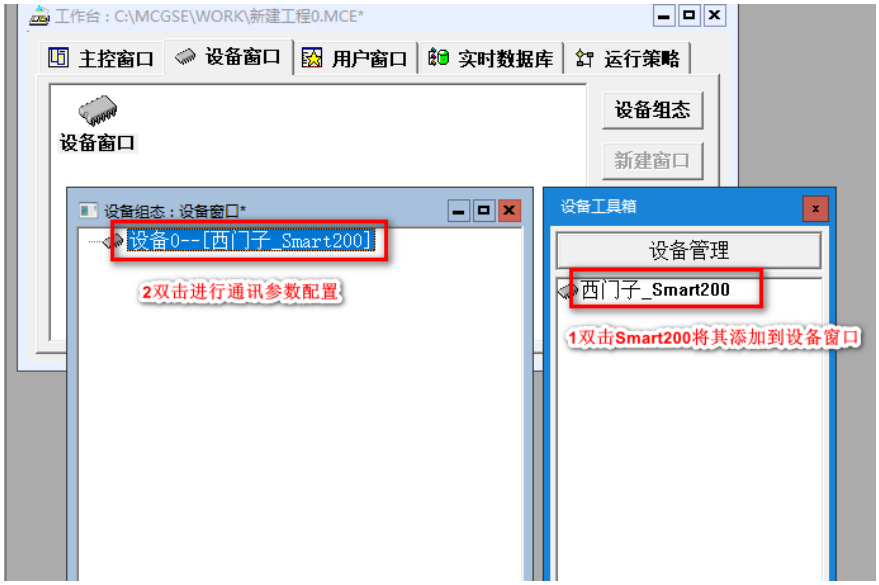
Example 2: S7 protocol communication between MCGS screen and CTH300-H PLC

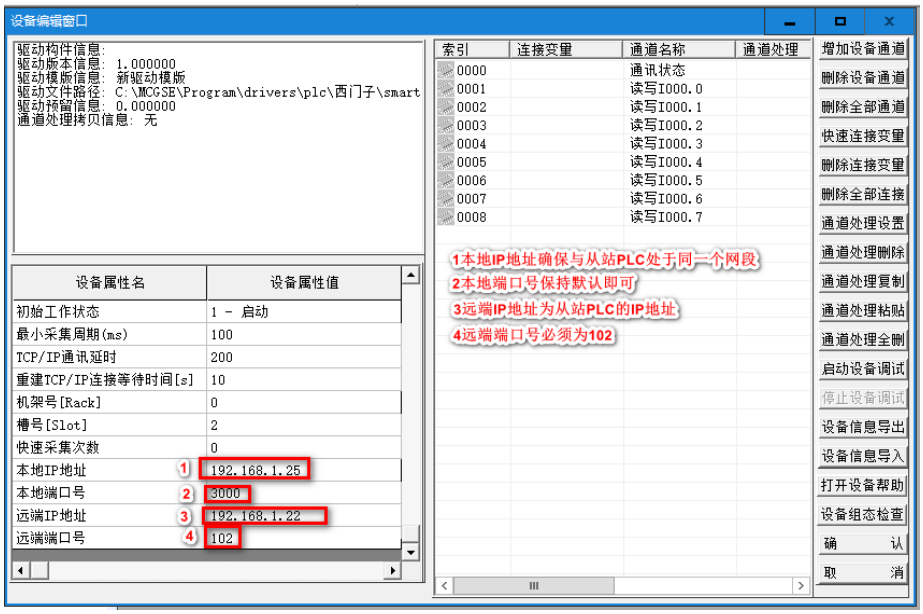
In this example, the MCGS screen is the master, and the CTH300-H PLC is the slave for S7 protocol communication.

Master configuration

Refer to the following steps to configure the master of the S7 protocol for the MCGS screen:



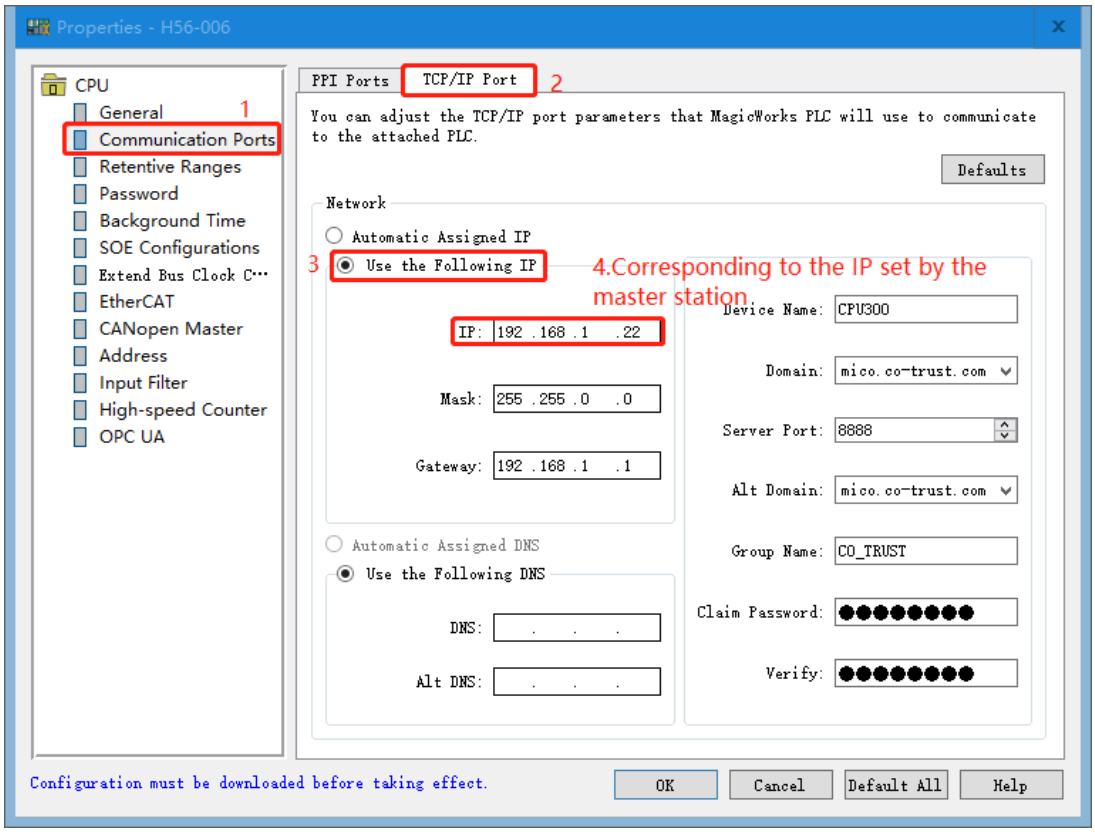




Slave configuration

configure the slave IP address in the hardware configuration of the CTH300-H PLC. The operation instructions are as follows:

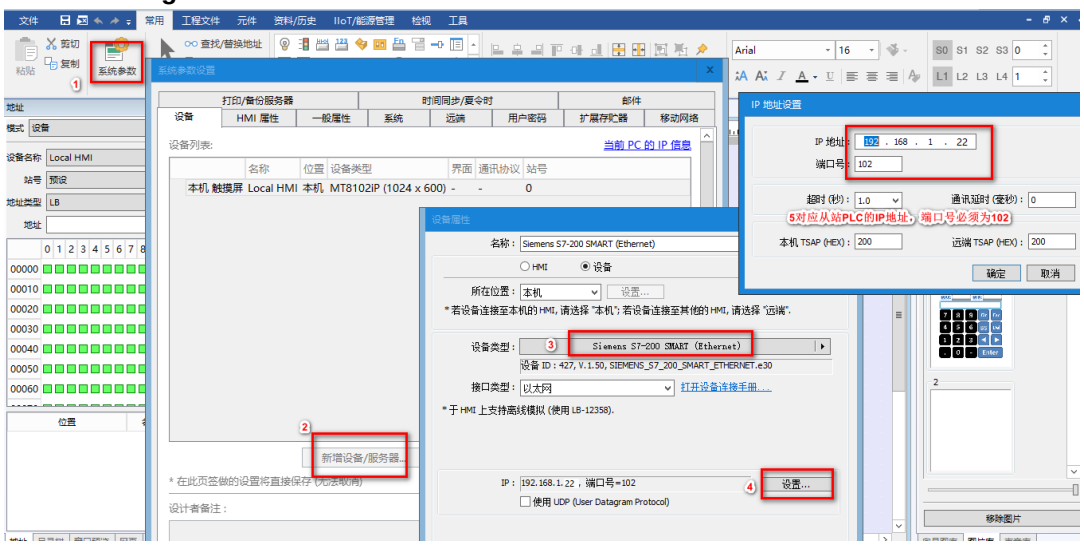
Add an H52-10 CPU to the rack, double-click the module and perform the following operations.



Example 3: S7 protocol communication between WeinView screen and CTH300-H PLC

In this example, the WeinView screen is the master, and the CTH300-H PLC is the slave to communicate with the S7 protocol.

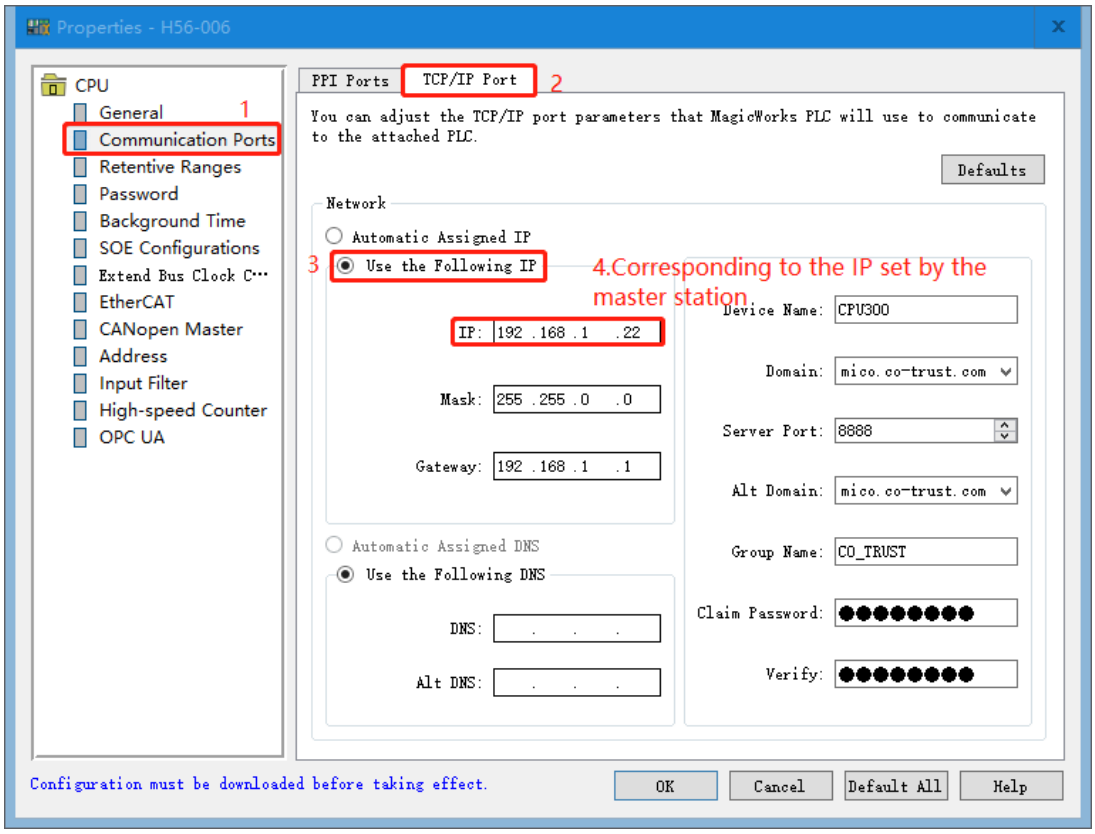
Master configuration



Slave configuration

Configure the slave IP address in the hardware configuration of the CTH300-H PLC. The operation instructions are as follows:

Add a H52-10 CPU to the rack, and double-click the module to perform an operation.



15.2 High-speed Counting Application Example

H56/H52 CPU comes with 6 high-speed counters, and can also be used with HSC high-speed counting modules. This section will introduce application examples of high-speed counting modules and CPU high-speed counters to realize counting and speed measurement of pulse signals feedback by servo motor encoders. .

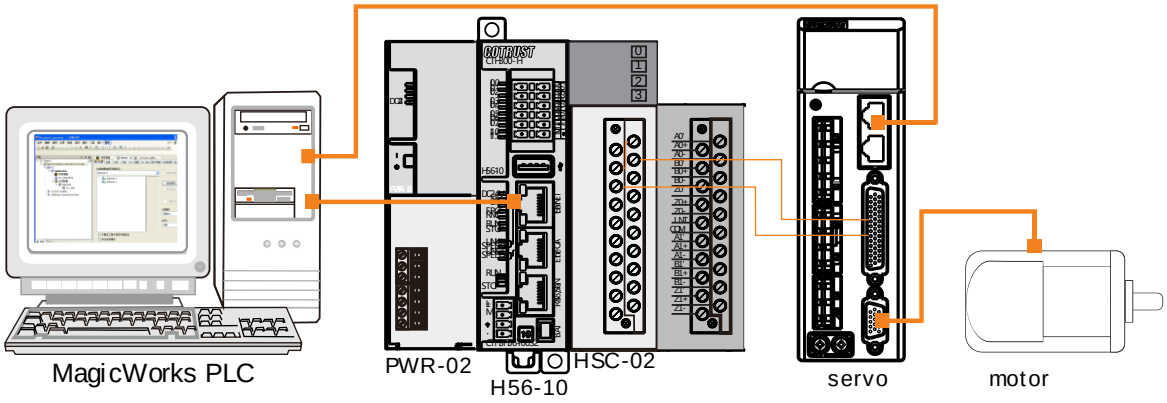
15.2.1 High-speed Counting Module Application Example

Table 14-22 Example components

Component	Function
Programming device PG\PC	Install MagicWorks PLC software (V2.19 or higher) to configure, program and debug the H56-10 motion controller.
Assembly rail	The CTH300 system rack is used to fix the modules in the system.
Power module PWR-02	Power the H56-10 motion controller and its 24 VDC load circuit.
External power supply	Provide A4S servo driver power supply
H56-10	The CPU executes the user program and provides 5V voltage to the CTH300 system backplane bus.
HSC-02 high-speed counting module	Connect with H56-10 through the bus to count the motor speed and position.
Drive equipment	A4S servo driver.
servo motor	Connect with A4S servo driver.
Standard network cable	Connect programming equipment and CPU with A4S servo driver.

Encoder cable	Connect A4S servo driver and motor.

System wiring diagram:



Step 1: Wiring

Open the front panels of the H56-10 and the power supply module PWR-02, and wire them. Specific steps are as follows:

1) Use a standard network cable to connect the programming device and H56-10 (EtherNET communication port).

2) H56-10 and high-speed counting module are connected through the Bus.

3) Use a standard network cable to connect the programming device and the A4S driver.

4) Use the encoder cable to connect the A4S driver and the motor.

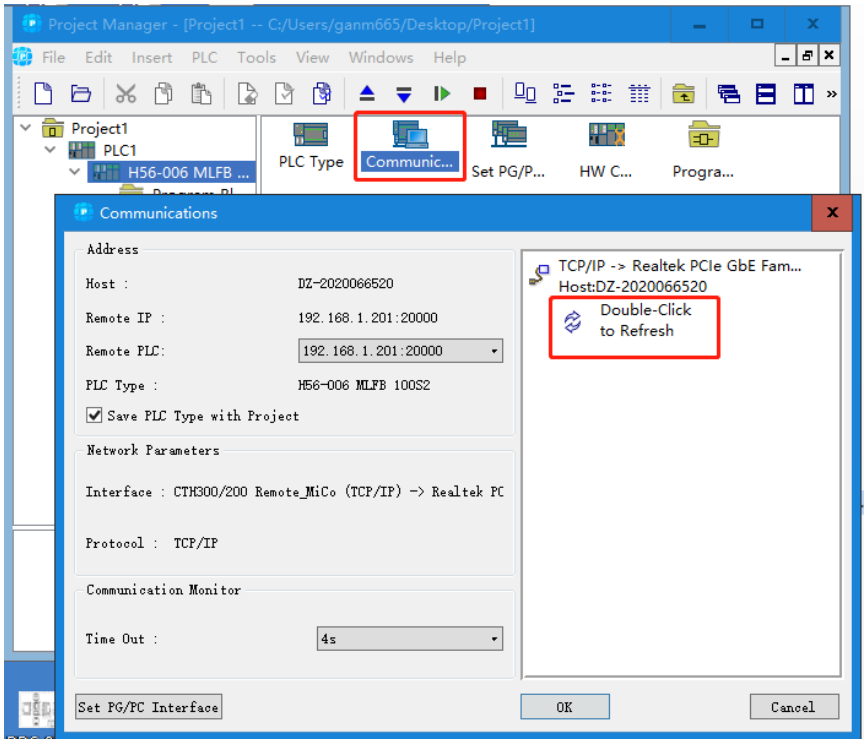
5) Wiring the high-speed counting module and A4S driver.

Step 2: Run the driver system

The motor starts to run normally by setting the parameters of the A4S servo drive. For specific operations, refer to the *A4S Series AC Servo Drive Instruction Manual*.

Step 3: Set up PLC communication

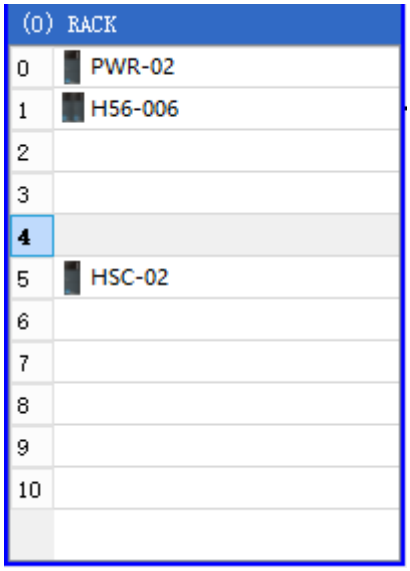
Refer to Chapter 2 Introduction to use, and set up the communication in MagicWorks PLC so that the programming device can establish communication with H56.



Step 4: Perform hardware configuration in MagicWorks PLC

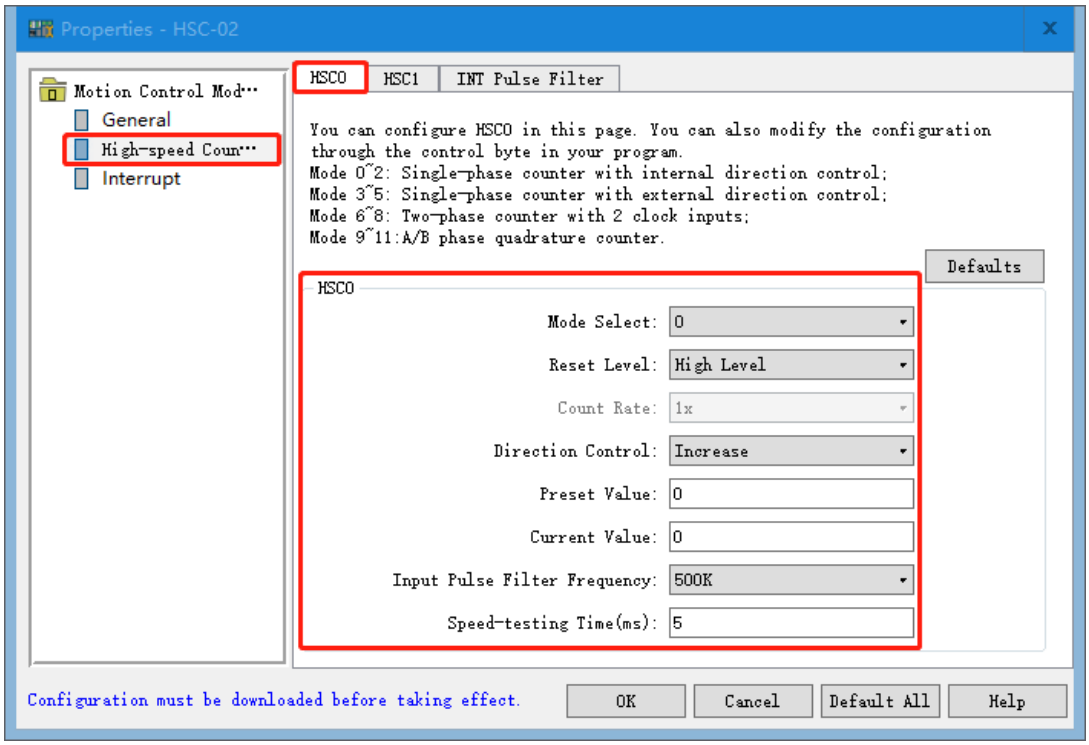
Click to select H56-10, and then double-click the hardware configuration  to enter the hardware configuration interface.

1) In the hardware configuration, add the power supply, CPU, and high-speed counting module to the rack through the device catalog. The completed configuration project is shown in the figure below:

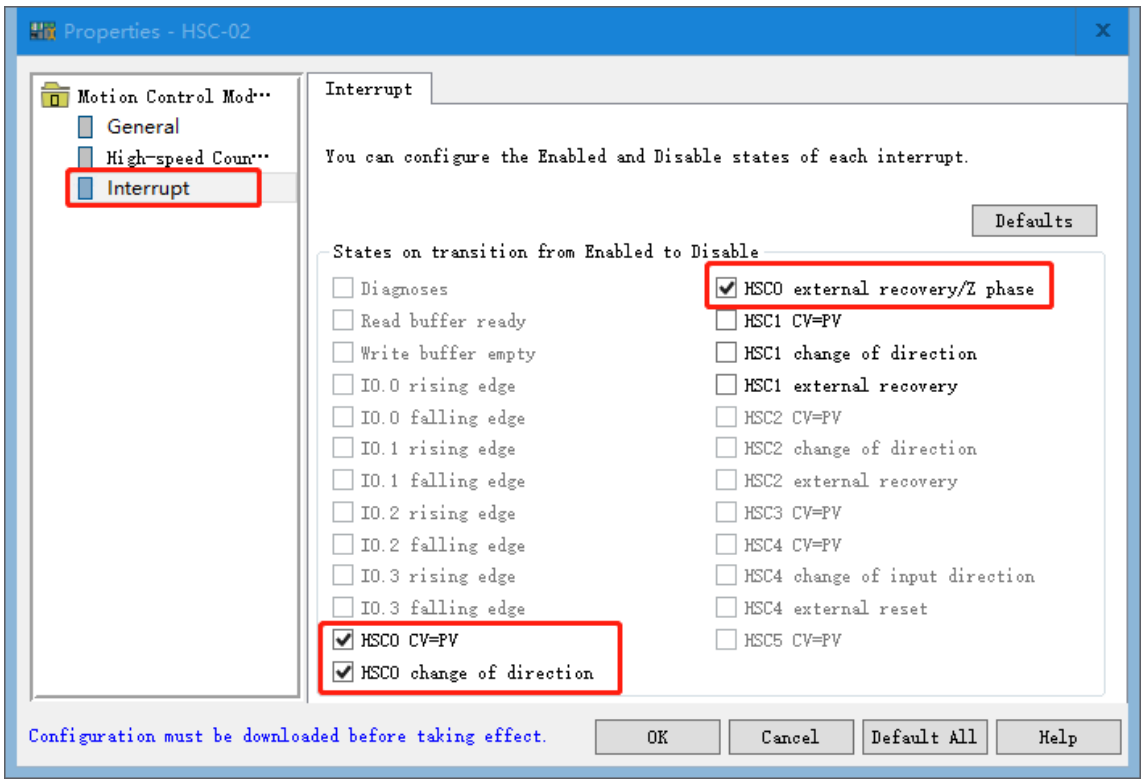


2) Configure high-speed counting module (HSC-02)

Double-click the module HSC-02 in the rack to open its configuration window, and configure the relevant parameters for the counting channel HSC0 in the tab "High Speed Counter" (this example only uses one channel, so only HSC0 is configured):



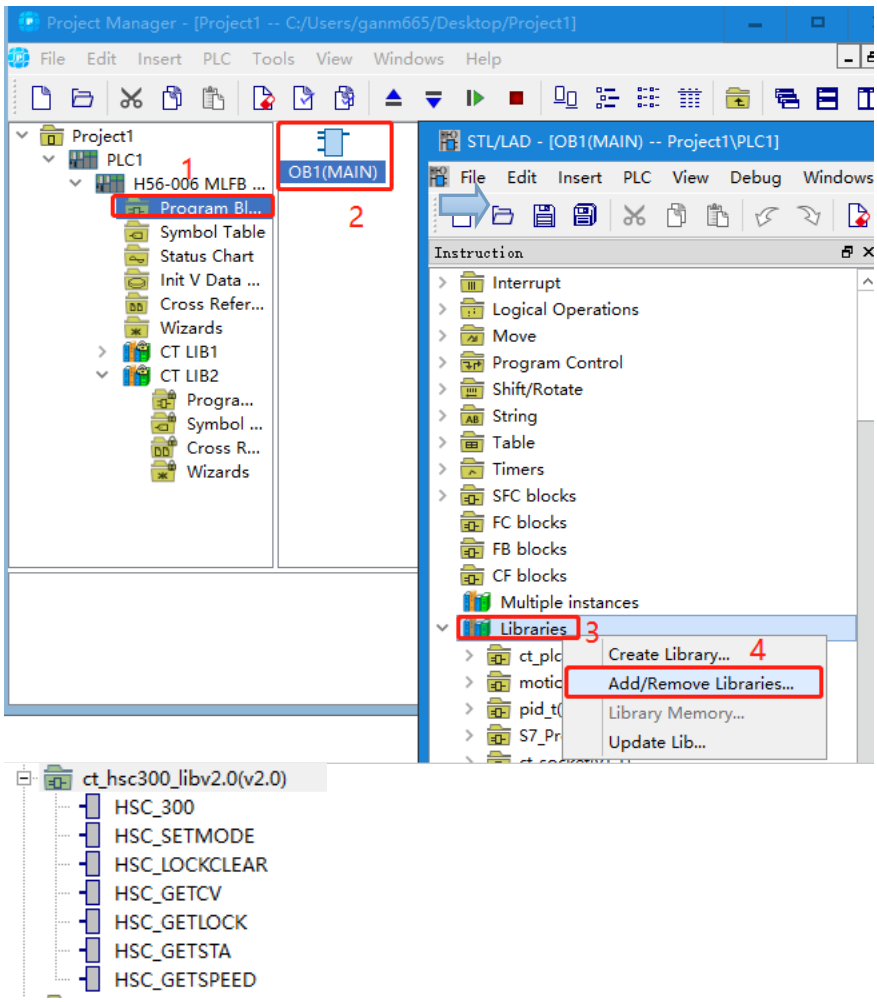
Then select the channel that needs to be interrupted (only one channel is used in this example, so only HSC0 is enabled):



Step 5: Programming

1) Add high-speed counter configuration library hsc_300_lib v2.0

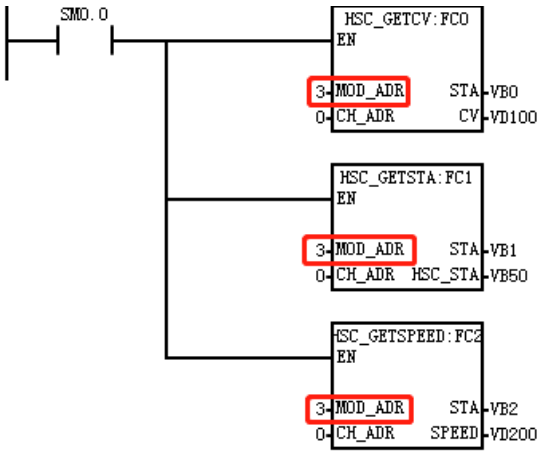
Open the Program Block, then right-click in the instruction tree of the program and select "Libraries" -> "Add/Remove Libraries", and click the "Add" button to select add library files. The successfully added library files will be displayed in the instruction tree:



2) Call hsc 300 lib library instructions for programming

Since the HSC-02 module has been configured in the hardware configuration, you can directly use instructions (HSC_GETCV, HSC_GETSPEED, HSC_STA) to read the speed, position, and status of the module.

Network 1: This example uses HSC-GETCV, HSC-DETSTA, HSC-GETSPPED to read the current position, state and speed of the motor.



Tips

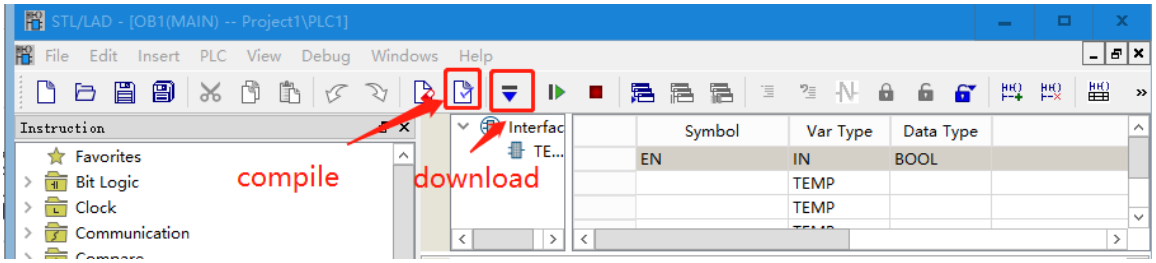
1. Please use hsc_300_lib V2.0 or higher version library.
2. If the HSC-02 module is not configured, it can be configured through the command HSC_300.

3. In this example, the high-speed counter module is in slot 3 of the rack, so its MOD_ADR is 3. if the high-speed counter of the CPU itself is selected, MOD_ADR should be 1 (the position of the CPU in the rack in the hardware configuration) .

Debug and monitor program

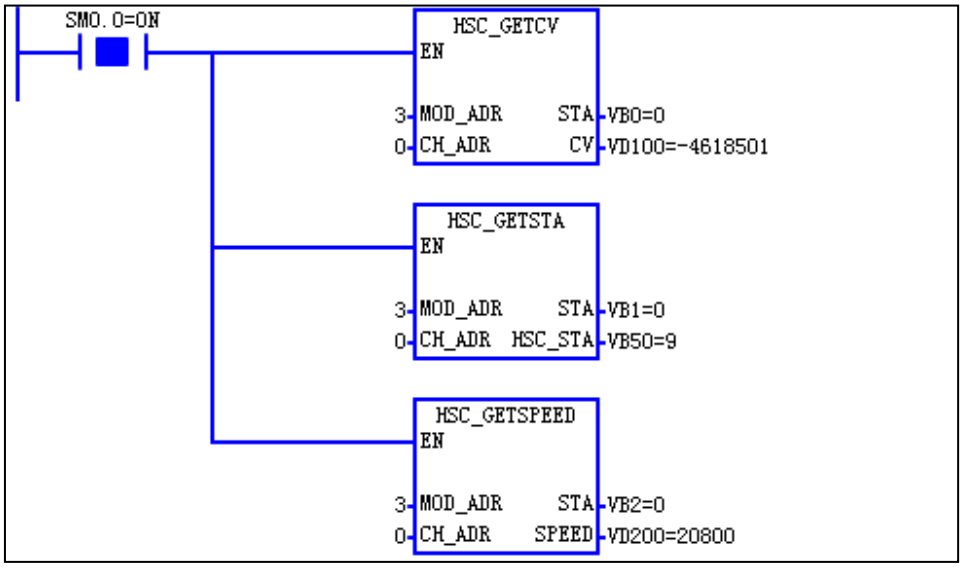
1) Compile and download the program

Select "File" → "Save" to save the current configuration, and then select "PLC" → "Compile" to compile the current project. if the compilation is successful, you can download it. Click "PLC" → "Download" to download the program blocks and hardware configuration from the programming device to H56-10.



2) Debugging program

After the program is downloaded successfully, put the PLC in the RUN mode, and then click "Online" to monitor the program status:

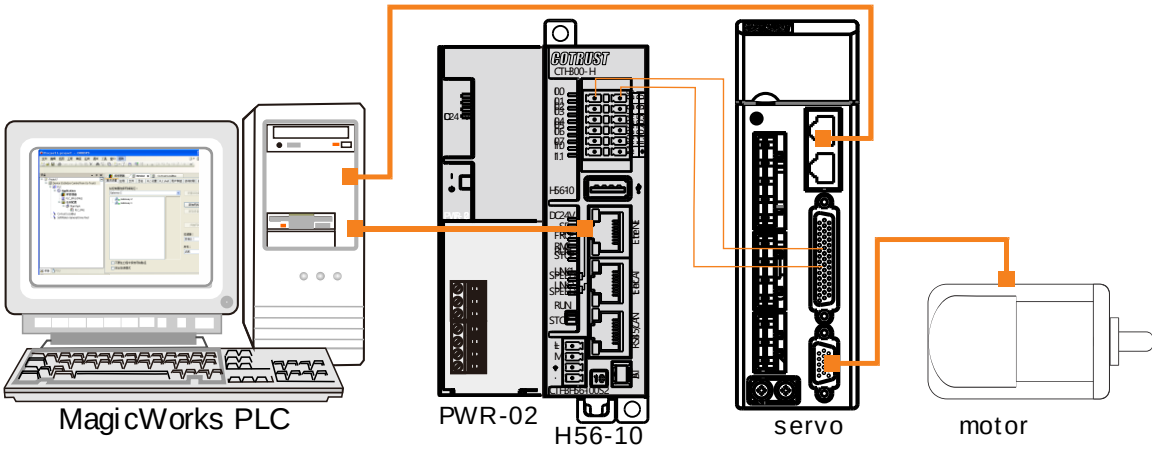


The current speed (frequency) and current position of the motor can be obtained by monitoring the above commands.

<Remarks> When the system fails, please refer to Chapter 5.8 CPU Fault Diagnosis

15.2.2 CPU High-speed Counter Application Example

System wiring diagram:



Step 1: Wiring

Open the front panel of the H56-10 and power supply module PWR-02, and then wire. Proceed as follows:

1) Connect the programming device and H56-10 (EtherNET communication port).

2) Use a standard network cable to connect the programming device and the A4S driver.

3) Use the encoder cable to connect the A4S driver and the motor.

Step 2: Run the driver system

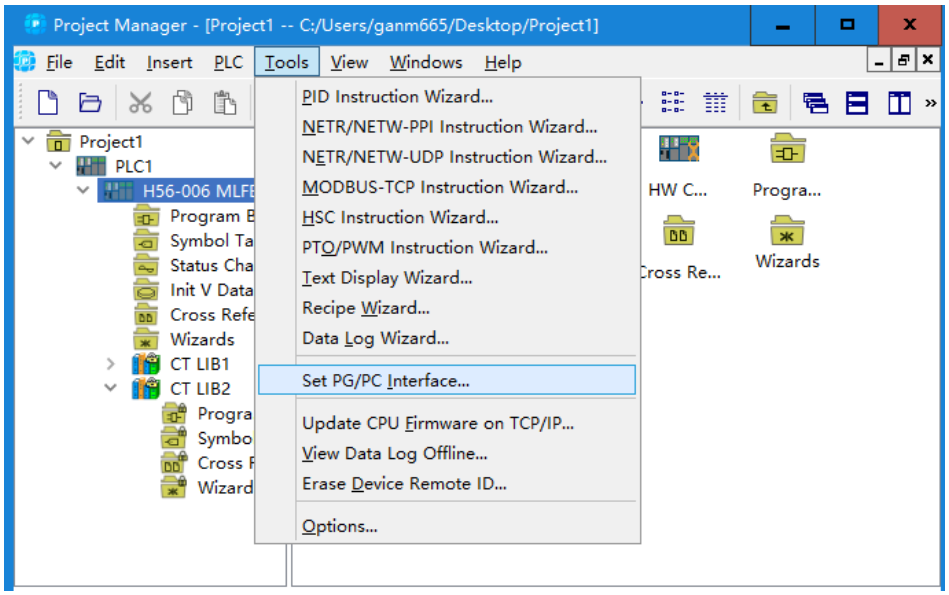
The motor starts to run normally by setting the parameters of the A4S servo drive. For specific operations, refer to the *A4S Series AC Servo Drive Instruction Manual.*

Step 3: Set up PLC communication

Create a new project, add H56-10 to the project, and then refer to the following steps to set up H56-10 communication.

1) Set PG/PC interface

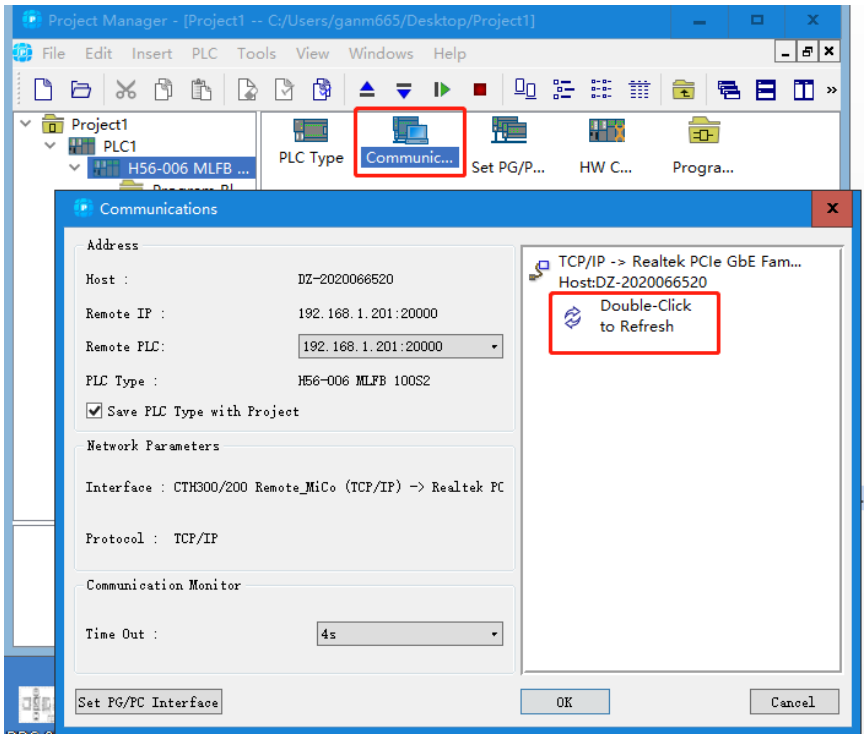
Click "Tools" → "Set PG/PC Interface", then select the interface "PC/PPI Cable (PPI)", then click "Properties" to enter the property settings, you can set the communication baud rate and serial port, and finally click "OK" complete the PG/PC interface settings.



2) Communication with H56-10

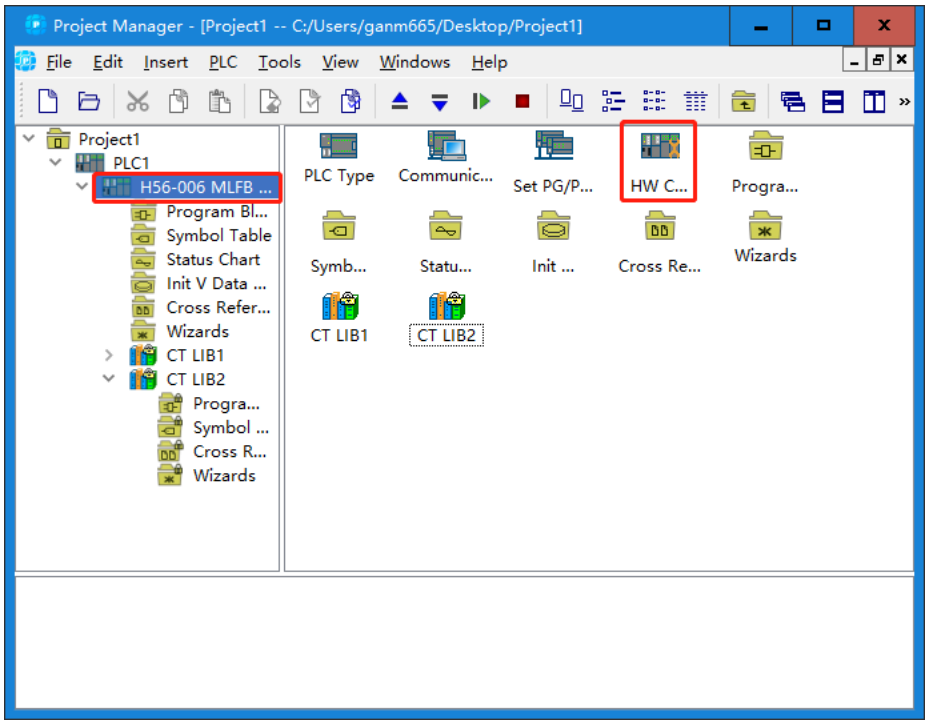
Double-click the communication icon, and then double-click in the communication to

search PLC, and the successfully searched H56-10 will be displayed in the communication dialog box.

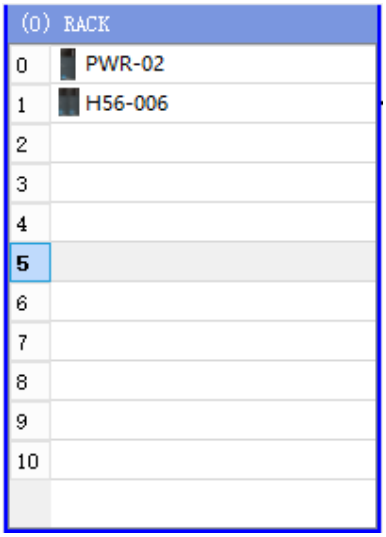


Step 4: Hardware configuration

In MagicWorks PLC, click to select H56-10, and then double-click the Hardware Config to enter the hardware configuration interface.

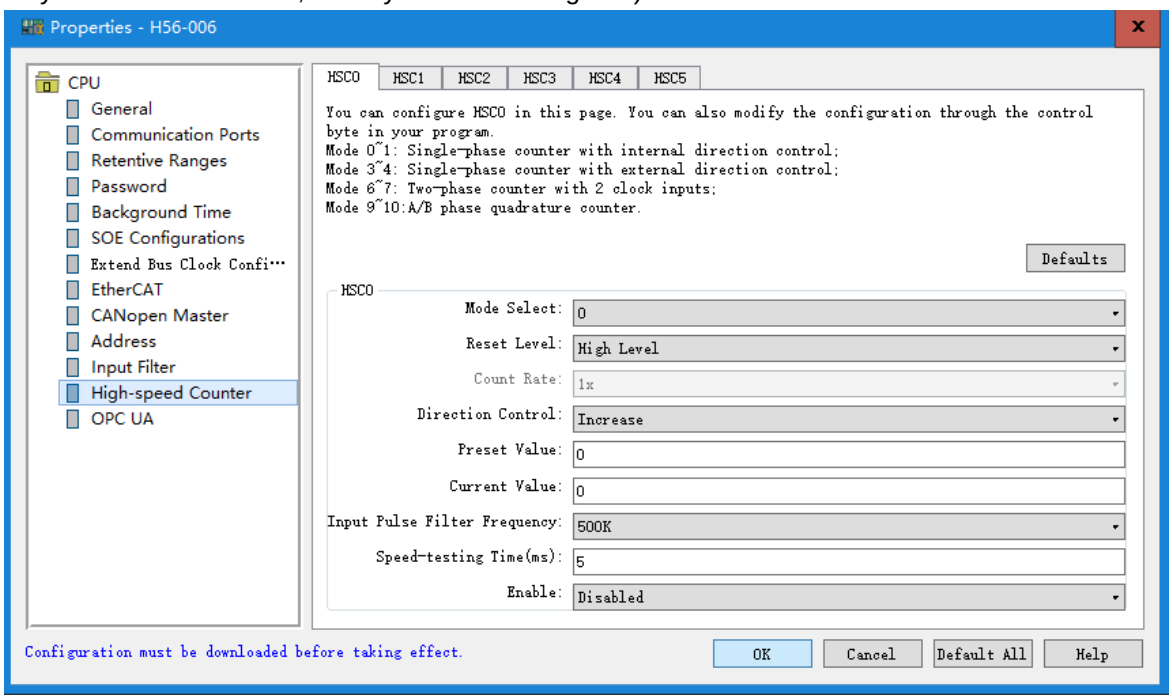


1) In the hardware configuration, add the power supply and CPU to the rack through the device catalog, and the configured project is shown in the figure below:



2) Configure H56-10

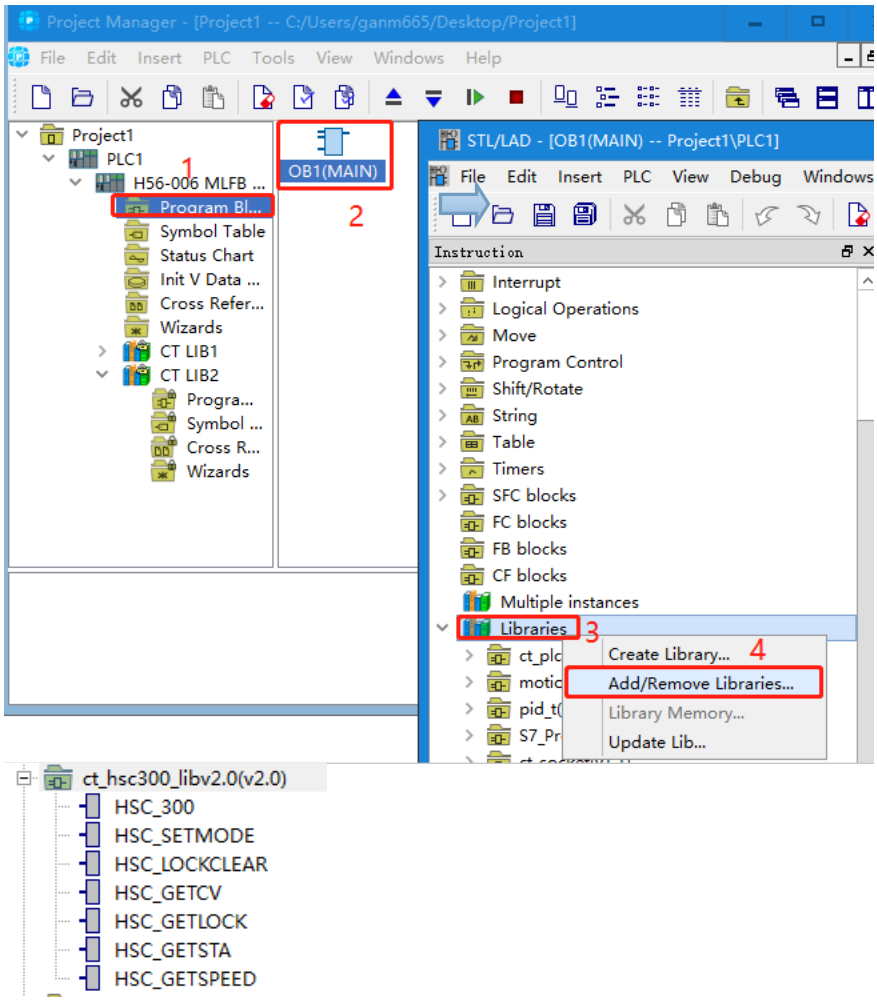
Double-click the H56-10 in the rack to open its configuration window, select "High-speed Counter", configure the relevant parameters for the counting channel HSC0 (in this example, only one channel is used, so only HSC0 is configured):



Step 5: Programming

1) Add High-speed counter configuration library hsc_300_lib v2.0

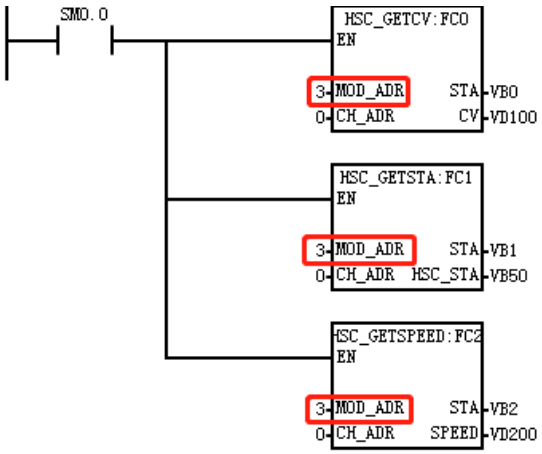
Open the Program Block, then right-click in the instruction tree of the program and select "Libraries" -> "Add/Remove Libraries", and click the "Add" button to select add library files. The successfully added library files will be displayed in the instruction tree:



2) Call hsc 300 lib library instructions for programming

Since the HSC-02 module has been configured in the hardware configuration, you can directly use instructions (HSC_GETCV, HSC_GETSPEED, HSC_STA) to read the speed, position, and status of the module.

Network 1: This example uses HSC-GETCV, HSC-DETSTA, HSC-GETSPPED to read the current position, state and speed of the motor.



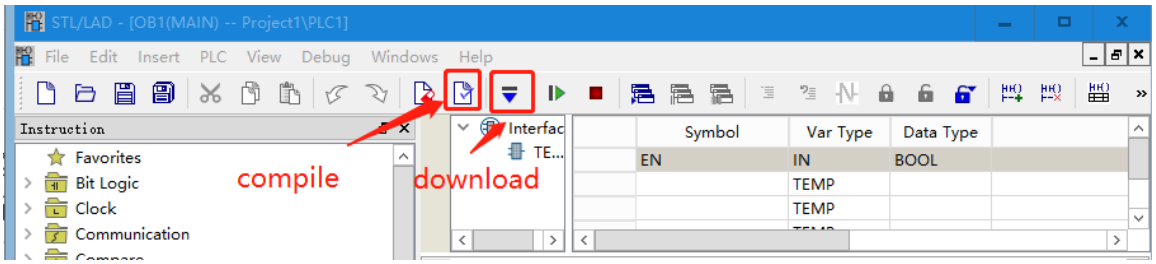
Tips

Using the high-speed counter that comes with the CPU, MOD_ADR should be 1 (the position of the CPU in the rack in the hardware configuration).

Debug and monitor program

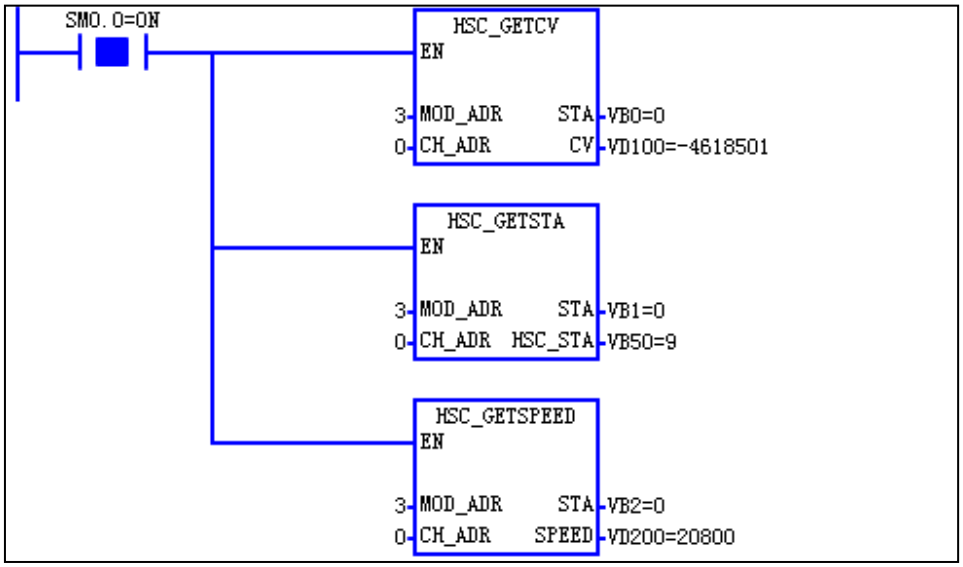
1) Compile and download the program

Select "File" → "Save" to save the current configuration, and then select "PLC" → "Compile" to compile the current project. if the compilation is successful, you can download it. Click "PLC" → "Download" to download the program blocks and hardware configuration from the programming device to H56-10.



2) Debug program

After the program is downloaded successfully, put the PLC in the running mode, and then click "OnLine" to monitor the program status:



The current speed (frequency) and current position of the motor can be obtained by monitoring the above commands.

<Remarks> When the system fails, please refer to Chapter 5.8 CPU Fault Diagnosis

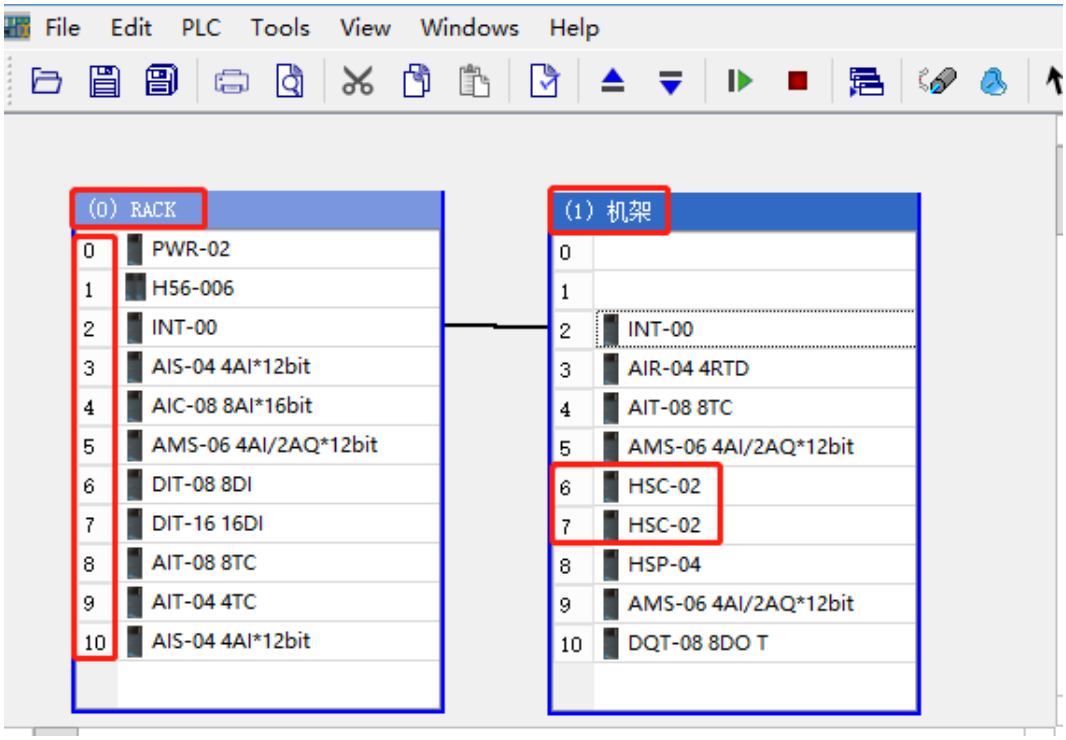
15.3 HSC Interrupt Example

The H56/H52 motion controller can generate interrupts by itself or by HSC module. This section takes the HSC interrupt event CV=PV as an example to introduce how to connect interrupts, enable interrupts, and determine whether the specific interrupt is the module or the CPU itself.

15.3.1 Hardware Config

- Create a new project in MagicWorks PLC, add H56-10 to the project, and then connect H56-10 to PC by referring to 2.3 HW Config

- Click to select the H56-10 in the MagicWorks PLC project view, and then double-click the hardware configuration icon in the right working window to enter the hardware configuration interface.
- In the hardware configuration interface, add the power supply, CPU and required expansion modules to the rack through the device catalog.



Instruction:

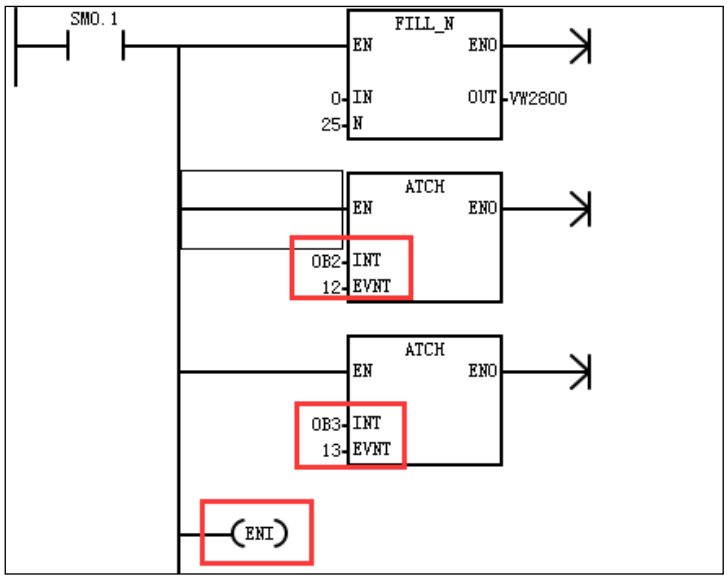
As shown in the figure above, there are two racks in the hardware configuration, namely No. 0 and No. 1, the CPU is located in the No. 1 slot of No. 0 rack, and the two HSC modules are inserted into the No. 1 slot 6/ 7.

15.3.2 Example of interrupt program

As described in "Hardware Configuration" above, slots 6/7 of rack 1 each carry an HSC module. The following takes HSC interrupt event CV=PV as an example to illustrate how to use HSC interrupt.

Connection interrupt

As shown in the program in the following figure, the interrupt event code 12/13 (CV=PV interrupt of HSC0/HSC1) is associated with the interrupt subroutine OB2 and OB3 respectively. When the interrupt condition corresponding to the interrupt number is triggered, the corresponding interrupt program will be executed.



Enable interrupt

Interrupt program execution is enabled by the ENI instruction.

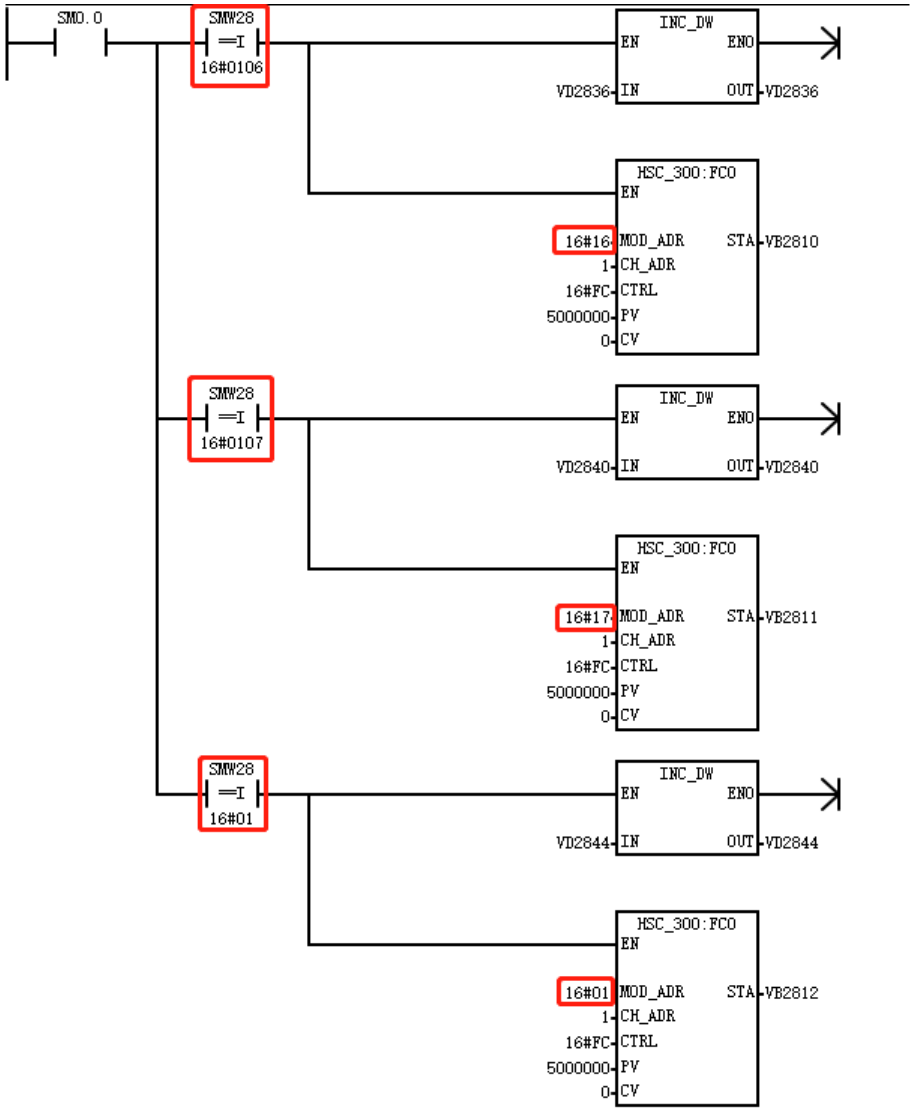
Determine the source of the specific interrupt

In the interrupt subroutines OB2 and OB3, according to the special memory SMW28 (double word), determine the logical address of the module that generates the diagnostic interrupt (fault), or the hardware interrupt.

As shown in the figure below, when the value of SMW28=16#0106, it means that the HSC module in the No.6 slot on the No.1 rack is interrupted. When the value of SMW=16#0107, it means that the No.7 slot on the No.1 rack HSC module of the slot is interrupted, when the value of SMW=16#01, it means that the CPU on the 0th rack is interrupted.

Tips

It must be noted that the special memory SMW28 is a word, and the MOD_ADR module address parameter is a byte. In the example below, SMW 16#0106 and MOD_ADR 16#16 are the same module address.


SM0.0
SMW28
==I
16#0106
INC_DW
EN
ENO
VD2836
IN
OUT
VD2836
HSC_300:FCO
EN
16#16
MOD_ADR
STA
VB2810
1
CH_ADR
16#FC
CTRL
5000000
PV
0
CV
SMW28
==I
16#0107
INC_DW
EN
ENO
VD2840
IN
OUT
VD2840
HSC_300:FCO
EN
16#17
MOD_ADR
STA
VB2811
1
CH_ADR
16#FC
CTRL
5000000
PV
0
CV
SMW28
==I
16#01
INC_DW
EN
ENO
VD2844
IN
OUT
VD2844
HSC_300:FCO
EN
16#01
MOD_ADR
STA
VB2812
1
CH_ADR
16#FC
CTRL
5000000
PV
0
CV

Shaft and CAM

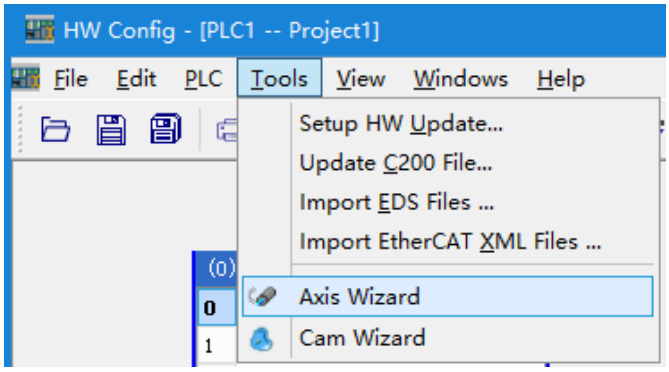
16

Combined with the programming software magicworks PLC, this paper introduces the electronic cam with shaft configuration of H56 / H52.

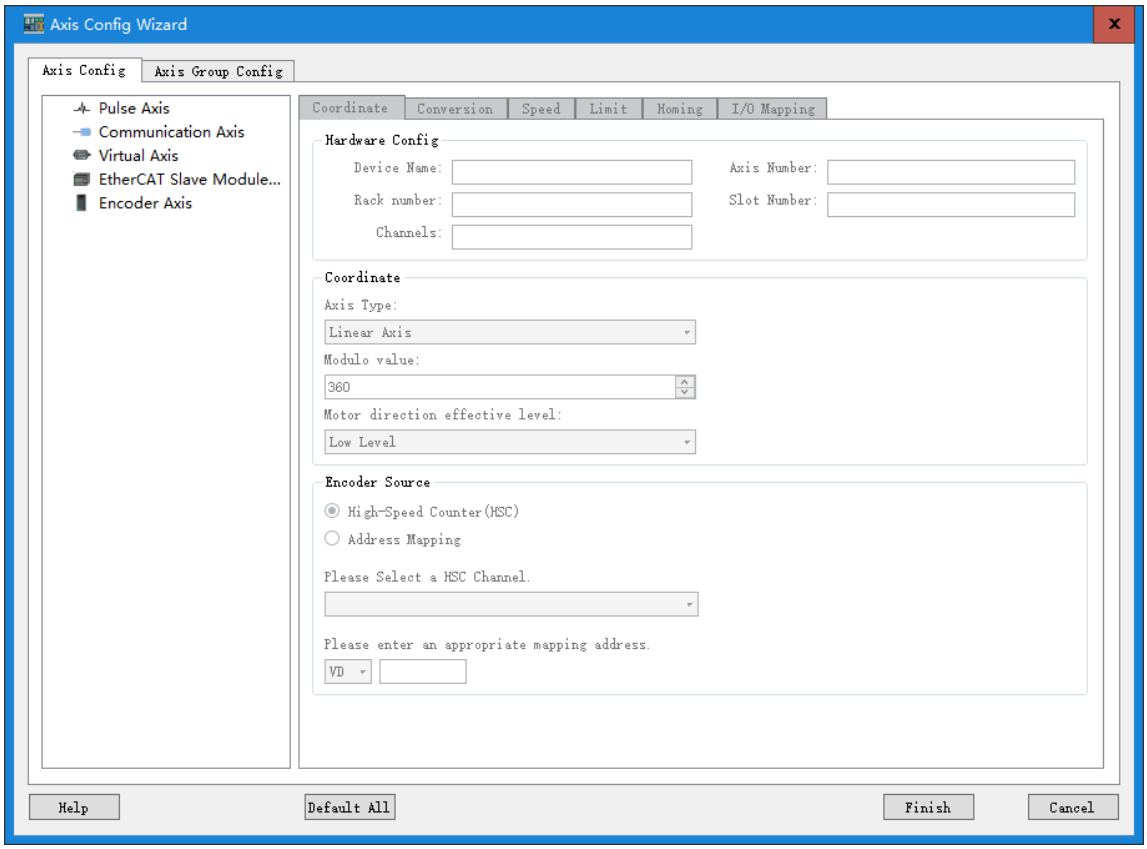
H56-10/H52-10 motion controller supports axis configuration and electronic cam, and can be combined with the programming software MagicWorks PLC (V2.19 and above) for related configuration.

16.1 Axis Wizard

Select "Tools" -> "Axis Wizard" in the HW Config interface to configure the axis, as shown in the figure below.



The axis configuration wizard interface is as follows:

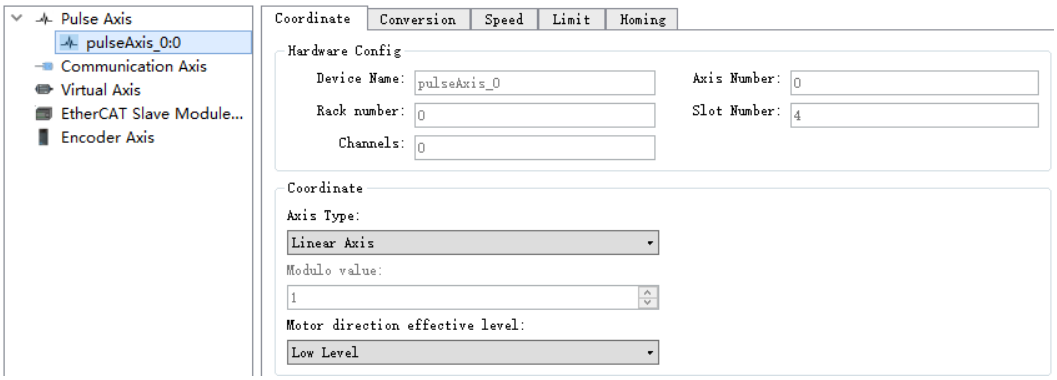


Tips

1. The total number of axes configured for all axes in the axis configuration is up to 64, and the number of single-type axes is up to 64.
2. In the PLC program, the current status of the axis can be obtained through the command MC_Readstatus. if you want to obtain the axis communication status, directly call SMB400,

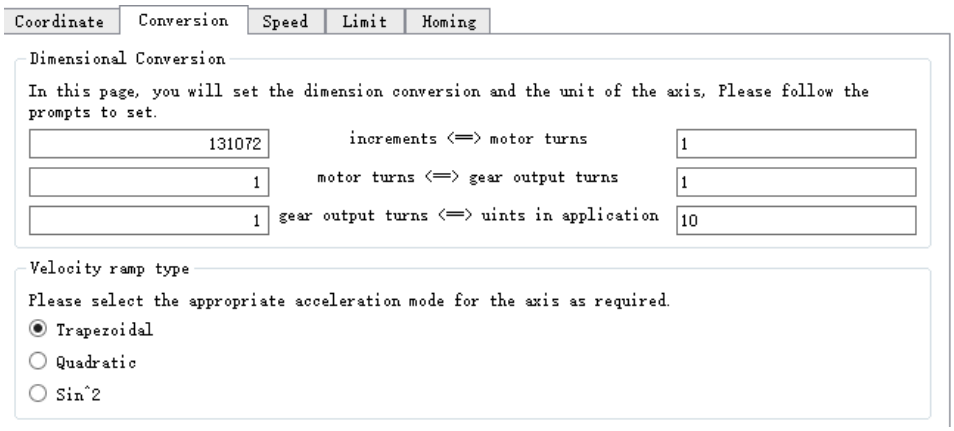
SMB401 and SMB402.

[Coordinate]



- Hardware config: "Device Name" is the name of the axis when the axis is added. "Axis Number" is the sequence ID of the axis addition, which cannot be modified, and increases sequentially with the addition of the axis. If an axis is deleted, the newly added axis number is the axis number of the last deleted axis. "Rack Number" and "Slot Number" are the Rack number and Slot number of the corresponding HSP module of the axis. "Channel" is the channel number of the axis corresponding to the HSP module, the range is 0 ~ 3 (only the Pulse axis and EtherCAT slave module axis have channel options).
- Coordinate: "Axis Type" can choose "Linear Axis" or "Rotation Axis". For the reciprocating mechanism of the screw type, the stroke is limited, and you need to know its absolute position within the range of the screw stroke. At this time, select " "Linear Axis" is better. if it is a single-direction type of Rotation Axis, linear mode is prone to position counting overflow and calculation errors, so it is better to choose " Rotation Axis". The remaining options are not selectable by default. "Motor direction effective level" can be "Low level" or "High level".

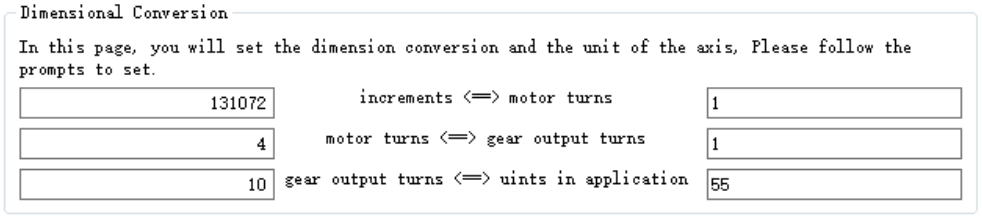
[Conversion]



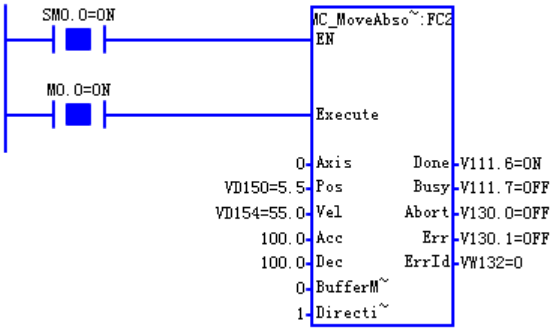
In the "Conversion" interface, you can set the type of dimension conversion and rate change according to the prompts. Set the "units in application" to 360, when the program command requires the servo to run 1 unit, the gear end will select 1/360 circle (axis movement 1 degree), and the servo motor end will rotate 5*1/360 circle (motor rotates 5 degrees), and so on. After setting the corresponding parameters (electronic gear ratio) according to the actual mechanical structure, the distance command can be input according to the physical unit of the movement distance of the application system, making the parameter control intuitive and easy to understand.

<Remarks> If any variable of the dimension conversion takes a negative value, the motor direction is opposite to the default direction.

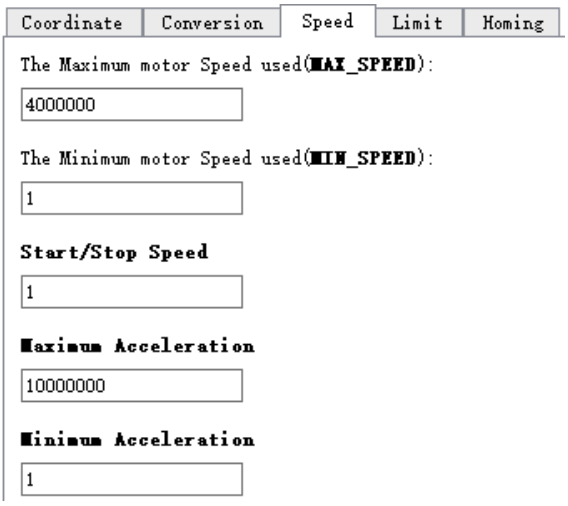
Example: After the servo motor is mechanically decelerated by the reducer 4:1, it drives the lead screw with a lead of 5.5mm, that is, when the lead screw rotates once, the screw slide moves 5.5mm. The dimension conversion settings are as follows:



The set dimension can be used as the dimension of the physical parameters of the MC control instruction. Download the above settings to the program block and program the workpiece to move to 5.5mm coordinates. The unit of position naming is the physical coordinate unit of the equipment. The program example is as follows:



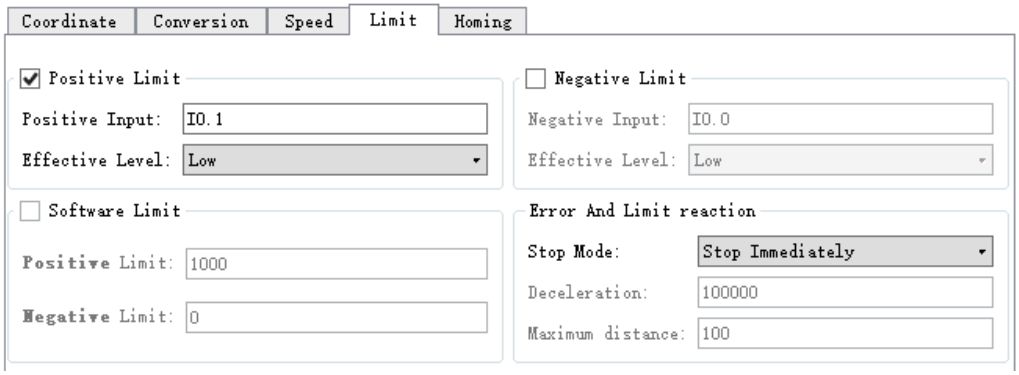
[Speed]



On this page, you can set The Maximum/Minimum Speed, Stop/Start Speed, and Maximum Acceleration and Minimum Acceleration of the motor. During the program running, if the set speed exceeds the configuration setting, the command will alarm. The error code is as follows:

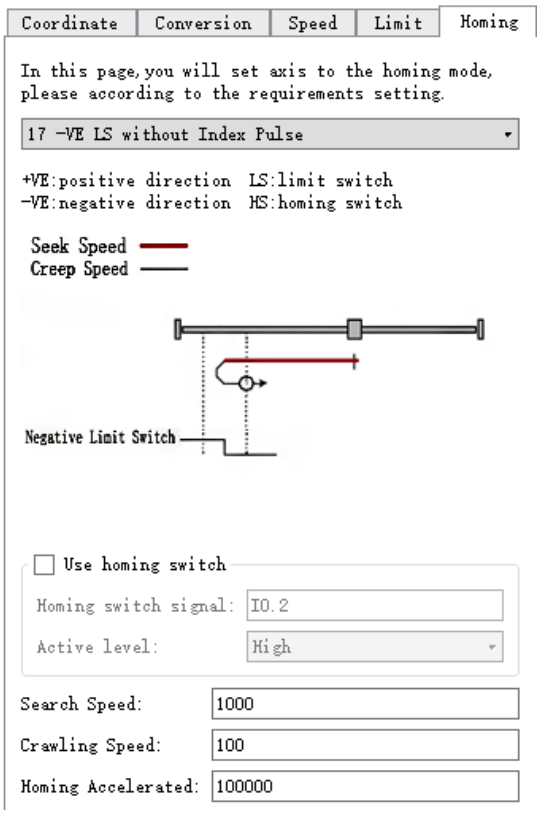
Errorcode	describe
5	The speed is too small (greater than the minimum speed)
6	The speed is too large (greater than the maximum speed)
7	Acceleration is too small
8	Too much acceleration
9	Deceleration is too small
10	Deceleration too much

[Limit]



Check "Positive Limit" and "Negative Limit" to modify the setting of positive/negative input, and select "High" or "Low " for "Effective Level". Check "Software Limit" "Position" can modify the positive/negative limit value. The "Stop Mode" of "Error and Limit Response" can choose "Stop Immediately " or "Deceleration Stop".

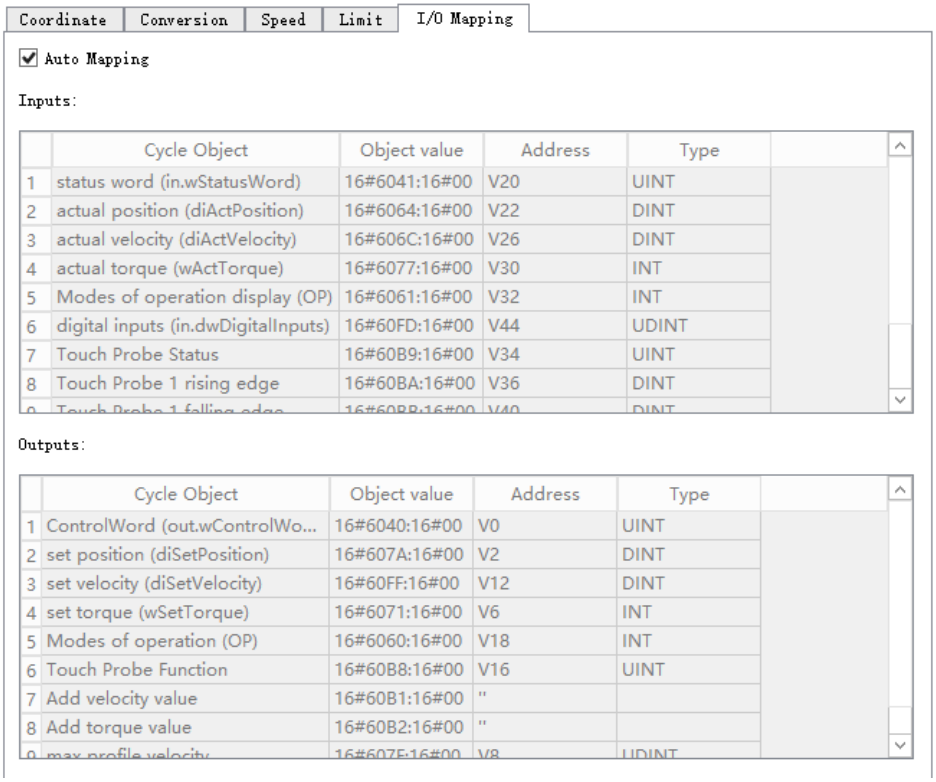
[Homing]



On this page, you can configure the Homing mode of the axis. This configuration is only supported by the pulse axis. Select the drop-down option to view the return mode. There are 15 return modes to choose from.

Refer to chapter G.6 Introduction to the home function for the specific application of the back-to-origin function

[I/O Mapping]

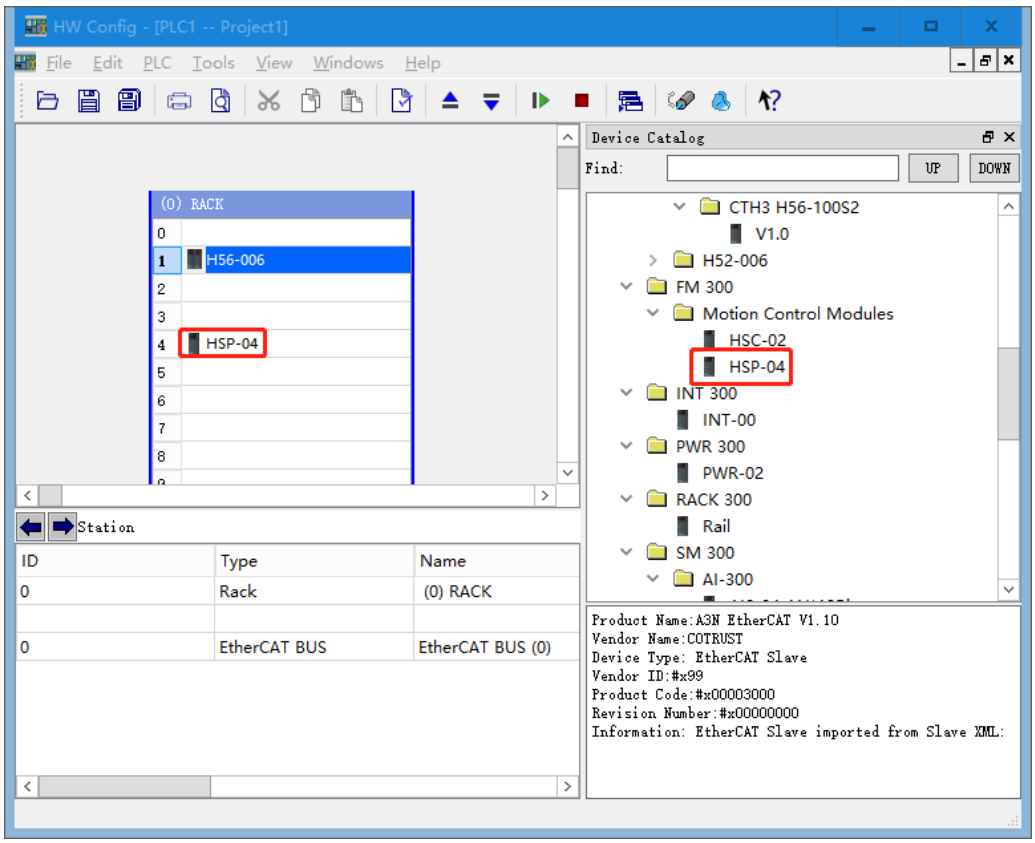


This configuration is only supported by Communication axis and EtherCAT slave axis. If you check "Auto Mapping" on this page, the input and output cannot be modified. It is recommended to check, because the servo index address of the axis with and without CAI402 is different.

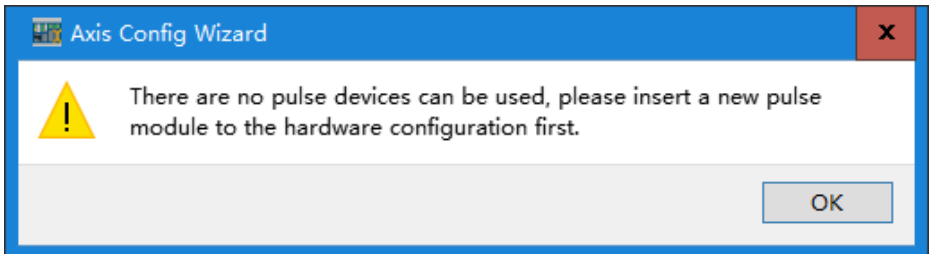
16.1.1 Axis Wizard

1. Pulse Axis configuration

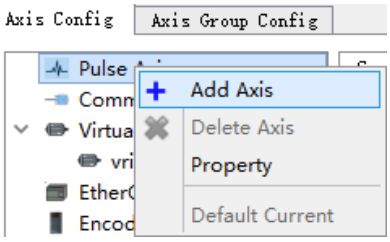
Before adding the Pulse Axis, you need to add the motion control module "HSP-04" to the rack where the master is located, as shown in the figure below:

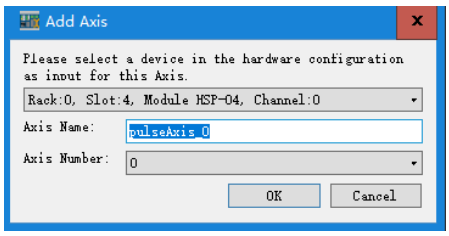


If "HSP-04" is not added, the following prompt will appear when you add a Pulse Axis in the Axis Wizard:

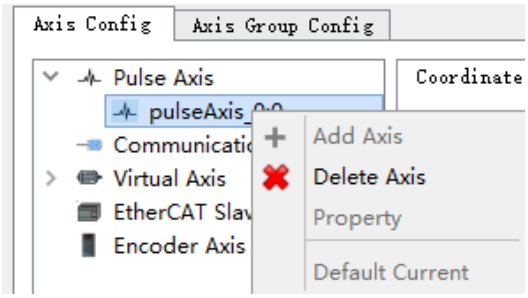


After completing the above configuration, select "Tools" -> "Axis Wizard" in the HW Config interface to configure the axis,, right-click on the Pulse Axis in the Axis Wizard Config interface to choose to add an axis, the axis name can be changed freely, click "OK" to successfully add the axis, as shown below Show:



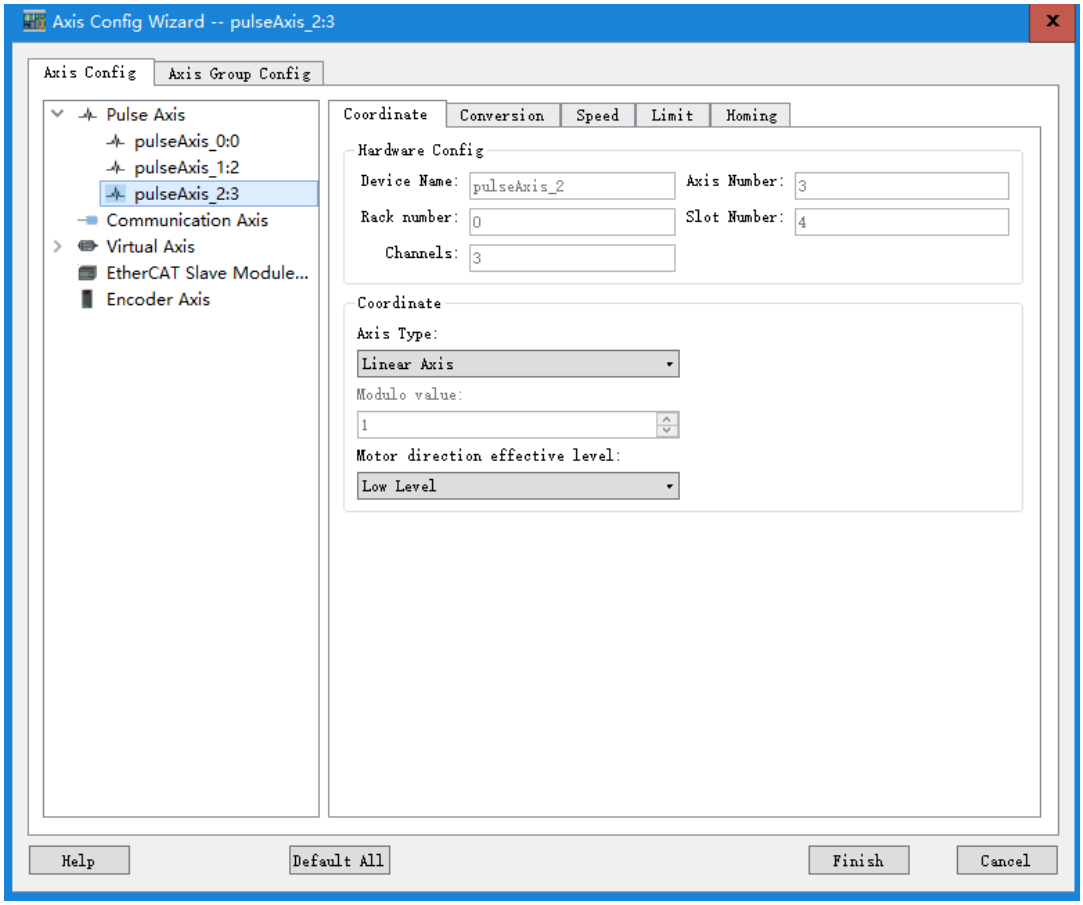


Right-click and select the added axis, you can choose to delete the axis, as shown in the figure below:

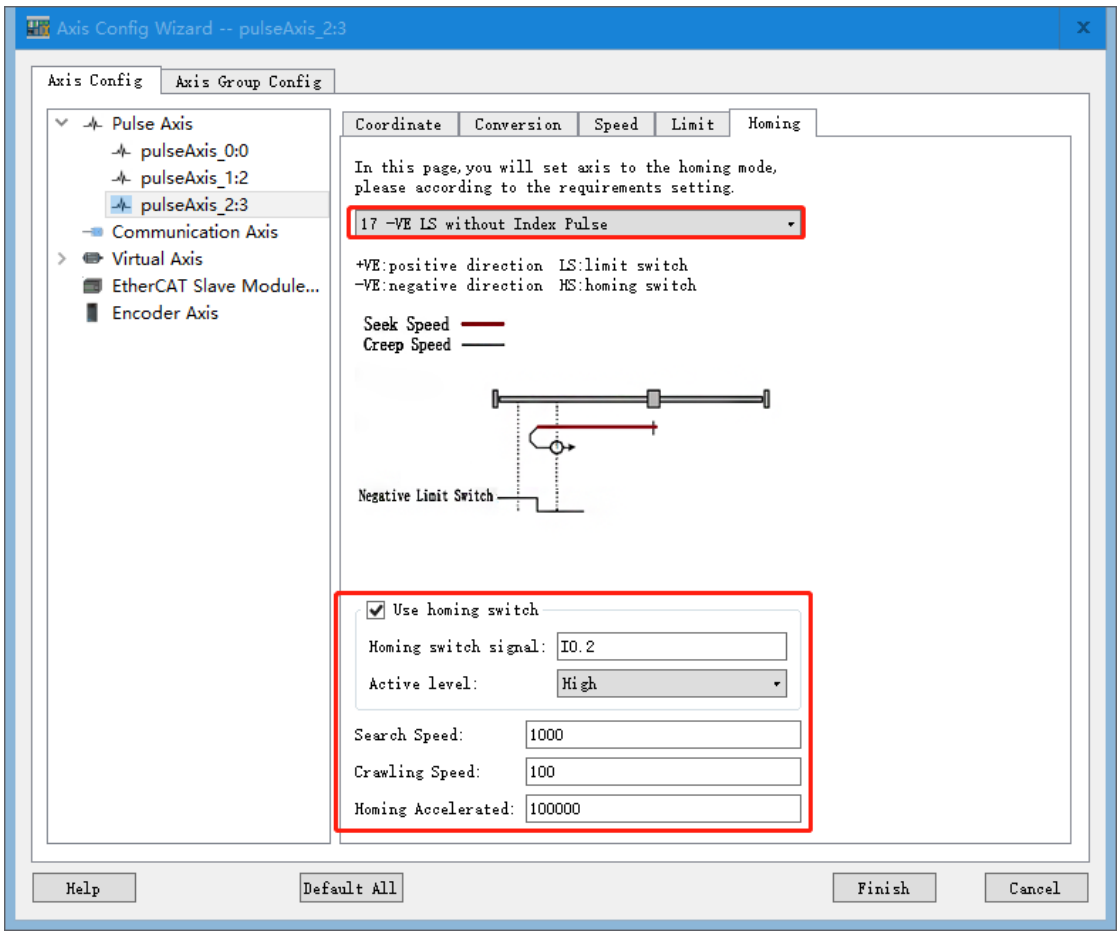


Tip:
One HSP module can add 4 pulse axes. If you need to add more pulse axes, you need to add HSP modules in the rack.

The added pulse axis interface is as follows:

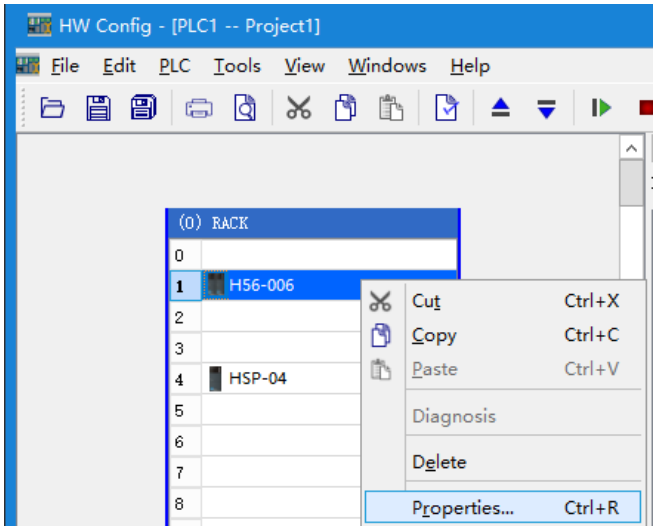


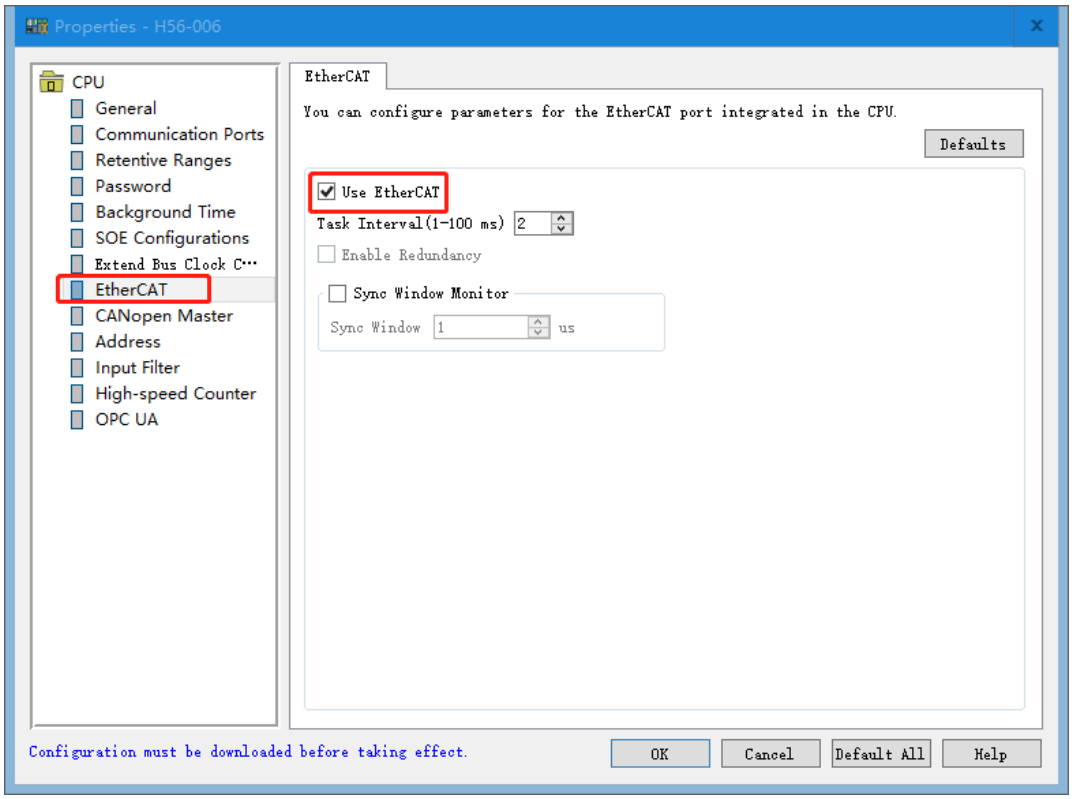
If you need to use Homing function, click "Homing", select the Homing mode (a total of 15 Homing modes), check the "Use homing switch", set the Homing parameters.



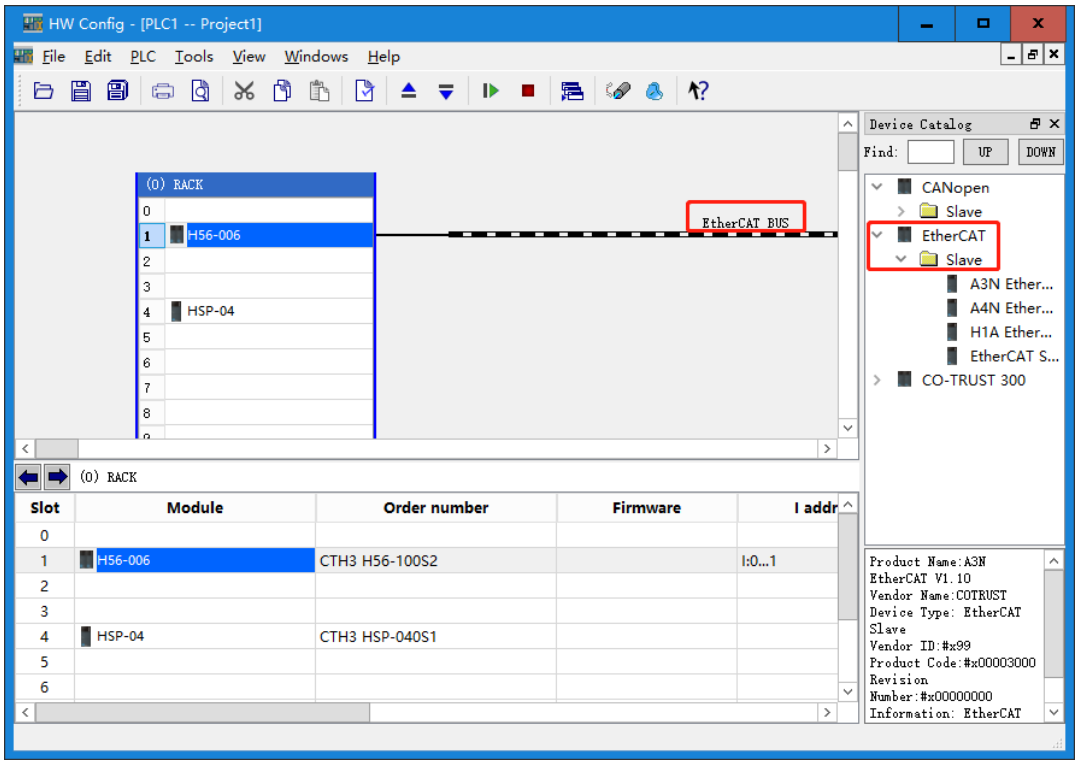
2. Communication Axis configuration

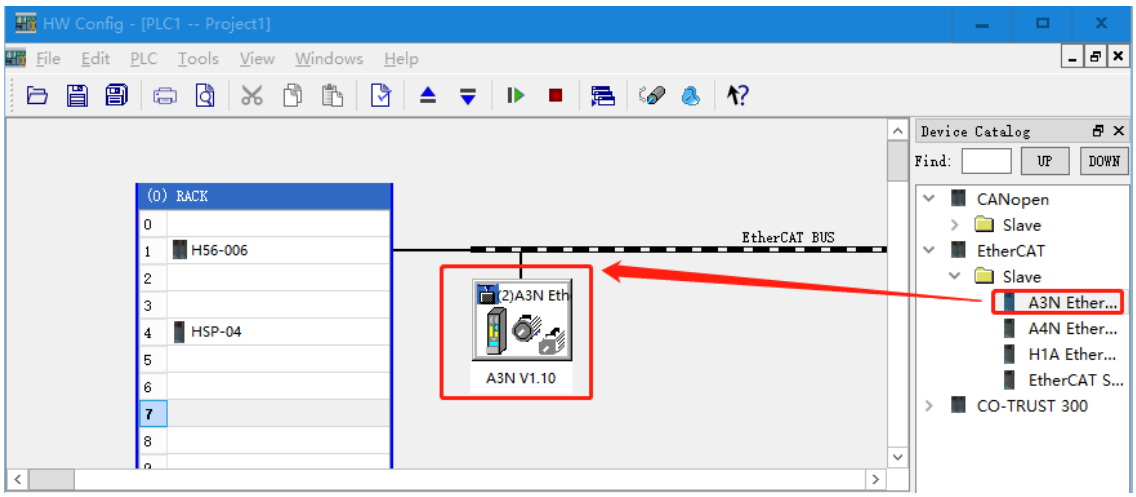
1) Right-click the H56 on the rack, select "Properties" "EtherCAT", and click "Use EtherCAT " on the page.



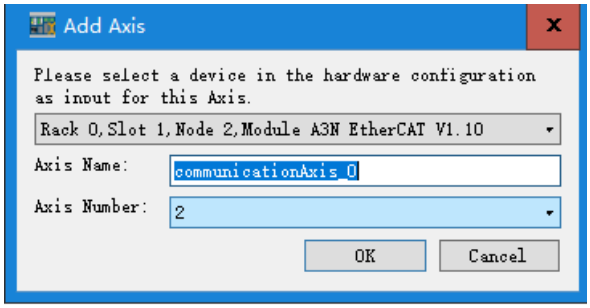


2) After confirming, you can see "EtherCAT BUS" on the Hardware Configuration interface, click on the right "EtherCAT", select the slave and drag the slave under "EtherCAT BUS".



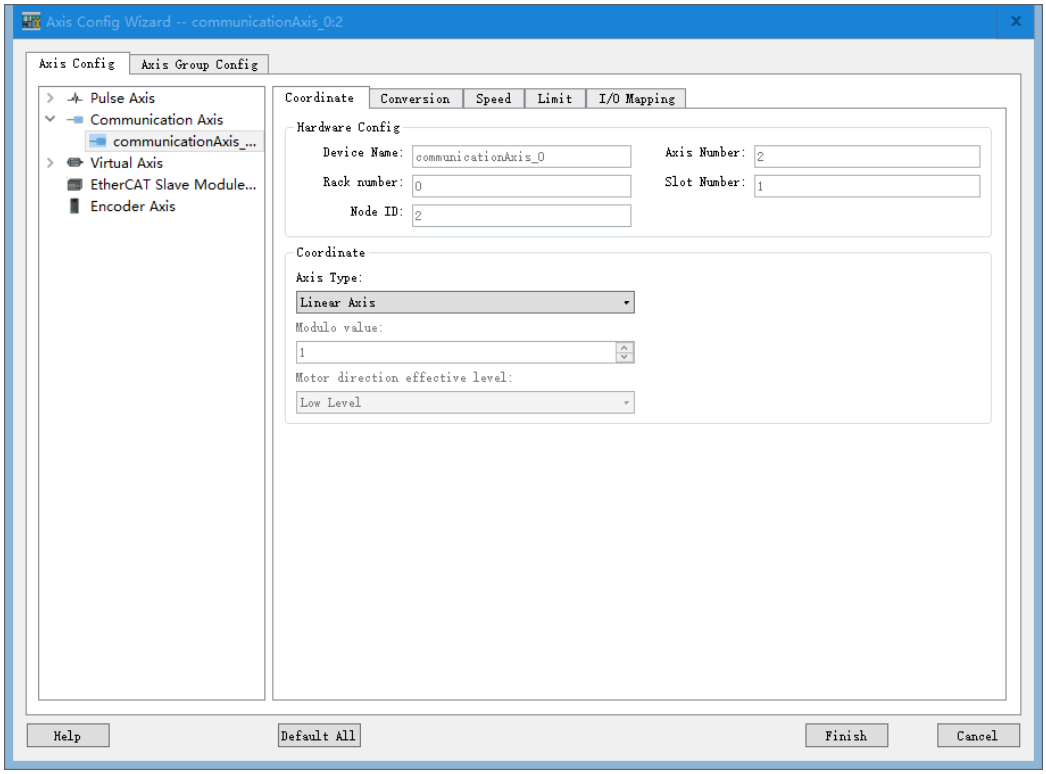


3) After completing the above configuration, Select "Tools" -> "Axis Wizard" in the HW Config interface to configure the axis, double-click "Communication Axis" in the axis wizard configuration interface to add a communication axis, and then select the input device for the axis. The optional device is the hardware group The slave added in the state. "Axis name" can be modified, click "OK" to complete the addition.



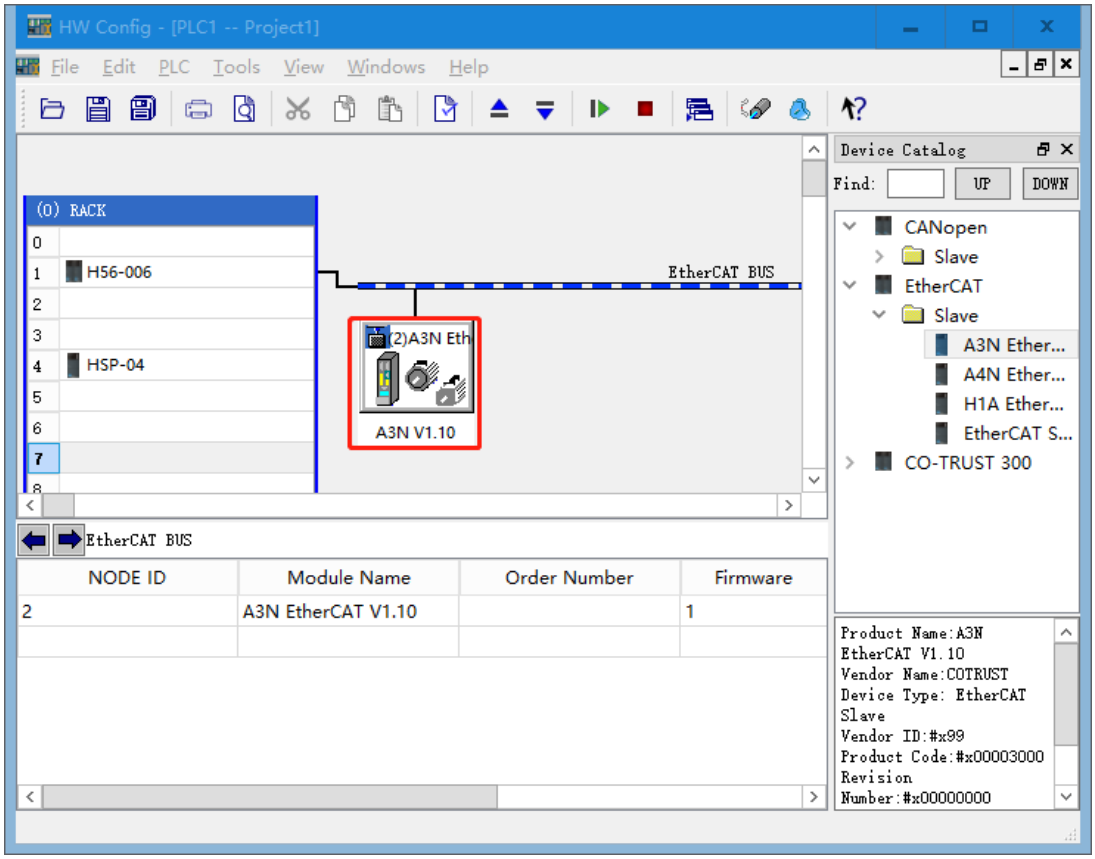
Note: A slave can only add one communication axis correspondingly.

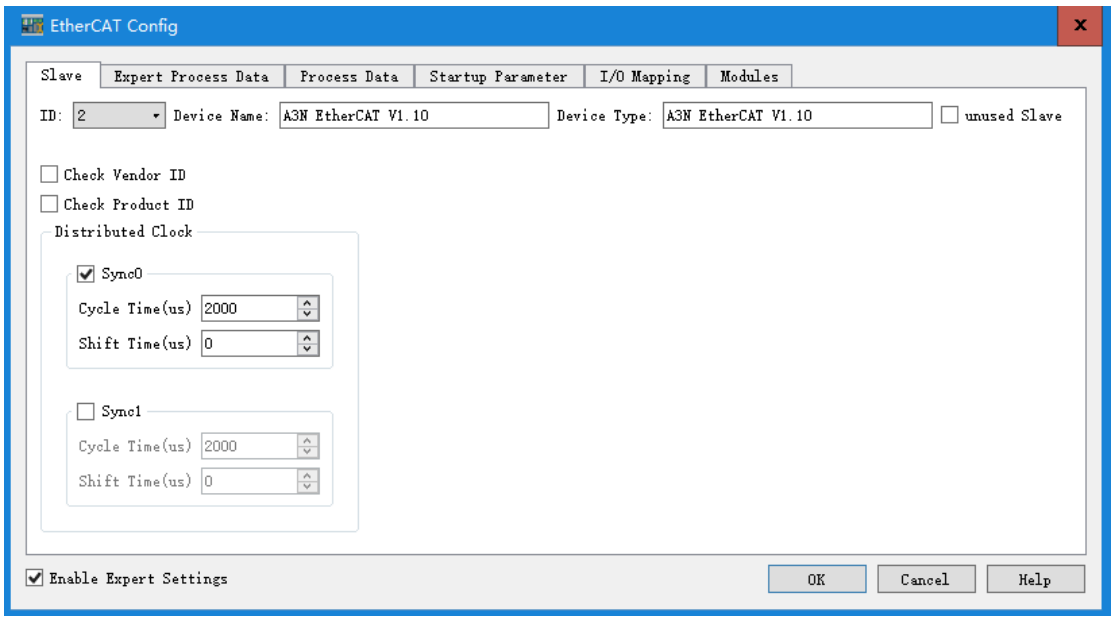
4) Th communication axis interface is as follows:



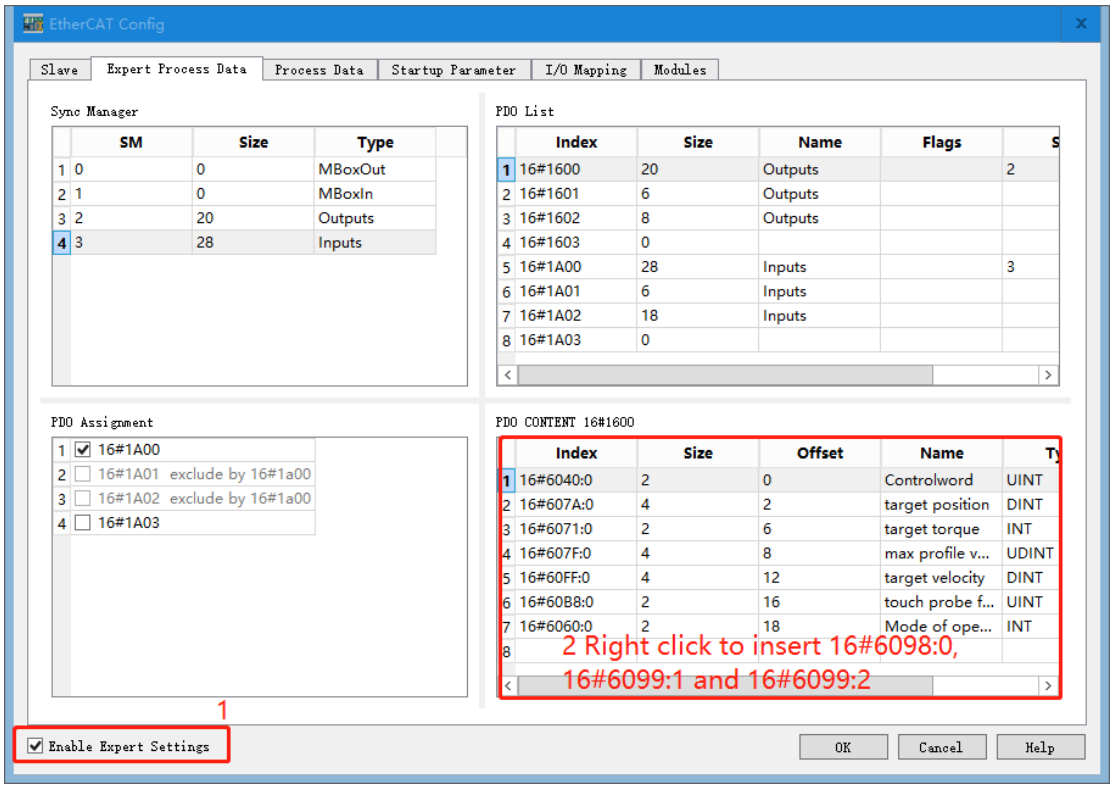
5) If you need to use the driver to homing (MC_Home instruction), you need to configure the homing method 16#6098:0 and Speed during search for switch 16#6099:1, Speed during search for zero 16#6099:2, the operation is as follows:

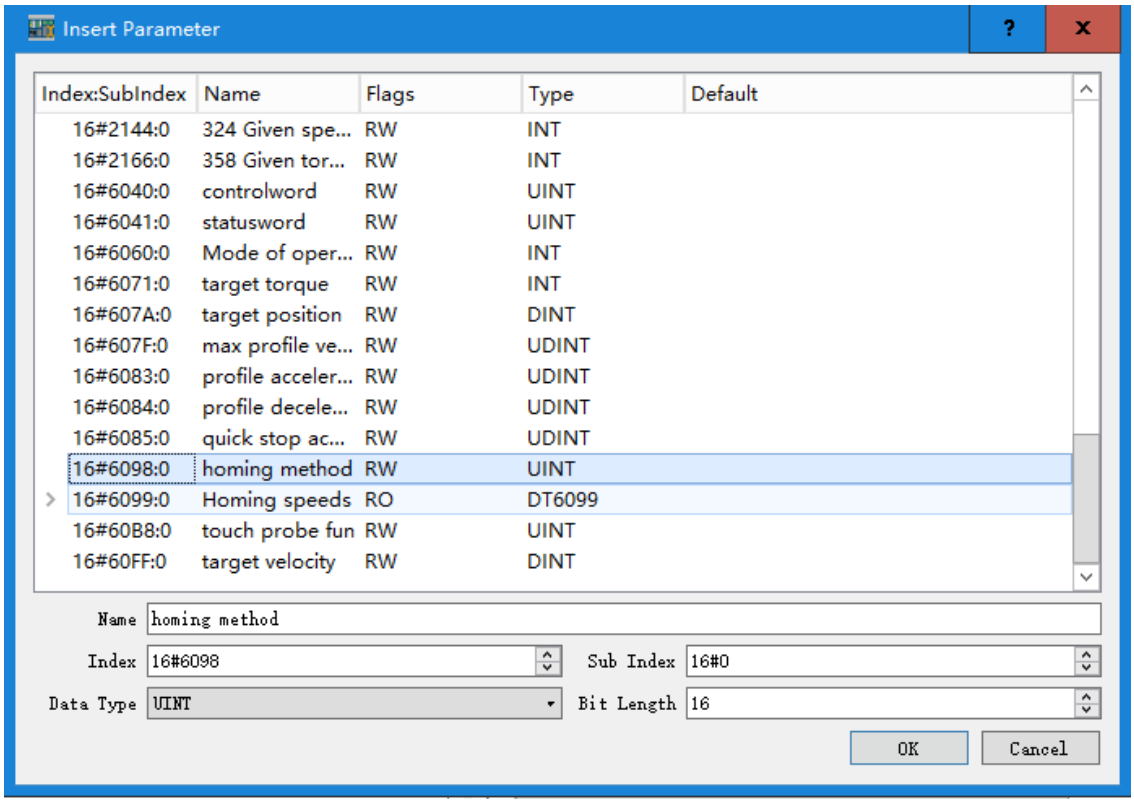
Double-click the slave connected under "EtherCAT BUS", the EtherCAT configuration interface appears.





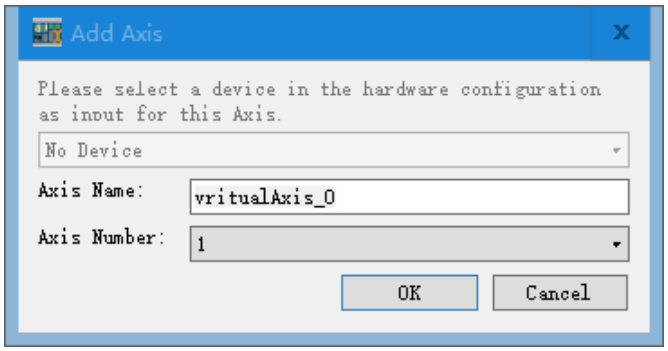
6) Click "Enable Expert Settings", right-click and insert the parameters 16#6098:0 (homing mothod), 16#6099:1 (Speed during search for switch), 16#6099:2 (Speed during search for zero).



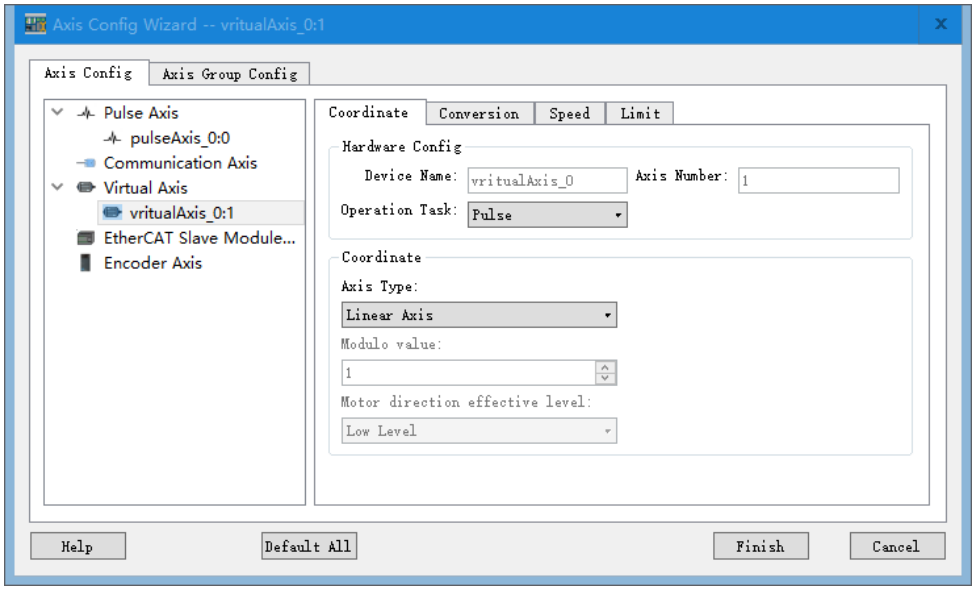


3. Virtual axis configuration

In the Axis Config Wizard interface, right-click the "Virtual Axis" and select Add axis. In the following interface that pops up, you can choose to modify the axis name, and click "OK" to complete the virtual axis addition.

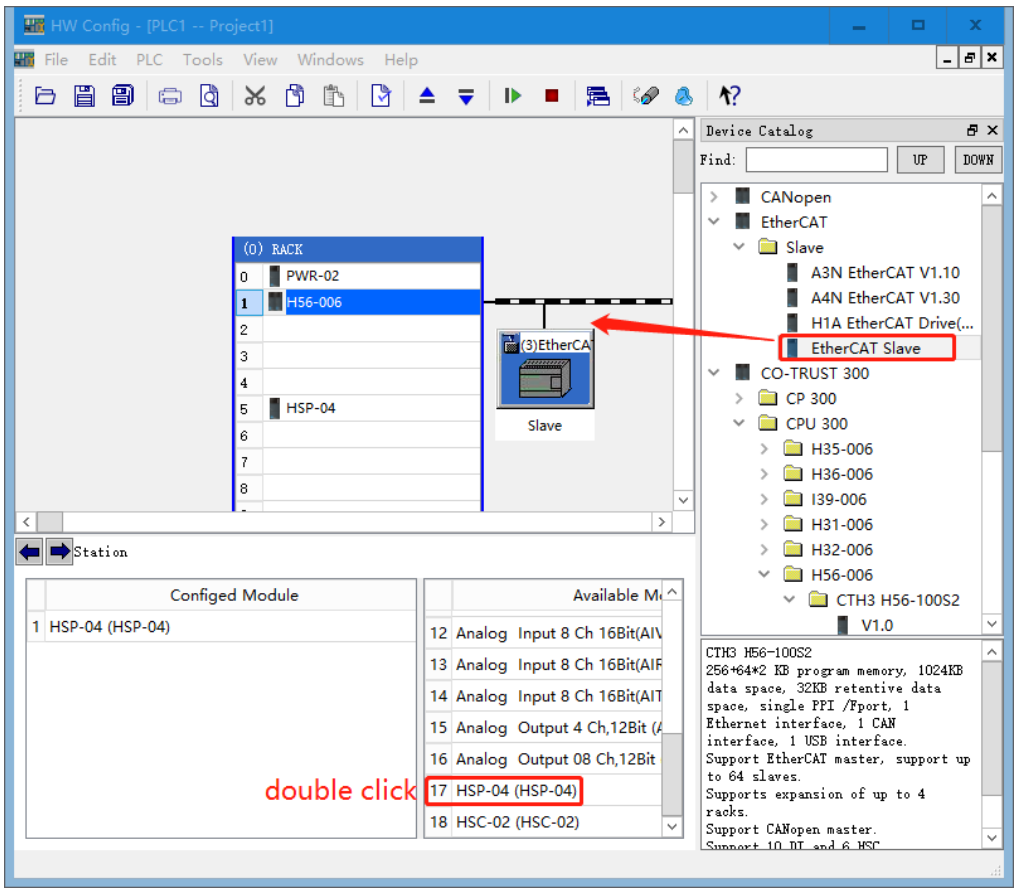


The added virtual axis interface is as follows:

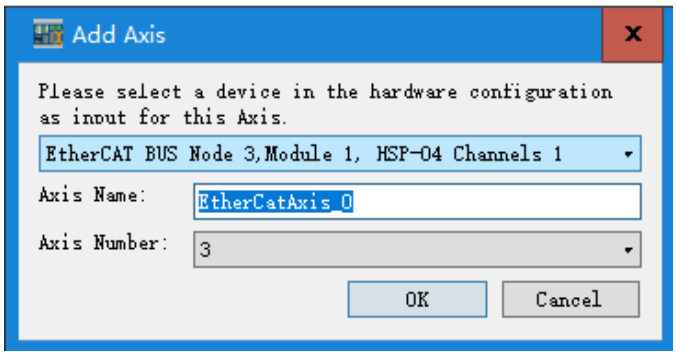


4. EtherCAT Slave Module Axis configuration

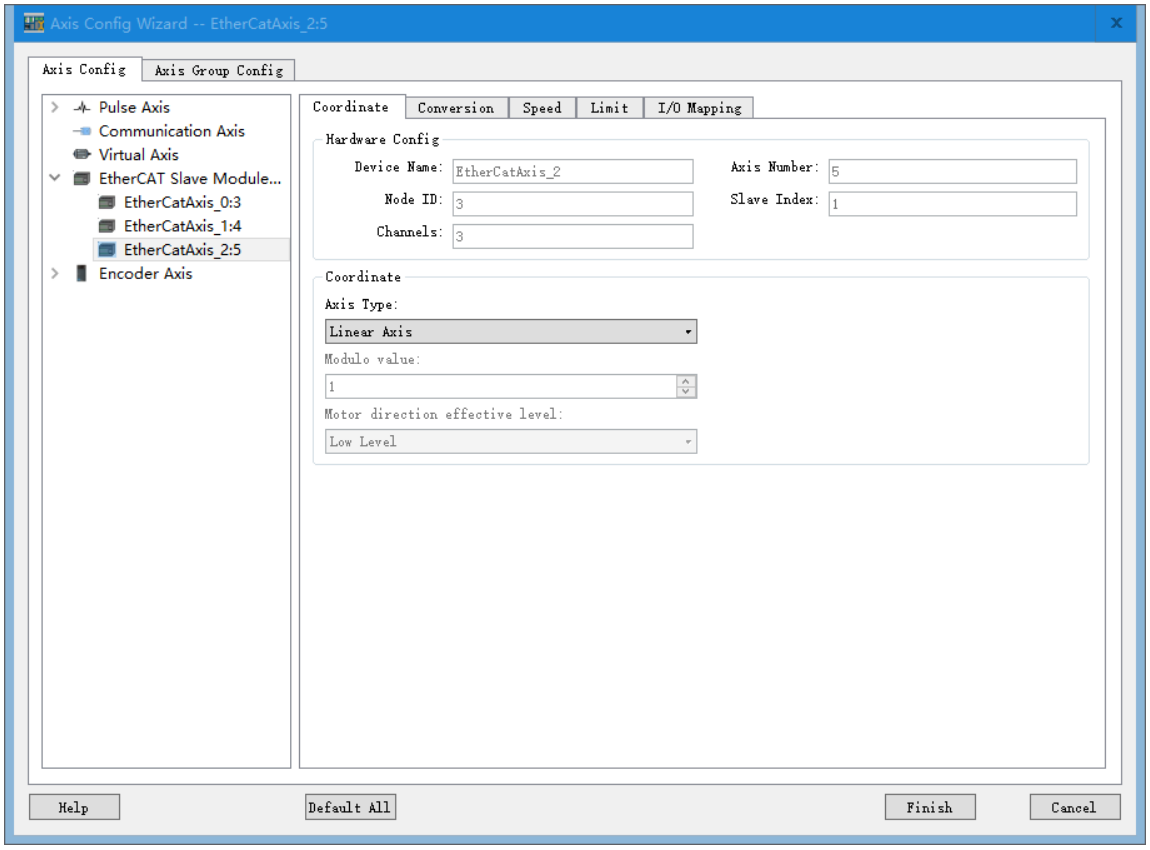
Before adding the EtherCAT slave module axis, you need to add the EtherCAT slave in the hardware configuration interface first, and choose to configure the HSP module for the added EtherCAT slave. As shown below:



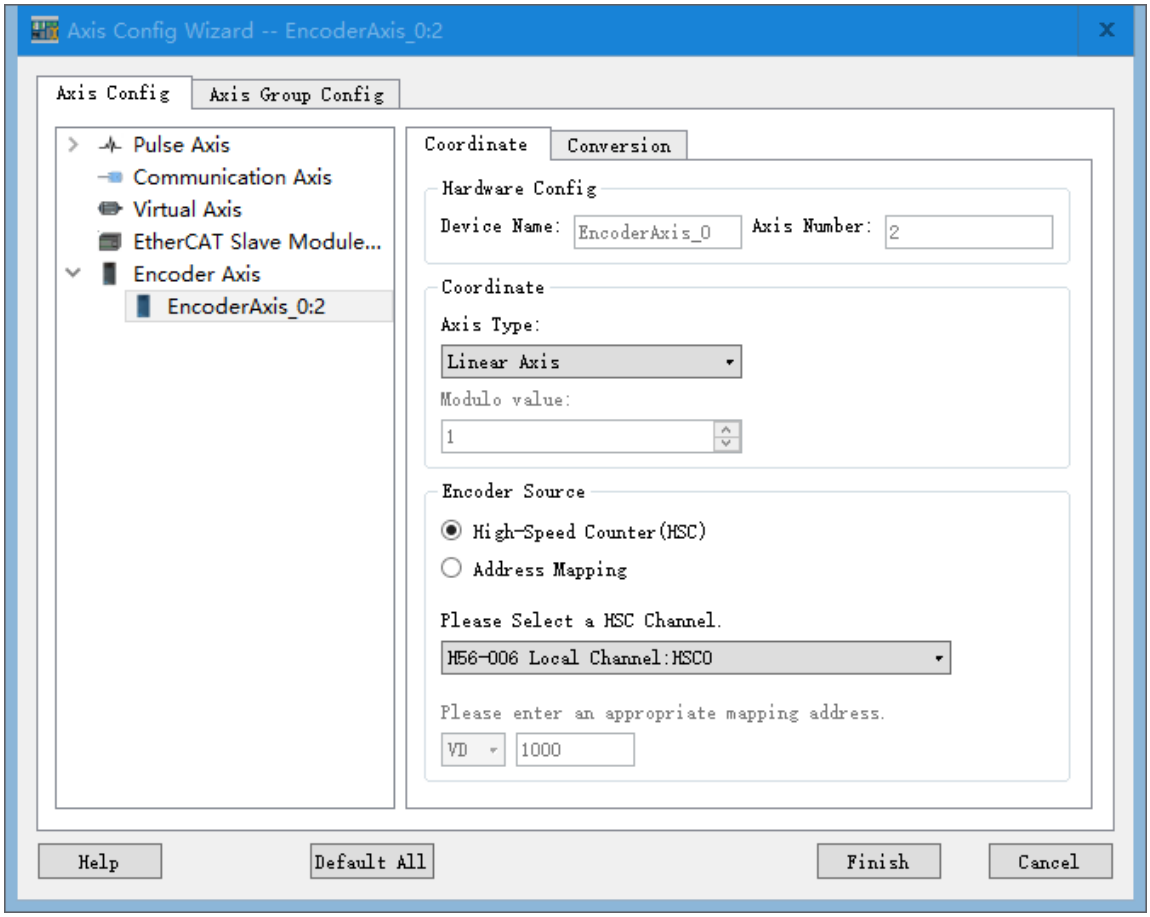
Right-click on the "EtherCAT Slave Module Axis" in the "Axis Wizard" and select "Add Axis", After naming, click "OK" to complete the addition.



The added EtherCAT slave module axis is as follows:



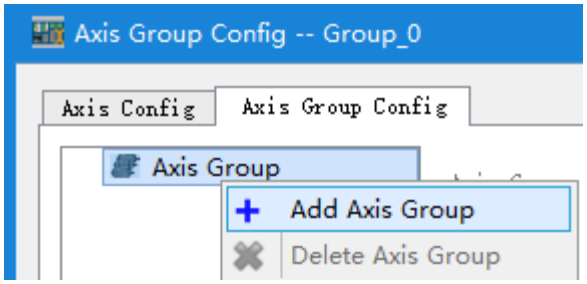
5. Encoder Axis configuration



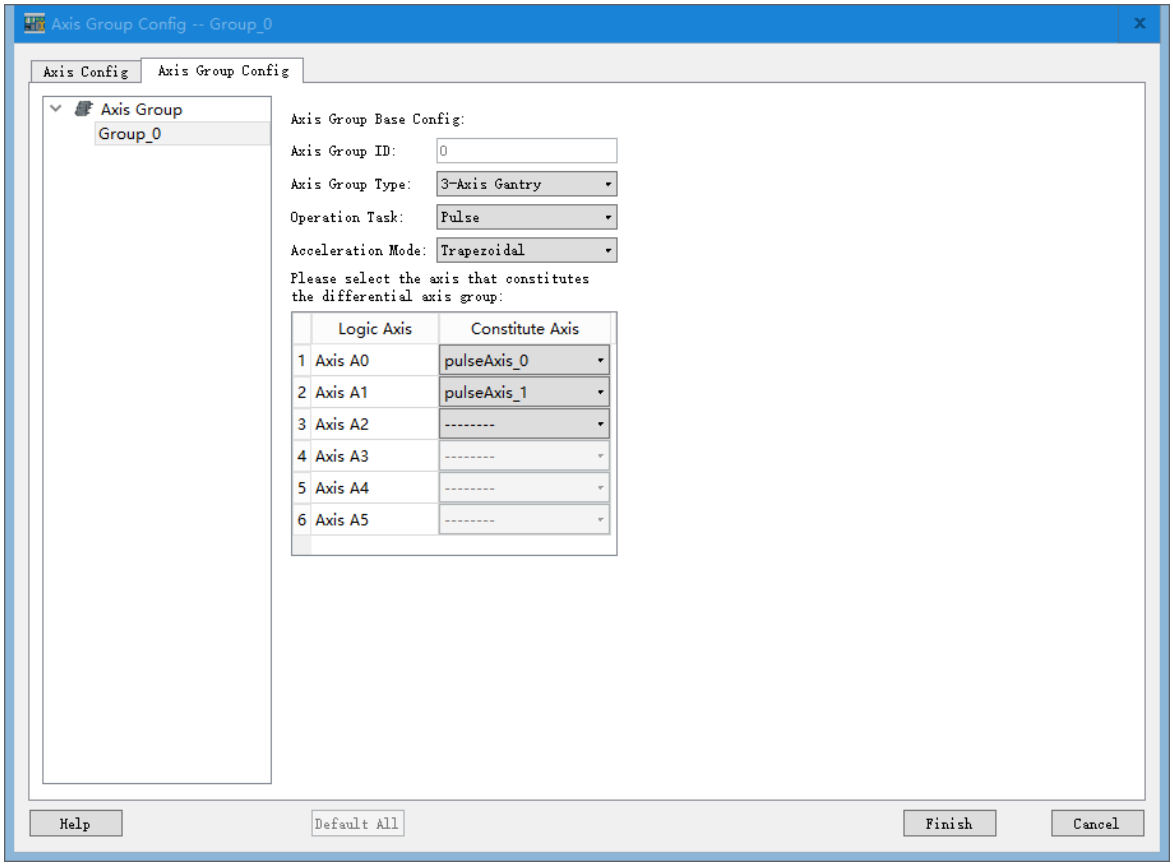
Add Encoder Axis. Under the "Encoder Source" option, select the corresponding encoder for the axis, and you can choose to enter the corresponding mapping address (default V area) to determine the encoder.

16.1.2 Axis t Group Configuration

The axis group configuration is about to configure two or three groups of axes freely in the form of axis groups. Right-click "Add axis group" to add the axis group in the "Axis group configuration".



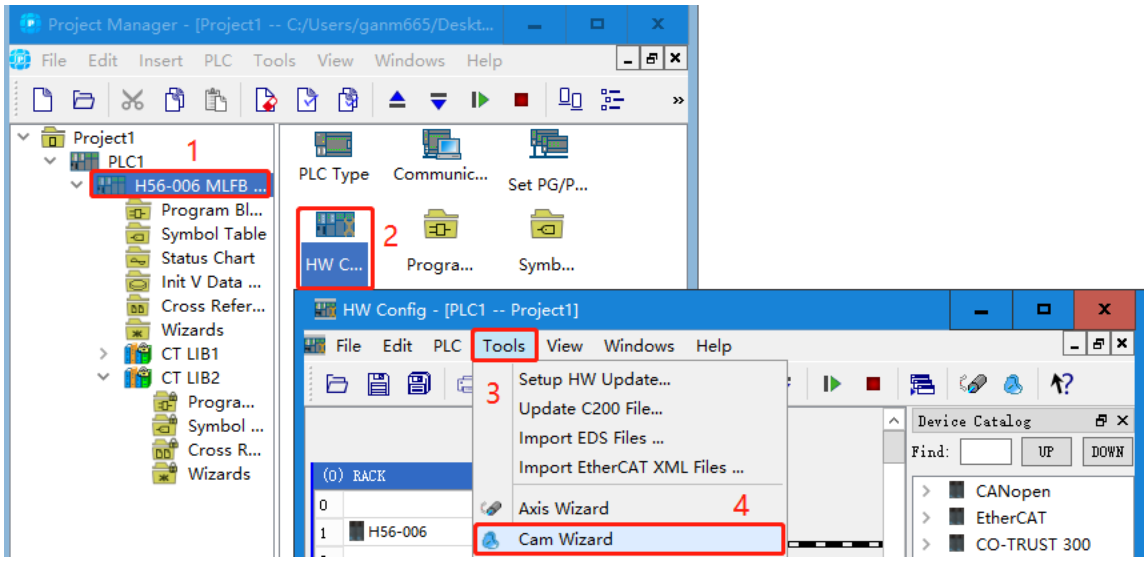
The axis group configuration is as follows:



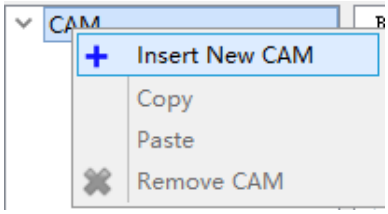
- "Axis group Basic Config": "Axis group ID" cannot be changed, "Axis group type" includes "2-axis Gantry" or "3-axis Gantry", Corresponding to two or three groups of "constitute axis ". "Operation Task" can be "Pulse" or "EtherCAT". "Acceleration mode" can be "trapezoidal", "sin*2" and "quadratic".
- "Logical Axis": There are six groups by default and cannot be changed.
- "Constitute Axis": "2-axis Gantry " corresponds to two component axes, and "3-axis Gantry" corresponds to three component axes. You can choose "pulse axis" or "virtual axis".

16.2 CAM Wizard

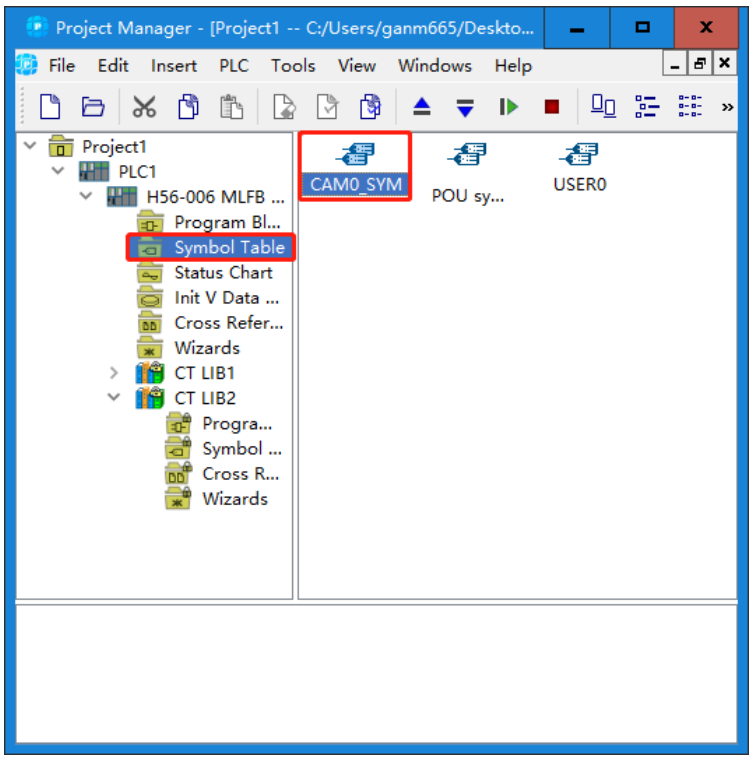
In the electronic cam editor, the electronic cam disc function (or cam switch) can be realized in the form of graphics or list. When the code is generated according to the relevant application, various global data structures (CAM data) will be created, which can Accessed by IEC programs. In the hardware configuration interface, select "Tools"→ "Electronic Cam Wizard" to configure the electronic cam.



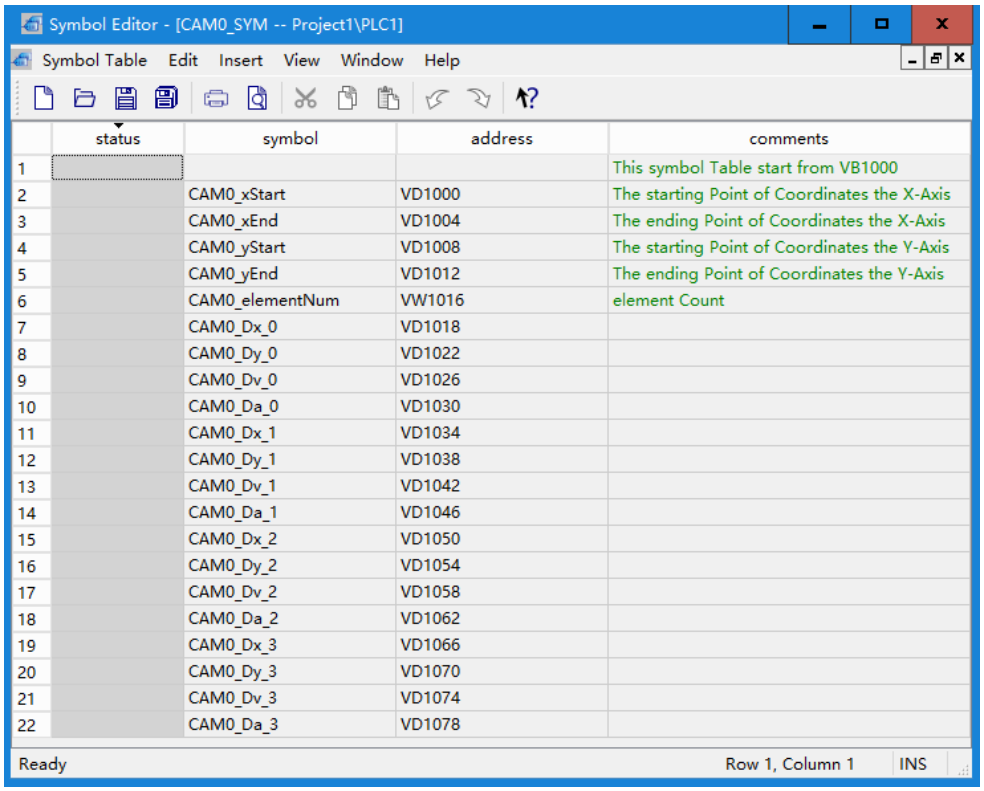
Right click "CAM" to insert a new CAM:



After successfully inserting the CAM, select "Symbol Table" on the project manager interface to see the newly added CAM.

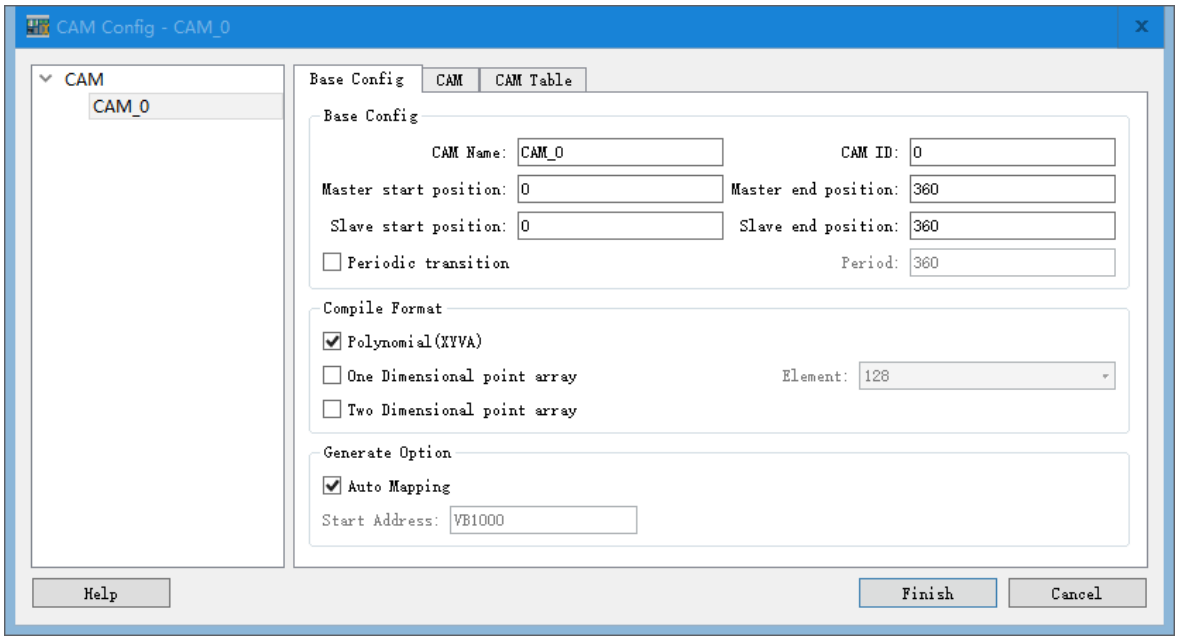


Click "CAM0_SYM" to view the symbol address of the corresponding CAM.



	status	symbol	address	comments
1				This symbol Table start from VB1000
2		CAM0_xStart	VD1000	The starting Point of Coordinates the X-Axis
3		CAM0_xEnd	VD1004	The ending Point of Coordinates the X-Axis
4		CAM0_yStart	VD1008	The starting Point of Coordinates the Y-Axis
5		CAM0_yEnd	VD1012	The ending Point of Coordinates the Y-Axis
6		CAM0_elementNum	VW1016	element Count
7		CAM0_Dx_0	VD1018	
8		CAM0_Dy_0	VD1022	
9		CAM0_Dv_0	VD1026	
10		CAM0_Da_0	VD1030	
11		CAM0_Dx_1	VD1034	
12		CAM0_Dy_1	VD1038	
13		CAM0_Dv_1	VD1042	
14		CAM0_Da_1	VD1046	
15		CAM0_Dx_2	VD1050	
16		CAM0_Dy_2	VD1054	
17		CAM0_Dv_2	VD1058	
18		CAM0_Da_2	VD1062	
19		CAM0_Dx_3	VD1066	
20		CAM0_Dy_3	VD1070	
21		CAM0_Dv_3	VD1074	
22		CAM0_Da_3	VD1078	

After inserting the CAM, the configuration interface is shown below.



[Base Config]

- "Base Config": The user can modify the "Cam name" and the start and end positions of the master and slave (the position of the master corresponds to the abscissa in the CAM graph, and the position of the slave corresponds to the ordinate in the CAM graph), check "Periodic change", the graph in CAM will change periodically, and the end position of the master is the period.
- "Compile Format": Check "Polynomial (XYVA)", CAM will display the horizontal and vertical coordinates, speed and acceleration status in the status table with the symbols of x, y, a, and v. Currently, it does not support "One Dimensional point array" and "Two Dimensional point array".
- "Generate Options": Check "Auto Mapping" to set the default starting address to VB1000. Uncheck, you can modify the start address, which corresponds to the start address in the symbol table.

The starting address of the cam data is specified in the basic configuration. After configuration, a data block will be generated (the function is temporarily not implemented), or the starting address (V memory, symbol table) will be specified. A block of memory after the start address stores CAM table data. When the polynomial mode is selected for the cam table, the data records are as follows:

Number	Name	Type	Remark
1		STRUCT	
2	xStart	REAL	The starting point of the X axis
3	xEnd	REAL	X coordinate endpoint
4	yStart	REAL	The Y coordinate starts
5	yEnd	REAL	Y-coordinate endpoint
6	nElements	WORD	Number of elements
7	dX0	REAL	The 0th x-coordinate
8	dY0	REAL	The 0th y-coordinate
9	dV0	REAL	The zero point velocity
10	dA0	REAL	Acceleration at the 0th point
11	dX1	REAL	The 0th x-coordinate
12	dY1	REAL	The 0th y-coordinate
13	dV1	REAL	The zero point velocity
14	dA1	REAL	
15	dX2	REAL	

	...		
		END_STRUCT	

The user can modify the data in the memory to change the cam table data. After modifying the data, call MC_CamTableSelect to notify the PLC of the change in the cam table data. If the system is in the running of the cyclic cam table at this time, the changed data will be in the next cam cycle to take effect, if it needs to take effect immediately, call the MC_CamIn instruction immediately after calling MC_CamTableSelect.

Cam Offset and Scaling:

The position of the spindle input transformation is made according to the following formula, and the transformed X is the output of the cam:

Calculation formula: **X=MasterScaling*MasterPosition+MasterOffset**

Therefore, if the ratio of the main shaft is greater than 1, the cam will run at a higher rate, and if the ratio is less than 1, the speed will be reduced accordingly.

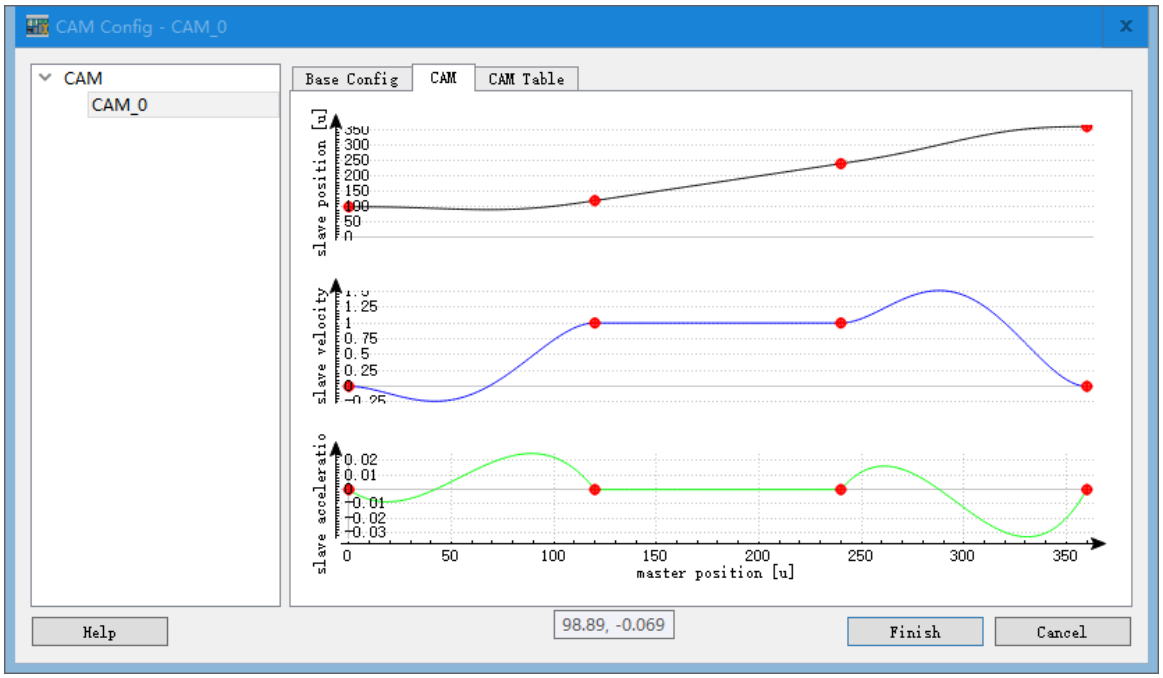
Slave Offset, SlaveScaling:

Calculation formula: **Y=SlaveScaling*CAM(X)+SlaveOffset**

If SlaveScaling > 1 causes a stretch of the cam effect, the range of the slave axis will increase; if SlaveScaling < 1 it will cause a shrink.

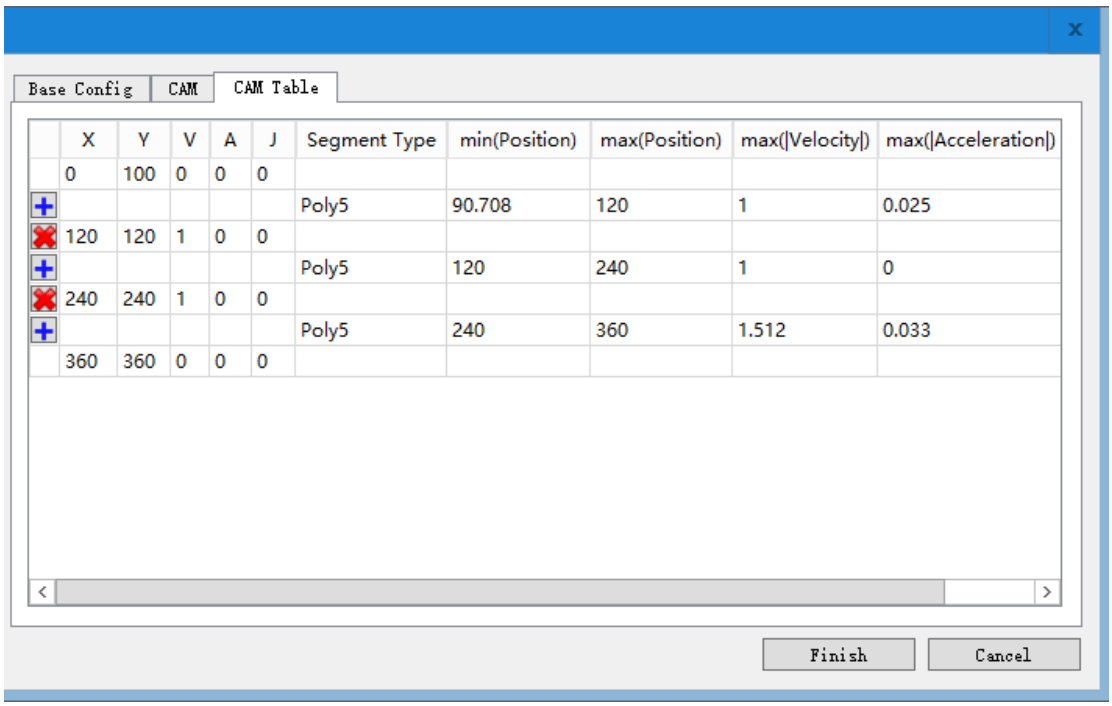
【CAM】

This page displays the relationship between "master position" and "slave position", "slave velocity" and "slave acceleration" in the form of a graph. The maximum value of the “master position” on the abscissa corresponds to the setting value of the “master end position” on the “basic configuration” interface, and the maximum value of the “slave position” on the ordinate corresponds to the setting value of the “slave end position” on the “basic configuration” interface. Click the red node to drag it.



【CAM Table】

This page corresponds to displaying the information of the red nodes in the curve graph of the CAM. A red node corresponds to a set of data. Click "+" to add a new node, and click "✖" to delete the added node. Double-click the corresponding value of "X, Y, V, A" to modify it manually.



	X	Y	V	A	J	Segment Type	min(Position)	max(Position)	max(|Velocity|)	max(|Acceleration|)
	0	100	0	0	0					
+						Poly5	90.708	120	1	0.025
✖	120	120	1	0	0					
+						Poly5	120	240	1	0
✖	240	240	1	0	0					
+						Poly5	240	360	1.512	0.033
	360	360	0	0	0					

C Language Program

17

This section introduces the specific application of C language programming of H56/H52 combined with the programming software MagicWorks PLC

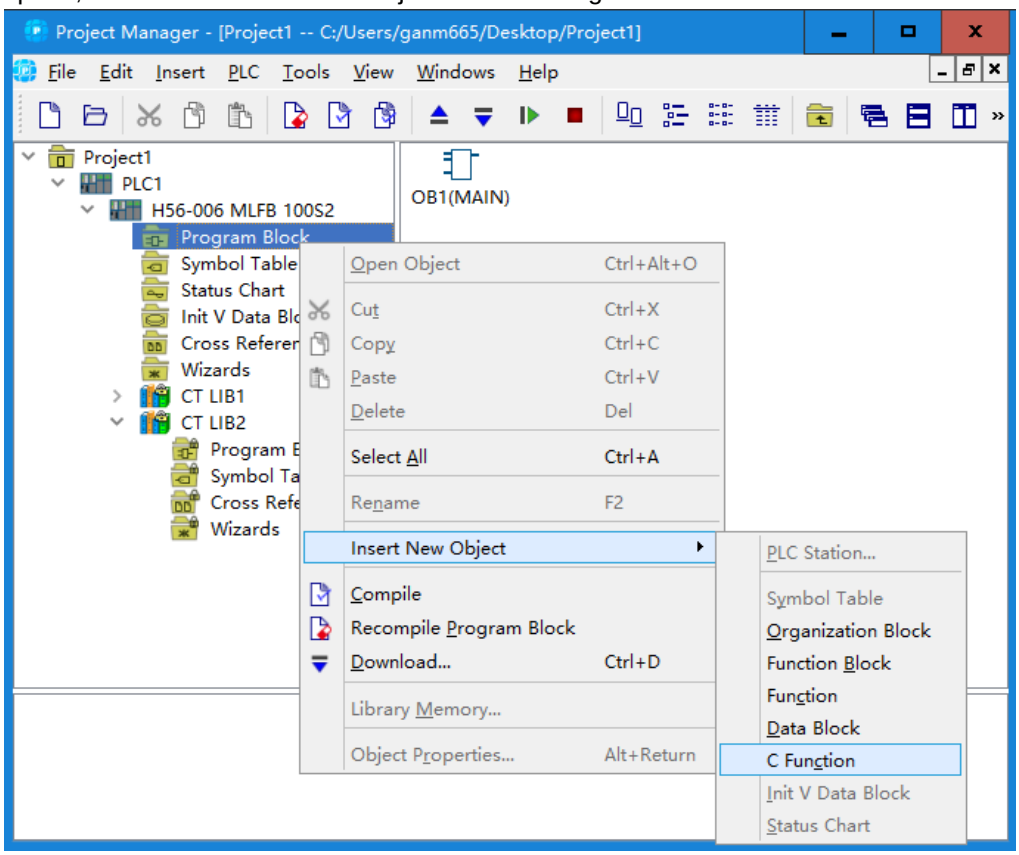
C function programming language is an upgrade and enhancement of MagicWorks PLC software functions. On the basis of the original ladder diagram and STL two programming languages, C function programming is allowed to work together with the ladder diagram and other programming languages. Using C language to write function blocks can enhance the readability and maintainability of the program. At the same time, the rich operation functions of C language can realize various calling functions, which saves the internal space and improves the programming efficiency. Currently H56-10/H52-10/H36-01/H32-01 supports C function programming language.

This chapter introduces the use of C language programming functions of MagicWorks PLC. The specific contents are as follows:

- C language programming steps in Magicworks PLC
- C language foundation
- Operators and expressions
- for loop statement
- Select statement
- Introduction of libplc300.a library

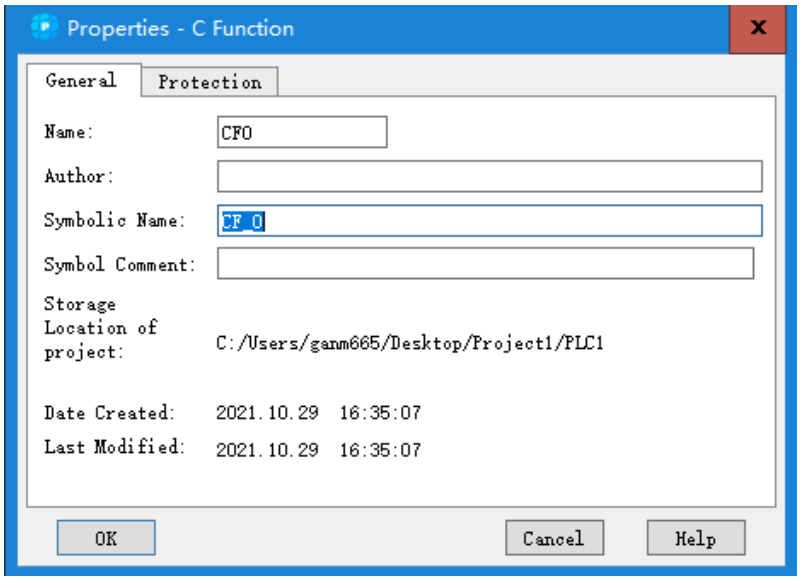
17.1 C language Programming Steps in Magicworks PLC

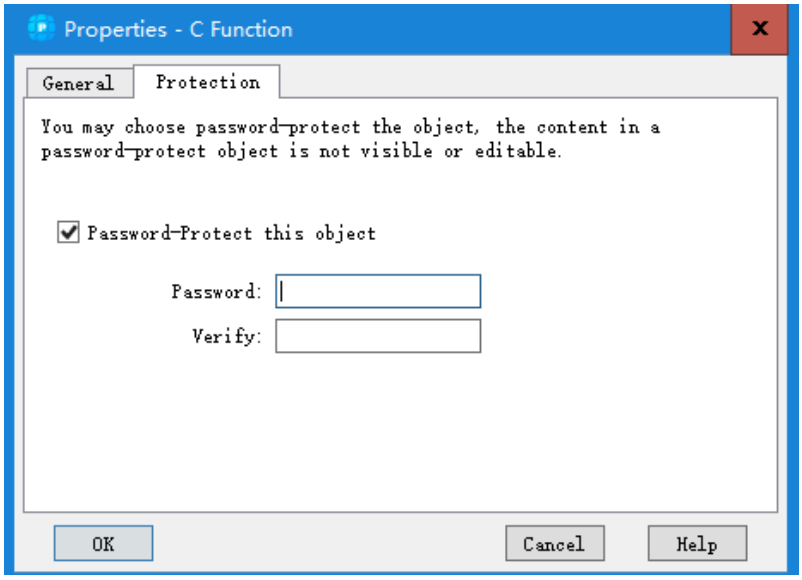
1. Create a new project in MagicWorks PLC, select "Program block", right-click on the blank space, and select "Insert New Object" → "C Program Block".



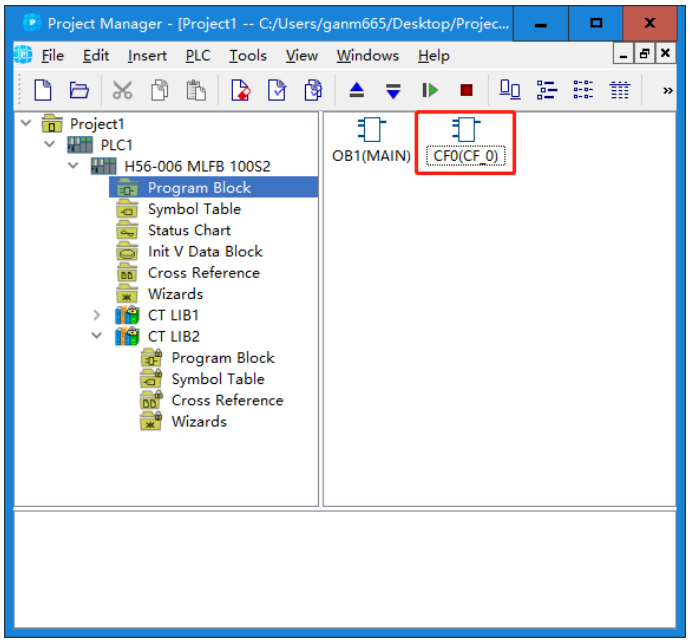
2. Edit the properties of the C program. The symbol name defined here will automatically be used

as the C function name. It is recommended not to define it in Chinese. When copying the code, the symbol name of the CF block must be the same as the C language function name, otherwise it will not be executed. If the engineer wants to protect the code, he can also refer to the following steps to increase the password protection.

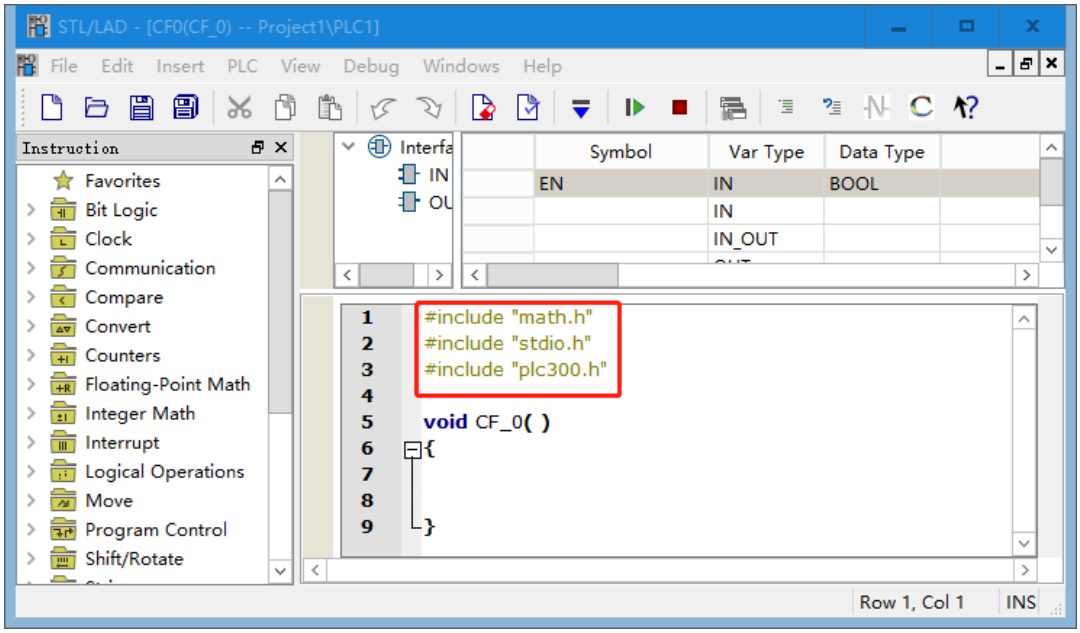




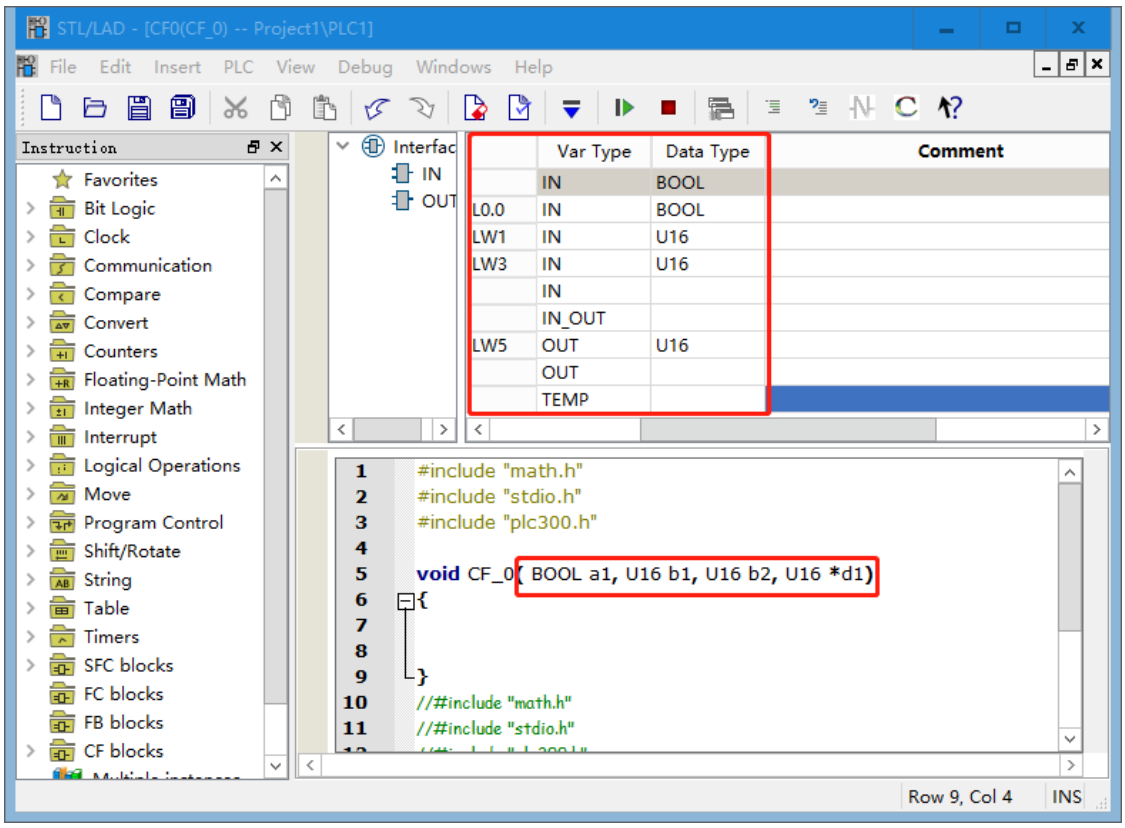
3. After the function is created successfully, you can see CF0 in the program block.



4. Double-click CF0 to enter the C function for programming. MagicWorks PLC will add the following three header files by default after creating CF0.



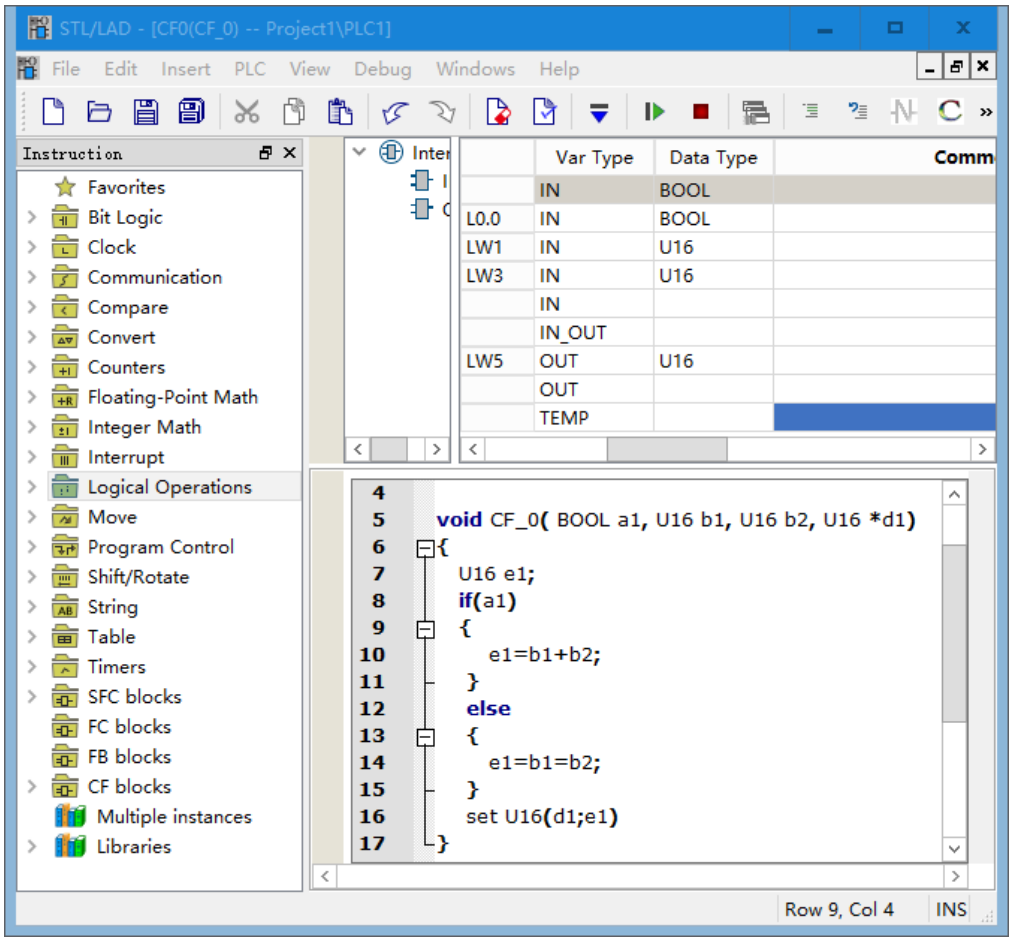
5. Define the external input or output parameter values in the local variable window. After all the definitions are completed, click "C" to initialize the C program block and regenerate the function template. Don't change the name of C function and function parameters casually.



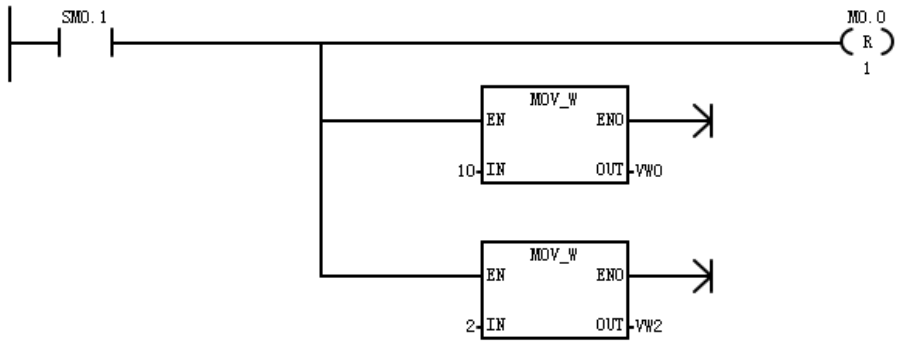
<Remarks> The local variable window can only define up to 16 variables.

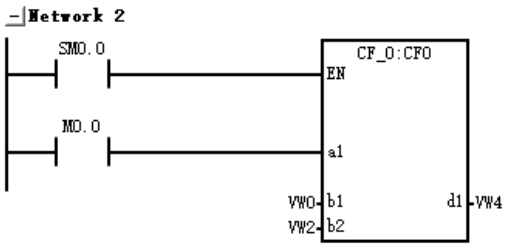
6. Program in the generated template. After programming, click to compile. If there is no error, it can be called by OB1 or other FCs.

The input parameters defined in the parameter symbol table will convert the input parameters of the corresponding type in C language, and the parameters can be used directly. The input/output and output parameters will become pointers of the corresponding type, and the results will be written back to the STL system through the pointer, that is, the C function has no return value, and the return value can only be returned through the pointer.



7. After the subprogram is programmed, it needs to be called in the OB1 main program or other executed FCs. When the program block is downloaded, the C subroutine will also be downloaded to the PLC. (Note: The C function cannot be monitored online, and the input and output can be monitored at the call of the ladder diagram). The function executed by the following program is: if M0.0=0, then vw4=vw0-vw2. if M0.0=1, then vw4=vw0+vw2.





17.2 Basics of C Language

17.2.1 Header files

The compiler provides many standard functions declared in header files. The header file is also called the include file. As a carrier file containing the declaration of function functions and data interfaces, the header file is mainly used to save the declaration of the program.

<stdio.h>
<stdarg.h>
<math.h>
<stdlib.h>
<string.h>
<stdbool.h>
<complex.h>
<ctype.h>
<wctype.h>

The header file contains declarations of various functions and variables. MagicWorks PLC will add the following three header files by default after creating a new C program block.

```
#include "math.h"
#include "stdio.h"
#include "plc300.h"
```

The header file < stdio. H > contains a large number of declarations of advanced input / output (I / 0) functions. If the program contains input and output operations, this header file must be used. The header file < math. H > declares some mathematical operations, such as power, square operation, trigonometric function, logarithm and other functions, which can be called directly.

plc300.h is the header file of the library libplc300.a developed by COTRUST. Through the library libplc300.a, users can call the underlying functions to interact with STL PLC. The functions that users can use can be viewed in the header file plc300.h. The header file plc300.h is in the programming software installation directory \ccompile\include\plc300.h.

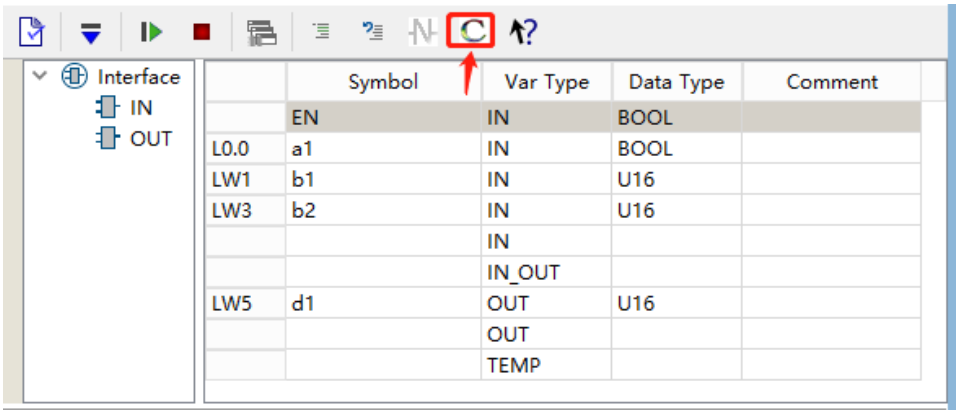
17.2.2 Variable

1. Define variables

The general form of variable definition: data type variable name

```
E.g:   FP32     real1.              //Defines a 32-bit floating point number
       S16      int1.                 //Define 16-bit signed integer
 const FP32 PI=3.1415926.       //Define 32-bit floating point constant PI
```

For the parameter value defined in the local variable window, execute "initialize C program block", and the parameter will be automatically defined in the subroutine.




```
1  #include "math.h"
2  #include "stdio.h"
3  #include "plc300.h"
4
5  void CF_0( BOOL a1, U16 b1, U16 b2, U16 *d1)
```

Notice:

• The variable name must start with a letter or an underscore, and the middle of the name can only consist of letters, numbers and an underscore "_". the last character can be a type specifier.

• Variables must be defined before use.

• A variable name can only be defined once. It is recommended to define variables in the program header. Chinese variables are not allowed for variable definitions, and they are strictly case sensitive.

• Some simple character variable names are easy to be confused with some original counters. For example, C represents a counter, and C1 is not allowed.

2. Static local variables and ordinary variables

Differences between local variables and ordinary variables

1) Memory allocation and release

• Ordinary local variables allocate space only when the statement defined by the variable is executed.

• Static local variables are in the compilation phase (the function has not been executed), and the variable space has been allocated.

• Ordinary local variables leave the scope {} and automatically free their space, so they cannot be used.

• Static local variables are automatically released only when the entire program ends.

2) Initialization

• Ordinary local variables are not initialized and are random values.

• The static local variable is not initialized and is 0.

• The static local variable initialization statement is only valid the first time it is executed, but it can be assigned multiple times.

• Static local variables can only be initialized with its constants.

The variable allocates memory space in the global data area. the compiler automatically initializes its scope to the local scope, and when the function that defines it ends, its scope ends.

The programming example is as follows: As long as var1 is not equal to 0, add one to var2.

```
void CF_3( S16 var1, S16 *var2 )
{
   static S16 temp.
```

```
    if(var1!=0)
    {
       temp=temp+1.
    }
    setS16(var2,temp).
}    // The temp variable uses the static modifier
```

17.2.3 Annotations

Comments help to understand the code. Comments are not compiled and will not affect the execution of the program. There are two ways of commenting in C language:

- Block comment starting with / * and ending with * /（block comment）
- A single line comment that starts with / / and ends with a newline character (line comment)

In magicworks PLC, either manual annotation or annotation key can be used for annotation. Click the comment key in the menu bar to comment, and click the cancel comment key in the menu bar to cancel the comment.

17.2.4 Mixed operations between various types of numerical data

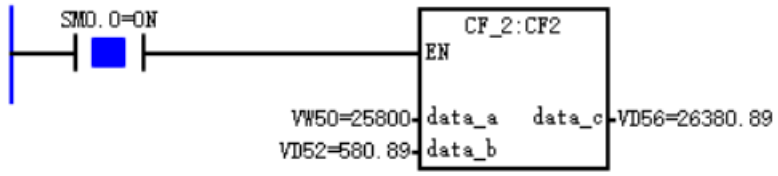
When performing calculations, different data types need to be converted into the same type of data, and then the calculations are performed.
Operation process: first automatically convert into similar data, and then calculate.
Conversion rule: low byte type is converted to high byte type.

For example: c = S16 a + FP32 b, a will be automatically converted to FP32 data type, and then added to b, the result is a FP32 type c.

C language programming is as follows:

```
#include "math.h"
#include "stdio.h"
#include "plc300.h"
void CF_2( S16 data_a, FP32 data_b, FP32 *data_c )
{
   FP32 temp.
   temp=data_a+data_b.
   setFP32(data_c,temp).
}
```

The CF block obtained after compilation is called through OB1 or other FC:



17.2.5 Coercion Type Conversion

To convert the value of the specified expression to the specified type, the programmer needs to specify the type to be converted.

Form: (type name) (expression)
For example: (S16)(a+ b)

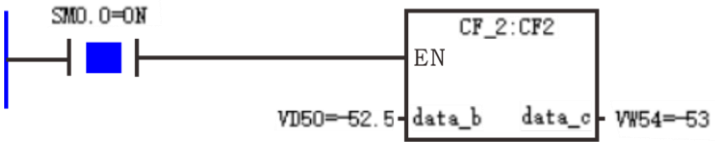
Note: 1) Type names and expressions need to be enclosed in parentheses

2) During the type conversion, an intermediate variable of the required type is obtained, and the original variable type has not changed.

C language programming example: round a 32-bit floating point number and convert it to a 16-bit integer.

```
void CF_2(FP32 data_b, S16 *data_c )
{
   S16 temp.
   If (data_b>0)
   {
      temp=(S16)(data_b+0.5).
   }
   else
   {
      temp=(S16)(data_b-0.5).
   }
   setS16(data_c,temp).
}
```

The CF block obtained after compilation is called through OB1 or other FC:



17.2.6 Array

An array is a collection of ordered data. If a limited set of variables of the same type is named, the name is the array name. The data type of each element in the array is the same, and the user determines the value in the array element by the array name and subscript.

One-dimensional arrays are defined by: type specifier array name [integer constant expression]

Example: FP32 real0[10].

Description: An array is defined, the name of the array is real0, there are 10 elements, and the type of each element is FP32. The 10 elements are real[0], real[1], real[2], ... real[8], real[9].

Initializing an array: Arrays can be initialized in the declaration. When initializing the array, the values of each index variable should be enclosed in {} and separated by commas, for example: S16 aa[3]={2,6,1}, equivalent to: S16 aa[3].aa[0]=2.aa[1]=6.aa[2]=1.

Notice:

• Subscripts start at 0. For example: the first element of the array is real1[0], and the eighth element is real1[7].

• The C language generally does not allow dynamic definition of the size of the array, such as FP32 real1[n], which is not allowed.

• The language does not check that the array subscript is out of bounds, so do not use it out of bounds. In this example, the number of array elements is 10, and the subscript range is: 0-9, and cannot exceed 9.
• The elements in the array are stored consecutively in the memory in sequential order.

C language programming example: copy 10 consecutive values from VD100 to VD200.

```
void CF_0( void* real1, void* real2 )
{
   S16 i,j.
   FP32 temp[10] ={0,0,0,0,0,0,0,0,0,0}.     //Define the array
   for (i=0.i<10.i++)
      {
         temp[i]=getFP32(real1+4*i).              //Array address
      }
   for (j=0.j<10.j++)
      {
         set FP32((real2+4*j),temp[j]).
      }
}
```

call the generated CF0

17.3 Operators and Expressions

17.3.1 Assignment Operators and Assignment Expressions

1. Assignment operator =

Assignment Expression: Assign a value to a variable or assign the value of one variable to another variable.

Assignment form: variable name = constant or expression.

Assignment: Assigns the value of the constant or expression on the right to the variable on the left. Such as: a=3, a=b*c

Notice:
• All statements, including assignment statements, must end with a semicolon ".".
• After the assignment, the original value on the left is replaced by the value of the expression on the right.
• If the value type of the expression on the right is inconsistent with the type of the variable on the left, the type of the variable on the left shall prevail, and the type of the value of the expression on the right is automatically converted to the type of the variable on the left.

2. Multiple assignment operator: Multiple assignment is to assign the value of the expression to

several variables at the same time . In this case, the assigned variable names should be written in order on the left side of the assignment number.
Assignment form: variable1=variable2=…=variable n=expression
Equivalent to: variable1=(variable2=(…=(variablen=expression)))
For example: a=b=c=7.

3. Compound assignment operator: Add other operators before the assignment character "=".
Common ones are: +=, -=, *=, /=, %=
The function of compound assignment operation: perform compound operation with the variable on the left and then assign it to the variable on the left.
For example:

```
a+=3.                    Equivalent to a = a+3
x*=y+8.                  Equivalent to x = x*(y+8)
x%=3.                    Equivalent to x = x%3
int a=1,b=2,c=2.  c*=a-b.  Equivalent to c=c*(ab)=2*(-1)=-2
```

17.3.2 Arithmetic Operators

Arithmetic operators in C language are :

+: plus or positive operator

-: Subtract or Negative operator

*: Multiplication operator

/: division operator

%: remainder operator
Among them, addition, subtraction, multiplication and division are remainder operators, and they require two operands.

17.3.3 Self-Increasing and Self-Decreasing Operators

++ means self-increment, the variable value is incremented by 1,
-- Indicates self-decrement, and the variable value is reduced by 1.

Usage 1: Used alone as an expression. For example: i ++ or ++ i is equivalent to i = i+1

Usage 2: Appears in other expressions to participate in the operation, ++ i /-- i means that i is incremented/decremented by 1 and then participates in other operations. and i ++/ i -- means that after i participates in the operation, the value of i is incremented/decremented by 1 again.

For example : S16 j,i=3.

```
    j=++i.      // the value of i,j is 4
    j=i++.      // the value of j is 4, then the value of i becomes 5
    j=-(i++).   // the value of j is -5, then the value of i becomes 6
```

17.3.4 Relational Operators and Expressions

Relational operation: Compare whether two operands satisfy a given relationship, which is used for simple conditional judgment. In C language there are the following relational operators:

(1) < less than

(2) <= less than or equal to

(3) > greater than

(4) >= greater than or equal to

(5) == equal to
(6) != not equal
The result of the operator can only be 0 or 1, 1 means true and 0 means false

For example: 3.24<=2.98. //The value is false, and the operation result is 0
0.5 != 3+1. //The value is true, the result of the operation is 1

Relational expression: An expression that connects two expressions (arithmetic expressions, relational expressions, logical expressions, assignment expressions, and character expressions) with relational operators is called a relational expression.
The general form of a relational expression is: expression relational operator expression.

The precedence of relational operators is lower than that of arithmetic operators and higher than that of assignment operators.
For example: a=1+2>3-1. //equivalent to a=((1+2)>(3-1)), the result is a=1

17.3.5 Logical Operators and Expressions

Logical operations, also known as Boolean operations, are used for complex conditional judgments.

operator	meaning	priority	ST description
!	logical not	1	NOT
&&	logical and	2	AND
ll	logical or	3	OR

General form of logical expressions: expressions logical operator expressions

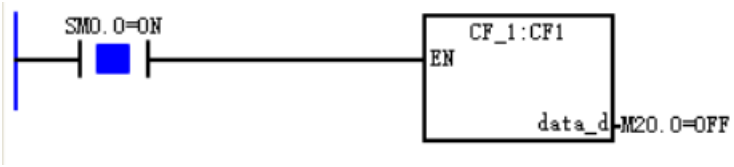
Truth table of logical operations

A value	B value	! A	A&&B	AllB
non-zero	non-zero	0	1	1
non-zero	0	0	0	1
0	non-zero	1	0	1
0	0	0	0	0

```
C language programming example: d=1 when the result of (a>b)&&(b>c) is true.
void CF_1( BOOL *data_d )
{
   S16 data_a=3,data_b=2,data_c=1.
   if ((data_a>data_b)&&(data_b>data_c))
   {
      *data_d=1.
   }
   else
```

```
  {
    *data_d=0.
  }
}
```

The CF block obtained after compilation can be called through ob1 or other FC:



17.3.6 Conditional Operators and Conditional Expressions

Conditional operator and conditional expression have two symbols: ? and :, which form a ternary operation with three operands.

General form: <expression1>? <expression2>: <expression3>

Priority: Logic > Condition > Assignment

For example: S16 max ,a =5.b=3.

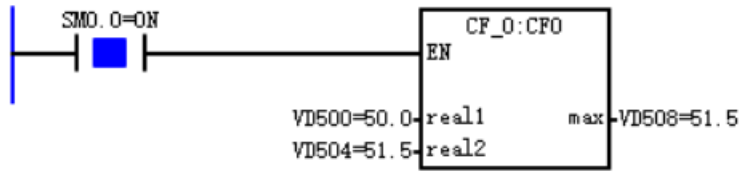
max= a> b?a:b

Result: max= 5

C language programming example: compare the size of 2 values and output the maximum value.

```
void CF_0( FP32 real1, FP32 real2, FP32 *max )
{
 FP32 max_t.
 max_t=real1>real2?real1:real2.
 setFP32(max,max_t).
}
```

The CF block obtained after compilation is called through OB1 or other FC:



17.4 For loop Statement

In an application program, some steps need to be executed many times, this process is called loop. for loop is a kind of loop statement in programming language, the design of loop program needs to establish a program so that it can loop back and execute its own program.

The general form of a for loop statement:

for (one-shot expression. conditional expression. end loop body)

{

Intermediate loop.

}

For example:

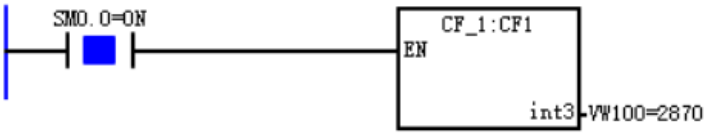
```
for (i=1.i<=100.i++)
        {
          sum=i+sum.
        }
```

The statements in the for loop are repeatedly executed, and the variable i will be automatically incremented by 1 after each loop. i = 1 and i <= 100 are the start and end values of the loop control variables . When the control variable reaches the termination value , the program executes the next instruction after the for statement.

For example: Calculate the sum of the squares of 1-20!

```
void CF_1( S16 *int3 )
{
  S16 int3_t=0.
  S16 i.
  for (i=1.i<=20.i++)
  {
    int3_t=int3_t+i*i.
  }
  setS16(int3,int3_t).
}
```

The CF block obtained after compilation is called through OB1 or other FC:



17.5 Select Statements

17.5.1 If select statement

if selection structure: a selection structure for judging if conditions and performing subsequent operations according to the judgment results.

1) The general format of a single-branch if selection structure:

```
if (expression)        //expression: refers to the judgment condition, true is 1, false is 0
{
execute statement . // execute the statement here only if the expression is true
}
```

For example: input a number, judge whether it is divisible by 4 or 7, and output 1 if the condition is satisfied.

```
void CF_0( S16 int1, S16 *out1 )
{
  S16 out_t=0.
  if (int1%4==0||int1%7==0)
  {
```

```
    out_t=1.
  }
  setS16(out1,out_t).
}
```

The CF block obtained after compilation is called through OB1 or other FC:

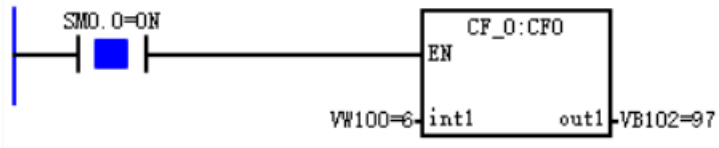
2) The general format of the double-branch if selection structure:

```
if (expression) //expression: refers to the judgment condition, true is 1, false is 0
 {
    execute statement . // execute the statement here only if the expression is true
 }
else
{
    Execute statement . //condition is 0, execute the statement after else
}
```

For example: input a number, even output a, odd output b.

```
void CF_0(S16 int1, U8 *out1 )
{
  char out.
  if (int1%2==0)
  {
    out='a'.
  }
  else
  {
    out='b'.
  }
  setU8(out1,out).
}
```

The CF block obtained after compilation is called through OB1 or other FC:

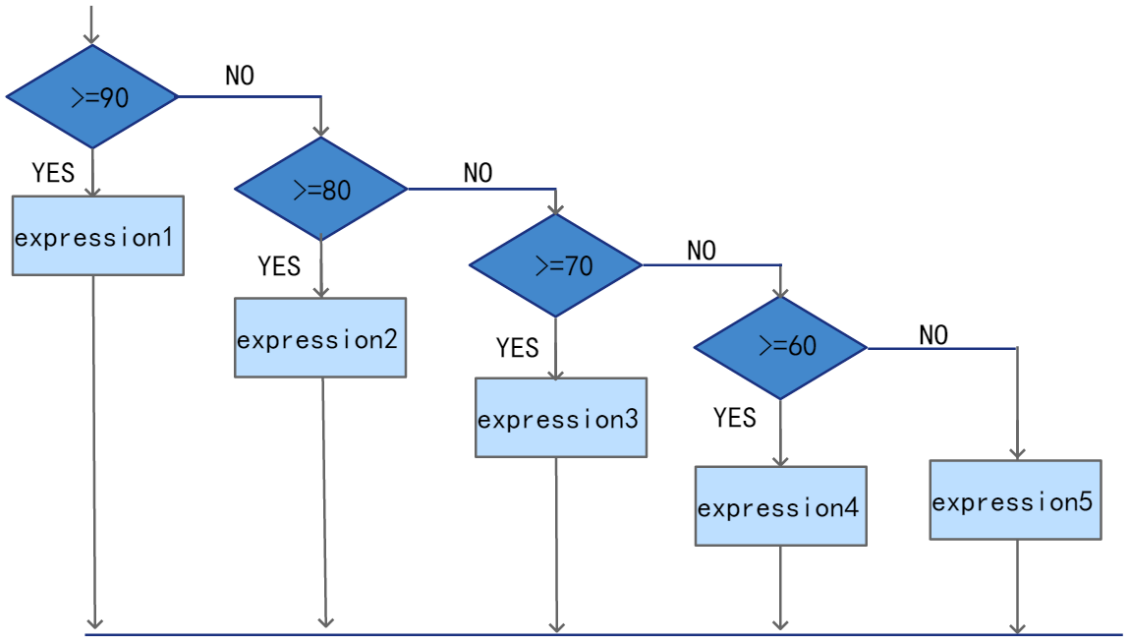


3) The general format of the multiple if selection structure :

```
if (expression)
{
  execute statement.
}
else if (expression1)
{
  execute statement.
```

```
}
else if (expression2)
{
   execute statement.
}
……
else
{
      execute statement.
}       // The last else {} can be omitted, if there is, it must be placed after the else if
```

For example: enter grades, A is greater than or equal to 90, B is 80-89, C is 70-79, D is 60 to 69, and E is less than 60.



The C language programming is as follows:

```
void CF_1( S16 score, U8 *level )
{
   char level_t.
   if (score>=90)
   {
      level_t='A'.
   }
   else if (score>=80)
   {
      level_t='B'.
   }
   else if(score>=70)
   {
      level_t='C'.
   }
   else if(score>=60)
```

```
    {
      level_t='D'.
    }
    else
    {
      level_t='E'.
    }
    setU8(level,level_t).
}
```

17.5.2 Switch multi-branch select statement

The switch statement is a multi-branch conditional judgment statement, which makes the program select a branch for execution from multiple branches according to the value of the expression.

General format of switch:

```
switch (expression)
{
   case constant expression 1: statement 1.
                  break.
    case constant expression 2: statement 2.
                  break.
    ……
   case constant expression n : statement n.
                  break.
    default : statement n+1.
}
```

Notice:

• After the case conditional execution statement, you need to add break to jump out of the switch structure, otherwise it will not be able to branch.

• The values of constant expressions must not be equal to each other, otherwise there will be opposites.

• The order of cases does not matter, default is optional.

• switch statement is not equivalent to the if statement. The switch can only check the equality of the values , not the logical judgment. the if statement can not only judge the equality of the values , but also calculate the relational expressions or logical expressions, and make the true or false of the logical judgment. .

E.g:

```
void CF_0( S16 Level, BOOL *heat1, BOOL *heat2, BOOL *heat3, BOOL *fan )
{
   *heat1=*heat2=*heat3=*fan=0.
   switch (Level)
   {
    case 1:*fan=1.
```

```
            break.
        case 2:*fan=0.
            break.
        case 3:*heat1=1.
            break.
        case 4:*heat1=*heat2=1.
            break.
        case 5:*heat1=*heat2=*heat3=1.
            break.
        default:*heat1=*heat2=*heat3=1.
      }
}
```

17.6 Introduction to “libplc300.a” library

The libplc300.a library comes with Magicworks PLC. The library contains the functions of getting the value from STL and writing the value back to STL, as well as the data conversion operation functions of STL system and C language system, and users can call these functions directly. The functions that users can use can be viewed in the header file plc300.h, and the header file plc300.h is in the programming software installation directory\ccompile\include\plc300.h.
The following will introduce the data types and various operation functions of the C program block.

17.6.1 Selectable data types in the CF block parameter symbol table

Key word	Type of data	Notes
unsigned char	BOOL	bit
unsigned char	U8	8-bit unsigned number
signed char	S8	8-bit signed number
unsigned short	U16	16-bit unsigned number
signed short	S16	16-bit signed number
unsigned int	U32	32-bit unsigned number
signed int	S32	32-bit signed number
float	FP32	float
	void*	Pointer: Point to any type of data

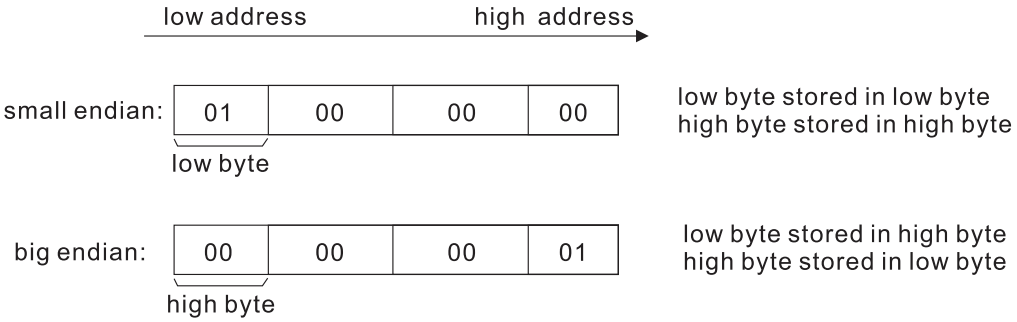
<Remarks> void* can point to any type of data, all provided pointers use void* type, allowing users to perform forced conversion in C code, the return value is empty, and the return result can be passed by pointer.

17.6.2 Endian Conversion

Since the data of the STL system uses the big-endian mode, and the C language uses the small-endian mode, the direct operation of the pointer assignment will cause the problem of byte order confusion, so we provide a set of values from the STL through pointers and the value is

written back to the STL function, and the specific form of the function is in the MagicWorks PLC installation directory \ccompile\include\plc300.h.

Big-endian mode: The high byte of the data is stored in the low address of the memory, and the low byte of the data is stored in the high address of the memory.
small -endian mode : The high byte of the data is stored in the high address of the memory, and the low byte of the data is stored in the low address of the memory.



When the parameters are passed between the C language system and the STL system through pointers, because the C language system uses the small -endian mode. While the STL system uses the big-endian mode, it is necessary to call the big-endian conversion interface.

17.6.3 Data conversion operation functions of STL system and C language system

1. Write the value back to STL through the pointer: Since the input, output and output parameters will become pointers of the corresponding type, and due to the problem of endian conversion, all input / output or output parameters must be written through the following instructions from the C language back to the STL .

void setU8(U8 *P, U8 d). // Write a U8 data from the C language back to the STL system

void setS8(S8 *P, S8 d). // Write a S8 data from the C language back to the STL system

void setU16(U16 *P, U16 d). // Write a U16 data from the C language back to the STL system

void setS16(S16 *P, S16 d). // Write a S16 data from the C language back to the STL system

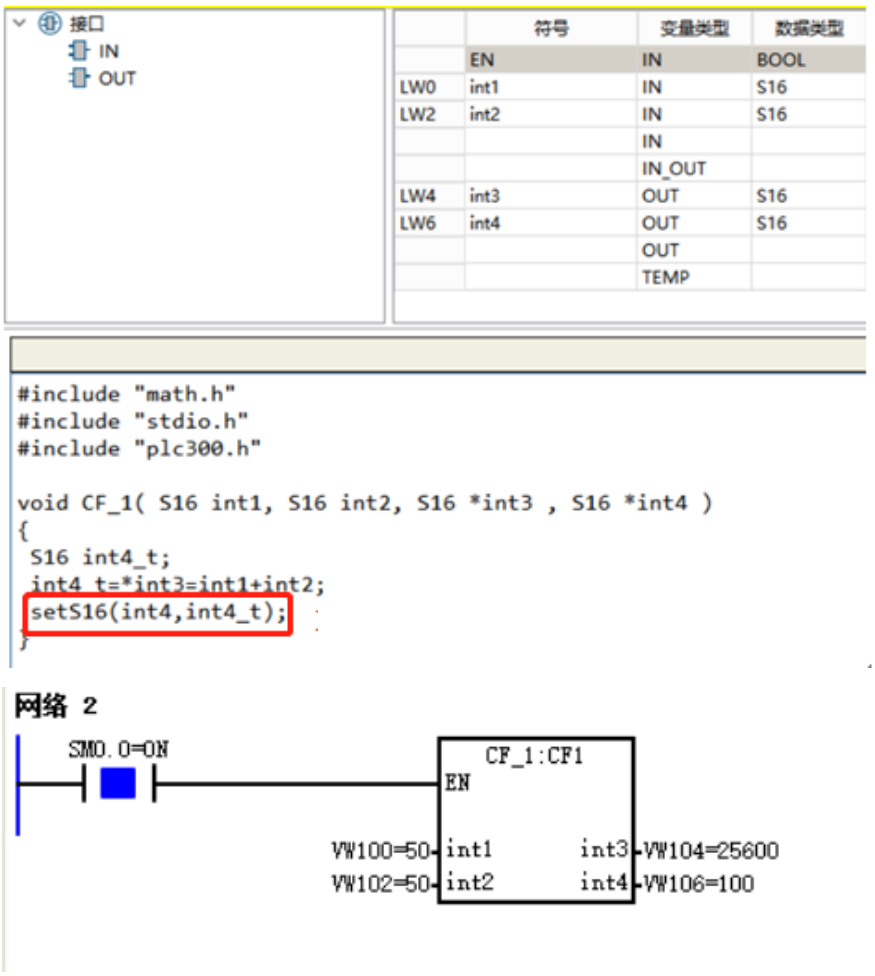
void setU32(U32 *P, U32 d). // Write a U32 data from the C language back to the STL system

void setS32(S32 *P, S32 d). // Write a S32 data from the C language back to the STL system

void setFP32(FP32 *P, FP32 d). // Write a FP32 data from the C language back to the STL system

void setBOOL(BOOL *P, BOOL d). // Write a BOOL data from the C language back to the STL system

An example is as follows: Add int1 and int2, and output INT4 is converted to the big or small end. The output is correct. The output int3 is not converted and the output is incorrect he example is as follows: the input int1 and int2 are added, the output int4 is converted to big and small, and the output is correct. the output int3 is not converted to big and small, and the output is incorrect.



2. Get the value from the STL through the pointer: If you directly call the PLC's V area, M area and other registers in the input parameters of the C function, you can input the pointer of the PLC register to the C function, and obtain the PLC memory area by taking the pointer value. But must be converted by the following functions.

U8 getU8(U8 *p). // Obtain a U8 data available to the C language system from the p address

S8 getS8(S8 *p). // Obtain a S8 data available to the C language system from the p address

U16 getU16(U16 *p). // Obtain a U16 data available to the C language system from the p address

S16 getS16(S16 *p). // Obtain a S16 data available to the C language system from the p address

U32 getU32(U32* p). // Obtain a U32 data available to the C language system from the p address

S32 getS32(S32* p). // Obtain a S32 data available to the C language system from the p address

FP32 getFP32(FP32* p). // Obtain a FP32 data available to the C language system from the p address

BOOL getBOOL(BOOL* p). // Obtain a BOOL data available to the C language system from the p address

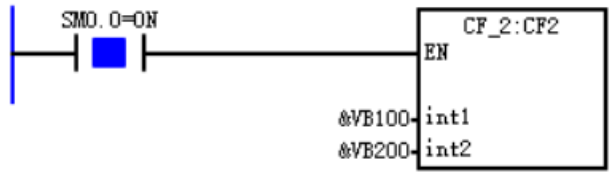
<Remarks> Get value from STL without pointer: The input variable does not pass the pointer, the input variable directly obtains the value of the PLC register, so there is no need to use the get function.

The C language programming is as follows:

1. Assign 20 consecutive VW values starting from VW100 to the address segment of VW200.

```
void CF_2( void* int1, void* int2 )
{
   S16 r,i.
   for(i=0.i<20.i++)
   {
      r=getS16(int1+2*i).
      setS16((int2+2*i),r).
   }
}
```

Call the generated CF block



The status table results are as follows :

3	VW100	有符号	+1
4	VW102	有符号	+2
5	VW104	有符号	+3
6	VW106	有符号	+4
7	VW108	有符号	+5
8	VW110	有符号	+6
9	VW112	有符号	+7
10	VW114	有符号	+8
11	VW116	有符号	+9
12	VW118	有符号	+10
13	VW120	有符号	+11
14	VW122	有符号	+12
15	VW124	有符号	+13
16	VW126	有符号	+14
17	VW128	有符号	+15
18	VW130	有符号	+16
19	VW132	有符号	+17
20	VW134	有符号	+18
21	VW136	有符号	+19

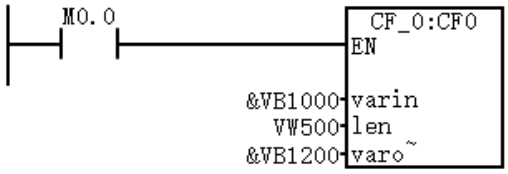
1	VW200	有符号	+1
2	VW202	有符号	+2
3	VW204	有符号	+3
4	VW206	有符号	+4
5	VW208	有符号	+5
6	VW210	有符号	+6
7	VW212	有符号	+7
8	VW214	有符号	+8
9	VW216	有符号	+9
10	VW218	有符号	+10
11	VW220	有符号	+11
12	VW222	有符号	+12
13	VW224	有符号	+13
14	VW226	有符号	+14
15	VW228	有符号	+15
16	VW230	有符号	+16
17	VW232	有符号	+17
18	VW234	有符号	+18
19	VW236	有符号	+19

2. Sort the LEN INT values at the beginning of the V area and display them to another V area .

```
void CF_0( void* varin, S16 len, void* varout )
{
   S16 i,j,n,k,l.
   S16 temp[len].
   for (l=0.l<len.l++)     // Store the value of area V into the temp array
   {
      temp[l]=getS16(varin+2*l).
   }
   for (i=len.i>1.i--)        // The bubble method sorts the size values, each time the largest "floats"
to the top of the array like a bubble
   {
      for(j=0.j<i-1.j++)       //Compare two adjacent elements in turn to move the larger element
```

```
                                backward
    if (temp[j]<temp[j+1])
    {
       n=temp[j].
       temp[j]=temp[j+1].
       temp[j+1]=n.
    }
    }
     for(k=0.k<len.k++)    // Pass out the temp array value to the V area
    {
       setS16((varout+2*k),temp[k]).
    }
}
```

The CF block obtained after compilation is called through OB1 or other FC:



The status table results are as follows:

地址	格式	当前值
VW1000	有符号	+24
VW1002	有符号	+231
VW1004	有符号	+2
VW1006	有符号	+141
VW1008	有符号	+4
VW1010	有符号	+23
VW1012	有符号	+1
VW1014	有符号	+7
VW1016	有符号	+6
VW1018	有符号	+4
VW1020	有符号	+0
VW1022	有符号	+578
VW1024	有符号	+98
VW1026	有符号	+74
VW1028	有符号	+15
VW1030	有符号	+97
VW1032	有符号	+88
VW1034	有符号	+17136
VW1036	有符号	+11
VW1038	有符号	+17136
VW1200	有符号	+17136
VW1202	有符号	+17136
VW1204	有符号	+578
VW1206	有符号	+231
VW1208	有符号	+141
VW1210	有符号	+98
VW1212	有符号	+97
VW1214	有符号	+88
VW1216	有符号	+74
VW1218	有符号	+24
VW1220	有符号	+23
VW1222	有符号	+15
VW1224	有符号	+11
VW1226	有符号	+7
VW1228	有符号	+6
VW1230	有符号	+4
VW1232	有符号	+4
VW1234	有符号	+2
VW1236	有符号	+1

17.6.4 Operation function to get memory base address

```
U8   *getLBase(void).          // Get the value of address LB0
U8   *getIBase(void).          // Get the value of addressIB0
U8   *getQBase(void).          // Get the value of address QB0
U16   *getAQBase(void).        // Get the value of address AQW0
U16   *getAIBase(void).        // Get the value of address AIW0
U8   *getSMBase(void).         // Get the value of address SMB0
U8   *getVBase(void).          // Get the value of address VB0
U8   *getTVBase(void).         // Get the value of address T0(timer value)
U8   *getTBBase(void).         // Get the value of address T0(timer bit)
U8   *getSBase(void).          // Get the value of address SB0
U8   *getCVBase(void).         // Get the value of address C0(counter value)
U8   *getCBBase(void).         // Get the value of address C0(counter bit)
U8   *getACBase(void)          // Get the value of address AC0
```

Appendix

The appendix mainly introduces library instructions, quick query information, order information, etc., as follows:

A Power Budget.

After determining the CPU, power supply module, intermediate expansion module and other expansion module of each rack, confirm whether the current consumption and power consumption of the system bus meet the following conditions:

Condition 1: Confirmation of Bus current consumption

The internal Bus voltage is 5VDC, and the current is provided by the CPU (when there is no intermediate expansion module) or the intermediate expansion module. The sum of the Bus current consumption of the expansion modules on each rack cannot exceed the maximum Bus current allowed by the CPU or relay module.

Condition 2: Power consumption confirmation

When using power modules, the sum of the power consumption of other modules in each rack cannot exceed the maximum power consumption allowed by the power module.

When using an external power supply, select the appropriate power model according to the sum of the connected power.

Table A-1 5VDC Bus current supply and consumption

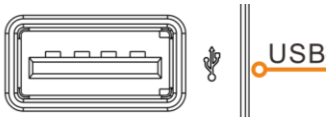
Product model	Supply current	Consumption current
H56-10	1600mA	--
H52-10	1600mA	--
ECT-00	1600mA	--
INT-00	1600mA	--
DIT-08	--	60mA
DIT-16	--	80mA
DIT-32	--	130mA
DQT-08	--	70mA
DQT-16	--	120mA
DQT-32	--	210mA
DQR-08	--	45mA
DQR-16	--	60mA
AIS-04	--	50mA
AIC-08	--	30mA
AIV-08	--	30mA
AQS-04	--	40mA
AQS-08	--	40mA
AMS-06	--	50mA
AIT-04	--	50mA
AIT-08	--	50mA
AIR-04	--	50mA
AIR-08	--	50mA
HSC-02	--	100mA
HSP-04	--	100mA
CAN-1M	--	100mA

Table A-2 24VDC Bus current supply and consumption

Product model	Supply current	Consumption current
PWR-02	2000mA	--
H56-10	--	800mA
H52-10	--	800mA
ECT-00	--	110~800mA
INT-00	--	800mA
DIT-08	--	--
DIT-16	--	--
DIT-32	--	--
DQT-08	--	50mA
DQT-16	--	95mA
DQT-32	--	180mA
DQR-08	--	64mA
DQR-16	--	130mA
AIS-04	--	65mA
AIC-08	--	50mA
AIV-08	--	50mA
AQS-04	--	110mA
AQS-08	--	200mA
AMS-06	--	110mA
AIT-04	--	50mA
AIT-08	--	50mA
AIR-04	--	60mA
AIR-08	--	80mA
HSC-02	--	--
HSP-04	--	100mA
CAN-1M	--	100mA

B Programming Card (USB flash disk) Instructions

H56-10/H52-10 motion controller supports ordinary USB flash disk as a programming card, which can save programs and data conveniently and improve work efficiency. Users don't need to purchase additional special programming cards.



In order to protect the user's data security, we not only encrypt the data files in the programming card, but also take special measures to ensure that even if the data files are copied to other USB flash disk, they cannot be used normally. In addition, users can set a limit on the number of uploads to ensure that their own programs and data will not be copied unlimitedly. Using the USB flash disk as a programming card will not affect other files in the USB flash disk, so you can use it with confidence.

<Note> At present, H56-10/H52-10 only supports USB flash disk with FAT32 file system, and supports hot swap. If the USB flash drive is pulled out during the process of reading and writing data from the USB flash drive, it will cause related functions to be abnormal, such as the communication timeout during the downloading of the program to the CPU. Therefore, you should exercise caution when using it.

The programming card can be downloaded, uploaded, set, and cleared. The following sections will explain each operation step.

The programming card (USB flash disk) can be downloaded, uploaded, set, and cleared. The following sections will explain each operation step separately.

Table B-1 Special memory (SM) related to the programming card

SM status bit	Description
SMB100	The number of uploads limited by the programming card, can only be read, write is meaningless
SMB101	The remaining upload times of the programming card, can only be read, write is meaningless
SM7.7	0: The memory card (USB flash disk)) exists and works normally. 1: The memory card (USB flash disk)) does not exist or works abnormally.

Programming card file format

After the PLC related data is downloaded to the programming card (USB flash disk), it is saved in the USB flash disk as a file. The name of each type of data file is fixed, and the suffix is .CTH or .DTH.

Programming card hot swap

The USB flash drive supports hot-swappable function. In order to ensure that the plug-in process does not affect the normal execution of the PLC program, the following regulations need to be followed:

1) When the CPU is in the RUN state, the inserted USB flash disk is not detected. You must first switch the system operation switch to STOP mode, or stop the CPU through a command, before you can detect and identify the USB flash disk.

2) When the CPU reads and writes the USB flash disk, the stop light will flash. When the light flashes, pulling out the USB flash disk may cause abnormalities or even damage to the USB flash disk.

3) After inserting the USB flash disk, do not let the CPU enter the run state before the CPU recognizes the USB flash disk. The flag correctly recognized by the CPU is that sm7.7 is cleared to 0. It may take 5 ~ 10 seconds from inserting USB flash disk to sm7.7 being cleared 0.

Programming card and CPU switch status

If the programming card (USB flash disk) is inserted into the USB interface when the CPU is powered on, the CPU will enter the corresponding working mode according to the current switch state:

1) If the switch is in the run state, enter the normal working mode.

2) The switch is STOP and enters the upload mode of the programming card. If the program block, initialization V memory data block and hardware configuration block (stored in the file with the suffix. CTH) in the programming card do not exist at this time, the CPU will still switch to the normal working mode.

B.1 Programming Card Download

The download function of the programming card: allows users to download the program blocks, secure library 1/2, hardware configuration blocks, initialized V memory data blocks, recipes and data records in the MagicWorks PLC project to the programming card.

The download steps are as follows:

Step1: download any block to the CPU, and the CPU will automatically write the downloaded block to the USB flash disk During writing, the stop light will flash. After writing, the stop light will always be on

Step2: if you want to limit the use times, make sure that the system block, data block and program block in the programming card have been downloaded, select "PLC"→"programming card configuration" to write the limit times. If it is not necessary to overwrite program blocks and data blocks, check the corresponding block under "block overwrite selection" (default overwrite). Click "OK" to prompt "programming card is set successfully". After the setting is successful, otherwise the corresponding error message will be prompted

Step3: after writing, the stop light changes from flashing to normally on. Wait for 3 seconds before pulling out the programming card

steps:

Step 1: Download any block to the CPU, and the CPU will automatically write the downloaded block to the U disk. During the writing process, the STOP light will flicker. After the writing is completed, the STOP light will always be on.

Step 2: If you want to limit the number of uses, make sure that the system blocks, data blocks, and program blocks in the programming card have been downloaded, and then select the MagicWorks PLC menu item "PLC" → "Programming Card Configuration" to write the limit value. If you do not need to overwrite the program block and data block, you can check the

corresponding block under "Overwrite selection of block" (default overwrite). Click "OK", after the setting is successful, it will prompt "Programming card setting is successful", otherwise it will prompt the corresponding error message.

Step 3: When the writing is completed, for the sake of safety, after the STOP light changes from flashing to constant light, wait 3 seconds before pulling out the programming card.

Tips

1) Before downloading, confirm that the programming card (USB stick) has been inserted into the USB interface and has been correctly recognized (sm7.7 = 0)

2) The free space of USB flash disk must be greater than 64MB to download successfully

3) The download contains any one of the hardware configuration block, program sequence block (the security library belongs to the program block) and init V memory data block. The existing program block, init V memory data block and hardware configuration block in the programming card will be overwritten.

4) If the number of upload limit is set for the programming card, downloading any block to the CPU again will invalidate the number of upload limit of the programming card.

5) During downloading, if the indicator light is flashing, do not power off, otherwise the contents of the programming card may be lost.

6) If the programming card is unplugged without power failure, the CPU may not work normally until it is powered on again.

Table B-2 stop indicator status

indicator	OFF	Flash	ON
Stop light - Orange	--	Transmitting data	normal

B.2 Programming Card Upload

The upload function of the programming card: the user can upload all the blocks in the programming card to the CPU and update the Flash memory of the CPU.

Steps:

Steps 1: Insert the programming card into the USB interface when the CPU is powered off.

Steps 2: Set the DIP switch to "STOP", and then power on the PLC.

Steps 3: If the programming is recognized and there is 002.CTH or 003.CTH or 004.CTH in the programming card, the system will enter the "programming card upload mode", otherwise it will enter the normal operation mode.

Steps 4: When the CPU detects the programming card, it clears all blocks in the CPU Flash memory and covers them with program blocks, data blocks, and hardware configuration blocks in the programming card. If the user configures the overwrite mark in the programming card, the block marked as "not overwritten" will not be uploaded, that is, the original block content will be retained. Only when the system block, data block, and program block all exist, will the upload be successful.

Steps 5: The RUN green light will flicker during the upload process. After the green light changes from flickering to constant light for more than 3 seconds, the upload is complete and the CPU will freeze and wait. At this time, the user can power off. If the upload fails, the SF light (red) will flash until the power is off.

Steps 6: Power on the PLC again, and the CPU will execute the program and data uploaded from the programming card.

<Note> If there is a U disk inserted in the USB interface, the CPU switch must be turned to the "STOP" state before powering on, otherwise it will enter the programming card upload mode and cannot work normally.

1) The usage times of the programming card have reached the limit, which will not affect the original data in the CPU.

2) Failed to verify the information in the programming card.

3) If the CPU is password protected, whether the program block, hardware configuration block and data block can be uploaded from the USB flash disk to the PLC.

Tip

1) During the upload process, when the light is flashing, the power cannot be cut off, otherwise the content of the programming card may be lost.

2) After the upload is completed, the CPU must be powered on again to execute the uploaded program.

Upload count limit

The limit of the number of programming cards allows the customer to set the number of times the programming card is used, and whether to overwrite the program blocks and data blocks when uploading. After each upload is successful, the number of times the programming card can be used is reduced by 1. "PLC" → "Programming Card" Configure" to configure options.

Table B-3 RUN indicator status

Indicator light	OFF	FLASH	ON
RUN (green)	--	Upload in progress	Upload complete
SF	System is normal	Error	--

C CT_ModBus Library

C.1 CT_ModBus Library Introduction

The CT_ModBus function block provides a set of embedded ModBus protocol libraries. The CT_ModBus function block is integrated in the CPU, does not occupy user data space, and is provided to users as a set of library functions. It consists of the following library files:

- **CT_ModBus_RTU library**

The CT_ModBus_RTU library file contains 4 libraries, which are the master-slave library of PORT0 and the master-slave library of PORT1.

<Remark> PORT0 and PORT1 of H56-10/H52-10 support four library files of CT_ModBus_RTU (ct_mBus_master, ct_mBus_master _port1, ct_mBus_slave, ct_mBus_slave _port1), PORT0 of H56-10/H52-10 supports two library files of CT_ModBus_RTU (ct_mBus_master and ct_mBus_slave).

- **CT_ModBus_TCP library**

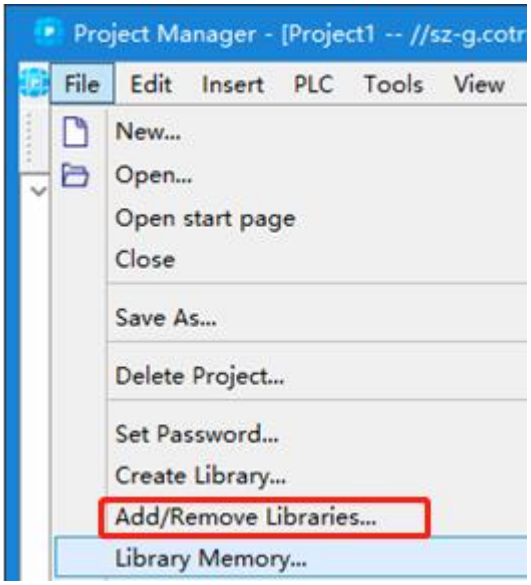
CT_ModBus_TCP contains a library CT_MBus_master_tcp , which is the ModBus TCP master library of the EtherNET communication port .

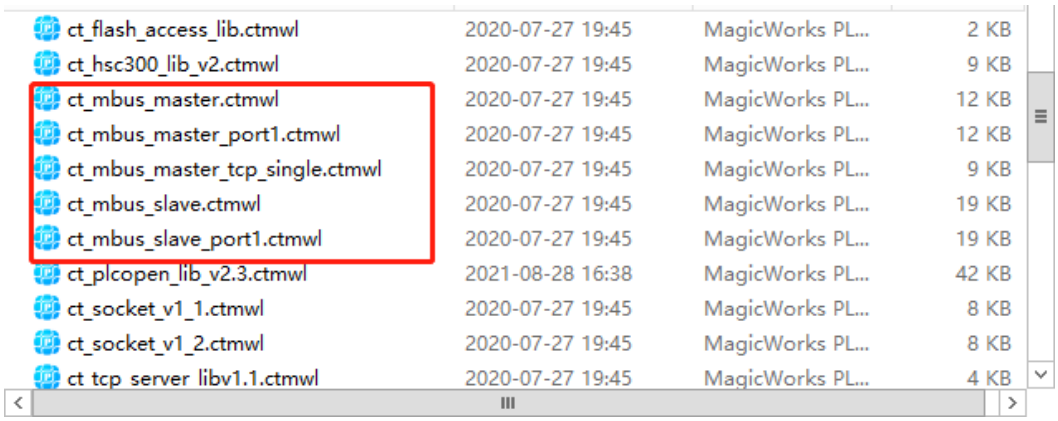
<Note> You can download the CT_ModBus library for free from the official website of Co- trust: http://www.co-trust.com

C.2 Install CT_ModBus Library File

Add library files

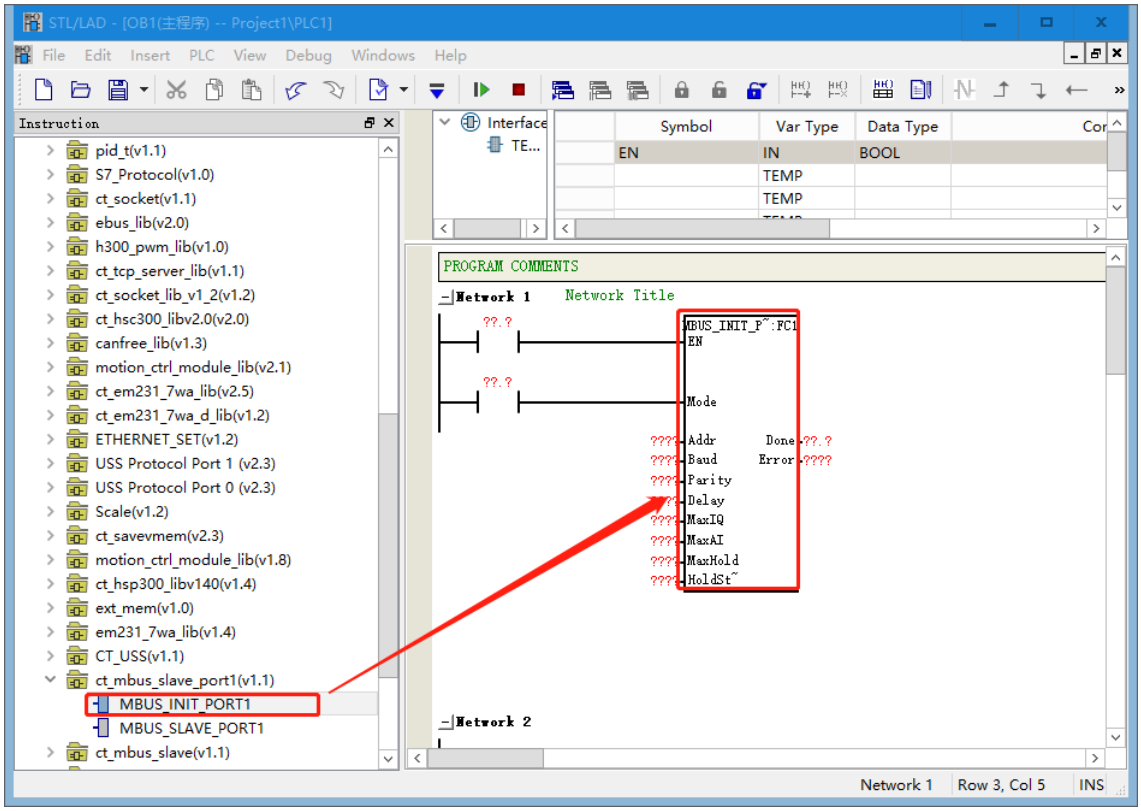
Select the MagicWorks PLC menu item "File" → "Add/Delete Library", then click the "Add", find the library files "ct_mBus_master", "ct_mBus_master_port1", "ct_mBus_slave", "ct_mBus_slave_port1", "ct_mBus_master_tcp", select the added library file, click "Open".





Call CT_ModBus library

Click "Network" to add function blocks, double-click "MBUS_INIT", "MBUS_SLAVE", "MBUS_CTRL", "MBUS_MASTER", " ModBus TCP_EXE" under "Library", the corresponding function blocks will appear in "Network".



C.3 CT_ModBus _RTU Library Function

ModBus address

A ModBus address is usually a 5 or 6 character value containing the data type and offset. The first or two characters determine the data type, and the last four characters are an appropriate value for the data type. The ModBus master then maps this address to the correct function.

ModBus slave commands support the following addresses: 00001 to 02048 are actual outputs, corresponding to Q0.0 ~ Q255.7. 10001 to 12048 are actual inputs, corresponding to I0.0 ~ I255.7. 30001 to 30512 are analog input registers, Corresponding to AIW0 to AIW1022, 40001 to 4XXXX are holding registers, corresponding to the V memory area.

All ModBus addresses are numbered from 1. The following table shows the correspondence

between the ModBus address and the CPU address. The ModBus Slave Protocol allows you to limit the number of inputs, outputs, analog inputs and holding registers (V-area) accessible to the ModBus master.

Table C-1 Correspondence between ModBus address and CPU address

ModBus address	CPU address
000001	Q0.0
000002	Q0.1
000003	Q0.2
……	……
002047	Q255.6
002048	Q255.7
010001	I0.0
010002	I0.1
010003	I0.2
……	……
012047	I255.6
012048	I255.7
030001	AIW0
030002	AIW2
030003	AIW4
……	……
030512	AIW1022
040001	HoldStart
040002	HoldStart+2
040003	HoldStart+4
……	……
04xxxx	HoldStart+2 x (xxxx-1)

Using ModBus RTU Slave Protocol Commands

※ CT_ModBus slave protocol instructions occupy CPU resources

1) Occupy PORT0 or PORT1 as ModBus slave protocol communication according to different ModBus protocol libraries . When PORT 0 or PORT 1 communicates as ModBus protocol, it can no longer be used for any other purpose, including communication with MagicWorks PLC and free port communication. The MBUS_INIT command controls whether the Port setting is ModBus protocol or PPI.

2) All SMs related to selected Port free port communication.

3) Occupies 92 bytes of program space.

※ Steps to use ModBus slave protocol instructions in CPU program

1) Insert the MBUS_INIT instruction in your program and execute this instruction only in one cycle. The MBUS_INIT instruction can be used to initialize or modify the ModBus communication parameters. When you insert the MBUS_INIT instruction, several hidden subroutines and

interrupt service routines are automatically added to your program.
2) Use only one MBUS_SLAVE instruction in your program. This instruction executes every loop cycle, servicing all requests received.
3) Connect the CPU communication port with the ModBus master with a communication cable.

※ Functions Supported by ModBus Slave Protocol Instructions

The ModBus Slave Protocol commands support the ModBus RTU protocol. The following are the supported ModBus features:

Table C-2 ModBus Features

Function	Describe
1	Read single/multiple coil (actual output) status. Function 1 returns the on/off status (Q) of any number of output points.
2	Read single/multiple contactor (actual input) status. Function 2 returns the on/off status (I) of any number of input points
3	Read single/multiple holding registers. Function 3 returns the contents of the V memory, the holding register is a word type under ModBus , and a maximum of 120 words can be read in one request.
4	Read single/multiple input registers. Function 4 returns the analog input value.
5	Write a single coil (actual output). Function 5 sets the actual output point to the specified value. The output point is not forced, the user program can overwrite the value written by ModBus request.
6	Write a single holding register. Function 6 writes the value of a single holding register to the CPU's V memory area.
15	Write multiple coils (actual output). Function 15 writes multiple actual output values to the Q image area of the CPU. The start output point must be the start of a byte (eg, Q0.0 or Q2.0), and the number of outputs to write is a multiple of 8. This is a limitation of the ModBus slave protocol commands. These points are not enforced, and the user program can override the values written by ModBus's request.
16	Write multiple holding registers. Function 16 writes multiple holding registers to the V area of the CPU. Up to 120 words can be written in one request.

※ MBUS_INIT

The MBUS_INIT instruction is used to enable and initialize or disable ModBus communication. The MBUS_INIT instruction must be executed without error before the MBUS_SLAVE instruction can be used. Before continuing to execute the next instruction, the MBUS_INIT instruction must be executed and the Done bit must be set immediately. The MBUS_INIT instruction should be executed only once every time the communication status changes. Therefore, the EN input should be triggered by pulses using edge detection elements, or executed only once in the first cycle period.

Table C-3 MBUS_INT instruction parameter description

Name	Description	Data type	Range	Remark
Mode	Select the communication protocol. 1: Define Port as	bit		

	ModBus protocol and enable the protocol. 0: Define Port as PPI and disable ModBus protocol.			
Addr	Set the address of the site.	byte	1~247	
Baud	Set the baud rate.	Double word	1200,2400,4800,9600,19200,38400,57600,115200	
Parity	Set up verification.	byte	0-- No verification 1-- Odd 2-- Even	All settings use one stop bit.
Delay	By adding a specified number of milliseconds to the standard ModBus message timeout, the standard ModBus message end timeout condition is extended.	int	0~32767	Unit: milliseconds
MaxIQ	Set the number of I and Q points that can be used. 0: Prohibit reading and writing of input and output.	int	0~2048	The recommended value of MaxIQ is 2048, which means that all I and Q points are allowed to be accessed.
MaxAI	Set the number of word input registers (AI) that can be used. 0: Prohibit reading analog input.	int	0~512	MaxAI recommended value is 512
MaxHold	Set the number of word holding registers that can be used in the V memory area.	int	0~32767	Unit: word
HoldStart	Set the starting address of the holding register of the V memory area that can be used.	byte	Pointer to the V storage area.	
Done	When the MBUS_INIT instruction is completed, the Done output is turned on.	bit		
Error	The Error output byte contains the execution result of the instruction.	byte		

※ MBUS_SLAVE

The MBUS_SLAVE instruction serves the request from the ModBus master and must be executed in each cycle to check and respond to ModBus requests. When EN is turned on, the instruction is executed in each cycle. The MBUS_SLAVE instruction has no input parameters.

Table C-4 MBUS_SLAVE parameter description:

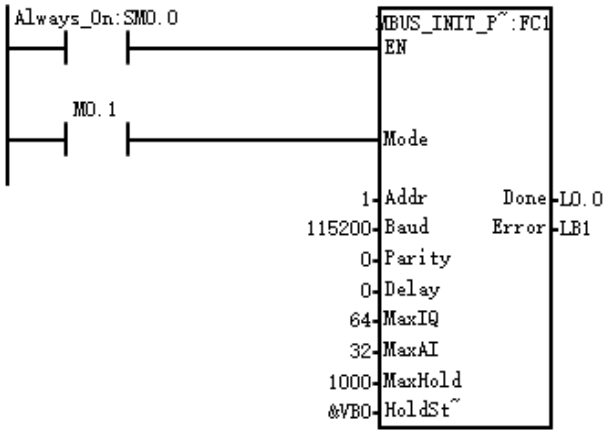
Parameter	Description	Data Type	Range	Remark
Done	When MBUS_SLAVE instruction responds to ModBus request, Done is turned on. If there is no service request, Done will disconnect.	bit		
Error	The output contains the execution result of the instruction.	byte	The error code is shown in the table below	This output is valid only when Done is connected. If Done is disconnected, the error code will not change.

Table C-5 Error code

Error code	Description
0	No error
1	Storage area error
2	Illegal baud rate or parity
3	Illegal slave address
4	Illegal value of modbus parameter
5	The holding register overlaps with the modbus slave symbol address
6	Receive check error
7	Receive CRC error
8	Illegal function request / unsupported function
9	Illegal storage area address in the request
10	Slave function is not enabled

※ ModBus slave protocol command usage example

Create a ModBus slave with a slave address of 1, a baud rate of 115200, and no parity



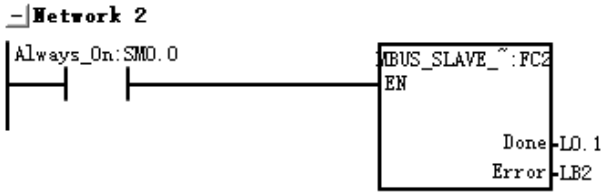


Table C-6 MBUS_INIT parameter configuration description

Addr	Set the slave address to 1
Boud	Set the communication baud rate to 115200
Party	Set parity check to no check
Delay	Set the delay time to 0 ms
MaxIQ	The setting can use up to 256 I and 256 Q (ie 000001~002048 and 010001~012048)
MaxAI	Set the maximum readable 512 analog inputs (030001-030512)
MaxHold	Set the maximum number of V area word storage registers that can be used (in word units).
StartHold	The ModBus master can access the starting address of the register in the V area of the CPU (such as &VB0).

Using ModBus RTU Master Protocol Commands

※ ModBus master protocol instructions occupy CPU resources

1) Occupy PORT0 or PORT1 as ModBus slave protocol communication according to different ModBus protocol libraries. When PORT0 or PORT1 communicate as ModBus protocol, it can no longer be used for any other purpose, including communication with MagicWorks PLC and free port communication. MBUS_INIT command controls whether the setting of PORT is ModBus protocol or PPI.

2) All SMs related to the selection of Port free port communication.

3) Occupies 119 bytes of program space.

※ MBUS_CTRL command

Call the MBUS_CTRL instruction by SM0.0 to complete the initialization of the master and start its function control.

Table C-7 MBUS_CTRL command parameter description

Parameter	Illustrate	Data type	Value range	Remark
Mode	Set the communication mode. 1: the ModBus protocol function is enabled. 0: the system is restored to the PPI protocol.	bit		
Baud	Set the baud rate.	DWORD	11200,2400,4800,9600,19200,38400,57600,115200	
Parity	Set parity	Byte	0: no parity 1: odd 2: even	All settings use one stop bit.
Timeout	The time the master waits for a response from the slave. units: ms	int	1 ~ 32767	A typical setting value is 1000 ms (1s).
Done	Completion bit , initialization is complete, this bit will be automatically set to 1.	bit		
Error	Initialization error code	Byte	0: no error 1: Check selection is illegal 2: Illegal baud rate selection 3: mode selection is illegal	Only valid when the Done bit is 1.

※ MBUS_MSG command

使用 SM0.0 调用 ModBus RTU 主站读写子程序 MBUS_MSG 指令,First 接通发送一个 ModBus 请求.同一时刻只能有一个读写功能（即 MBUS_MSG）使能.

Table C-8 MBUS_MSG command parameter description

Parameter	Illustrate	Data type	Value range	Remark
First	write request bits	bit		Every new read or write request must be pulse-triggered.
Slave	Set the slave address	byte	1~247	
RW	Read and write commands	byte	0: read 1: write	
Addr	Select the data type to read and write	DWORD	00000~0xxxx: switch output 10000~1xxxx:	

			switch input 30000~3xxxx: analog input 40000~4xxxx: holding registers	
Count	Number of data communicated (number of bits or words)	int		The maximum amount of data that can be read/written by each MBUS_MSG instruction of the ModBus master is 120 words.
DataPtr	Data pointer, if it is a read command, the data read back is placed in this data area. if it is a write command, the data to be written is placed in this data area.	DWORD		
Done	Completion bit , read and write function completion bit	bit		
Error	Error code		Error codes are shown in the table below	Error codes are only valid when the Done bit is 1.

Table C-9 Error codes

Error code	Describe
0	No error
1	Response check error
2	Unused
3	Receiving timeout (no response from slave)
4	Request parameter error
5	ModBus/Freeport is not enabled
6	ModBus is busy with other requests
7	Response error (response is not the requested operation)
8	Response to CRC check and error
101	The slave does not support the requested function
102	Slave does not support data address
103	The slave does not support this data type
104	Slave equipment failure
105	The slave received the information, but the response was delayed
106	The slave is busy and rejected the message

107	The slave rejected the message
108	Slave memory parity error

C.4 CT_ModBus_master_tcp_single library Function

MBTCPS_EXE is a single read and write command of the ModBus TCP master

The ModBus TCP master writes 120 data (starting from VB0) into the ModBus TCP slave (IP: 192.168.1100), and the starting address is a section of memory at 40001.

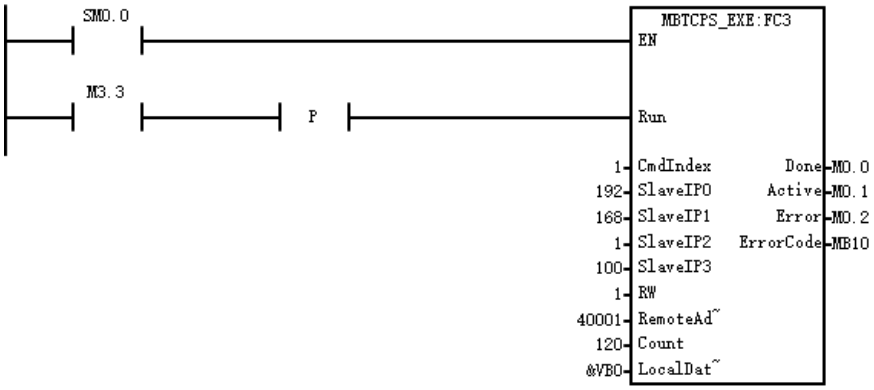


Table C-10 Command parameter description

Parameter	Illustrate	Data type	Value range
Run	Communication enable bit	BOOL	Please use edge trigger
CmdIndex	Call the serial number of MTCPS_EXE, and the serial number of each call cannot be repeated	BYTE	The range is 1 to 255
Slave IP0	First digit of slave IP address	BYTE	Assuming the slave IP address is 192.168.1.201, then Slave IP0~Slave IP3 should be set to: 192/168/1/201 respectively
Slave IP1	Second digit of slave IP address	BYTE	
Slave IP2	The third digit of the slave IP address	BYTE	
Slave IP3	Fourth digit of slave IP address	BYTE	
RW	Read = 0,Write = 1	BYTE	
Count	Number of read and write units, valid range: 1~120 word or 1~1920 bit	WORD	
RemoteAddr	Remote data address (eg 40001)	DWORD	
LocalDataPtr	Local data pointer (eg &VB1000)	WORD	
Done	Completion flag: 0 = not done, 1 = done	BOOL	
Active	Active bit, 0 = not active or completed, 1 = operation active and not completed	BOOL	
Error	Error bit, 0 = no error, 1 = error	BOOL	
ErrCode	Error code, valid when done bit is 1 (see table below)	BYTE	

Table C-11 Error code table

Error code	Describe
0	No error
1	The total number of connections has reached the maximum number of connections allowed
2	Establishing connection
3	Operation timeout
4	Request parameter error
5	Command is not activated
6	The connection is Busy (such as activating multiple MBTCPS_EXE at the same time)

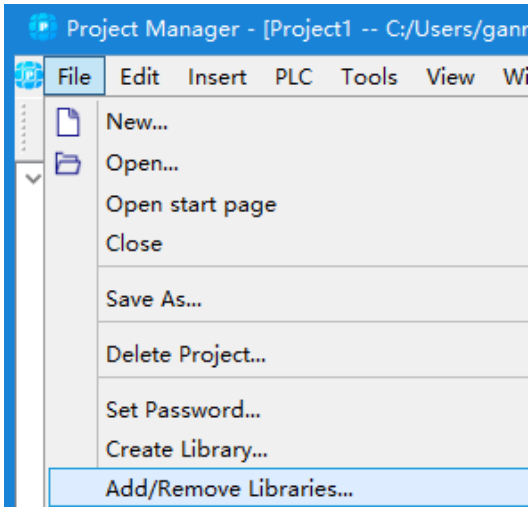
D The Use of the PID_T Library

The PID_T function block is integrated in the main control module, does not occupy user data space, and is provided to users as a library function. PID_T is mainly for the intelligent PID function of temperature control, with self-tuning and self-adapting functions. No complicated programming is required, just call and set some simple parameters to use, and the temperature control is accurate.

You can download the library for free from the official website of COTRUST. download link:http://www.co-trust.com/Download/index.html.

Installation Notes

Select MagicWorks PLC menu item "File" → "Add/Delete Library", then click the "Add" button, find this file in the location of your stored pid_t.ctmwl, and click "OK" to add successfully.



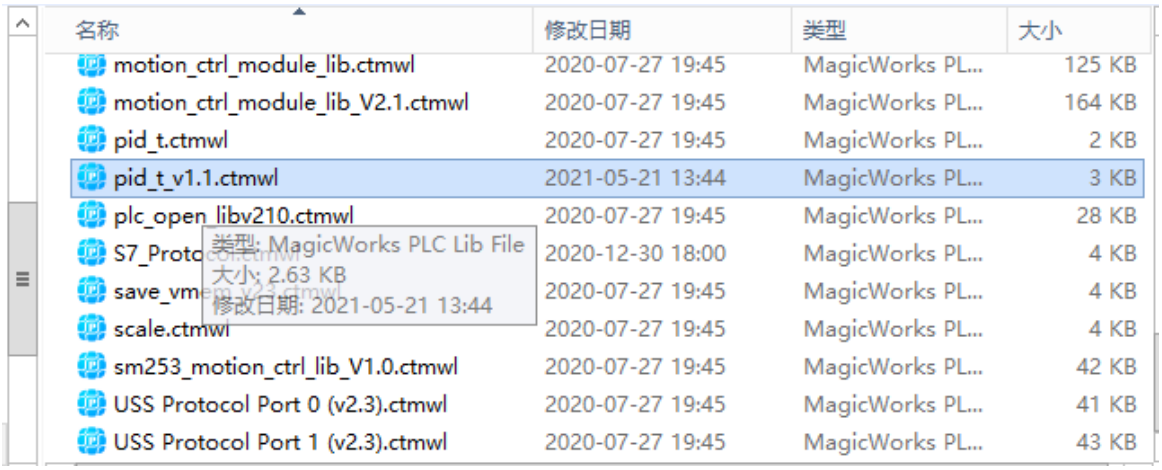


Table D-1 Instruction parameter

Parameter	Description	Data type	Range	Remark
LOOP	The number of the PID of the circuit to which it belongs, can not be repeated starting from 0	Word, constant or variable	0-63	Indicates the control loop ID.
CTRL_WORD	Control word Control PID operation	Word, constant or variable		Commonly used control words: 1) 16#03 (only heating output, with adaptive function) 2) 16#07 (heating and cooling output, with adaptive function)
SP	Set value	Word, constant or variable	-32768-32767	Unit: 0.1℃
PV	Measured value (feedback value)	Word, variable	-32768-32767	Unit: 0.1℃
MAX_PV	Maximum value of the measured value	Word, constant or variable	-32768-32767	Unit: 0.1℃
OUT_CYCLE	Pulse output period	Word, constant or variable	1-255	Unit: second
TUNING_K	Self-tuning coefficient	Double word, floating point	0.5-2.0	0.5: The system control overshoot is required to be small. 1.0: Normal response. 2.0: The system requires fast response and large overshoot.
TUNING_ON	Start auto tuning	Bit, variable		It will reset automatically after auto-tuning.
Kp	Scale factor	Word, integer, constant or		If it is a constant, the auto-tuning function

		variable		cannot be executed.
Ti	Integration time	Word, integer, constant or variable	1-3600	Unit: second
Td	Differential time	Word, integer, constant or variable	0-3600	If it is a constant, the auto-tuning function cannot be executed.
STATUS_WORD	Status word	Word, variable		Unit: second
HEAT_ON	Heating output	Bit		
COOL_ON	Cooling output	Bit		
PID_OUT	PID analog output	Word, integer, variable	When there is only heating output: 0-32000. With cooling output: -32000-32000	

Table D-2 Control Word Bit Address Definition

Control word bit	Value	Remark
0	0	PID stop
	1	PID operation
1	0	Integral always works, the proportional coefficient Kp is not automatically adjusted
	1	Integral separation and automatic adjustment of proportional coefficient
2	0	PID unipolar output
	1	PID bipolar output
3	0	Reserve
	1	Reserve
4	0	Points work
	1	Points don't work
5	0	Differentiation works
	1	Differentiation does not work
6	--	Reserve
7	--	Reserve

Table D-3 Status word bit address definition

Status word bit	Value	Description
0	0	No disconnection fault
	1	Disconnection fault
1	0	Auto-tuning is not in progress
	1	Auto-tuning
2	0	No self-tuning failure
	1	Self-tuning failure

3	0	No heating
	1	Heating
4	0	No cooling
	1	Cooling
5	0	PID stop state
	1	PID operating status
6	--	Reserve
7	--	Reserve

E The Ethernet "ct_socket" Communication Library

E.1 CT_socket library Introduction

At present, the CPU supports 2 UDP connections and 2 TCP client connections. The instructions provided are SOCK_Open, SOCK_Send, SOCK_Recv, SOCK_Close.

Table E-1 ct_socket library introduction

Item	Range	Illustrate
Connection number (SockID)	0~255	255 is invalid connection number
Connection type (SockType)	0,1	0: UDP connection, 1: TCP client connection
Port number (Lport/Rport)	1~65535	The local port number should be a port number that is not occupied by other connections. The default port number 20000 in the system block is used for PLC monitoring connections. The local port number of the TCP client is allocated by the bottom layer, Lport can be set to any value (cannot be set to 65535 in UDP mode), and Rport must be the same as the server port number.

Table E-2 Socket error code

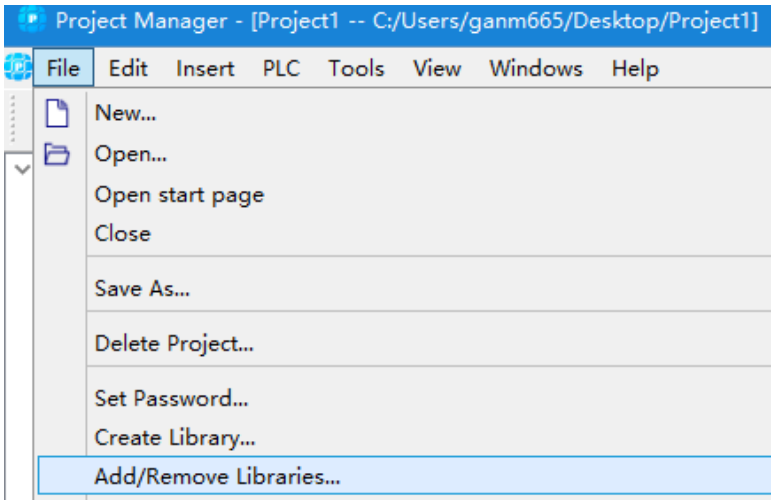
Error code	Description
0	No error
1	Connection closed
2	The port number is wrong., the local port number is occupied, or the port number is out of range, an error is reported.
3	Device disconnection
4	UDP connection failed to establish. Commonly used, connection creation fails, and connection number 255 is returned.
5	The TCP client failed to connect to the remote server
6	TCP server listening failed
7	The data length is wrong, the maximum data length is 512 bytes
8	Data area is empty error
9	Wrong connection number
10	Received data error (data not received)

11	Wrong connection type
12	IP information error
13	Timeout error
14	TCP client needs to reconnect flag. Commonly used, TCP reconnect

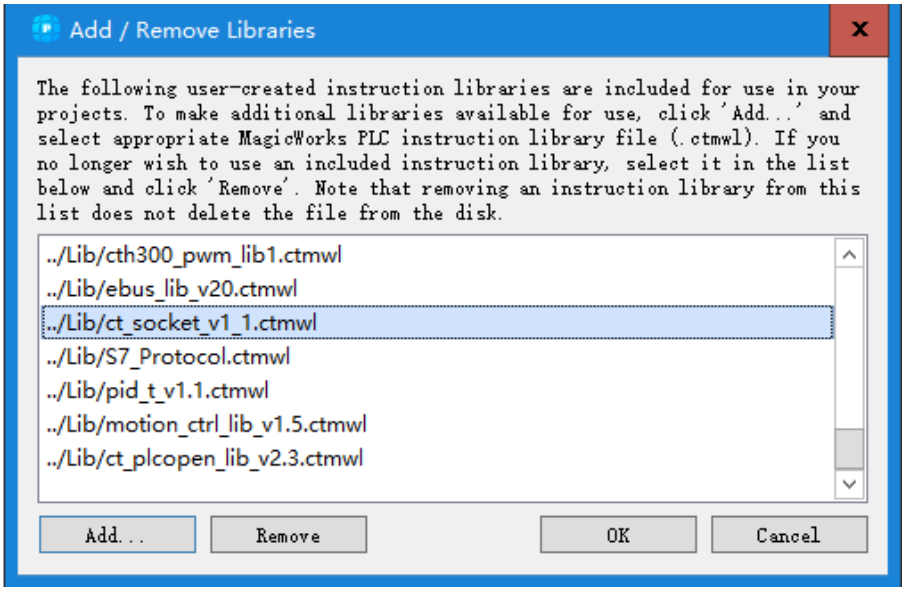
E.2 Instructions For Use

[Add library file]

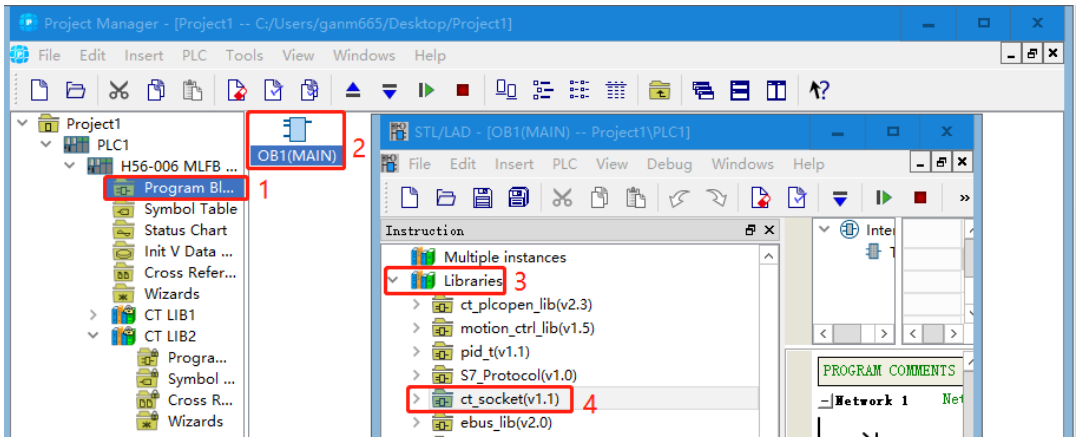
In the project manager interface or programming interface, select "File"-"Add/Remove Library", as shown in the figure below:



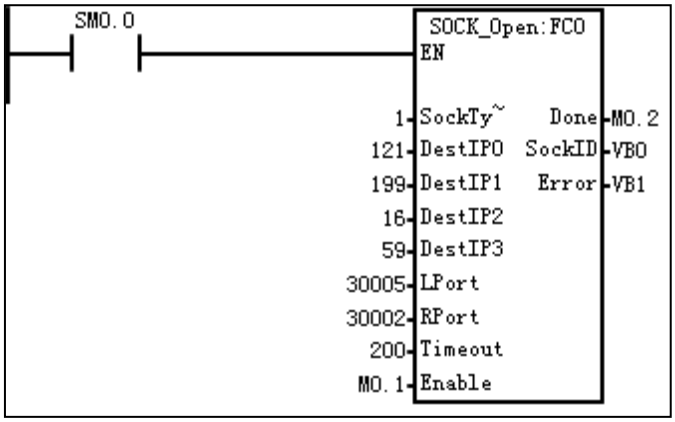
Find "ct_socket_v1_1.ctmwl" as shown in the figure below, and click the "OK" button.



After the installation is successful, you can see the newly added "ct_socket (v1.1)" under "Library" in the directory tree of the program block interface:【指令功能说明】



SOCK_Open



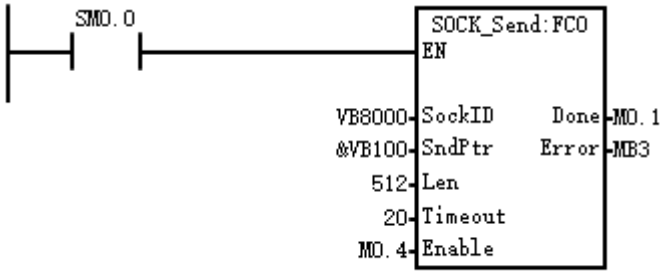
Function: Create a connection

Parameter Description:

Parameter	Description	Input/ output	Data type	Remark
SockTy	Connection type, UDP(0) and TCP-CLIENT(1)	IN	BYTE	TCP server connection is not supported.
DestIP0	The first byte of the destination IP address.	IN	BYTE	For example, the target address is 192.168.1.202, and the first byte is 192.
DestIP1	The second byte of the destination IP address.	IN	BYTE	For example, the target address is 192.168.1.202, and the second byte is 168.
DestIP2	The third byte of the destination IP address.	IN	BYTE	For example, the target address is 192.168.1.202, and the third byte is 1.
DestIP3	The 4th byte of the destination IP address.	IN	BYTE	For example, the target address is 192.168.1.202, and the fourth byte is 202.
Lport	Local port, used by UDP	IN	WORD	TCP-CLIENT mode, LPort is randomly allocated by the bottom layer.
Rport	Destination port	IN	WORD	It is the same as the port number of the remote server.

Timeout	overtime time	IN	BYTE	Unit: 100ms.
Enable	Enable bit	I/O	BOOL	
Done	Read and write function complete bit	OUT	BOOL	It is automatically cleared after being enabled.
SockID	Output connection number	OUT	BYTE	For low-level allocation, the application must provide a global memory to save the connection number.
Error	error code	OUT	BYTE	error code. 0: no error

SOCK_Send

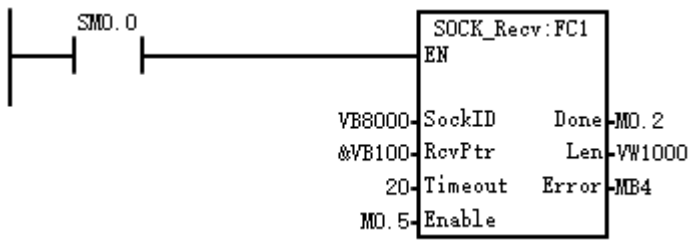


Function: Send data

Parameter Description:

Parameter	Description	Input/ output	Data type	Remark
SockID	Output connection number	OUT	BYTE	For the underlying allocation, the application shall provide a global memory to save the connection number.
SndPtr	Data buffer	IN	DWORD	Pointer to the data to be sent, which can point to I, Q, m or V memory (e.g. &amp. VB100)
Len	Bytes to be sent	IN	WORD	The range is 1 ~ 512 bytes.
Timeout	Timeout	IN	BYTE	Unit: 100ms.
Enable	Enable bit	I/O	BOOL	Enable
Done	Read write function completion bit	OUT	BOOL	It is automatically cleared after being enabled.
Error	error code	OUT	BYTE	Error code. 0: no error

SOCK_Recv

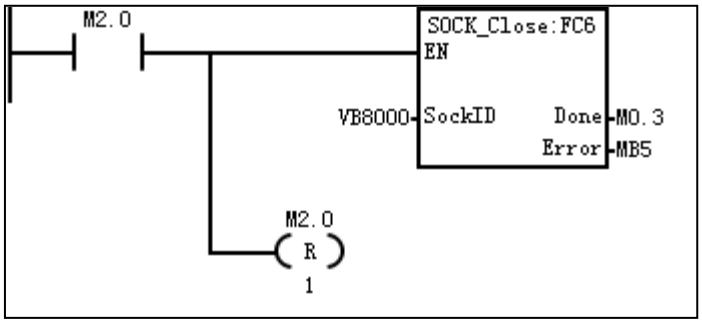


Function: Receive data

Parameter Description:

Parameter	Description	Input/ output	Type of data	Remark
SockID	Output connection number	OUT	BYTE	For the underlying allocation, the application shall provide a global memory to save the connection number.
RcvPtr	Data buffer	IN	DWORD	Pointer to the storage location of the received data, which can point to I, Q, m or V memory (e.g. &VB100)
Timeout	overtime time	IN	BYTE	Unit: 100ms.
Enable	Enable bit	I/O	BOOL	Enable
Done	Read and write function complete bit	OUT	BOOL	It is automatically cleared after being enabled.
Len	Actual number of bytes received	IN	WORD	The range is 1~512 bytes.
Rport	Destination port	IN	WORD	It is the same as the port number of the remote server.
Error	error code	OUT	BYTE	error code. 0: no error

SOCK_Close



Function: Close the connection

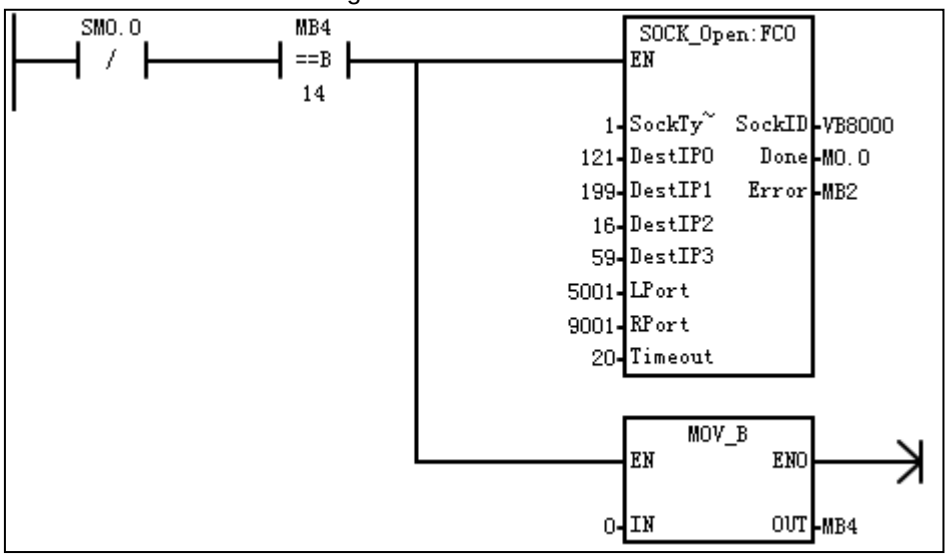
Parameter Description:

Parameter	Description	Input/ output	Data type	Remark
SockID	Output connection number	OUT	BYTE	For the underlying allocation, the application shall provide a global memory to save the connection number.
Done	Read and write function complete bit	OUT	BOOL	It is automatically cleared after being enabled.
Error	error code	OUT	BYTE	error code. 0: no error

[Applications]

TCP client disconnects and reconnects, by sending an error flag to determine whether to reopen

the connection: reconnect flag 14.

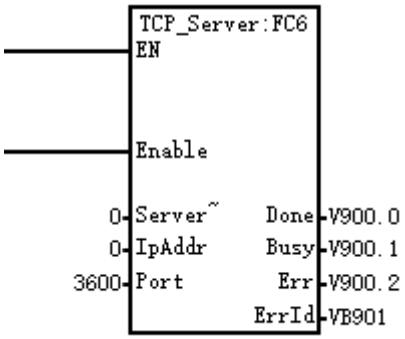


F CT_tcp_server_lib Library Introduction

CT_ tcp_ server_ The Lib library contains four instructions: TCP_ server、TCP_ connect、TCP_ send、TCP_ recv.

< note > you can download CT_ tcp_ server_ Lib Library for free from COTRUST official website: http://www.co-trust.com .

F.1 TCP_Server



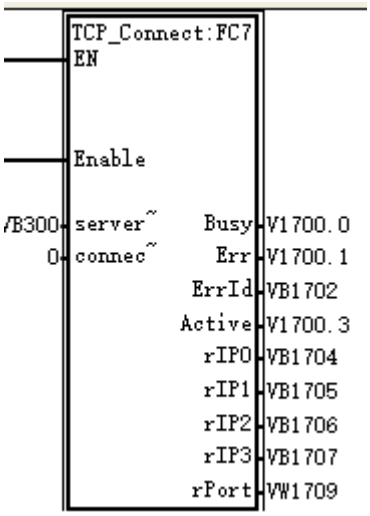
Function: This command is used to start Tcp Server.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Enable	IN	Instruction execution condition	BOOL	Start the TCP server when enable is 1 Stop TCP server when enable is 0
ServerId	IN	Tcp Server ID number	BYTE	Provide 2 TCP Servers, Service Id is (0, 1)
IpAddr	IN	Remote connection address	BYTE	Temporarily can only take 0
Port	IN	Port number	WORD	

		monitored by Tcp Serve		
Done	IN	Instruction complete bit	BOOL	Done is set when the instruction is completed or an error occurs. When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. When the execution condition of the instruction is Off, the Busy bit is reset.
Err	OUT	Command error bit	BOOL	When the instruction is wrong. Err is set. When the execution condition of the instruction is Off, the Err bit is reset.
ErrId	OUT	error code	BYTE	0. No error 1. Serverid error 2. IPADDR error 3. The port number is occupied When the execution condition of the instruction is off, errid is cleared to 0.

F.2 TCP_Connect



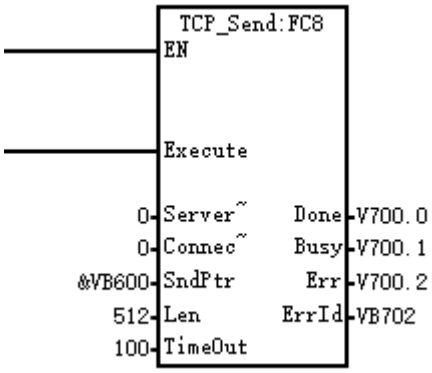
Function: This command is used to get the status of TCP connection.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Enable	IN	Connection enable bit	BOOL	Get the connection status when Enable is 1. When the Enable falling edge is in the connected state, the connection is disconnected.
ServerId	IN	Tcp Server Id	BYTE	Value 0, 1

		number		
ConnectId	IN	Connection Id number	BYTE	Each Server supports up to 4 connections, and the connect Id is (0, 1, 2, 3).
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. When the execution condition of the instruction is Off, the connection is successfully disconnected and the Busy bit is reset.
Err	OUT	Command error bit	BOOL	When the instruction is wrong. Err is set.
ErrId	OUT	error code	BYTE	0: no error 1: ServerId error 4: ConnectId error
Active	OUT	Connection status bit	BOOL	When the connection is established, Active is set. When the connection is disconnected, Active is reset.
rIP0	OUT	IP address of remote connection 0	BYTE	
rIP1	OUT	IP address of remote connection 1	BYTE	
rIP2	OUT	IP address of the remote connection 2	BYTE	
rIP3	OUT	IP address of remote connection 3	BYTE	
rPort	OUT	Port number for remote connection	WORD	

F.3 TCP_Send



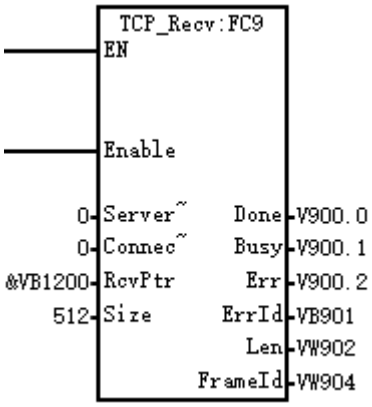
Function: This command is used to send data to the TCP connection.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Excute	IN	Command trigger bit	BOOL	When Excute is a rising edge, it triggers the sending instruction.
ServerId	IN	Tcp Server Id number	BYTE	Value 0, 1
ConnectId	IN	Connection Id number	BYTE	Each Server supports up to 4 connections, and the connect Id is (0, 1, 2, 3).
SndPtr	IN	Send pointer bit	DWORD	
Len	IN	Send data length	WORD	Up to 512 bytes
Timeout	IN	Send timeout ms	WORD	
Done	IN	Send complete bit	WORD	When the data transmission is complete, Done is set. When the execution condition of the instruction is Off, the Done bit is reset. When Execute is 0, the transmission is complete and Done will be set for one cycle.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. When the execution condition of the instruction is Off, the Busy bit is reset.
Err	OUT	Command error bit	BOOL	When the instruction is wrong. Err is set.

ErrId	OUT	error code	BYTE	0: no error 1: ServerId error 4: ConnectId error 5: Server is not open 6: The connection status is wrong 7: Sending timeout

F.4 TCP_Recv



Function: This command is used to receive data from the Tcp connection

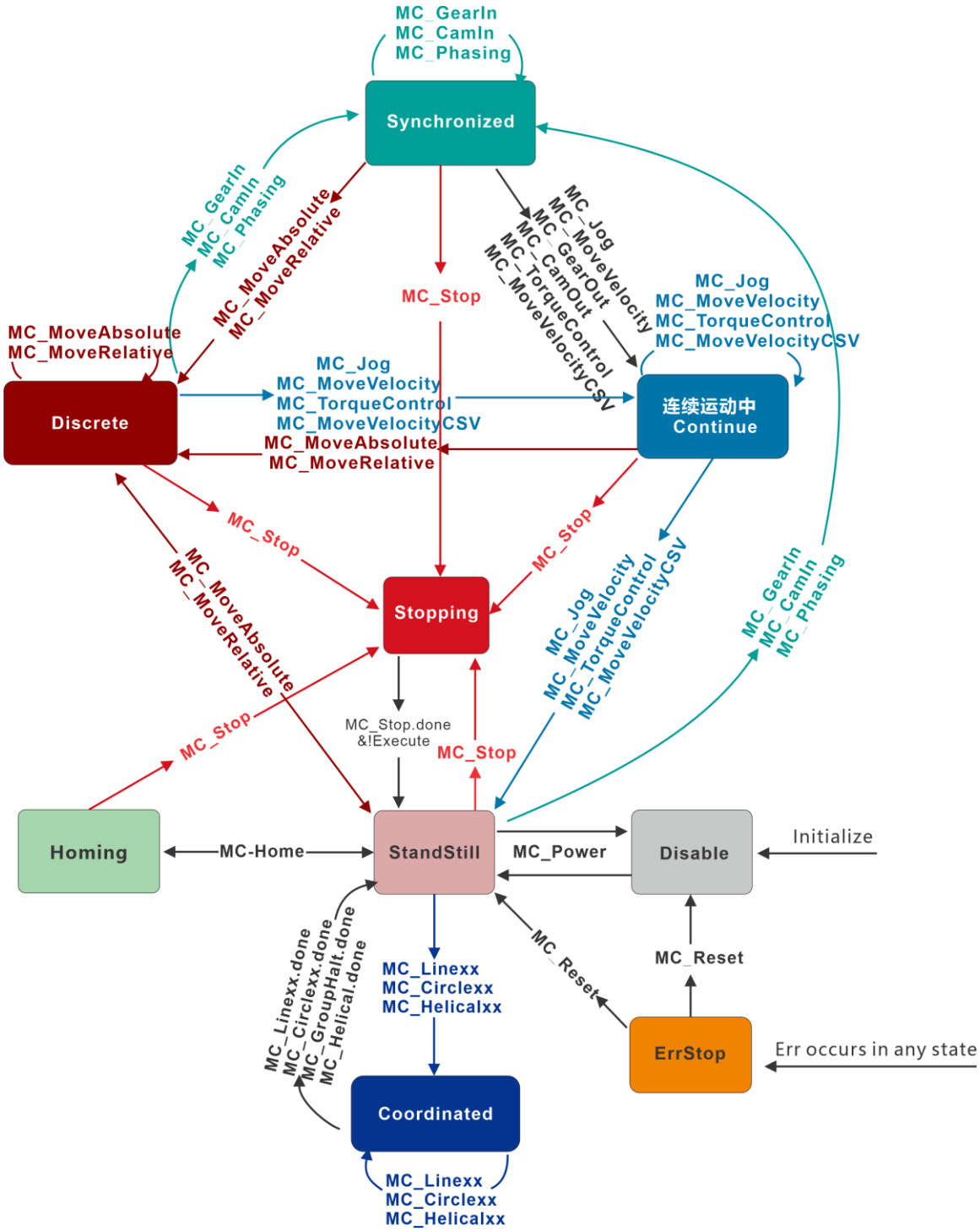
Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Enable	IN	Receive enable bit	BOOL	When enable is 1, the instruction is in the state of receiving data. When enable is 0, the instruction does not receive data.
ServerId	IN	Tcp Server Id	BYTE	Value 0, 1
ConnectId	IN	Connection ID number	BYTE	Each server supports up to 4 connections with connection ID (0, 1, 2, 3).
RcvPtr	IN	Receive data pointer bit	DWORD	
Size	IN	Maximum length of received data	WORD	Up to 512 bytes
Done	IN	Receive complete bit	WORD	Done is set when data is received. When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. When the execution condition of the instruction is Off, the Busy bit is reset.
Err	OUT	Command error bit	BOOL	When the instruction is wrong. Err is set.

ErrId	OUT	error code	BYTE	0: no error 1: Server id error 4: Connect id error 5: Server is not open 6: Connection status error 7: Send timeout
FrameId	OUT	Frame ID of the received data	WROD	Whenever a frame of data is received, the FrameId will increase by 1, and the user can judge whether there is new data based on this.

G Axis Command “ct_plcopen_lib(v2.3)”

Axis command is designed according to PLCopen standard, and the flow chart of axis status command is shown in the following figure:



提示

1、脉冲轴不能与 ct_hsp300_lib 库里 MC_PTP_R、MC_PTP_A、MC_SPEED_CTRL 同时使用.

G.1 Single Axis Control Command

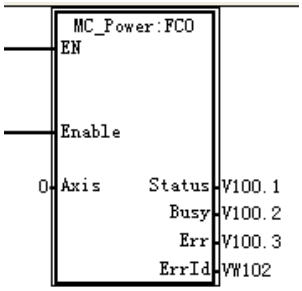
Single-axis control instructions are generally used for positioning control, that is, servo motors drag external mechanisms to a specified position.

The MC function blocks of single-axis control are as follows:

Name	Command function	Remark

MC_Power	Axis enable command	
MC_MoveAbsolute	Absolute displacement instruction	
MC_MoveRelative	Relative displacement command	
MC_MoveVelocity	Speed command	
MC_Stop	Stop instruction	
MC_Halt	Interruptible stop command	
MC_Home	Return to origin command	
SMC_Home	Special return to origin command	
MC_MoveSuperImposed	Superimpose relative position command	
MC_Reset	Reset error command	
MC_MoveFeed	Motion instruction triggered by interrupt event	
MC_MoveJog	Jog command	
MC_SetPosition	Set current position command	
SMC_CtrlByActPos	Position following instructions	
MC_ReadStatus	Read axis status command	
MC_ReadSetPos	Read axis setting position command	
MC_WriteParameter	Write parameter command	
MC_ReadParameter	Read parameter command	
MC_TouchProbe	Probe instructions	
MC_TorqueControl	Torque control command	
MC_MoveVelocityCSV	Speed command (CSV mode)	
MC_ReadVelPos	Read axis speed and position commands	

MC_Power

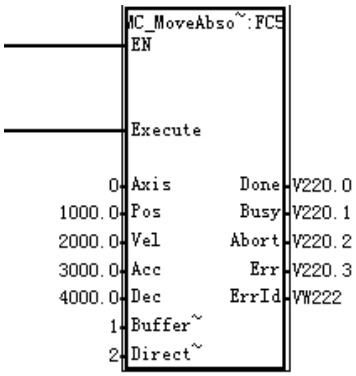


Function: This instruction is used to enable or disable the corresponding axis.

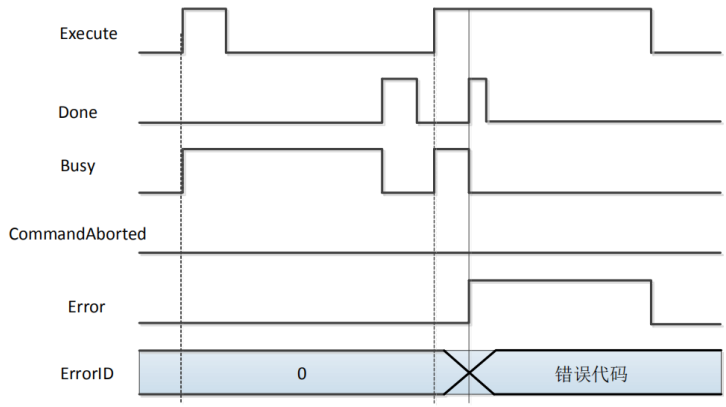
Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Enable	IN	Axis enable	BOOL	Set to TRUE to enter the operable state, set to FALSE to release the operable state.
Axis	IN	Axis ID number	BYTE	
Status	OUT	Status bit	BOOL	Changes to TRUE when entering a runnable state. Entering the non-runnable state changes to FALSE.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Reset Busy when the instruction completes. When the execution condition of the instruction is Off, the Busy bit is reset.
Err	OUT	Error bit	BOOL	If an Error is detected, the Error bit is set. When the execution condition of an instruction changes from On to Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveAbsolute



Function: This command is used to control the terminal actuator to accelerate and decelerate to the target position relative to the zero point according to the given speed. Once this command is terminated during the operation, the remaining distance will be discarded and the new command function will be performed.

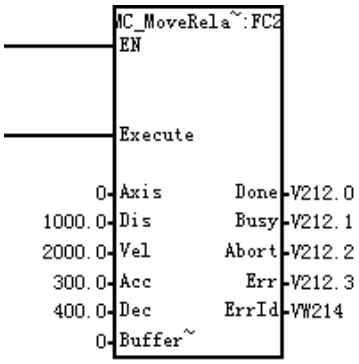


Parameter Description:

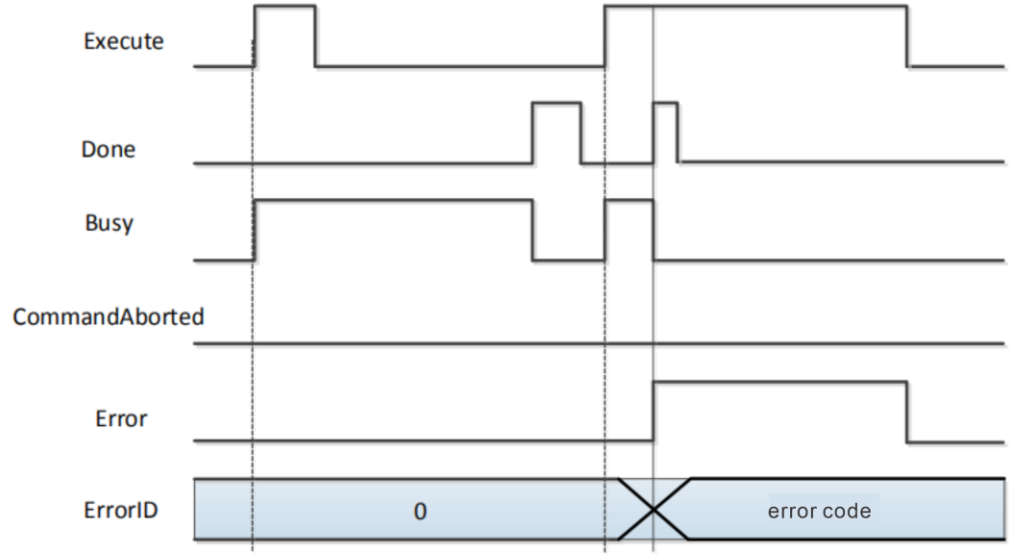
Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Pos	IN	Location	REAL	The target position of the terminal actuator relative to the zero point, unit: unit
Vel	IN	speed	REAL	The operating speed of the terminal actuator, this parameter is always positive. (Unit: unit/second)
Acc	IN	Acceleration	REAL	The acceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
BufferMode	IN	Abort mode	BYTE	Define the action of the axis: 0: Abort (directly interrupt the ongoing command) Other: illegal
Direction	IN	direction	INT	Direction of rotation: 0: The direction with the shortest distance. 1: Positive direction. -1: reverse direction. 2: Continue the current direction. This parameter is valid for rotary axis, but invalid for linear axis.
Done	OUT	Done bit	BOOL	When the absolute displacement action is completed, Done is set. When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, Abort is set. When the execution condition of the instruction is Off, the Abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set.

				When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveRelative



Function: It is used to control the terminal actuator to take the current position as the reference point and move a given distance at a given speed. Once this command is terminated during operation, the unfinished distance will be discarded and a new command function will be executed.

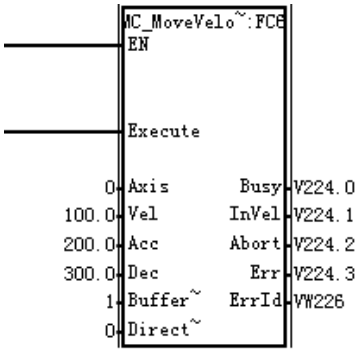


Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Dis	IN	distance	REAL	Taking the current position as a reference, the target distance to be moved by the terminal actuator. Setting this value to a negative number means that the axis will be reversed. Unit: unit.

Vel	IN	speed	REAL	The operating speed of the terminal actuator, this parameter is always positive. (Unit: unit/second)
Acc	IN	Acceleration	REAL	The acceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
BufferMode	IN	Abort mode	BYTE	0: Abort (direct interruption of ongoing instructions) Others: illegal
Done	OUT	Done bit	BOOL	When the absolute displacement action is completed, the done bit is set. When the execution condition of the instruction is off, the done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. When the command completes Busy reset. When the execution condition of the instruction is off, the Busy bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the Abort bit is set. When the execution condition of the instruction is off, the Abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveVelocity

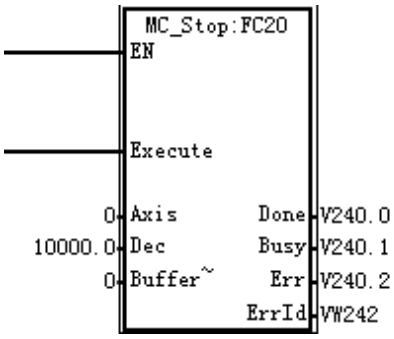


Function: This instruction is used to control the terminal actuator to move to a given speed according to a given acceleration and deceleration, and run at a constant speed. When the terminal mechanism reaches the given speed, the execution of this instruction is completed, but the terminal mechanism continues to run at this speed.

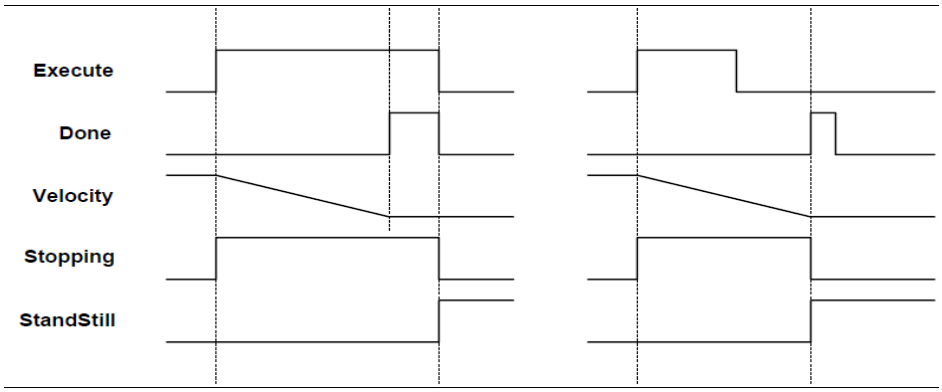
Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution conditions	BOOL	This command is executed when the execution condition changes from off to on.
Axis	IN	Shaft ID number	BYTE	
Vel	IN	speed	REAL	The operating speed of the terminal actuator. This parameter is always positive. (unit: Unit / second)
Acc	IN	acceleration	REAL	Acceleration of terminal actuator, this parameter is always positive. (unit: Unit / second / second)
Dec	IN	deceleration	REAL	Deceleration of terminal actuator, this parameter is always positive. (unit: Unit / second / second)
BufferMode	IN	Abort mode	BYTE	0: Abort (directly interrupt the instruction in progress) Others: illegal
Direction	IN	direction	INT	1: Positive direction -1: Negative square
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. When the command completes Busy reset. When the execution condition of the instruction is off, the Busy bit is reset.
Invel	OUT	Speed reach bit	BOOL	After reaching the target speed, the invelocity bit is set. When instructed When the execution condition changes from on to off, the inversion bit is reset.
Abort	OUT	Command termination bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the error bit is set. When the execution condition of the instruction is off, the error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_Stop



Function: This instruction is used to control the terminal actuator to decelerate at the given deceleration rate until it stops. When this instruction is running, it cannot be terminated by any other instruction. After the speed drops to 0, the axis changes to the Stopping state, and will not switch to the StandStill state until the Execute changes to 0.

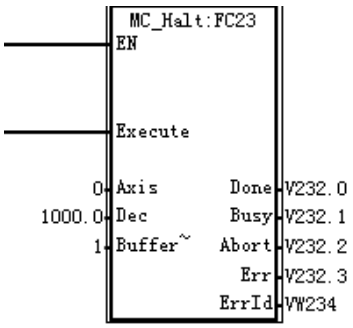


Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second)
BufferMode	IN	Abort mode	BYTE	0: Abort (directly interrupt the ongoing command) Other: illegal
Done	OUT	Done bit	BOOL	After the speed drops to 0, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset.

Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_Halt



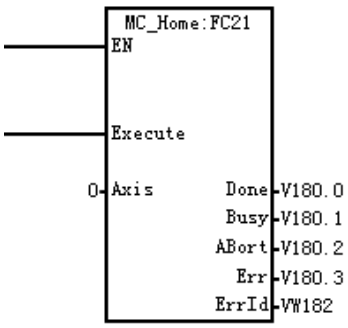
Function: This instruction is used to control the terminal actuator to decelerate at the given deceleration rate until it stops. When this instruction is running, it can be terminated by other instructions.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
BufferMode	IN	Abort mode	BYTE	0: Abort (directly interrupt the ongoing command) Other: illegal
Done	OUT	Done bit	BOOL	After the speed drops to 0, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the Abort bit is set. When the execution condition of the instruction is Off, the Abort bit is reset.

Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_Home



Function: This instruction is used to control the axis to execute the homing action according to the mode and speed given by the axis parameters. The homing mode and speed are set in the axis parameter setting interface.

Note: The configuration of pulse axis and communication axis is different.

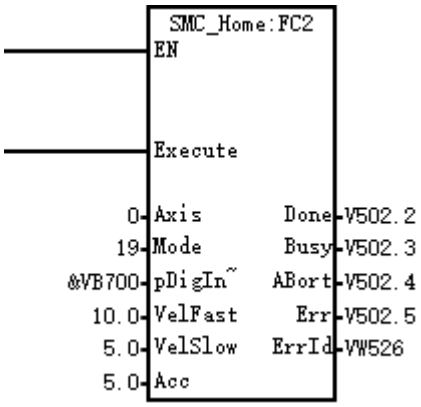
- The home mode and home speed of the pulse axis are set in the parameter setting interface. For details, see Chapter 15.1 Axis Wizard
- For communication axis, it is necessary to rewrite the value of the home mode 16#6098:0 and configure the home speed 16#6099:0, 16#6099:1. for the specific operation, please refer to chapter 15.1 Axis Wizard

• Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Done	OUT	Done bit	BOOL	After the speed decelerates to 0, the Done bit is set.When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. When the command completes Busy reset. When the execution condition of the instruction is off, the Busy bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the

				instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

SMC_Home



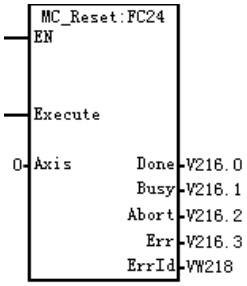
Function: By calling this instruction, the controller controls the axis to execute the return-to-origin action according to the mode and speed given by the parameters. The return-to-origin mode and speed are set in the instruction. note that in general, the origin switch/limit signal of the axis is connect to the PLC system. After returning to the original, the controller will offset the axis coordinate to 0.0, and the 16#6064:0 (position actual value) of the axis itself will not change due to the operation command.

Parameter Description:

parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Mode	IN	Return to zero mode	BYTE	Refer to Example 1: Pulse axis return to origin
pDigInput	IN	Input signal pointer	pointer	The memory address pointed to by the origin switch, forward and reverse limit switch position numbers. bit0: Negative switch bit1: Forward switch bit2: Origin switch. The switch signal 1 is valid, and 0 is invalid. For example, when pDigInput is equal to &IB0, I0.0: Negative switch I0.1: Forward switch I0.2: Origin switch.
VelFast	IN	Quick zero return speed	REAL	The fast speed when searching the origin, this parameter is always positive. (Unit: unit/second)

VelSlow	IN	Slow return to zero speed	REAL	The slow speed when searching the origin, this parameter is always positive. (Unit: unit/second)
Acc	IN	Acceleration	REAL	The acceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
Done	OUT	Finish bit	BOOL	After returning to zero, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_Reset



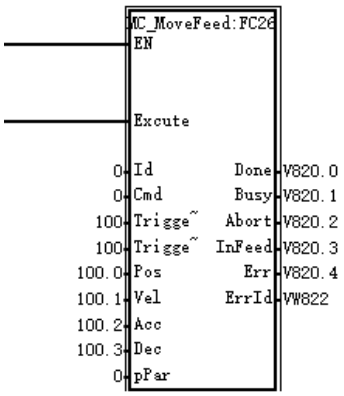
Function: Used to remove the abnormality of the axis. After the abnormality is removed, the axis state will change from ErrorStop state to StandStill state.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Done	OUT	Finish bit	BOOL	When the axis state is reset to the ready-to-execute state, the Done bit is set. When the execution condition of the

				instruction changes from On to Off, the Done bit is reset
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveFeed



Function: After the specified input interrupt event occurs, the axis performs the action specified by Cmd. Before the designated interrupt occurs, Execute becomes 0 and the current instruction can be cancelled. After the movement is completed, it needs to be triggered again to respond to the interrupt again.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Id		Axis or axis group Id	BYTE	See Schedule 2 below
CMD	IN	What happened	BYTE	0: Stop. 1: Relative movement. 2: Absolute exercise. 3: Speed control. 16#80: Axis group stop
TriggerInput	IN	Interrupt event	BYTE	See Schedule 1 below
TriggerVar	IN	Interrupt	DWORD	

		event parameters		
Pos	IN	Location	REAL	See Schedule 2 below
Vel	IN	speed	REAL	
Acc	IN	Acceleration	REAL	
Dec	IN	decrease speed	REAL	
pPar	IN	Other parameter pointers	DWORD	Reserved. When the above parameters are not enough to describe the parameters, the parameters are placed in the memory indicated by pPar.
Done	IN	Done bit	BOOL	When the axis state is reset to the ready-to-execute state, the Done bit is set. When the instruction execution condition changes from On to Off, the Done bit is reset
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction execution process is terminated, the Abort bit is set. When the execution condition of the instruction is Off, the Abort bit is reset.
InFeed	OUT	Movement position	BOOL	When an interrupt occurs and a movement is triggered, InFeed is set. When the instruction is completed, InFeed is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

Attached Table 1 Interrupt trigger events

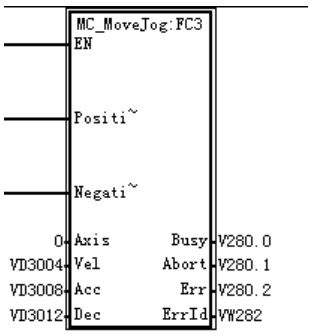
TriggerInput	Explain	TriggerVar
0	Rising edge, I0.0	Module with interrupt event Triggervar = rack number * 256 + slot number If the module is placed at the position with rack No. 1 and slot No. 3, Then triggervar = 16#0103
2	Rising edge, I0.1	
4	Rising edge, I0.2	
6	Rising edge, I0.3	
1	Falling edge, I0.0	
3	Falling edge, I0.1	
5	Falling edge, I0.2	
7	Falling edge, I0.3	
12	HSC0 CV=PV	
27	HSC0 direction change	
28	HSC0 external recovery/Zphase	

13	HSC1CV=PV	
14	HSC1 direction change	
15	HSC1 external recovery/Zphase	
128	V memory rising edge interrupt	Triggervar = memory offset address * 8 + bits of memory. If v0.0, this value is 0. V127.7 this value is 1023 (127 * 8 + 7).
129	V memory falling edge interrupt	

Schedule 2 axis movement

CMD	Parameter	Detailed description
0: STOP	ID: shaft ID number Dec: deceleration	It is equivalent to executing MC after interrupt trigger_ Stop instruction.
1: relative motion	ID: shaft ID number Vel: speed ACC: acceleration Dec: deceleration Pos: distance of movement	It is equivalent to executing MC after interrupt trigger_ MoveRelative
2:absolute motion	ID: shaft ID number Vel: speed ACC: acceleration Dec: deceleration Pos: Location When the axis is a rotating axis: The direction of vel > 0 is positive rotation Vel < 0 direction is reverse rotation	It is equivalent to executing MC after interrupt trigger_ MoveAbsolute
3:speed movement.	ID: shaft ID number Vel: speed ACC: acceleration Dec: deceleration Vel > 0.0 forward Vel < 0.0 reverse	It is equivalent to executing MC after interrupt trigger_ MoveVelocity
16#80:Shaft group stop	ID: axis group ID number Dec: deceleration	It is equivalent to executing MC after interrupt trigger_ GroupHalt

MC_MoveJog

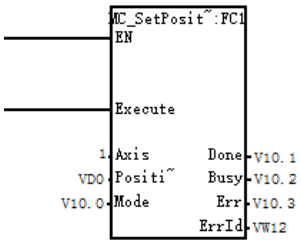


Function: perform inching movement according to the specified velocity. Set positiveenable to true when positive direction micro movement is required, and set negativeenable to true when negative direction micro movement is required.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
PositiveEnable	IN	Positive direction is effective	BOOL	Set to TRUE to start moving in the positive direction. Set to FALSE to end the move.
NegativeEnable	IN	Valid in negative direction	BOOL	Set to TRUE to start moving in the negative direction. Set to FALSE to end the move.
Axis	IN	Axis ID number	DWORD	Axis ID number
Vel	IN	Target speed	REAL	Specify the target speed. The unit is [command unit/sec].
Acc	IN	Acceleration	REAL	The acceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second) When set to 0, realize emergency stop.
Busy	OUT	Busy bit	BOOL	The instruction is being executed as TRUE, and the execution is completed as FLASE
Abort	OUT	Stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_SetPosition



Function: This command sets the current coordinate value (unit).

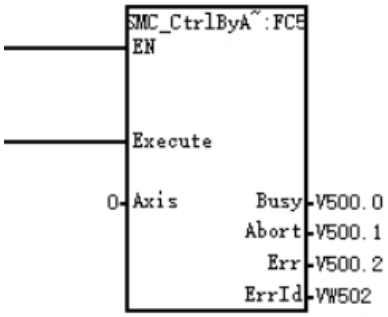
Note: This instruction cannot be used when the axis is in motion. When the axis is used as the main axis of synchronous motion, it cannot be used, otherwise unpredictable consequences may occur.

Parameter Description:

Parameter	Input/output	Description	Data type	Remark
Execute	IN	Instruction	BOOL	When the execution condition changes from

		execution condition		Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	Axis ID number
Position	IN	Specified coordinate position	REAL	Specified coordinate position (unit) Value -1E+13<=Position<=1E+13
Mode	IN	Location mode	BOOL	0: The absolute position mode is specified as this mode, and the absolute position of the axis is Position 1: The relative position mode is specified as this mode, the absolute position of the axis is the current position plus Position
Done	OUT	Status bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction changes from On to Off, the Error bit is reset.
Busy	OUT	Busy bit	BOOL	The instruction is being executed as TRUE, and the execution is completed as FLASE.
Error	OUT	Error bit	OBOL	If an error is detected, the Error bit is set. When the execution condition of the instruction changes from On to Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

SMC_CtrlByActPos



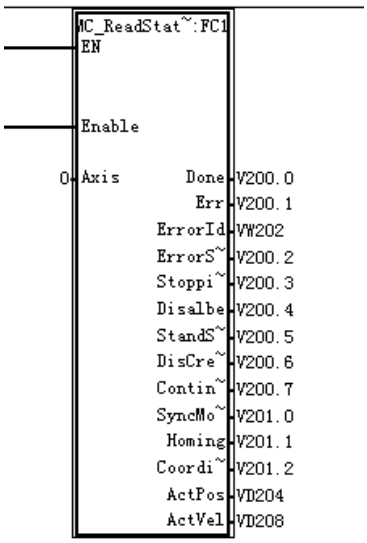
Function: After calling this command, the axis position will be controlled by an external command, and the axis setting position will follow the current position.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed. When the execution condition changes from ON to Off, cancel the current instruction
Axis	IN	Spindle ID number	BYTE	Axis ID number
Busy	OUT	Command processing	BOOL	True, the order is being processed

Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_ReadStatus



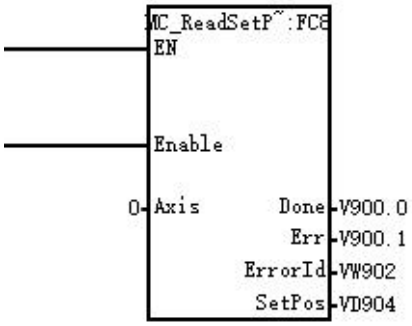
Function: This command is used to read the value of the corresponding axis state.

parameter.Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
enable	IN	Instruction execution condition	BOOL	When it is On, execute the instruction.
Axis	IN	Axis ID number	BYTE	Axis ID number
Done	OUT	Status bit	BOOL	After the instruction is executed successfully, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction changes from On to Off, the Error bit is reset.
ErrId	OUT	error code	WORD	error code When Error is TRUE, ErrId is the error of the instruction. When ErrStop is TRUE, ErrId is the current error of the axis.

ErrStop	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the state ErrStop
Stopping	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the Stopping state
Disabled	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the Disabled state
StandStill	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the state StandStill
DisCreteMotion	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the state DisCreteMotion
Continous Motion	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the state ContinousMotion
SyncMotion	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the state SyncMotion
Homing	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the state Homing
Coordinate	OUT	Status indicator bit.	BOOL	TRUE, if the axis is in the state Coordinate
ActPos	OUT	Current coordinates	REAL	unit
ActVel	OUT	Current speed	REAL	Unit/sec

MC_ReadSetPos



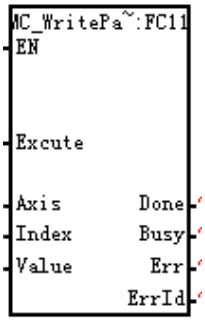
Function: This command is used to read the current setting position of the axis.

Parameter Description:

Parameter	Input/output	Description	Data type	Remark
enable	IN	Instruction execution condition	BOOL	When it is On, execute the instruction.
Axis	IN	Axis ID number	BYTE	Axis ID number
Done	OUT	Status bit	BOOL	After the instruction is executed successfully, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.

Error	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction changes from On to Off, the Error bit is reset.
ErrId	OUT	error code	WORD	When Error is TRUE, ErrId is the error of the instruction. When ErrStop is TRUE, ErrId is the current error of the axis.
Setpos	OUT	Current set position	REAL	unit

MC_WriteParameter

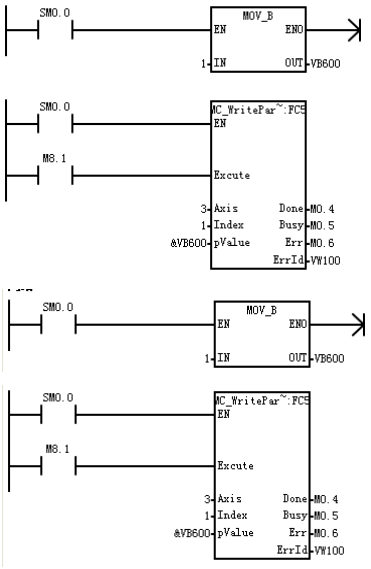


Function: This instruction is used to write the value of the configuration parameter of the corresponding axis. After the write is successful, it is equivalent to modifying the hardware configuration and is permanently effective. This instruction can only be run when the axis is not enabled.

Parameter Description:

Parameter	input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Index	IN	Parameter number	WORD	See the parameter description table in the following table
pValue	IN	Pointer to parameter value	DWORD	Parameter pointer
Done	OUT	Status bit	BOOL	After the instruction is executed successfully, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Error	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction changes from On to Off, the Error bit is reset.
ErrId	OUT			Error code, see chapter G.4 Error code

Parameter description table

Index	Parameter	pValue	Descripition
0	Dimensional conversion	pValue is a pointer to the dimension description. Dimensional description is shown in Table 3 below.	
1	Acceleration and deceleration mode	pValue: Pointer to acceleration/ deceleration mode. Acceleration and deceleration mode is a BYTE Acceleration mode: 0: trapezoid 1: S shape 2:Sin acceleration and deceleration	Axis 3 acceleration and deceleration mode is S-shaped acceleration and deceleration 
2	speed	pValue: It is a pointer to the speed setting. Refer to Table 4 for pointing structure	

Attached Table 3 data structure description of dimensional conversion

Name	Data type	Descripition
ScalPulseInc	REAL	Dimensional conversion, pulse increment (Hardware configuration dimension conversion left 1)
ScalMotorTurns	REAL	Dimensional conversion, motor turns (Hardware configuration dimension conversion right 1)
ScalMotorTurnsInc	REAL	Dimensional conversion, motor turns increment (Hardware configuration dimension conversion left 2)
ScalGearTurnsInc	REAL	Dimensional conversion, gear output turns increment. (Hardware configuration dimension conversion right 2)
ScalGearTurnsInc	REAL	Dimension conversion, output turns increment (Hardware configuration dimension conversion left 3)
ScalAppUnitsc	REAL	Dimensional conversion, units in application. (Hardware configuration dimension conversion right 3)

Attached Table 4 Speed setting instructions

Name	Type	Description
VelMax	REAL	Maximum speed
VelMin	REAL	Minimum speed
VelStartStop	REAL	Start stop speed

AccMax	REAL	Maximum acceleration
AccMin	REAL	Minimum acceleration

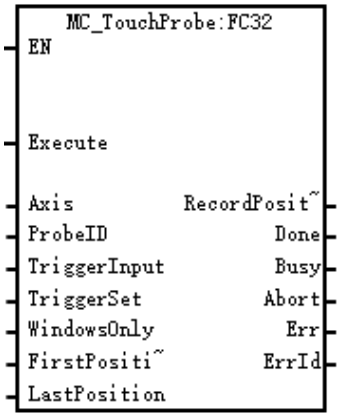
MC_ReadParameter

Function: This instruction is used to read the value of the configuration parameter of the corresponding axis.

Parameter Description:

Parameter	Input/ output	Description	Type data	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Index	IN	Parameter number	WORD	See parameter description table
pValue	OUT	Parameter pointer	DWORD	
Done	OUT	Status bit	BOOL	After the instruction is executed successfully, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Error	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction changes from On to Off, the Error bit is reset.
ErrId	OUT	error code		Error code, see chapter G.4 Error code

MC_TouchProbe



Function: Record the actual position when the trigger event occurs.

If this command selects the communication axis as the source, the PDO needs to add the object dictionary of the probe function. Probe 1 is 0x60B8, 10x60B9, 0x60BA, 0x60BB. Probe 2 is 0x60B8, 0x60B9, 0x60BC, 0x60BD.

Parameter Description:

Parameter	Input/ output	Description	Type data	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	Axis ID number
ProbeID	IN	ProbeID	BYTE	0: PLC interrupt 1: Communication axis Probe1 2: Communication axis Probe2
TriggerInput	IN	Probe source	BYTE	To PLC interrupt: interrupt event number, see the table below for details-PLC interrupt probe source For communication axis: • Communication axis Probe1: After the command is enabled, this setting will be mapped to the 2-3 bits of the drive 16#60B8: 00 0: DI input 1: Z phase 2, 3: Driver definition • Communication axis Probe2: After the command is enabled, this setting will be mapped to bits 10-11 of the drive 16#60B8:00) 0: DI input 1: Z phase 2, 3: Driver definition
TriggerSet	IN	Probe settings	BYTE	• For PLC interrupt: If it is the interrupt of H56/H52 itself:

				the rack number is 0 and the slot number is 1, and the address is 16#01. If the expansion module is interrupted, if the module is placed in rack number 1, slot number 5, the address is 16#15. • For communication axis: After enabling the command, this setting will be mapped to the 4-5 bits of the drive 16#60B8: 00 0: rising edge 1: Falling edge
WindowsOnly	IN	Window is valid	BOOL	The designated window 1 is valid, and 0 is invalid.
FirstPosition	IN	starting point	REAL	Specify the start position of the receive trigger.
LastPosition	IN	End position	REAL	Specify the end position of the receive trigger.
Recordposition	OUT	Trigger recording position	REAL	The current position when the trigger occurs
Done	OUT	Status bit	BOOL	After the instruction is executed successfully, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Busy bit	BOOL	The instruction is being executed as TRUE, and the execution is completed as FLASE
Abort	OUT	Abort bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Error	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

The following is a description of the triggerinput parameter in the probe instruction as the interrupt event number table when PLC is interrupted, the windows only window is valid, and the driver 16#60b8 bit definition.

1. PLC interrupt probe source: interrupt event number table

Priority group	Interrupt event number	Priority group	Illustrate
Discrete (medium priority)	0	1	Rising edge, I0.0
	2	1	Rising edge, I0.1
	4	1	Rising edge, I0.2
	6	1	Rising edge, I0.3
	48	1	Rising edge, I0.4
	50	1	Rising edge, I0.5
	52	1	Rising edge, I0.6
	54	1	Rising edge, I0.7
	56	1	Rising edge, I1.0
	58	1	Rising edge, I1.1
	1	1	Falling edge, I0.0
	3	1	Falling edge, I0.1
	5	1	Falling edge, I0.2
	7	1	Falling edge, I0.3
	49	1	Falling edge, I0.4
	51	1	Falling edge, I0.5
	53	1	Falling edge, I0.6
	55	1	Falling edge, I0.7
	57	1	Falling edge, I1.0
	59	1	Falling edge, I1.1
	12	1	HSC0 CV=PV
	27	1	HSC0 direction change
	28	1	HSC0 external recovery/Zphase
	13	1	HSC1 CV=PV
	14	1	HSC1 direction change
	15	1	HSC1 external recovery/Zphase
Discrete (medium priority)	16	1	HSC2 CV=PV
	17	1	HSC2 direction change
	18	1	HSC2 external recovery/Zphase(not support)
	32	1	HSC3 CV=PV
	38	1	HSC3 direction change
	40	1	HSC3 external recovery/Zphase(not support)
	29	1	HSC4 CV=PV
	30	1	HSC4 direction change(not support)
	31	1	HSC4 external recovery/Zphase(not support)
	33	1	HSC5 CV=PV
	39	1	HSC5 direction change(not support)

	41	1	HSC5 external recovery/Zphase(not support)
	19	1	PTO 0 completion interrupt (not supported)
	20	1	PTO 1 completion interrupt (not supported)

2. WindowsOnly

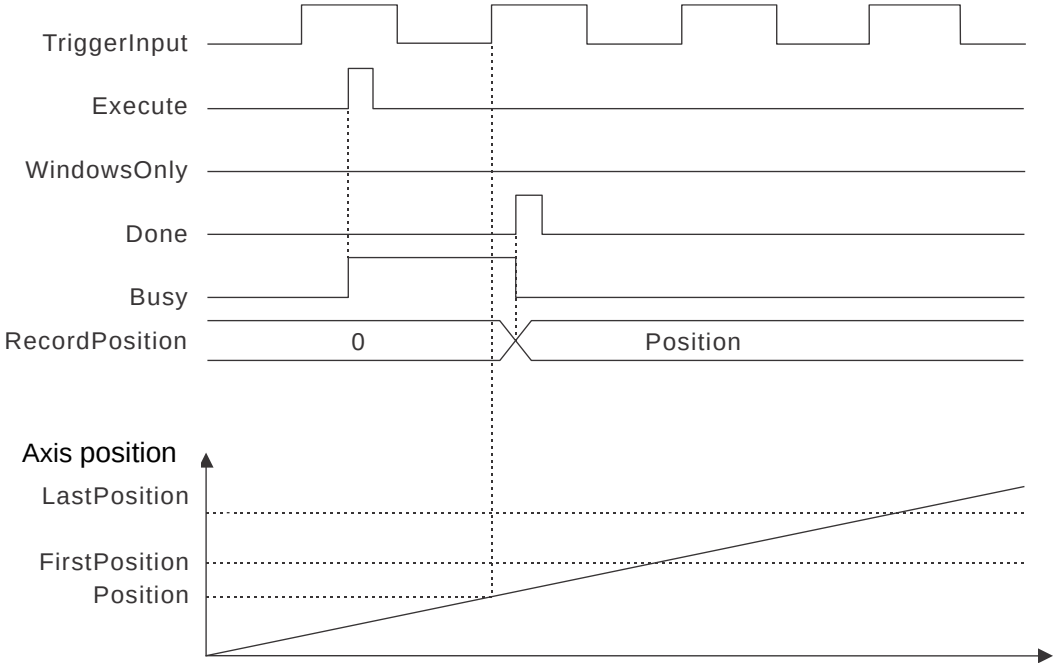
- In WindowsOnly, 1 is valid and 0 is invalid.
- When Disable, the trigger is detected at all axis positions.
- When enabled, only the axis position is detected within the range of FirstPosition (start position) and LastPosition (end position) to trigger.

<Remarks> The moment when WindowsOnly changes from FALSE to TRUE and the time between activation of the lock function cannot be locked.

The lock function takes time to start, so the effective range of WindowsOnly cannot be locked when it is extremely short. The critical value of the lockable effective range depends on the performance of the servo drive and the encoder input terminal and EtherCAT communication.

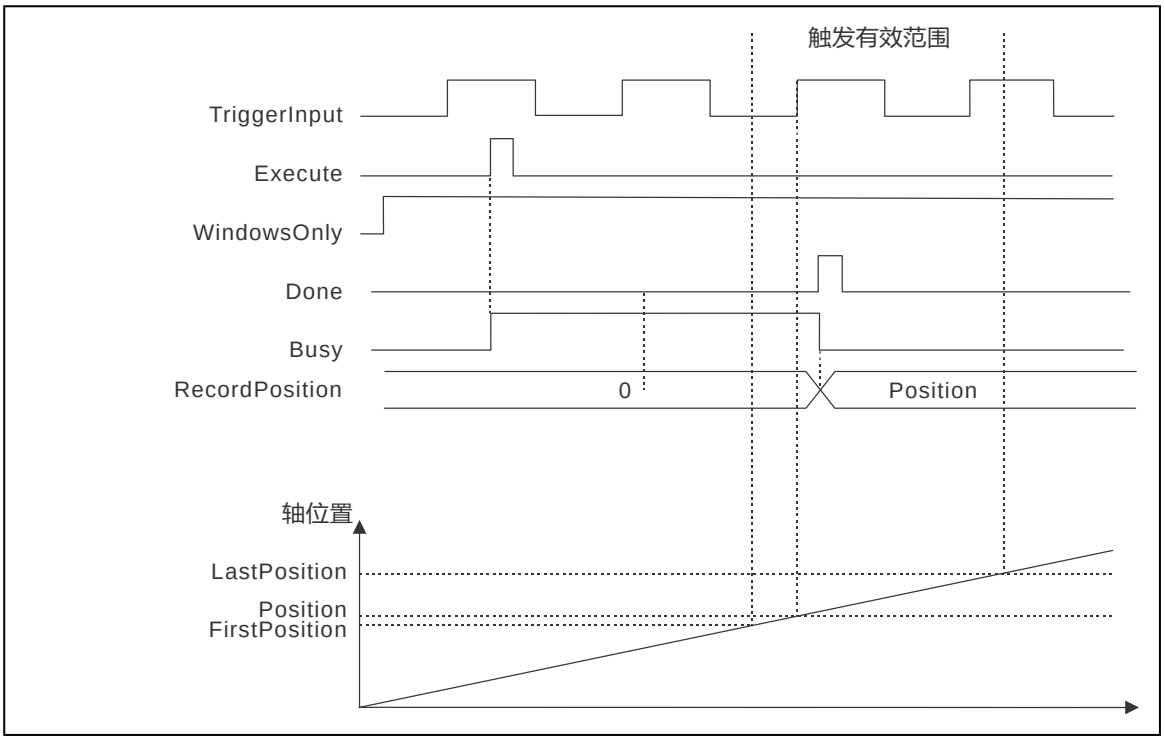
1) WindowsOnly is different between 0 and 1, and the actions are also different, as shown in the following timing diagram.

• When WindowsOnly is 0, after the Execution (execution condition) is changed to TRUE, the axis position initially triggered is output to the Recordposition (lock position).



Window invalid timing diagram

• WindowsOnly is 1, only the trigger input is detected in the range of the window, and the axis position is obtained



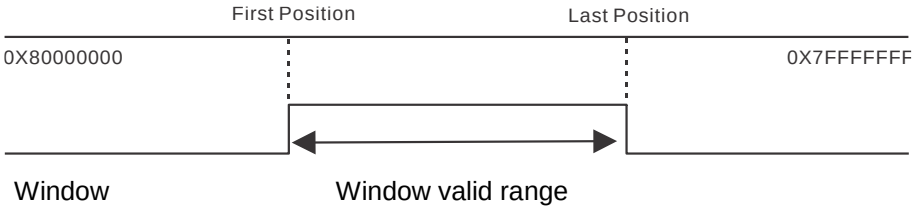
Window effective timing diagram

2) The range of FirstPosition (starting position) and LastPosition (ending position) in different counting modes

Linear mode

The effective range of the window in linear mode is shown in the figure below:

The effective range of the window includes FirstPosition (start position) and LastPosition (end position)



<Remarks> When FirstPostion> LastPostion, LastPostion <=RecordedPostion<= firstPostion window is valid.

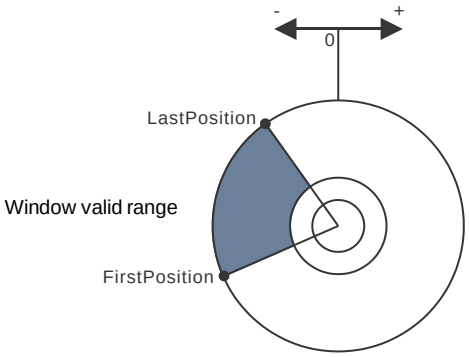
Rotation mode

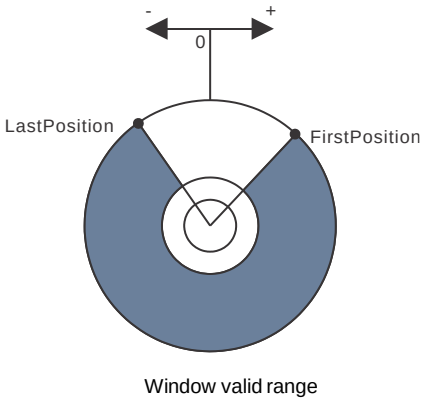
• FirstPosition (start position) ≤ LastPosition (end position), FirstPosition (start position)> LastPosition (end position) both can be specified.

• When FirstPosition (start position)> LastPosition (end position), the set value crosses the upper and lower limit positions of the ring counter.

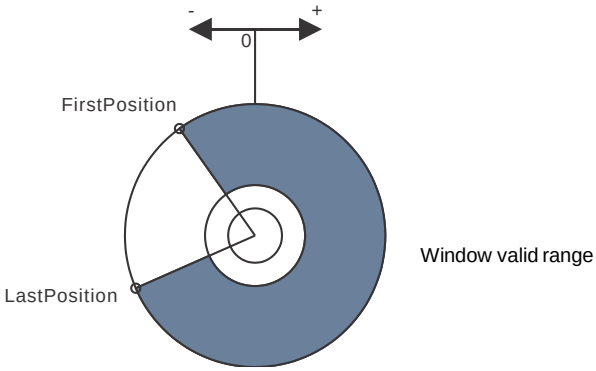
• An exception will occur when the upper and lower limit range of the ring counter is exceeded.

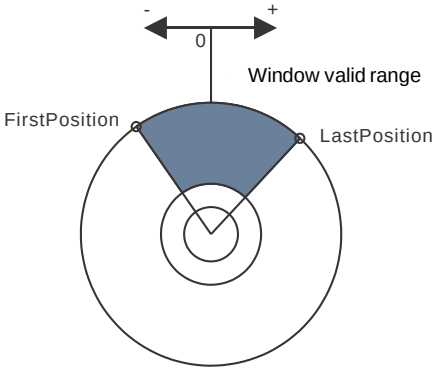
FirstPosition (start position) ≤ LastPosition (end position)





FirstPosition (start position) > LastPosition (end position)





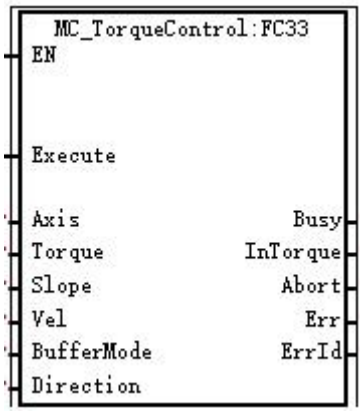
<备注>: When the FirstPostion and LastPostion settings exceed the modulus range of the rotary axis, they will be corrected to fall within the modulus range.

3. Bit definition of driver 16#60B8:

Bit	Describe	
0	Probe 1 enable 0: Probe 1 is not enabled 1: Probe 1 is enabled	Bit0~Bit5: Probe 1 settings
1	Probe 1 trigger mode 0: Single trigger, trigger only when the trigger signal is valid for the first time. 1: Continuous trigger	
2	Probe 1 trigger signal selection 0: DI8 input signal 1: Z signal	
3	NA	
4	Probe 1 rising edge enable 0: The rising edge is not latched 1: rising edge latch	
5	Probe 1 falling edge enable 0: The falling edge is not latched 1: Falling edge latch	
6~7	NA	
8	Probe 2 enable 0: Probe 2 is not enabled 1: Probe 2 is enabled	Bit8~Bit13: Probe 2 settings

9	Probe 2 trigger mode 0: Single trigger, trigger only when the first trigger signal is valid. 1: Continuous trigger	
10	Probe 2 trigger signal selection 0: DI9 input signal 1: Z signal	
11	NA	
12	Probe 2 rising edge enable 0: The rising edge is not latched 1: rising edge latch	
13	Probe 2 falling edge enable 0: The falling edge is not latched 1: Falling edge latch	
14~15	NA	

MC_TorqueControl



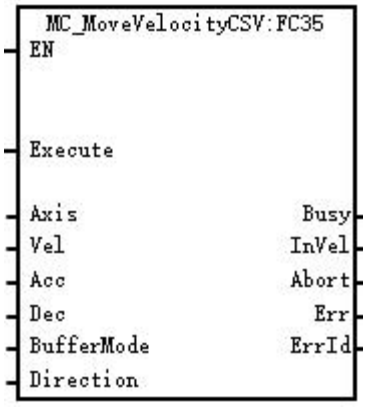
Function: Torque control command, through the torque mode to control the standard 402 axis for torque motion, PDO needs to add three object dictionaries 0x6071, 0x6060 and 0x6071. this command is only used for communication axis, not for pulse axis and virtual axis.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Torque	IN	Target torque	REAL	Specify the target torque output to the servo drive in units of "0.1%". Specify the target torque at the ratio when the rated torque is "100.0%"
Slope	IN	Torque acceleration	REAL	The torque change rate from the current torque to the target torque, unit: %/s
Vel	IN	Speed limit	REAL	Limit speed, unit: unit/second

BufferMode	IN	Abort mode	BYTE	0: Abort (directly interrupt the ongoing command) Other: illegal
Direction	IN	direction	INT	1: Positive direction -1: negative direction
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset..
InTorque	OUT	Torque reach position	BOOL	After reaching the target torque, the InTorque bit is set. when the execution condition of the instruction changes from On to Off, the InTorque bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveVelocityCSV



Function: Control the terminal actuator to move to a given speed according to a given acceleration and deceleration, and run at a constant speed. When the terminal mechanism reaches a given speed, the command execution is completed, but the terminal mechanism continues to run at this speed.

This instruction has the same function as MC_MoveVelocity, but uses the periodic synchronous speed mode to control the servo axis. Three object dictionaries 0x6060, 0x6061 and 0x60ff need to be added to PDO. This instruction is only used for communication axis.

Note that when calling this instruction, the MC_MoveSuperImosed instruction cannot be called for motion superimposition.

Parameter Description:

Parameter	Input/ output	Description	Type data	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Vel	IN	speed	REAL	The operating speed of the terminal actuator, this parameter is always positive. (Unit: unit/second).
Acc	IN	Acceleration	REAL	The acceleration of the terminal actuator, this parameter is always positive.
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive.
BufferMode	IN	Abort mode	BYTE	(Unit: unit/second/second)
Direction	IN	direction	INT	1: Positive direction -1: negative direction
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset..
Invel	OUT	Speed reached bit	BOOL	After reaching the target speed, the Invel bit is set. when the execution condition of the instruction changes from On to Off, the Invel bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	Error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_ReadVelPos

Function: Read the speed and position of the axis.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Axis	IN	Axis ID number	BYTE	axis number
Err	OUT	error bit	BOOL	If an error is detected, the err bit is set.
ErrId	OUT	error code	WORD	When Err is TRUE, ErrId is the error of the instruction.
Actpos	OUT	Current actual location	REAL	unit
ActVel	OUT	Current actual speed	REAL	unit/sec
Setpos	OUT	Current setting position	REAL	unit
SetVel	OUT	Current setting speed	REAL	unit/sec

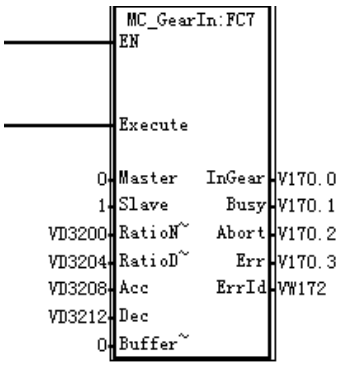
G.2 Synchronous Control Instructions

Synchronous control is generally used for electronic gears, electronic cams, phase shift commands, etc.

The MC function blocks commonly used in synchronous control instructions are as follows:

Name	Command function	Remark
MC_GearIn	Electronic gear coupling instruction	
MC_GearOut	Electronic gear disengagement command	
MC_CamTableSelect	Cam table selection command	
MC_CamIn	Electronic cam related instructions	
MC_CamOut	Disable electronic cam related instructions	
MC_Phasing	Phase shift command	
MC_MoveSuperImposed	Superimposed relative displacement command	

MC_GearIn



Function: This command is used to establish the gear relationship between the master and slave axes. When establishing gear relationship, parameters such as gear ratio can be set. After the

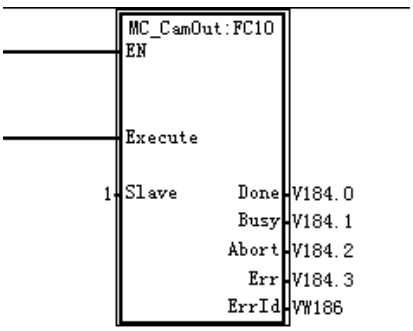
gear relationship is established, the slave axis follows the main motion with a given proportional relationship to realize the synchronous control of the master and slave axis.

Slave axis speed after synchronization = main axis speed * RatioNum/RatioDen.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Master	IN	Spindle ID number	BYTE	
Slave	IN	Slave axis ID number	BYTE	
RatioNum	IN	Numerator of electronic gear	REAL	Not 0
RetioDen	IN	Denominator of electronic gear	REAL	Not 0
Acc	IN	Acceleration	REAL	
Dec	IN	decrease speed	REAL	
BufferMode	IN	Abort mode	BYTE	0: Abort (directly interrupt the ongoing command) Other: illegal
InGear	OUT	Coupling complete bit	BOOL	When the coupling is complete, InGear is set.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_GearOut

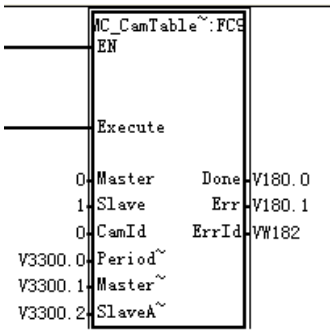


Function: This command is used to cancel the gear relationship between the master and slave axes. After the relationship is released, the slave axis will continue to run at the speed before the separation.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Slave	IN	Slave axis ID number	BYTE	
Done	OUT	Done bit	BOOL	When the cancel gear association instruction is executed successfully, the Done bit is set. When the instruction execution condition is Off, the Done bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	c	WORD	Error code, see chapter G.4 Error code

MC_CamTableSelect



Function: This instruction is used to select the cam curve and at the same time specify the mode when the master and slave axes are associated.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Master	IN	Spindle ID number	BYTE	
Slave	IN	Slave axis ID	BYTE	

		number		
CamId	IN	ID number of cam table	BYTE	
Periodic	IN	cycle	BOOL	When this parameter is 1, the slave axis will execute the cam movement periodically. When this parameter is 0, the slave axis only executes one cycle of cam movement.
MasterAbsolute	IN	Master axis absolute	BOOL	When this parameter is 1, the master axis is in absolute mode.
SlaveAbsolute	IN	Slave axis absolute	BOOL	When this parameter is 0, the slave axis is in relative mode.
Done	OUT	Done bit	BOOL	When the cam parameter setting is successful, the Done bit is set. When the instruction execution condition is Off, the Done bit is reset.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset..
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

The starting address of the cam data is specified in the basic configuration. After configuration, a data block will be generated (the function is temporarily not implemented), or the starting address (V memory, symbol table) will be specified. When the polynomial mode is selected for the cam table, the data records are as follows:

XYVZ mode:

No.	Name	Type	Comment
1		STRUCT	
2	xStart	REAL	X axis coordinate starting point
3	xEnd	REAL	X axis coordinate end point
4	yStart	REAL	Y axis coordinate starting point
5	yEnd	REAL	Y-axis coordinate end point
6	nElements	WORD	Number of elements
7	dX0	REAL	0th X coordinate
8	dY0	REAL	0th Y coordinate
9	dV0	REAL	0th speed
10	dA0	REAL	0th acceleration
11	dX1	REAL	1st X coordinate
12	dY1	REAL	1st Y coordinate
13	dV1	REAL	1st speed
14	dA1	REAL	1st acceleration
15	dX2	REAL	2nd X coordinate
	...	...	...
		END_STRUCT	

One-dimensional array mode:

No.	Name	Type	Comment
1		STRUCT	
2	xStart	REAL	X axis coordinate starting point
3	xEnd	REAL	X axis coordinate end point
4	yStart	REAL	Y axis coordinate starting point
5	yEnd	REAL	Y-axis coordinate end point
6	nElements	WORD	Number of elements
7	dY0	REAL	0th Y coordinate
8	dY1	REAL	1st Y coordinate
9	dY2	REAL	2nd Y coordinate
	...	...	...
		END_STRUCT	

Two-dimensional array mode:

No.	Name	Type	Comment
1		STRUCT	
2	xStart	REAL	X axis coordinate starting point
3	xEnd	REAL	X axis coordinate end point
4	yStart	REAL	Y axis coordinate starting point
5	yEnd	REAL	Y-axis coordinate end point
6	nElements	WORD	Number of elements
8	dX0	REAL	0th X coordinate
12	dY0	REAL	0th Y coordinate
15	dX1	REAL	1st X coordinate
	dY1	REAL	1st Y coordinate
	dX2	REAL	2nd X coordinate
	dY2	REAL	2nd Y coordinate
	...	...	...
		END_STRUCT	

By modifying the data in this memory, the cam table data can be changed. After modifying the data, call MC_CamTableSelect to notify the PLC of the change of cam table data. If the system is in the running of the cyclic cam table at this time, the changed data will take effect in the next cam cycle. To take effect immediately, call the MC_CamIn instruction immediately after calling MC_CamTableSelect.

Cam offset and Scaling:

The position of the Master input conversion is calculated by the following formula, and the converted X is used as the output of the cam:

Calculation formula: **X=MasterScaling*MasterPosition+MasterOffset**

Therefore, if the ratio of the Master is greater than 1, the cam will run at a higher speed, if the ratio is less than 1, the rate will decrease accordingly.

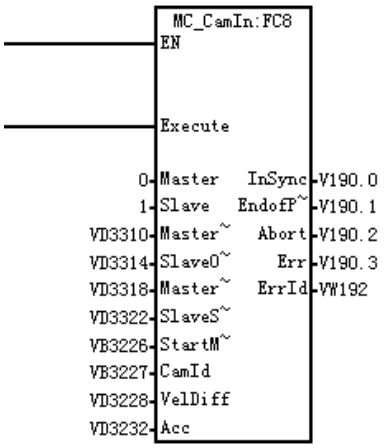
SlaveOffset, SlaveScaling:

Calculation formula: **Y=SlaveScaling*CAM(X)+SlaveOffset**

If SlaveScaling>1 causes the cam effect to stretch, the range of the slave axis will increase. if SlaveScaling<1 will cause a contraction.

When the CAM table data is unchanged, this command only needs to be called once, and there is no need to call it later. When the CAM table data changes, this command needs to be called again.

MC_CamIn



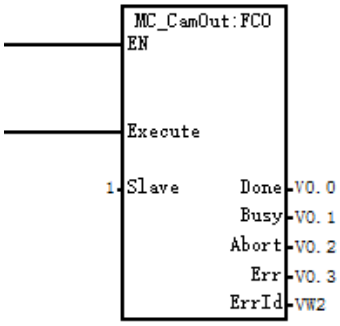
Function: Establish the cam relationship between the master and slave axes. When establishing the cam relationship, you can specify the offset value, zoom ratio and start mode of the master and slave axes according to application requirements. When the instruction is executed, the slave axis follows the main axis movement according to the cam curve

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Master	IN	Spindle ID number	BYTE	
Slave	IN	Slave axis ID number	BYTE	
MasterOffset	IN	Master Offset	REAL	The Master axis position is offset. Unit: Unit
SlaveOffset	IN	Slave Offset	REAL	The cam position of the slave axis is offset. Unit: Unit
MasterScaling	IN	Master zoom	REAL	Spindle zoom ratio configuration parameter. This parameter is not 0
SlaveScaling	IN	Slave zoom	REAL	Configure the parameters of the slave axis zoom ratio. This parameter is not 0
StartMode	IN	Start Mode	BYTE	0: Forward immediate jump 1: Acceleration (shortest distance) jump 2: Slope forward rotation 3: Slope reversal Note: forward rotation and reverse rotation are only for the case that the slave axis is a rotating axis. When the Startmode of the linear axis is 0 or1, it is jump, and 2,3 is

				slope start. The four modes of this parameter are only for the slave axis, and the slave axis is absolute or relative mode by MC_ Camtableselect command configuration.
CamId	IN	ID of the CAM table	BYTE	
VelDiff	IN	Cam table coupling speed	REAL	
Acc	IN	Cam table acceleration	REAL	
InSync	OUT	Cam relationship establishment	BOOL	After the master axis and slave axis establish a relationship, InSync is set. When the execution condition of the instruction is Off, the InSync bit is reset.
Abort	OUT	Command stop bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	error code,See attached table 1 for details
EndOfProfile	OUT	Cam relationship ends	BOOL	If the Period parameter is 0 when the MC_CamTableSelect instruction is executed, EndOfProfile is set after the cam profile is executed once. The Perioc parameter is 1. When the cam profile is executed once, EndOfProfile is set and reset after one cycle. When the instruction execution condition is Off, the EndOfProfile bit is reset.

MC_CamOut

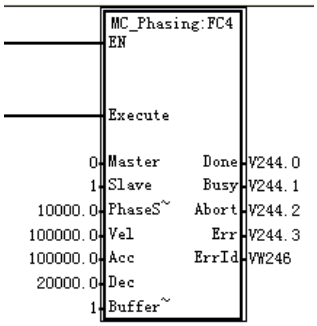


Function: Release the cam relationship between the master and slave axis. After the relationship is released, the slave axis will continue to move at the speed before the separation.

Parameter Description:

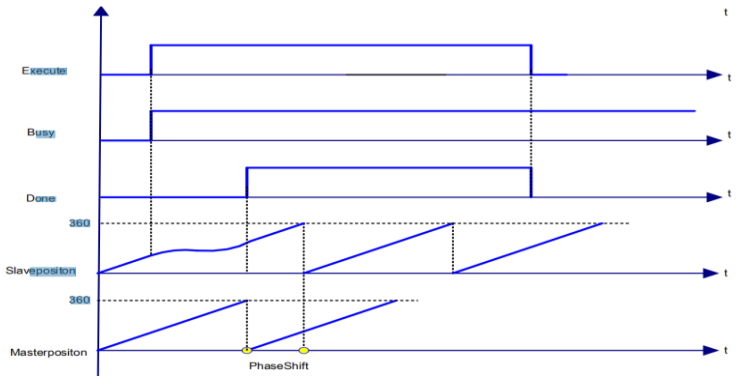
Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Slave	IN	Slave axis ID number	BYTE	
Done	OUT	Done bit	BOOL	When the cam related instruction is canceled, the Done bit is set. When the instruction execution condition is Off, the Done bit is reset.
Abort	OUT	Abort bit	BOOL	When the instruction is terminated during execution, the abort bit is set.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_Phasing



Function: This instruction is used to adjust the phase difference between the master axis and the slave axis. After the instruction is called, the phase difference between the master axis and the slave axis is PhaseShift, that is, the position of the slave axis = the position of the master axis-PhaseShift.

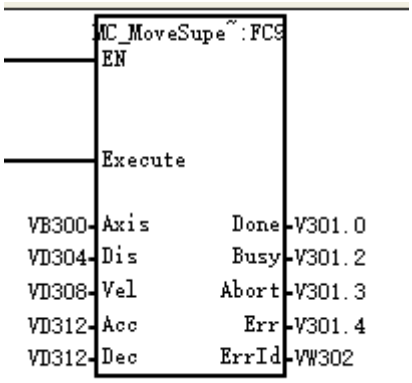
As shown in the figure below, the master and slave axes move in 360 cycles, and the adjustment is performed on the rising edge of the Execute signal. After the adjustment is completed, the phase deviation between the slave axis and the master axis is the value set by PhaseShift.



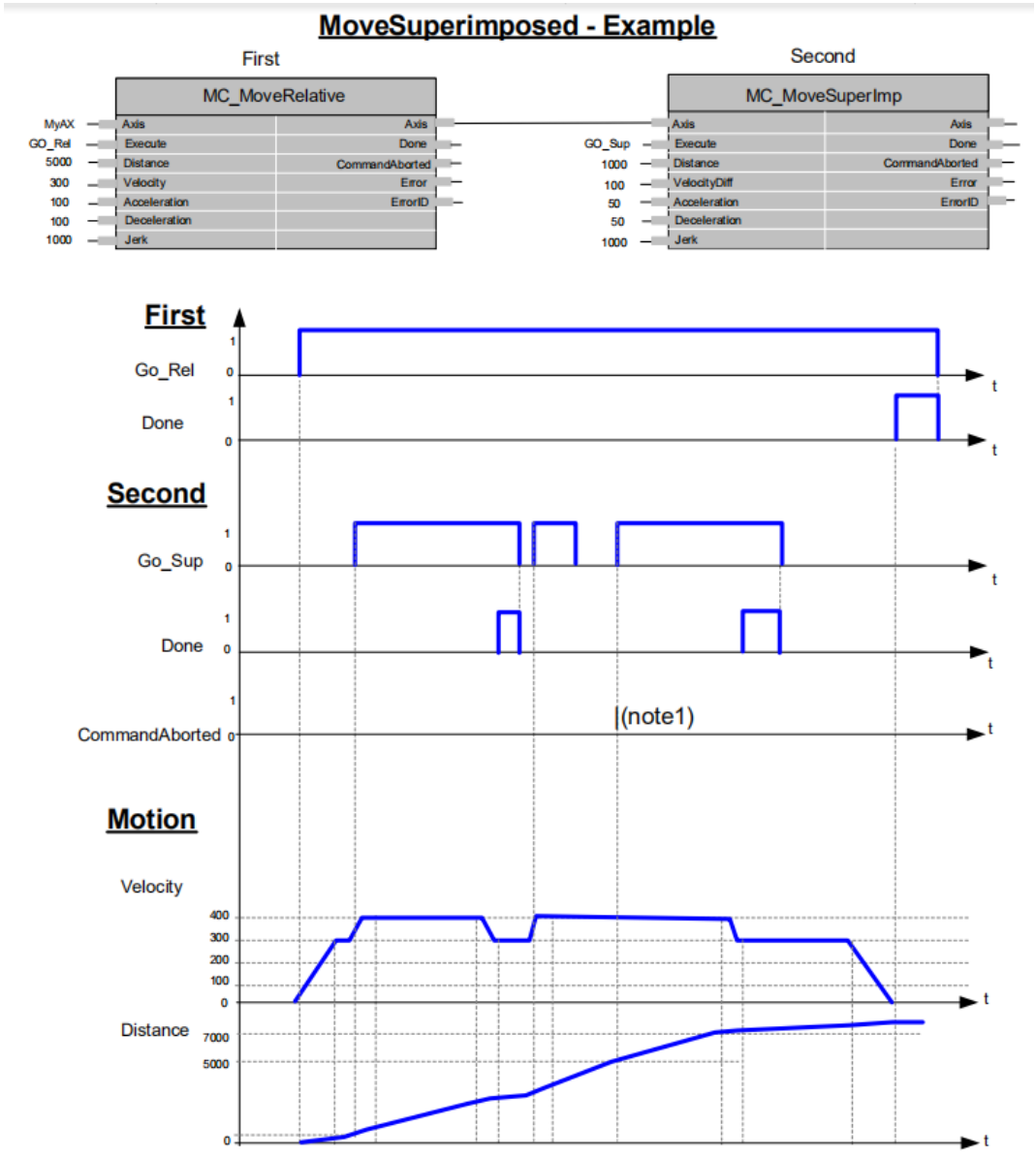
Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Master	IN	Spindle ID number	BYTE	
Slave	IN	Slave axis ID number	BYTE	
PhaseShift	IN	Phase shift	REAL	Phase shift
Vel	IN	speed	REAL	Phase adjustment speed.
Acc	IN	Acceleration	BOOL	Phase adjusted acceleration.
Dec	IN	decrease speed	BOOL	Phase adjustment deceleration.
BufferMode	IN	Abort mode	BYTE	0: Abort (directly interrupts an in-progress instruction) Others: illegal
Done	OUT	Phase offset adjustment completed	BOOL	Done is set when the phase adjustment is completed. When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Command processing	BOOL	True, the command is being processed.
Abort	OUT	Abort bit	BOOL	When the instruction is terminated during execution, the Abort bit is set. When the execution condition of the instruction is off, the Abort bit is reset.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveSuperImposed



Function: Control the terminal actuator to independently add a given distance according to the given speed and acceleration/deceleration in the current motion state. When this instruction is executed, the execution of the previous instruction will not be terminated. The two instructions are executed at the same time, distance, speed, acceleration and deceleration will be superimposed in real time. When one of the instructions is executed, its speed, acceleration and deceleration will no longer be superimposed, and the other instruction will still run independently.

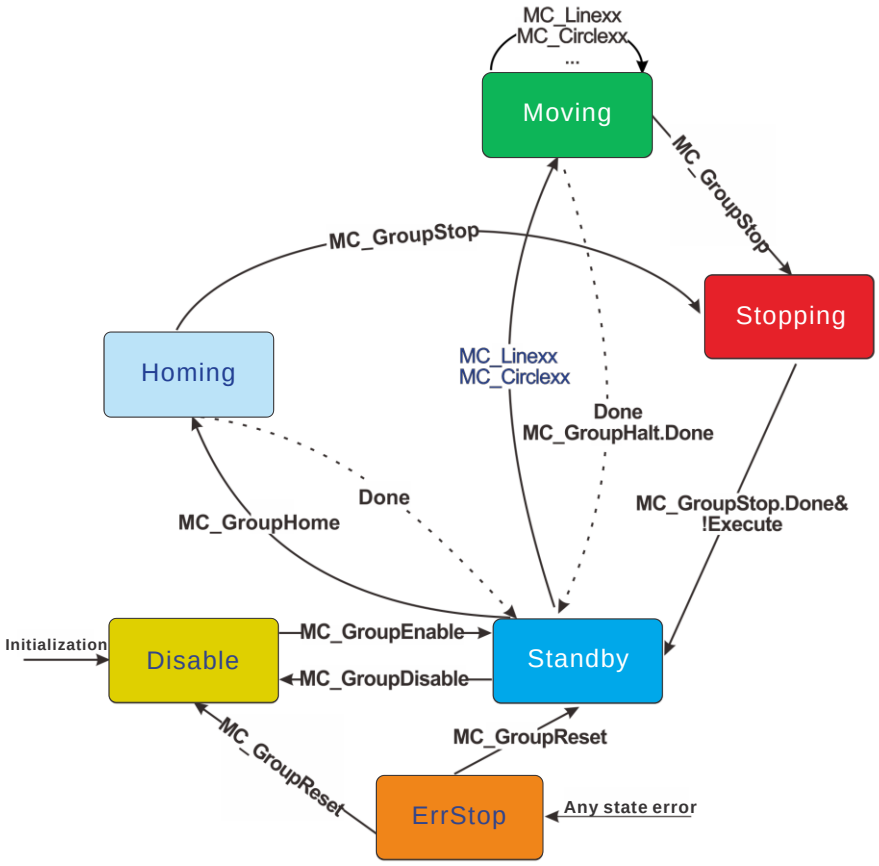


Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
Axis	IN	Axis ID number	BYTE	
Dis	IN	distance	REAL	Taking the current position as the reference, the target distance to be moved by the terminal actuator. If the value is set to negative, it means that the axis will be reversed. Unit: unit.
Vel	IN	speed	REAL	The operating speed of the terminal actuator. This parameter is always positive. (unit: Unit / second)
Acc	IN	Acceleration	REAL	Acceleration of terminal actuator, this parameter is always positive. (unit: Unit / second / second)
Dec	IN	deceleration	REAL	Deceleration of terminal actuator, this parameter is always positive. (unit: Unit / second / second)
Done	OUT	Done bit	BOOL	When the absolute displacement action is completed, the Done bit is set. When the execution condition of the instruction is off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The busy bit is set. When the command completes busy reset. When the execution condition of the instruction is off, the busy bit is reset.
Abort	OUT	Abort bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	错Error code, see chapter G.4 Error code

G.3 Axis Group Command

The motion control commands of the axis group are generally used for multi-axis linear interpolation and circular interpolation commands. The flow chart of the axis group command status is shown in the figure below:



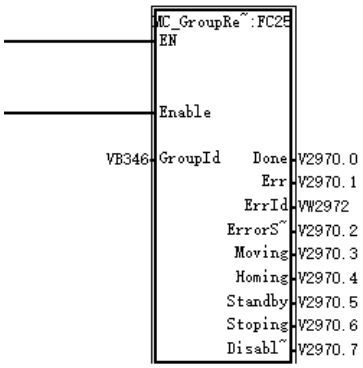
Tips

The functions of MC Group Home and Group Stop in the figure have not been implemented yet. Therefore, the Homing and Stopping shaft group shapes are temporarily retained.

MC function blocks commonly used in axis group instructions are as follows:

Name	Command function
MC_GroupReadState	Read axis group status command
MC_GroupEnable	Axis group effective command
MC_GroupDisable	Invalid axis group command
MC_GroupReset	Axis group error reset command
MC_GroupHalt	Axis group stop command
MC_MoveLinerRelative	Relative displacement linear interpolation command
MC_MoveLinerAbsolute	Absolute displacement linear interpolation command
MC_MoveCircularRalative	Relative displacement arc interpolation command
MC_MoveCircularAbolute	Absolute displacement arc interpolation command
MC_Helical	Spiral interpolation command

MC_GroupReadState

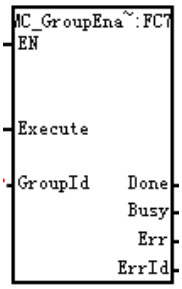


Function: This instruction is used to read the value of the corresponding axis group status parameter.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Enable	IN	Instruction execution condition	BOOL	When it is On, execute the instruction.
GroupId	IN	Axle group ID	BYTE	Axis group number
Done	OUT	Status bit	BOOL	After the instruction is executed successfully, the Done bit is set.
Err	OUT	error bit	BOOL	If an error is detected, the Err bit is set. When the execution condition of the instruction changes from On to Off, the Err bit is reset.
ErrId	OUT	error code	WORD	When Err is TRUE, ErrId is the error of the instruction. When ErrStop is TRUE, ErrId is the current error of the axis group.
ErrStop	OUT	Status indicator bit.	BOOL	TRUE, if the axis group is in the state ErrStop
Stopping	OUT	Status indicator bit.	BOOL	TRUE, if the axis group is in the Stopping state
Moving	OUT	Status indicator bit.	BOOL	TRUE, if the axis group is in the Moving state
Standby	OUT	Status indicator bit.	BOOL	TRUE, if the axis group is in the Standby state
Homing	OUT	Status indicator bit.	BOOL	TRUE, if the axis group is in the state Homing
Disabled	OUT	Status indicator bit.	BOOL	TRUE, if the axis group is in the state Disabled

MC_GroupEnable

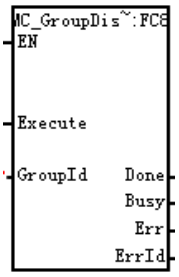


Function: Make the axis group effective. The axis group status GroupDisalbed changes to GroupStandby.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
GroupId	IN	Axle group ID	BYTE	Axis group number
Busy	OUT	Busy bit	BOOL	The instruction is being executed as TRUE, and the execution is completed as FLASE
Done	OUT	Status bit	BOOL	After the instruction is executed successfully, the Done bit is set.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_GroupDisable



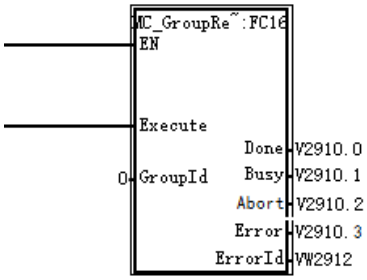
Function: make the axis group invalid. Change the axis group state from GroupStandby to GroupDisalbed.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
GroupId	IN	axle group ID	BYTE	Axis group number
Done	OUT	status bit	BOOL	After the instruction is executed successfully, the Done bit is set.
Busy	OUT	Busy bit	BOOL	The instruction is being executed as TRUE, and the execution is completed as FLASE
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set.

ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_GroupReset

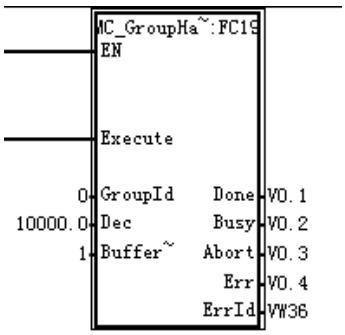


Function: When the axis group is in the ErrStop state, call this instruction to clear the axis group and axis error, and switch the axis group state to Disabled (the axis group was not enabled before the error) or Standby.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
GroupId	IN	Axle group ID	BYTE	Axis group number
Done	IN	Instruction complete bit	BOOL	Done bit
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Reset Busy when the instruction completes. When the execution condition of the instruction is Off, the Busy bit is reset.
Abort	OUT	Command stop bit	BOOL	Keep the interface, it will not appear
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_GroupHalt



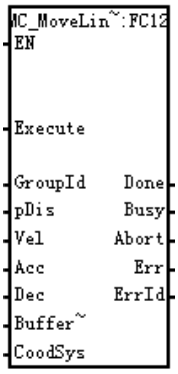
Function: decelerate and stop all axes in interpolation action.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution	BOOL	When the execution condition changes from Off to On, the instruction is executed.

		condition		
GroupId	IN	Axle group ID	BYTE	Axis group number
Dec	IN	decrease speed	REAL	Deceleration of terminal actuator, this parameter is always positive. (unit: Unit / s). When it is set to 0, the emergency stop is realized.
BufferMode	IN	Abort mode	BYTE	0: abort (directly interrupt the instruction in progress) Others: illegal
Done	OUT	Done bit	BOOL	After the instruction is executed successfully, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Abort	OUT	Abort bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Reset Busy when the instruction completes. When the execution condition of the instruction is Off, the Busy bit is reset.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset..
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveLinerRelative



补 Function: This command interpolates linearly according to the relative displacement

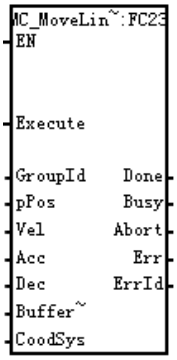
Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
GroupId	IN	Axle group ID	BYTE	

pDis	IN	Relative position pointer	DWORD	Pointer to target distance. The target distance is described as 6 REAL types of data. Representation (spatial displacement under MCS) X: REAL Y: REAL Z: REAL A: REAL B: REAL C: REAL Or (displacement of each axis under ACS) A0: REAL A1: REAL A2: REAL A3: REAL A4: REAL A5: REAL (Unit: Unit)
Vel	IN	speed	REAL	The operating speed of the terminal actuator, this parameter is always positive. (Unit: unit/second)
Acc	IN	Acceleration	REAL	The acceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
BufferMode	IN	Abort mode	BYTE	0: Abort (Interrupt the ongoing command directly and disable the transition) 1: Buffered(Function block that starts to execute buffering after the deceleration of the previous stage is completed) 2: BendingPrevious(Go to the end of the previous segment at the current speed and start the next segment at the rate of the previous segment, disable transitions) 16#12: TMCorner (additional angle transition, transition to the next segment with an additional angle when the previous segment starts to decelerate). see the detailed introduction of BufferMode continuous interpolation for details
CoordSystem	IN	coordinate system	BYTE	0: ACS 1: MCS (only this is supported) 2: WCS 3: PCS_1 4: PCS_2
Done	OUT	Done bit	BOOL	When the absolute displacement action is completed, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Abort	OUT	Abort mode	BOOL	When the instruction is terminated during execution, the abort bit is set.

				When the execution condition of the instruction is off, the abort bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Reset Busy when the instruction completes. When the execution condition of the instruction is Off, the Busy bit is reset..
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveLinerAbsolute



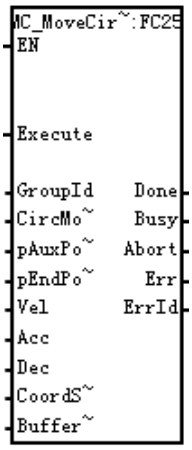
Function: This command performs linear interpolation according to absolute displacement.

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
GroupId	IN	Axle group ID	BYTE	
pPos	IN	Absolute position pointer	DWORD	A pointer to the target location. The target location is data described as 6 REAL types. Representation (spatial displacement under MCS) X: REAL Y: REAL Z: REAL A: REAL(reserved) B: REAL (reserved) C: REAL(reserved) Or (displacement of each axis under ACS) A0: REAL A1: REAL A2: REAL (reserved for two axes) A3: REAL(reserved) A4: REAL(reserved) A5: REAL(reserved) (unit: Unit)
Vel	IN	Speed	REAL	The operating speed of the terminal actuator, this parameter is always positive.

				(Unit: unit/second)
Acc	IN	Acceleration	REAL	The acceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
BufferMode	IN	Abort mode	BYTE	0: Abort (Interrupt the ongoing command directly and disable the transition) 1: Buffered(Function block that starts to execute buffering after the deceleration of the previous stage is completed) 2: BendingPrevious(Go to the end of the previous segment at the current speed and start the next segment at the rate of the previous segment, disable transitions) 16#12: TMCorner (additional angle transition, transition to the next segment with an additional angle when the previous segment starts to decelerate). see the detailed introduction of BufferMode continuous interpolation for details
CoordSystem	IN	coordinate system	BYTE	0: ACS 1: MCS (only this is supported) 2: WCS 3: PCS_1 4: PCS_2
Done	OUT	Done bit	BOOL	When the absolute displacement is completed, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Abort	OUT	Abort bit	BOOL	When the instruction is terminated during execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Reset Busy when the instruction completes. When the execution condition of the instruction is Off, the Busy bit is reset..
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveCircularRalative



Function: This command performs circular interpolation according to relative displacement.

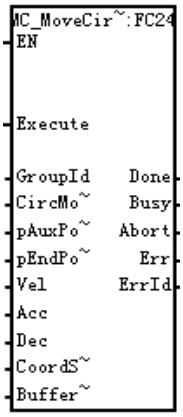
Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
GroupId	IN	Axle group ID	BYTE	
CircMode	IN	Arc interpolation mode	BYTE	Bit0~Bit3 0: Three-point type 1: Center point type. 8: Three-point full circle 9: Center point full circle Bit4: Interpolation direction (valid for center point and full circle) 0: Invert (CCW) 1: Forward rotation (CW) The direction of the arc is defined according to the right-hand rule: the CCW direction is along the direction of the bend of the finger. When the center point type, if the center point is not exactly the same distance from the start and end points (this is usually due to the limited accuracy of programming the center point), it is projected on the vertical bisector of the start and end points. Once the center point is too far away (more than 1% of the radius), an error is returned.
pAuxPoint	IN	reference position pointer	DWORD	A pointer to a reference location. In three-point mode, pAuxPostion refers to the passed reference point. When the center point type, pAuxPostion refers to the center of the circle. The reference location is data described as 6 REAL types. Representation (spatial displacement under MCS) X: REAL Y: REAL

				Z: REAL A: REAL(Reserve) B: REAL(Reserve) C: REAL(Reserve) (unit: unit)
pEndPoint	IN	target position pointer	DWORD	A pointer to the target location. The target location is data described as 6 REAL types. Representation (spatial displacement under MCS) X: REAL Y: REAL Z: REAL A: REAL(Reserve) B: REAL(Reserve) C: REAL(Reserve)
Vel	IN	speed	REAL	The running speed of the terminal actuator, this parameter is always positive. (unit: unit/s)
Acc	IN	Acceleration	REAL	Acceleration of the terminal actuator, this parameter is always positive. (unit: unit/s/s)
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (unit: unit/s/s)
BufferMode	IN	Abort mode	BYTE	0: Abort (Interrupt the ongoing command directly and disable the transition) 1: Buffered(Function block that starts to execute buffering after the deceleration of the previous stage is completed) 2: BendingPrevious(Go to the end of the previous segment at the current speed and start the next segment at the rate of the previous segment, disable transitions) 16#12: TMCorner (additional angle transition, transition to the next segment with an additional angle when the previous segment starts to decelerate). see the detailed introduction of BufferMode continuous interpolation for details
CoordSystem	IN	coordinate system	BYTE	0: ACS 1: MCS (only this is supported) 2: WCS 3: PCS_1 4: PCS_2
Done	OUT	done bit	BOOL	Done is set when the absolute displacement action is completed. Done is reset when the execution condition of the instruction is off.
Abort	OUT	Command stop bit	BOOL	Abort is set when the instruction execution process is terminated. Abort is reset when the execution condition of the instruction is off.
Busy	OUT	Instruction	BOOL	When the instruction is executed. The Busy

		execution bit		bit is set. Reset Busy when the instruction completes. When the execution condition of the instruction is off, the Busy bit is reset.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_MoveCircularAbsolute



Function: Perform circular interpolation according to absolute displacement.

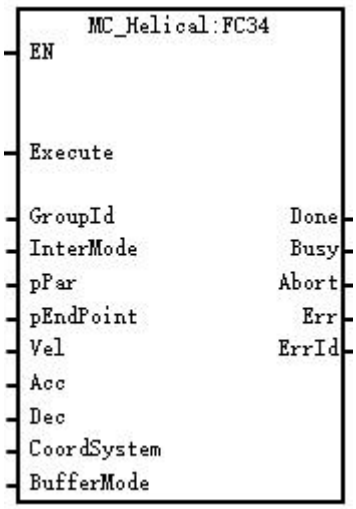
Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	When the execution condition changes from Off to On, the instruction is executed.
GroupId	IN	Axle group ID	BYTE	
CircMode	IN	Arc interpolation mode	BYTE	Bit0-Bit3 0: Three-point 1: Center point 8: Three-point complete circle 9: Center point complete circle Bit4: Interpolation direction (the center point type is valid) 0: reverse (CCW) 1: Forward rotation (CW) The direction of the arc is defined according to the right-hand rule: the CCW direction is along the direction of the bend of the finger. When the center point type, if the center point is not exactly the same distance from the start and end points (this is usually due to the limited accuracy of programming the center point), it is projected on the vertical bisector of the start and end points. Once the center point is too far away (more than 1% of the radius), an error is returned..

pAuxPoint	IN	reference position pointer	DWORD	A pointer to a reference location. In three-point mode, pAuxPostion refers to the passed reference point. In the center point type, pAuxPostion refers to the center of the circle. The reference position distance is described as 6 REAL type data. Representative (under MCS, PCS, WCS) X: REAL Y: REAL Z: REAL A: REAL(Reserve) B: REAL(Reserve) C: REAL(Reserve) (Unit: Unit)
pEndPoint	IN	end position pointer	DWORD	Pointer to the end location. The end position distance is described as 6 REAL type data. X: REAL Y: REAL Z: REAL A: REAL(Reserve) B: REAL(Reserve) C: REAL(Reserve) (unit: unit)
Vel	IN	speed	REAL	The operating speed of the terminal actuator, this parameter is always positive.
Acc	IN	Acceleration	REAL	(Unit: unit/second)
Dec	IN	decrease speed	REAL	The acceleration of the terminal actuator, this parameter is always positive.
BufferMode	IN	Abort mode	BYTE	0: Abort (Interrupt the ongoing command directly and disable the transition) 1: Buffered(Function block that starts to execute buffering after the deceleration of the previous stage is completed) 2: BendingPrevious(Go to the end of the previous segment at the current speed and start the next segment at the rate of the previous segment, disable transitions) 16#12: TMCorner (additional angle transition, transition to the next segment with an additional angle when the previous segment starts to decelerate). see the detailed introduction of BufferMode continuous interpolation for details
CoordSystem	IN	coordinate system	BYTE	0: ACS. 1: MCS (Currently only these 2 are supported) 2: WCS 3: PCS_1. 4: PCS_2
Done	OUT	Done bit	BOOL	When the absolute displacement action is completed, the Done bit is set.

Abort	OUT	Command stop bit	BOOL	When the execution condition of the instruction is Off, the Done bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Reset Busy when the instruction completes. When the execution condition of the instruction is Off, the Busy bit is reset.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

MC_Helical



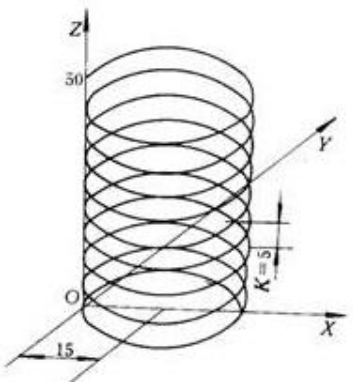
Function: This command is interpolated in a spiral way

Parameter Description:

Parameter	Input/ output	Description	Data type	Remark
Execute	IN	Instruction execution condition	BOOL	This command is executed when the execution condition changes from Off to On.
GroupId	IN	Axle group ID	BYTE	
InterMode	IN	Circular interpolation method	BYTE	0: XY plane counterclockwise circle, relative position, ignoring the XY coordinates of the end point. 1: Clockwise circle on xy plane, relative position, ignoring the XY coordinates of the end point. 0x10: The xy plane is counterclockwise, the relative position, the XY coordinates of the end point are not on the helix, and an error will be reported. 0x11: Clockwise circle on xy plane, relative position, XY coordinates of the end point are not on the helix, and an error will be reported.

pPar	IN	Parameter pointer	DWORD	pointer to parameter pPar points to the parameters of helical interpolation, described as 6 REAL types. 0: The abscissa of the center of the circle on the arc plane. 1: The ordinate of the center of the circle on the arc plane. 2: The lead of the helix. The distance the helix moves around the cylinder once it moves along the axis of the cylinder. 3, 4, 5: reserved. (unit: unit)
pEndPoint	IN	Target position pointer	DWORD	A pointer to the target location. The target position is described as 6 REAL types of data. Represents the spatial displacement (relative) under MCS X: REAL Y: REAL Z: REAL A: REAL(Reserve) B: REAL(Reserve) C: REAL(Reserve)
Vel	IN	speed	REAL	The operating speed of the terminal actuator, this parameter is always positive. (Unit: unit/second)
Acc	IN	Acceleration	REAL	The acceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
Dec	IN	decrease speed	REAL	The deceleration of the terminal actuator, this parameter is always positive. (Unit: unit/second/second)
BufferMode	IN	Abort mode	BYTE	0: Abort (Interrupt the ongoing command directly and disable the transition) 1: Buffered(Function block that starts to execute buffering after the deceleration of the previous stage is completed) 2: BendingPrevious(Go to the end of the previous segment at the current speed and start the next segment at the rate of the previous segment, disable transitions) 16#12: TMCorner(Additional angle transition, transition to the next segment at an additional angle when the previous segment starts to decelerate)
CoordSystem	IN	coordinate system	BYTE	0: ACS 1: MCS (Only this is supported) 2: WCS 3: PCS_1 4: PCS_2
Done	OUT	Done bit	BOOL	When the action is completed, the Done bit is set. When the execution condition of the instruction is Off, the Done bit is reset.
Abort	OUT	Command	BOOL	When the instruction is terminated during

		stop bit		execution, the abort bit is set. When the execution condition of the instruction is off, the abort bit is reset.
Busy	OUT	Instruction execution bit	BOOL	When the instruction is executed. The Busy bit is set. Busy reset when the instruction is completed. When the execution condition of the instruction is Off, the Busy bit is reset.
Err	OUT	error bit	BOOL	If an error is detected, the Error bit is set. When the execution condition of the instruction is Off, the Error bit is reset.
ErrId	OUT	error code	WORD	Error code, see chapter G.4 Error code

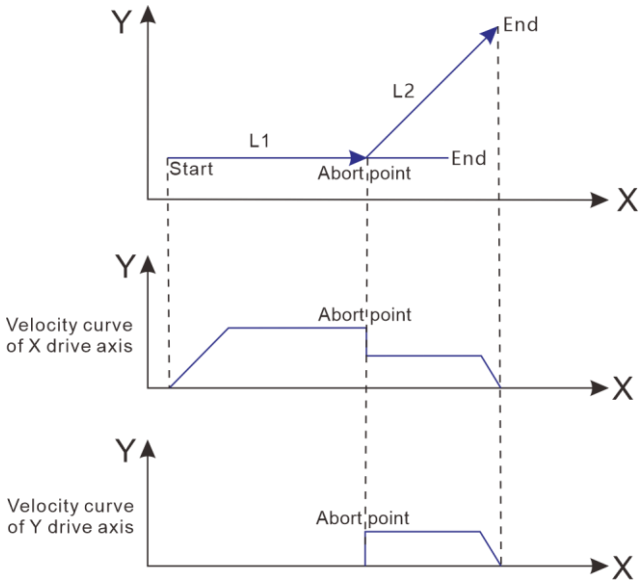


BufferModeDetailed introduction of continuous interpolation

Value	Description	Detailed
0	Abort	Interrupt the ongoing command directly and disable the transition
1	Buffered	The deceleration of the previous stage is completed, and the buffered command starts to be executed.
2	BendingPrevious	Go to the end of the previous segment through the previous segment and start the next segment at the rate of the previous segment, disable transitions
16#12	TMCorner	Additional angle transition, transition to the next segment with an additional angle when the previous segment starts to decelerate

Abort

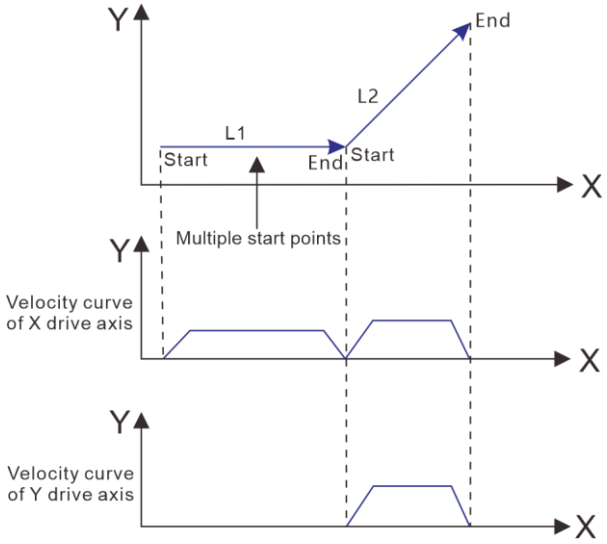
Line L1 executes the first interpolation instruction first, and triggers the second interpolation instruction before L1 finishes walking. If the buffer mode of the second interpolation instruction is set to "Abort", the second interpolation instruction is interrupted immediately The first interpolation command and start the execution of a new interpolation curve.



Buffered

Line L1 first executes the first interpolation instruction, and triggers the second interpolation instruction before L1 finishes walking. If the buffer mode of the second interpolation instruction is set to "Buffered", the interpolator will continue to execute the first interpolation instruction.

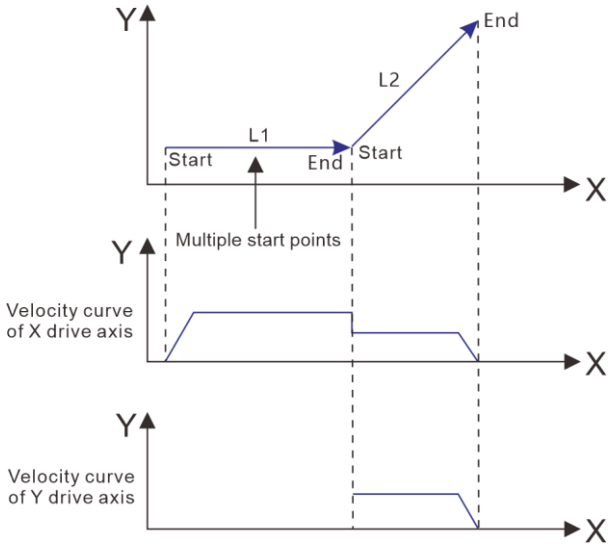
After the first interpolation instruction is executed and the Done signal output is valid, the second interpolation instruction starts to execute, as shown in the following figure:



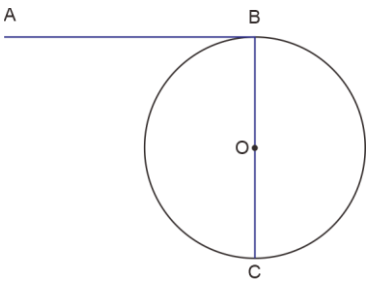
BendingPrevious

Line L1 executes the first interpolation instruction first, and triggers the second interpolation instruction before L1 finishes walking. If the buffer mode of the second interpolation instruction is set to "BendingPrevious", the interpolator will try to keep the target speed of the first instruction and walk a complete straight line.

After the execution of the first interpolation command is completed and the Done signal output is valid, the second interpolation command starts to be executed, the switching point will keep the speed unchanged, and the sub-speed of the coordinate axis will be redistributed. As shown below:

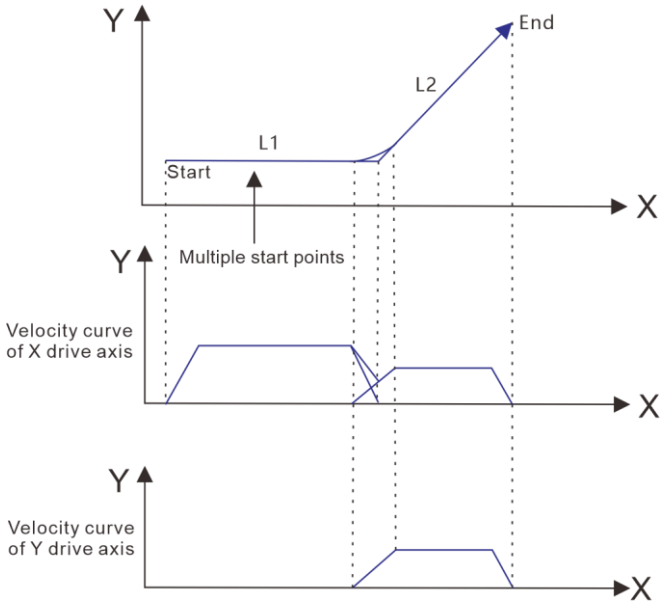


This mode is suitable for switching between a straight line and an arc and the straight line is located on the tangent of the arc. Using this mode can keep the speed of the interpolation curve continuous.



TMCorner

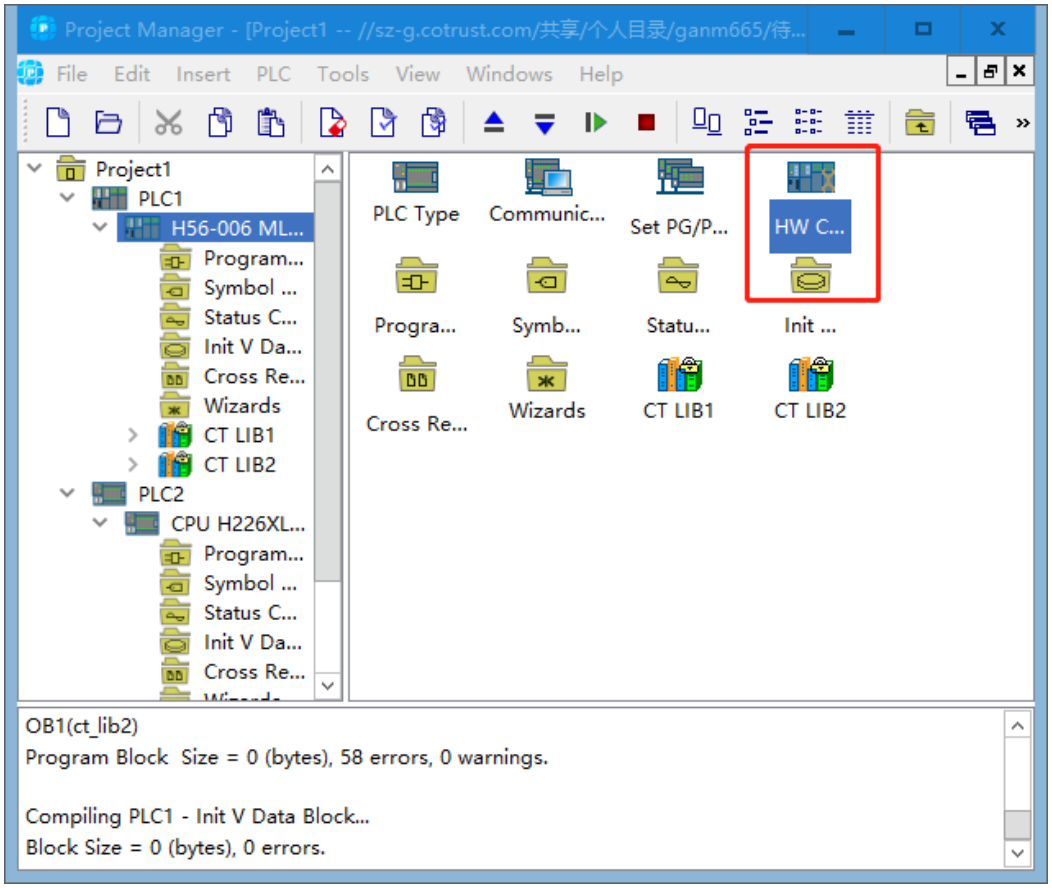
Line L1 first executes the first interpolation instruction, and triggers the second interpolation instruction before L1 finishes walking. If the buffer mode of the second interpolation command is set to "TMCorner", when the interpolator detects that the first straight line L1 starts to decelerate, it starts the second interpolation command and transitions to the second interpolation by means of an additional angle. Keep the speed continuous in all directions, as shown in the following figure:



Examples of continuous interpolation functions

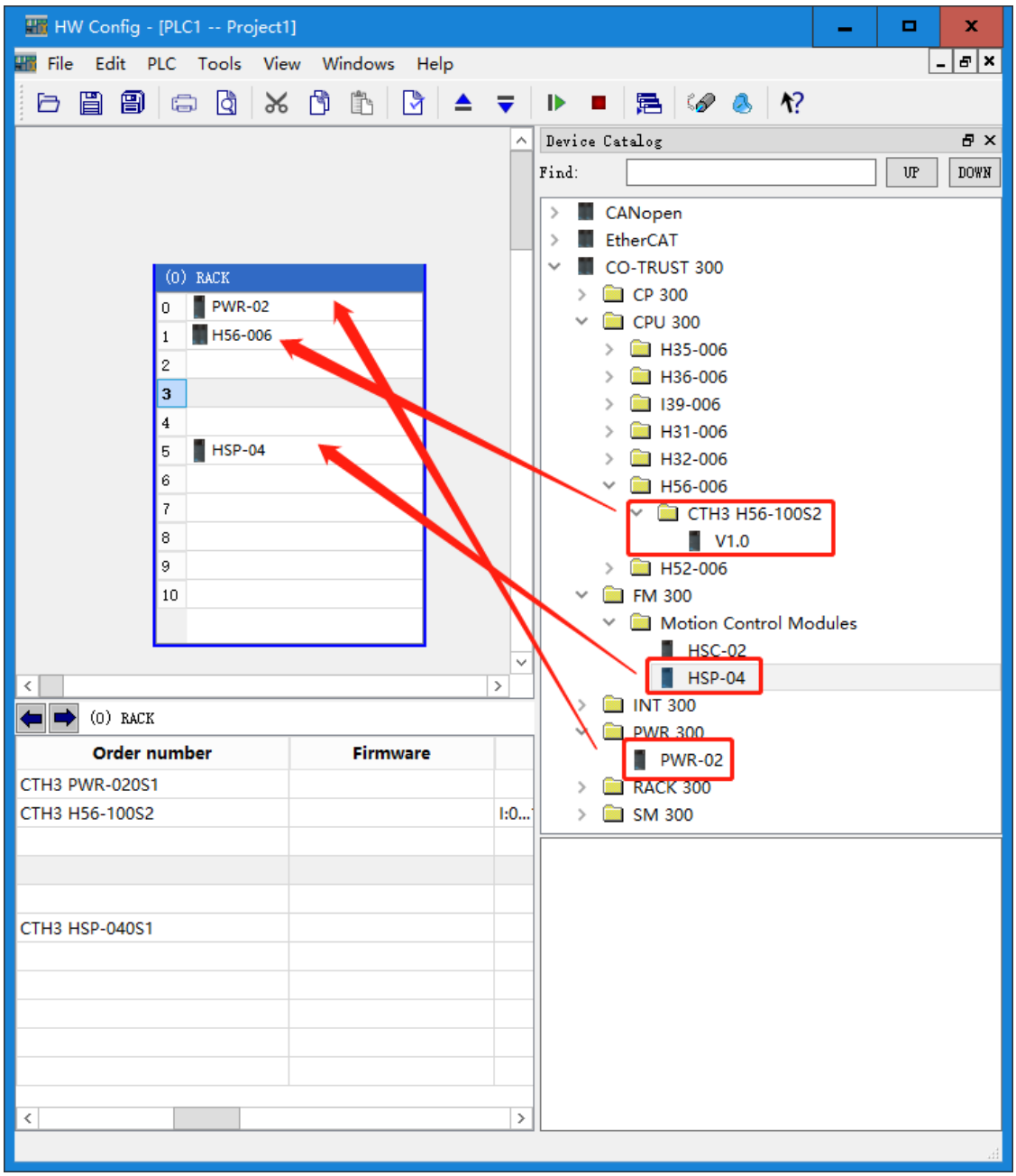
Take CPU H56-006 as an example to introduce continuous interpolation.

In the H56-006 CPU, select "HWare Config" to hardware configure, and then call the interpolation instructions in the main program for programming. The interpolation command called in this example is a relative displacement linear interpolation command.



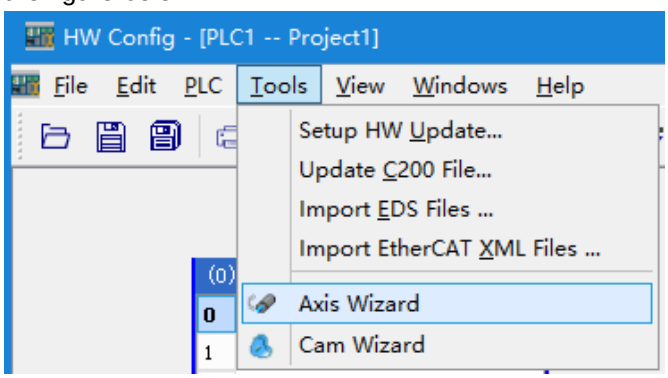
Step 1: Add the power supply module, CPU and HSP module to the rack

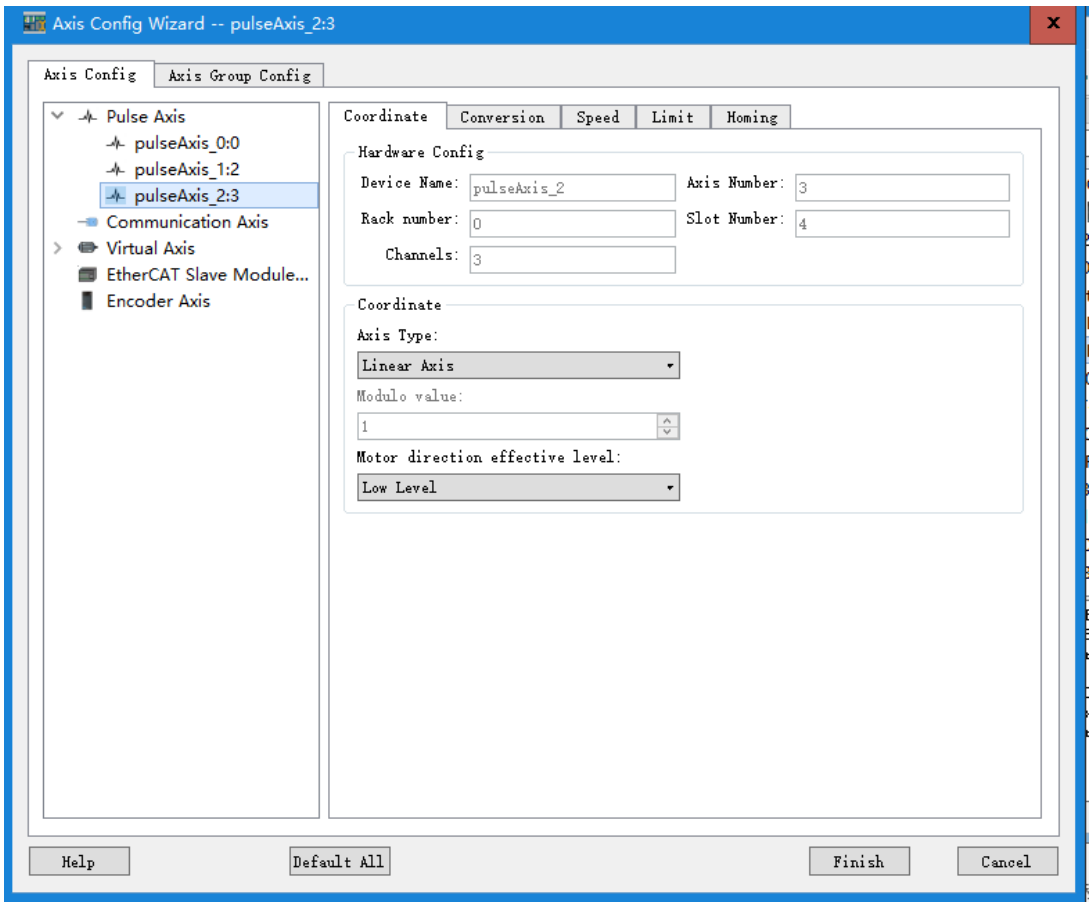
In the hardware configuration, under the "CO-TRUST 300" project tree, expand the "PWR 300", "CPU 300" and "FM 300" folders in order, and add the power supply module, CPU and HSP module. The power module can only be placed in slot 0 of the rack, the CPU can only be placed in slot 1 of the rack, and the HSP module can be placed in card slots 3-10, as shown in the figure below.



Step 2: Configure the axis

In the hardware configuration, select "Tools" → "Axis Wizard" to add two pulse axis, as shown in the figure below.

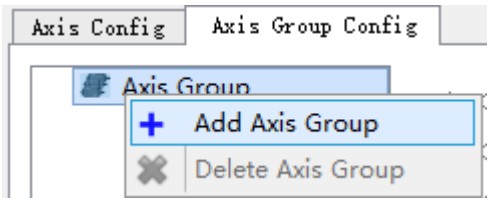




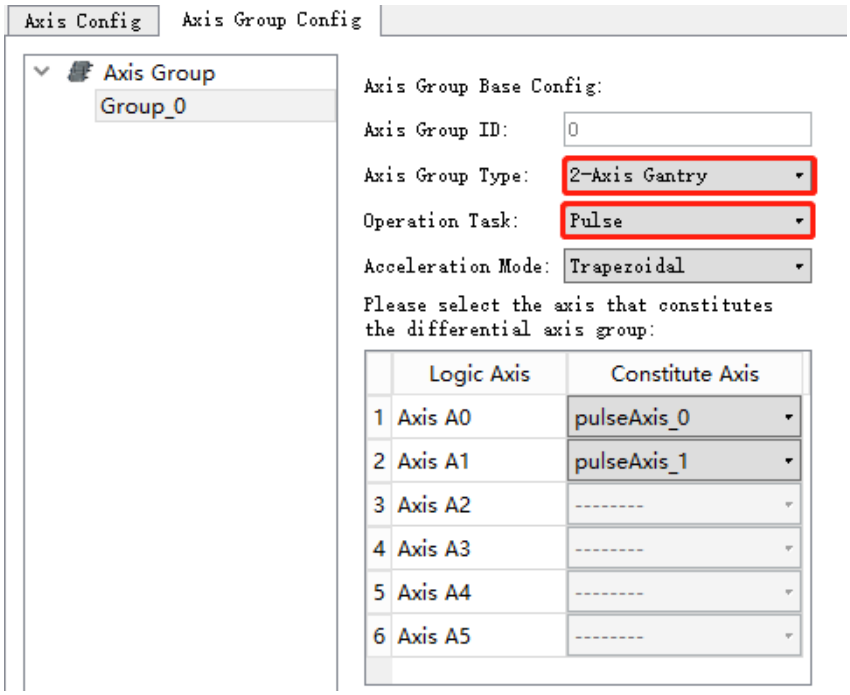
<Remarks> The interpolation function needs to configure two or more pulse axes or EtherCAT axes.

Step 3: Configure the axis group

Combine two or three of the configured axes in the form of axis groups. In this example, there are two axes configured. in the "Axis Group ", right-click "Add Axis Group" to add the axis group, as shown in the figure below Show:



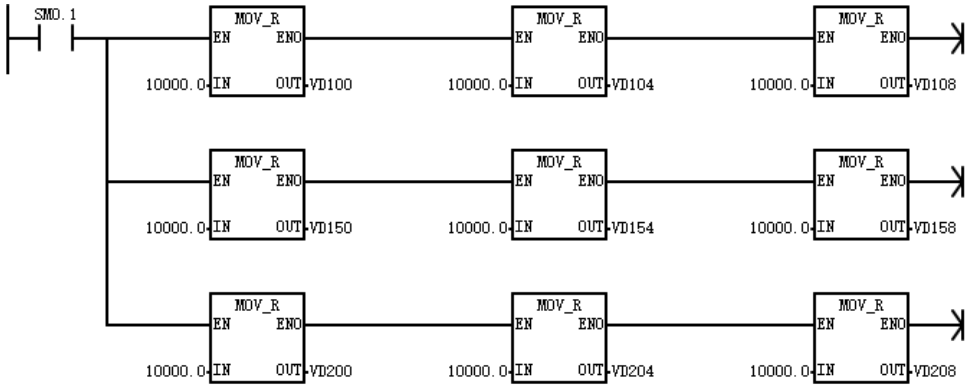
The added axis group is as follows:



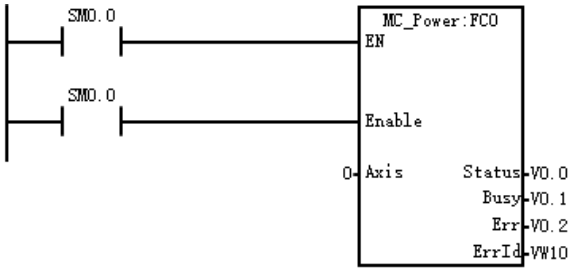
- "Axis Group Base Config ": "Axis Group ID" cannot be changed, Axis Group Type" can select "2-Axis Gantry" or "3-Axis Gantry", corresponding to two or three groups of "Constituent axes"; "operation cycle" "Optional "Pulse" or "EtherCAT". "Acceleration Mode" selectable "Trapezoidal", "sin*2" and "Secondary".
- "Logical Axis": default six groups, cannot be changed.
- "Constituent axis": "2-Axis Gantry" corresponds to two constituting axes, "3-axis Gantry" corresponds to three constituting axes, and "pulse axis" or "virtual axis" can be selected.

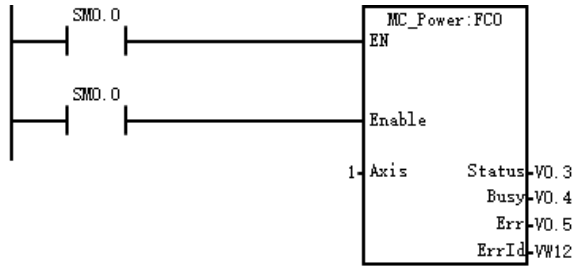
Step 4: Call the interpolation instruction in the main program for programming.

Network 1: Set the speed of interpolation

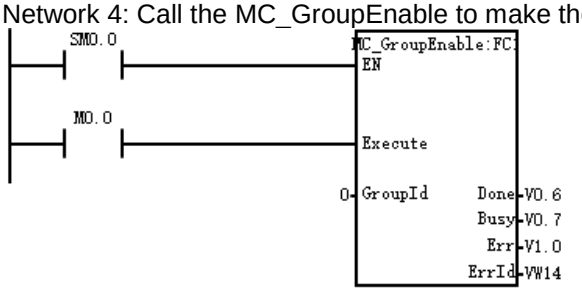


Network 2, Network 3: Enable the two pulse axes respectively.

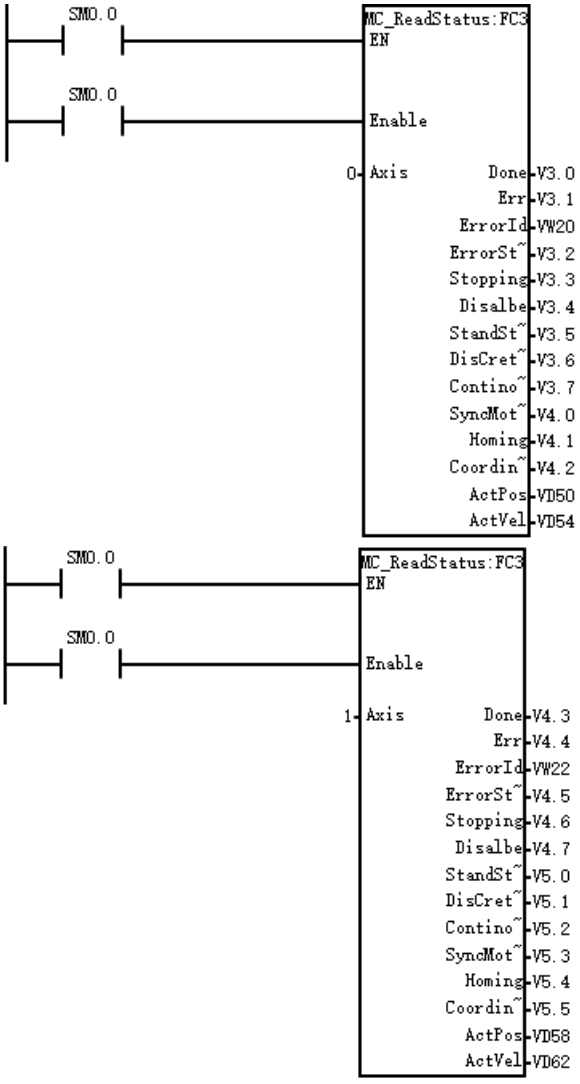




Network 4: Call the MC_GroupEnable to make the axis group effective

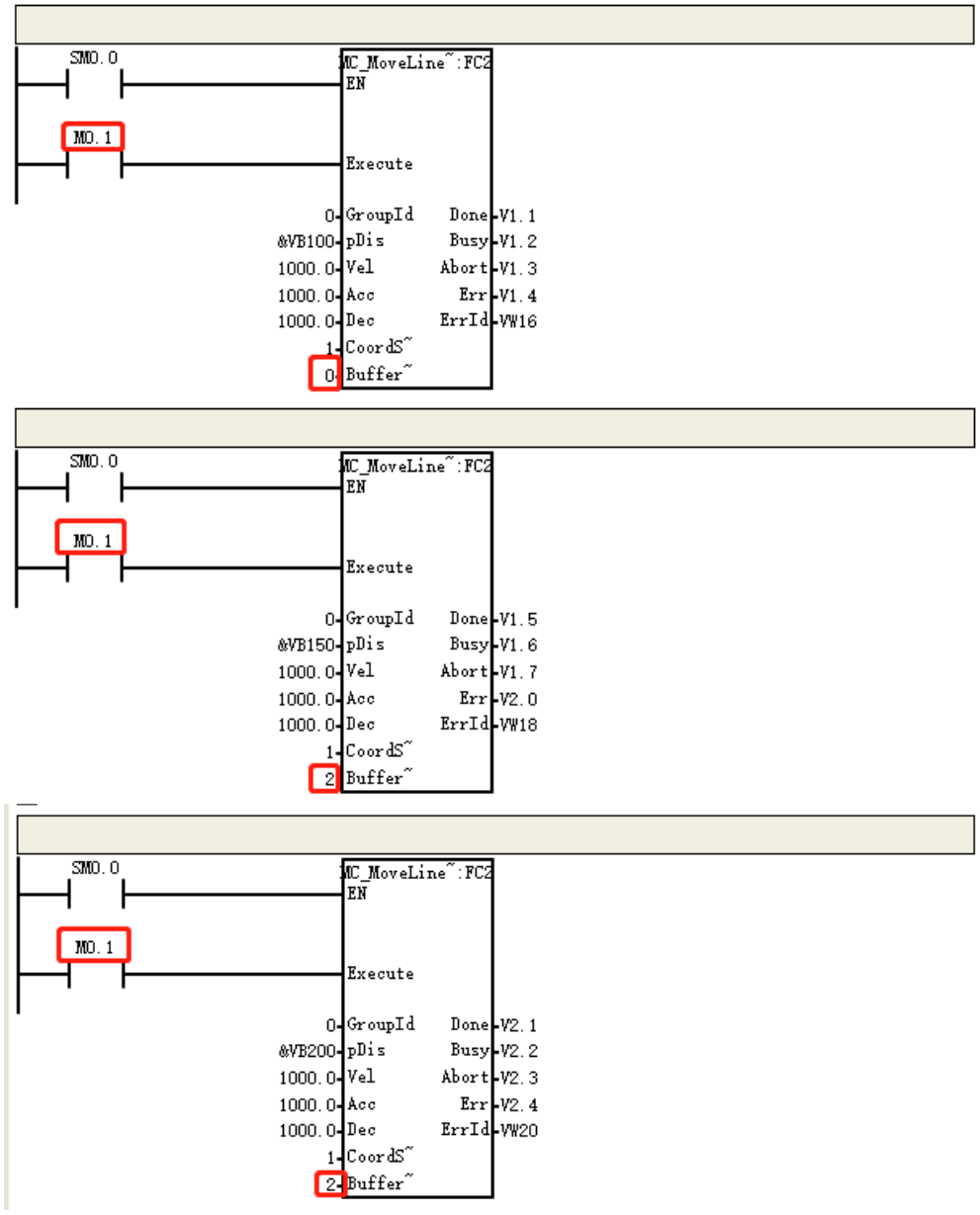


Network 5, Network 6: Read the corresponding axis status.



Network 7, 8, 9: set the running speed, acceleration, deceleration, buffer mode and other parameters. The instruction enable bit can be triggered by the same bit at the same time (two continuous interpolation instructions are cached in the program), and then carry out continuous linear interpolation movement in accordance with the order of instructions.

< Note >: The maximum number of consecutive interpolation cache instructions is 8. If you want to cache the ninth instruction, enable the cache the ninth instruction after the first instruction runs.



G.4 Error Code

Error code	Description	Error code	Description
0	No error	50	Communication error
1	Wrong axis number value	51	Servo internal error
2	Axis status error	52	There is no error to clear MC_Reset
3	Command is missing	53	Motion interruption enable
4	Axis Busy	60	Incorrect RetioNumerator value
5	The speed value is too small (less than the minimum speed)	61	Incorrect RetioMenominator value
6	The speed value is too large	65	The cam table and MC_TableSelect are

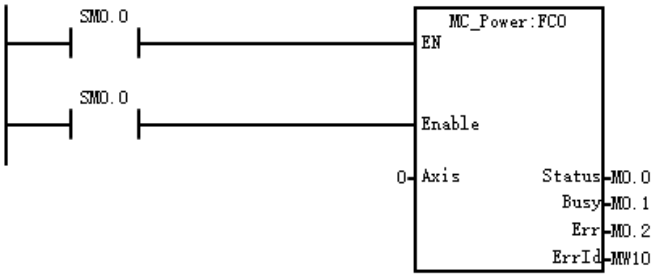
	(greater than the maximum speed)		inconsistent.
7	The acceleration value is too small	66	Cam table ID is wrong (or not created)
8	The acceleration value is too large	67	Cam table data error
9	Deceleration value is too small	68	MasterScaling value is incorrect
10	Deceleration value is too large	69	SlaveScaling value is incorrect
11	Wrong direction value	70	StartMode value is wrong
12	HSP axis value is wrong	71	Target torque (16#6071:0) is not mapped
13	HSP module number error	72	Operation mode (16#6060:0) is not mapped
15	Shaft not ready	73	The current torque (16#6077:0) is not mapped
16	Reset error failed	74	SLOP setting error
17	GroupId error	75	ProbeID setting error
19	Pointer error	76	Touch probe function (16#60B8:0) is not mapped
20	BufferMode value is wrong	77	Touch probe staus (16#60B9:0) is not mapped
21	Incorrect CoordSys value	78	Touch probe value(16#60BA:00 ~60BD) is no mapping
22	Internal software error	79	Window setting error
23	Memory application error	80	Left limit switch reached
24	CircMode 设置错误	81	Right limit switch reached
25	Three arcs into a line	82	Reach the negative software limit
26	The distance between two points and the center of the circle is not equal	83	Reach the positive software limit
27	interMode setting error	90	cmd value error
30	Back to the original mode lacks Z-phase configuration	91	TriggerInput value is wrong
31	Back-to-origin mode lacks forward switch configuration	92	TriggerVar value is incorrect
32	Back to original mode lacks negative switch configuration	93	MC_Feed execution failed
33	Return to the original mode lacks the origin switch configuration	94	MC_Feed instruction is being executed
34	Wrong value in return mode	95	There is no such interrupt in the module, or the interrupt is not enabled
35	Servo internal return error	96	Set torque error

43	Position parameter setting is too large	100	Axis error in the axis group
44	Position parameter setting is too small	101	Axis group status setting error
45	Spindle coordinate out of range	102	An axis in the axis group is already under the control of another axis group.
46	Running overspeed (the running speed exceeds the maximum speed)	110	Exceed the maximum number of buffer instructions
47	Incorrect lead setting	1000	This configuration will only be supported in the future
48	End point is not on the curve	2001~ 2022	Module error
49	The center coordinate is set incorrectly	3000	There is an error in the configuration

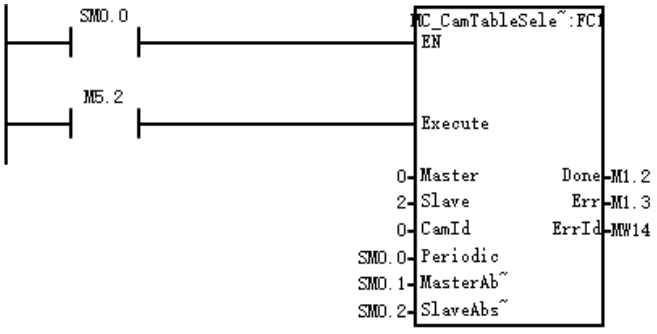
G.5 Electronic Cam Program Example

The following program is an example of CTH300-H Ethercat CPU calling axis instruction ct_plcopen_lib (v2.3) to realize the electronic cam function.

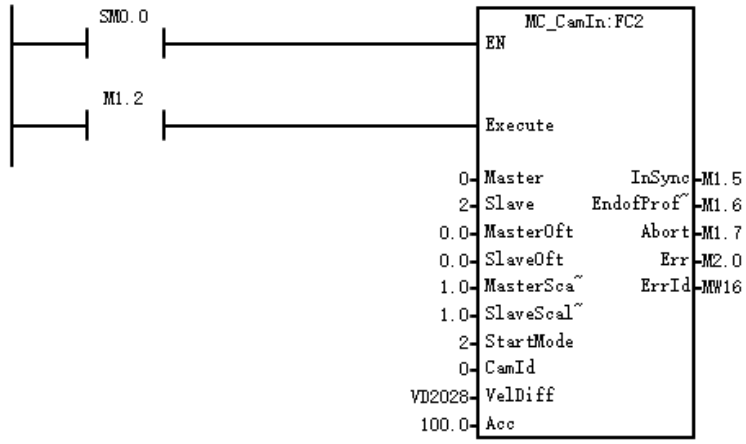
Network 1: Enable the axis.



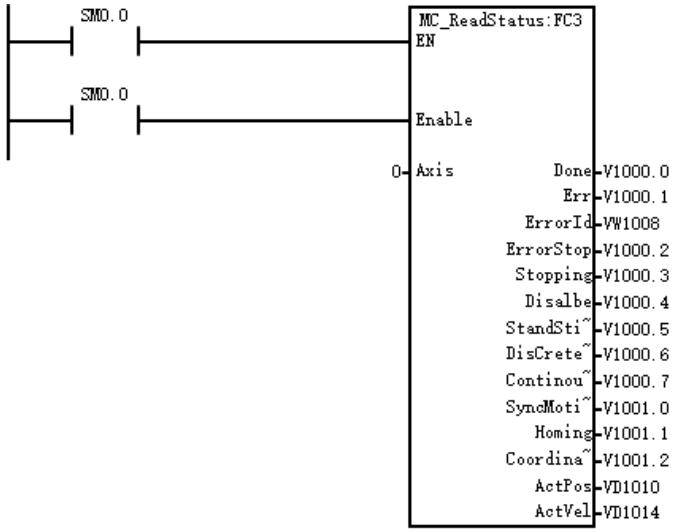
Network 2: Select the cam curve, set the master, slave information, status information and other parameters.



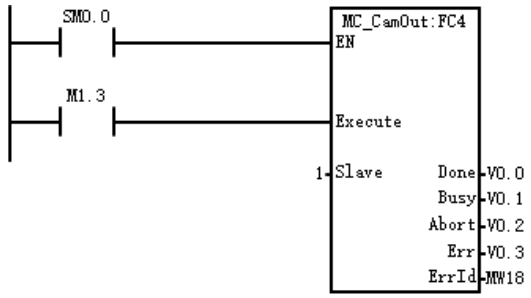
Network 3: Establish the cam relationship between the master and slave axes. When establishing the cam relationship, specify the offset value, zoom ratio and start mode of the master and slave axes according to application requirements. When the instruction is executed, the slave axis follows the main axis movement according to the cam curve.



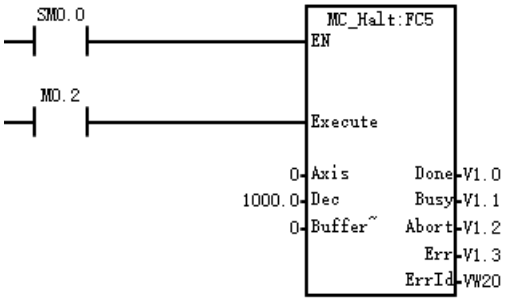
Network 4: Read the status of the corresponding axis.



Network 5: Release the cam relationship between the master and slave axis. After the relationship is released, the slave axis will continue to move at the speed before the separation.



Network 6: The axis decelerates at the given deceleration rate until it stops.



G6 Introduction to the Homing Function

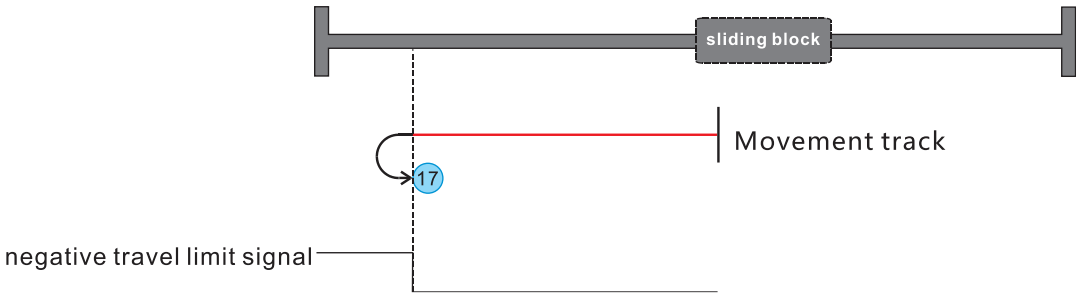
Homing Mode Principle

It can be selected from the following 15 Homing modes according to their own accuracy requirements and practical application requirements.

Mode	Description
17	Reference Negative Limit Position Falling Edge
18	Reference Positive Limit Position Falling Edge
19	Reference Forward Home Signal Switch
20	Reference Forward Home Signal Switch
21	Reference Negative Home Switch
22	Reference Negative Home Switch
23	Reference Home Signal Switch and Positive Travel Limit
24	Reference Home Signal Switch and Positive Travel Limit
25	Reference Home Signal Switch and Positive Travel Limit
26	Reference Home Signal Switch and Positive Travel Limit
27	Reference Home Signal Switch And Negative Travel Limit
28	Reference Home Signal Switch And Negative Travel Limit
29	Reference Home Signal Switch And Negative Travel Limit
30	Reference Home Signal Switch And Negative Travel Limit
35	Set the current position as the origin

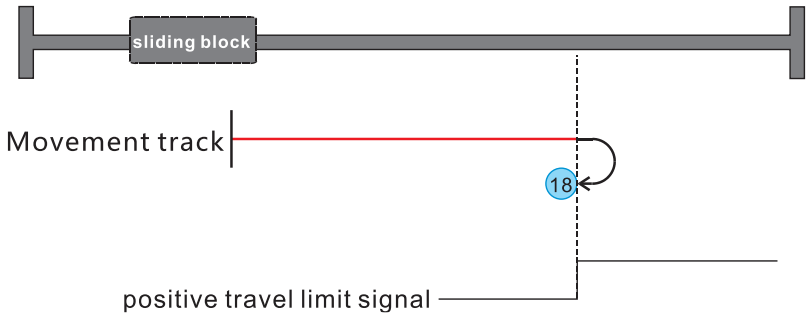
Regardless of the initial position of the machine, when the equipment (origin switch, positive travel limit switch, negative travel limit switch) is installed well, the origin of the equipment sought by the servo is always unique The vertical line "|" in the schematic diagram of the following modes represents the initial position of the machine, the circle "X" represents the origin position, "——" represents the search speed and "——" represents the crawling speed.

Home mode 17: Reference negative limit Swicth



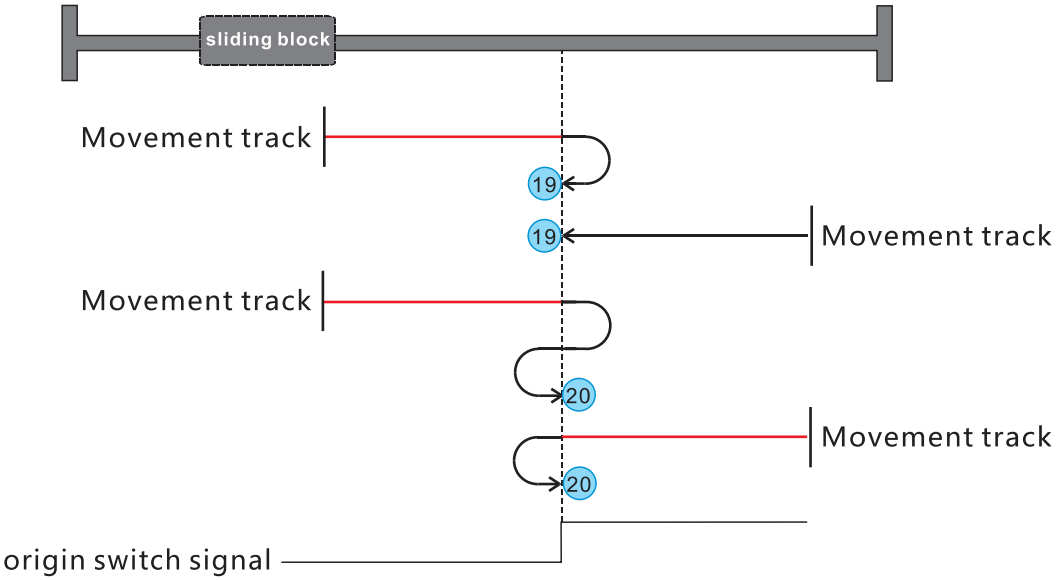
At the beginning of homing, the negative travel limit signal is 0, and the homing is carried out at high speed in the reverse direction. After encountering the rising edge of the travel limit signal, it will decelerate and reverse, and run at low speed until it encounters the falling edge of the travel limit signal. This position is the origin position.

Home mode 18: Reference positive limit position falling edge



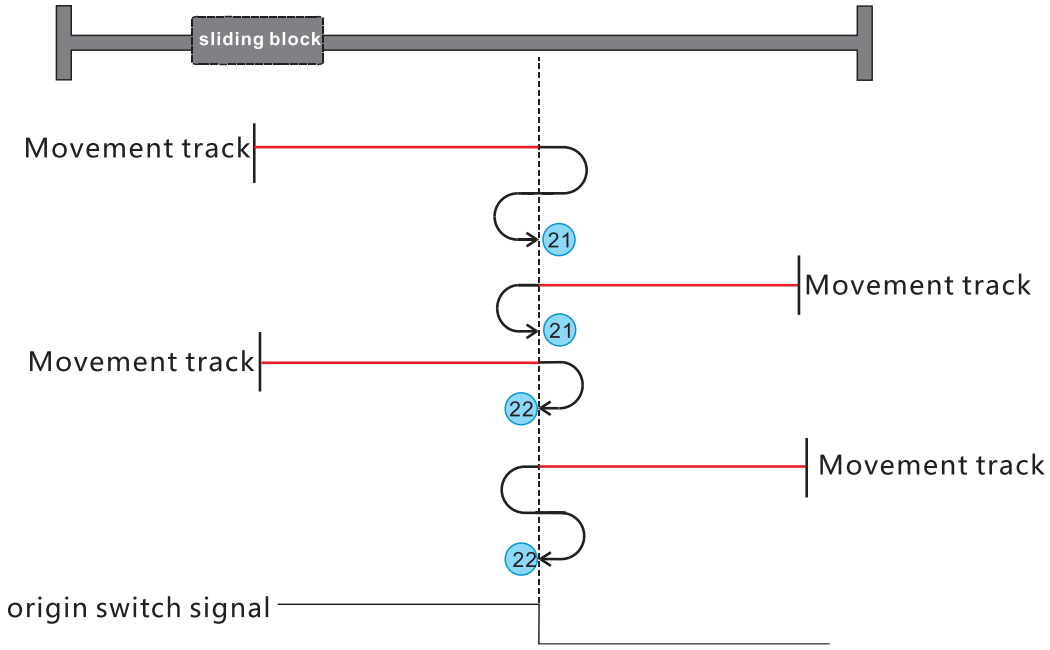
At the beginning of homing, the forward travel limit signal is 0, and the homing is carried out at high speed in the forward direction. After encountering the rising edge of the travel limit signal, it will decelerate and reverse, and run at low speed until it encounters the falling edge of the travel limit signal. This position is the origin position.

Home mode 19,20: Reference Forward Home Signal Switch



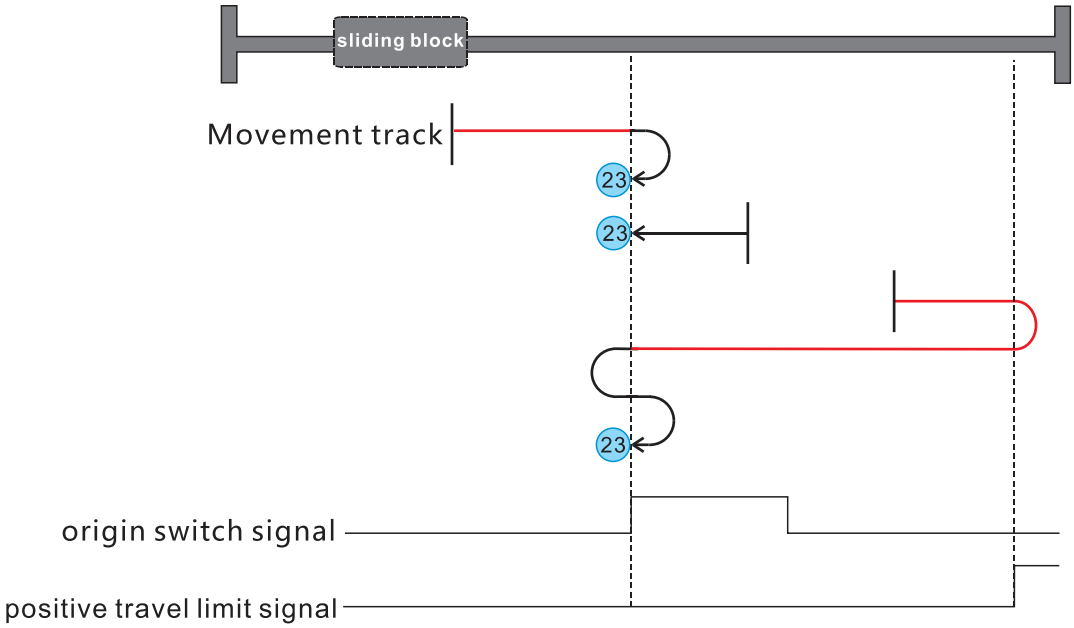
The starting movement direction depends on the high and low state of the origin switch signal. The origin of mode 19 is at the falling edge position of the origin switch signal from high to low. The origin of mode 20 is at the rising edge of the origin switch signal from low to high.

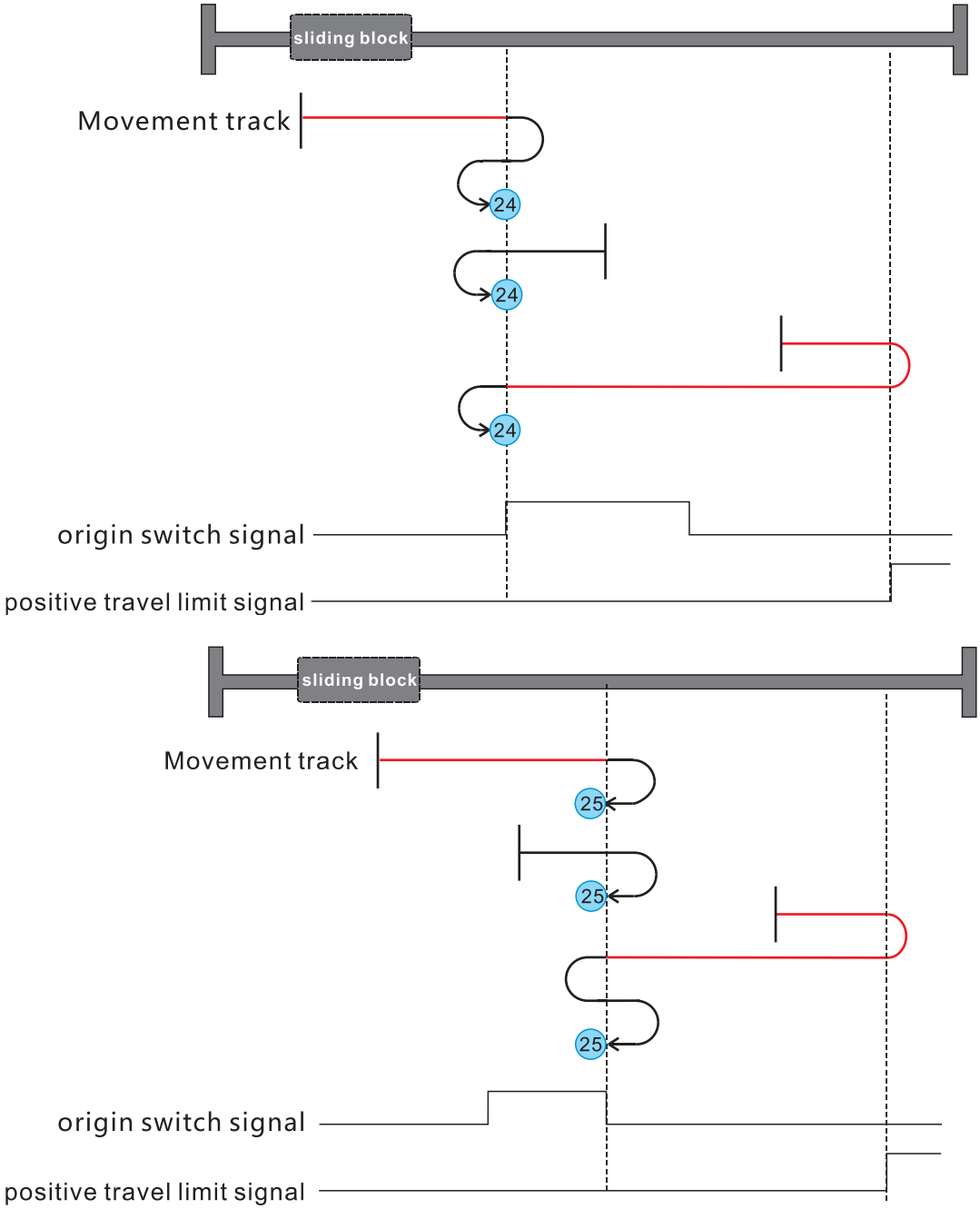
Home mode 21,22: Reference Negative Home Signal Switch

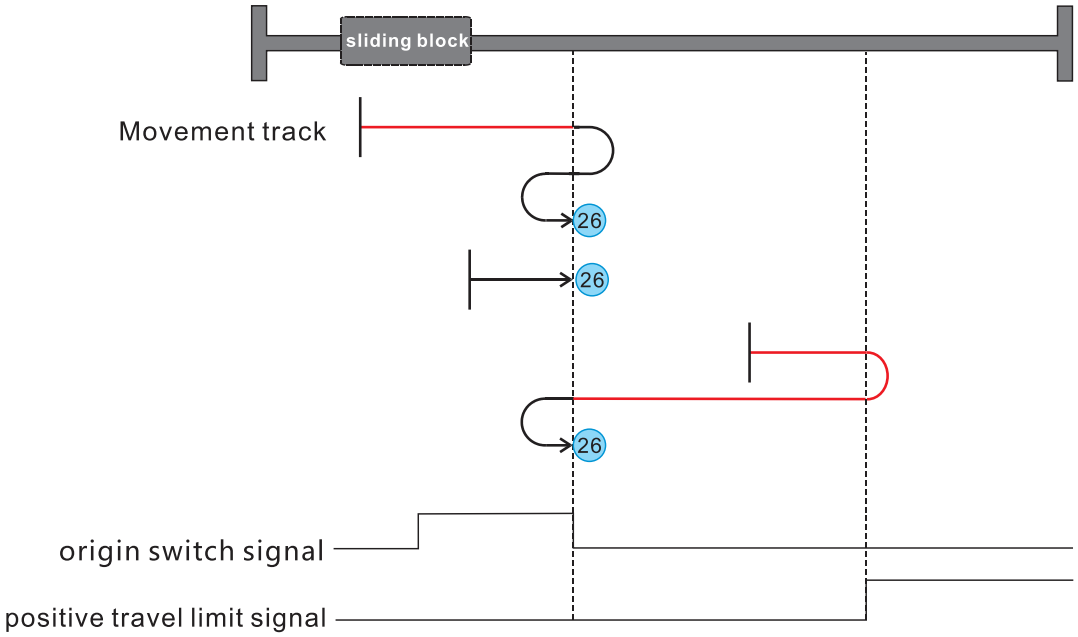


The starting movement direction depends on the high and low state of the origin switch signal. The origin of mode 21 is at the falling edge position of the origin switch signal from high to low. The origin of mode 22 is at the rising edge of the origin switch signal from low to high.

Home mode 23~26: Reference Home Signal Switch And Positive Travel Limit

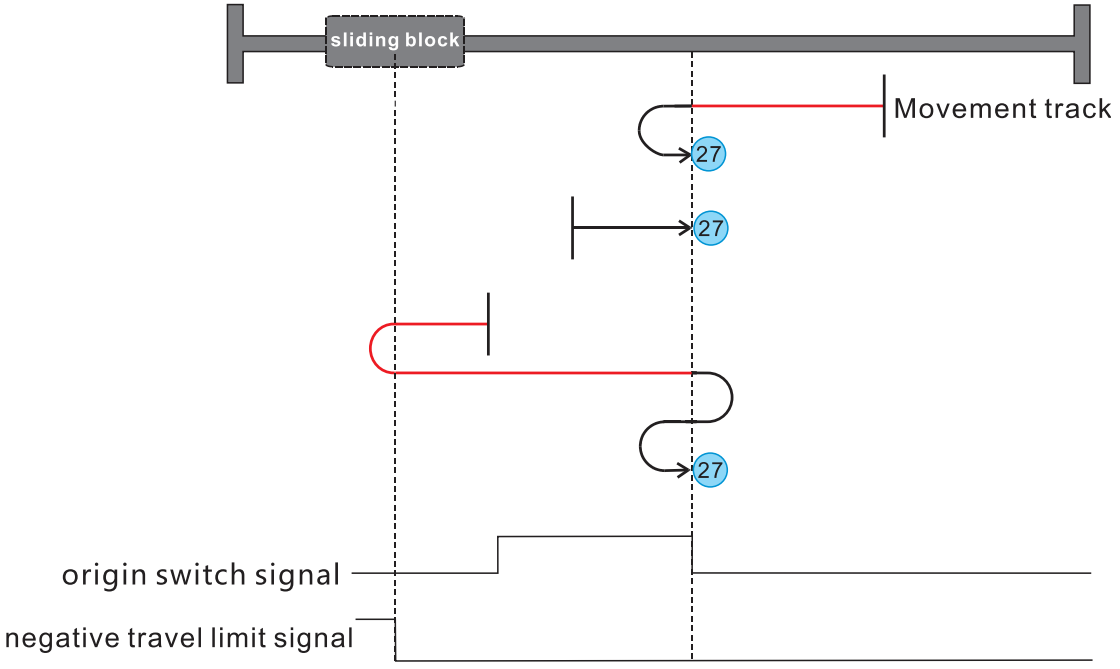


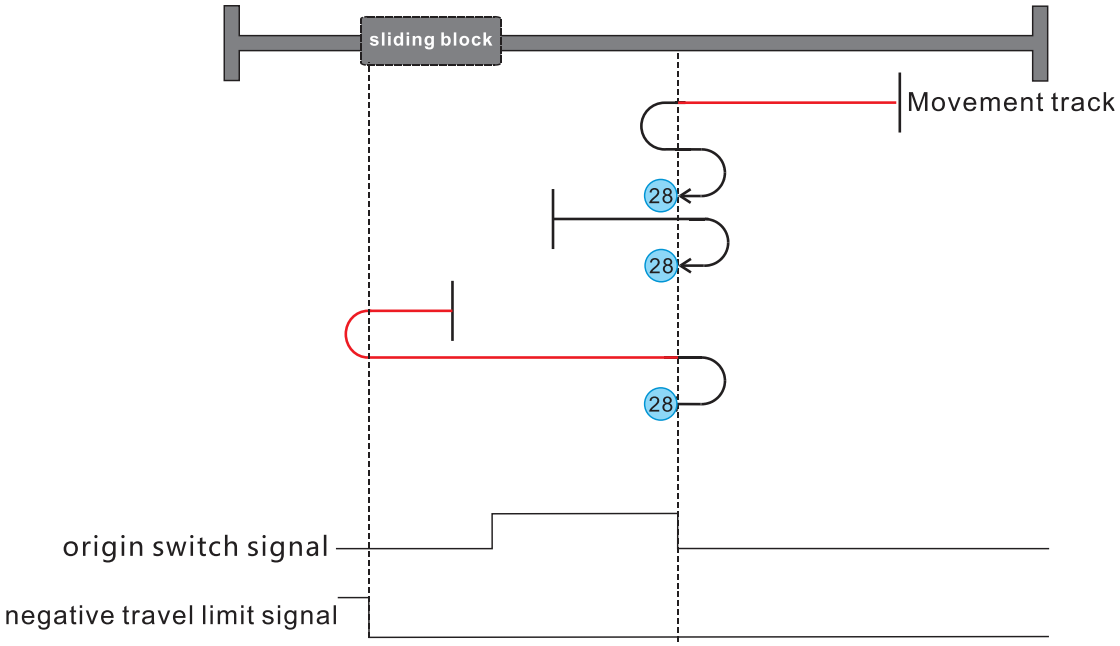

sliding block
Movement track
24
24
24
origin switch signal
positive travel limit signal
sliding block
Movement track
25
25
25
origin switch signal
positive travel limit signal

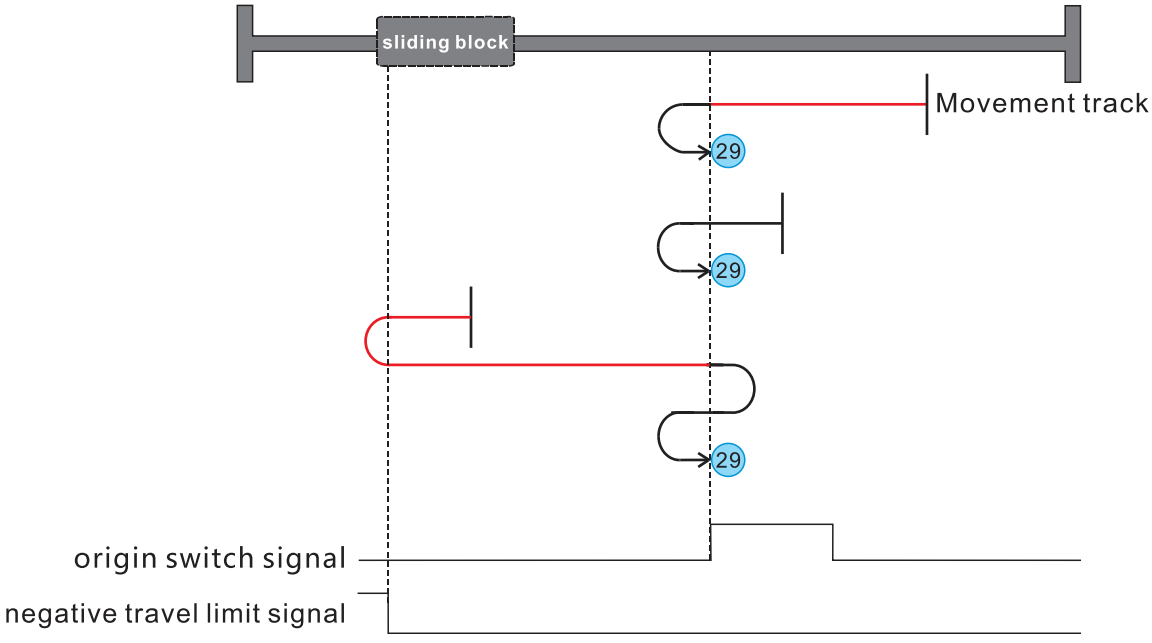


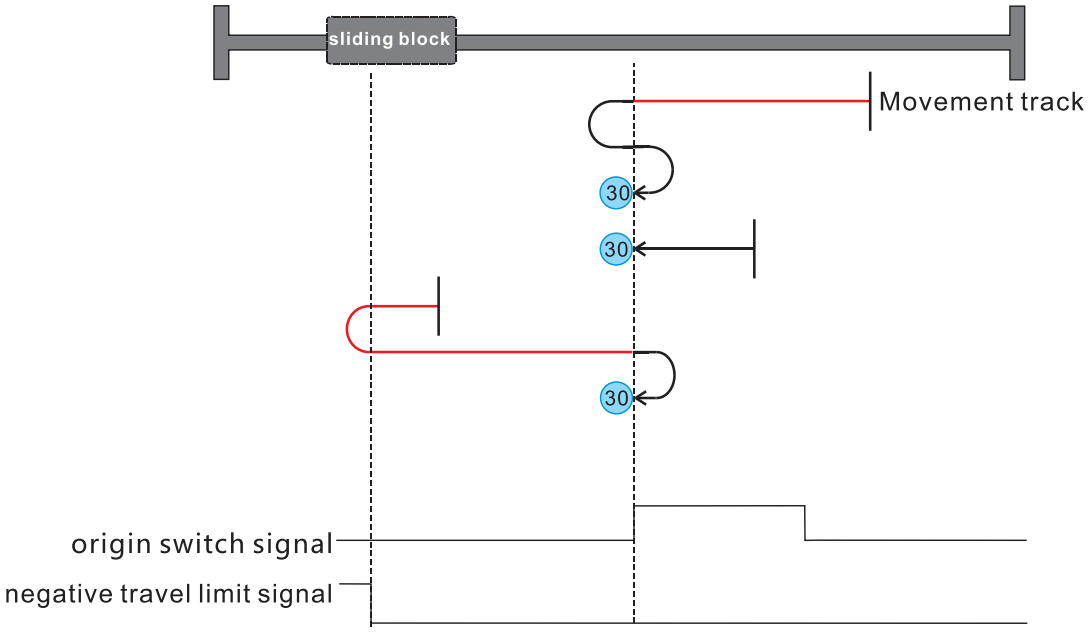
The starting movement direction depends on the state of the origin switch signal and the forward travel limit signal. The origin of mode 23 and 26 is at the falling edge position of the origin switch signal from high to low. The origin of mode 24 and 25 is at the rising edge position of the origin switch signal from low to high.

Home mode 27~30: Reference Home Signal Switch And Negative Travel Limit



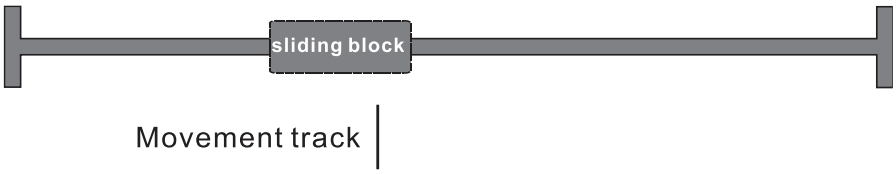

sliding block
Movement track
28
28
28
origin switch signal
negative travel limit signal


sliding block
Movement track
29
29
29
origin switch signal
negative travel limit signal



The starting movement direction depends on the state of the origin switch signal and the negative travel limit signal. The origin of mode 27 and 30 is at the falling edge position of the origin switch signal from high to low. The origin of mode 28 and 29 is at the rising edge position of the origin switch signal from low to high.

Home mode 35: Take The Current Position As The Home



The motor stops at the current position, and the original signal is given to complete the original signal.

Application Example of the Homing

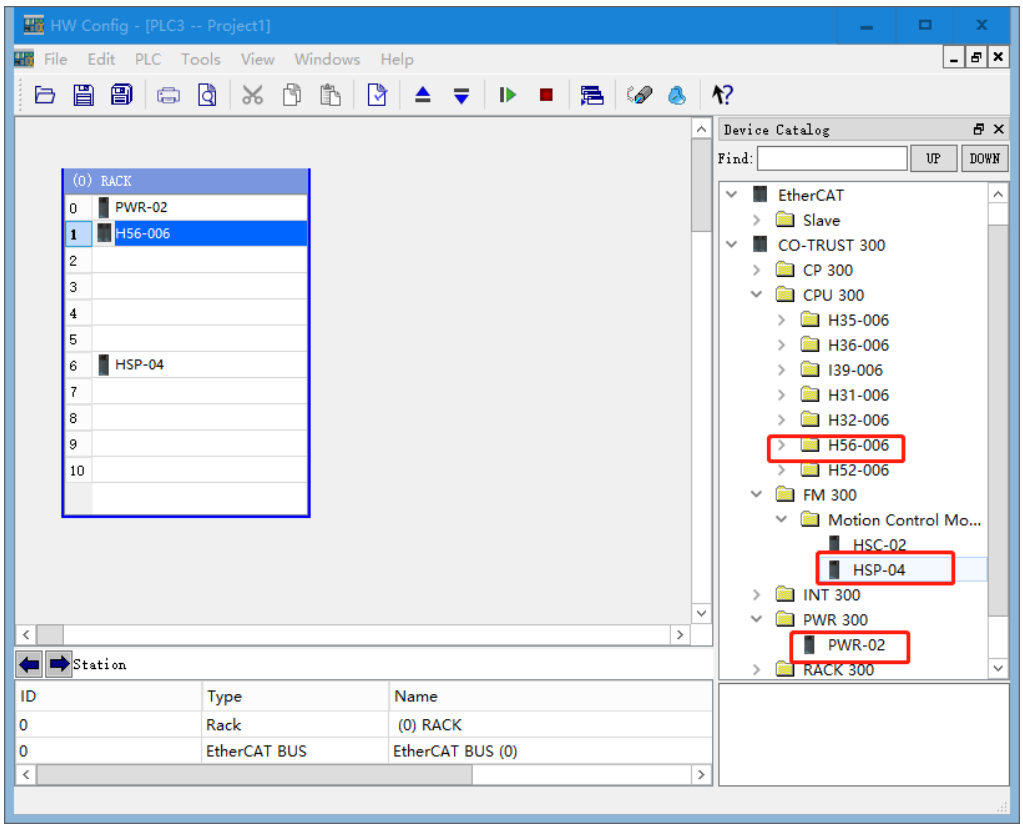
Take H56-006 CPU as an example to introduce the homing in detail.

Example 1: Pulse axis return to origin

Select "Hardware Configuration" in the project manager interface of the H56-006 CPU

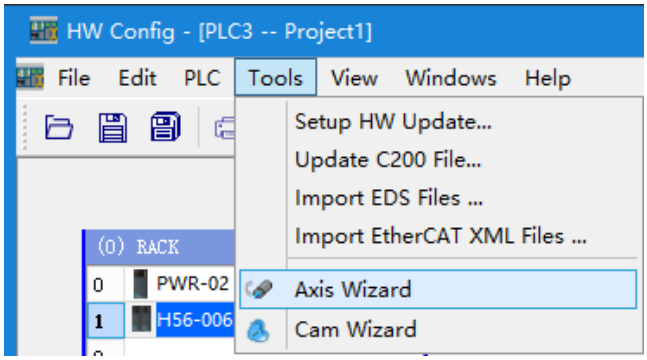
Step1: Add Power Module, CPU and HSP Module

In the hardware configuration, add the power supply module, CPU and HSP module, the power supply module can only be placed in the 0th slot of the rack, the CPU can only be placed in the 1st slot of the rack, and the HSP module can be placed in the 3~10th slot. As shown in the figure below.

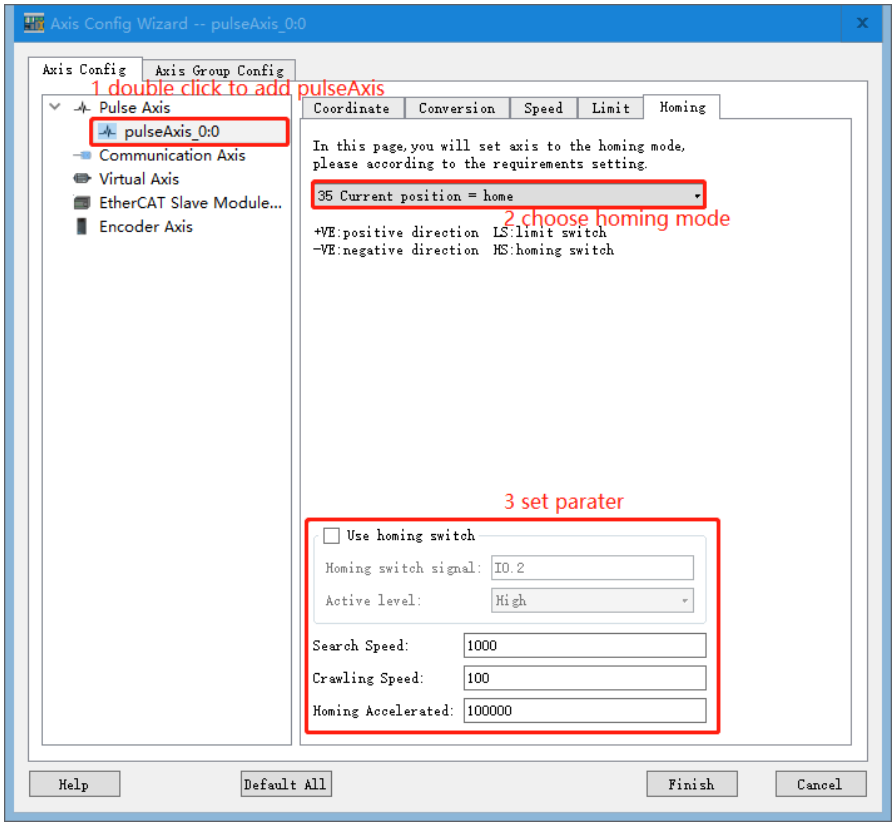


Step2: Configure the axis

In the hardware configuration interface, select "Tools"→"Axis Configuration Wizard", add a pulse axis, and then configure the axis.

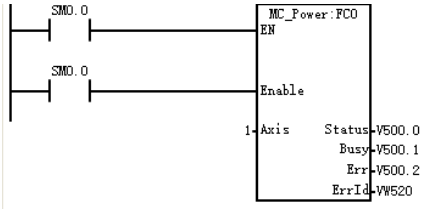


Click "Homing", on this page, you can configure the Homing mode of the axis. This configuration is only supported by the pulse Axis. There are 15 return modes to choose.

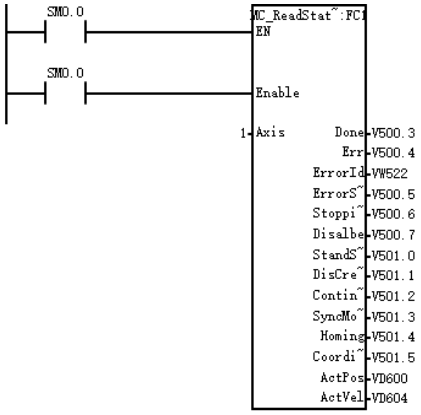


Step3: Call the MC_Home instruction (origin return instruction) in the main program for programming.

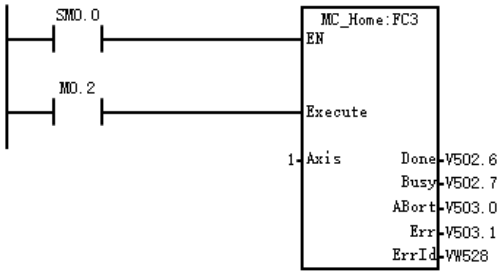
Network1: Enable the axis, and set the axis ID number Axis to 1.



Network2: Read axis status



Network3: The homing mode is 19, the homing signal is I0.2, and the search speed of the axis is 1000.

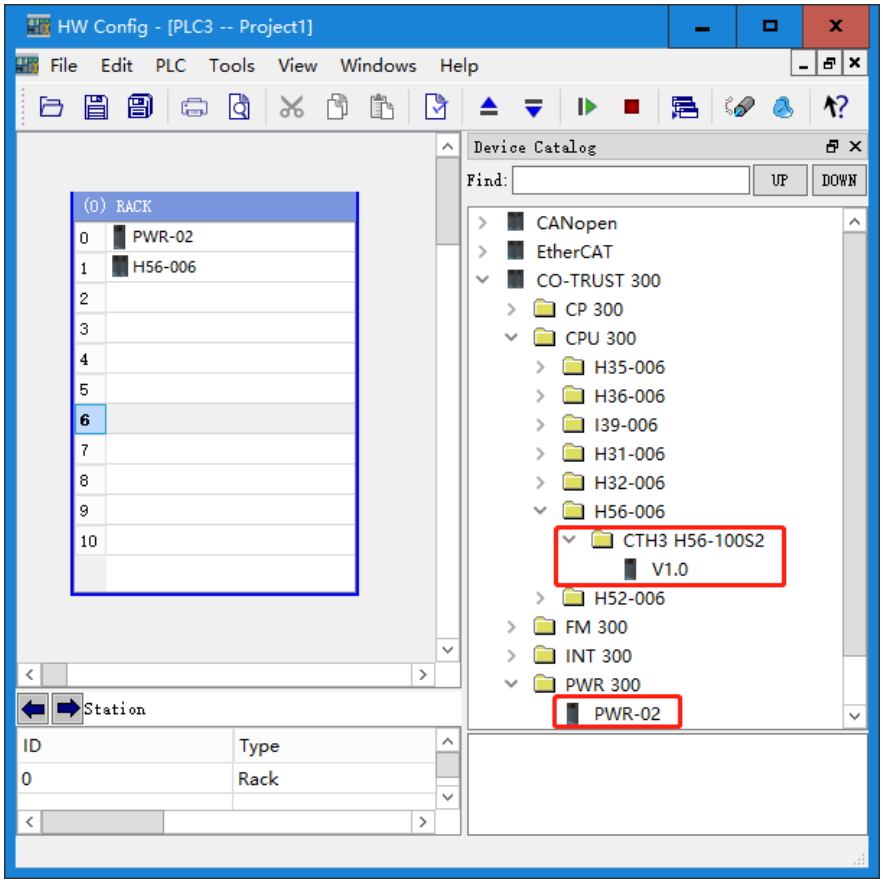


Example 2: Communication Axis return to origin

Select "Hardware Configuration" in the project manager interface of the H56-006 CPU

Step1: Add Power Module and CPU

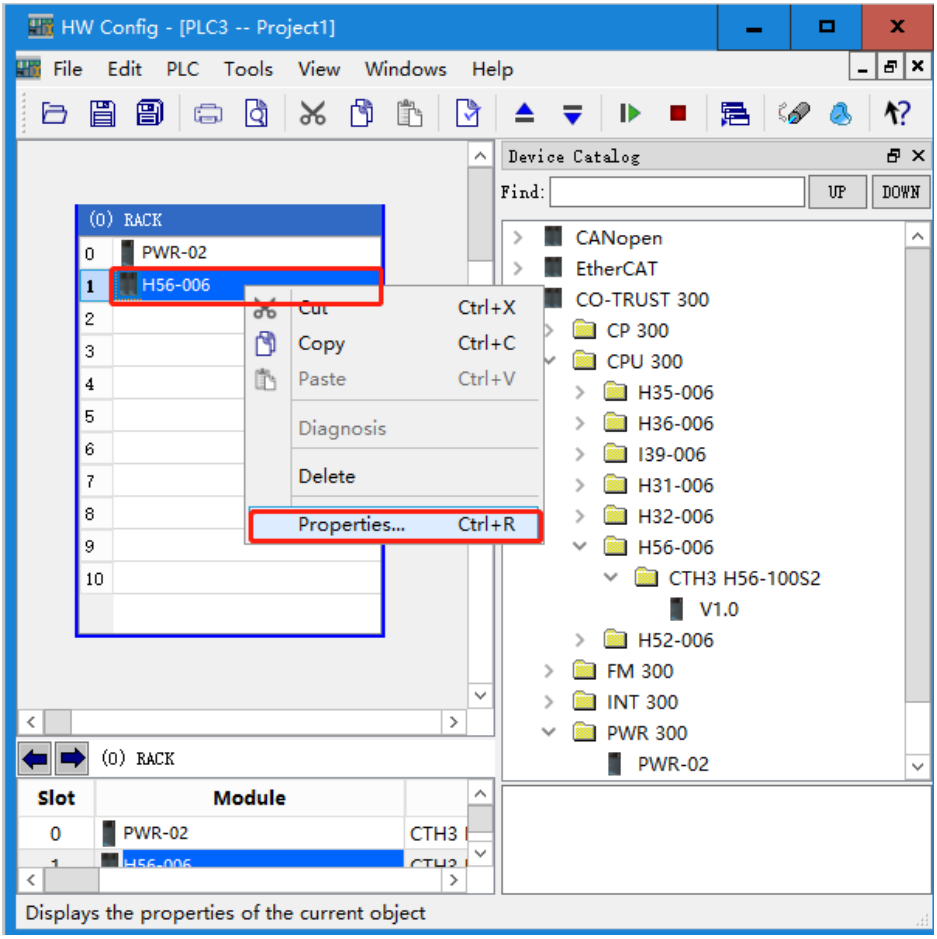
In the hardware configuration, add the power supply module and CPU, the power supply module can only be placed in the 0th slot of the rack, the CPU can only be placed in the 1st slot of the rack.

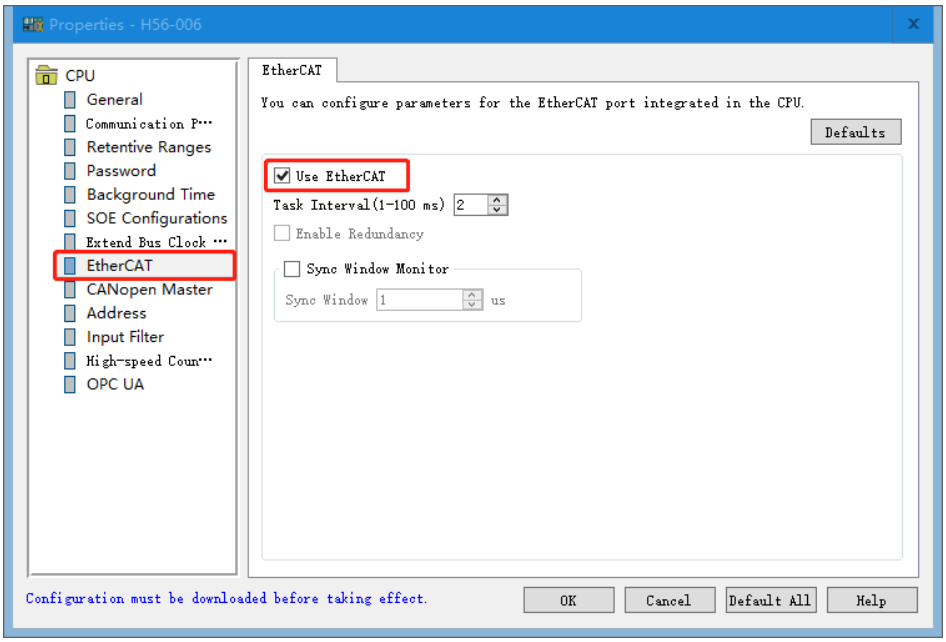


Step 2 : Configure the axis

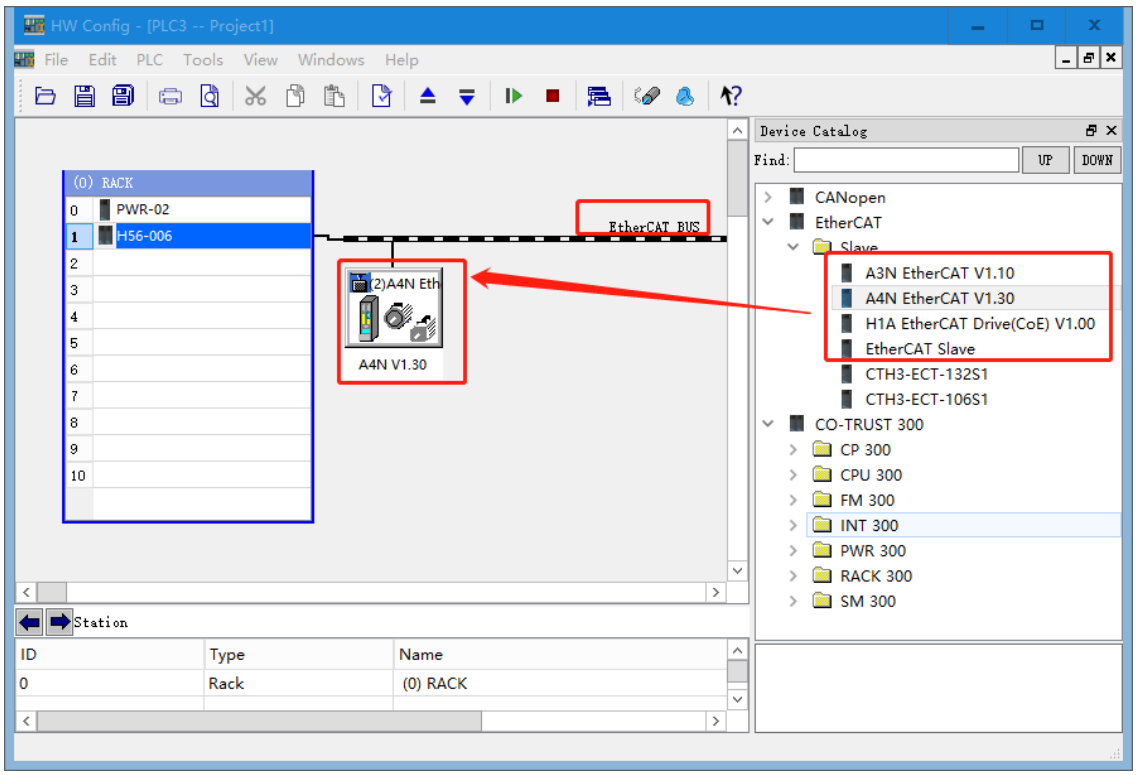
adding a communication axis, you need to add the corresponding EtherCAT slave in the hardware configuration interface, otherwise the communication axis cannot be added. The operations are as follows :

1) Right-click the H56 on the rack, select "Properties" " EtherCAT ", and click "Use EtherCAT".

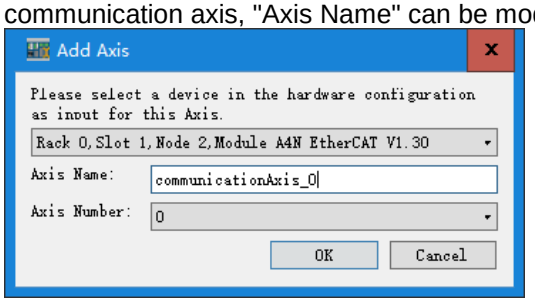




2) After enable EtherCAT Master, the EtherCAT BUS will appear in the configuration interface, and you can connect the EtherCAT slave to the EtherCAT BUS.

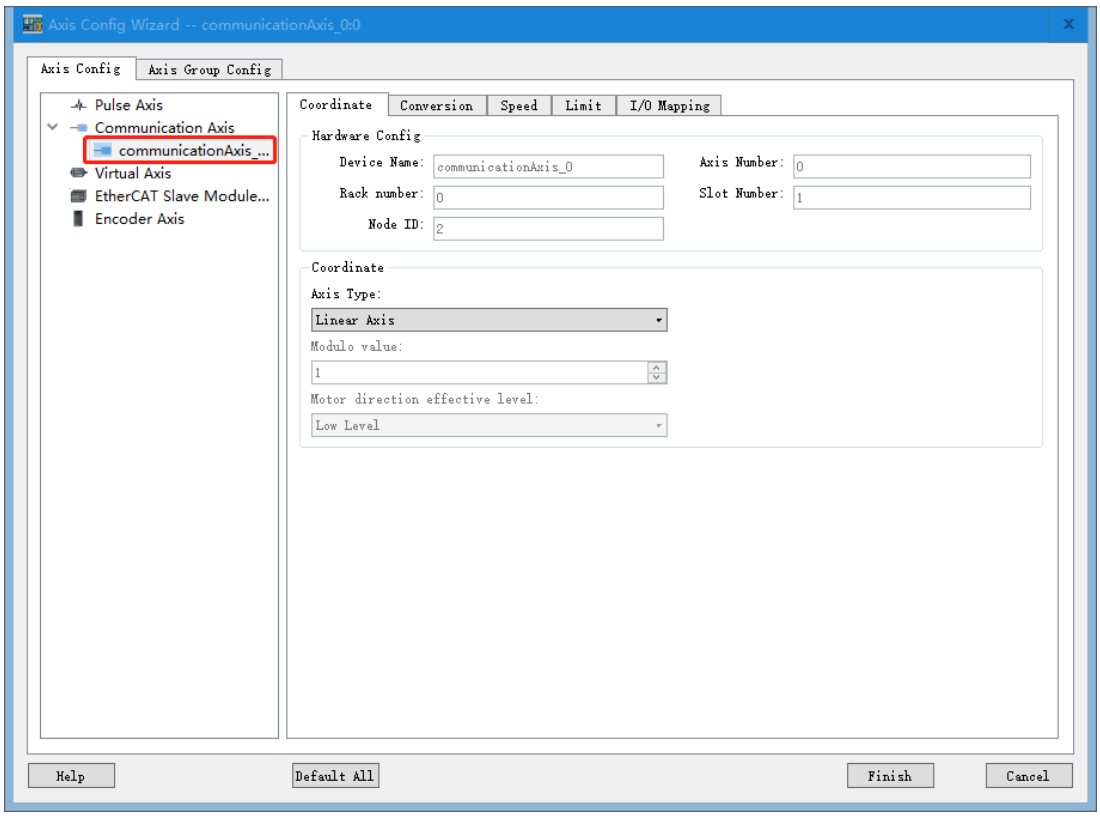


3) After completing the above configuration, select "Tools" "Axis Wizard Configuration", double-click "Communication Axis" in the axis wizard configuration interface to add a communication axis, "Axis Name" can be modified, click "OK" to complete the addition.



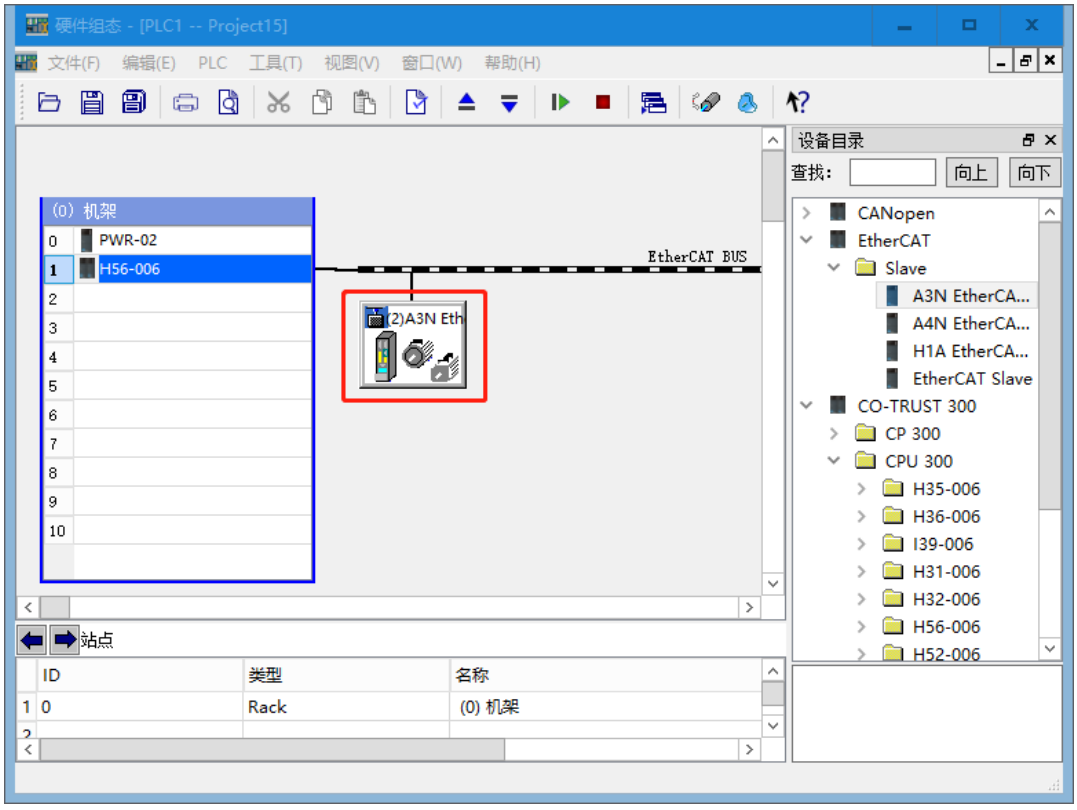
Note: Only one communication axis can be added to a slave.

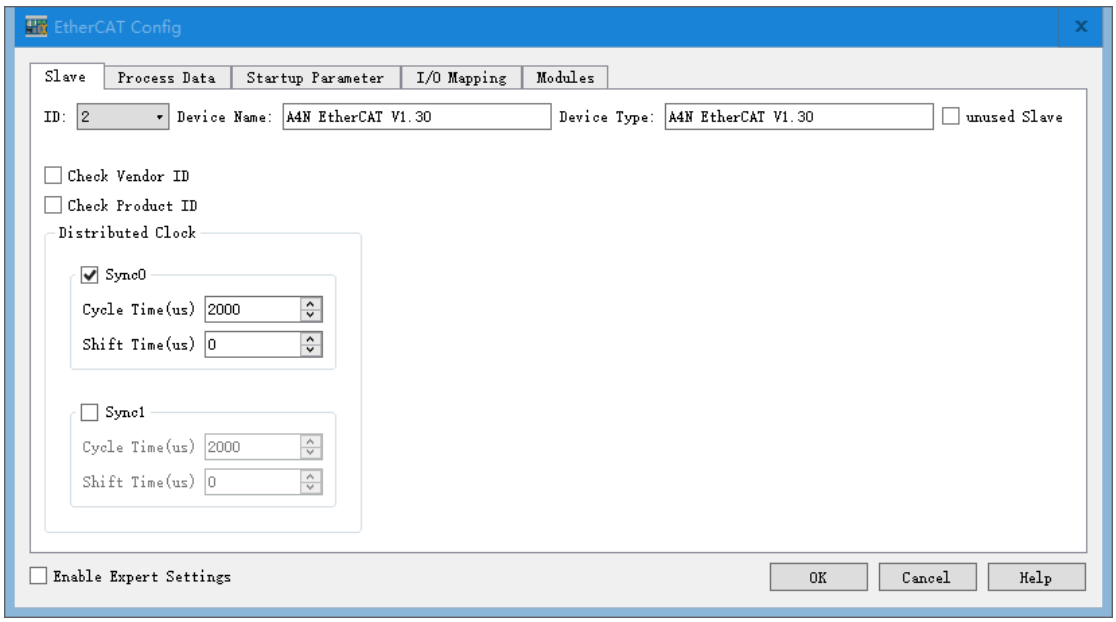
4) The interface of the added communication axis is as follows:



5) To realize the homing function, when configuring the homing function of the communication axis, it is necessary to set 16#6098:0 (homing mode), 16#6099:1 (homing search speed), 16#6099:2 (homing crawl speed) value.

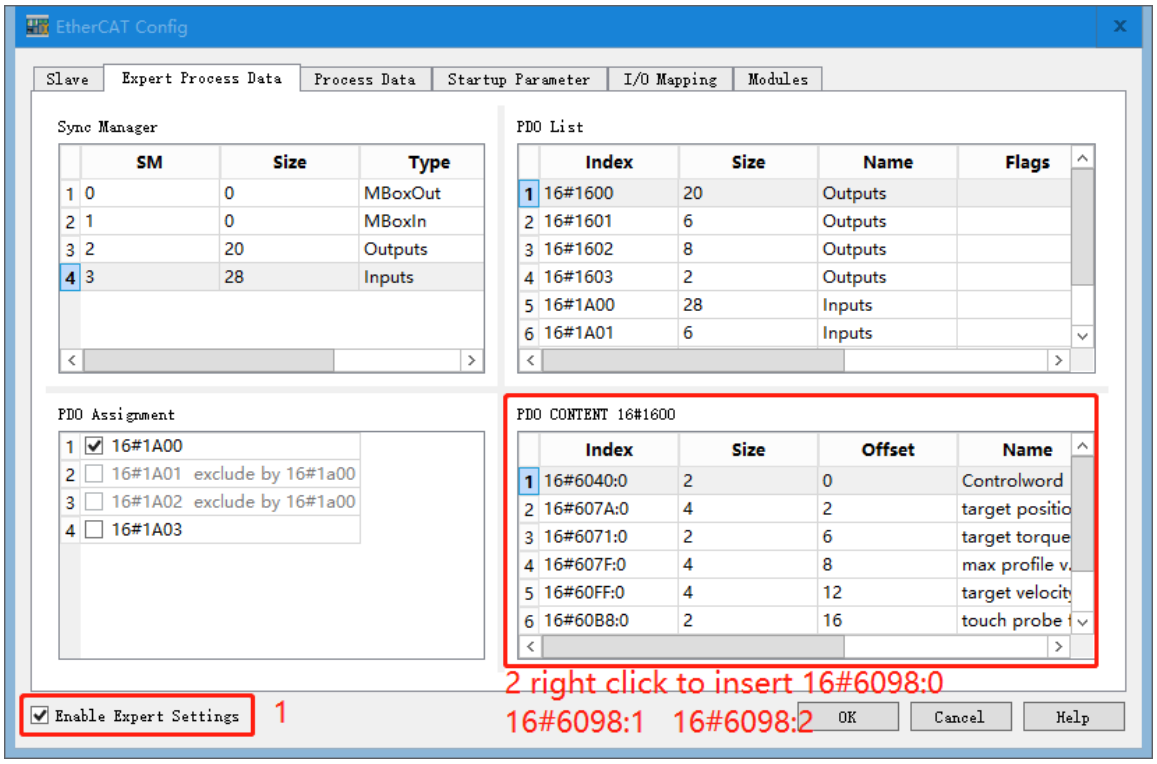
First, double-click the slave connected under "EtherCAT BUS", and the EtherCAT configuration interface will appear.





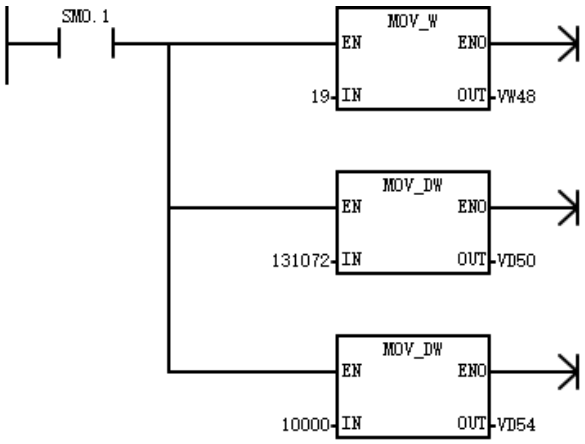
6) Click "Enable Expert Settings", right click and insert 16#6098:0 (return to original mode), 16#6099:1 (return to original search speed), 16#6099:2 (return to original crawl speed).

<Remarks> There are two units for the search speed and the creeping speed, respectively: pps (number of pulses per second) and rpm (number of revolutions per minute. Please refer to the driver manufacturer's manual for details.

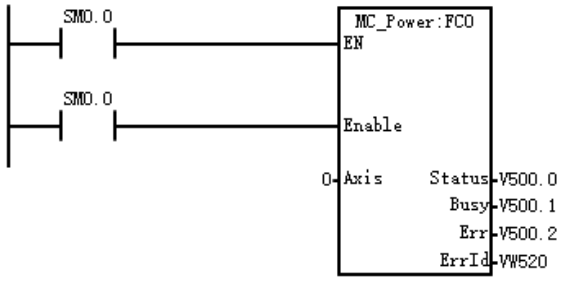


Step3: Call the MC_Home instruction (origin return instruction) in the main program for programming.

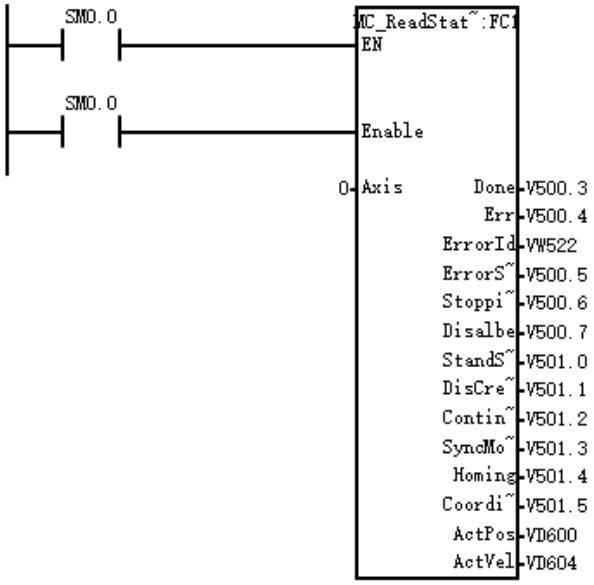
Network 1: The Homing mode is 19, the search speed is 131072, and the crawl speed is 1000.



Network 2: Enable the axis.

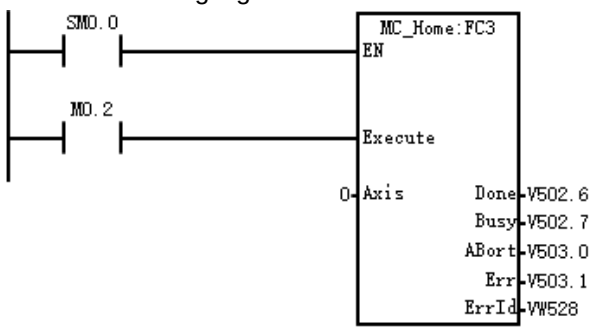


Network 3: read axis status



The homing mode is 19, the homing signal is I0.2, the axis runs in the positive direction of the search speed, when the origin signal rises, the axis runs in the negative direction at the crawling speed, and when the origin signal falls, the axis stops and the position is cleared to 0, Servo parameter 16#6064:0 (position actual value) is cleared to 0.

Note: The homing signal is on the external DI terminal of the servo.

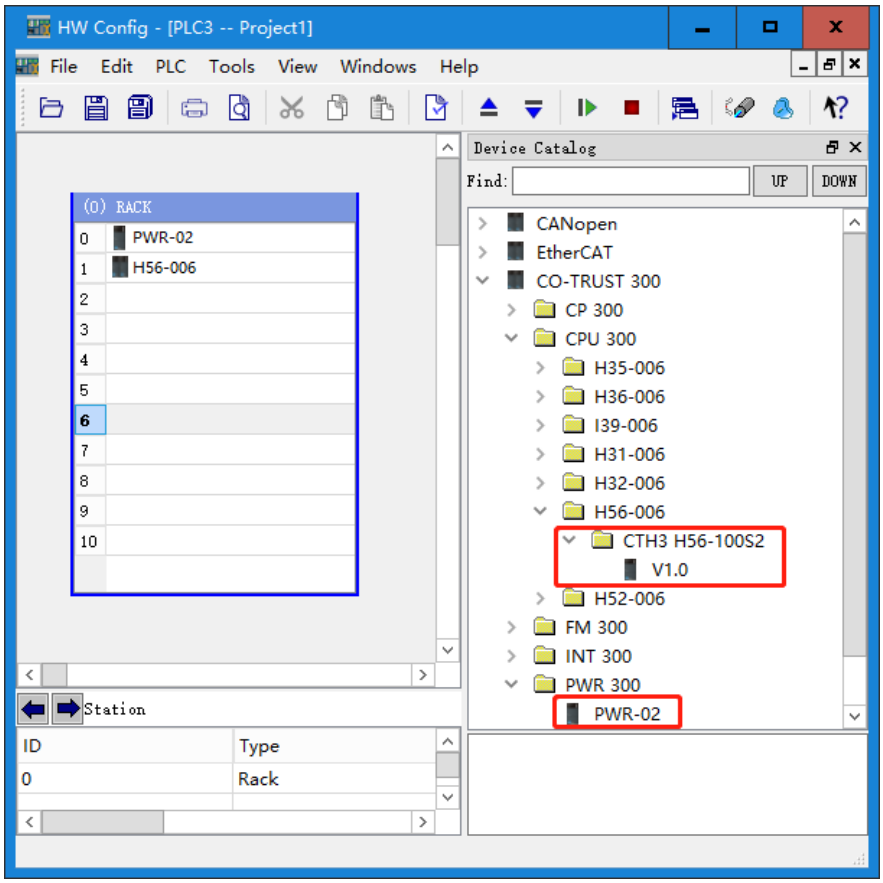


Example 3: Communication Axis return to special origin

Select "Hardware Configuration" in the project manager interface of the H56-006 CPU

Step1: Add Power Module and CPU

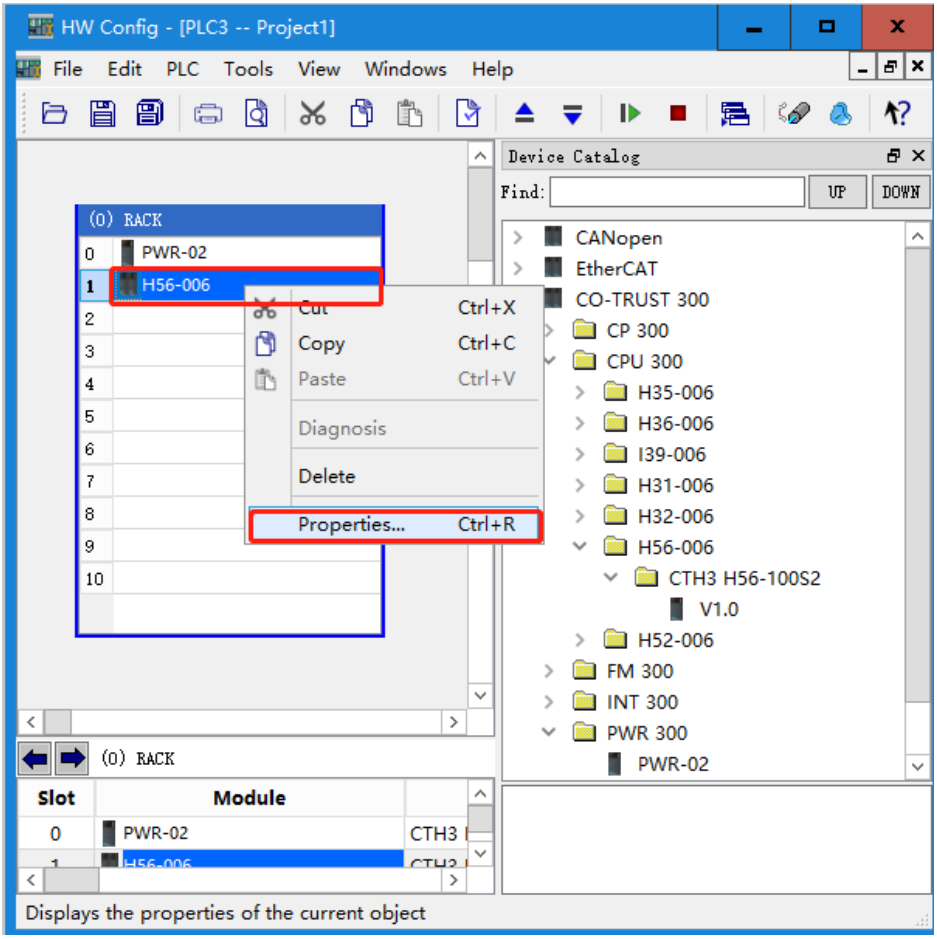
In the hardware configuration, add the power supply module and CPU, the power supply module can only be placed in the 0th slot of the rack, the CPU can only be placed in the 1st slot of the rack.

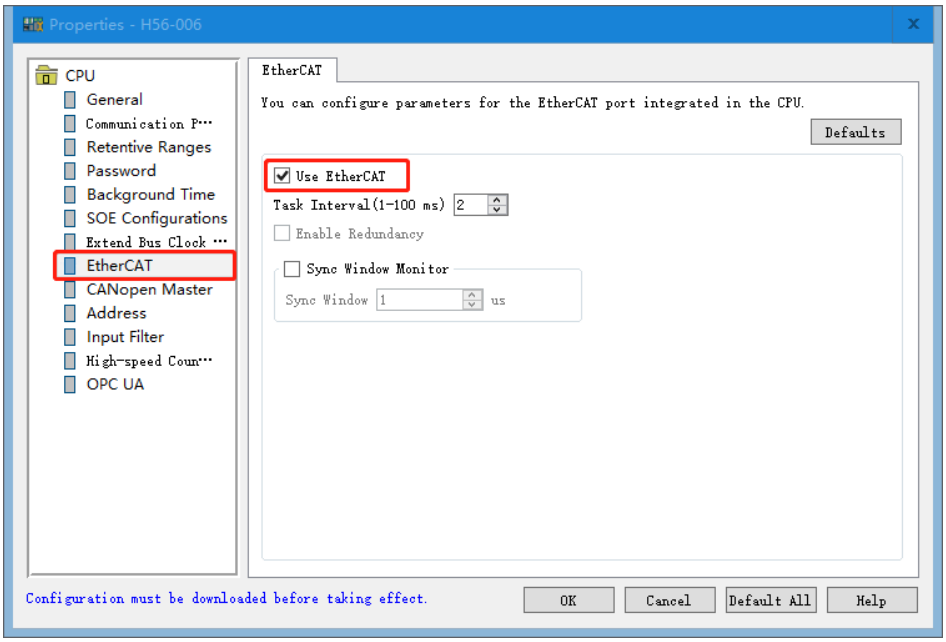


Step 2 : Configure the axis

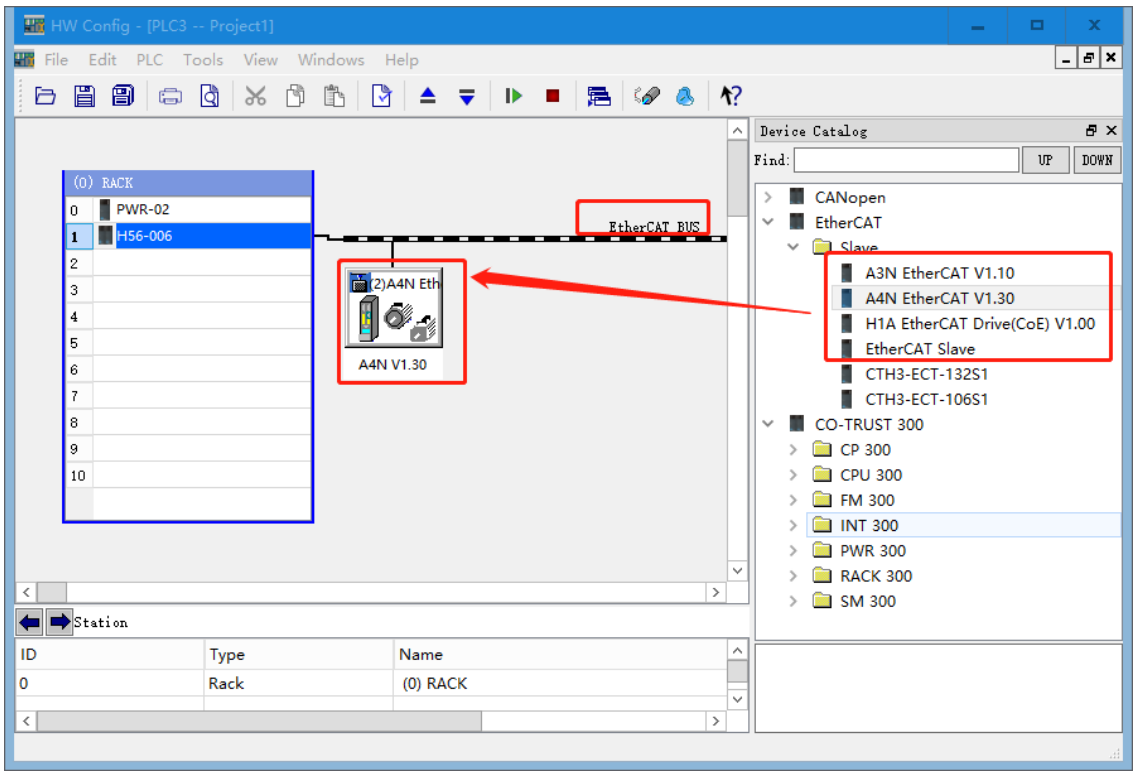
adding a communication axis, you need to add the corresponding EtherCAT slave in the hardware configuration interface, otherwise the communication axis cannot be added. The operations are as follows :

1) Right-click the H56 on the rack, select "Properties" " EtherCAT ", and click "Use EtherCAT".

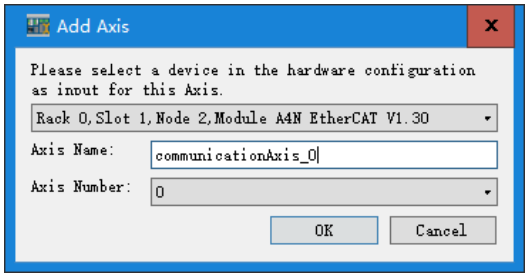




2) After enable EtherCAT Master, the EtherCAT BUS will appear in the configuration interface, and you can connect the EtherCAT slave to the EtherCAT BUS.

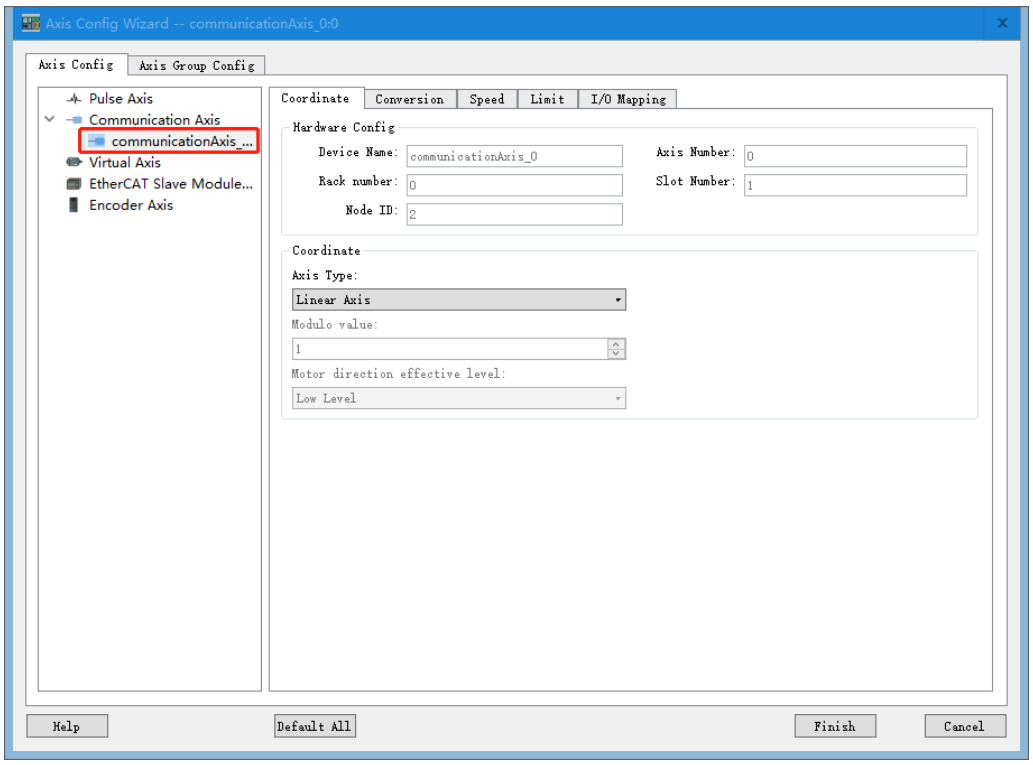


3) After completing the above configuration, select "Tools" → "Axis Wizard Configuration", double-click "Communication Axis" in the axis wizard configuration interface to add a communication axis, "Axis Name" can be modified, click "OK" to complete the addition.



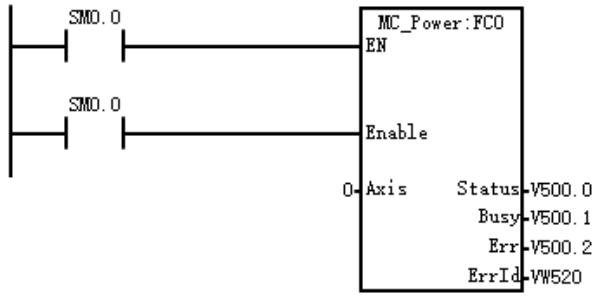
Note: Only one communication axis can be added to a slave.

4) The interface of the added communication axis is as follows:

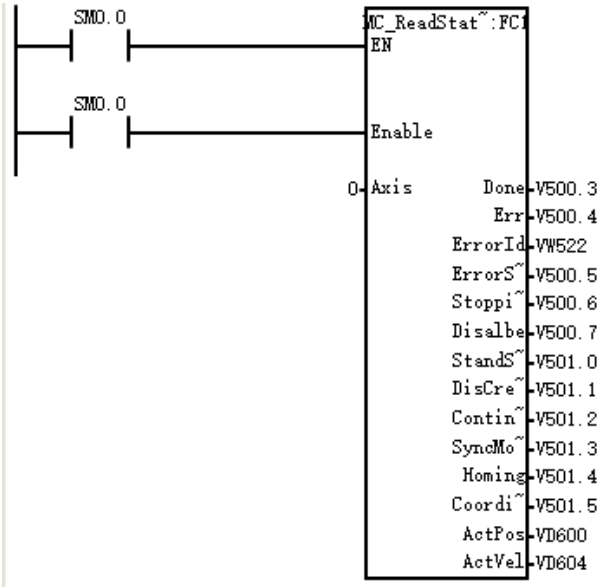


Step3: Call the SMC_Home instruction (Special origin return instruction) in the main program for programming.

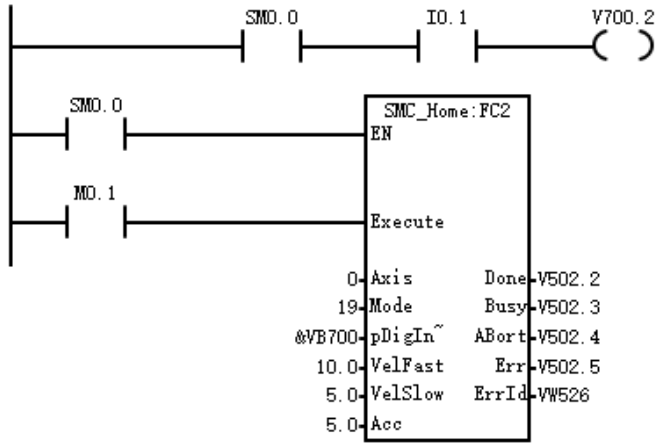
Network 1: Enable the axis, and the axis ID is set to 0.



Network 2: read axis status



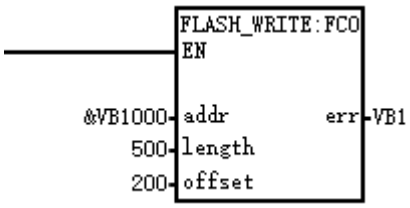
Network 3: Call the SMC_Home command, the homing mode is 19, the axis runs in the positive direction with a search speed of 10.0, and runs in the negative direction at a speed of 5.0 after encountering the rising edge of the origin signal V700.2. And encounters the origin signal V700.2 After the falling edge, the axis stops and the position is cleared. The return is successful, the servo parameter 16#6064:0 (position actual value) does not change.



H Ct_flash_access_lib（v1.0）

ct_ flash_ access_ Lib provides two interfaces for users to save data to or read data from flash, up to 256Kb.

H.1 FLASH_WRITE

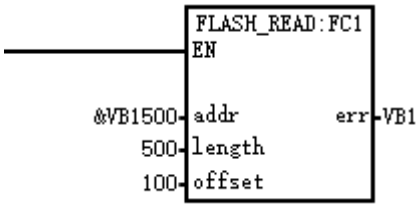


Function: save data to flash

parameter

Name	Input/Output	Data type	Description
addr	IN	DWORD	Start address pointer of data written to FLASH
length	IN	DWORD	Data length written to FLASH
offset	IN	DWORD	The offset of the written data in the FLASH block
err	OUT	BYTE	Write error code 0 No error 1 Exceeding the range of user data storage block 2 Part of the data address is illegal 3 Failed to read data from flash

H.2 FLASH_READ



Function: read data from flash

parameter

Name	Input/Output	Data type	Description
addr	IN	DWORD	Read data start address pointer
length	IN	DWORD	Read data length
offset	IN	DWORD	The offset of the read data in the flash block
err	OUT	BYTE	Read error code 0: No error 1: Out of range of user data storage block 2: Some data addresses are illegal 3: Failed to read data from flash

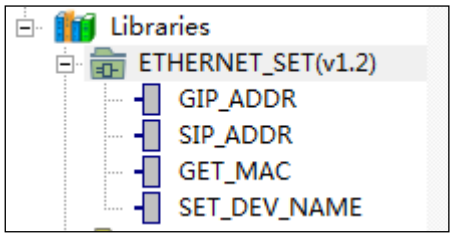
I Introduction of ETHERNET_SET (V1.2) Instructions

I.1 Features

The Ethernet setting instruction library ETHERNET_SET (V1.2) is used to set the local communication parameters of the PLC Ethernet port. You can set and obtain the IP address, MAC address and device name of the remote Ethernet PLC without stopping the CPU.

I.2 Detailed Instructions

The Ethernet setting instruction library contains the following four instructions. The related library files can be downloaded for free from the Co-trust official website at http://www.co-trust.com.

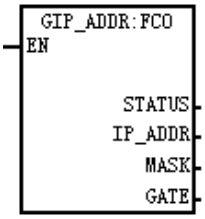


Get IP address instruction

① Name: GIP_ADDR

② Function: Get IP address

Parameter description

Library Functions	Name	Input/ output	Data type	Description
	EN	IN	BOOL	Enable terminal, allow SM0.0 to call
	STATUS	OUT	BYTE	Status word bit0=1 means successful acquisition bit1=1 means the acquisition failed Other bits are not used
	IP_ADDR	OUT	DWORD	IP address,contains four bytes, and each byte represents the corresponding four values of the IP address from low to high.
	MASK	OUT	DWORD	The subnet mask contains four bytes, and each byte represents the corresponding four values of the subnet mask from low to high.
	GATE	OUT	DWORD	Gateway: contains four bytes. Each byte represents four values of the gateway from low to high.

Set the IP address

① Name: SIP_ADDR

② Function: Set IP address

Parameter description

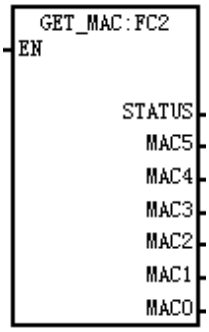
Library Functions	Name	Input/ output	Data type	Description
	EN	IN	BOOL	The enable terminal is triggered by an edge, and the setting command cannot be called cyclically, because the setting of IP and EPPROM has the same number of times, and the continuous writing of the Ethernet will not be able to communicate.
	IP_ADDR	IN	DWORD	The IP address has four bytes in total, and each byte represents the corresponding four values of the IP address from low to high.
	MASK	IN	DWORD	The subnet mask has a total of four bytes, and each byte represents the corresponding four values of the subnet mask from low to high.
	GATE	IN	DWORD	Gateway, a total of four bytes, each byte represents the corresponding four values of the gateway from low to high.
	STATUS	OUT	BYTE	Status word bit0=1 means the setting is successful bit1=1 means illegal IP address bit2=1 means that the IP does not match the mask Bit3=1 means that the IP does not match the gateway Other bits are not used

Obtain the MAC address

① Name: GET_MAC

② Function: Get MAC address

Parameter description

Library Functions	Name	Input/ output	Data type	Description
	EN	IN	BOOL	Enable terminal, allow SM0.0 to call
	STATUS	OUT	BYTE	Status word bit0=1 means successful acquisition Other bits are not used
	MAC0~ MAC5	OUT	BYTE	MAC address xx:xx:xx:xx:xx:xx MAC5------MAC0

Set device name

① Name: SET_DEV_NAME

② Function: Set device name

Parameter description

Library Functions	Name	Input/ output	Data type	Description
	EN	IN	BOOL	The enable terminal is triggered by an edge, and the setting command cannot be called cyclically, because the setting of IP and EPPROM has the same number of times, and the continuous writing of the Ethernet will not be able to communicate.
	NAME	IN	DWORD	A pointer to the device name, the first byte pointing to the area indicates the length. The entire string contains a maximum of 32 bytes in length.
	STATUS	OUT	BYTE	Status word Bit0 = 1 means the setting is successful. Bit1 = 1 means the length is wrong. Bit2 = 1 means saving failed. bit3 represents illegal characters

I.3 Application Example

The application example program of the Ethernet instruction setting library is as follows:

Network 1

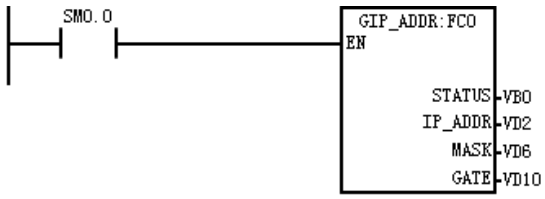
Get IP address

STATUS: Status word, when bit0 is 1 it means the acquisition is successful.When bit1 is 1, it means that the acquisition failed,Other bits are not used.

IP_ADDR: P address

MASK: subnet mask

GATE: Gateway



Network 2

Set IP address

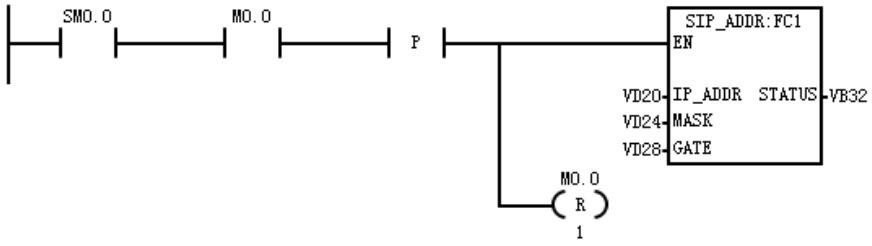
IP_ADDR: P address

MASK: subnet mask

GATE: Gateway

STATUS: Status word bit0 is 1 when the setting is completed.When bit1 is 1, it means illegal IP address. When bit2 is 1, it means that the IP does not match the mask. When bit3 is 1, it means that the IP does not match the gateway. Other bits are not used

Note that this setting instruction uses edge trigger and cannot be used cyclically, because there is a limit on the number of times to set IP and EPPROM.

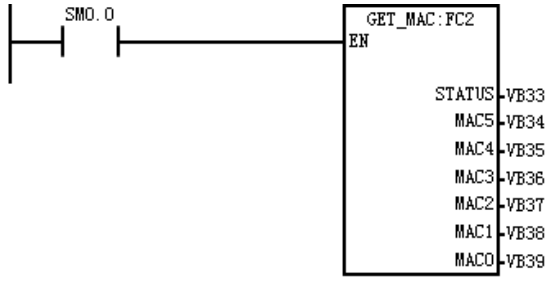


Network 3

Get MAC address

STATUS--Status word bit0 is 1 when it means the acquisition is successful. Other bits are not used.

MAC0~5: MAC address

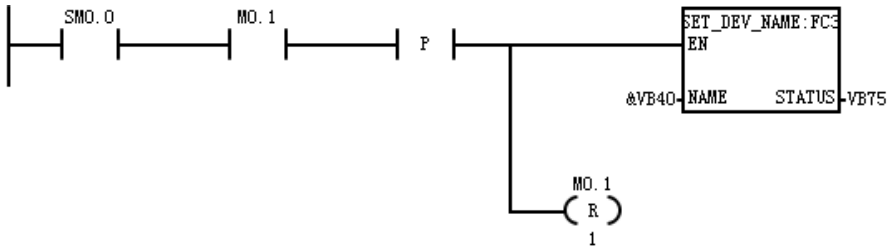


Network 4 program notes

NAME: Pointer to the device name, the first byte pointing to the area indicates the length, the entire string contains length bytes up to 32bytes.

STATUS: bit0 is 1 means setting is successful, bit1 is 1 means length is wrong, bit2 is 1 means saving failed, bit3 is 1 means illegal character.

Note: This command cannot be called cyclically, because the setting of IP and EPPROM is limited in the number of times, and edge triggering is used.



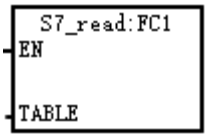
J Introduction of S7_Protocol（v1.0）) Instructions

The S7 protocol library has the following commands:

- S7_read
- S7_write

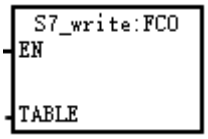
The PLC models that support S7_Protocol (v1.0) are: H56-10, H52-10.

J.1 S7-read



Function: S7 master performs network read operation

J.2 S7- write



Function: S7 master performs network write operation

TABLE parameter table of S7_read/S7_write instruction:

D	A	E	0	error code	0
The first byte of the slave IP					1
The second byte of the slave IP					2
The third byte of the slave IP					3
The fourth byte of the slave IP					4
Reserved (must be set to 0)					5
Reserved (must be set to 0)					6
Point to the first byte of the slave target area pointer					7
Point to the second byte of the target area pointer of the slave					8
Point to the 3rd byte of the slave target area pointer					9
Point to the 4th byte of the slave target area pointer					10
Length (not more than 200)					11
Point to the first byte of the target area pointer of the local station					12
Point to the second byte of the target area pointer of the local station					13
Point to the 3rd byte of the target area pointer of the local station					14
Point to the 4th byte of the target area pointer of the local station					15

D: complete bit, 0 is not completed, 1 is completed

A: Active bit, 1 when the instruction is being executed, 0 if it is not being executed

E: Error bit, 0 is no error, 1 is error

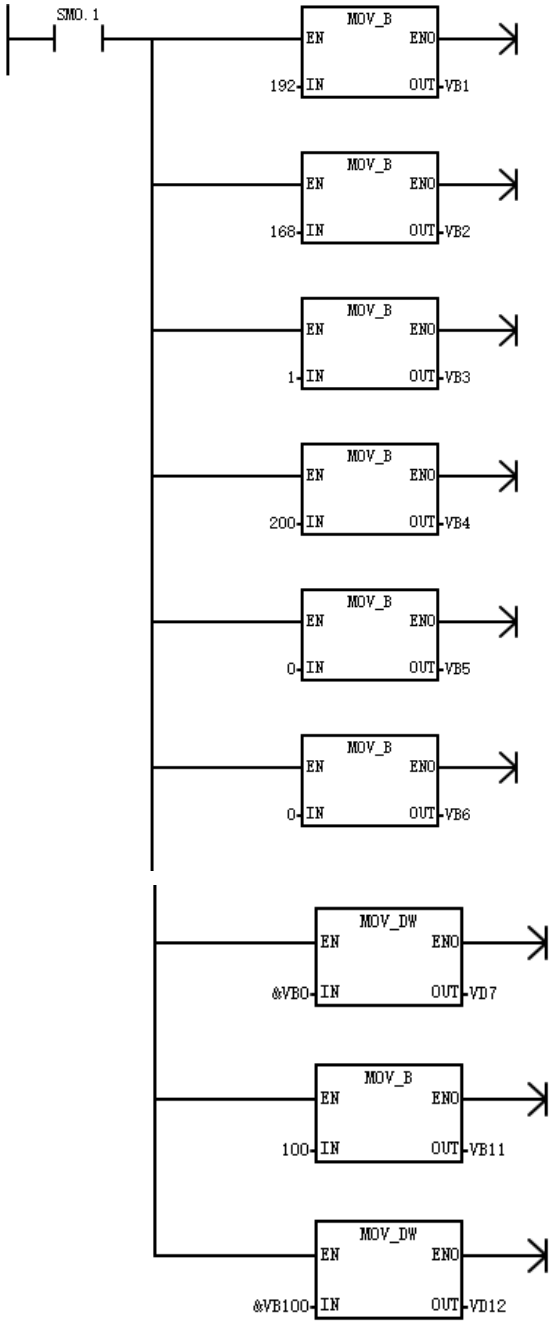
Error code table

Code	Describe
0	No error
1	Timeout error: the slaves remote station does not answer
2	Reserved
3	Offline errors: conflicts caused by duplicate site addresses or faulty hardware
4	Queue overflow error: 8 S7_write/S7_read boxes are activated
5	Receiving error: The reply received was wrong
6	Illegal parameter: The S7_write/S7_read table contains an illegal or invalid value

7	Reserved
8	Reserved
9	Information error: Incorrect data length
10	More than 8 connections
11	The network cable is not plugged in

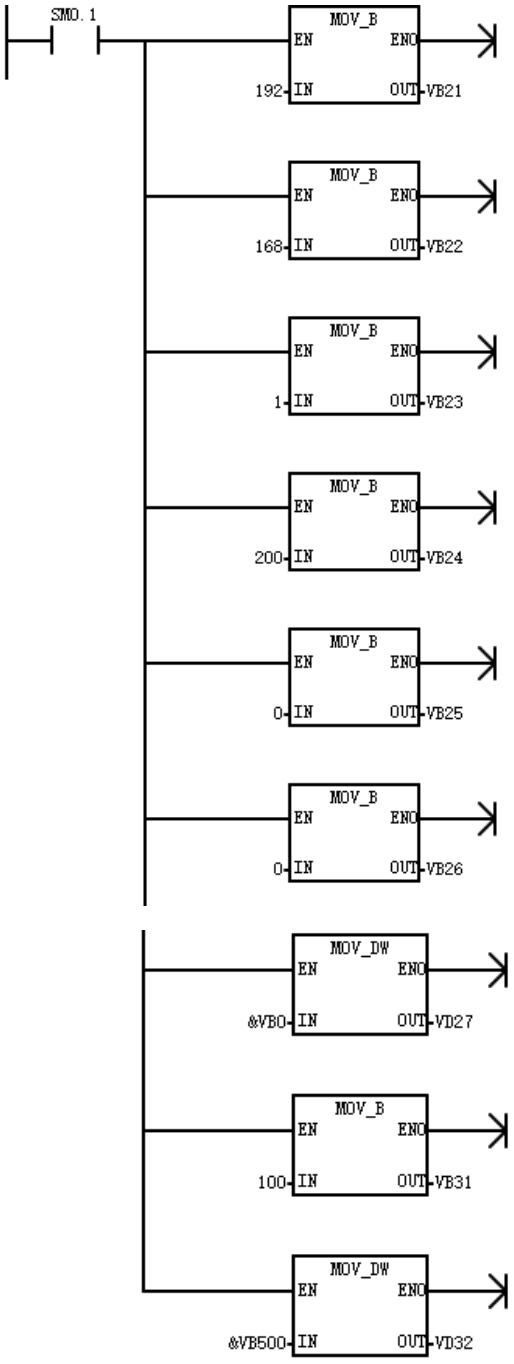
J.3 Example

Network 1: According to the TABLE parameter table of the S7_read/S7_write instruction, set the IP of the slave to 192.168.1.200, set the target area pointer of the slave to &VB0 (must use the pointer), set the read length to 100 bytes, and set the target of the local station The area pointer is set to &VB100 (the pointer must be used), and the parameter table is configured.

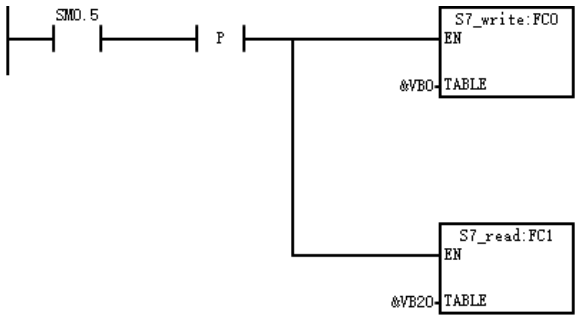


Network 2: Configure the parameter table according to the TABLE parameter table of the

S7_read/S7_write instruction. Set the IP of the slave to 192.168.1.200, set the target area pointer of the slave to &VB0 (must use pointer), set the write length to 100 bytes, and set the target area pointer of the master to &VB100 (must use pointer).



Network 3: Invoke the S7_read/S7_write instruction, read and write data through the parameters of the parameter table, and the TABLE parameter of the instruction must use pointers.



K Special Function Register

Special memory function description

Special register	Description
SMB0	System status bit
SM0.0	The bit is always on.
SM0.1	The bit is turned on during the first scan cycle. One use is to call the initialization subroutine.
SM0.2	If the reserved data is lost, the bit is turned on for one scan cycle. And the bit can be used as an error memory bit or as a mechanism to activate a special boot sequence.
SM0.3	When entering the RUN mode from the power-on condition, this bit is turned on for one scan cycle. This bit can be used to provide machine warm-up time before starting operation.
SM0.4	This bit provides a clock pulse, which is OFF for 30 seconds and ON for 30 seconds within a period of 1 minute. This bit provides an easy-to-use delay or 1 minute clock pulse.
SM0.5	This bit provides a clock pulse, which is OFF for 0.5 seconds and ON for 0.5 seconds within a period of 1 second. This bit provides an easy-to-use delay or 1 second clock pulse.
SM0.6	This bit is the scan cycle clock, which is turned on for one scan, and then turned off for the next scan. This bit can be used as a scan counter input.
SM0.7	This bit represents the current position of the "mode" switch (close = "stop" position, open = "run" position). When the switch is in the RUN position, you can use this bit to enable the free port mode, and you can use the method of switching to the "stop" position to re-enable the normal communication with the PC/programming device.
SMB1	Instruction execution status bit
SM1.0	When the operation result is zero, the execution of certain instructions turns on this bit.
SM1.1	When an overflow result or an illegal digital value is detected, the execution of certain instructions turns on this bit.
SM1.2	When a mathematical operation produces a negative result, this bit is turned on.

SM1.3	When trying to divide by zero, the bit turns on.	
SM1.4	This bit is turned on when the "add to table" command attempts to overfill the table.	
SM1.5	When a LIFO or FIFO instruction attempts to read from an empty table, this bit is turned on.	
SM1.6	When trying to convert a non-BCD value to a binary value, this bit is turned on.	
SM1.7	When the ASCII value cannot be converted into a valid hexadecimal value, this bit is turned on.	
SMB2	Freeport to receive characters	
SMB3	Freeport check error	
SM3.0	This bit indicates that a parity error occurred in port 0 and port 1. (0 = no error. 1 = error)	
SM3.1~SM3.3	Reserved	
SM3.4	If there are any I/O errors, this bit is turned on.	
SM3.5	If too many digital I/O modules are connected to the I/O Bus, this bit is turned on.	
SM3.6	If too many analog I/O modules are connected to the I/O Bus, this bit is turned on.	
SM3.7	If too many intelligent I/O modules are connected to the I/O Bus, this bit is turned on.	
SMB4	Interrupt queue overflow, runtime program error, interrupt enable, free port transmitter is forced	
SM4.0	When the communication interrupt queue overflows, this bit turns on.	
SM4.1	When the input interrupt queue overflows, this bit turns on.	
SM4.2	When the timer interrupt queue overflows, this bit is turned on.	
SM4.3	When a runtime programming error is detected, this bit is turned on.	
SM4.4	This bit reflects the global interrupt enable status. When interrupts are enabled, this bit is turned on.	
SM4.5	When the transmitter is idle (port 0), this bit is turned on.	
SM4.6	When the transmitter is idle (port 1), this bit is turned on.	
SM4.7	This bit is turned on when any memory location is forced.	
SMB5	The interrupt event number that triggered the current OB	
SMB6		
SM6.0	Reserved (channel information exists)	Module diagnosis information
SM6.1	Reserved (User information exists)	
SM6.2	Reserved (from alternate diagnostic interrupt)	
SM6.3	Reserved	
SM6.4～SM6.7	PLC type: 1110: H35-00,H36-00,I39-00,H56-10,H52-10	
SMB7		
SM7.0	CPU external error	
SM7.1	CPU external error	
SM7.2	Communication error	
SM7.3	Operation mode (0: run, 1: stop)	

SM7.4	Watchdog timer expired	
SM7.5	Reserved (Internal power failure)	
SM7.6	Reserved (backplane failure)	
SM7.7	0: The programming card exists and works normally	
SMD8	1: The programming card does not exist or works abnormally	
SMB12~SMB21	Reserved	
SMW22	Last scan time	
SMW24	Minimum scan time since entering RUN mode	
SMW26	Maximum scan time since entering RUN mode	
SMW28	The logical address of the module that generated the diagnostic interrupt (fault occurred) or hardware interrupt is only valid when accessed in the interrupt function. SMB28: the rack number of the module (0~3), SMB29: the slot number of the module (0~10).	
SMB30	Freeport control register	
Bit7:6	00: no parity. 01: even parity. 10: no parity. 11: odd parity	
Bit5	0: 8 data bits per character. 1: 7 data bits per character	
Bit4:2	000: 38400bps. 001: 19200bps. 010: 9600bps. 011: 4800bps. 100: 2400bps. 101: 1200bps. 110: 115200bps. 111: 57600bps	
Bit1:0	00: Point-to-point protocol (PPI/slave mode). 01: Freeport protocol 10: PPI/master mode. 11: reserved (PPI/slave mode default value)	
SMB31	Reserved	
SMW32	Reserved	
SMB34	Millisecond timer interrupt 0 time interval value (ms), corresponding to interrupt event number 10	
SMB35	Millisecond timer interrupt 1 time interval value (ms), corresponding to interrupt event number 11	
SMB36~SMB65	Reserved	
SMB66~SMB85	Reserved	
SMB86~SMB94 SMB186~SMB194	Freeport receiving information control	
SMW96	Microsecond interrupt 0 time interval value (100us), corresponding to interrupt event number 34	
SMW98	Microsecond interrupt 1 time interval value (100us), corresponding to interrupt event number 35	
SMB100	Programming card limit times (0 means no limit on times)	
SMB101	The remaining use times of the programming card (including this time), if there is no limit, 255 will be displayed.	
SMW102	CPU firmware version	
SMW104	CPU hardware version	
SMB106~SMB135	Reserved	
SMB136~SMB165	Reserved	

SMB166~SMB185		Reserved
SMB200~SMB205		Save and display the MAC address of the host
SMB206		Number of online master s for MODBUS TCP communication
SMD207		MODBUS TCP data packet statistics: the number of data packets received
SMD211		MODBUS TCP data packet statistics: the number of data packets sent
SMB215~SMB222		Reserved
SMB223		Number of online master for UDP_PPI communication
SMD224		UDP_PPI data packet statistics: the number of data packets received
SMD228		UDP_PPI data packet statistics: the number of data packets sent
SMB232~SMB389		Reserved
SMB390		
SM390.0		EtherCAT initialization complete bit. 0: Not initialized 1: Initialization completed
SM390.1		EtherCAT redundancy activation bit. 0: Redundancy is not activated 1: Redundancy activated, you need to check the connection at this time
SM390.2~SM390.7		Reserved
SMB391		Reserved
SMD392		EtherCAT task cycle execution time
SMD396		EtherCAT task cycle interval time
SMB400~SMB465 (EtherCAT register)	SMB400	Number of EtherCAT slaves found
	SMB401	EtherCAT error: 0: no error 1: Configuration parameter error 2: No slave was found 3: State switching error 4: An error occurred while writing the configuration 5: The number of slaves is wrong 6: The slave does not match
	SMB402	Slave 1 status: 0: No connection 1: Initialization state 2: Pre-operational state 4: Safe operation 8: Operating status 16#80 Product ID does not match 16#81 Vendor ID does not match 16#82 SDO write error Other: wrong state
	SMB403~SMB465	Slave 2 status ~ Slave 64 status

SMB466~SMB499	Reserved
SMW470	Number of EtherCAT error packets
SMB500	The first module (Rack0, Slot3) ierror
SMB501	The second module (Rack0, Slot4) error
SMB502	The third module (Rack0, Slot5) error
SMB503	The fourth module (Rack0, Slot6) error
SMB504	The fifth module (Rack0, Slot7) error
SMB505	The sixth module (Rack0, Slot8) error
SMB506	The seventh module (Rack0, Slot9) error
SMB507	The eighth module (Rack0, Slot10) error
SMB508	The ninth module (Rack1, Slot3) error
SMB509	The tenth module (Rack1, Slot4) error
SMB510	The eleventh module (Rack1, Slot5) error
SMB511	The twelfth module (Rack1, Slot6) error
SMB512	The thirteenth module (Rack1, Slot7) error
SMB513	The fourteenth module (Rack1, Slot8) error
SMB514	Fifteenth module (Rack1, Slot9) error
SMB515	Sixteenth module (Rack1, Slot10) error
SMB516	The seventeenth module (Rack2, Slot3) error
SMB517	Eighteenth module (Rack2, Slot4) error
SMB518	The nineteenth module (Rack2, Slot5) error
SMB519	Twentieth module (Rack2, Slot6) error
SMB520	The twenty-first module (Rack2, Slot7) error
SMB521	The 22nd module (Rack2, Slot8) error
SMB522	Twenty-third module (Rack2, Slot9) error
SMB523	Twenty-fourth module (Rack2, Slot10) error
SMB524	The twenty-fifth module (Rack3, Slot3) error
SMB525	The twenty-sixth module (Rack3, Slot4) error
SMB526	The twenty-seventh module (Rack3, Slot5) error
SMB527	The twenty-eighth module (Rack3, Slot6) error
SMB528	The twenty-ninth module (Rack3, Slot7) error
SMB529	The thirtieth module (Rack3, Slot8) error
SMB530	The thirty-first module (Rack3, Slot9) error
SMB531	Thirty-second module (Rack3, Slot10) error
SMD532	Total number of link layer errors of the expansion Bus
SMB536~SMB549	Reserved
SMB550~SMB589	The first CAN module information (arranged by hardware configuration)
SMB550	Master status 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run 0xFF: configuration data error

SMB551	Slave 1 status (arranged by node ID) 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB552~SMB581	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID)
SMB582	Slave 32 status (arranged by node ID) 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB583	CAN module software version number
SMB584~SMB589	Reserved
SMB590~SMB629	The second CAN module information (arranged by hardware configuration）
SMB590	Master status 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run 0xFF: configuration data error
SMB591	Slave 1 status (arranged by node ID) 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB592~SMB621	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID）
SMB622	Slave 32 status (arranged by node ID) 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB623	CAN module software version number
SMB624~SMB629	Reserved
SMB630~SMB669	The third CAN module information (arranged by hardware configuration）
SMB630	Master status 0x00: Initialize 0x01: Disconnect

	0x04: stop 0x05: Run 0x7F: pre-run 0xFF: configuration data error
SMB631	Slave 1 status (arranged by node ID) 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB632~SMB661	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID)
SMB662	Slave 32 status (arranged by node ID) 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB663	CAN module software version number
SMB664~SMB669	Reserved
SMB670~SMB709	Information of the fourth CAN module (arranged by hardware configuration)
SMB670	Master status 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run 0xFF: configuration data error
SMB671	Slave 1 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB672~SMB701	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID)
SMB702	Slave 32 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB703	CAN module software version number
SMB704~SMB709	Reserved
SMB710~SMB749	The fifth CAN module information (arranged by hardware

	configuration)
SMB710	Master status 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run 0xFF: configuration data error
SMB711	Slave 1 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB712~SMB741	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID)
SMB742	Slave 32 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB743	CAN module software version number
SMB744~SMB749	Reserved
SMB750~SMB789	The sixth CAN module information (arranged by hardware configuration)
SMB750	Master status 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run 0xFF: configuration data error
SMB751	Slave 1 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB752~SMB781	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID)
SMB782	Slave 32 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run

	0x7F: pre-run
SMB783	CAN module software version number
SMB784~SMB789	Reserved
SMB790~SMB829	The seventh CAN module information (arranged by hardware configuration)
SMB790	Master status 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run 0xFF: configuration data error
SMB791	Slave 1 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB792~SMB821	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID)
SMB822	Slave 32 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB823	CAN module software version number
SMB824~SMB829	Reserved
SMB830~SMB869	The eighth CAN module information (arranged by hardware configuration)
SMB830	Master status 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run0x7F: Pre-run 0xFF: configuration data error
SMB831	Slave 1 status (arranged by node ID) 0x00: Initialization 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB832~SMB861	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID)
SMB862	Slave 32 status (arranged by node ID) 0x00: Initialization

	0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB863	CAN module software version number
SMB864~SMB869	Reserved
SMB870	Module internal diagnosis control word 1: Obtain the internal diagnosis of the module specified by SMB871 and write it into the byte behind SMB872
SMB871	Internal diagnostic module number Need to get the module number of internal diagnosis, 0~3: SlotID. 4~7: RackID.
SMB872~SMB905	Module internal diagnosis
SMB906~SMB999	Reserved
SMB1000~SMB1015	Description information of the first module (rack0, slot3)
SMB1000~SMB1002	Actual module information
SMB1003	Module version number
SMD1004	Module error count
SMB1008~SMB1015	Reserved
SMB1016~SMB1031	Description information of the second module (rack0, slot4)
SMB1016~SMB1018	Actual module information
SMB1019	Module version number
SMD1020	Module error count
SMB1024~SMB1031	Reserved
SMB1032~SMB1047	The third module (Rack0, Slot5) description information
SMB1032~SMB1034	Actual module information
SMB1035	Module version number
SMD1036	Module error count
SMB1040~SMB1047	Reserved
SMB1048~SMB1063	The fourth module (Rack0, Slot6) description information
SMB1048~SMB1050	Actual module information
SMB1051	Module version number
SMD1052	Module error count
SMB1056~SMB1063	Reserved
SMB1064~SMB1079	The fifth module (Rack0, Slot7) description information
SMB1064~SMB1066	Actual module information
SMB1067	Module version number
SMD1068	Module error count
SMB1072~SMB1079	Reserved
SMB1080~SMB1095	Description of the sixth module (Rack0, Slot8)
SMB1080~SMB1082	Actual module information
SMB1083	Module version number
SMD1084	Module error count
SMB1088~SMB1095	Reserved

SMB1096~SMB1111	The seventh module (Rack0, Slot9) description information
SMB1096~SMB1098	Actual module information
SMB1099	Module version number
SMD1100	Module error count
SMB1104~SMB1111	Reserved
SMB1112~SMB1127	The eighth module (Rack0, Slot10) description information
SMB1112~SMB1114	Actual module information
SMB1115	Module version number
SMD1116	Module error count
SMB1120~SMB1127	Reserved
SMB1128~SMB1143	Description of the ninth module (Rack1, Slot3)
SMB1128~SMB1130	Actual module information
SMB1131	Module version number
SMD1132	Module error count
SMB1136~SMB1143	Reserved
SMB1144~SMB1159	The tenth module (Rack1, Slot4) description information
SMB1144~SMB1146	Actual module information
SMB1147	Module version number
SMD1148	Module error count
SMB1152~SMB1159	Reserved
SMB1160~SMB1175	Description of the eleventh module (Rack1, Slot5)
SMB1160~SMB1162	Actual module information
SMB1163	Module version number
SMD1164	Module error count
SMB1168~SMB1175	Reserved
SMB1176~SMB1191	Description information of the twelfth module (Rack1, Slot6)
SMB1176~SMB1178	Actual module information
SMB1179	Module version number
SMD1180	Module error count
SMB1184~SMB1191	Reserved
SMB1192~SMB1207	The thirteenth module (Rack1, Slot7) description information
SMB1192~SMB1194	Actual module information
SMB1195	Module version number
SMD1196	Module error count
SMB1200~SMB1207	Reserved
SMB1208~SMB1223	The fourteenth module (Rack1, Slot8) description information
SMB1208~SMB1210	Actual module information
SMB1211	Module version number
SMD1212	Module error count
SMB1216~SMB1223	Reserved
SMB1224~SMB1239	fifteenth module (Rack1, Slot9) description information
SMB1224~SMB1226	Actual module information
SMB1227	Module version number
SMD1228	Module error count
SMB1232~SMB1239	Reserved

SMB1240~SMB1255	Sixteenth module (Rack1, Slot10) description information
SMB1240~SMB1242	Actual module information
SMB1243	Module version number
SMD1244	Module error count
SMB1248~SMB1255	Reserved
SMB1256~SMB1271	Seventeenth module (Rack2, Slot3) description information
SMB1256~SMB1258	Actual module information
SMB1259	Module version number
SMD1260	Module error count
SMB1264~SMB1271	Reserved
SMB1272~SMB1287	Description of the eighteenth module (Rack2, Slot4)
SMB1272~SMB1274	Actual module information
SMB1275	Module version number
SMD1276	Module error count
SMB1280~SMB1287	Reserved
SMB1288~SMB1303	The nineteenth module (Rack2, Slot5) description information
SMB1288~SMB1290	Actual module information
SMB1291	Module version number
SMD1292	Module error count
SMB1296~SMB1303	Reserved
SMB1304~SMB1319	Description of the twentieth module (Rack2, Slot6)
SMB1304~SMB1306	Actual module information
SMB1307	Module version number
SMD1308	Module error count
SMB1312~SMB1319	Reserved
SMB1320~SMB1335	The twenty-first module (Rack2, Slot7) description information
SMB1320~SMB1322	Actual module information
SMB1323	Module version number
SMD1324	Module error count
SMB1328~SMB1335	Reserved
SMB1336~SMB1351	Description of the twenty-second module (Rack2, Slot8)
SMB1336~SMB1338	Actual module information
SMB1339	Module version number
SMD1340	Module error count
SMB1344~SMB1351	Reserved
SMB1352~SMB1367	Twenty-third module (Rack2, Slot9) description information
SMB1352~SMB1354	Actual module information
SMB1355	Module version number
SMD1356	Module error count
SMB1360~SMB1367	Reserved
SMB1368~SMB1383	Description information of the twenty-fourth module (Rack2, Slot10)
SMB1368~SMB1370	Actual module information
SMB1371	Module version number
SMD1372	Module error count
SMB1376~SMB1383	Reserved

SMB1384~SMB1399	Description of the twenty-fifth module (Rack3, Slot3)
SMB1384~SMB1386	Actual module information
SMB1387	Module version number
SMD1388	Module error count
SMB1392~SMB1399	Reserved
SMB1400~SMB1415	The twenty-sixth module (Rack3, Slot4) description information
SMB1400~SMB1402	Actual module information
SMB1403	Module version number
SMD1404	Module error count
SMB1408~SMB1415	Reserved
SMB1416~SMB1431	The twenty-seventh module (Rack3, Slot5) description information
SMB1416~SMB1418	Actual module information
SMB1419	Module version number
SMD1420	Module error count
SMB1424~SMB1431	Reserved
SMB1432~SMB1447	Description of the twenty-eighth module (Rack3, Slot6)
SMB1432~SMB1434	Actual module information
SMB1435	Module version number
SMD1436	Module error count
SMB1440~SMB1447	Reserved
SMB1448~SMB1463	The twenty-ninth module (Rack3, Slot7) description information
SMB1448~SMB1450	Actual module information
SMB1451	Module version number
SMD1452	Module error count
SMB1456~SMB1463	Reserved
SMB1464~SMB1479	Thirtieth module (Rack3, Slot8) description information
SMB1464~SMB1466	Actual module information
SMB1467	Module version number
SMD1468	Module error count
SMB1472~SMB1479	Reserved
SMB1480~SMB1495	The thirty-first module (Rack3, Slot9) description information
SMB1480~SMB1482	Actual module information
SMB1483	Module version number
SMD1484	Module error count
SMB1488~SMB1495	Reserved
SMB1496~SMB1511	The 32nd module (Rack3, Slot10) description information
SMB1496~SMB1498	Actual module information
SMB1499	Module version number
SMD1500	Module error count
SMB1504~SMB1549	internal use
SMB1550~SMB1589	Native CANOpen information
SMB1550	Master status 0x00: Initialize 0x01: Disconnect 0x04: stop

	0x05: Run 0x7F: pre-run 0xFF: configuration data error
SMB1551	Slave 1 status (arranged by node ID) 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB1552~SMB1581	Slave 2 status (arranged by node ID) ~ Slave 31 status (arranged by node ID)
SMB1582	Slave 32 status (arranged by node ID) 0x00: Initialize 0x01: Disconnect 0x04: stop 0x05: Run 0x7F: pre-run
SMB1583~SMB2019	internal use
SMD2020	Expansion module interrupt times
SMB2024~SMB2027	internal use
SMD2028	Number of local interrupts
SMB2032~SMB2047	internal use

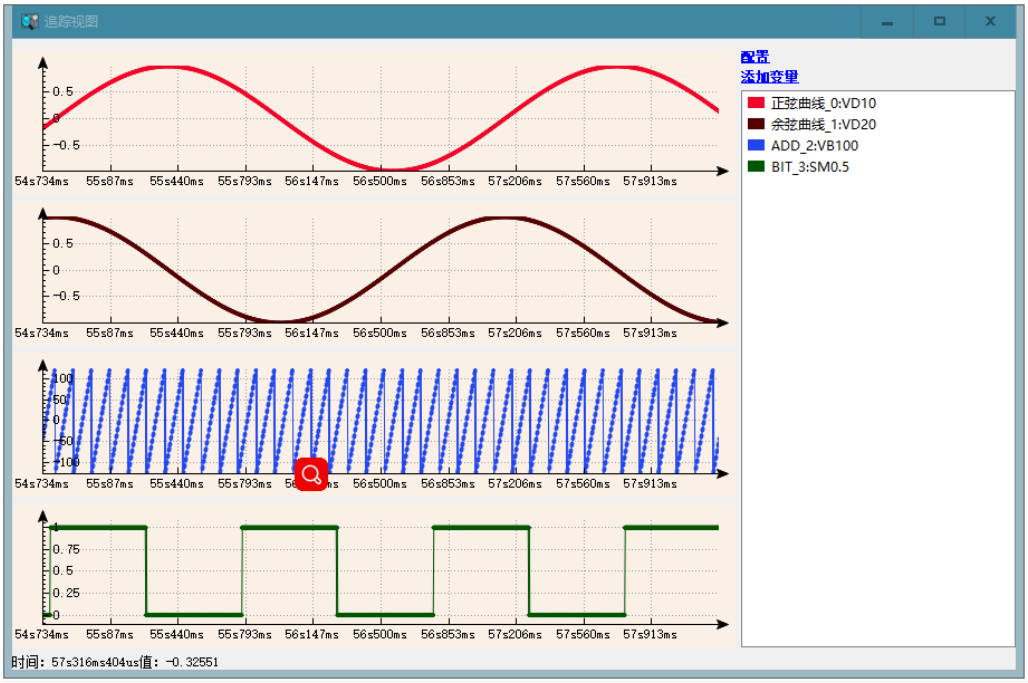
L Trace Function

The Trace function is a newly added function in the State Char control of MagicWorks PLC software, which is used to configure and display application-specific trace data in one or more charts.

When the application is running, you can view the value curve record of the trace variable in the "Trace View" of MagicWorks PLC. The requirement is to set up the trace configuration, transfer the trace configuration to the PLC, and start the trace recording. The recorded data will be transmitted to the host computer and displayed in chart form according to the configuration. You can browse the data while trace.

The user can start the trace view in the “Status Char -> menu -> view -> trace”or click the icon  in the State Char toolbar.

As shown in the following figure: the data display chart is on the left and the variable list is on the right. At present, up to 8 variables and 8 charts can be added.



Configuration

The dialog box includes trace configuration of data records of trace project. Click "configuration" at the top right of "trace view" or right click "configuration" in the pop-up menu to enter the configuration dialog box.

Add variable: add a variable to be monitored. Current specification: up to 8 variables can be added.

• You can click add variable in the upper right corner of "trace view".

• You can add in the right-click menu in the trace record view.

• Select the display variable node of one of the charts in the chart view and add it in the right-click pop-up menu.

Delete variable: select a variable in the trace view and right-click to delete the variable. Confirm to delete the variable. Note: you cannot undo the deletion of the variable.

Assign variable: select a variable in the trace view and select assign to chart in the right-click pop-up menu to assign the variable to the specified chart. When adding variables, they will be assigned to the first chart by default. At present, up to 8 charts can be added.

Add chart: in the chart view, you can insert a new chart in the view through the right-click menu "add chart".

Delete chart: select the chart to delete in the chart view, right-click delete chart and confirm in the dialog box to delete the chart. Note: deleting a chart cannot be undone.

Record setting: Click "Trace" in the trace record to make "Record setting" on the right.

Enable trigger: Record data over a period of time by enabling a variable. Click the check box on the right to enable/disable. When enabling the trigger, you also need to configure the trigger variable, trigger edge and post-trigger sampling time .

Trigger variable: It is valid when the trigger is enabled, and is used as a trigger signal. The valid trigger signal is a variable, a bit address in the PLC. For example, M0.0.

Trigger edge: valid when the trigger is enabled, define the trigger edge detection:

Rising edge: For BOOL trigger variables, the trigger occurs when the value changes from FALSE to TRUE.

Falling edge: For BOOL trigger variables, the trigger occurs when the value changes from TRUE to FALSE.

Rising edge and falling edge: For BOOL trigger variables, the trigger occurs when the value changes.

Sampling time after trigger: The trigger continues sampling for a specified time after triggering and stops sampling.

Sampling period: The defined PLC sampling time period. You can select PLC period, EtherCAT period, pulse period and 1ms interrupt in the drop-down box.

Variable setting: Configure the added variable or the existing variable.

Variable address: Enter the effective address in the PLC to be monitored. When the address is invalid, an error will be reported and must be configured.

Variable name: The name of the current variable, which is convenient for users to identify the variable more easily. It does not need to be configured.

Display format: There are 4 display formats for users to choose: signed, unsigned, floating point and bit.

Curve color: When adding variables, a color will be randomly generated, and users can also modify it by themselves.

Line type: The type of connection between two points is straight line, step and no connection.

Point type: There are three styles of dot, cross and none.

Time axis configuration: Click the time axis node in the chart view to configure the time axis settings.

Display mode: The change mode of the time axis (X axis) during the monitoring data:

Auto: Timeline auto zoom.

Fixed length: Display a constant length.

Fixed: Fixed display a segment of value from minimum to maximum.

Minimum value: The initial value of the segment. It is valid when the display mode is "fixed".

Maximum value: The end value of the segment. It is valid when the display mode is "fixed".

Length: The length of the constant section.

Grid: The grid lines of the time axis direction map.

Y axis configuration

Click the Y-axis node in the chart view to configure the Y-axis settings.

Display mode: the change mode of Y-axis during the process of monitoring data:

Auto: The time axis is automatically zoomed.

Fixed: fixed display a segment of value from minimum to maximum

Minimum value: the initial value of the segment. It is valid when the display mode is "fixed".

Maximum value: the end value of the segment. It is valid when the display mode is "fixed".

Grid: The grid lines of the Y-axis direction map.

Description: The Y axis is marked as "Description".
Preview of the trace chart: You can preview the chart configured in the table, and you can also adjust the color of the chart.

Trace view right-click menu

Configuration: You can open the configuration dialog box, if it is currently in the monitoring state, it will be automatically suspended

Download configuration: Download the configured variables to the PLC. If the data is wrong, an error will be reported, otherwise it will automatically start to read the data. If the variable is modified, the data must be re-downloaded before reading: such as adding or deleting the variable, modifying the address of the variable, display format, etc.

Note: Before downloading, make sure that the software has been connected to the PLC and the "CTH300/200 Local (TCP/IP)" communication interface is selected, otherwise it cannot be connected to the PLC.

Start/temporary reading: After downloading the data, you can start/pause the reading.

Cursor: The cursor  is a trace cursor, parallel to the vertical black line in the y-axis. It can be used to identify the time of the cursor position and the corresponding y value. If there is no corresponding Y value at the time point where the cursor is located, the value of the point closest to the cursor will be displayed.

When no trace cursor is available, add a cursor to the trace graph.

When a trace cursor is available, add another cursor to the trace graph.

When two trace cursors are available, delete the displayed trace cursor.

Cursor trace graph without any trace: In this mode, you can run through the mouse pointer trace graph. The X value centered on the cursor will be displayed in the status bar in a normal style (for example, time: 1m23s456ms Value: 1).

A trace graph for trace cursor: In the status bar and y value, the main mark of the output result is the time of trace the cursor. Example: Time:1m23s456ms.

Trace graph of two trace cursors: In the status bar, output twice, by the two trace cursors (for example, the marked time interval, Time: 1m23s456ms-Time: 1m24s456ms (Δ 1s)).

User input for trace graph: If one or two trace cursors are available, then you can move along the X axis.

Mouse: Drag the triangle that trace the cursor to other positions. Synchronously update the y value in the status bar.
Keyboard: Left and right arrow keys can move the black trace cursor.

Show/hide variables: If you don't want to view a certain variable, you can select the variable "Hide Variable" in the right-click pop-up menu of the variable in the variable list on the right, and the variable will not be displayed in the chart. If a variable has been hidden and you want to review the curve of the variable, you can select the variable in the variable list on the right and select "Show Variable" in the pop-up menu of the variable, and the variable will be displayed on

the chart again.

Reset view: According to the display mode of the time axis and Y axis configuration, the view will be restored to the default refresh state.

Multi-channel: In the trace view, use the first chart as a template to configure a chart for each variable and reinitialize the chart.

Note: The old chart configuration data will be deleted after using the "multi-channel" function.

Single channel: Trace view puts all variables in the first chart and reinitializes the chart.

Note: The old chart configuration data will be deleted after using the "single channel" function.

Save trace: save the read data to a *.cttrace file.

Load trace: read and load the saved data from the *.cttrace file.

M Instruction

CTH300(H56-10/H52-10) instruction

Bit logic instruction			
LD	Normally open contact load instruction LD SM0.0	ALD	AND load instruction ALD
A	Normally open contact AND instruction A SM0.0	OLD	OR load instruction OLD
O	Normally open contact OR instruction O SM0.0	LPS	Logic into stack instruction LPS
LDN	Normally close contact load instruction LDN SM0.0	LDS	Load stack instruction LDS 1
AN	Normally close contact AND instruction AN SM0.0	LRD	Logic read instruction LRD
ON	Normally close contact OR instruction ON SM0.0	LPP	Logic out stack instruction LPP
LDI	Normally open contact immediately load instruction LDI I0.0	=	Output instruction = Q0.0
AI	Normally open contact AND immediately instruction AI I0.0	=I	Immediately output instruction =I Q0.0
OI	Normally open contact OR immediately instruction OI I0.0	S	Set instruction S Q0.0 1
LDNI	Normally close contact	SI	Immediately set instruction

	immediately load instruction LDNI I0.0		SI Q0.0 1
ANI	Normally close contact AND immediately instruction ANI I0.0	R	Reset instruction R Q0.0 1
ONI	Normally close contact OR immediately instruction ONI I0.0	RI	Immediately reset instruction RI Q0.0 1
NOT	Logic stack top revere instruction NOT	AENO	Power flow AND instruction AENO
EU	Rising edge detection instruction EU	NOP	Null operation instruction NOP 1
ED	Falling edge detection instruction ED	--	--
Comparison instruction			
LDBx	Byte comparison (=,<,>,<=,>=,<>) load instruction LDB= 1, VB0	OBx	Byte operand comparison (=, <,>,<=,>=,<>) OR instruction OB= 1, VB0
LDWx	Integer comparison (=,<,>,<=,>=,<>) load instruction LDW= 10000, VW0	OWx	Integer comparison (=,<,>,<=, >=,<>) OR instruction OW= 10000, VW0
LDDx	Long integer comparison (=,<,>,<=,>=,<>) load instruction LDD= 100000, VD0	ODx	Double integer comparison (=,<,>,<=,>=,<>) OR instruction OD= 100000, VD0
LDRx	Floating number comparison (=,<,>,<=,>=,<>) load instruction LDR= 1.0, VD0	ORx	Floating number comparison (=, <, >, <=, >=, <>) OR instruction OR= 1.0, VD0
ABx	Byte operand comparison (=,<,>,<=,>=,<>) AND instruction AB= 1, VB0	LDSx	String comparison (=,<>) load instruction LDS="1234567890",VB0
AWx	Integer comparison (=,<,>, <=,>=,<>) AND instruction AW= 10000, VW0	ASx	String comparison (=,<>) AND instruction AS="1234567890", VB0
ADx	Double integer comparison (=,<,>, <=,>=,<>) AND instruction AD= 100000, VD0	OSx	String comparison (=,<>) OR instruction OS="1234567890",VB0

ARx	Floating number comparison (=,<,>, <=,>=,<>) AND instruction AR= 1.0, VD0	--	--
Transmit instruction			
MOVB	Byte move instruction MOVB 1 VB0	SWAP	High and low byte exchange instruction SWAP VW0
MOVW	Word move instruction MOVW 1000,VD0	BIR	Move byte immediately read instruction BIR IB0, VB0
MOVD	Double Word move instruction MOVD 100000,VD0	BIW	Move byte immediately write instruction BIW VB0, QB0
MOVR	Floating number move instruction MOVR 1.0,VD0	SRCP	Recipe to storage card
BMB	Block move byte instruction BMB VB0,VB100, 1	LRCP	Load recipe from storage card
BMW	Block move Word instruction BMW VW0,VW100, 1	SDL	Record data to storage card
BMD	Block move double Word instruction BMD VD0,VD100, 1	--	--
Integer calculate instruction			
+I	Integer addition instruction +I 10000, VW0	MUL	Integer multiply with double Integer MUL 10000, VD0
-I	Integer subtraction instruction -I 10000, VW0	DIV	Integer divide with double Integer DIV VW0, VD0
*I	Integer multiply instruction *I 10000, VW0	INCB	Byte increasing instruction INCB VB0
/I	Integer division instruction /I 10000, VW0	DECB	Byte decreasing instruction DECB VB0
+D	Double Integer addition instruction +D 100000, VD0	INCW	Integer increasing instruction INCW VW0
-D	Double Integer subtraction instruction -D 100000, VD0	DECW	Integer decreasing instruction DECW VW0
*D	Double Integer multiply instruction *D 100000, VD0	INCD	Long Integer increasing instruction INCD VD0
/D	Double Integer division	DECD	Long Integer decreasing

	instruction /D 100000, VD0		instruction DECD VD0
Float-point calculate instruction			
+R	Real addition instruction +R 1.0, VD0	COS	Cosine calculate instruction COS 1.0, VD0
-R	Real addition instruction -R 1.0, VD0	TAN	Tangent calculate instruction TAN 1.0, VD0
*R	Real multiplication instruction *R 1.0, VD0	LN	Nature exponential calculate instruction LN 1.0, VD0
/R	Real division instruction /R 1.0, VD0	EXP	Nature exponential calculate instruction EXP 1.0, VD0
SQRT	Square root instruction	PID	PID circuit calculate instruction PID VB0, 1
SIN	Sine calculate instruction SIN 1.0, VD0	--	--
Convert instruction			
BTI	Convert from byte to Integer BTI 1, VW0	DTR	Convert from Double Integer to real number DTR 100000, VD0
ITB	Convert from Integer to byte ITB 10000, VB0	DTS	Convert from Double Integer to string DTS 100000, VB0, 10
ITD	Convert from Integer to double Integer ITD 10000, VD0	ROUND	Carry integer instruction ROUND 1.0, VD0
ITS	Convert Integer to string ITS 10000, VB0, 10	TRUNC	Cut bit integer instruction TRUNC 1.0, VD0
DTI	Convert from double Integer to Integer DTI 100000, VW0	RTS	Convert from real to string RTS 1.0, VB0, 10
BCDI	Convert from BCD to Integer BCDI VW0	STI	Convert from string to Integer STI"1234567890", 5, VW0
IBCD	Convert from Integer to BCD IBCD VW0	STD	Convert from string to double Integer STD"1234567890", 5, VD0
ITA	Convert from Integer to ASCII ITA 10000,VB0,10	STR	Convert string to real STR"1234567890", 5, VD0
DTA	Convert from double Integer to ASCII DTA 100000, VB0, 10	DECO	Decode instruction DECO 1, VW0
RTA	Convert from real to ASTII	ENCO	Coding instruction

	instruction		ENCO 10000, VB0
ATH	Convert from ASCII to 16 system ATH VB0, VB100, 10	SEG	Segment instruction SEG 1, VB0
HTA	Convert from 16 system to ASCII HTA VB0, VB100, 10	--	--
Real-time clock			
TODR	Read real-time clock TODR VB0	TODRX	Read extend real-time clock TODRX VB0
TODW	Set real-time clock TODW VB0	TODWX	Set extend real-time clock TODWX VB0
Shift cycle instruction			
SLB	Left shift byte SLB VB0, 4	RLW	Left rotation Word RLW VW0, 8
SLW	Left shift Word SLW VW0, 8	RLD	Left rotation double Word RLD VD0, 16
SLD	Left shift double Word SLD VD0, 16	RRB	Right rotation byte RRB VB0, 4
SRB	Right shift byte SRB VB0, 4	RRW	Right rotation Word RRW VW0, 8
SRW	Right shift Word	RRD	Right rotation double Word RRD VD0, 16
SRD	Right shift double Word SRD VD0, 16	SHRB	Shifting register bit instruction SHRB I0.0, V0.0, 8
RLB	Left rotation byte RLB VB0, 4	--	--
Logic calculate instruction			
INVB	Reverse byte instruction INVB VB0	ORB	OR byte instruction ORB 1, VB0
INVW	Reverse Word instruction INVW VW0	ORW	OR Word instruction ORW 10000, VW0
INVD	Reverse double Word instruction INVD VD0	ORD	OR double Word instruction OD 100000, VD0
ANDB	AND byte instruction ANDB 1, VB0	XORB	XOR byte instruction XORB 1, VB0
ANDW	AND Word instruction ANDW 10000, VW0	XORW	XOR Word instruction XORW 10000, VW0
ANDD	AND double Word instruction ANDD 100000, VD0	XORD	XOR double Word instruction XORD 100000, VD0
Counter instruction			
CTU	Count-up instruction CTU C1, 10000	HDEF	HSC definition HDEF 0, 0

CTD	Count-down instruction CTD C1, 10000	HSC	HSC instruction
CTUD	Count down and up instruction CTUD C1, 10000	PLS	Pulse output instruction
Timer instruction			
TON	Open delay timer TON T37, 10000	BITIM	Read start timer BITIM VD0
TONR	Reserved open delay timer TONR T31, 10000	R_UTIM	Read current microsecond value instruction
TOF	Turn off delay timer TOF T37, 10000	CITIM	Calculate interval timer CITIM VD0, VD100
String instruction			
SLEN	Get string length instruction SLEN"1234567890", VB0	SCAT	Concatenation string instruction SCAT"1234567890", VB0
SCPY	Copy string instruction SCPY"1234567890", VB0	SFND	Seek string in string instruction SFND"12345678890", "321", VB0
SSCPY	Copy substring from string instruction SSCPY"1234567890", 1, 10, VB0	CFND	Seek a character in string instruction CFND"12345678890", "a", VB0
Table instruction			
FILL	Memory fill instruction FILL 10000, VW0, 10	FIFO	First in-first out instruction FIFO VW200, VW400
ATT	Add to table instruction ATT 10000, VW0	LIFO	Last in-first out instruction LIFO VW200, VW400
FNDx	Seek table (=, <, >, <=, >=, <>) instruction FND=VW0,9999,VW1000	--	--
Interrupt instruction			
CRETI	Return from interrupt routine instruction	ATCH	Add interrupt instruction
ENI	Start interrupt instruction ENI	DTCH	Separate interrupt instruction
DISI	Forbid interrupt instruction DISI	CEVNT	Clear interrupt event instruction CEVNT 10
Program control instruction			
FOR	FOR-NEXT start loop FOR-NEXT FOR VW0, 1, 10 NEXT	LSCR	Load sequence control interrupt

NEXT	FOR-NEXT loop end	SCRE	Sequence control relay end
JMP	Jump to label instruction	SCRT	Sequence control relay transfer
LBL	Label instruction	CSCRE	Conditional Sequence control relay end
WDR	Monitor reset instruction	CRET	Return from subroutine with condition instruction
DLED	Diagnosis LED instruction DLED 1	END	Conditionally end
CALL	Call subroutine instruction	STOP	Stop instruction
Communication instruction			
XMT	FPORT send instruction	SPA	Set port instruction
RCV	FPORT receive instruction	GPA	Get port address instruction
NETR	Network read instruction	EBUSR	Read module information instruction
NETW	Network write instruction	EBUSW	Write module information instruction
UDPNETR	Ethernet read function	EBUSGETDIA	Get module internal diagnosis information instruction
UDPNETW	Ethernet write function	EBUSSNDCMD	Send module operation command instruction
MBSNDMSG	Send ModBus message	--	--

< note > please refer to the *magicworks PLC user manual* for detailed instructions of the above instructions, which can be downloaded free from the website of COTRUST company: http://www.co-trust.com/Download/index.html

N Product Oder Info.

List of ordering information

Specification	Order number
CPU	
H56-10, motion controller, 256KB+64KB program space, 1MB data space, 24V DC power supply, 10 digital inputs, 6*500KHz high-speed counter, 1 EtherNET interface, 1 EtherCAT interface, 1 CAN Interface, 1 RS485 interface, 1 USB interface. support single-axis motion control, interpolation function, electronic cam and electronic gear and other functions, support C language programming.	CTH3 H56-100S2
H52-10 motion controller, 256KB+64KB program space, 1MB data space, 32K data block space, power-down retention. 24V DC power supply, 10 digital inputs, 6*500KHz high-speed counter, 1 EtherNET interface, 1 EtherCAT interface, 1 CAN interface, 1 RS485 interface, 1 USB interface. support single-axis control functions (such as positioning, speed and return to the original), support linear/arc interpolation function, support continuous interpolation function, Support functions such as electronic cams and electronic gears, and support C language programming.	CTH3 H52-100S2
CTH300 module	
PWR-02, power module, input 85~264V AC, output 24V DC/2A	CTH3 PWR-020S1
INT-00, Intermediate extension module	CTH3 INT-000S1
CAN-1M master communication module,1 can communication port	CTH3 CAN-1M0S1
Hsc-02 high speed counting module, 2-way differential / single ended signal input	CTH3 HSC-020S1
HSP-04 pulse output module, 4 differential/single-ended signal output	CTH3 HSP-040S1
DIT-08 digital module, digital input 8 x 24VDC	CTH3 DIT-080S1
DIT-16 digital module, digital input 16 x 24VDC	CTH3 DIT-160S1
DIT-32 digital module, digital input 32 x 24VDC	CTH3 DIT-320S1
DQT-08 digital module, digital output 8 x 24VDC	CTH3 DQT-080S1
DQT-16 digital quantity module, digital output 16 x 24VDC	CTH3 DQT-160S1
DQT-32 digital quantity module, digital output 32 x 24VDC	CTH3 DQT-320S1
DQR-08 digital quantity module, digital output 8 x relay	CTH3 DQR-080S1
DQR-16 digital module, digital output 16 x relay	CTH3 DQR-160S1
AIS-04 analog module, analog voltage and current input, 4AI x 12bits	CTH3 AIS-040S1
AMS-06 analog module, analog voltage and current input and output, 4AI x 12bits, 2AQ x 12bits	CTH3 AMS-060S1
AIV-08 analog module, analog voltage input, 8AI x 16bits	CTH3 AIV-080S1
AIC-08 analog module, analog current input, 8AI x 16bits	CTH3 AIC-080S1
AQS-04 analog module, analog voltage and current output, 4AQ x 12bits	CTH3 AQS-040S1
AQS-08 analog module, analog voltage and current output, 8AQ x	CTH3 AQS-080S1

12bits	
AIT-04 temperature module, thermocouple input module, 4 TC, isolated 16bits accuracy	CTH3 AIT-040S1
AIT-08 temperature module, thermocouple input module, 8 TC, isolated 16bits accuracy	CTH3 AIT-080S1
AIR-04 temperature module, thermal resistance input module, 4 RTD, isolated 16bits accuracy	CTH3 AIR-040S1
AIR-08 temperature module, thermal resistance input module, 8 RTD, isolated 16bits accuracy	CTH3 AIR-080S1
Profinet slave module	CTH3 PNT-000S1
Accessories	
PLC lithium battery, voltage 3.6V, capacity 2700mAh	CTH3 BAT-000S1

Service Hotline

+86 400-700-4858

Address: 9/F, block a, building 6, international innovation Valley, Nanshan District, Shenzhen

0755-86226822

market@co-trust.com

www.co-trust.com

Scan for more info