COTRUST
合信技术

CTH300 C5X-10 motion controller User Manual

❍ Version: V1.00

❍ Release date: 10/2022

Copyright Notice

Copyright Notice

Trademark statement

 ® 、TrustPLC®、MagicWorks®、COTRUST®、MagiCampus®、COMOTION®、  ® are registered trademarks of COTRUST TECHNOLOGIES CO., LTD. or its affiliates, and are protected by law, and infringement must be investigated. Without the written permission of COTRUST TECHNOLOGIES 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. COTRUST TECHNOLOGIES CO., LTD. has tried its best to ensure that the contents are accurate and reliable when compiling this manual. However, COTRUST TECHNOLOGIES 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 C5X-10 products of COTRUST TECHNOLOGIES CO., LTD.!

Before using CTH300 C5X -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 C5X-10 motion controller can be combined with a variety of expansion modules or units to form a powerful programmable logic controller system. Compared with the previous CTH300-C series PLC, the C5X-10 motion controller adds digital input, high-speed counter and other functions while inheriting the advantages, which greatly improves the performance of COTRUST medium-sized PLC.

Suitable

This manual is designed for engineers, installers, maintenance personnel and electricians with common sense of automation. Provide installation and debugging information about CTH300 C5X-10 motion controllers.

Service support

COTRUST TECHNOLOGIES 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 COTRUST TECHNOLOGIES CO., LTD. https://www.co-trust.com or official WeChat.

Safety Precautions

Before starting to operate the product, please read the precautions in the user manual carefully to avoid accidents.

The personnel responsible for the installation and operation of the product must undergo strict training, abide by the safety regulations of the relevant industry, strictly abide by the relevant equipment precautions and special safety instructions provided in this manual, and operate the equipment according to the correct operation method.

In order to prevent harm to people and damage to property, the following items must be observed. Please refer to the relevant symbol description for the possible harm and damage degree caused by the wrong use of this product.

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
Oct.2022	V1.00	First edition release

Content

Product Overview

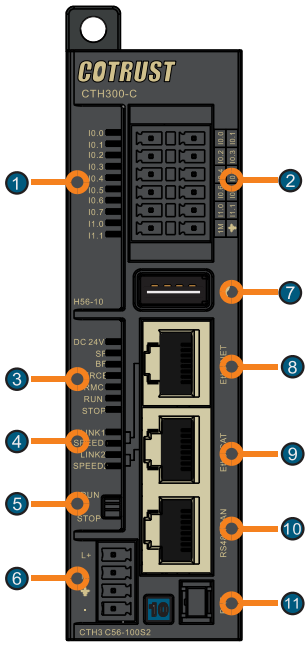
1

1.1 Product Description

The CTH300 C5X-10 motion controller can be combined with a variety of expansion modules or units to form a powerful programmable logic controller system. Compared with the previous CTH300-C PLC, the C5X-10 motion controller adds digital input, high-speed counter and other functions while inheriting the advantages, which greatly improves the performance of COTRUST medium-sized PLC.

1.1.1 CPU Introduction

The following is a schematic diagram of the appearance of CTH300 C5X-10:

CTH300 C5X-10 Motion Controller		
	1	DI indicator
	2	DI interface
	3	LED indicator light
	4	Network port communication indicator
	5	System operation switch
	6	Power terminal
	7	USB interface
	8	EtherNET port
	9	EtherCAT port
	10	RS485/CAN port
	11	External battery interface

- **High calculation speed**

The bit instruction execution speed of the CPU is 0.015μs/step.

Floating t instruction execution speed is 1μs/step

- **High-speed transmission**

The high-speed bus between CPU and expansion module adopts M-LVDS technology, and the data transmission rate reaches 55Mbps. The CPU integrates 10 digital inputs and supports 6 high-speed counters.

- **Powerful system expansion capability**

Through INT-00, a maximum of 32 I/O modules (4 racks, 8 modules per rack) are allowed to be expanded, and a maximum of 1032 I/O modules (128 ECTs, each ECT) are allowed to be expanded through ECT-00. Expandable up to 8 modules).

- **Supports multiple communication protocols**

The CPU natively supports communication protocols such as EtherNET, EtherNET/IP, EtherCAT, CANopen, and Modbus.

- **Programming Software**

CODESYS V3.5 SP11 is used for programming, and the CPU has its own USB port host device interface.

◆ **Perfect operation control function**

Supports single-axis motion function, electronic cam and electronic gear function SoftMotion commands and CNC functions that conform to the PLCopen standard.

◆ **Support external battery**

C5X-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

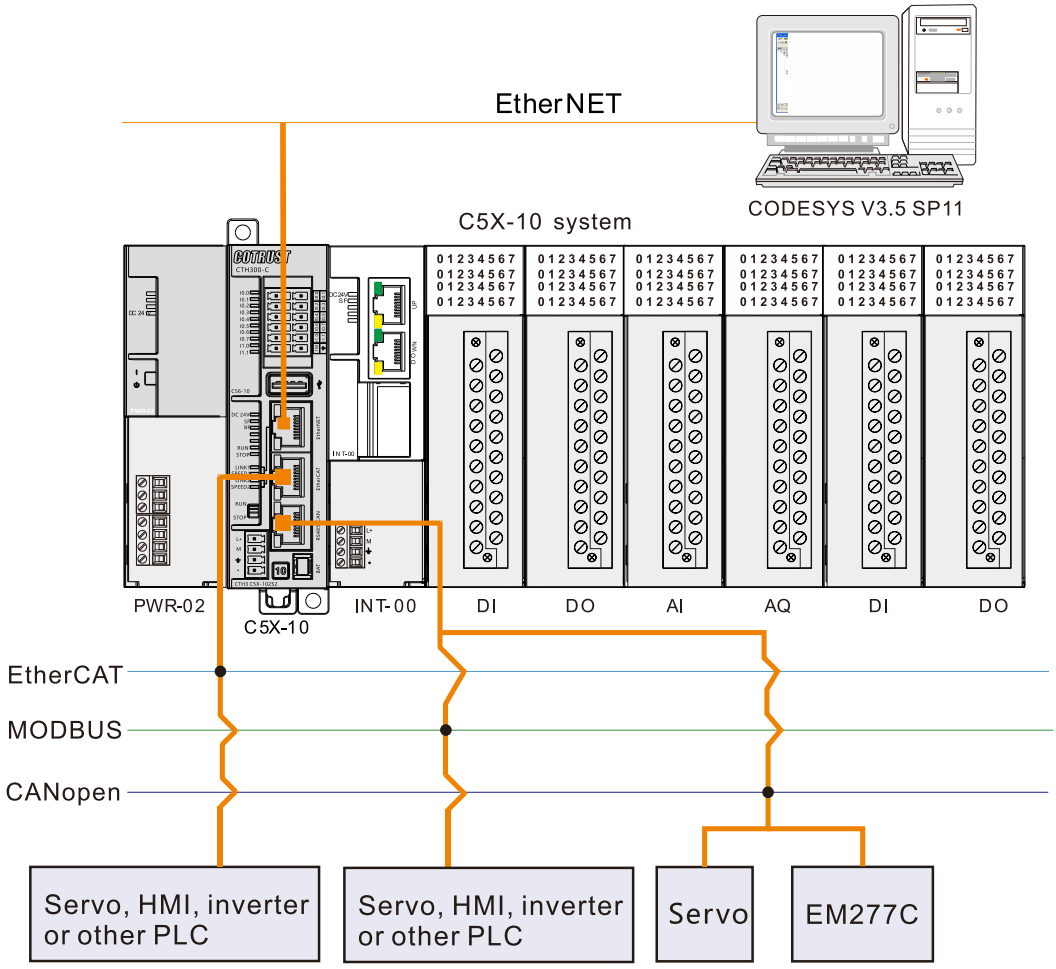
C5X-10 motion controllers belong to the CTH300 of medium-sized PLC family, which support CTH300 expansion modules, and their module types include: digital input and output, analog input and output, temperature acquisition, high-speed counting, CAN communication module, etc., Please refer to the table 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 CTH3 HSP-040S1
	Power module CTH3 PWR-020S1		

	Intermediate expansion module CTH3 INT-000S1		EtherCAT slave module CTH3 ECT-000S1	

1.2 System Structure



- C5X-10 supports RS485, EtherCAT, EtherNET, EtherNET/IP, CANopen and other communication methods, and can realize CANopen communication through the CAN communication port of the CPU body (as shown in the figure above).
- C5X-10 can communicate with the host computer through the EtherNET communication port (using a standard network cable) or the RS485 communication port.
- The application system consists of a central rack and three expansion racks at most, and a maximum of 32 expansion units can be installed.

- C5X-10 support multiple communication methods such as RS485, EtherCAT, EtherNET, EtherNET/IP, CANopen, etc.

1.3 Electrical Specifications and Environmental Conditions

Electromagnetic compatibility (EMC) refers to the ability of electrical equipment to function properly in its electromagnetic environment without interfering with the environment. Table 1-2 and Table 1-3 describe the electrical specification environmental condition standards that the C5X-10 should comply with. 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 C5X-10 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

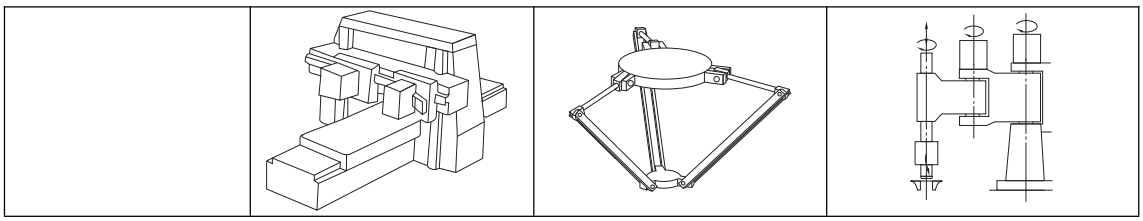
1.4 Functional application

C5X-10 motion controller has two control functions: motion control function and logic control function.

1.4.1 Motion control function

Table 1-4 Motion Control Functions

PLCopen-based single-axis control/master-slave control			
Single axis	Homing, relative/absolute positioning, position/velocity/acceleration trajectory planning, continuous motion, JOG		
Master-slave axis	Electronic gear, electronic cam, phase shift		
CNC function (only C57-10 supports)			
Standard CNC functions	3-5 axis linkage, GCODE import, coordinate transformation, etc.		
	Gantry(Gantry workbench)	Parallel(parallel robot)	Scara (Choose compliance to equip the robotic arm)



1.4.2 Logic control function

3 Program Organization Units

- ❒ Program
- ❒ Function Block
- ❒ Function

6 programming languages

- ❒ IL
- ❒ ST
- ❒ SFC
- ❒ FBD
- ❒ LD
- ❒ CFC

Standardized PLC functions

Type conversion, numeric function, arithmetic function, shift function, Boolean operation function, selection function, comparison function, string function, timer, counter, edge detection, bistable element

Getting Started

2

Based on an example, this paper introduces how to create a simple project including a PLC program, download the program to the target device (C56-10 motion controller), run and monitor the program.

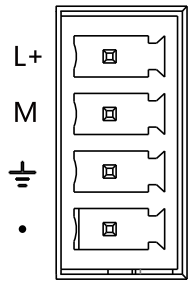
The sample program is written in LD language and includes a program: PLC_PRG. PLC_PRG contains two timers, of which timer 1 starts timing when the program starts, and is set after 5 seconds. Timer 2 starts timing after timer 1 is set, and timer 2 starts to reset after 5S. Then Timer 1 and Timer 2 are set and reset repeatedly in a 5-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**

POWER terminal of PLC:



Warning

Before installing or removing C5X-10 motion controllers, corresponding safety protection specifications must be observed and their power supply must be disconnected.

❑ **Connecting cable**

Take C56-10 as an example, connect the Ethernet communication port between the programming device and C56-10 with a standard network cable, or connect the RS485 communication port between the programming device and C56-10 with a programming cable.

Remark> Please refer to Section 5.3.3 Make standard network cables Cables to make standard network cables.

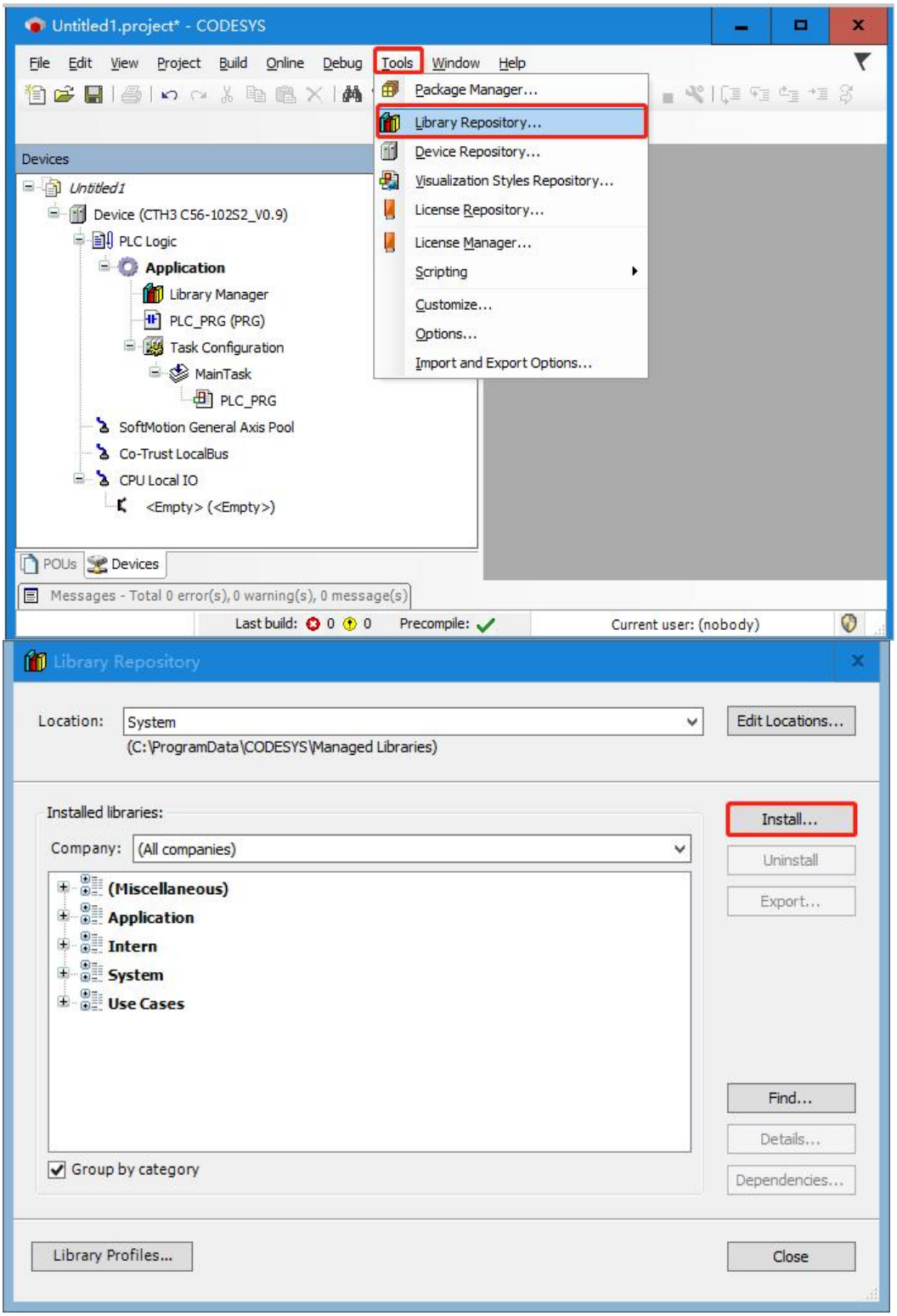
2.2 Start the CODESYS software

Start the CODESYS software and perform the following steps:

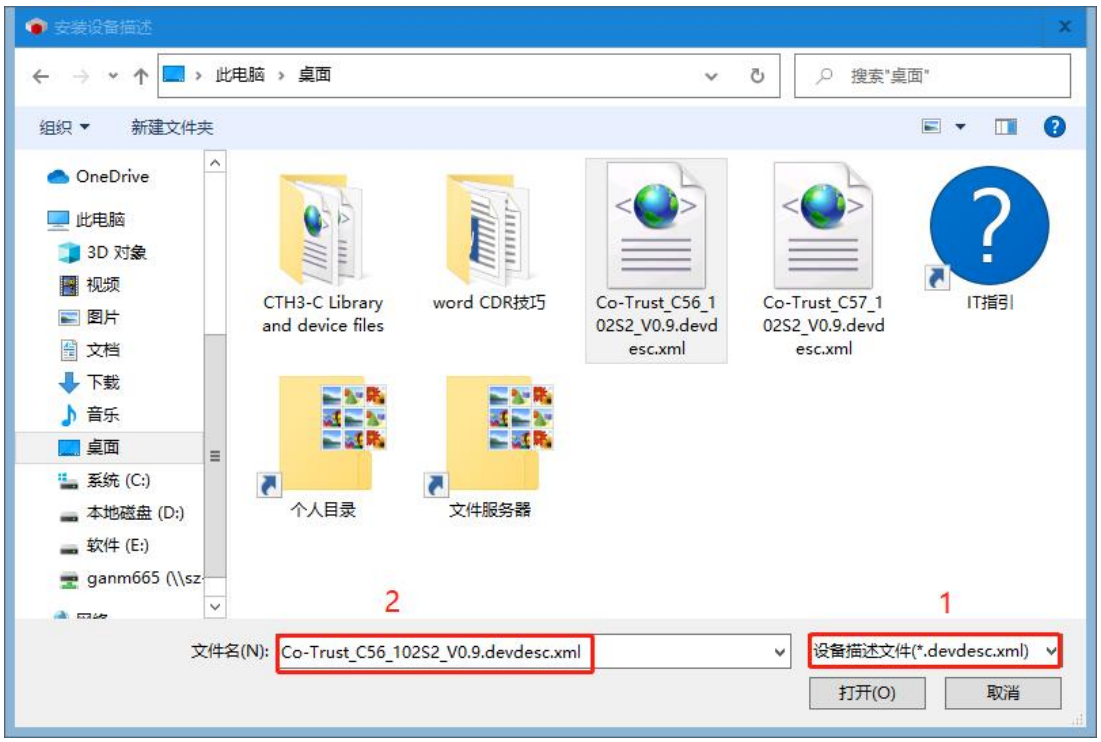
1. Install the COTRUST device description file (Co-Trust_C56_102S2_V0.9.devdesc.xml)

First, you need to install the COTRUST device description configuration file (Co-Trust_C56_102S2_V0.9.devdesc.xml). After the installation is successful, you can use the COTRUST device in the program system. The specific operations are as follows:

❒ Select the menu item "Tools" → "Device Repository" to open the following dialog box, which lists the device descriptions in the system, click "Install" in the dialog box.

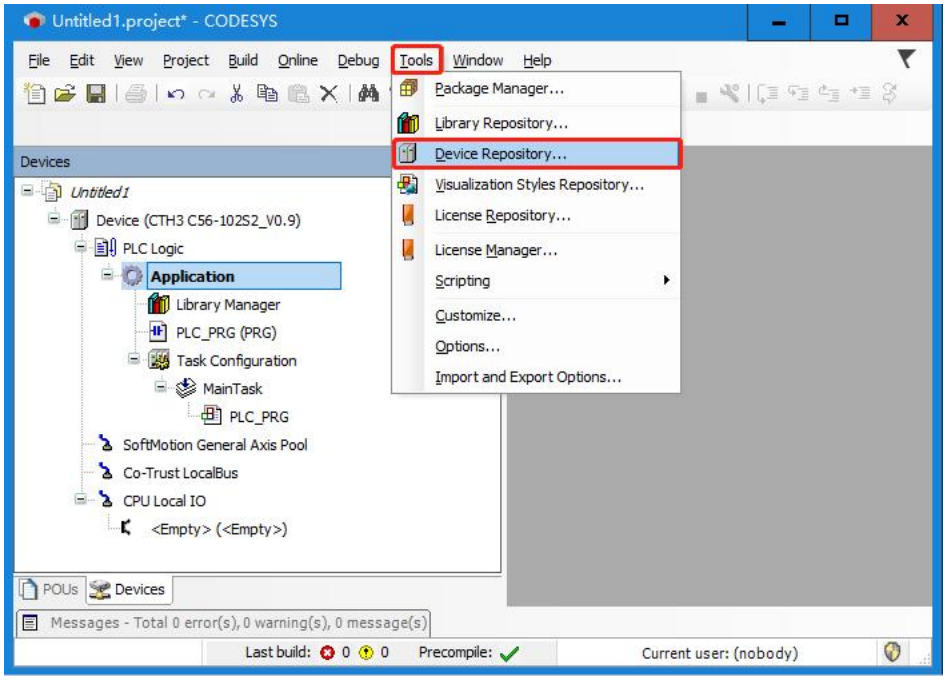


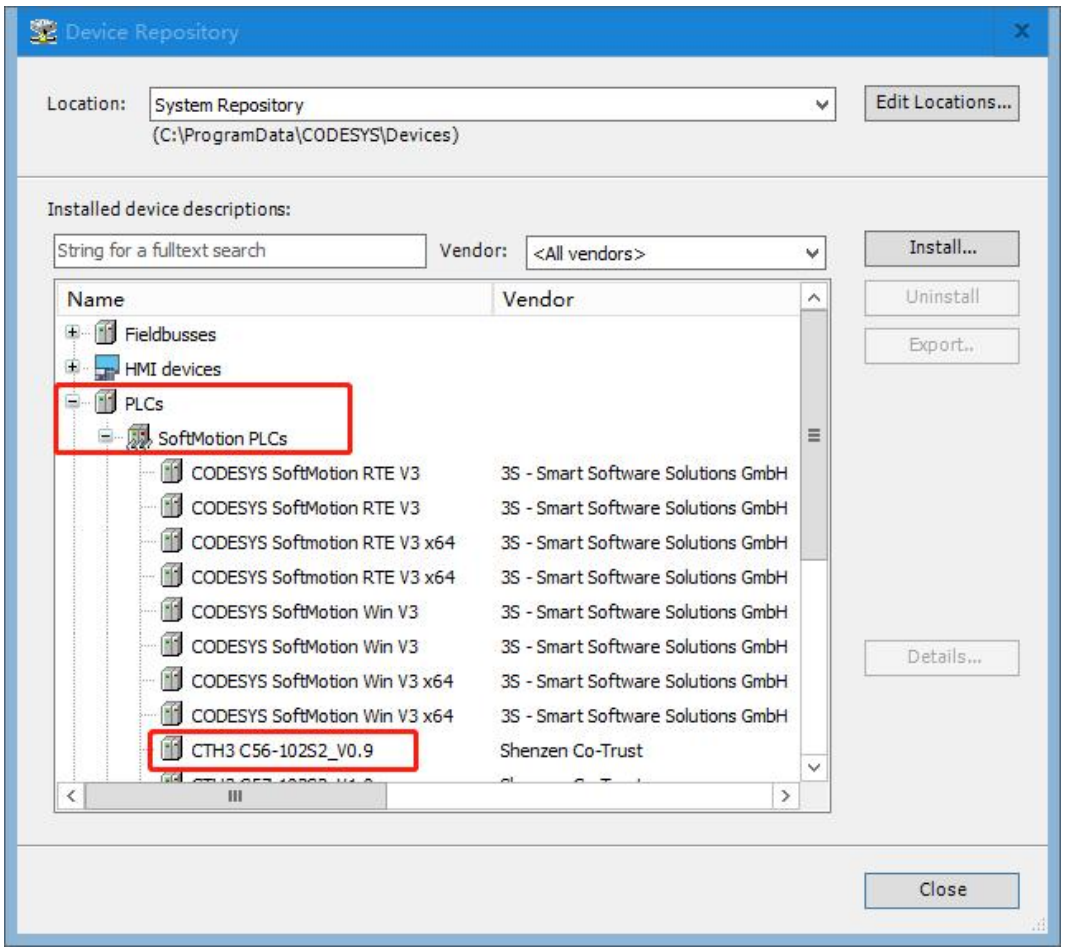
❒ First select the file format, the format is "device description file (*.devdesc.xml)" and then select the desired Co-Trust_C56_102S2_V0.9.devdesc.xml, click the "Open" to confirm the option, the new device will be added to the "Device" Libraries" in the device directory. The profiles provided by COTRUST can be selected by setting the corresponding filters.



After the above operations are completed, you can refer to the following steps to check whether the files are installed correctly:

Select "Tools" → "Device repository", as shown below:





If you continue to install the device files, please uncheck "Enable Auto Connect Dialog" in "Tools" → "Options".

提示

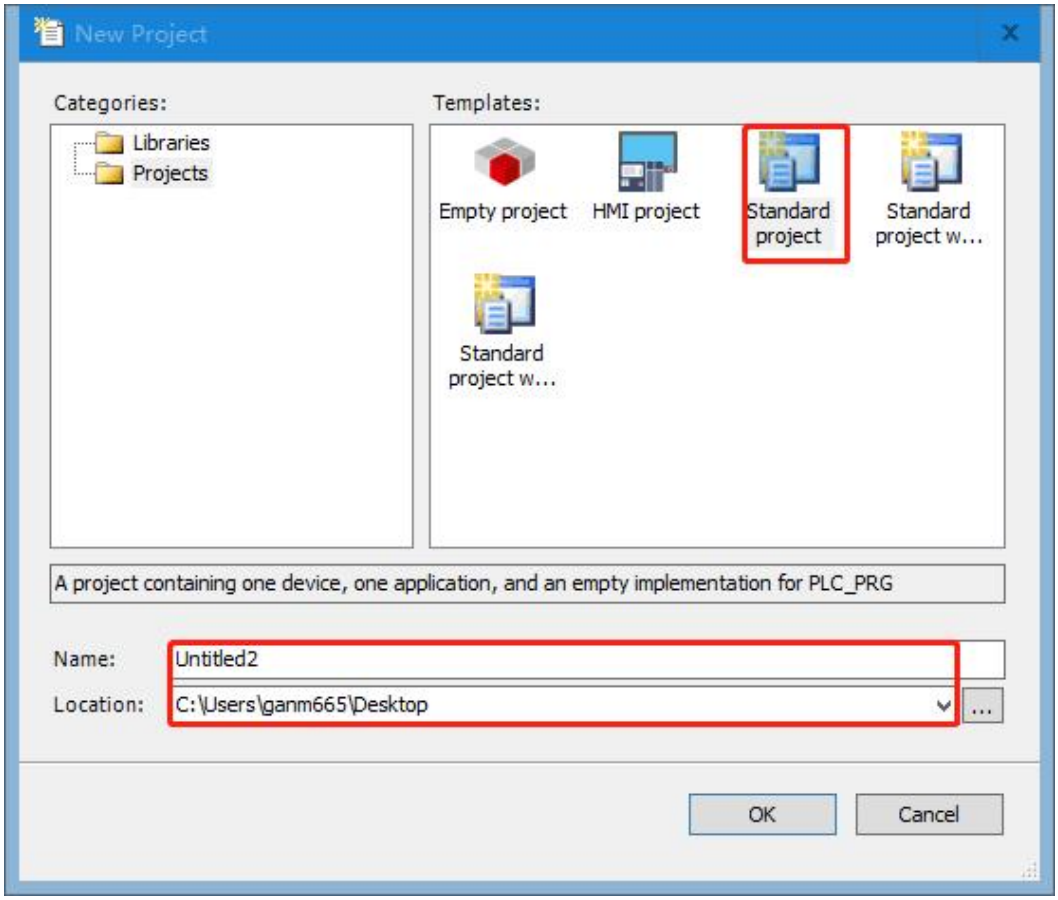
1) If you need other COTRUST devices (CANopen or EtherCAT slave devices), you need to install the following device description files:

- CANopen slave device configuration file (*.eds): Co-Trust E10.eds, Co-Trust EM277C.eds, A4S-CAN V003.eds;
- EtherCAT slave device configuration file (*.xml): COTRUST_A4N_ V1.3.xml;

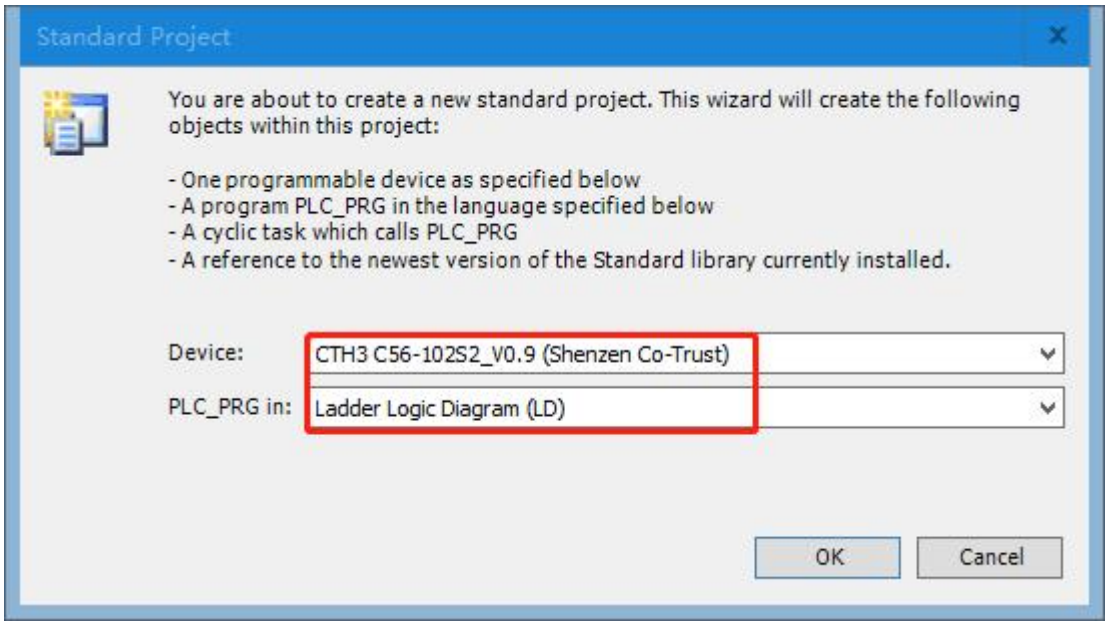
2) Please download the required device configuration file from our website: http://www.co-trust.com。

2. Create a new project

（1）(1) Select the menu item "File" → "New Project", then select "Standard project" (standard project) and set the file name and storage directory).



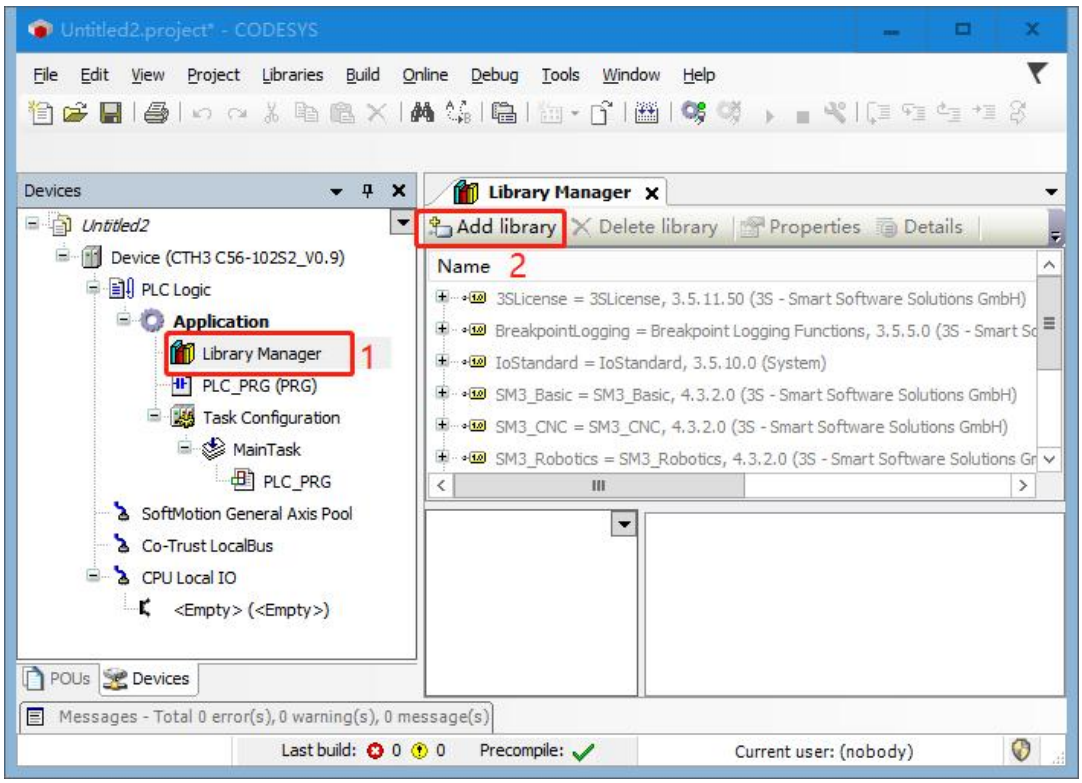
（2）(2) Then select the desired file device (CTH3 C56) and programming language, and click "OK" to complete the creation of the project.



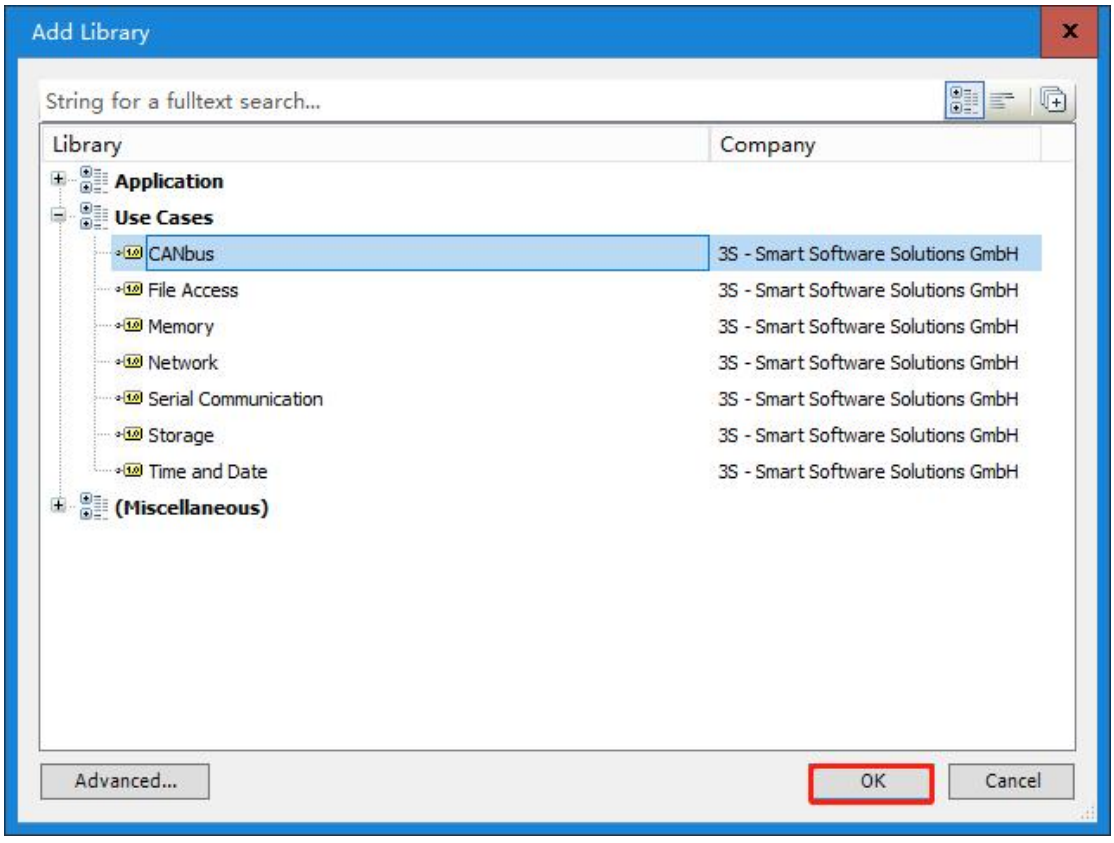
3. Add library

Users can add instruction library in CODESYS according to their specific needs. The specific operations are as follows:

- Open the Library Manager in the project tree and select Add Library.



◆ In the displayed interface, select the library you want to add and click "OK" to add the required library to the library manager.



If CODESYS is installed and prompted that there is no library that comes with CODESYS.

Make sure that the computer is connected to the external network, and click "DownLoad missing library". After the download is successful, there will be a download completion prompt behind the corresponding library.

2.3 Set up communication

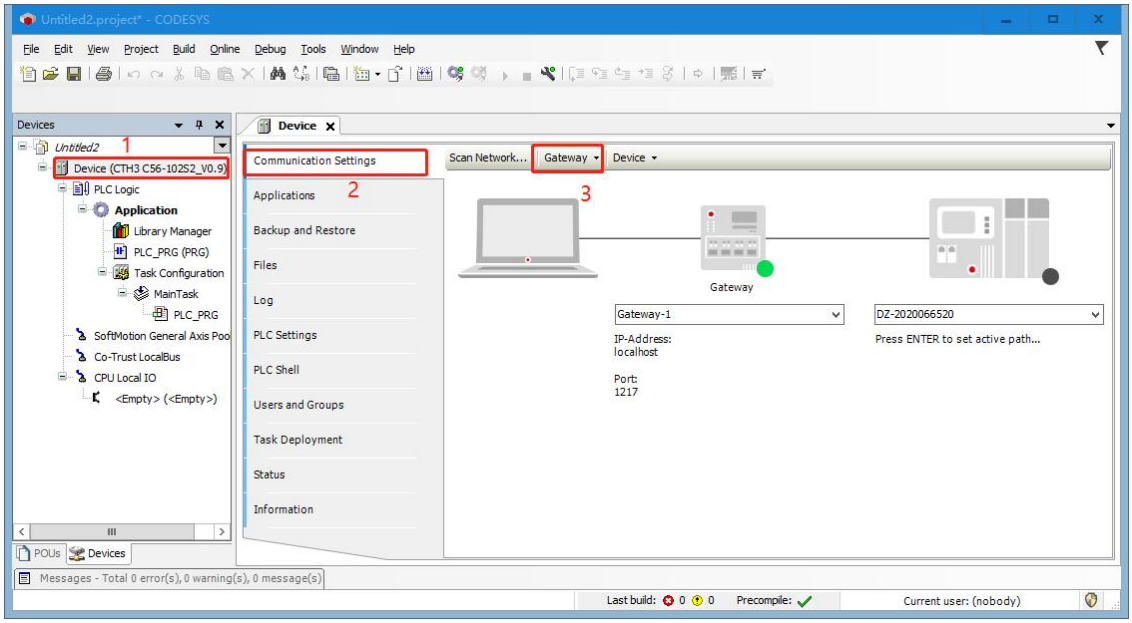
Please refer to the following steps to set up the communication between C56-10 and the host computer:

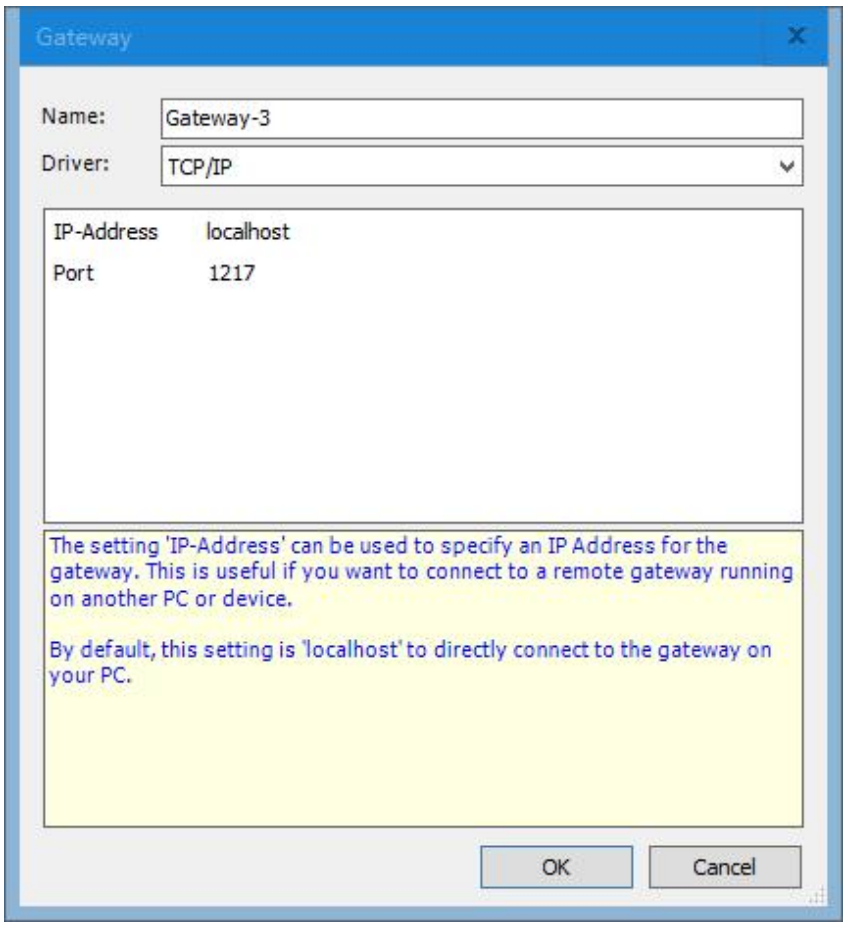
1. Set the IP of the programming device to the same network segment as the C56-10

Before setting the communication, it is necessary to set the IP of the programming device PG/PC to the same network segment as the C56-10 (IP: 192.168.0.x). Setting method: open the local connection properties of the PC, double-click the TCP/IP protocol, Change "Obtain an IP address automatically" to "Use the following IP address", and then fill in "192.168.0.X" in the IP address.

2. Execute "Communication Settings" in the CODESYS device view

Double-click "Device (CTH3 C56-10-10S2_V0.9)" in the device view to open the device dialog box, then click the "Gateway" button in the "Communication Settings" tab, then click "Add Gateway", in the pop-up "Gateway" "In the dialog box, enter "Name", "Driver" and select "TCP/IP", "IP Address" and select "localhost", and finally click "OK" to close the dialog box, C56-10 is added to the communication dialog box.





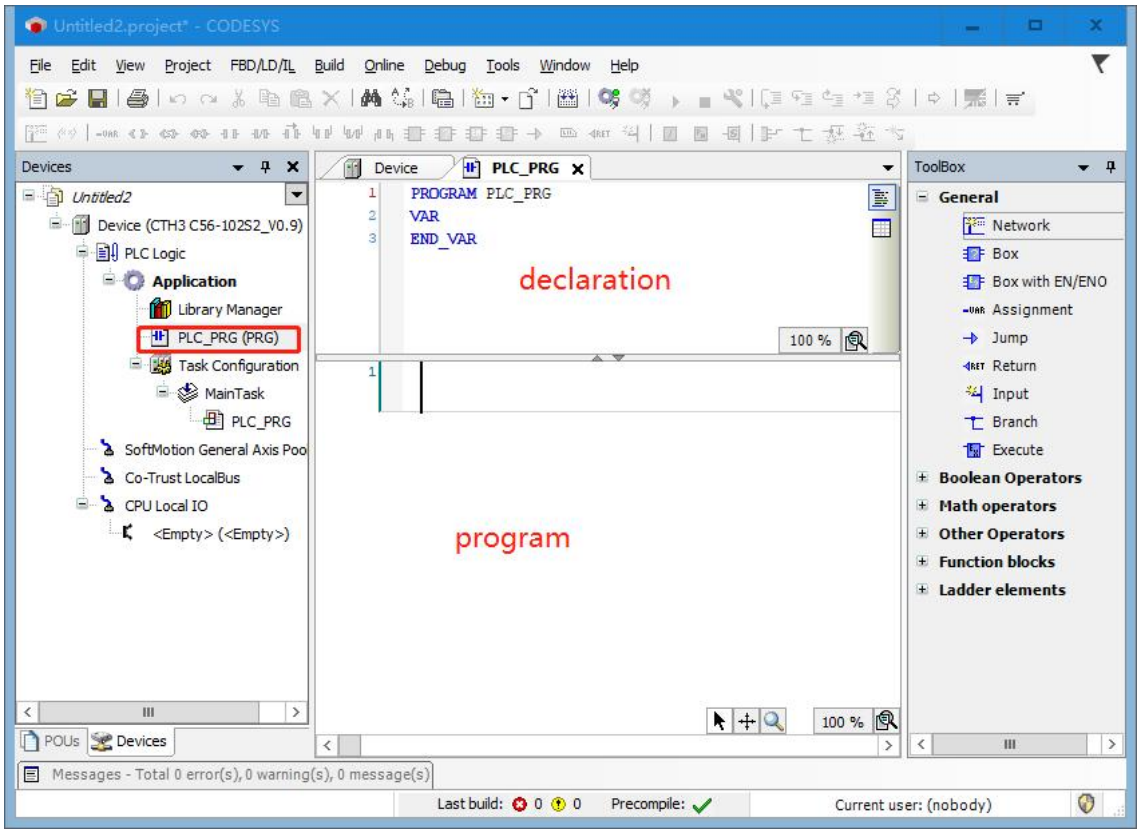
Select the successfully added C56-10, and then click "Scan Network" to search for available devices on the local network. If the search is successful, select the searched device and click "Set Active Path", this operation will activate the communication channel setting, that is, all communication-related operations will be associated with this channel.

<Remarks> When the system starts, CODESYS related service programs (such as , , etc.) will appear in the system tray. If there are no special requirements, there is no need to operate the service programs in the tray.

2.4 Programming

This program realizes the function: Timer 1 and Timer 2 cycle setting and reset operations in 5S intervals.

In the device window, the default POU is "PLC_PRG". Double-click "PLC_PRG" to open the LD language editor. The LD language editor includes a declaration part and an implementation part.

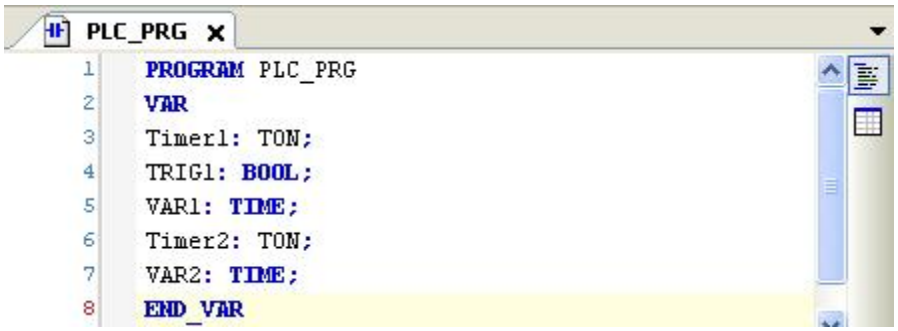


The declaration part includes: the line number, POU type and name (eg "PROGRAM PLC_PRG") displayed in the left border, and variable declarations between the keywords "VAR" and "END_VAR".

The implementation section shows a contact and an output coil.

1. Declare variables in PLC_PRG

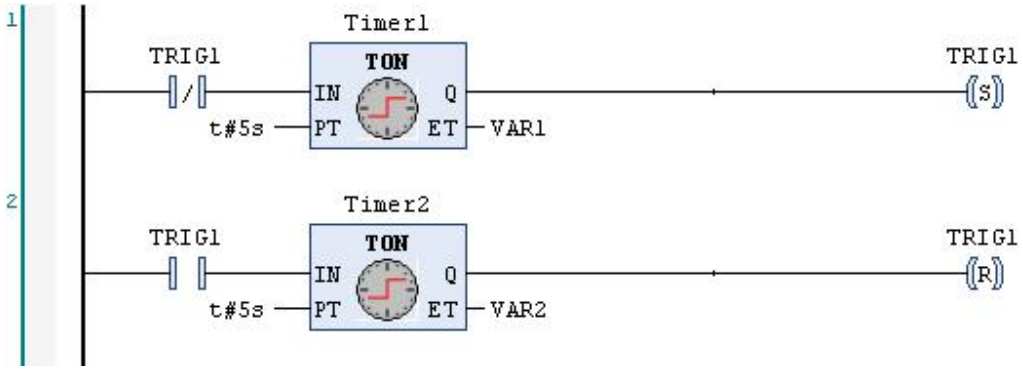
In the Declarations section of the editor, move the cursor to VAR, hit Enter, insert a new blank line, and declare the variables you need to use. Or in the implementation part of the program, use the automatic declaration function: enter the command in the implementation part of the program, click Enter, if there are undeclared variables in the new line, the system will open the automatic declaration dialog box, where you can make declaration settings. The declaration part of this example is as follows:



2. Enter commands in the implementation part of PLC_PRG

Expand "Ladder Diagram Elements" in the toolbox on the right side of the LD language editor, drag "Timer TON" to the implementation part of the language editor, and then drag a reset coil to

the back of the TON instruction output point Q. Using the same operation, create a TON instruction in network 2.



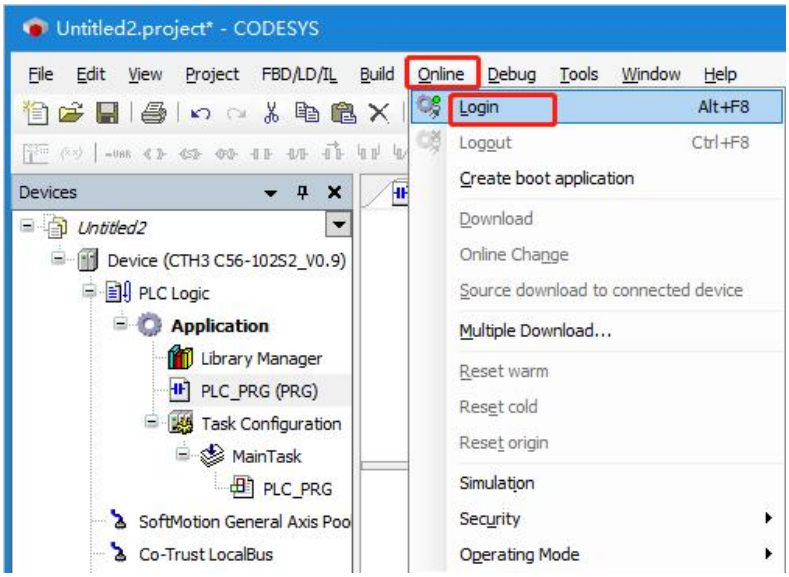
2.5 Compile and run the program

1. Save the current project and compile it

After the program is written, save the project, and then select "Compile" to check the syntax of the current object. When the syntax check is completed, any error messages and warnings will be displayed in the message window of the "Compile" class. If there are no errors, the compilation is successful.

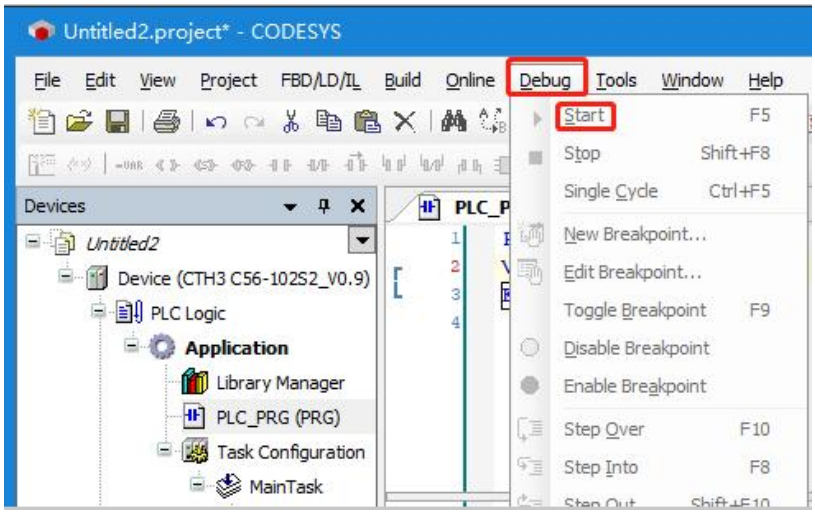
2. Login, download and start the program

Select "Online" → "Login" to establish a connection between the application and the C56-10, and enter the online state.



If communication has already been set up, a warning will appear, Click "Yes" to start the compilation and download of the application.

Finally, "Debug" → "Start" to make the application program in C56-10 start to run, and then you can monitor and debug the current project.



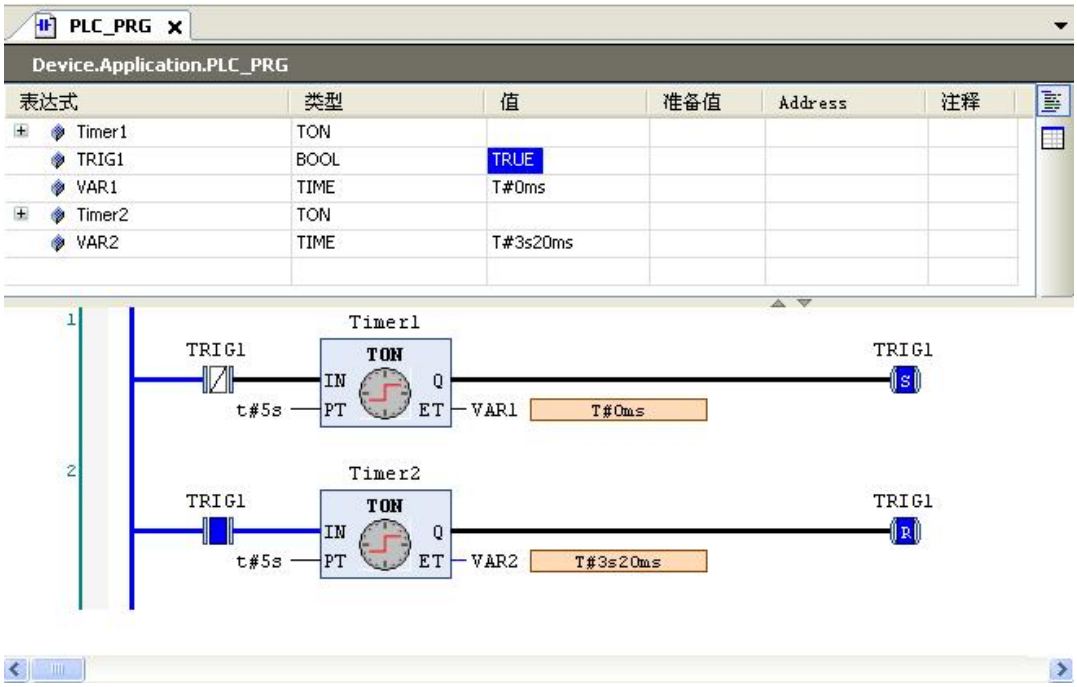
2.6 Monitor and debugg

There are three ways to monitor variables in your application:

- Watch window with defined watch list
- Write and force variables
- Online view of special POUs

1. Open the window of the program

Double-click "PLC_PRG", the following online view will appear: the upper part corresponds to the program main body of PLC_PRG to realize the partial view, and the actual value is displayed by the internal monitoring window behind each variable.



2. Write and force variables

By writing or mandatory, a "preparation value" is assigned to a variable TRIG1. Starting in the next cycle, the variable is the value. In the "preparation value" input, the integer value is required, enter or click the outside of the area, and then execute the command "writing value" or "forced value" to force the value to PLC.

3. Use the monitor window

Select "View" → "Monitor" → "Monitor 1" to open the monitoring window. Then, the mouse click the first line of the expression, open the editing box, and enter the complete path of the variable TRIG1 to be monitored: "Device.Application.PLC_PRG.TRIG1", and then the variables can be written and force value.

Installing

3

3.1 Installation Precautions

You can use the mounting holes to fix the C5X-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 C5X-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 C5X-10 motion controller.

When installing the C5X-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.

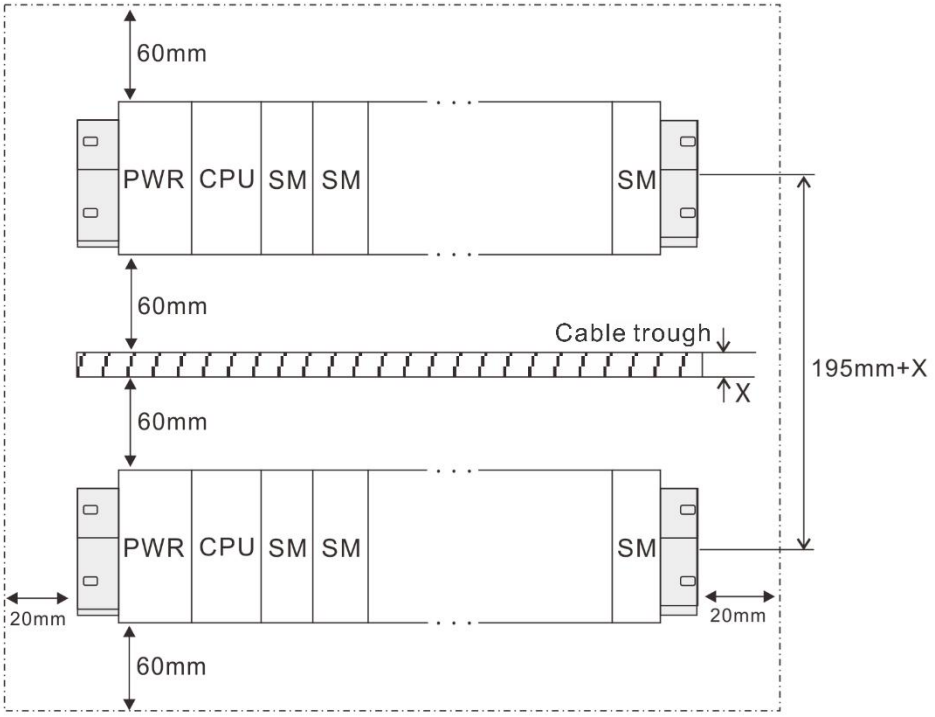


Figure 3-1 Installation diagram

❑ 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 C5X-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 C5X-10 motion controller, be sure to use the correct or equivalent module. When replacing the C5X-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 C5X-10 motion controller may produce incorrect functions.

● Failure to use the same module to replace the C5X-10 motion controller in the same direction and sequence may cause serious personal injury and equipment damage.

3.3 Use of the Rack

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

The C5X-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.

❑ Module layout on the rack

The rack is an assembly rail. This guide rail can be used to install the module to which the C5X-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 C5X-10 backplane bus must not exceed 1.6A.

The following figure shows the layout of eight signal modules in the C5X-10 system.

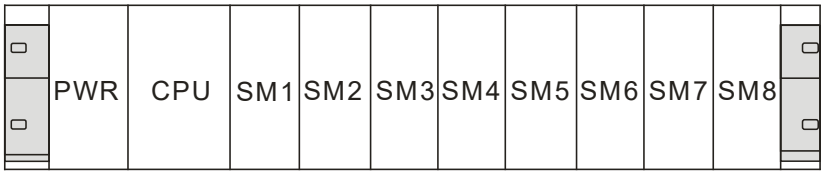


Figure 3-3 C5X-10 application system layout

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 C5X-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 C5X-10 system on 4 racks.

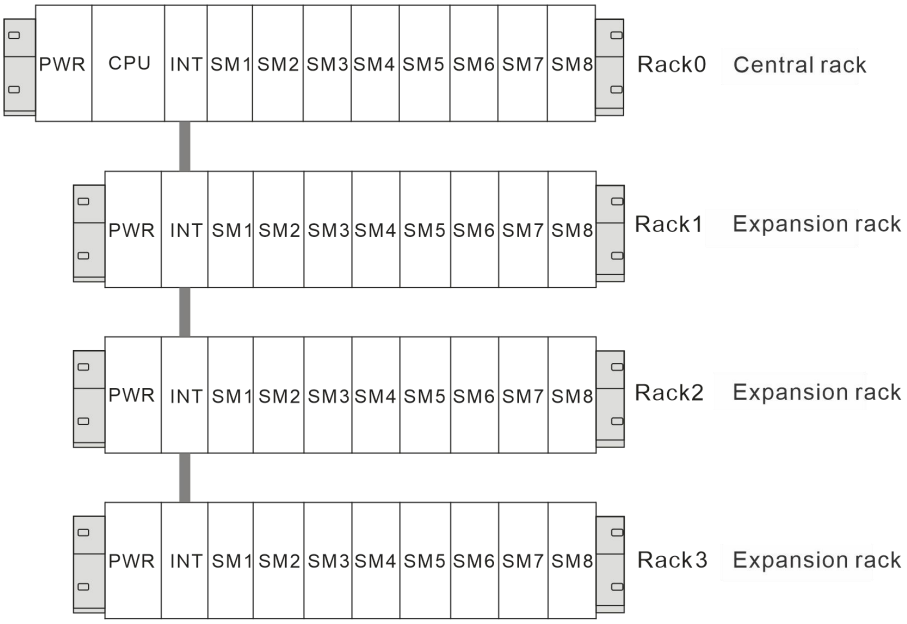


Figure 3-4 C5X-10 application system module layout with 4 racks

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.

3.2 Installation Dimension

C5X-10 motion controllers have mounting holes, which can be easily installed on the back panel. The installation dimensions are shown in figure 3-2.

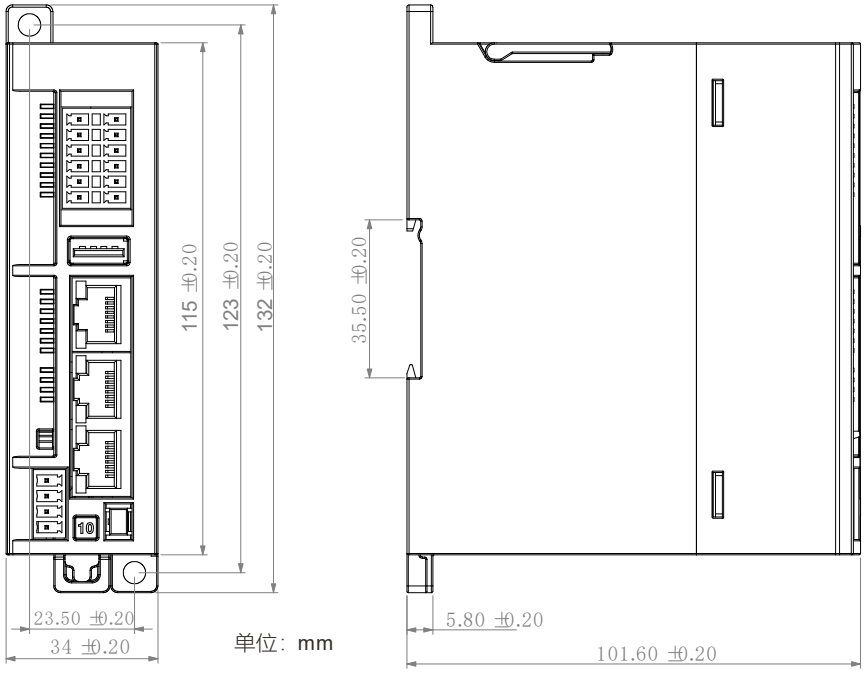


Figure 3-2 C5X-10 motion controller installation dimensions

3.4 Installation Method

Installation mode

C5X-10 motion controller can be installed on the back plate of control cabinet or standard DIN rail. It can be installed horizontally or vertically.

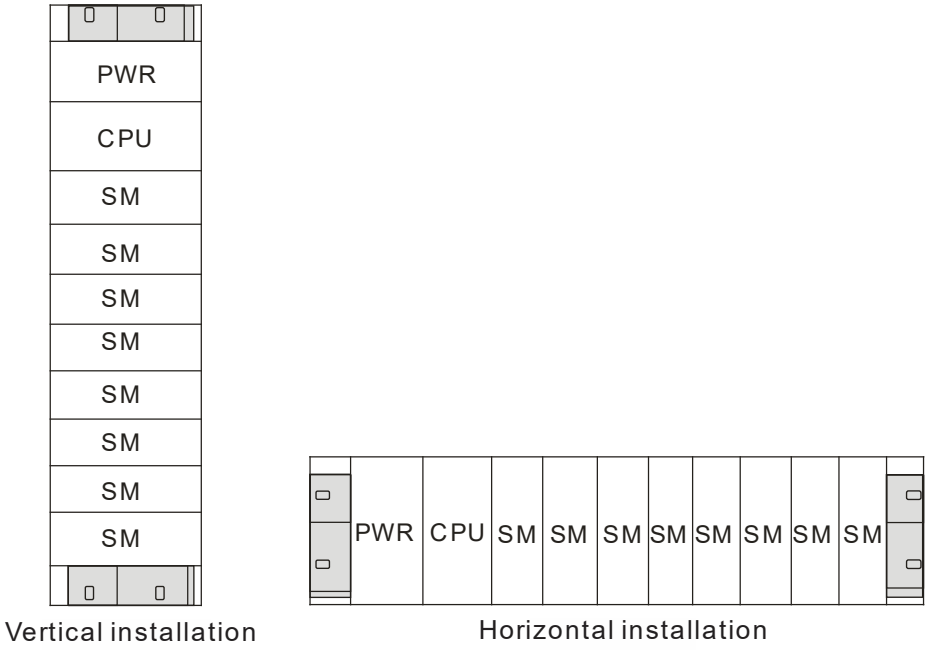


Figure 3-5 CPU and module installation method

Installation and removal operations

Please install or remove the C5X -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 C5X-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 C5X -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.

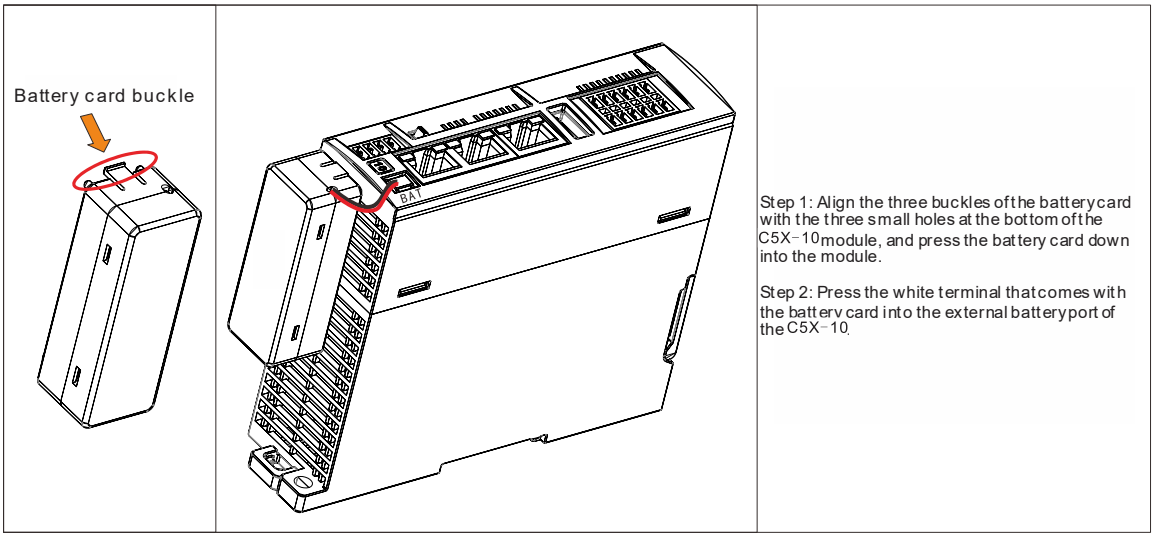
2)Press the terminal block down into the module to ensure that the terminal block is aligned and locked.

- **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 C5X -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 C5X -10.



Wiring

Wiring precautions:

Be careful

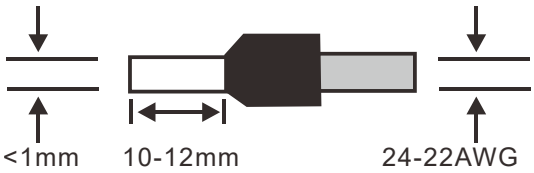
- When installing or wiring, ensure that all external power supplies are turned off.Failure to turn off all power may cause electric shock to the user or damage to the product.
- After installation or wiring, when starting the power supply or operating the PLC, confirm whether the wiring terminal block 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 Instructions:

1) Terminal blocks with crimped insulating sleeves cannot be used for the terminal block. It is recommended to use a sleeve with a label or insulating material to cover the crimp terminal lugs.

2) Please use 24-22AWG single-core wire or multi-core wire for wiring connecting to the terminal block, and use pin-type terminals with a diameter of less than 1mm for wiring, so as to improve the stability and reliability of wiring. Specifications are shown below:



3.5 Grounding and Wiring

❑ C5X-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 C5X-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 C5X-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, C5X-10 system must have emergency stop function, electromechanical interlock or other redundant safety facilities independent of C5X-10 motion controller.

3.6 Suppression Circuit

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

Notice

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.

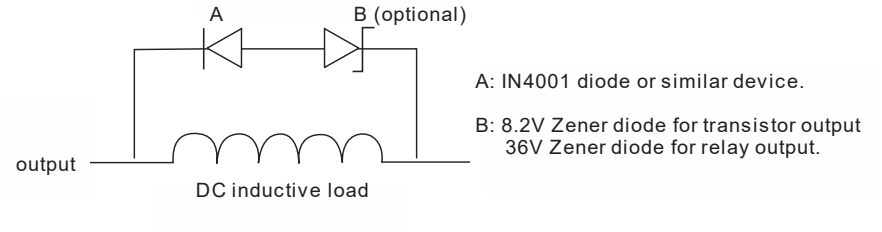


Figure 3-6 suppression circuit of DC load

❑ 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.

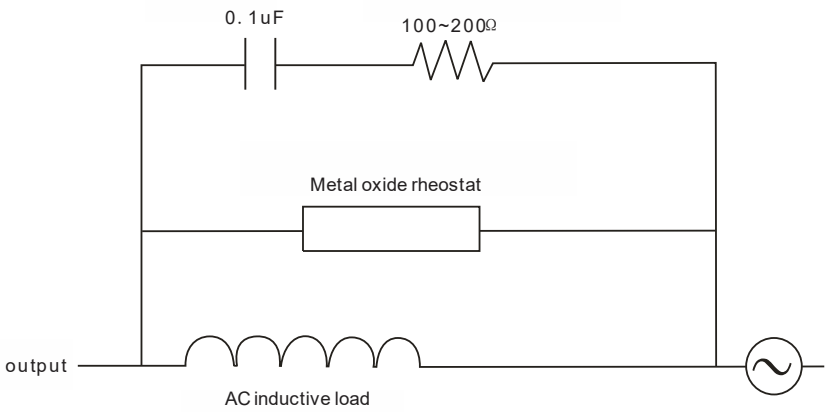


Figure 3-7 suppression circuit of AC load

Power module

4

4.1 Technical specifications

The PWR-02 power supply module can provide 24V DC for CTH300 PLC 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

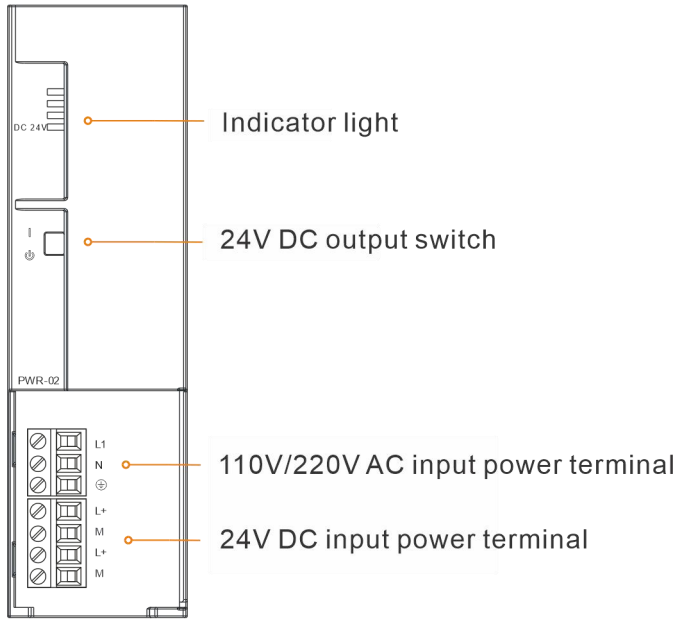
Name	Descreption	Order no.
PWR-02	Input: 85~264V AC output:24V DC/2A	CTH3 PWR-020S1

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

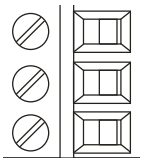
Removable terminal	Signal	Definition
	L	FireWire
	N	Zero line
		Grounded

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

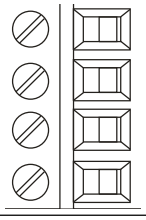
Removable terminal	Signal	Definition
	L+	24V+
	M	24V-
	L+	24V+
	M	24V-

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

Table 5-1 Basic characteristics of the main control module

Name	Specification Description	Order No.
C56-10	32MB program data space, 64KB power-down retention data, 55M expansion bus, 24V DC power supply, 10 digital inputs, 6*500kHz high-speed counter, 1 RS485 communication port (MODBUS free port protocol), 1 EtherCAT interface, 1 EtherNET interface, 1 CAN port, 1 USB port, supports SoftMotion instruction functions such as single-axis motion control, interpolation, electronic gear and electronic CAM, and supports the SP11 version of the CODESYS programming platform.	CTH3 C56-102S2
C57-10	32MB program data space, 64KB power-down retention data, 55M expansion bus, 24V DC power supply, 10 digital inputs, 6*500kHz high-speed counter, 1 RS485 communication port (MODBUS free port protocol), 1 EtherCAT interface, 1 EtherNET interface, 1 CAN port, 1 USB port, supports SoftMotion instruction set functions and CNC functions such as single-axis motion control, interpolation, electronic gear and electronic cam, and supports CODESYS programming platform SP11 version.	CTH3 C57-102S2

5.1 Basic Performance Parameters

Table 5-2 Regular feature

Physical characteristics	
Dimension(W×H×D)	34×115×101.6 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
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	4
Number of modules supported per rack	Main Rack: 11(power supply module, CPU, Intermediate expansion module, 8 modules) From Rack: 10(power supply module, Intermediate expansion module, 8 modules)
Number of ECT-00 slave supported by one CPU	Up to 128
Number of modules supported per ECT-00	8
Power down hold	
Clock power down holding time	The supercapacitor lasts for 112 hours. After C5X-10 is connected to an external battery, the power-off retention time can reach at least 2 year, (for the order information of the external battery, see J Product Oder Info..
Programming software	
programming package	CODESYS V3(SP11 version)
Programming language	Programming languages compliant with IEC61131-3: CFC/FBD/LD/IL/ST/SFC
Motion Controller Features	
Motion Controller Features	C56-10: SoftMotion C57-10: SoftMotion and CNC

5.2 CPU input function

5.2.1 Digital input

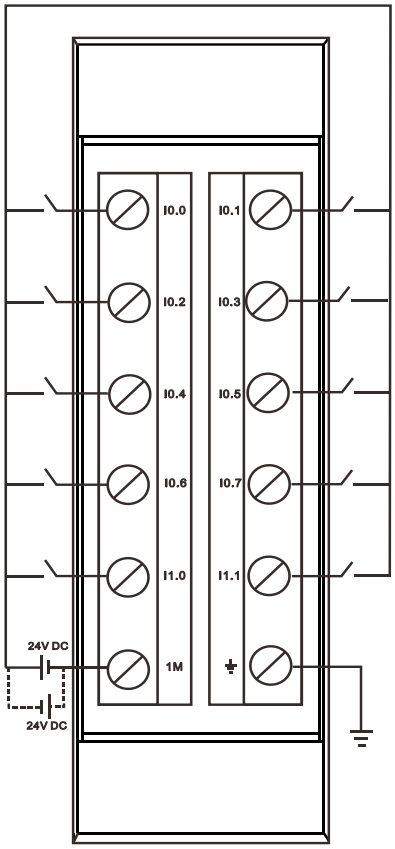
The CTH300 C5X-10 motion controller integrates 10 digital inputs. The input characteristics are shown in the table below.

Table 5-3 C5X-10 motion controller digital input characteristics

Digital input characteristics	
Native integrated IO number	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,default 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)
Digital Input Characteristics		
Pulse capture input		10
High counter	total	6
	single phase	6×500kHz
	double-phase	4×250kHz

Wiring diagram of CPU's own IO



5.2.2 High-Speed Counting Input

Table 5-4 C5X-10 motion controller high-speed counter input

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	double-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					

Table 5-5 High-speed counter parameters

Parameter	Data type	Value	Defaults	Describe
Module Id	DWORD	16#0a000605	16#0a000605	Module Id
channel0-3filter	BYTE	6	6	filter time 1: 0.2ms 2: 0.4ms 3: 0.8ms 4: 1.6ms 5: 3.2ms 6: 6.4ms 7: 12.8ms 8: 0.2us 9: 0.4us 10: 0.8us 11: 1.6us 12: 3.2us 13: 6.4us 14: 12.8us
channel4-7filter	BYTE	6	6	filter time 1: 0.2ms 2: 0.4ms 3: 0.8ms 4: 1.6ms 5: 3.2ms 6: 6.4ms 7: 12.8ms 8: 0.2us 9: 0.4us 10: 0.8us 11: 1.6us 12: 3.2us 13: 6.4us 14: 12.8us
channel8-9filter	BYTE	6	6	filter time 1: 0.2ms 2: 0.4ms 3: 0.8ms 4: 1.6ms 5: 3.2ms 6: 6.4ms 7: 12.8ms 8: 0.2us 9: 0.4us 10: 0.8us 11: 1.6us 12: 3.2us 13: 6.4us 14: 12.8us
HSC0 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1:not clear
HSC0 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(Hz) 1: 750k 2: 500k 3: 375k 4: 250k 5: 125k 6: 100k 7: 75k
HSC0 Current	DINT	0	0	

Value				
HSC0 Preset value	DINT	0	0	
HSC0 Speed Test Time(ms)	BYTE	5	5	
HSC0 Ctrl	BYTE	16#F9	16#F9	bit0: Reset Level 0: Low 1: High bit1~bit2: quad Rate 0: 4x 1: 2x 2: 1x bit3: Direction 0:Decrease 1: Increase bit4: Direction Update 0: Not Update 1: Update bit5: Preset Value Update 0: Not update 1: Update bit6: Current Value Update 0: Not update 1: Update bit7: HSC Enable 0: Disable 1: Enable
HSC1 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1:not clear
HSC1 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(Hz) 1: 750k 2: 500k 3: 375k 4: 250k 5: 125k 6: 100k 7: 75k
HSC1 Current Value	DINT	0	0	
HSC1 Preset value	DINT	0	0	
HSC1 Speed Test Time(ms)	BYTE	5	5	
HSC1 Ctrl	BYTE	16#F9	16#F9	bit0: Reset Level 0: Low 1: High bit1~bit2: Quad Rate 0: 4x 1:2x 2:1x bit3: direction 0:decrease 1: increase bit4: direction update 0: not update 1: update bit5: preset value update 0: not update 1: update bit6: Current Value update 0: not update 1: update bit7: HSC Enable 0: disable 1: enable
HSC2 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0: clear 1: not clear
HSC2 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(Hz) 1: 750k 2: 500k 3: 375k 4: 250k

				5: 125k 6: 100k 7: 75k
HSC2 Current Value	DINT	0	0	
HSC2 Preset value	DINT	0	0	
HSC2 Speed Test Time(ms)	BYTE	5	5	
HSC2 Ctrl	BYTE	16#F9	16#F9	bit0: Reset Level 0: Low 1: High bit1~bit2: Quad Rate 0: 4x 1: 2x 2: 1x bit3: direction 0:decrease 1: increase bit4: direction update, 0: not update 1: update bit5: preset value update, 0: not update 1: update bit6: Current Value update, 0: not update 1: update bit7: HSC Enable 0: disable 1: enable
HSC3 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0: clear 1: not clear
HSC3 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(Hz) 1: 750k 2: 500k 3: 375k 4: 250k 5: 125k 6: 100k 7: 75k
HSC3 Current Value	DINT	0	0	
HSC3 Preset value	DINT	0	0	
HSC3 Speed Test Time(ms)	BYTE	5	5	
HSC3 Ctrl	BYTE	16#F9	16#F9	bit0: Reset Level 0: Low 1: High bit1~bit2: Quad Rate 0: 4x 1: 2x 2: 1x bit3: direction 0: decrease 1: increase bit4: direction update, 0: not update 1: update bit5: preset value update, 0: not update 1: update bit6: Current Value update' 0: not update 1: update bit7: HSC Enable 0: disable 1: enable
HSC4 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0: clear 1: not clear

HSC4 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(Hz) 1: 750k 2: 500k 3: 375k 4: 250k 5: 125k 6: 100k 7: 75k
HSC4 Current Value	DINT	0	0	
HSC4 Preset value	DINT	0	0	
HSC4 Speed Test Time(ms)	BYTE	5	5	
HSC4 Ctrl	BYTE	16#F9	16#F9	bit0: Reset Level 0: Low 1: High bit1~bit2: Quad Rate 0: 4x 1: 2x 2: 1x bit3: direction 0: decrease 1: increase bit4: direction update, 0: not update 1: update bit5: preset value update, 0: not update 1: update bit6: Current Value update, 0: not update 1: update bit7: HSC Enable 0: disable 1: enable
HSC5 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1: not clear
HSC5 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(Hz) 1: 750k 2: 500k 3: 375k 4: 250k 5: 125k 6: 100k 7: 75k
HSC5 Current Value	DINT	0	0	
HSC5 Preset value	DINT	0	0	
HSC5 Speed Test Time(ms)	BYTE	5	5	
HSC5 Ctrl	BYTE	16#F9	16#F9	bit0: Reset Level 0: Low 1: High bit1~bit2: Quad Rate 0: 4x 1: 2x 2: 1x bit3: direction 0:decrease 1: increase bit4: direction update, 0: not update 1: update bit5: preset value update, 0: not update 1: update bit6: Current Value update, 0: not update 1: update

5.3 Communication function

5.3.1 Communication Port Specification

Table 5-6 Integrated Port Specifications

Integrated communication functions	
bus interface	Provide expansion module interface
USB port	Provides USB host interface
EtherNET port	Provide 1 EtherNET interface
RS485 port	Provide 1 RS485 interface
EtherCAT port	Provides 1 EtherCAT interface
CAN port	Provide 1 CAN interface

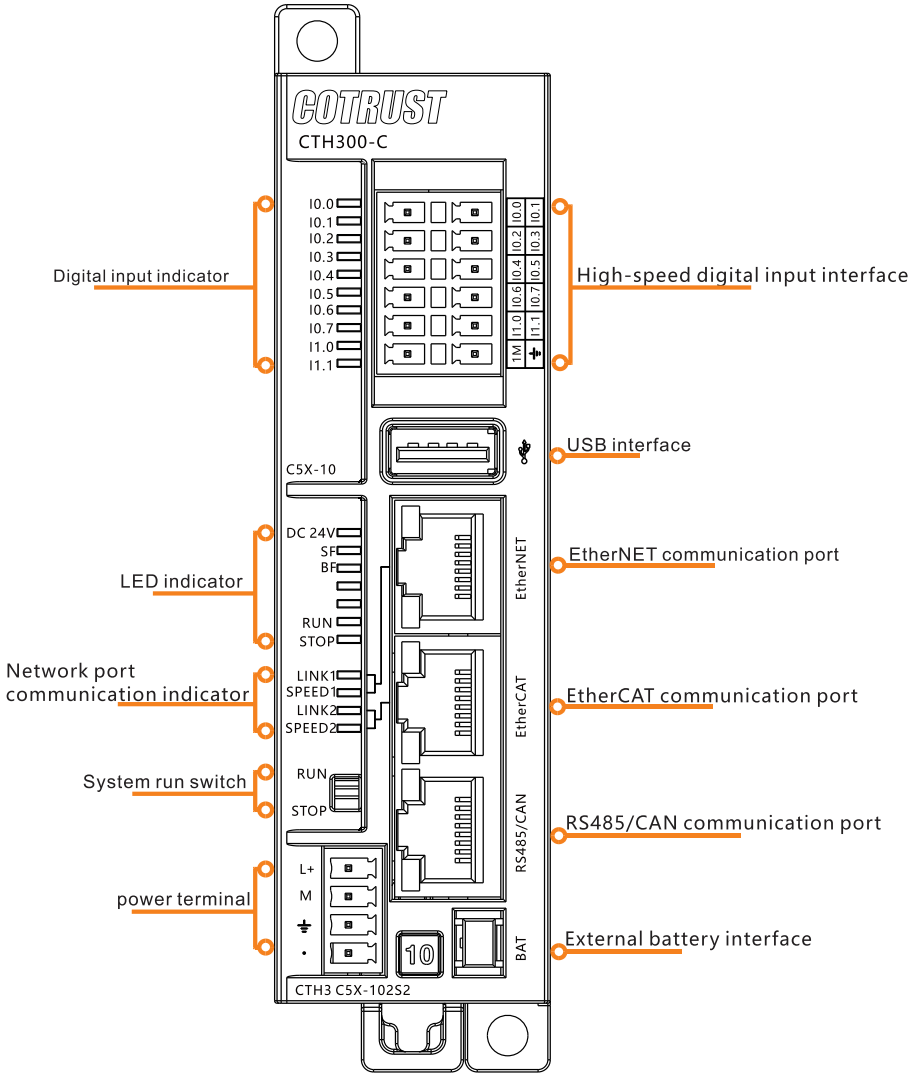
Table 5-7 Communication port specifications

RS485 communication port			
Communication interface and protocol	1 RS485 communication port (Modbus free port mode)		
Free port baud rate	1.2kbps~115.2kbps, Built in MODBUS master / slave function		
Maximum cable length per segment	Isolation repeater: 1000m (115.2kbps) / 1200m (38.4kbps) Isolation repeater not used: 50m		
Maximum number of stations	32 stations per segment, 126 stations per network		
Isolation	Communication port isolation		
EtherNET communication port			
Communication Interface	1 EtherNET port		
baud rate	10/100Mbps adaptive		
Protocol type	UDP protocol, Modbus TCP protocol, EtherNET/IP protocol		
Maximum cable length per segment	100m		
Maximum number of connections for a site	UDP supports up to 16 connections and Modbus TCP supports up to 32 connections.		
Isolation	Communication port isolation		
Ethernet network read / write instruction	parameter	data type	describe
TCP_MBUS_MSG	EN		Enable bit
	First	BOOL	Read / write request bit, each new read / write request must be triggered by pulse
	IP	STRING	Target IP address, MODBUS over TCP fixed port 502
	RW	BYTE	Operation command: 0: read, 1: write
	Addr	DWO	Select the data type of reading and writing, 0000

		RD	to 0xxxx--switch output 10000~1xxxx: Switch input 30000~3xxxx: Analog input 40000~4xxxx: holding register
	Count	INT	Number of communication data (number of bits or words), each MBus_MSG of Modbus master The maximum amount of data that the instruction can read / write is 120 words
	DataPtr	INT	Data pointer. If it is a read instruction, the read back data is placed in this data area; If it is a write instruction, the data to be written is placed in this data area
	Done	BOOL	Completion bit, read / write function completion bit
	Error	BYTE	The error code is valid only when the done bit is 1. The error code is as follows: 0 = no error 1 = response check error 2 = not used 3 = Receive timeout (no response from slave) 4 = Request parameter error (slave address, Modbus address, count, RW) 5 = Modbus/Freeport not enabled 6 = Modbus is busy with other requests 7 = response error (response is not the requested operation) 8 = response CRC check and error 101 = Slave does not support the requested function 102=Slave does not support data address 103＝Slave does not support this data type 104＝Slave device failure 105＝The slave accepts the message, but the response is delayed 106＝The slave is busy and rejected the message 107＝Slave rejected message 108= Slave memory parity error
EtherCAT communication port			
Communication interface	1 EtherCAT communication master interface		
baud rate	10/100Mbps adaptive		
Protocol type	EtherCAT interface protocol		
Maximum cable length per segment	100m		

Maximum number of slave	Each master supports up to 128 EtherCAT slaves						
Supported functions	Support distributed clock configuration, redundant settings Support startup parameter configuration Support for configuring PDO parameters and mapping Supports configuration of bus cycle period, configuration startup checks vendor ID and product ID						
Power protection	The power supply end provides reverse polarity protection and surge absorption						
Interface protection	Communication port lightning protection						
Isolation	Communication port isolation						
CANopen communication port							
Communication Interface	1 CAN communication master interface						
Maximum number of slave sites	Up to 32 slaves can be connected behind 1 master						
agreement type	CANopen DS301 standard protocol						
Support function	Auto start CANopen manager, optional slave polling, start slave, NMT, synchronous production, synchronous consumption, heartbeat generation, create activation time.						
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						

5.3.2 External interface definition



Schematic diagram of external interface

Table 5-8 Definition of power interface

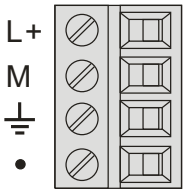
4-position removable terminal	Symbol	Signal definition
	L+	24V power +
	M	24V power +
	⏚	24V ground
	--	--

Table 5-9 Definition of USB interface

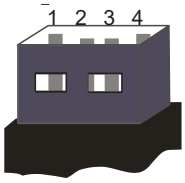
USB interface	NO	Signal	Signal definition
	1	V_BUS	+5V power supply
	2	Data-	
	3	Data+	
	4	GND	

Table 5-10 Definition of EtherNET/EtherCAT interface

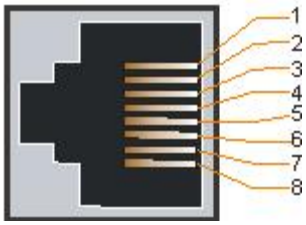
Dual network port interface	NO	Signal	Signal definition
	1	TX+	Data sending +
	2	TX-	Data sending -
	3	RX+	Data receiving +
	4	TERM	--
	5	TERM	--
	6	RX-	Data receiving -
	7	TERM	--
	8	TERM	--
	shell	PE	ground

Table 5-11 Definition of RS485/CAN communication interface

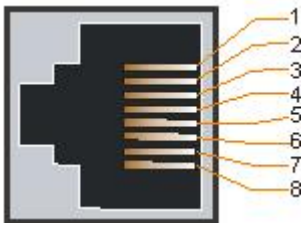
Single network port	No	Signal	Signal definition
	1	CAN_H	data sending +
	2	CAN_L	Data sending -
	3	--	--
	4	A0	RS485 SignalA/—
	5	B0	RS485 SignalB/+
	6	--	--
	7	CAN_GND	CAN/RS485 signal ground
	8	--	--
	shell	PE	Chassis ground

Table 5-12 Definition of system running DIP switch

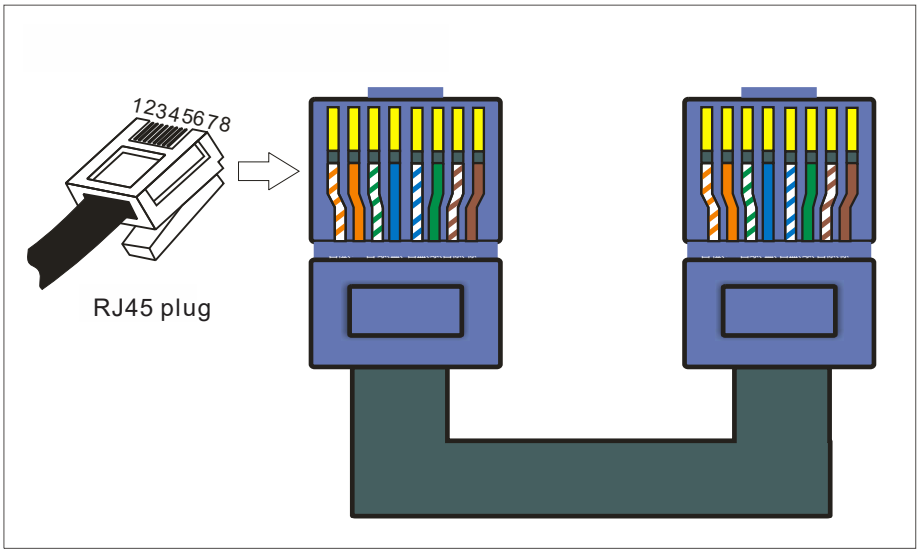
System run switch	symbol	Dial direction	Signal definition
	RUN	up	system running
	STOP	dowm	system stopped

5.3.3 Make standard network cables

When the C5X-10 controller communicates with the following, it is recommended to use a standard network cable:

- The main control module communicates with the host computer via EtherNET
- Master control module for EtherCAT communication
- Main control module for CANopen communication
- The main control module conducts CANopen communication with CAN slaves such as E10/A3S/A4S/EM277C

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



5.4 Data Memory Specifications

C5X-10 stores information to different memory units. Each unit has a unique address. You can clearly clear the memory address to be accessible, which allows user programs to directly access this information.

The memory position and size are represented by a special character sequence. Grammar: %<range prefix> <Lifferent prefix> <Number |. Number |. Numbers ....>

Table 5-13 The memory area that can be used:

Storage area	Name	Bytes
I	The input area (input the physical input drive, "sensor")	1280
Q	The output area (the physical output driven by the output, the "actuator")	1280
M	Memory area	32767

Table 5-14 Length prefixes:

X	Single binary
None	Single binary
B	byte(8 bits)

W	word(16 bits)
D	double word(32bits)

Table 5-15 Example

%QX7.5	Address 7 of the output area, bit 5
%IW215	Address of input area 215, 1 word
%QB7	Address of the output area 7,1 byte
%MD48	Address 48 of the memory area, double word.
%IW2.5.7.1	Depends on PLC settings
ivar AT %IW0 : WORD;	Example of variable declaration with specified address

See the table below for a comparison of byte addressing and word-oriented IEC addressing for bits, bytes, words and dwords. This table plots the overlapping memory areas in byte addressing mode (see example after table).

Regarding the writing method, please note that for bit addresses, the IEC addressing mode is always word-oriented, that is, the area before the dot corresponds to the number of words, and the area after the name corresponds to the number of bits.

Table 5-16 Comparison of byte and word-oriented addressing for address sizes D, W, and X:

double word/word				byte	X(bit)					
byte addressing		Word-Oriented IEC Addressing			byte addressing			Word-Oriented IEC Addressing		
D0	W0	D0	W0	B0	X0.7	...	X0.0	X0.7	...	X0.0
D1	W1			B1	X1.7	...	X1.0	X0.15	...	X0.8
...	W2		W1	B2				X1.7	...	X1.0
	W3			B3				X1.15	...	X1.8
	...	D1	W2	B4						
				B5						
			W3	B6						
				B7						
		D2	W4	B8						
		...		...						
		...								
		...								
D(n-3)		D(n/4)								
	W(n-1)		W(n/2)							
				Bn	Xn.7	...	Xn.0	X(n/2).15	...	X(n/2).8

n = byte count

In byte addressing mode, the memory overlap varies, for example:

D0 includes B0-B3, W0 includes B0 and B1, W1 includes B1 and B2, and W2 includes B2 and B3. To avoid overlapping, neither W1 nor D1, D2, D3 can be used for addressing!

<Note> If no explicit single-bit address is specified, Boolean value is one byte. For example: changing the value of varbool1 AT %QW0 will affect the content from QX0.0 to QX0.7.

<Note> Be careful when using pointers to addresses, online modifications may change the contents of the address!

5.5 C5X-10 controller characteristics

1. Communication function

Select the communication interface for the network

The C5X-10 supports various types of communication networks, and the interfaces that can access these networks include:

- bus interface
- RS485 interface (RJ45 standard network port)
- EtherCAT interface (RJ45 standard network port)
- CANOpen interface (RJ45 standard network port)
- EtherNET interface (RJ45 standard network port)

Choose a communication protocol for your network

Protocols supported by C5X-10 include:

- bus protocol
- UDP protocol
- Modbus TCP protocol
- Modbus RTU protocol
- CANopen DS301 standard protocol
- EtherCAT protocol
- EtherNET/IP protocol

Building a network based on communication protocols

Table 5-17 Introduction to the communication network

Communication method	Communication interface and protocol	Implement Communication Operation in CODESYS on CODESYS
Bus communication	Bus communication can be realized by connecting C5X-10 expansion modules through the bus interface of C5X-10 motion controller (supporting bus protocol).	Add the device that matches the actual expansion module through the "Co-trust LocalBus" of the device catalog, and then read and write the memory through the IO map of the corresponding module.
Ethernet communication	EtherNET communication port (support UDP, EtherNET/IP industrial Ethernet protocol), connect with communication	Add EtherNET/IP communication devices through the device catalog: EtherNet Master, EtherNET/IP Slave, etc.

	equipment, and can carry out EtherNET/IP communication.	
EtherCAT communication	EtherCAT interface (support EtherCAT protocol), connect C5X-10 motion controller and EtherCAT slave station via standard Ethernet cable to build a linear network.	Add EtherCAT communication devices by the device catalog: EtherCAT Master, EtherCAT Slave, etc.
Modbus communication	RS485/CAN port: Modbus RTU communication. EtherNET interface: Modbus TCP	The ct_Modbus library file provided by COTRUST can be used for MODBUS RTU and MODBUS TCP communication. For the use of the Ct_Modbus library, please refer to the appendix "E Introduction to the Use of the CT_MODBUS Library".
CANopen communication	RS485/CAN port, connect with the CAN slave station to realize CANopen communication.	Add CANopen communication devices through the device catalog: CANBUS, CAN master, CAN slave, CAN slave module, etc.

<Remarks> Please refer to chapter 7 Communication example for the usage of various communication methods.

2. Motion control function

Using a high-performance motion control software platform, embedded single-axis or multi-axis motion control algorithms and instructions.

Electronic cam function

◆ Support synchronization functions such as electronic gear/electronic cam, virtual axis and encoder axis.

◆ Graphical cam editor enables fast execution of complex motion control trajectories. The cam table can be corrected in real time and dynamically.

◆ Motion control of general functions can be realized through PLCopen function blocks in the motion control library.

CNC function (only supported by C57-10)

◆ Support DIN66025 G code and DXF file import.

◆ Support 5-axis linkage, support parallel robot control.

◆ CNC Path View, all changes made in the CNC text editor are automatically updated in the Path View.

3. Logic control function

program organization unit

The C5X-10 controller consists of the following program organization units:

Program: The program is the location where the control application program instructions are stored. The instructions in the program are executed in sequence and in different scanning modes.

Function Block: A function block is a block written by the user with its own storage area. Each time a function block is called, various types of data should be provided to the function block, and the function block should also return variables to the calling block.

Function (Function): Function (FC) belongs to the block of personal programming. A function is a logical block without memory. Temporary variables belonging to FCs are kept in the local data stack. Since the FC itself has no memory, it must always be assigned actual parameters. An initial value cannot be assigned to the local data of the FC.

Programming language

All programming languages specified in the IEC_61131 standard are supported:

- Textual language: Instruction List (IL), Structured Text (ST)
- Graphical language: Sequential Function Chart (SFC), Function Block Diagram (FBD), Ladder Diagram (LD), Continuous Function Chart (CFC) based on Function Block Diagram

<Remark> Please refer to chapter 6.2 Programming language for the description of programming language.

Standardized PLC functions

Standardized PLC functions can be realized by using related instructions, such as: type conversion, numerical function, arithmetic function, shift function, Boolean operation function, selection function, comparison function, string function, timer, counter, edge detection, bistable state elements, etc.

5.6 Real time clock function

- Power-down retention time is about 112 hours (typical value)
- C5X-10 comes with an external battery port. After the external battery is connected, the power-down retention time is a typical value of more than 2 years.
- Monthly deviation < 60 seconds
- Read/set the clock through the command setrtc

To set or read the real-time clock of the C5X-10 series motion controller in CODESYS: Double-click to open "Device (CTH3 C56-10S2_V0.9)" in the device view, and then select the tab "PLC Shell":

1) Enter a command in the PLC shell, such as "setrtc 2013-09-23 08:30:00", and then press the ENTER key to set the current RTC.

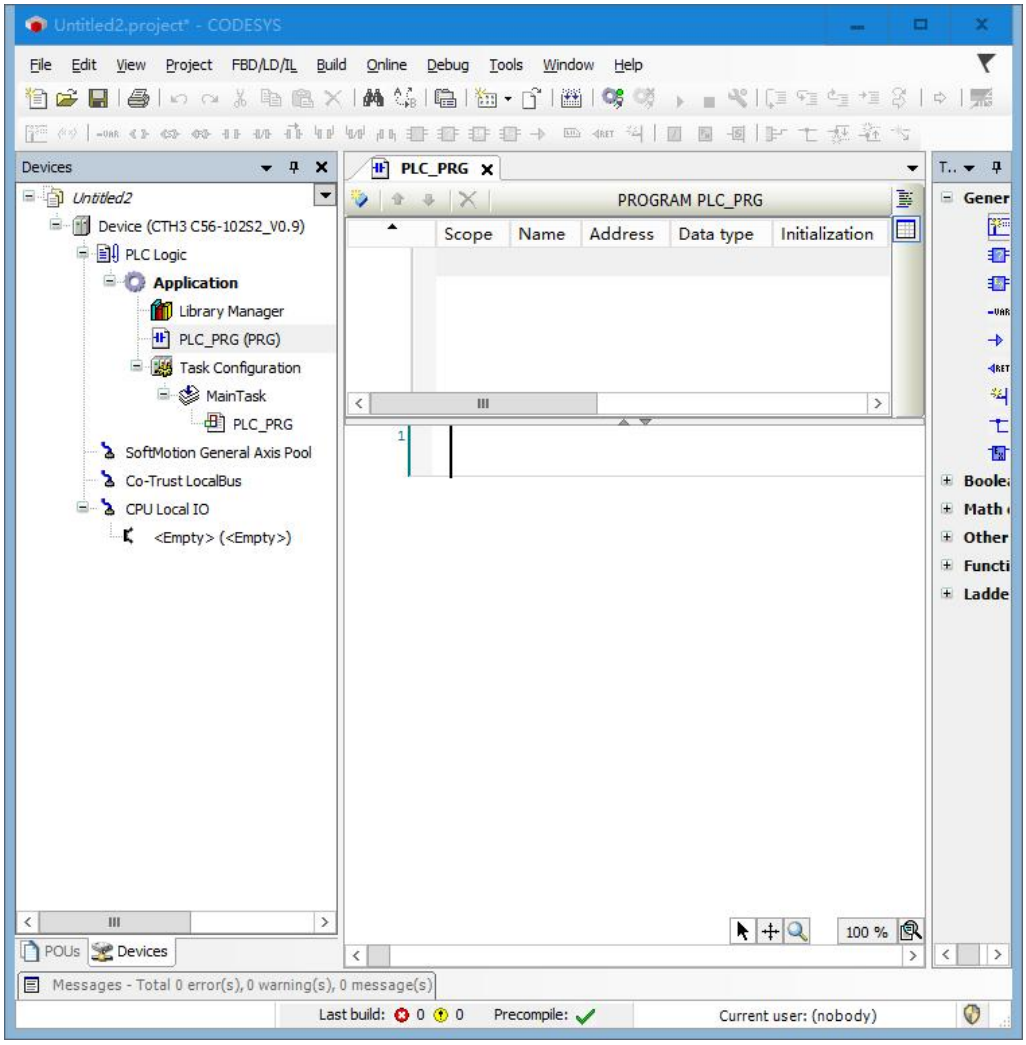
2) Enter the command "setrtc" in the PLC shell, and then press the ENTER key to read the current RTC.

<Remarks> The C5X-10 motion controller has a random time when it leaves the factory, so you need to manually set the current time first, and then read the RTC again to display the correct time.

Structure of CODESYS

6

CODESYS supports all programming languages compliant with the IEC 61131-3 standard and also supports the C language. Combined with the CODESYS real-time system, multiple controller devices can be configured in one project. The CODESYS programming user interface consists of an arrangement of "components", refer to the following figure:



<Remarks> This manual only briefly introduces the application of CODESYS. For the detailed use of CODESYS, please refer to the online help of the CODESYS software.

6.1 Composition of the project

A project contains all the objects in the PLC program, and is stored in the name of the project file. The project contains the following objects: POUs, data types, visualization interfaces, resources and libraries.

POU

POU refers to "Program Organization Unit" which can be of program, function block or function type, and they can be added at any time.

Each POU contains a local and subject declaration definition, the main part can be written in IEC languages, these languages include Instruction List (IL), Structured Text (ST), Sequential

Function Chart (SFC), Function Block Diagram (FBD)), Ladder Diagram (LD) or Continuous Function Chart (CFC).

CODESYS supports all IEC standard POUs. If these POUs are used in the project file, the standard library file standard.lib must be included in the project file. POUs can call other POUs, but recursive calls are not allowed.

Function

A function is a POU that correctly produces a data element (which contain several elements, such as Fields or structures), which can be called as an operand in a textual language expression during processing. When declaring a function, to determine the data type for the function, add a colon after the function name followed by a data type.

The function declaration can refer to the following example:

FUNCTION Fct: INT

Additionally, a result must be assigned to the function.

The declaration of a function begins with the keyword FUNCTION.

In IL, the invocation of a function is within a single step or a single transformation. A call to a function in ST can be used as an operand in an expression.

Function block

A function block is a POU that provides one or more values in a program. The function block has no return value. The declaration of a function block begins with the keyword FUNCTION_BLOCK, and a copy or instance of the function block can be created.

Program

A program is a POU, which has a return value during operation, and the program is global in the project file. All final values of the program are retained until the next program run.

PLC_PRG

PLC_PRG is a special predefined POU, each project file must contain a PLC_PRG. Actually this POU is called only once per control loop.

After creating a new project file, add an object, click "Project" → "Object Add", the default project in the POU dialog box is a POU of program type named PLC_PRG. These default settings cannot be changed.

If tasks are defined, PLC_PRG may not be included in the project, because in this case, the timing of the program depends on the assignment of tasks.

<Note> Do not delete or rename the POU PLC_PRG (if no task configuration is used), PLC_PRG is the main program in a single task program

Action

Actions can be defined and assigned to function blocks and programs. Actions represent a further execution. It can be created in other languages. Each action has a name.

Each action works with data in a function block or program, and the action is executed using the same input/output variables and local variables.

Resource

You need resources to configure and organize project files and track variable values.

- Global variables are used in all project files or in the network.
- The library manager is used to add library files to the project.
- Log files record actions during the online period.

.

- PLC Configuration configures the hardware of the programmable controller.
- Task configuration guides the work of the program through task division.
- Monitor and Receive Manager to display variable values and set default variable values.
- Target System Settings is used to select and, if necessary, make the final configuration of the target system.
- Workspace as an image of project options.

To build the target system and target system settings in the CODESYS project, the following resources may be used:

- Sampling trace for graphical display of variable values.
- Parameter manager for exchanging data with other controllers in the same network.
- PLC browser for controller monitoring.
- Tools, related to the target system, call external tool programs inside and outside CODESYS.

Library file

A series of library files can be included in the project file, and the POUs, data types, and global variables of the library files can be used like custom variables. The standard.lib and util.lib in the library file are programs that can be called freely

Type of data

With reference to the standard data types, users can define their own data types, and can create structure enumeration types and reference types.

Visual interface

CODESYS provides a visual interface, so the variables of the project can be displayed, and the geometry can be drawn offline with the help of visualization, and their shape, color and text output can be changed according to the value of specific variables in the online mode.

The visualized interface can be used as a PLC utility operator interface for HMI with CODESYS, or displayed as a web page or target system, visualized directly via the Internet or PLC.

6.2 Programming language

CODESYS provides corresponding editors that support all programming languages specified in the IEC_61131 standard:

- Languages in text form: Instruction List (IL), Structured Text (ST)

• Graphical language: Sequential Function Chart (SFC), Function Block Diagram (FBD), Ladder Diagram (LD), Continuous Function Chart (CFC) based on Function Block Diagram

6.2.1 Instruction List (IL)

The instruction list contains a series of instructions, depending on the type of operation, each instruction starts on a new line and contains an operation symbol and one or more operands separated by commas. In front of an instruction, there can also be a label, followed by a colon. The comment section is at the end of a line, and blank lines can be inserted between instructions.

The following operators and qualifiers will be used in the instruction list:

Qualifier:

C is used in conjunction with operators JMP, CAL, RET: This instruction is only executed when the result of the preceding expression processing is TRUE.

N is used in conjunction with operators JMPC, CALC, RETC: This instruction is only executed when the result of the previous expression processing is FALSE.

N is used in other cases: negates the operand (excluding the accumulator).

The following are the operators and their possible qualifiers and their associated meanings:

Operator and Qualifier Meaning	
LD N	make the current value equal to the operand
ST N	saves the current value at the operand's place
S	Set the boolean operand to TRUE when the current value is TRUE
R	When the -current value is TRUE, set the boolean operand to FALSE
AND N,(	Bitwise logical operator "AND"
OR N, (	bitwise logical operator "OR"
XOR N,(	Bitwise logical operator "XOR"
ADD (	addition
SUB(	subtraction
MUL(	multiplication
DIV(	division
GT(	>
GE(	>=
EQ(	=
NE(	<>
LE(	<=
LT(	<
JMP CN	jump
CAL CN	Caller function block
RET CN	leave the POU and return to where it was called
)perform a delayed operation	

Example: Program written with qualifiers

LD TRUE (* load TRUE into accumulator*)

AND N BOOL1 (*Execute "AND" after negation of AND and BOOL1 variables *)

JMPC mark (*When the above result is TRUE, jump to the label "mark" *)

LDN BOOL2 (*save the reverse of BOOL2 *)

ST ERG (*save BOOL2 in ERG *)

Lable:

LD BOOL2 (*Save the value of BOOL2 *)

ST ERG (*save BOOL2 in ERG *)

In IL you can put a parenthesis after the operation. The value in parentheses is considered an operand. E.g:

LD 2

MUL 2

ADD 3

Erg

Here Erg is 7, but if you add a parenthesis:

LD 2

MUL (2

ADD 3

)

ST Erg

The result of Erg is 10. When ")" is reached, the operation MUL starts to calculate. At this time, the operand 5 calculates the MUL.

6.2.2 Structured Text (ST)

Structured text contains a series of instructions written in a high-level language that can be executed (eg IF...THEN...ELSE) or in a loop (WHILE...DO).

Expression

An expression is a structure that returns a value after an operation. Expressions consist of operators and operands, which can be constants, variables, function calls, or other expressions.

Expression calculation

The value of the operand calculation expression is processed according to certain rules. The operator with the highest priority participates in the operation first, and is calculated step by step according to the priority, until all operators are processed. Operation symbols of the same precedence are processed in left-to-right order.

Here is the arrangement of operator precedence in structured text:

Operate	Symbol	The priority

put in parentheses	(expression)	highest priority
function call	function name (parameter list)	
exponentiation	EXPT	
Reverse	-	
multiplication	*	
division	/	
Take mold	MOD	
addition	+	
subtraction	-	
Compare	<,>,<=,>=	
equal to	=	
not equal to	<>	
Boolean operation "AND"	AND	
Boolean operation "XOR"	XOR	
Boolean operation "OR"	OR	lowest priority

The following are other directives in structured text, arranged in a table with examples.

Type of instruction	Example
assignment	A: =B; CV : = CV + 1 ;C: =SIN(X);
Call a function block and use the function block output	CMD_TMR(IN : = %IX5, PT : = 300) ; A: =CMD_TMR.Q
RETURN	RETURN ;

IF	D: =B*B ;
	IF D<0.0 THEN
	C: =A;
	ELSIF D=0.0 THEN
	C: =B;
	ELSE
	C: =D;
	END_IF;

CASE	CASE INT1 OF 1: BOOL1: = TRUE; 2: BOOL2: = TRUE; ELSE BOOL1: = FALSE; BOOL2: = FALSE; END_CASE;

FOR	J: =101 ;
	FOR I: =1 TO 100 BY 2 DO
	IF ARR[I] = 70 THEN
	J: =I;
	EXIT;

	END_IF; END_FOR;
WHILE	J: =1 ; WHILE J<= 100 AND ARR[J] <> 70 DO J: =J+2; END_WHILE;
REPEAT	J: =-1 ; REPEAT J: =J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT ;
Empty instruction	;

Assignment operator

The left of the assignment symbol is an operand (variable, address), and the right of the ": =" is the value of the expression assigned to it, for example:

Var1: =Var2*10

After the operation, the variable Var1 gets 10 times the value of Var2.

Calling function blocks in structured text

A function block is called by writing the instance name of the function block, and then assigning values to parameters in parentheses. In the following example, a timer is called by assigning values to two parameters IN and PT, and then the value of the resulting variable Q is assigned to variable A.

RETURN instruction

The return instruction can be used to conditionally leave a POU (POU).

IF instruction

The IF instruction can check a condition and, according to this condition, execute the instruction.

grammar:

IF <Boolean_expression1> THEN

<IF_instructions>

{ELSIF <Boolean_expression2> THEN

<ELSIF_instructions1>

.

.

ELSIF <Boolean_expression n> THEN

<ELSIF_instructions n-1>

ELSE <ELSE_instructions>}

END_IF;

Parts within {} are optional.

If the Boolean expression <Boolean expression> returns TRUE, only the part of the if instruction is executed, and the other parts are not executed. Otherwise, the Boolean expression starts with <Boolean expression 2> and is evaluated one by one until a Boolean expression returns TRUE. Then, after this Boolean operation expression 2, the part before ELSE or ELSE IF is calculated.

If none of the Boolean expressions return TRUE, then only the instructions under ELSE are evaluated. E.g:

IF temp<17

THEN heating_on : = TRUE;

ELSE heating_on : = FALSE;

END_IF;

Heating starts when the temperature drops below 17 degrees, otherwise it remains off.

CASE instruction

Using the CASE instruction, you can combine multiple conditional judgment instructions with the same condition variable in one structure. Sentence:

CASE <Var1> OF

<Value1>: <Instruction 1>

Languages...

2-14 CoDeSys V2.3

<Value2>: <Instruction 2>

<Value3, Value4, Value5>: <Instruction 3>

<Value6 .. Value10>: <Instruction 4>

...

<Value n>: <Instruction n>

ELSE <ELSE instruction>

END_CASE;

CASE instructions are processed according to the following patterns:

If the variable Var1 has the value Value1, then execute the instruction Instruction1.

If the variable Var1 is not the specified value, then execute the ELSE Instruction.

If there are multiple variable values to execute the same instruction, then these conditions execute a common instruction.

If the same instruction is executed within a range of values for a variable, the initial value and the final value are separated by two periods, so common conditions can be specified.

E.g:

CASE INT1 OF

1, 5: BOOL1 : = TRUE;

BOOL3 : = FALSE;

2: BOOL2 : = FALSE;

BOOL3 : = TRUE;

10..20: BOOL1 : = TRUE;

BOOL3: = TRUE;

ELSE

BOOL1 : = NOT BOOL1;

BOOL2 : = BOOL1 OR BOOL2;

END_CASE;

FOR loop

Through the FOR loop program, it is possible to write processing procedures that are executed repeatedly.

Sentence:

INT_Var :INT;

FOR <INT_Var> : = <INIT_VALUE> TO <END_VALUE> {BY <Step size>} DO

<Instructions>

END_FOR;

Parts within {} are optional.

As long as the counter INT_Var is not greater than END_VALUE, Instructions will always be executed. Before executing Instructions, check the value of the counter first. If INIT_VALUE is greater than END_VALUE, Instructions will not be executed.

When Instructions are executed, INT_Var usually increases by a Step size. Step size can be any integer value. If there is no Step size, it will be set to 1. When INT_Var reaches a certain value, the loop ends.

E.g:

FOR Counter: =1 TO 5 BY 1 DO

Var1: =Var1*2;

END_FOR;

Erg: =Var1;

We assume that the default value of Var1 is 1, then it will get the value 32 after the loop ends.

<Note> END_VALUE must not be greater than or equal to the limit value of the counter INT_VAR, for example: If the variable counter is a SINT type and END_VALUE is 127, then this will be an infinite loop.

WHILE loop

The WHILE loop can be used like a FOR loop, the difference is that the exit condition of the WHILE loop can be any Boolean expression, and when the condition is met, the loop will be executed. Sentence:

WHILE <Boolean expression>

<Instructions>

END_WHILE;

As long as the Boolean_expression returns TRUE, the Instructions are repeated. If the Boolean_expression evaluates to FALSE for the first time, the instructions will not be executed. If the Boolean_expression never appears FALSE, the Instructions will be repeated endlessly.

<Note> The programmer must ensure that an infinite loop does not occur. This can be achieved by changing the conditions in the instruction part of the loop, for example, by incrementing or decrementing the counter.

E.g:

WHILE counter<>0 DO

Var1 : = Var1*2;

Counter : = Counter-1;

END_WHILE

For the WHILE and REPEAT loops, it is not necessary to know the number of loops before the loop. In this sense, these two kinds of loops are more powerful than FOR.

Therefore, in such cases, these two loops can be used. If the number of loops is more explicit, it is better to use FOR because there is no infinite loop.

REPEAT loop

The difference between a REPEAT loop and a WHILE loop is that its break condition is checked after the loop executes, which means that the loop executes at least once, regardless of the break condition.

Sentence:

REPEAT

<Instructions>

UNTIL <Boolean expression>

END_REPEAT;

Instructions are executed until the Boolean expression returns TRUE. If the Boolean expression is assigned a true value for the first time, the Instructions are executed only once, otherwise the Instructions will be executed repeatedly and will cause a time delay.

<Note> The programmer can ensure that no infinite loop occurs by changing the condition of the instruction part of the loop, for example, by incrementing or decrementing the counter.

E.g:

REPEAT

Var1 : = Var1*2

Counter : = Counter-1;

UNTIL

Counter=0

END_REPEAT;

EXIT instruction

If there is an EXIT instruction in a FOR, WHILE or REPEAT loop, the inner loop ends, regardless of the interrupt condition.

6.2.3 Sequential Function Chart (SFC)

Sequential function chart is a graphical language that can describe the sequence of different actions in a program. Because these actions are assigned to single-step elements, and transition variables are used to control the order of processing.

Step

A POU written with SFC contains a series of steps, which are realized through directional connections (transition conditions).

There are two types of steps:

•Simple type: Each step includes an action and a flag, which is used to indicate whether the step is active. If a single step action is being performed, a small triangle will appear in the upper right corner of the step.

• IEC Type: Action or Boolean variable with each step containing a flag and one or more assignments. The associated action appears to the right of the step.

Action

An action can contain a series of instruction lists or structured text instructions, a function block diagram or a ladder diagram of many networks, or another sequential function diagram.

In simple steps, actions are often linked with steps. In order to edit an action, double-click the mouse on the step or select the step, and then select the menu command "Extras" "Zoom Action / Transition". In addition, only one input or output action is allowed in each step.

The action of the IEC step is attached in the Object Manager within the SFC-POU, which can be loaded by double-clicking or pressing the Enter key in its editor. You can also create a new action through "Project" "Add Action". Up to nine actions can be assigned to an IEC step.

Entry and exit actions

An additional entry and exit action can be added to a step. After a step is activated, an entry action can only be executed once. The exit action is executed only once before the step expires.

A step with an entry action is represented by an "E" in the lower left corner, and an exit action is represented by an "X" in the lower right corner.

Transition / Transition Condition

There are transitions between steps.

The value of the transition condition must be TRUE or FALSE, so it can be a Boolean variable, a Boolean address, or a Boolean constant. It can also include a sequence of commands with boolean results in structured text patterns (eg (I<=100) AND b) or in any desired language (see 'Extras' 'Zoom Action/Transition') . A transformation cannot include programs, function blocks, or assignments.

<Note> In addition to transitions, you can also use progressive mode to skip to the next step, see SFCtip and SFCtipmode.

Activation step

After calling the POU of the SFC, the actions of the initialization step (surrounded by a double-sided line) will be executed first. The step action being executed, the status is active, in

the online mode, the active step is displayed in blue. All actions of the active step in a control loop will be executed. So, when the transition condition after the active step is TRUE, the step after it is active. The currently active step will be executed again in the next cycle.

<Note> If the active step contains an output action, such as the transition condition below it is TRUE, then it can only be executed in the next cycle.

IEC step

Standard IEC steps can be used in SFC.

In order to use the IEC step, the Iesfc.lib library file must be linked in the project file.

An IEC step cannot be assigned more than 9 actions. The IEC action is not fixed as the action of input or output to a certain step like a simple step, but is stored separately from the step and can be reused many times in a POU. Therefore, they must be associated with a single step with the command "Extras Associate action".

In addition to actions, Boolean variables can also be assigned to steps.

Ability to use qualifiers to control active and inactive actions and boolean variables. There may be a time delay, and if an action is still active and the next step has already started processing, a concurrent process can be achieved by means of the qualifier S(set).

With each invocation of the sequential function block, the associated Boolean variable is set or reset, that is, with each invocation, the value will change back and forth between TRUE and FALSE.

The associated actions of an IEC step are represented in two rectangles to the right of the step, the left area contains qualifiers, possibly with time constants, and the right area contains the action name and the respective Boolean variable name.

Qualifier

To associate actions with IEC steps, use the following qualifiers

N	non-storage	Actions and steps are activated together
R	reset	Action is not active
S	set up	Action is activated and remains activated until reset.
L	time limit	The action is activated for a period of time, and the maximum value is the same as the step activation time
D	time delay	If the step is still active, the action is active after a certain amount of time, then it remains active as long as the step is active
P	pulse	If the step is active, the action is executed only once.
SD	storage and time delay	The action is activated after a certain time and remains activated until the next reset begins.
DS	delay and hold	The action is activated after a certain time as long as the step is still active and remains active until the next reset
SL	Hold and time limit	The action activates and stays for a while

The qualifiers L, D, SD, DS, and SL require a time value in TIME constant format.

<Note> When an action is deactivated, it will be executed again. This means that each action is executed at least twice.

Implicit Variables in SFC

Use some implicitly declared variables in SFC.

Each step has a marker, which stores the state of the step. For IEC steps, the step tag (active or inactive) is called <StepName>.x or <StepName> for a simple step. This Boolean variable has the value TRUE when the associated step is active, and FALSE otherwise. It can be used in every action and transition in the SFC module.

Whether an IEC step is active can be queried by querying <ActionName>.x.

The implicit variable <StepName>.t can be used to query the time of step activation.

Implicit variables can also be accessed by other programs, for example, boolvar1: =sfc1.step1.x; where step1.x is an implicit boolean variable, which represents the state of IEC step step1 in POUsfc1.

SFC identifier

The identifier controls the operation of the SFC POU, which is implicitly created during project operation. In order to be able to read these identifiers, appropriate global or local variables must be defined. For example, if a step in an SFC POU is active longer than its defined properties, then a flag is set, which can be accessed by using a "SFCError" variable (in which case SFCError gets true).

The following identifier variables can be defined:

SFCEnableLimit: The type of this variable is Boolean. When its value is TRUE, the timeout of this step will be registered into SFCError, and other timeouts will be ignored.

SFCInit: When the value of this Boolean variable is TRUE, the SFC is reset to the initial state, and other SFC identifiers are also reset. The initial step remains active until the variable value is TRUE, and execution begins. The module will work properly only when SFCInit is reset to FALSE.

SFCReset: It is similar to SFCInit, except that further processing takes place after initialization and the SFCReset flag can be reset to FALSE in the initialization step.

SFCQuitError: When it is TRUE, the execution of the SFC will be stopped, the timeout will be reset in the variable SFCError, when this variable is FALSE, all times in the active step will be reset, the precondition is that SFCError is predefined in the SFC, and register any time-out set identifiers.

SFCPause: When the value of this Boolean variable is TRUE, the SFC chart stops executing.

SFCError: This variable gets TRUE when there is a timeout in the SFC chart. If other timeouts occur after this, unless this variable has been reset, these states will not be recorded. If you want to use other time control flags (SFCErrorStep, SFCErrorPOU, SFCQuitError, SFCErrorAnalyzation) the precondition is to define SFCError.

SFCTrans: This boolean variable gets true when a transition is activated.

SFCErrorStep: This variable is a string variable. If a timeout is registered in SFCError, this variable will store the name of the timeout step. The precondition is that the flag SFCError to register any timeout has been defined in the SFC.

SFCErrorPOU: This string variable contains the name of the module where the timeout occurred. The precondition is that the flag SFCError to register any timeout has been defined in the SFC.

SFCCurrentStep: This string variable stores the name of the activated step, it has nothing to do with time monitoring. In the case of simulation, this step is stored in the branch of external authority. If a timeout occurs, the timeout will no longer be registered, and SFCError will not be reset.

SFCErrorAnalyzationTable: Array variable ARRAY [0..n], which provides the analysis result of a conversion expression. For each element that has an effect, the conversion is registered as FALSE, and the previous step time-out record, so the following information should be written into the ExpressionResult structure, written in: name, address, comment, current value.

It can hold up to 16 elements, so the range of the array is from (0~15).

The ExpressionResult structure and implicitly used analysis modules are provided by the AnalysisNew.lib library file, and the analysis modules can also be used explicitly by other POUs not written in SFC.

The timeout registration in the previous step is a prerequisite for parsing a conversion expression, so there must be a time watch applied here, and SFCError must be defined in the declaration window.

SFCTip, SFCTipMode: This boolean variable allows the progressive mode of the SFC. When used to toggle it with SFCTipMode=TRUE, it may only skip to the next step if SFCTip is set to TRUE, and it may skip the transition as long as SFCTipMode is set to FALSE.

Optional branch

Two or more optional branches can be defined in SFC, each branch must start and end with a transition. Alternative branches can contain parallel branches and other alternative branches, an alternative branch starts at a horizontal line and ends at a horizontal line (end of alternative), or a jump.

If the step before the alternative branch start line is active, the first transition of each alternative branch will be calculated from left to right, the first transition will be from the left transition condition is TRUE, then the following steps will be active .

Parallel branch

Two or more branches can be defined as parallel branches in SFC. Each parallel branch must have one step at the beginning and end. Parallel branches can contain optional branches or other parallel branches. A parallel branch begins with a double dash and ends with a double dash or a jump, which can provide a jump indicator.

If the previous step of the parallel branch is active, and the transition condition value after this step is TRUE, then the first step of the parallel branch is active. These branches are processed in parallel with each other. When all parallel steps are activated and the transition condition after these steps is TRUE, then the step after the end line of the parallel split is activated.

A jump is a link to the step name indicated under the jump symbol. Jumps must be used when creating upward or cross-connections is not allowed.

6.2.4 Functional block diagram(FBD)

A function block diagram is a graphics-based programming language that works with a series of networks, each network containing a structure that provides arithmetic or logical expressions, function block calls, jumps, or return instructions.

6.2.5 Continuous Function Chart(CFC)

The continuous function chart editor does not operate like a function block diagram, but elements can be placed freely, which allows the use of feedback.

6.3 On-line debug function

Sampling Track

Sampling tracking allows tracking the continuously changing value of a variable, depending on the trigger event, which is the rising or falling edge of a previously defined Boolean variable (trigger variable). CODESYS allows tracking of 20 variables, each of which can track 500 values.

Debug

The debugging capabilities of CODESYS make errors easy to find. In order to debug, run the command "Project" "Options", and select the activated option in the automatically pop-up dialog box of creation options.

Breakpoint

A breakpoint is a place where program processing stops, so it can observe changes in variable values at specific points in the program.

The breakpoint can be set in the editor, in the text editor the breakpoint is set at the line number, in the continuous function diagram and the ladder diagram at the network number, in the CFC at the POU, in the SFC It is set at the step, and the breakpoint cannot be set in the instance of the function block diagram.

Single step

In the instruction list: Execute the program until the CAL LD and JMP commands are run.

In structured text: Execute the next instruction.

In the function block diagram ladder diagram: Execute the next step network.

In SFC: Continue the current action until the next step starts.

Logic errors in the program can be checked by running it step by step.

Single cycle

If single loop is selected, execution ends when each loop ends.

Change value in online mode

During operation, variables can be set to a specific value (write value), or redefined to a specific value (force value) after each loop. In online mode, the value of a variable can be changed by double-clicking on its value, and a Boolean variable can be changed from TRUE to FALSE, or from FALSE to TRUE. The Write Variable xy dialog box opens for each type of variable, where the actual value of the variable can be edited.

Monitor

In online mode, all variables can be read from the controller and displayed in real time. These displays can be found in the Definition and Program editors. The current values of variables can also be read out in watchers and receivers, and they can be seen. If you want to monitor variables in an instance of a function block, the corresponding instance block must already be open.

Simulation

In the simulation process, the created PLC program does not run in the actual PLC, but runs in the calculator in the CODESYS system. All online functions are available. It allows checking the correctness of logic without PLC hardware.

<Note> POUs of external library files cannot be run in emulation mode.

Log

The log records user operations, internal processes, state transitions, and unexpected situations that occur during online mode processing. It is used to monitor and track errors.

6.4 Ladder Diagram(LD)

Ladder diagram is also a graphics-based programming language, which is close to the structure of electronic circuits. On the one hand, ladder diagram is very suitable for building logic switches. On the other hand, it can also create network diagrams like FBD. It is suitable for controlling when calling other POUs.

The ladder diagram contains a series of networks. There is a bus on both sides. The network diagram is limited to the range between the left and right buses. In the middle is a circuit diagram composed of coil contacts and connecting lines.

Each network contains a series of contacts on the left. These contacts pass on and off state from left to right according to TRUE and FALSE of Boolean variable value. Each contact is a Boolean variable. If the value of the variable is TRUE, the circuit is connected from left to right through a connecting line. Otherwise, the right side receives the "OFF" value.

Contact

There are contacts on the left side of each network diagram in the ladder diagram (contacts are represented by two parallel lines | |), which are used to indicate the "on" and "off" states of the circuit.

These states correspond to Boolean variables TRUE and FALSE. If the variable value is TRUE, then the state can be passed from left to right by connecting lines. Otherwise, "disconnect" is received on the right.

Contacts can be used in parallel, one of the parallel branches must pass the "open" state, the parallel branch can pass "on". Or the contacts are connected in series. At this time, when the contacts must transmit the "open" state, the last contact transmits "open", which is consistent with the circuit connected in series and parallel.

Coil

There are some coils on the right side of the ladder network diagram, which are represented by ( ) and can only be connected by horizontal lines. The coil transmits the connection state from left to right, and copies the state into a boolean variable, which can describe the state of the entry wire as "on" (corresponding to TRUE for the boolean variable) or "off" state (corresponding to the boolean variable FALSE).

Contacts and coils can also be negated (contact SWITCH1 and coil %QX3.0 in the example above are negated). If the contact value is negated (represented by "/" in the contact symbol), then it copies the negated value to the corresponding boolean variable. If a contact is negated, the circuit can be connected only when the corresponding Boolean variable is FALSE.

Function Blocks in Ladder Diagram

Function blocks and programs can be added to the network diagram, but they must have Boolean-valued inputs and outputs, and can be used on the left side of the ladder diagram like contacts.

Contact setting/resetting

Contacts can be defined as set/reset contacts. A set contact is represented by an "S" in the contact symbol, and it never overrides the TRUE value in the corresponding Boolean variable, that is, if the variable is set to TRUE, it will remain in the same state.

The reset contact is represented by "R", it never overwrites the FALSE value in the corresponding Boolean variable, and if the variable has been set to FALSE, it will remain in this state.

When using the ladder diagram, the result of the contact switch can be used to control other POUs. On the one hand, the coil can be used to output the result to a global variable, which can be used elsewhere. It is also possible to insert possible calls directly in the Ladder diagram by introducing a POU with EN input.

These POUs are complete normal operands, functions, programs or function blocks. They all have an additional input designator EN. The EN input is a boolean variable, a POU with an EN input will only be counted if the value of EN is TRUE

Communication example

7

7.1 Application Example of Communication

C5X-10 motion controller supports bus, CANopen, EtherCAT, EtherNET and other communication methods. This section will introduce the communication methods supported by C56-10 through specific examples.

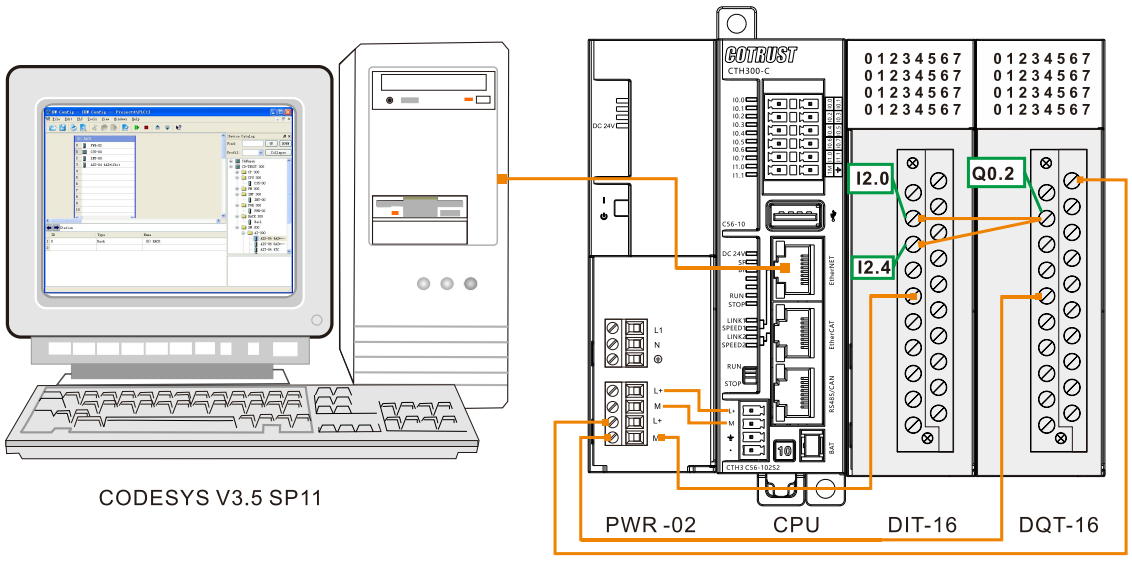
7.1.1 Bus Communication

1. Preparation before communication

Table 7-1 Example Components for Bus Communication

Components	Function
Programming device PG\PC	CODESYS V3.5 SP11 for configuration, programming and commissioning of the C56-10
Assembly rail	C56-10 rack, used to hold the modules in the system
Power Module PWR-02	Powers the C56-10 motion controller and its 24 VDC load circuit
C56-10 main control module	The CPU executes the user program, provides 5V voltage to the C56-10 backplane bus, and communicates with other modules through the Ethernet interface
Expansion module SM	The SM converts the different process signal levels to match the C56-10.
Standard network cable	In this example, the digital input module DIT-16 and the digital output module DQT-16 are used

2. Network connection



Implement function

The output Q0.2 is lit, and the inputs I2.0 and I2.4 are lit at the same time.

3. Operation steps

Step 1: Wiring

Open the front panels of the C56-10, power supply module, digital input (DIT-16), and output module (DQT-16) and wire them by referring to the communication connections in this section. The specific operations are as follows:

1) Use the programming cable to connect the PC and C56-10

2) DIT-16 is connected to C56-10 by bus

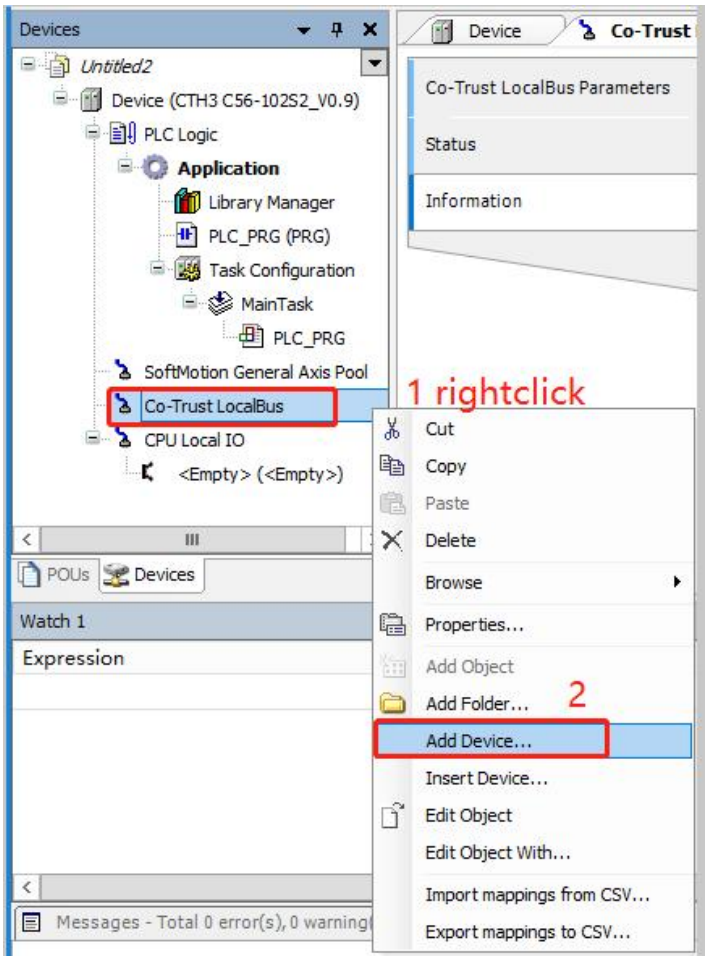
3) DQT-16 is connected to DIQ-16 by bus

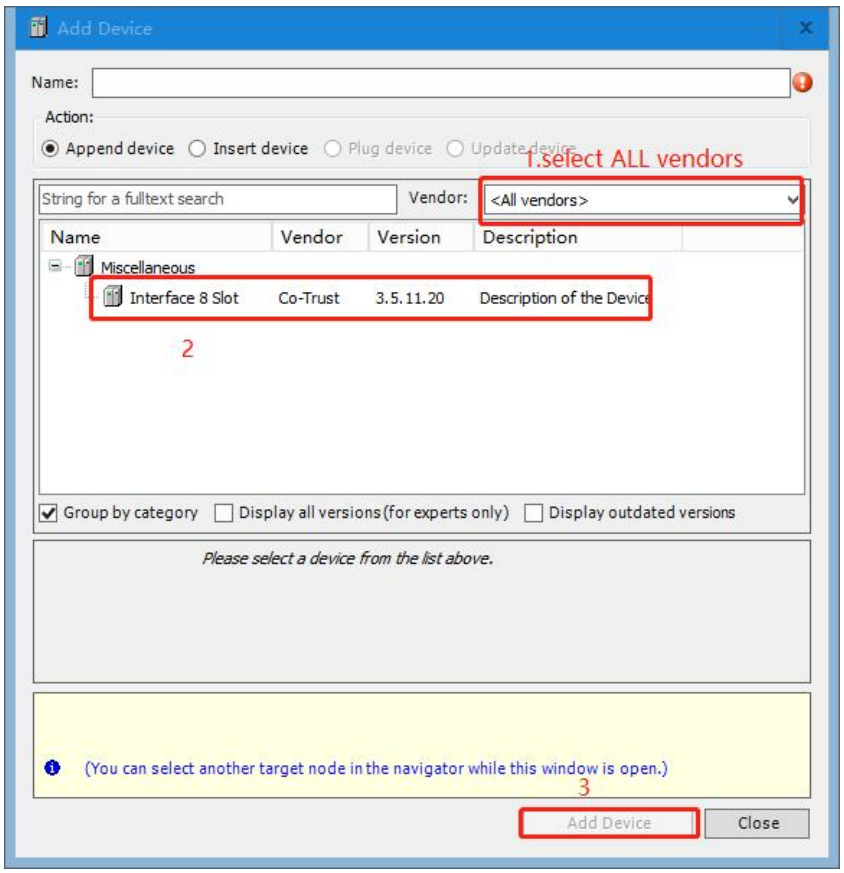
Warning

Do not touch the power cable after the power module is powered on or after connecting the power cable to the main power supply, and any wiring work must be carried out under the condition of power off!

Step 2: Configuration in CODESYS

1) Select "Co-trust LocalBus" in the device view and right-click to select "Add Device", you can select and add a Intermediate expansion module in the pop-up dialog box, refer to the following diagram:



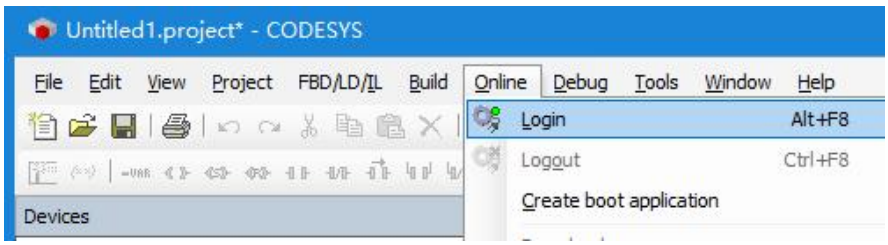


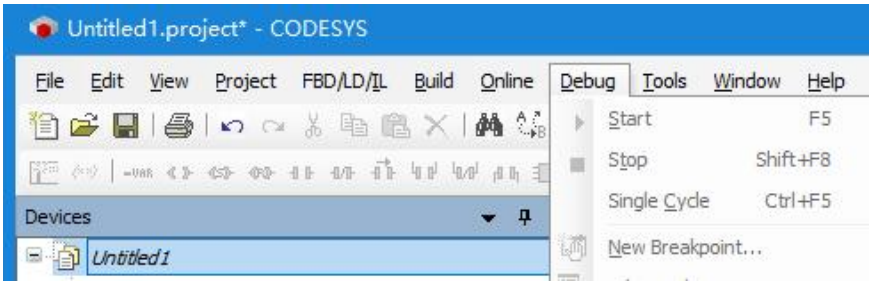
2) Referring to the above operations, right-click the intermediate expansion module in the device view and select "Add Device", then in the "Add Device Dialog Box" you can select to add extension modules DIT-16 and DQT-16. Or right-click the intermediate expansion module and select "Scan Device", and the expansion modules connected to the C56-10 will be displayed under the intermediate expansion module.

Step 3. Refer to the chapter 2.3 Set up communication to set the communication between C56-10 and the host computer

Step 4. Running, debugging and monitoring

1) Select "Online" → "Login ..." to establish a connection between the application and the C56-10 and enter the online state. Then, select "Debug" → "Start" to make the application program in C56-10 start to run, at this moment, the current project can be monitored and debugged.





2) Write 1 to %QX0.2 in the Internal I/O mapping table, then the output Q0.2 of the DQT-16 module will be lit. At the same time, the inputs I2.0 and I2.4 of the DIT-16 module are lit, and the values of %IX2.0 and %IX2.4 are changed to 1 in the Internal I/O mapping table of the DIT-16 module.

7.1.2 Modbus RTU Communication

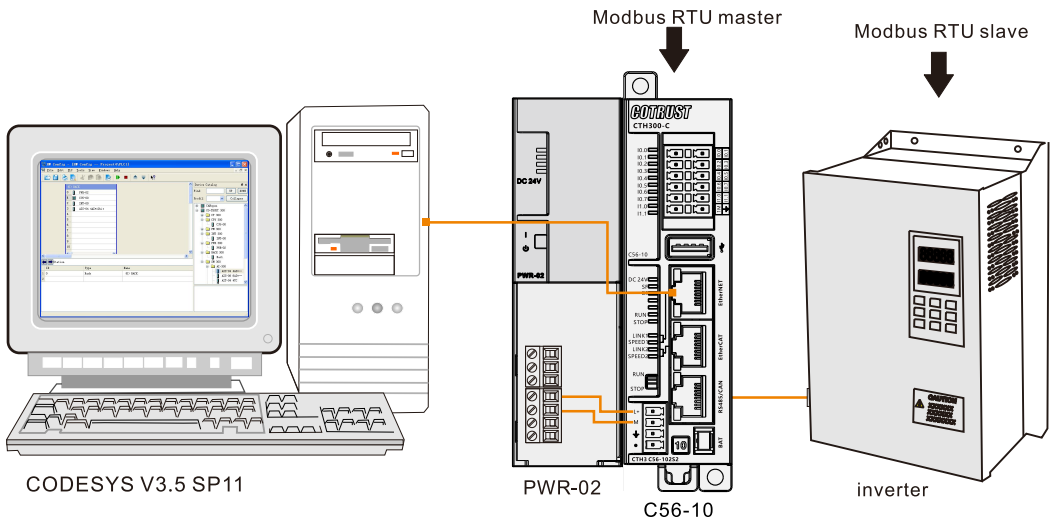
1. Preparation before communication

Table 7-2 Example Components for Modbus RTU Communication

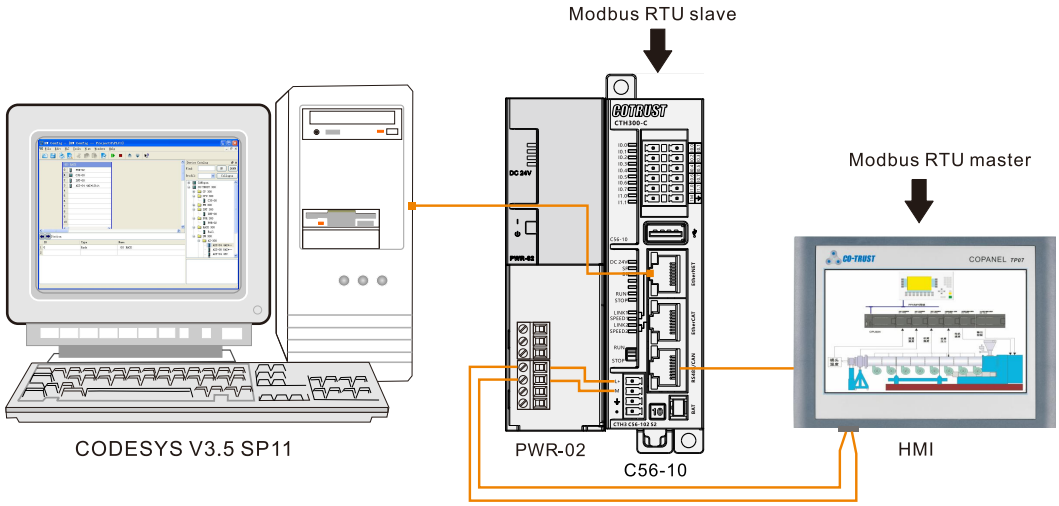
Components	Function
Programming device PG\PC	CODESYS V3.5 SP11 for configuration, programming and commissioning of the C56-10
Power Module PWR-02	Powers the C56-10 motion controller and its 24 VDC load circuit
C56-10 main control module	The CPU executes the user program, provides 5V voltage to the C56-10 backplane bus, and communicates with other modules through the Ethernet interface.
Copanel HMI	Copanel series HMI, as Modbus master station to communicate with C56-10
Inverter	Inverter that supports Modbus protocol, communicates with C56-10 as Modbus RTU slave station
Standard network cable	In this example, the digital input module DIT-16 and the digital output module DQT-16 are used.

2. Network connection

◆ C56-10 communicates with the inverter as a Modbus RTU master



◆ C56-10 communicates with Copanel HMI as Modbus RTU slave



3. Operation steps

Step 1: Wire the C56-10, Power Module, HMI and Drive

Refer to the above network connection diagram for wiring of C56-10, power module PWR-02, Copanel HMI and inverter.

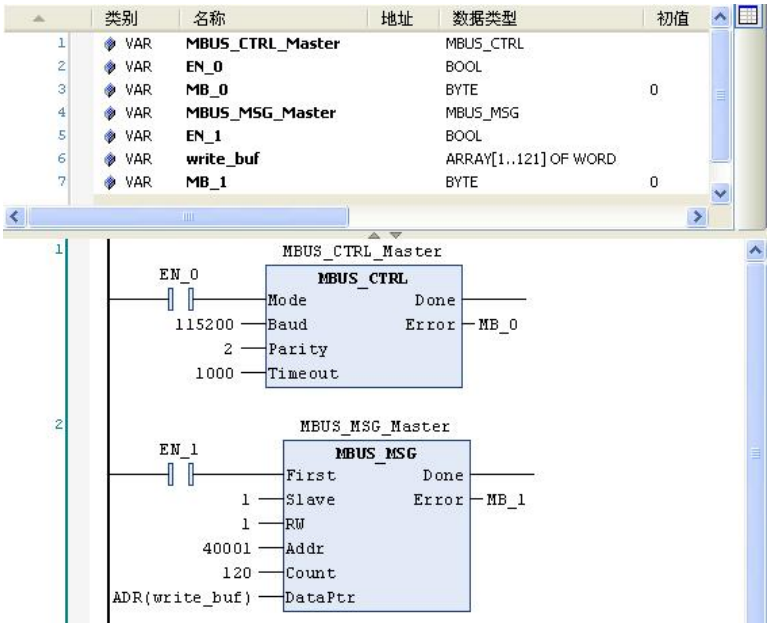
Step 2: Connect the cables

1) Use standard network cable to connect PC and C56-10.

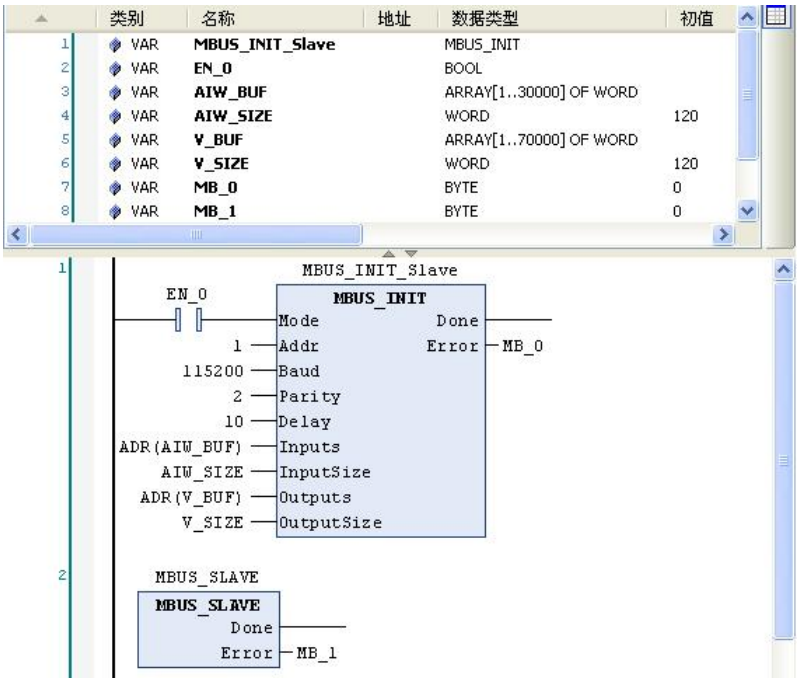
2) When the C56-10 is used as the Modbus RTU master to communicate with the inverter, use the communication cable to connect the RS485 communication port of the C56-10 to the Modbus communication port of the inverter.

3) When C56-10 communicates with Copanel HMI as a Modbus RTU slave station, use a communication cable to connect the RS485 communication port of C56-10 to the Modbus communication port of Copanel HMI. 1) Use standard network cable to connect PC and C56-10.

Step 3: When the C56-10 is communicating with the inverter as a Modbus RTU master, program the C56-10 as follows:



Step 4: When C56-10 communicates with Copanel HMI as Modbus RTU slave, the program for Modbus slave (C56-10) is as follows:



7.1.3 Modbus TCP Communication

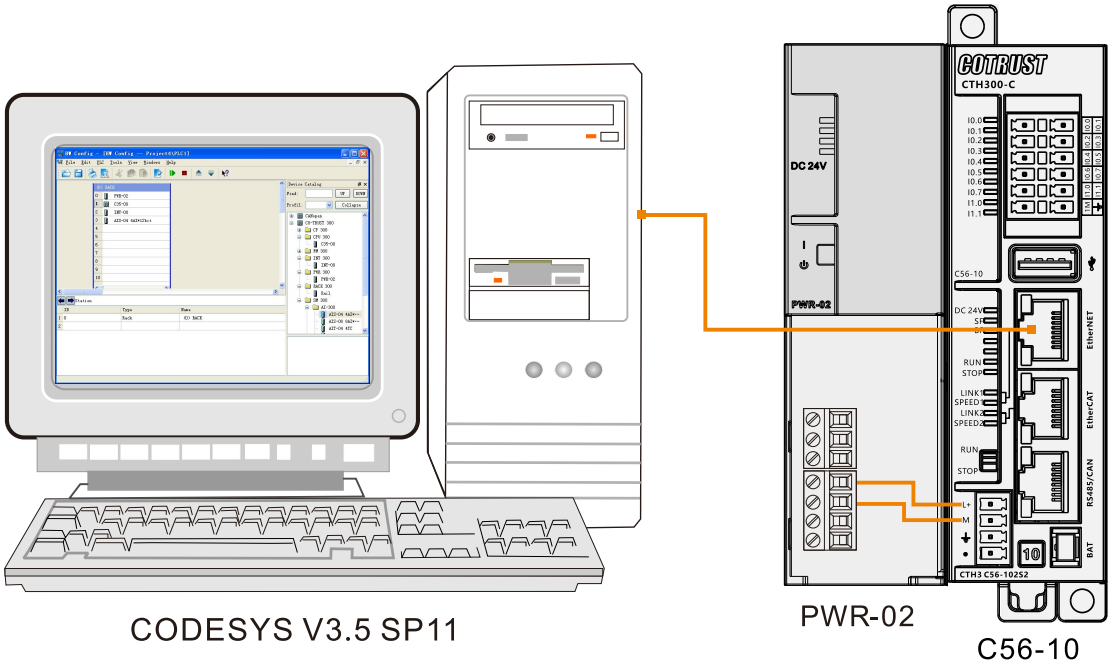
This section will demonstrate the Modbus TCP communication function of C56-10 motion controller through a concrete example.

1. Preparation before use

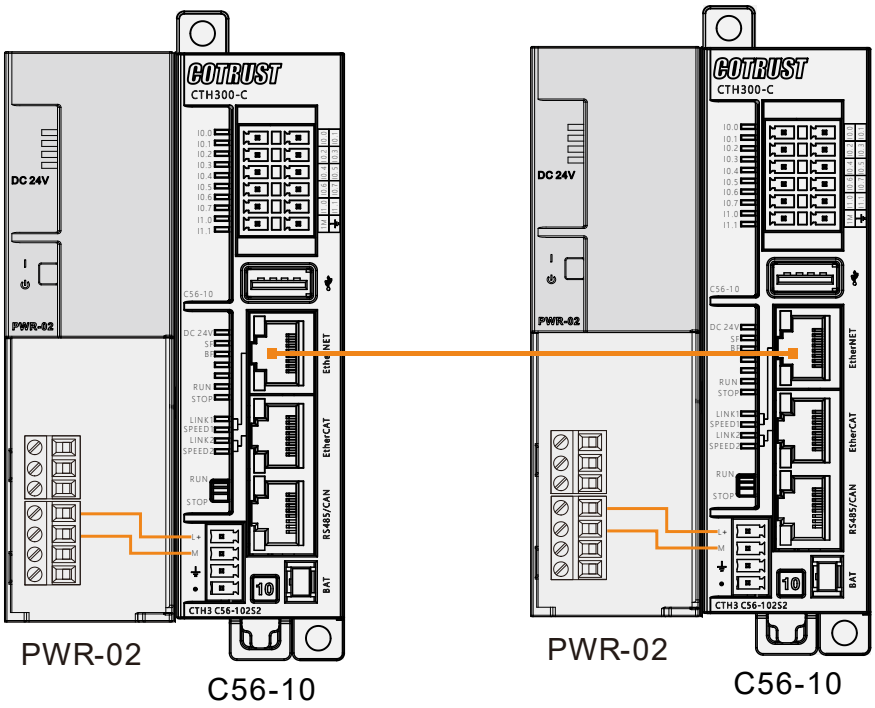
Table 7-3 Example Components for Modbus TCP Communication

Components	Function
Programming device PG\PC	A programming device with CODESYS V3.5 SP11 is installed to configure, program and debug the C56-10 programmable controller.
Power Module PWR-02	Powers the C56-10 motion controller and its 24 VDC load circuit
C56-10 main control module	Execute the user program, supply 5 V to the bus, and communicate with other modules via the Ethernet interface.
Standard network cable	connect C56-10 and programming device

2. Network connection



Use a standard network cable to connect two C56-10s as ModbusTCP master and ModbusTCP slave, and then Modbus TCP communication can be performed:



3. Operation steps

Step 1: Wire the C56-10 to the Power Module

Open the front panel of C56-10 and power supply module PWR-02, and then connect them according to the above network connection diagram.

Step 2: Connect the device

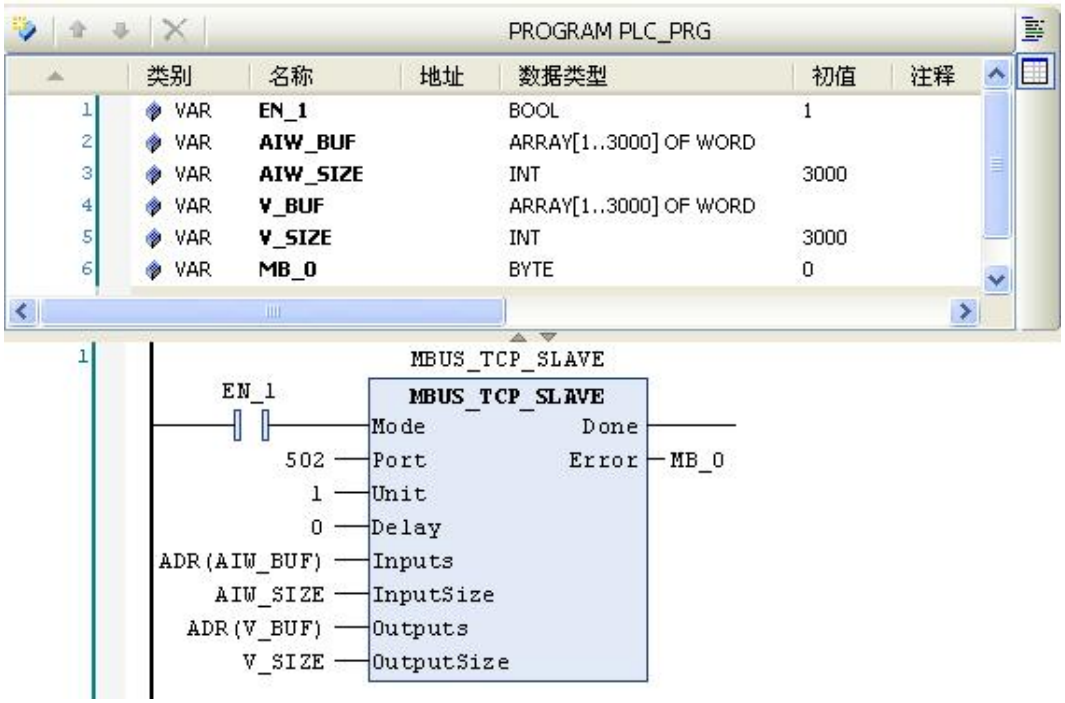
Use a standard network cable to connect the programming device and the ModbusTCP slave.

3: Program the Modbus TCP Slave C56-10

1) Enter a command in the PLC shell, such as "setip 192.168.0.1" and press ENTER to set the IP address.

2) Program the Modbus TCP slave in PLC_PRG (PRG)

The following instructions are the "Modbus_TCP_Slave" library instructions in the "CO-TRUST MODBUS LIBRARY" library file:

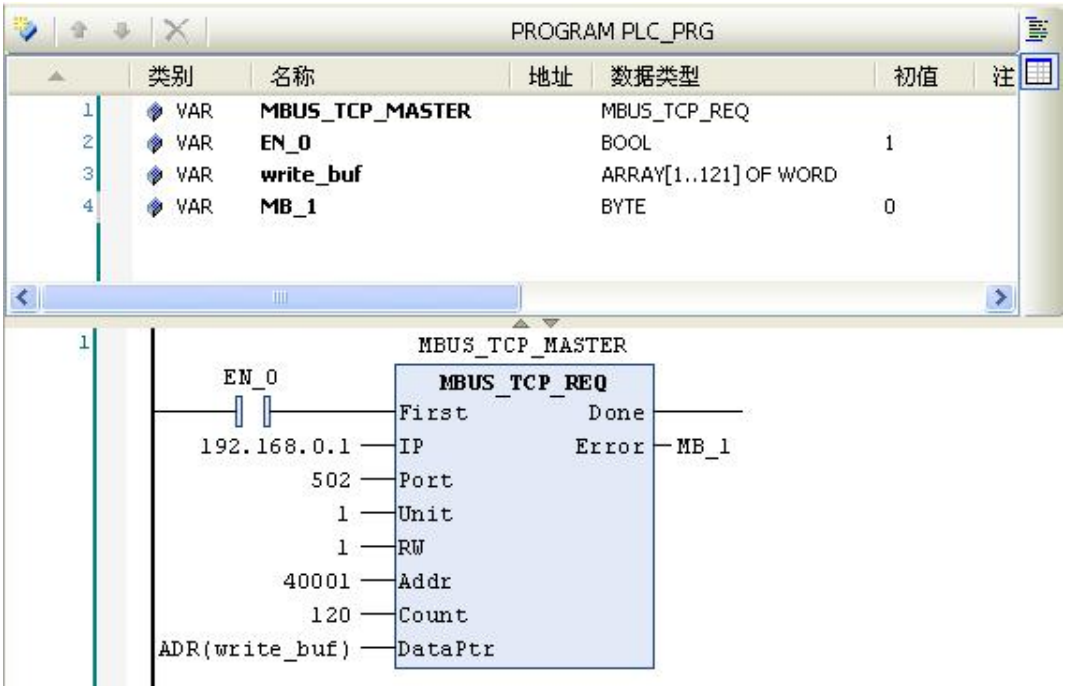


3) Compile and download the program to the Modbus TCP slave device

Step 4: Program the ModbusTCP Master C56-10

1) Program the Modbus TCP master in PLC_PRG (PRG)

The following instructions are the "Modbus_TCP_REQ" library instructions in the "CO-TRUST MODBUS LIBRARY" library file.



Step 5: Modbus TCP master and Modbus TCP slave communicate with each other

1) Power on the Modbus TCP master and slave

2) Use a standard network cable to connect the Modbus TCP master station and the slave station, that is, they start Modbus TCP communication.

7.1.4 CANopen Communication

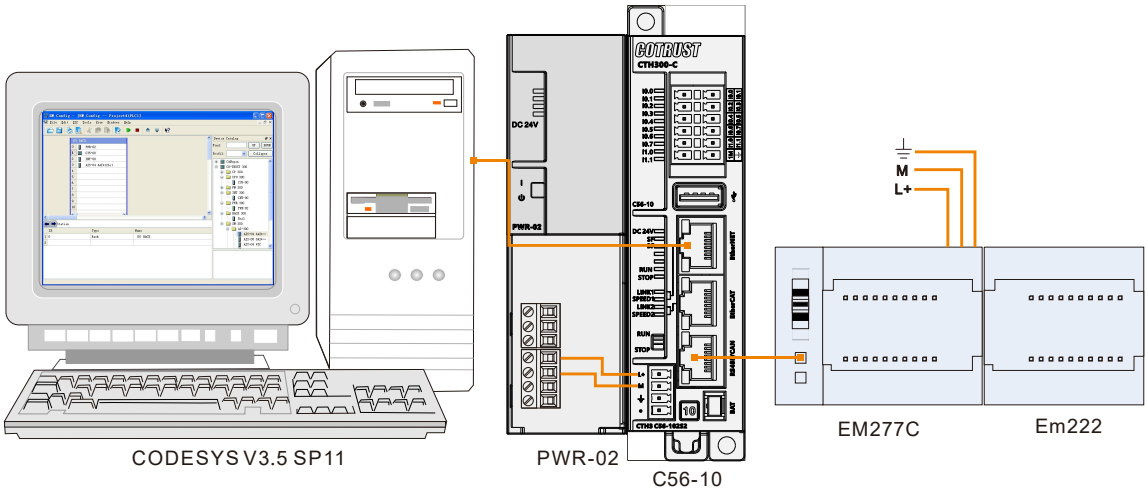
This section will demonstrate the CANopen communication function of C56-10 motion controller through a concrete example.

1. Preparation before communication

件 Table 7-4 Example components for CANopen communication

Components	Function
Programming device PG\PC	The CODESYS V3.5 SP11 software is installed to configure, program and debug the C56-10 motion controller.
Power Module PWR-02	Powers the C56-10 motion controller and its 24 VDC load circuit
C56-10 main control module	The CPU executes the user program, supplies 5V to the bus, and communicates with other modules through the Ethernet interface.
EM277C	as CAN slave
Expansion module for EM277C	Expansion module of CAN slave, mounted behind EM277C This example uses: digital module EM222-1HH32
Standard network cable	Connecting the programming device to the C56-10 Connect C56-10 with CAN Slave

2. Network connection



3. Operation steps

Step 1. Power on each device

Open the front panels of the power modules PWR-02, C56-10 and EM277C, and then connect the power to them by referring to the CANopen communication connection method.

Step 2. Connect the cable

Refer to the CANopen communication connection method to connect each device, the specific operation is as follows:

1) Use a standard network cable to connect the PC and C56-10

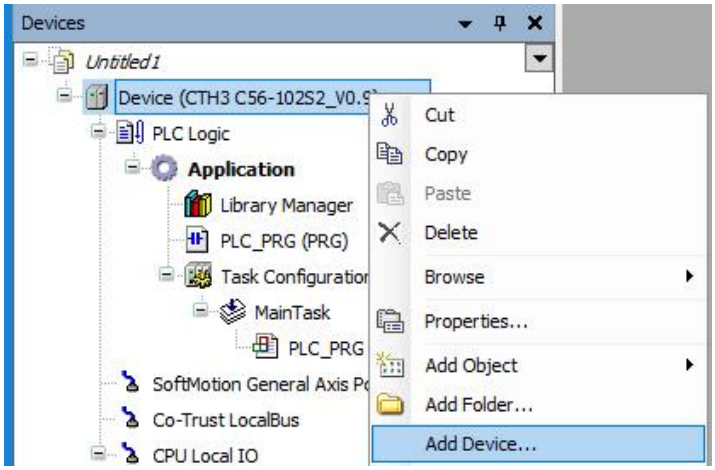
2) Use a standard network cable to connect C56-10 and EM277C

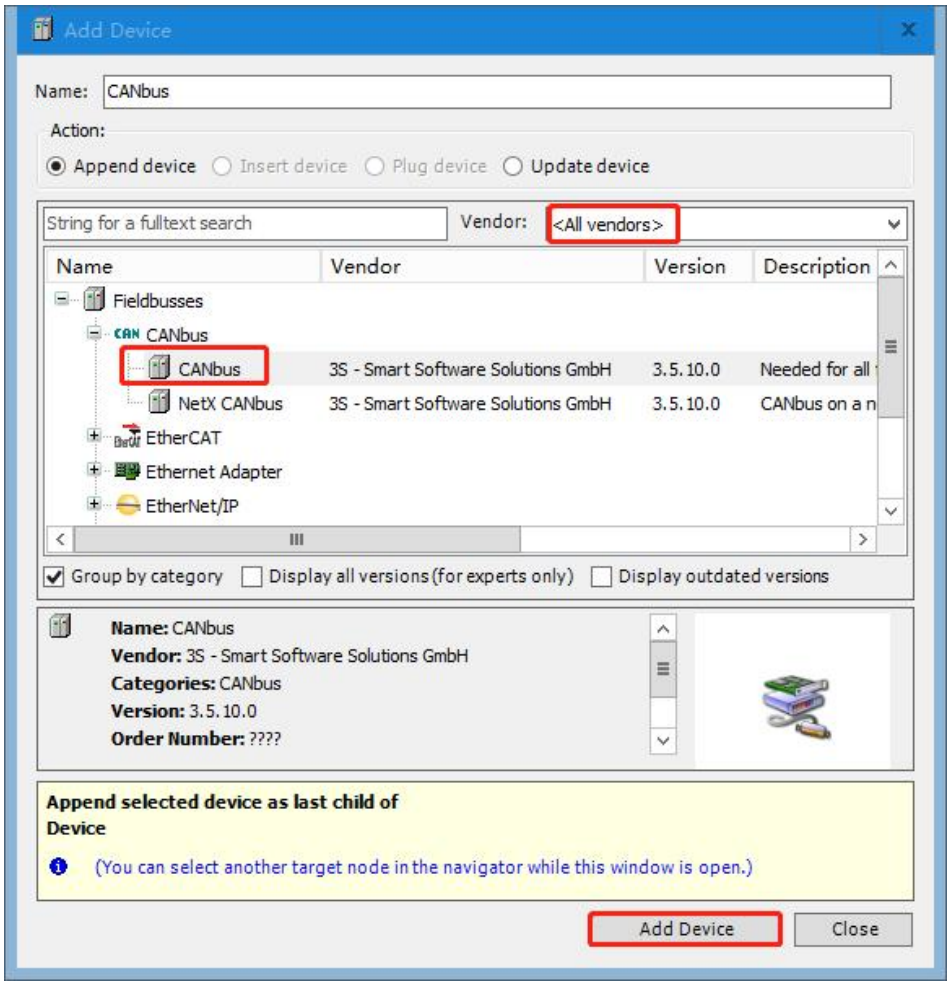
3) The digital quantity module EM222 is connected to the EM277C through the bus

Step 3. Configure CANopen in CODESYS

1) Add CANbus

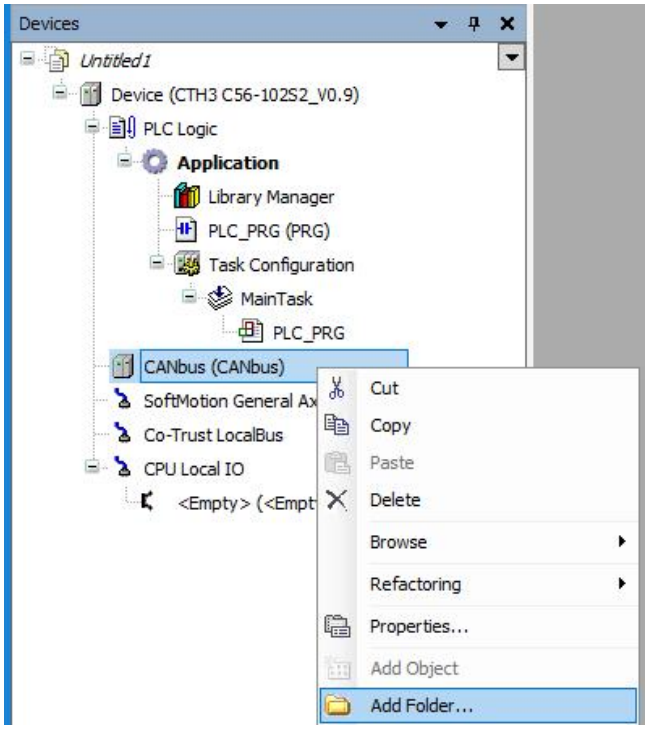
Right-click on "Device(CTH3 C56-10S2_V0.9)" in the device view and select "Add Device", you can choose to add CANbus in the pop-up dialog box: Supplier select "<all suppliers>", Fieldbus select " CANBUS" as the CAN bus.

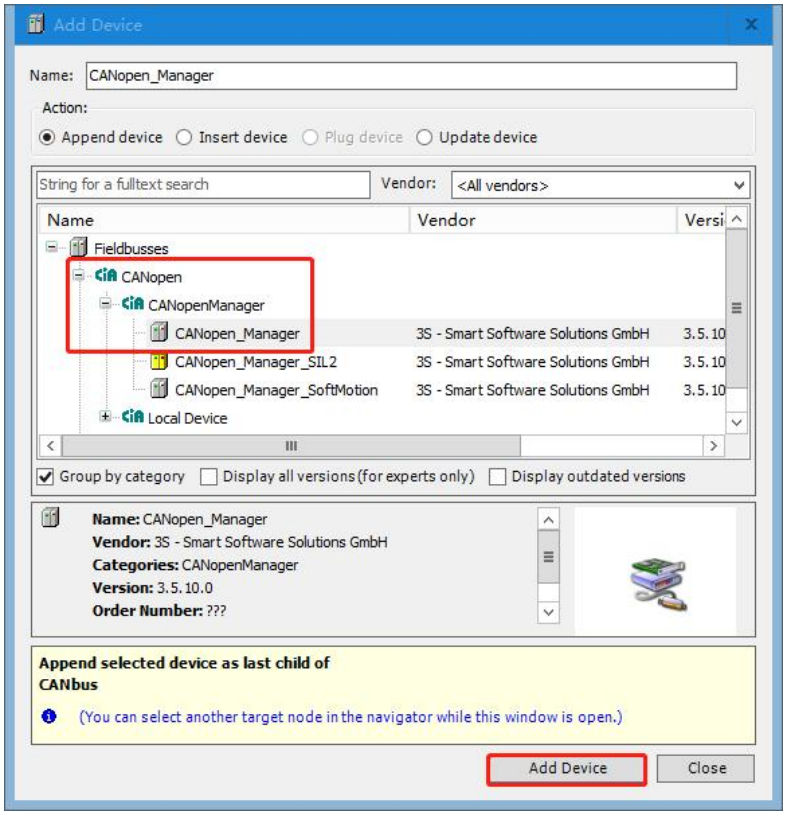




2) Add CAN master

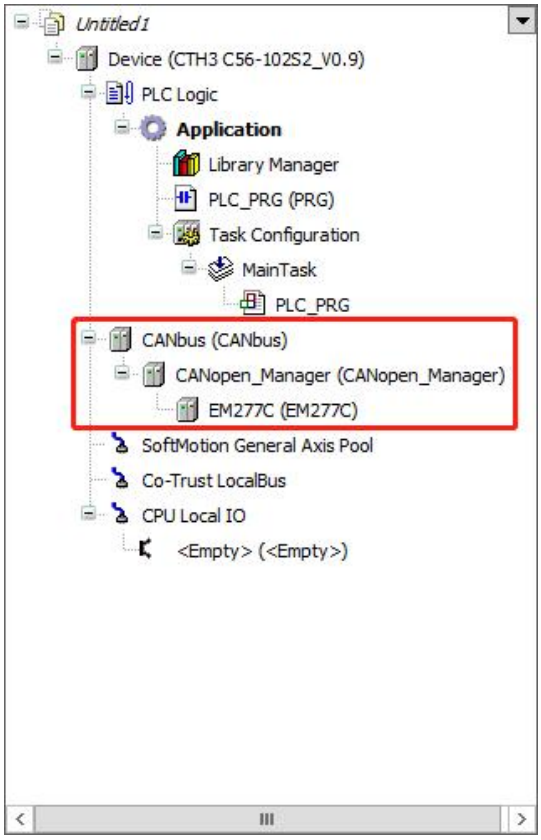
Select the successfully added "CANBUS" in the device view and right-click to select "Add Device" to add a CAN master in the "Add Device" dialog: Fieldbus → CANopen → CANopen Manager→CANopen_Manager.





3) Add or scan CAN slaves

Select the successfully added "CANopen_Manager" in the device view and right-click to select "Add Device" to add a CAN slave in the "Add Device" dialog: Fieldbus→CANopen→Remote Device→EM277C, or right-click "CANopen_Manager" "Select "Scan Device", the slave device (EM277C) will be displayed under the CAN master.



4) CANbus settings

CANbus Tab: In this example "Network" is set to 0 and "Baud Rate" is set to 1000000.

5) Configure CAN master

The NODE_ID in the CANopen_Manager master is set to 127.

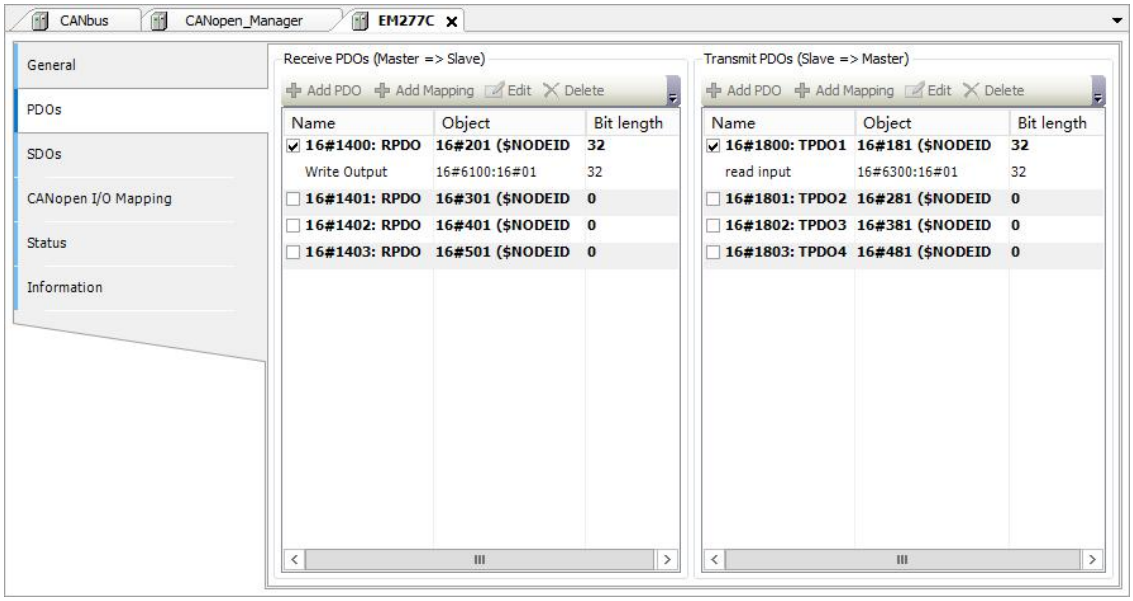
6) Configure CAN slave (EM277C)

"General" tab: Set the node ID to 4 and check "Enable Expert Settings".

<Remarks> It is necessary to set the DIP switch of EM277C to 4. For the specific configuration method, please refer to "CANopen User Manual".

"Server Data Object": Newly add 16#01 parameter of 16#2200.

"PDO" tab: Check the parameter groups to be displayed in the "Receive PDOs (Master => Slave)" and "Transmit PDOs (Slave => Master)" configuration boxes.



<Remark> Please refer to Appendix F.2 Configuring CANopen Slave Devices for more details on CANopen slave configuration.

Step 4. Refer to chapter 2.3 Set up communication to set the communication between C56-10 and the host computer

Step 5. Running, debugging and monitoring

1) Select "Online" → "Login" to establish a connection between the application and the C56-10, and enter the online state. Then, select "Debug" → "Start" to make the application program in C56-10 start running, at this moment, the current project can be monitored and debugged.

2) Debug the connected module in the tab "CANopen I/O Mapping" of the EM277C slave (check "Always in the bus cycle task"), in this example, the digital module EM222 is selected, then click Turn on its output Q0.5.

Result: After Bit13 writes TRUE, Q0.5 of the EM222 module is lit.

7.1.5 Single-axis motion control based on CANopen communication

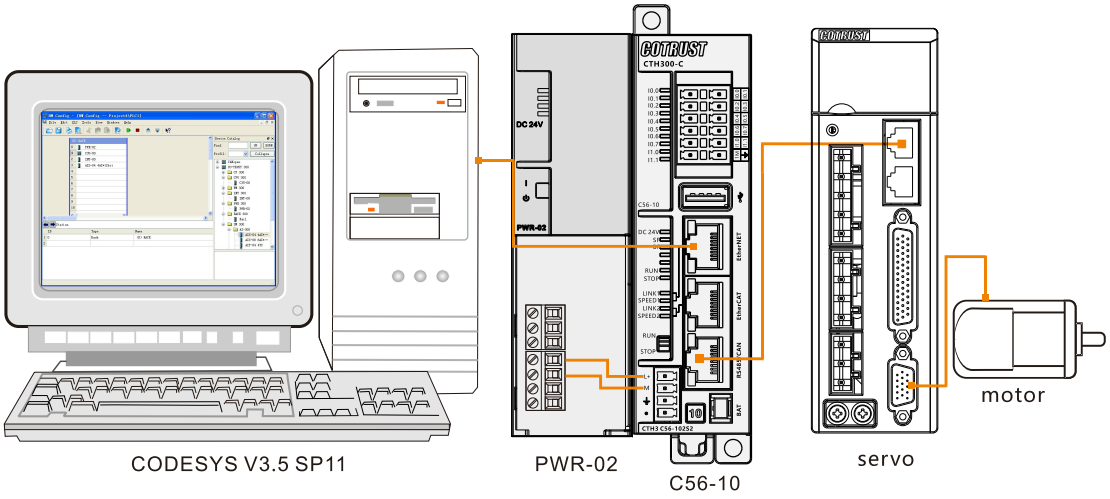
1. Preparation before communication

Table 7-5 Example components for CANopen communication

Component	Function
Programming device with CODESYS V3.5 SP11 installed	Configure, program and debug C5X-10 series motion controllers
Assembly rail	Used to secure the modules in the system.
Power Module PWR-02	Provides the C5X-10 motion controller and its 24 VDC load circuit.
External power supply	Provides working power for A4S servo drives.
C56-10 motion controller	Execute the user program, supply 5 V to the bus, and communicate with other modules via the Ethernet interface.

A4S servo driver and motor	This example uses A4S servo driver and supporting motor
Standard network cable	Connecting the programming device to the C56-10
Encoder cable	Connect A4S Servo Drive and Motor

2. Network connection



3. Operation steps

Step 1. Power on each device

1) Open the front panel of C56-10 and power supply module PWR-02, and then connect C56-10 to PWR-02 according to the above connection diagram.

2) Connect the main power and control power for the A4S servo drive.

Step 2. Use cables to connect each device

Network connection diagram to connect each device, the specific operation is as follows:

1) Use standard network cable to connect PC and C56-10.

2) Use standard network cable to connect C56-10 and A4S servo drive.

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

Step 3. Configure CANopen in CODESYS

1) Add CANbus

In the device view, right-click "Device (CTH3 C56-10S2_V0.9)" and select "Add Device", then you can choose to add CANbus as the CAN bus.

2) Add CAN master

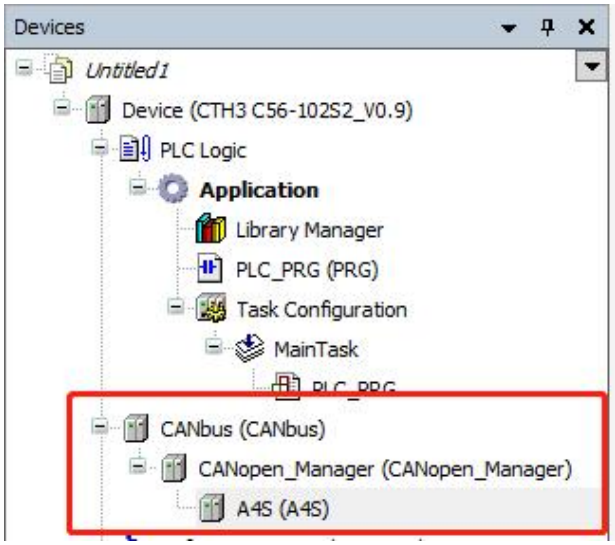
Right-click "CANbus" in the device view, select "Add Device" and then add a CAN master in the "Add Device": Fieldbus→CANopen→CANopen Manager→CANopen_Manager.

3) Add or scan CAN slaves

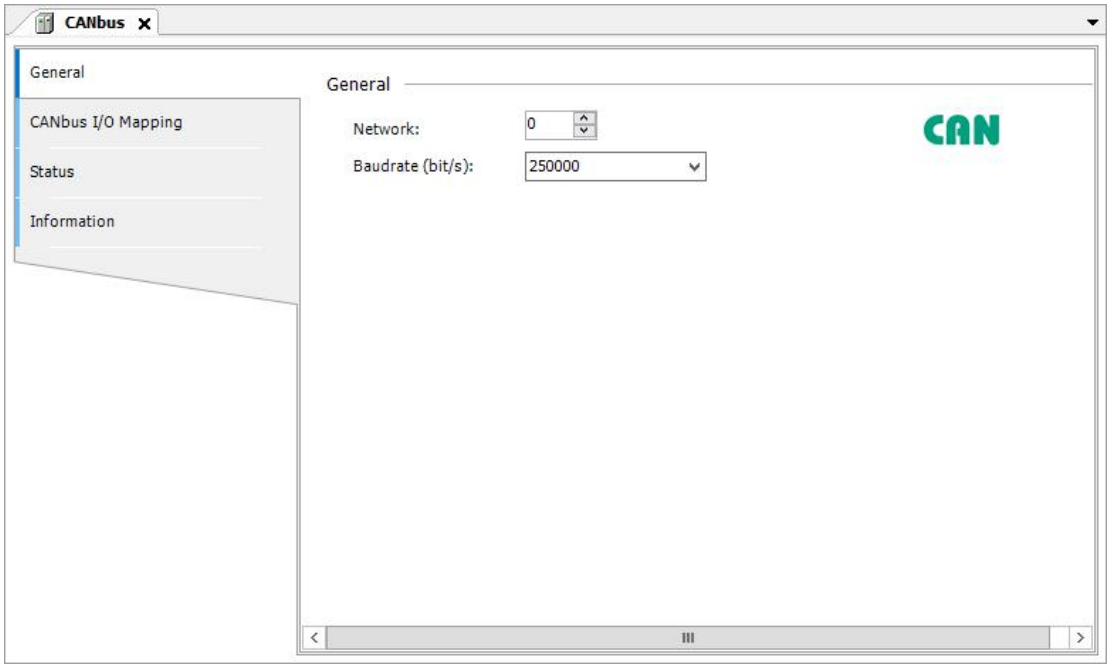
Right-click on "CANopen_Manager" in the device view, and select "Add Device" in the pop-up menu item to add a CAN slave in the "Add Device" dialog box: Fieldbus→CANopen→Remote Device→A4S, or right-click CANopen_Manager select "Scan Device" to display the CAN slave (A4S) below the CAN master.

4) Add driver for CAN slave (A4S)

If the A4S is added as a CAN slave in step 3), you can add a driver for the A4S here: Select the CAN slave (A4S) in the device view and right-click, and select "Add SoftMotion CiA 402".

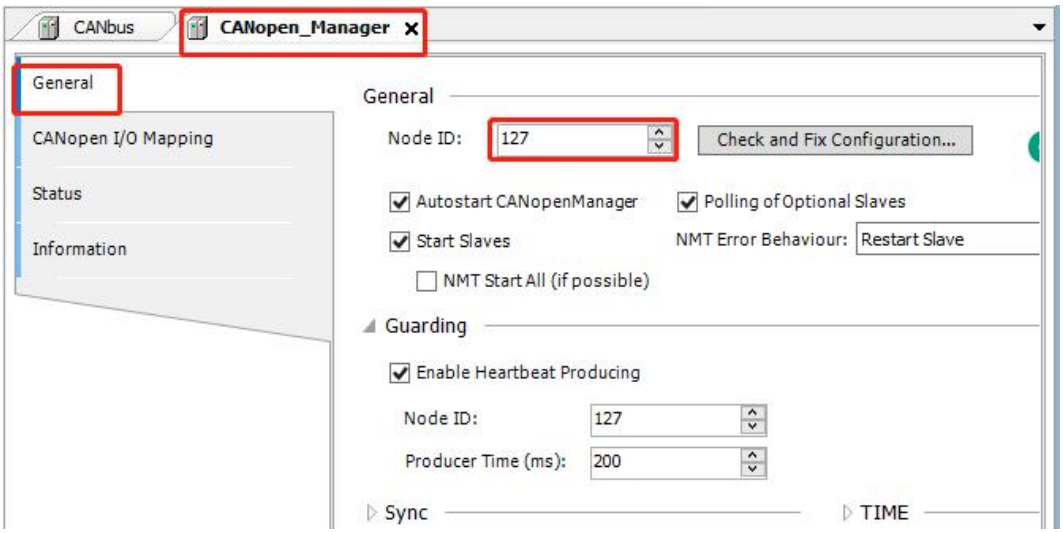


5) Configure CANbus



6) Configure CANOpen master(CANOpen manager)

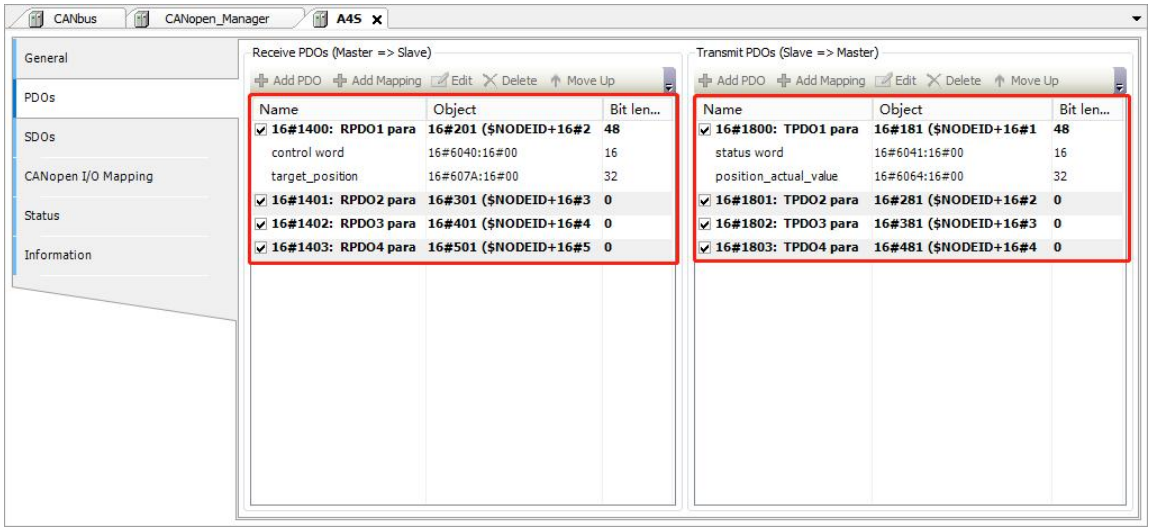
Enter the node ID of the CAN master in the "Overview" of the master: 127.



7) Configure CANopen slave (A4S)

First, set the node ID of the slave (ie the address of the slave A4S) in the "Overview" tab, and then check "Enable Expert Settings".

Then, select the parameters you want to use in the "PDOs" tab, in this case the settings are as shown below:

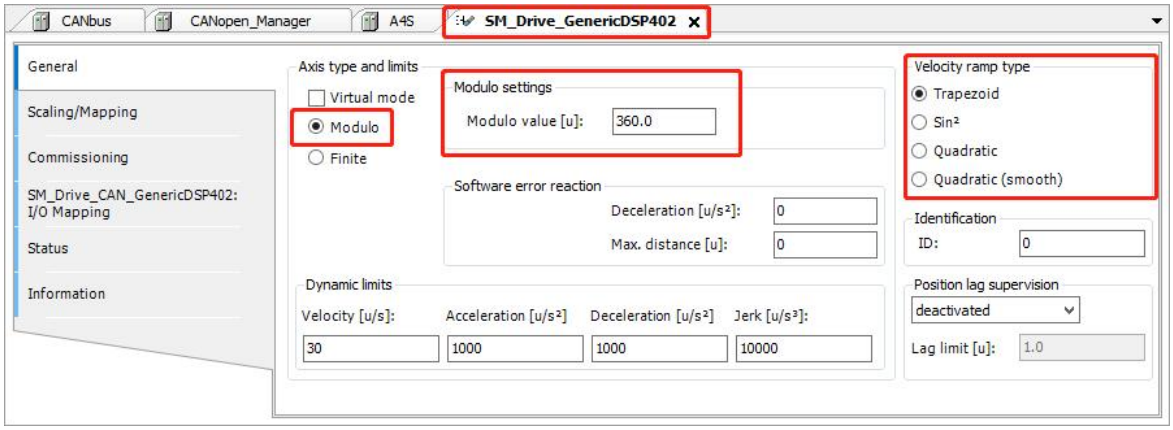


In the "CANopen I/O Mapping" tab, tick "Always in Bus cycle task".

<Remark> Please refer to Appendix F.2 Configuring CANopen Slave Devices for more details on CANopen slave configuration.

8) Configure A4S driver (SM_Driver_GenericCiA402)

The "Softmotion Driver: Basic" tab of SM_Driver_GenericCiA402 is set as follows:



The Axis type and limits are selected as "Modulo", the Modulo value is set to 360.0, and the velocity ramp type is selected as trapezoid.

Step 4. Refer to the chapter 2.3 Set up communication to set the communication between C56-10 and the host computer

Step 5. Running and debugging

1) Refer to the *A4S User Manual* to configure the parameters of the current A4S servo slave.
Set the control mode and CANopen baud rate through the front panel of the A4S servo drive: P01=1 (communication position control mode), P11=1 (CANopen baud rate is set to 1000Kbps).

2) Select "Online" → "Login" to establish a connection between the application and the C56-10 and enter the online state, then, select "Debug" → "Start" to start the application in the C56-10 run.

3) Set P97, P290 and P50 in the "CANopen I/O Mapping" tab of the A4S. Among them, P97 sets the running speed of the motor, P290 sets the given position size, and P50 (communication mode command interpolation mode selection) selects whether the motor runs at the speed set by P97 or at the internal speed of the controller. P50 in this example is set to 0, that is, choose to run at the speed set by P97.

4) Double-click "SM_Drive_GenericDSP402" in the device view to monitor the operation of the motor in its "Softmotion Drive" tab.
Result: The servo motor connected to the A4S servo driver executes the given position 10000 according to the speed (500) set by P97.

7.1.6 EtherCAT Communication

This section will demonstrate the EtherCAT communication function of C56-10 through a concrete example.

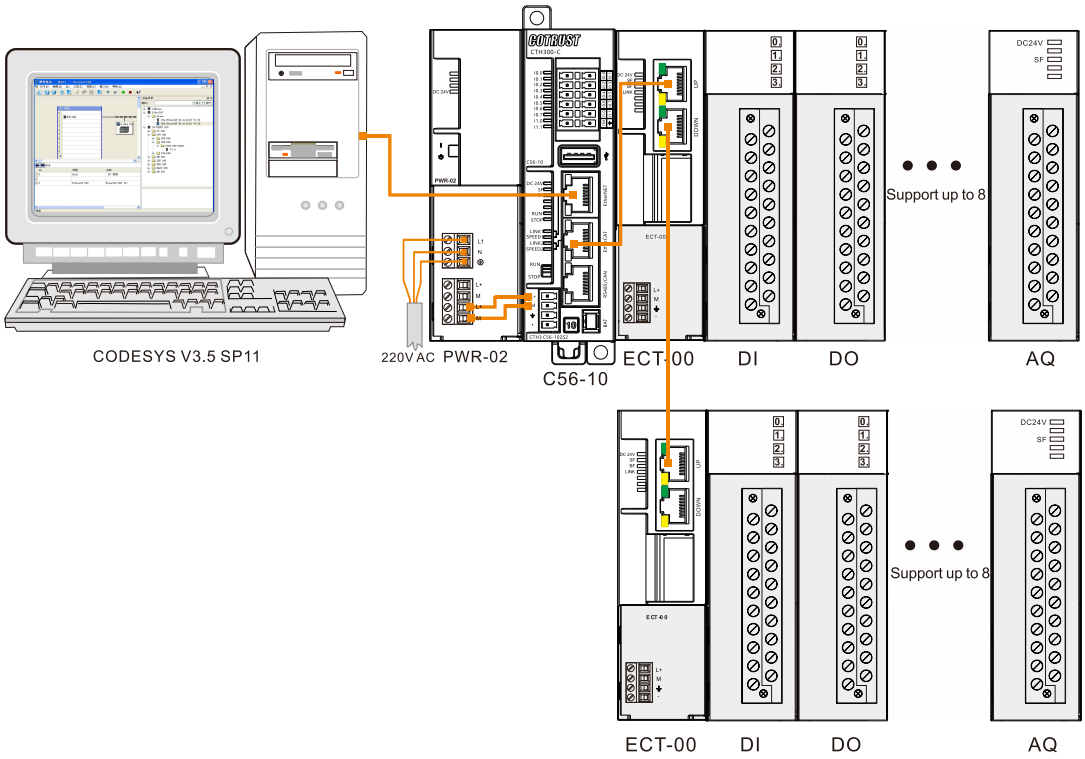
1. Preparation before communication

Table 7-6 Example components for EtherCAT communication

Component	Function
Programming device PG\PC	A programming device with CODESYS V3.5 SP11 is installed to configure, program and debug the C56-10 motion controller.
Assembly rail	Used to secure the modules in the system.
Power Module PWR-02	Powers the C56-10 motion controller and its 24VDC load circuit.

C56-10 main control module	The CPU executes the user program, supplies 5 V to the bus, and communicates with other nodes in the Ethernet network through the Ethernet interface.
EtherCAT master module	CTH300 CPU, C56-10
EtherCAT slave device	ECT-00
Expansion I/O Modules	16 CTH300 digital and analog I/O modules, 8 are connected to two EtherCAT slave modules respectively
3 standard network cables	● Connect C56-10 with programming device. ● Connect the OUT port of the EtherCAT communication port of the C56-10 to the IN port of the first ECT-00. ● Connect the OUT port of the first ECT-00 to the IN port of the second ECT-00.

2. Network connection



Refer to the network connection diagram above to wire the system.

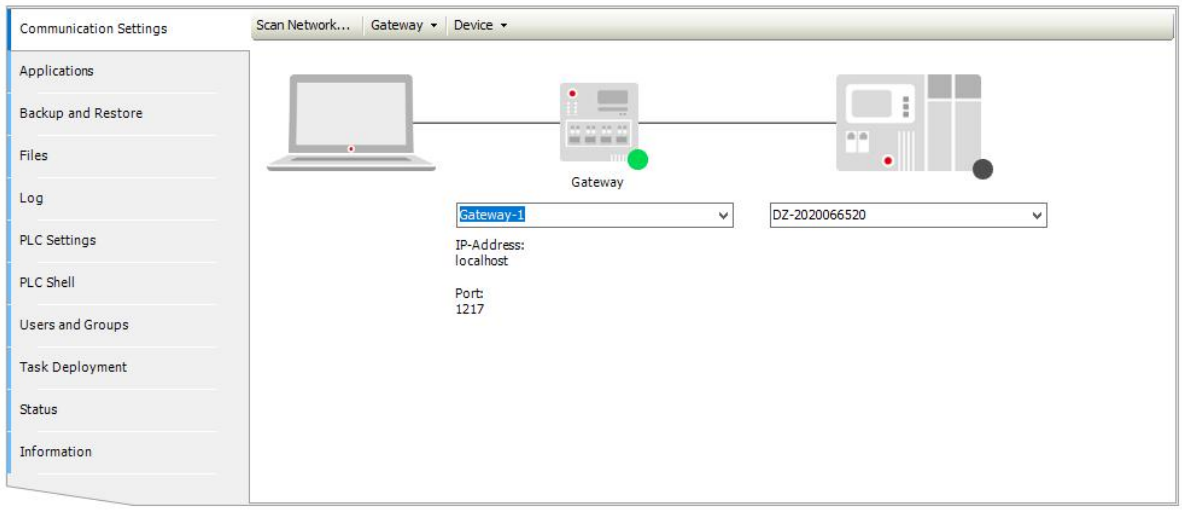
3. Set up communication

Before setting the communication, it is necessary to set the IP of the programming device PG/PC to the same network segment as the C56-10 (IP: 192.168.0.x). Setting method: Open the local connection properties of the PC, double-click the TCP/IP protocol, Change "Obtain an IP address automatically" to "Use the following IP address", and then fill in "192.168.0.x" in the IP address.

4. Set the CPU

Double-click "Device (CTH3 C56-10S2_V0.9)" to open the device dialog box, click the "Add Gateway" in the "Gateway" drop-down menu in the "Communication Settings" tab, and enter "Gateway" in the pop-up "Gateway" dialog box. Name", "Driver" select "TCP/IP", "IP Address" select "localhost", and finally click "OK" to close the dialog box, C56-10 is added to the communication dialog box.

After the C56-10 is added successfully, click "Scan Network" to search for available devices on the local network. If the search is successful, select the searched device and click the "Set Active Path", this operation will activate the communication channel setting, and all communication-related operations will be associated with this channel.



<Remarks> When the system starts, CODESYS related service programs (such as , , etc.) will appear in the system tray. If there are no special requirements, there is no need to operate the service programs in the tray.

5. Operation steps

Step 1. Power on each device

1) Open the front panel of C56-10 and power supply module PWR-02, and then connect C56-10 to PWR according to the network connection.

2) Connect the main power and control power to the system.

Step 2. Use cables to connect each device

Refer to the network connection diagram to connect each device, the specific operation is as follows:

1) Use a standard network cable to connect the PC and the EtherNET communication port of the C56-10.

2) Use a standard network cable to connect the EtherCAT communication OUT port of the C56-10 and the IN port of the first EtherCAT slave module.

3) Use a standard network cable to connect the OUT port of the first EtherCAT slave to the IN port of the second EtherCAT slave station module.

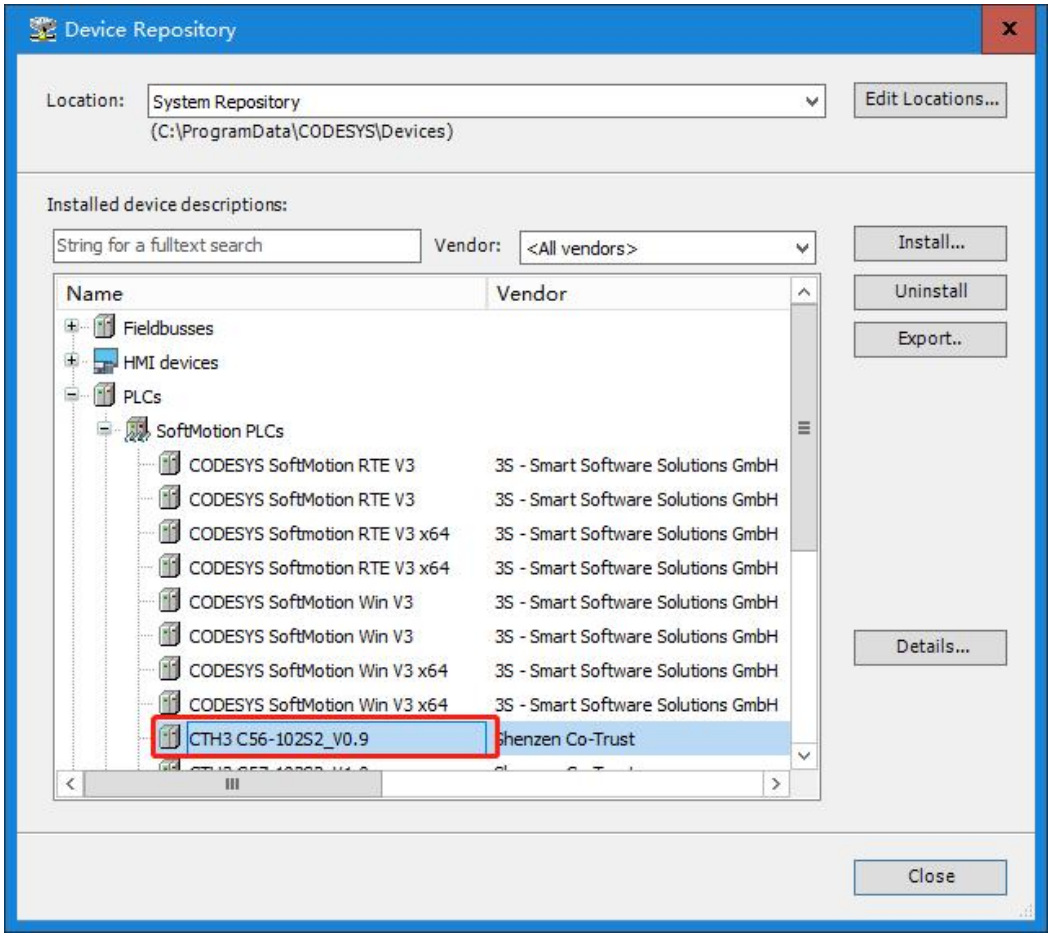
Step 3. Configure EtherCAT in CODESYS

1) Install the device description file

Install the COTRUST device description files (Co-Trust_C56_102S2_V0.9.devdesc.xml) and (Co-Trust_ECAT_SLAVE.V1.2.xml) of the CPU C56-10 and EtherCAT slave modules used in the example components. COTRUST equipment is used in the system, and the specific installation operations are as follows:

Select "Tools" → "Install Device" to install the device description file. The description files provided by COTRUST can be selected by setting the corresponding filter (Note: EtherCAT XML Device Description Profile must be selected). Click "Open" to confirm the option, and the new device will be added to the device catalog in the "Device Library".

- Select "Tools" → "Device Repository" to open the device library, as shown below:



Select "PLC" → "SoftMotion PLCs" to see the successfully installed COTRUST device "CTH3-C56-10S2_V0.9". Expand "Fieldbus" → "EtherCAT" → "Slave" to see "EtherCAT Slave". After checking, if you need to continue to install the device file, please uncheck "Enable Device Library Dialog" in "Tools" → "Options".

Tips
1) If you also need other devices that use COTRUST (CANopen or other EtherCAT slave devices), then you also need to install the following device description files: ◆ CANopen slave device configuration file (*.eds): Co-Trust E10.eds、Co-Trust EM277C.eds、A4S_CAN V003.eds; ◆ EtherCAT slave device configuration file (*.xml): COTRUST_A4N_ V1.3.xml. 2) Please download the required device configuration file from COTRUST official website: http://www.co-trust.com。

2) Create a new project

- Choose "File" → "New Project", then choose "Standard Project" and set the file name and storage directory.

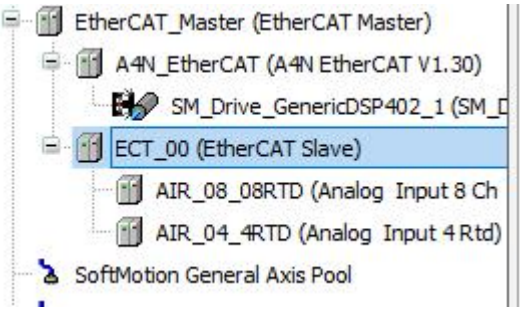
- After performing the above operations and confirming, another dialog box will pop up. In the dialog box, select the newly added CPU file device (CTH3-C56-10S2_V0.9) and the programming language used, and then click "OK" to complete the project. create.

3) Add EtherCAT master

In the opened project, right-click "Device (CTH3-C56-10S2_V0.9)" and select "Add Device", then you can choose to add EtherCAT master in the pop-up dialog box: Supplier select "<all suppliers>", Select "EtherCAT" → "Master" → "EtherCAT Master" for the field bus.

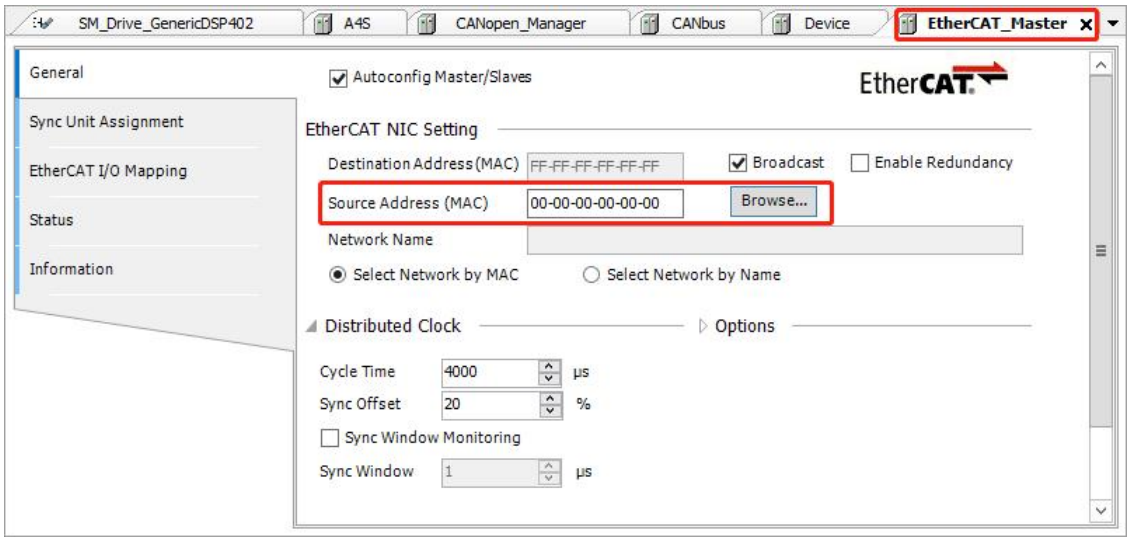
4) Add EtherCAT slave

Select the EtherCAT master and right-click to select "Add Device", you can choose to add an EtherCAT slave in the pop-up dialog box: "<all suppliers>" for suppliers, "EtherCAT" → "Slave" → "ECT-00", the EtherCAT slave (ECT-00) connected to the C56-10 will be displayed below the EtherCAT master (C56-10).



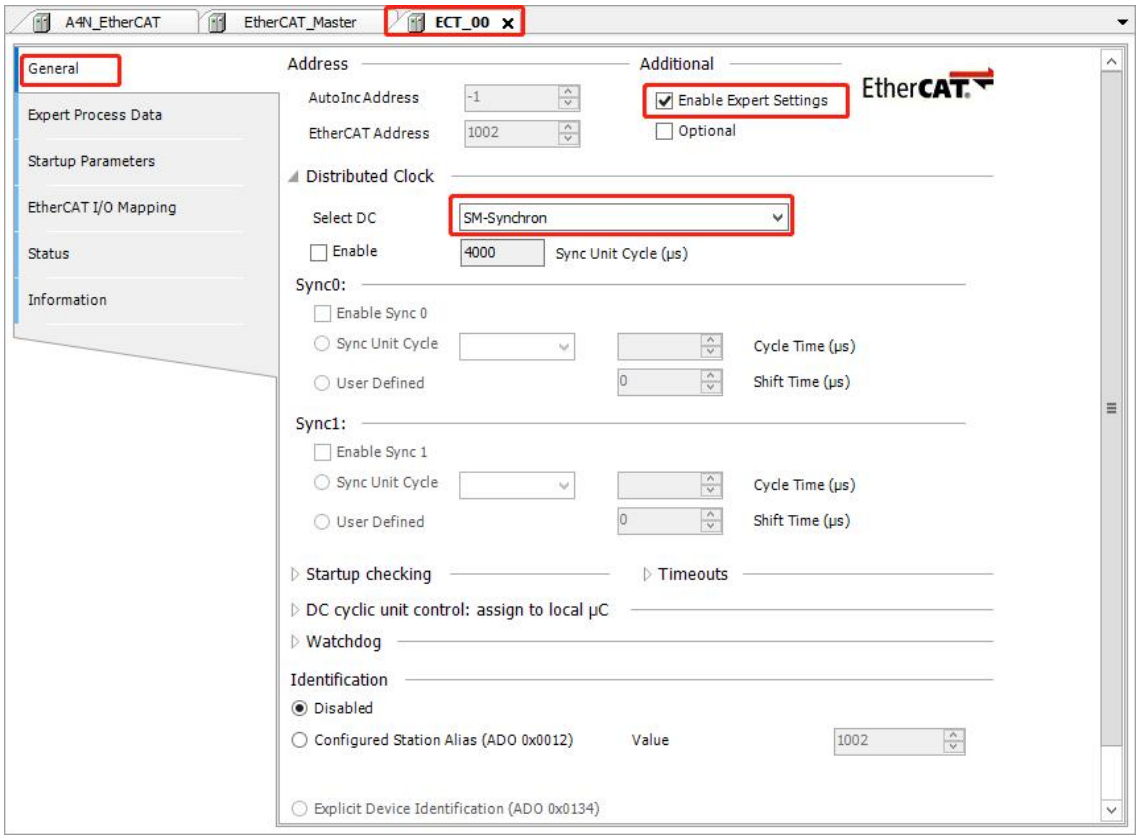
5) Configure the EtherCAT master

Double-click "EtherCAT-Master", and set the source address (MAC) in "Overview": Click "Browse" to the right of "Source Address (MAC)", and then select the corresponding MAC address.



6) Configure EtherCAT Slave (ECT-00)

Check "Enable Expert Settings" in "Overview", select DC "SM-Synchron" for distributed clock.



<Remark> Please refer to Appendix F.3 Configuring EtherCAT Slave Devices for more details on EtherCAT slave configuration.

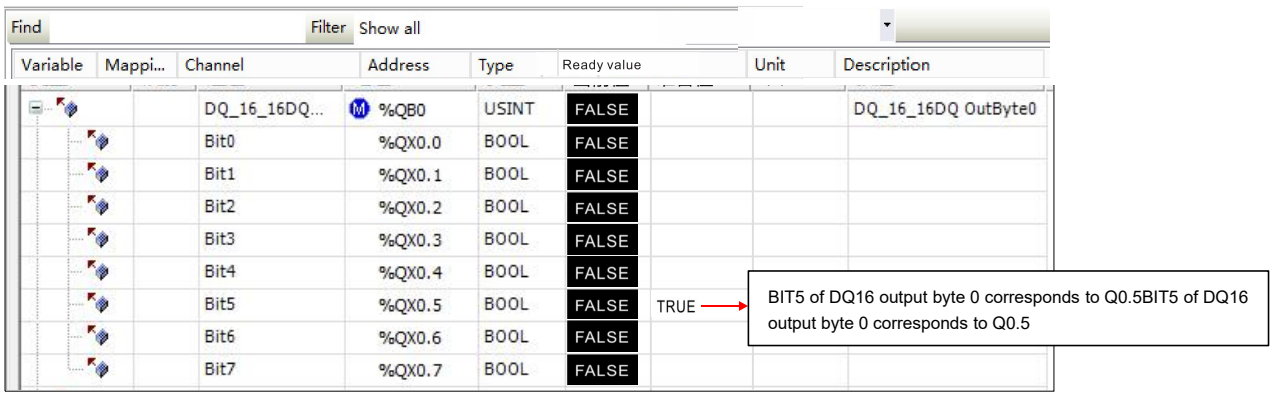
Step 5. Refer to "Set Communication Settings" C56-10 Communication with the host computer

Step 6. Running and debugging

1) Select "Online" → "Login" to establish a connection between the application and C56-10, and enter the online state. Then, select "Debug" → "Start" to start the application in C56-10.

2) Monitor and debug the current project

Debug the connected module in the tab "Module I/O Mapping" of the ECT_00 slave (check "Always in the bus cycle task"), take the Digital Output 16 Bits digital input module as an example, click "DQ_16_16DQ (Digital Output 16 Bits)", light up its output Q0.5, the operation is as follows:



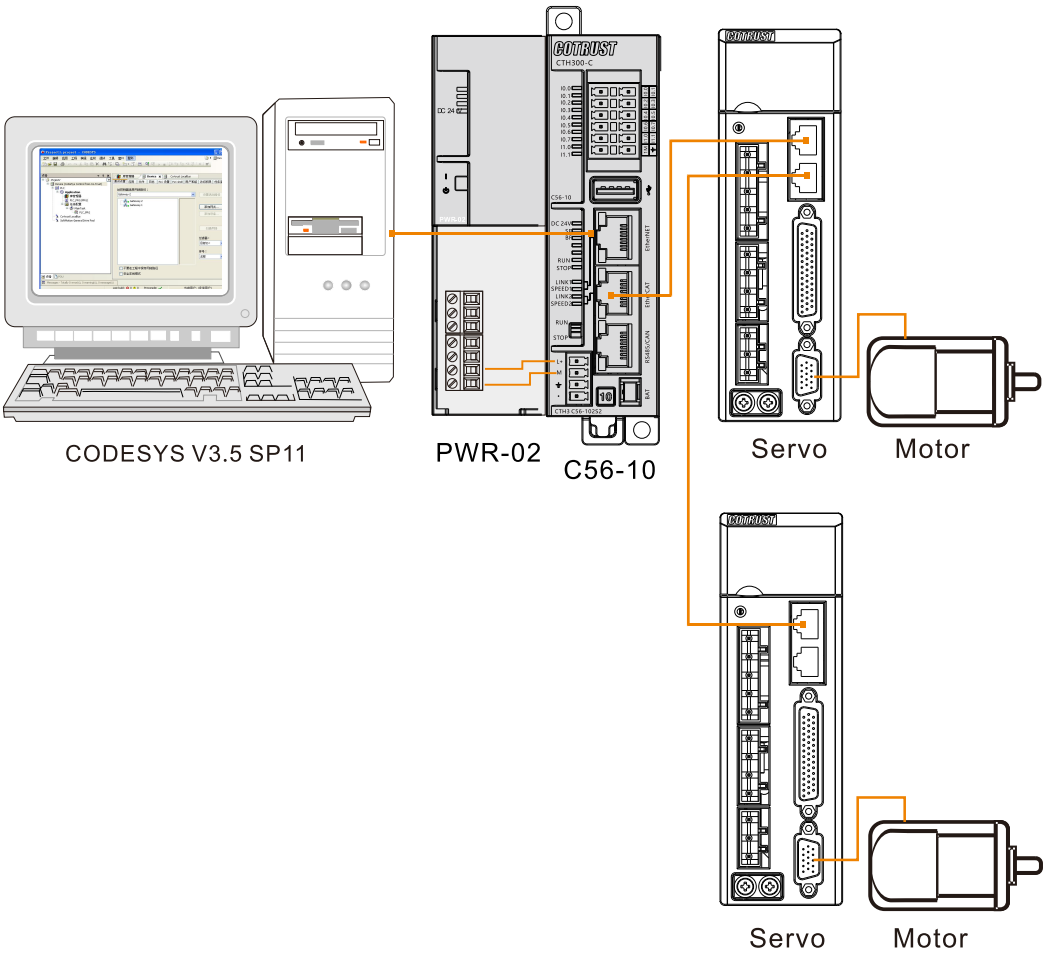
7.1.7 Electronic Cam based on EtherCAT Communication

1. Preparation before communication

Table 7-7 Example components for EtherCAT communication

Component	Function
Programming device with CODESYS V3.5 SP11 installed	Configure, program and debug C5X-10 series motion controllers
Assembly rail	For fixing the modules in the system
Power Module PWR-02	Provide C5X-10 series motion controller and its 24 VDC load circuit
External power supply	Provide A4N servo drive working power
C56-10 motion controller	Execute the user program, provide 5V voltage for the bus, and communicate with other modules through the Ethernet interface.
A4N Servo Driver and Motor	This example uses: A4N servo driver and supporting motor
Standard network cable	◆ Connect programming device to C56-10 ◆ Connect C56-10 with A4N servo driver ◆ Interconnection between A4N Servo Drives
Encoder cable	Connect A4N Servo Drive and Motor

Internet connection



3. Operation steps

Step 1. Power on each device

1) Open the front panel of C56-10 and power module PWR-02, and then connect C56-10 to PWR-02 according to the above network connection.

2) Connect the main power and control power to the A4N servo drive.

Step 2. Use cables to connect each device

Refer to the network connection diagram to connect each device, the specific operation is as follows:

1) Use standard network cable to connect PC and C56-10

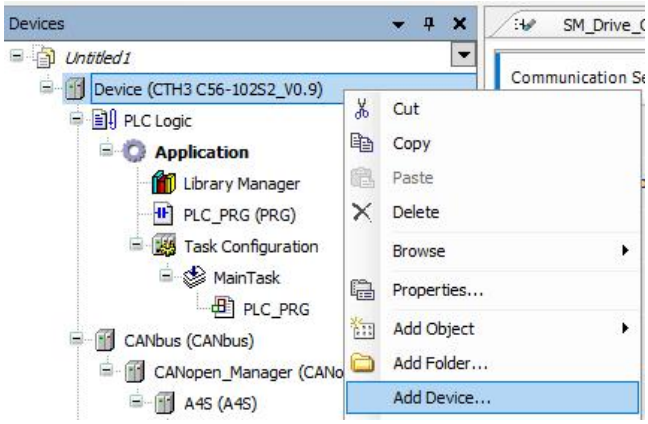
2) Use standard network cable to connect C56-10 and A4N servo drive

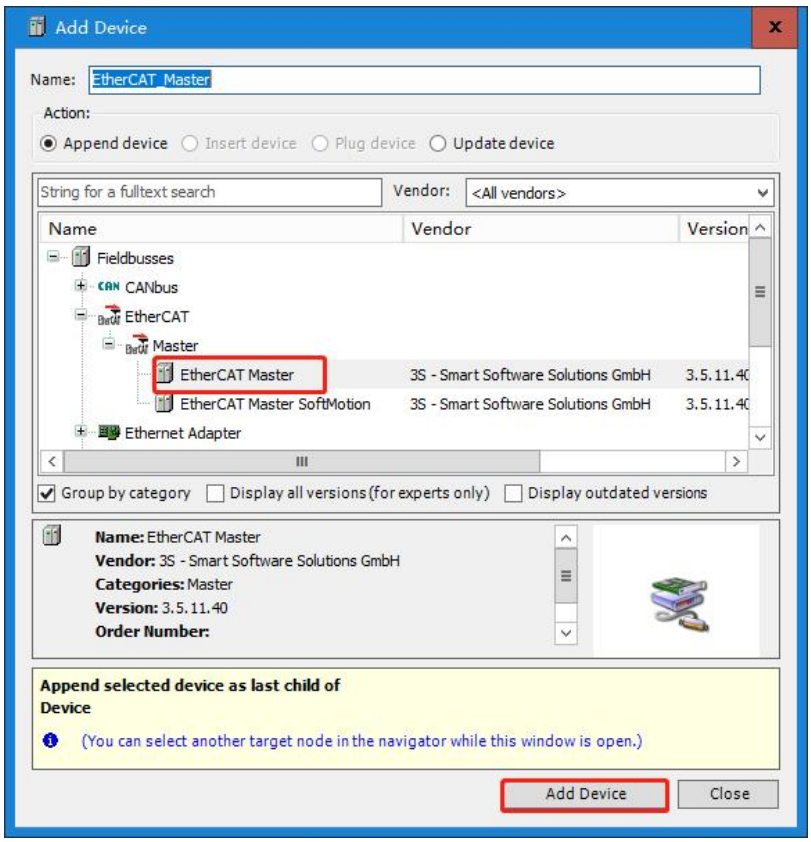
3) Use the encoder cable to connect the A4N driver to the motor

Step 3. Configure EtherCAT in CODESYS

1) Add EtherCAT master

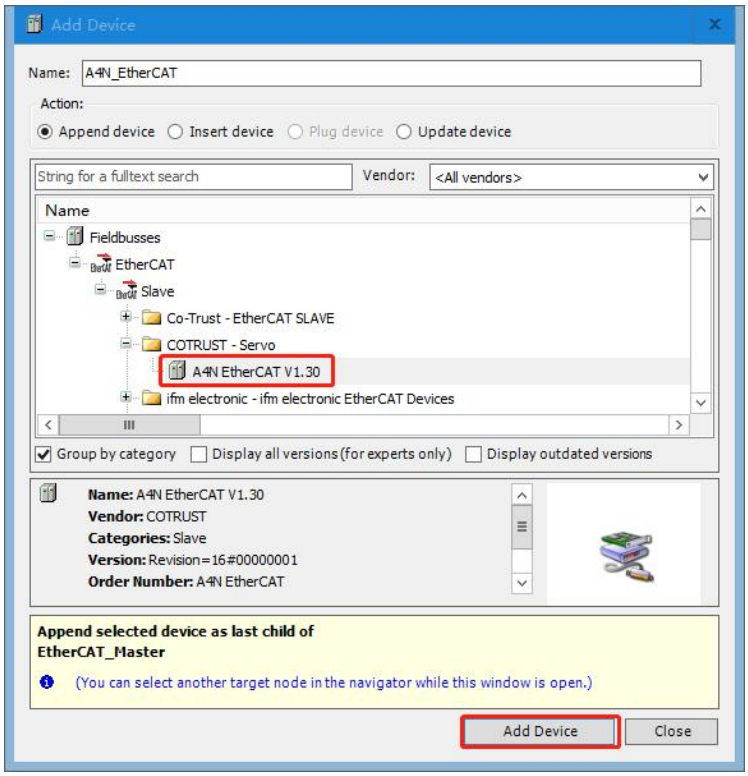
Right-click "Device(CTH3 C56-10S2_V0.9)" in the device, select "Add Device", and then add an EtherCAT master: "<all suppliers>" for the supplier, "EtherCAT" → "Master" for the fieldbus → "EtherCAT Master".





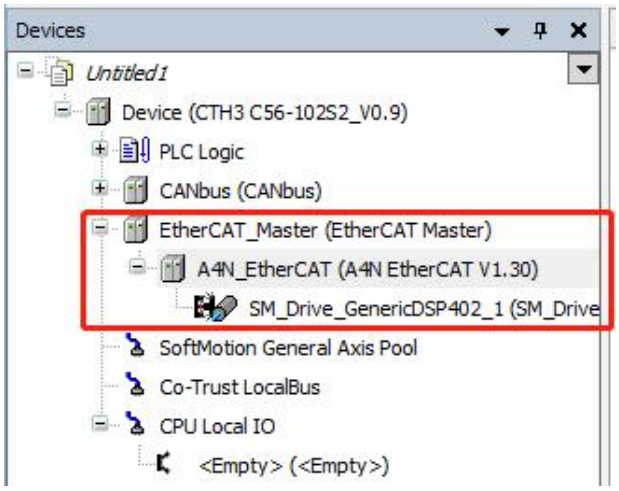
2) Add or scan EtherCAT slaves

Select the EtherCAT master in the device view and right-click to select "Add Device" to add an EtherCAT slave: "<all suppliers>" for the supplier, "EtherCAT" → "Slave" → "COTRUS-Servo" → "A4N". Or right-click on the EtherCAT master (C56-10) and select "Scan Device" to display the EtherCAT slave (A4N) connected to the C56-10 under the EtherCAT master (C56-10).



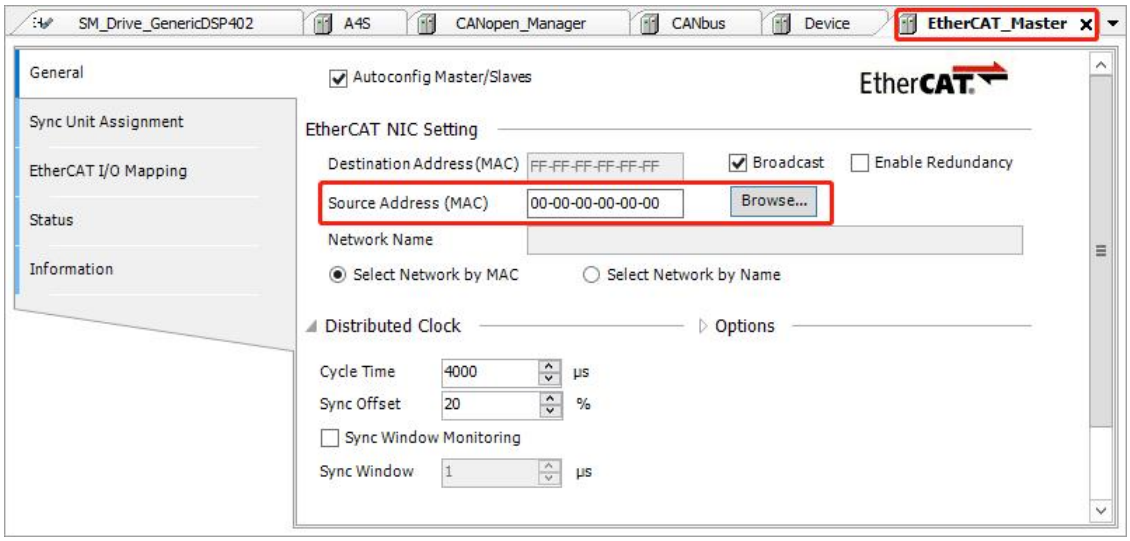
3) Add the driver of the EtherCAT slave

Right-click on the EtherCAT slave (A4N) in the device view and select "Add softmotion-CiA402-axis".



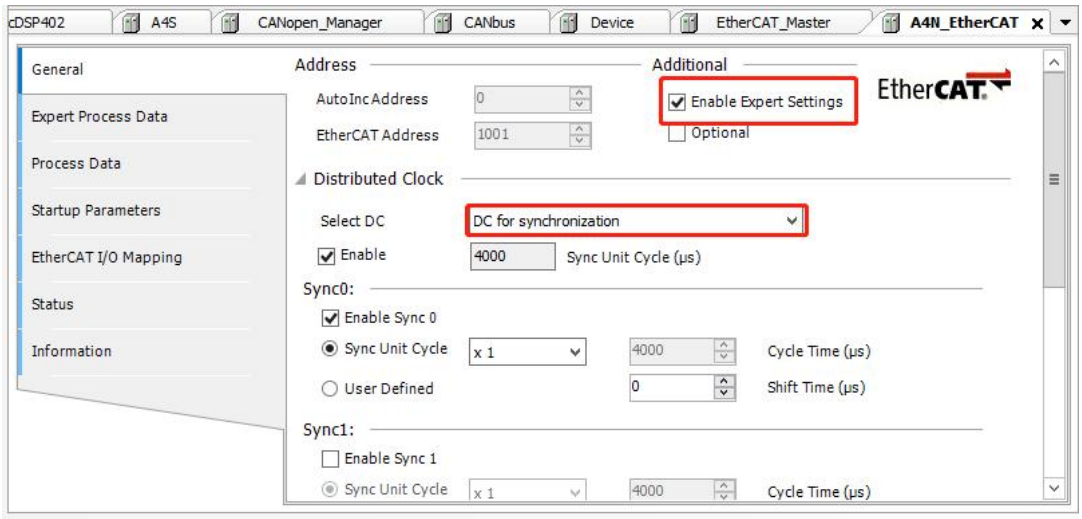
4) Configure the EtherCAT master

Set the source address (MAC) in the "Overview" tab: Click "Browse" and select the appropriate MAC address.



5) Configure EtherCAT slave (A4N)

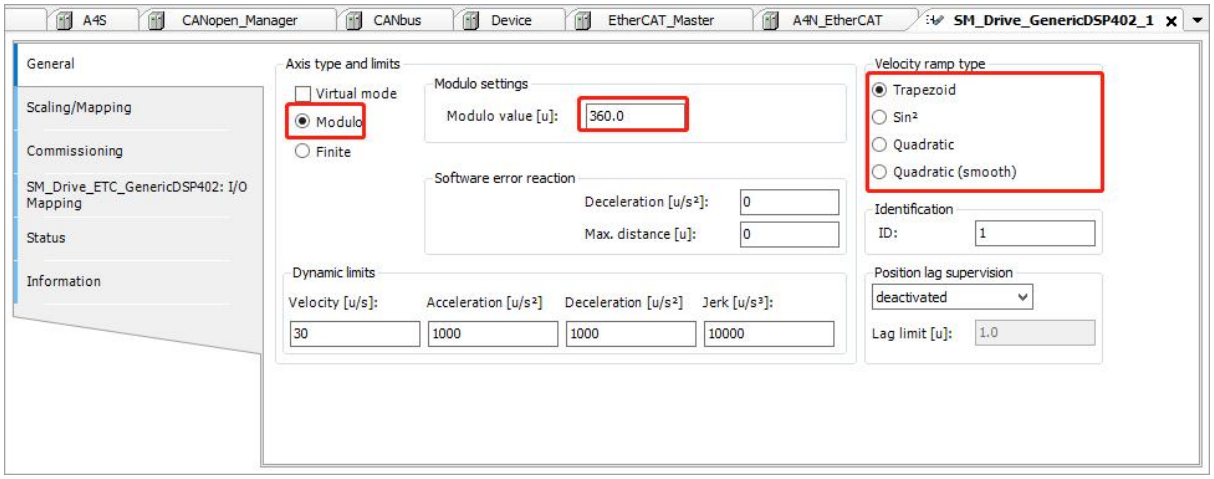
In the "Slave" tab, check "Enable expert settings", and select "DC for synchronization" for distributed clocks.



<Remark> Please refer to Appendix F.3 Configuring EtherCAT Slave Devices for more details on EtherCAT slave configuration.

6) Configure A4N driver (SM_Driver_GenericDSP402)

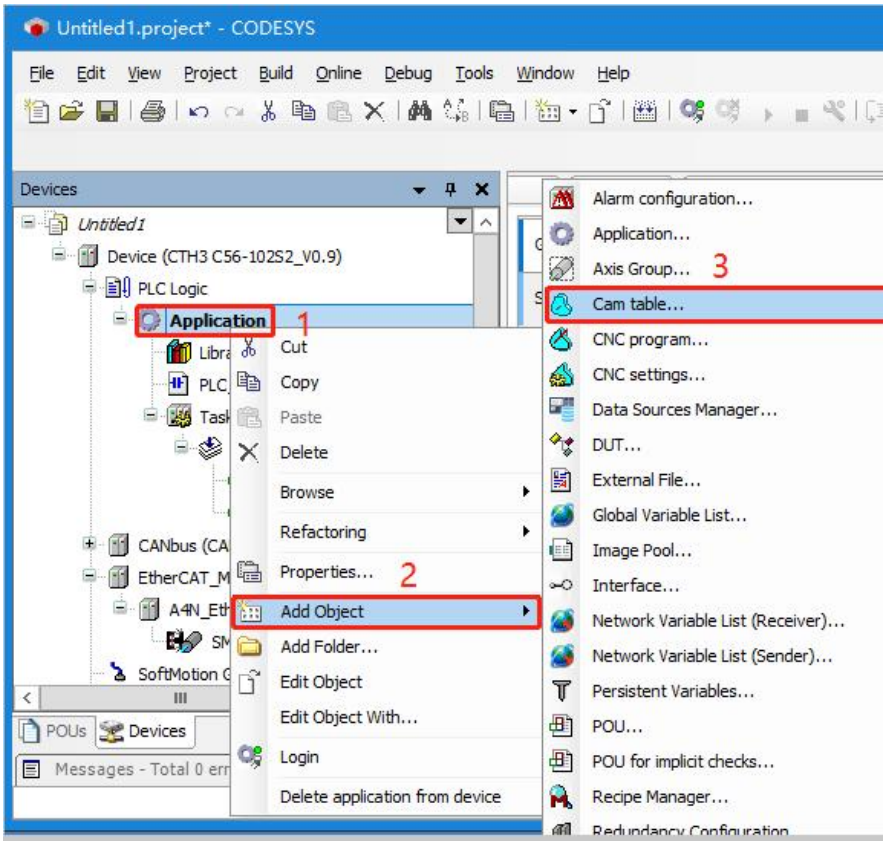
The "Softmotion Driver: Basic" tab of SM_Driver_GenericDSP402 is set as follows:

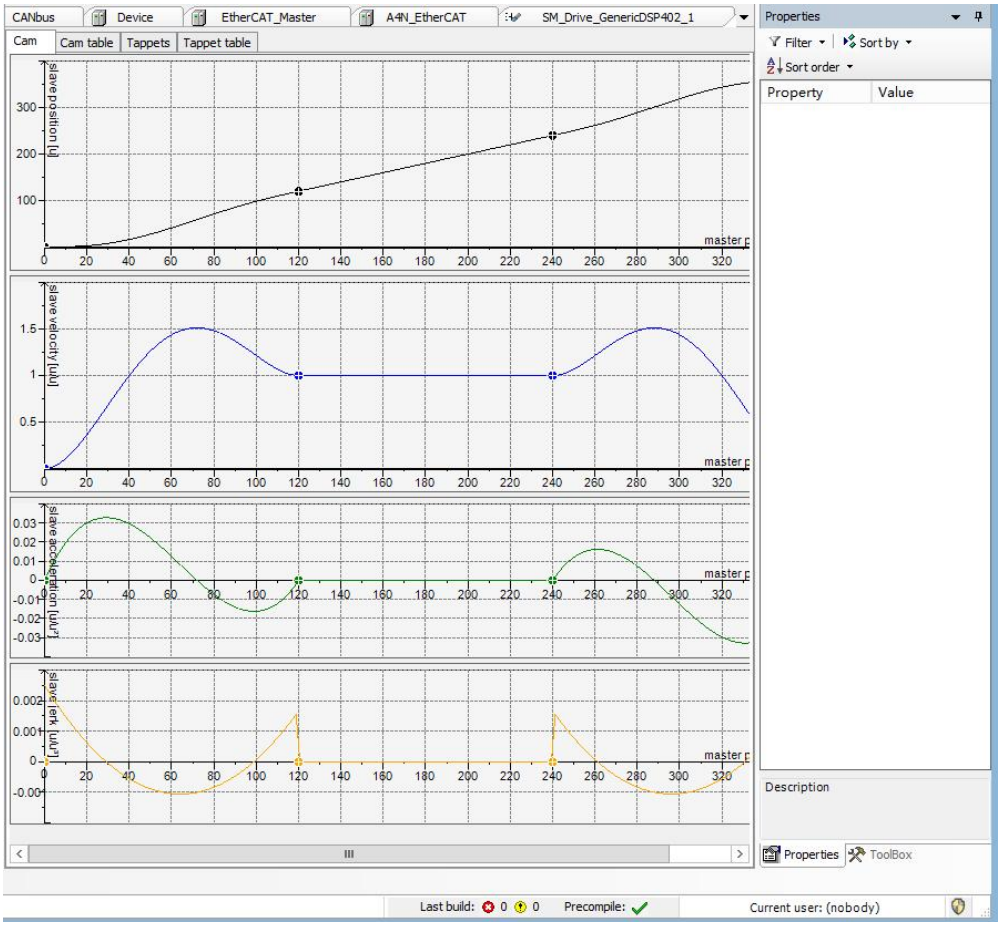


The Axis type and Limit are selected as "Modulo", the modulo value is set to 360.0, and the speed ramp type is selected as Trapezoid.

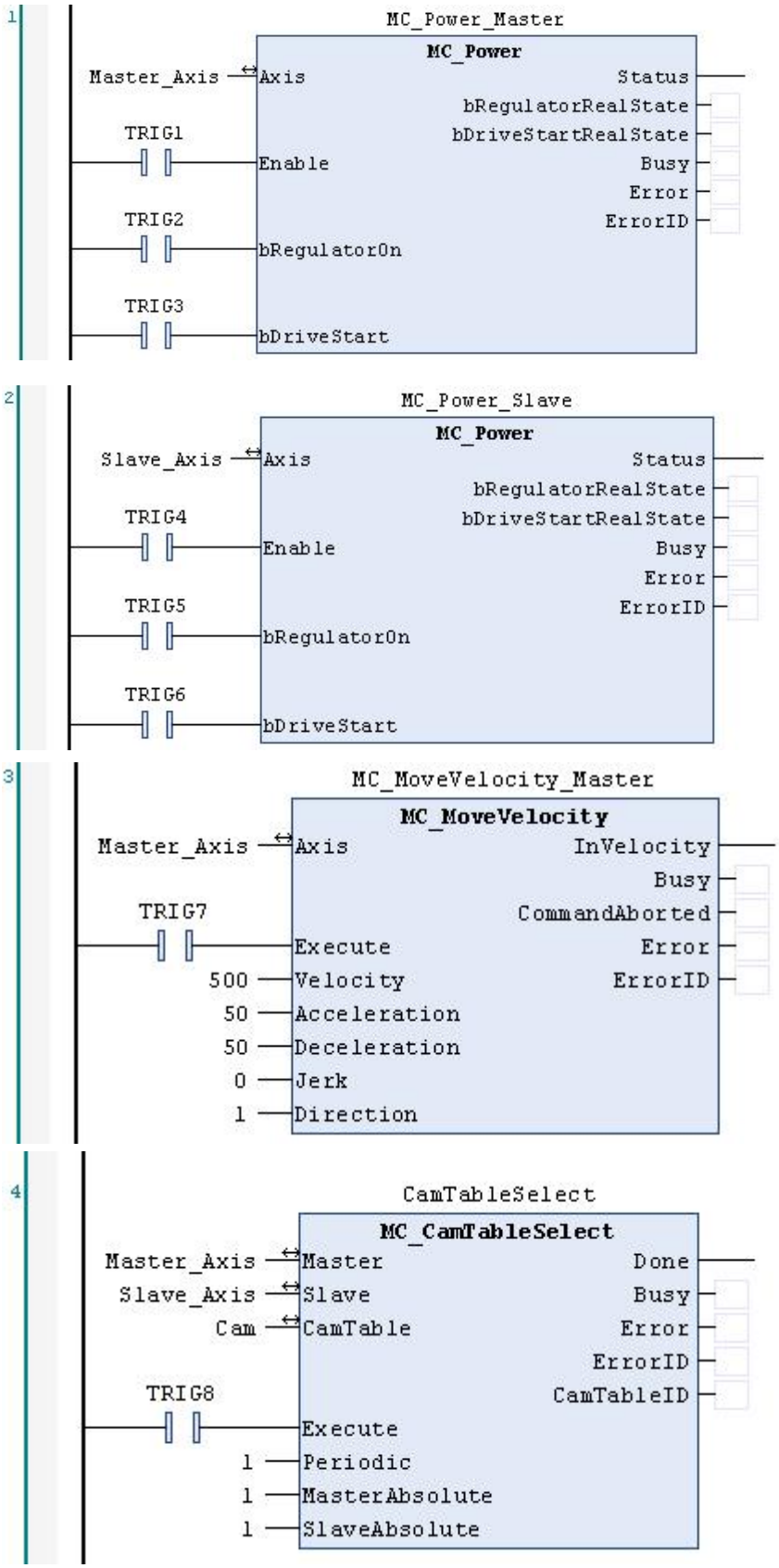
7) Create a new Cam table

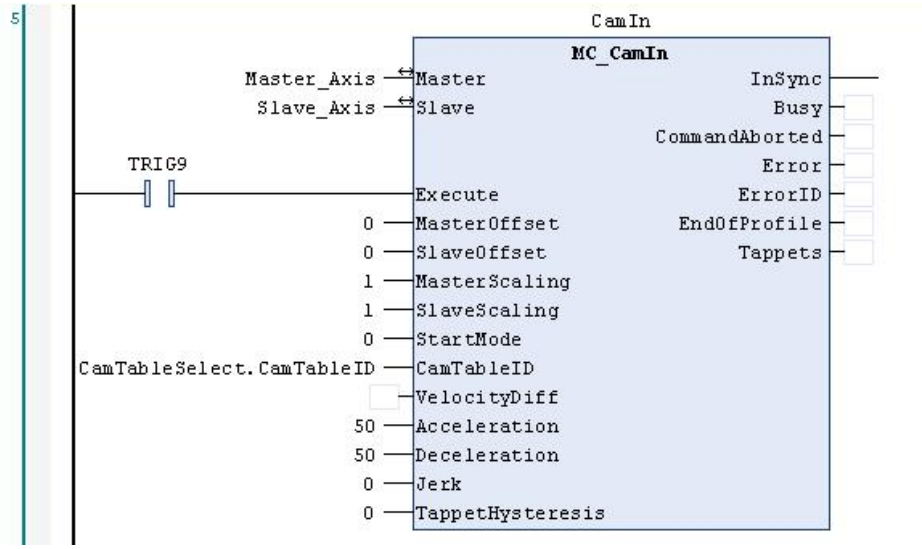
Expand "Device" → "PLC" in the device view, then right-click "Application" and select "Add Object" → "Cam Table", enter the name of the table in the pop-up dialog box, and click "Open", the Cam table will be Created successfully. Double-click to open the newly created cam table:


Untitled1.project* - CODESYS
File Edit View Project Build Online Debug Tools Window Help
Devices
Untitled1
Device (CTH3 C56-102S2_V0.9)
PLC Logic
Application
1
Cut
Copy
Paste
Delete
Browse
Refactoring
Properties...
2
Add Object
Add Folder...
Edit Object
Edit Object With...
Login
Delete application from device
Alarm configuration...
Application...
Axis Group...
3
Cam table...
CNC program...
CNC settings...
Data Sources Manager...
DUT...
External File...
Global Variable List...
Image Pool...
Interface...
Network Variable List (Receiver)...
Network Variable List (Sender)...
Persistent Variables...
POU...
POU for implicit checks...
Recipe Manager...
POUs
Devices
Messages - Total 0 err


CANbus
Device
EtherCAT_Master
A4N_EtherCAT
SM_Drive_GenericDSP402_1
Cam
Cam table
Tappets
Tappet table
slave position [u]
slave velocity [u/u]
slave acceleration [u/u²]
slave jerk [u/u³]
master p
Properties
Filter
Sort by
Sort order
Property
Value
Description
Properties
ToolBox
Last build: 0 0
Precompile:
Current user: (nobody)

Step 4. Programming





Step 5. Refer to the chapter [2.3 Set up communication] to set the communication between C56-10 and the host computer

Step 6. Running and debugging

1) Select "Online" → "Login." to establish a connection between the application and the C56-10 and enter the online state; then, select "Debug" → "Start" to make the application in the C56-10 The program starts running.

2) Monitor and debug the current project

Double-click "PLC_PRG (PRG)" in the device view to open the program table, trigger and download each instruction in turn.

7.1.8 EtherNET/IP Communication

This section will demonstrate the EtherNET/IP communication function of C56-10 through a concrete example.

When choosing EtherNET/IP communication, the router needs to establish communication with C56-10, R51C1-EP/Pro, and then the C56-10 can be programmed and monitored through the CODESYS programming software.

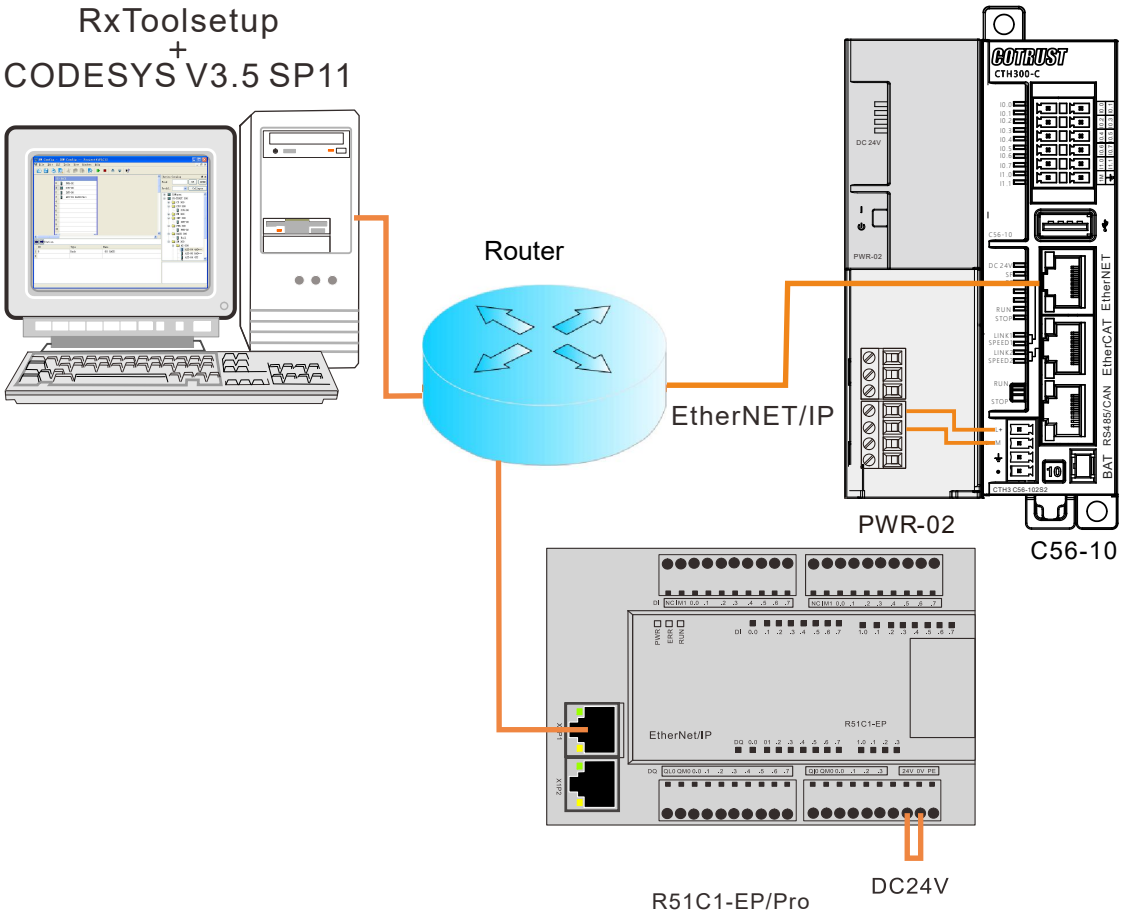
1. Preparation before communication

Table 7-8 Example Components for EtherNET/IP Communication

Components	Function
Programming device PG\PC	A programming device with CODESYS V3.5 SP11 and RxToolsetup V1.4 is installed to configure, program and debug the C56-10 motion controller.
Assembly rail	Used to secure the modules in the system.
Power Module PWR-02	Powers the C56-10 motion controller and its 24VDC load circuit.
C56-10 main control module	The CPU executes the user program, supplies 5 V to the bus, and communicates with other nodes in the Ethernet network through the Ethernet interface.
Router	Provide connection interface, connect computer, master C56-10

	and slave R51C1-EP/Pro.
R51C1-EP/Pro	as a slave.
3 standard network cables	• Connect the C56-10 to the router • Connect R51C1-EP/Pro slave station and router • Connect the computer to the router

2. Network connection



Refer to the network connections above to wire the system.

3. Set up communication

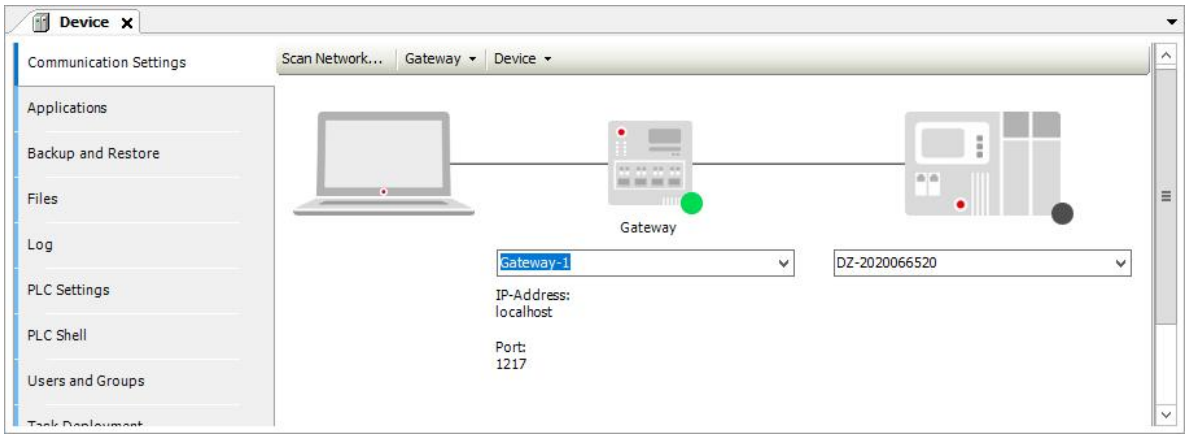
Before setting the communication, it is necessary to set the IP of the programming device PG/PC to the same network segment as the C56-10 (IP: 192.168.0.x). Setting method: Open the local connection properties of the PC, double-click the TCP/IP protocol, Change "Obtain an IP address automatically" to "Use the following IP address", and then fill in "192.168.0.x" in the IP address.

4、Set up CPU

Double-click "Device (CTH3 C56-102S2_V1.0)" to open the device dialog box, as shown in the figure below. Select "Gateway" in "Communication Settings", click "Add Gateway", then input "Name", "Driver", select "TCP/IP", "IP Address" and select "localhost", and finally click "OK" to close the dialog box , C56-10 is added to the communication dialog box.

After the C56-10 is added successfully, click the "Scan Network" to search for available devices on the local network. If the search is successful, select the searched device and click the "Set

Active Path", this operation will activate the communication channel setting, and all communication-related operations will be associated with this channel.



1) Install the device description file

Install the device description files (Co-Trust_C56_102S2_V1.0.devdesc.xml) and (R51CX_EPPROv203.eds) of the CPU C56-10 and EtherNET slave module used in the sample components. After the installation is successful, you can use the device in the program system. The operation is as follows:

Select "Tools" → "Device Repository" to open the following dialog box, which lists the device descriptions in the system, click "Install".

5. Operation steps

Step 1. Power on each device

1) Open the front panel of C56-10, power supply module PWR-02, and then connect C56-10 to PWR according to the network connection.

2) Power on the slave R51C1-EP/Pro.

Step 2. Use cables to connect each device

Refer to the network connection diagram to connect each device, the specific operation is as follows:

1) Use a standard network cable to connect the router and the EtherNET communication port of the C56-10.

2) Use standard network cable to connect R51C1-EP and router.

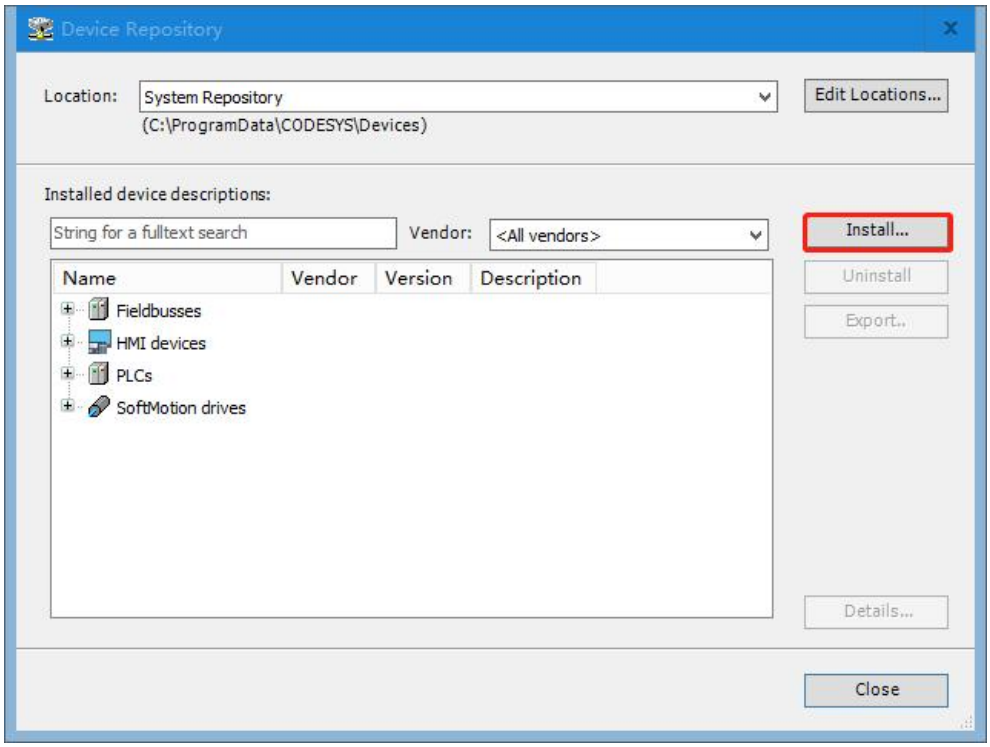
3) Use a standard network cable to connect the computer and the router.

Step 3. Configure in CODESYS

1) Install the device description file

Install the device description files (Co-Trust_C56_102S2_V1.0.devdesc.xml) and (R51CX_EPPROv203.eds) of the CPU C56-10 and EtherNET slave module used in the sample components. After the installation is successful, you can use the device in the program system. The operation is as follows:

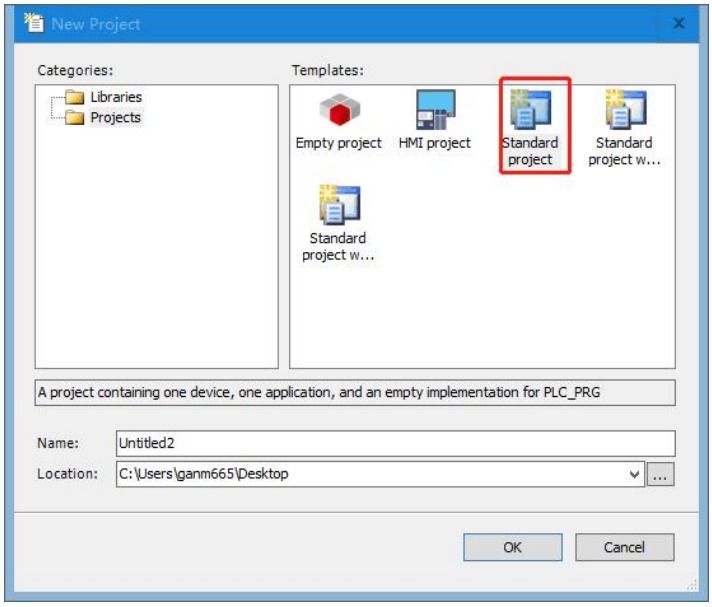
Select "Tools" → "Device Repository", which lists the device descriptions in the system, click "Install" in the dialog box.



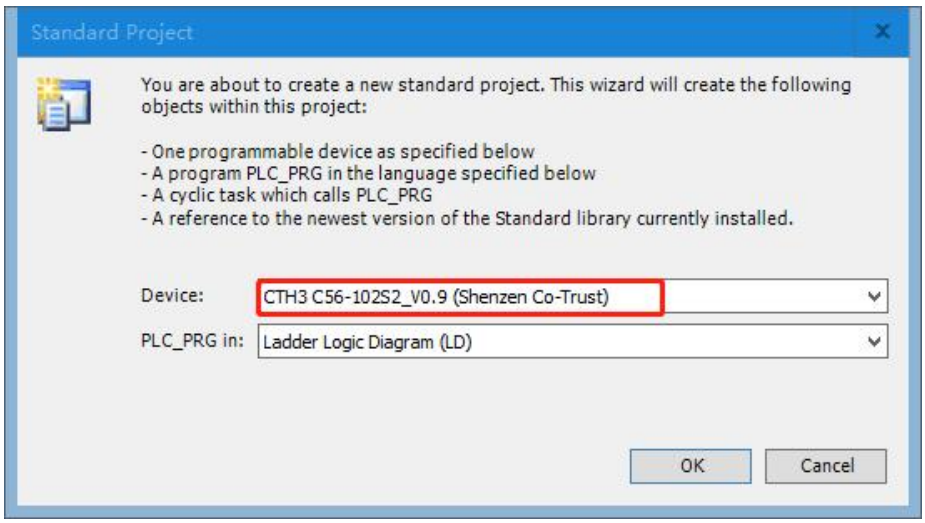
Select the file format, the description file can be filtered by the file format, then select the desired file, click the "Open" to confirm the option, and the new device will be added to the device catalog in the "Device Library".

2) Create a new project

- Select "File" → "New Project" on the main page of CODESYS, then select "Standard Project" and set the file name and storage directory.

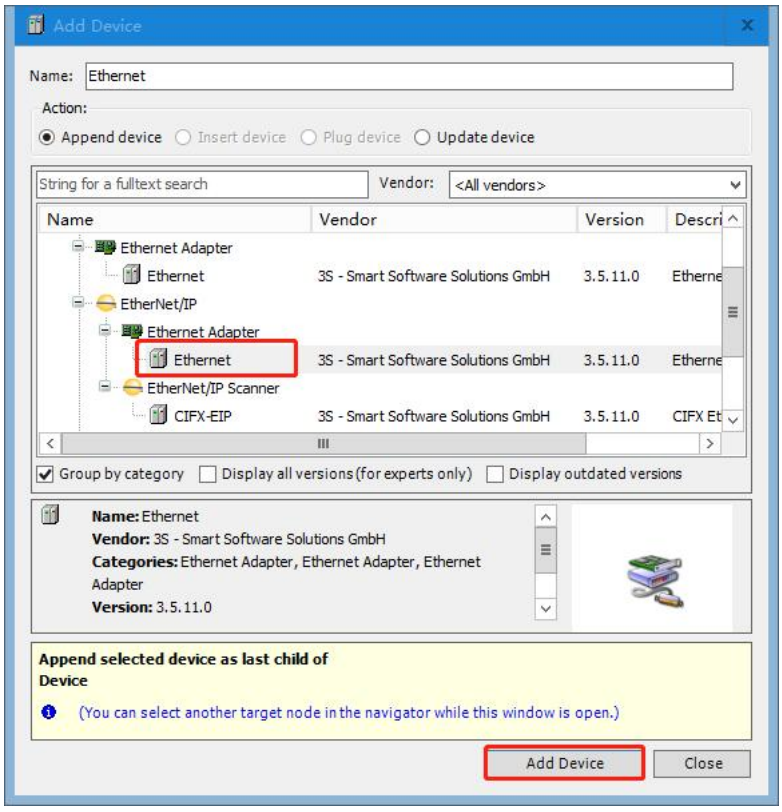


- After performing the above operations and confirming, another dialog box will pop up. In the dialog box, select the newly added CPU file device (CTH3-C56-102S2_V1.0) and the programming language used, and then click "OK" to complete the project.



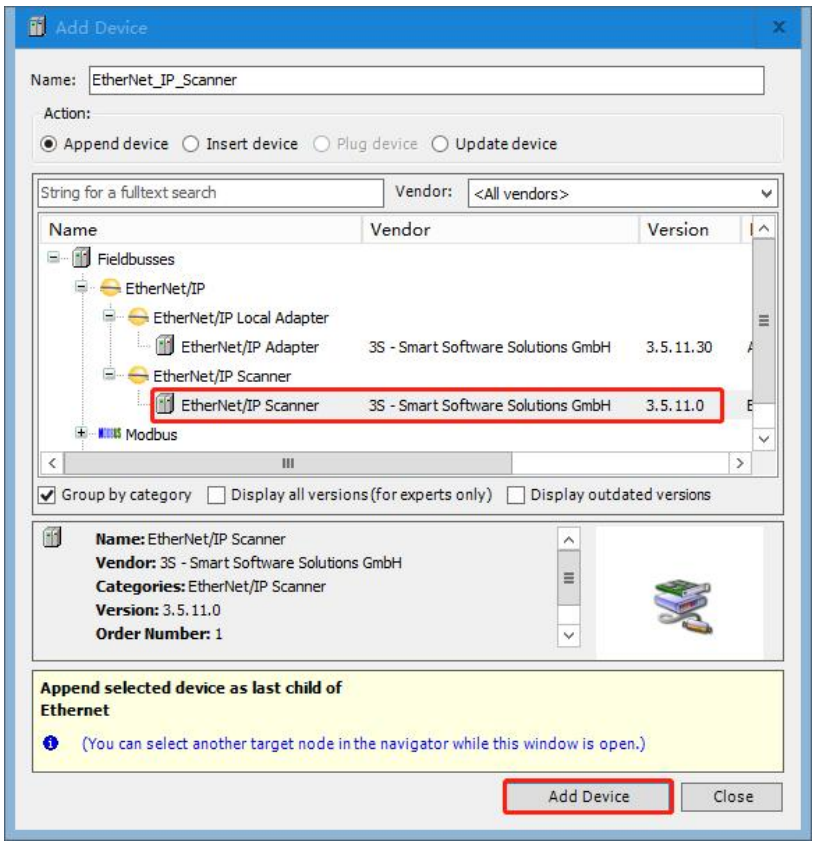
3) Add EtherNET master

Right-click on "Device (CTH3-C56-10S2_V1.0) and select "Add Device", then you can choose to add an EtherNET master in the pop-up dialog box: "<all suppliers>" for the supplier, and "EtherNET/ IP" → "Ethernet".



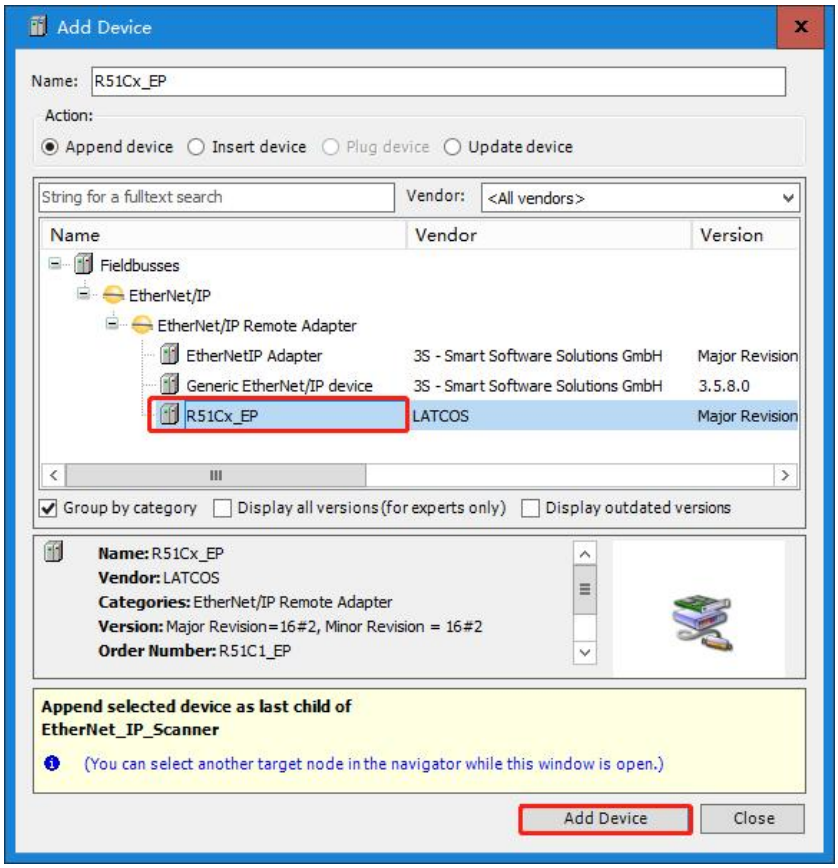
4) Add EtherNET/IP Scanner

Select the Ethernet master in the device view and right-click to select "Add Device", you can choose to add an EthernetIP scanner: "<all suppliers>" for the supplier, "Ethernet/IP" → "Ethernet_IP" for the fieldbus Scanner" → "Ethernet/IP Scancer".



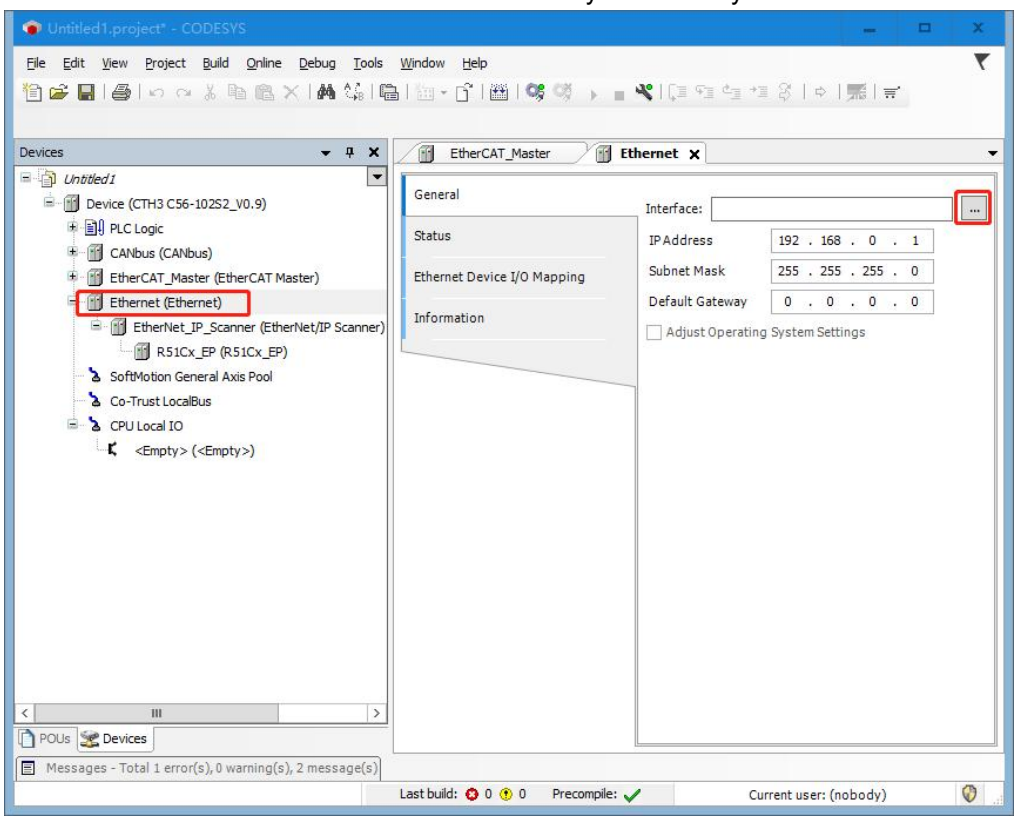
5) Add EtherNET/IP slave

Select Ethernet/IP Scancer in the device view and right-click to select "Add Device", you can choose to add an Ethernet slave (R51Cx-EP): Select "<all suppliers>" for suppliers, select "Fieldbus" "EthernetIP" → "EthernetIP target" → "R51Cx-EP", you can add EthernetIP slave.



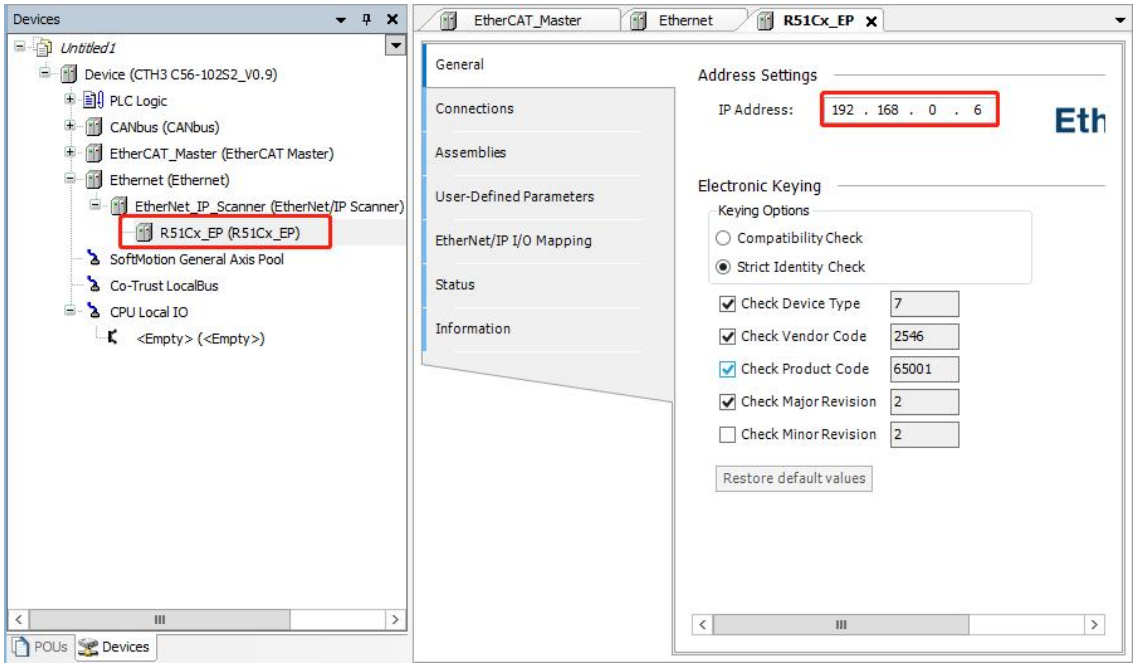
6) Configure the EtherNET master

Double-click "Ethernet" and set the interface in the "General" tab: Click the "..." on the right side of "Interface", and then select "eth0" in the pop-up dialog box. After selecting eth0, the IP address of the master station can be automatically obtained by CODESYS.



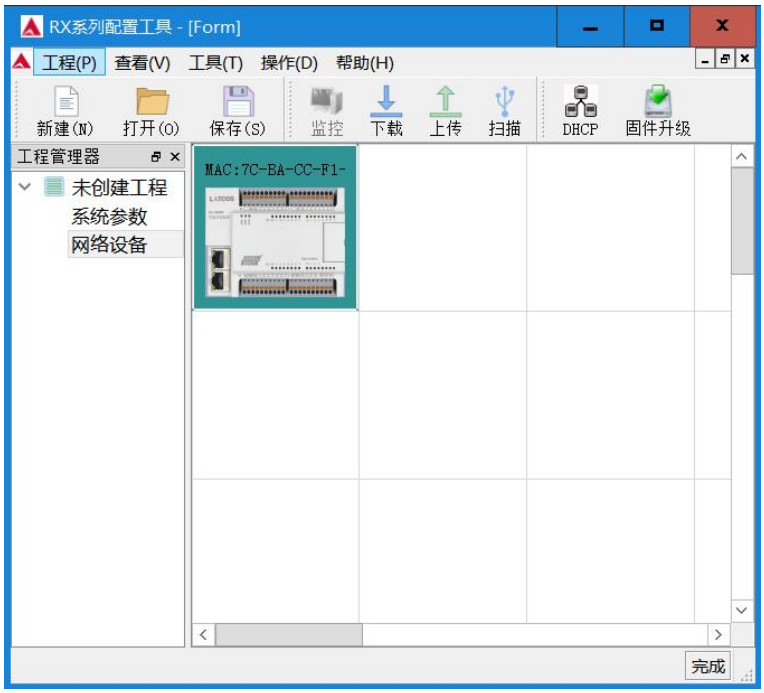
7) Configure EtherNET Slave

Set the configuration IP address of the slave in CODESYS, click "R51Cx-EP", and select "Scanner Settings" to set the configuration IP address of the EtherNET slave. The configuration IP address should be consistent with the actual IP address. See step 4 for setting the IP address.

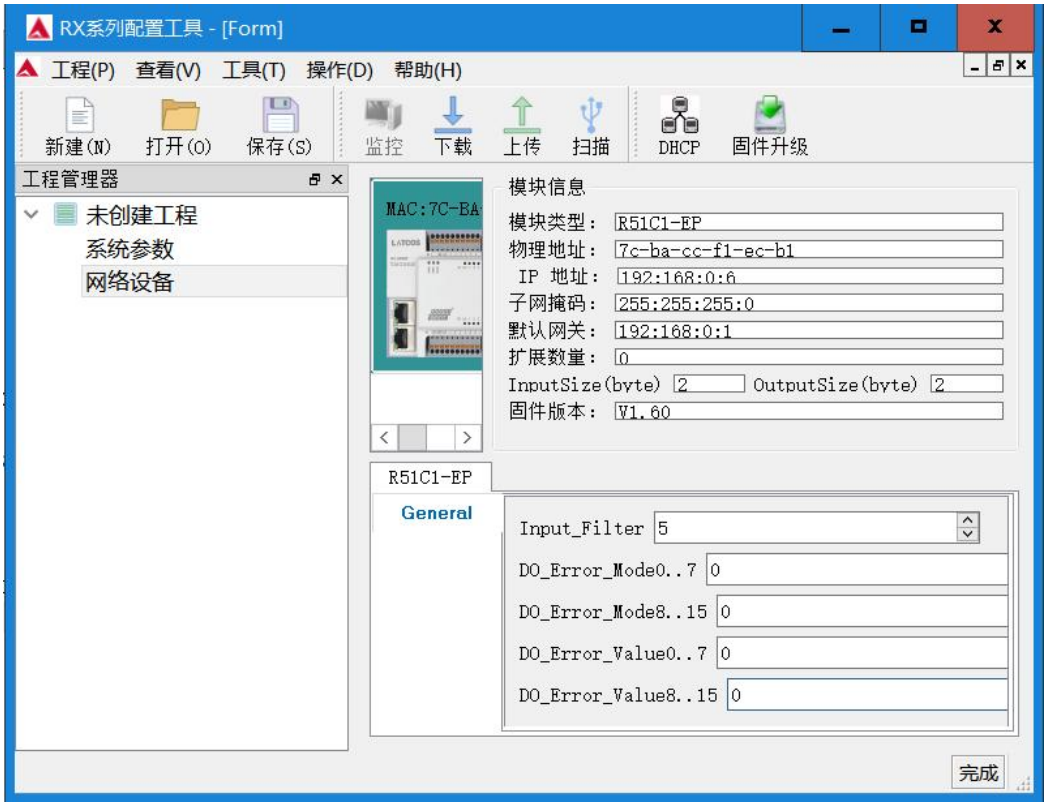


Step 4. Open the RxToolsetup software and set the IP address of the R51Cx-EP to 192.168.0.6

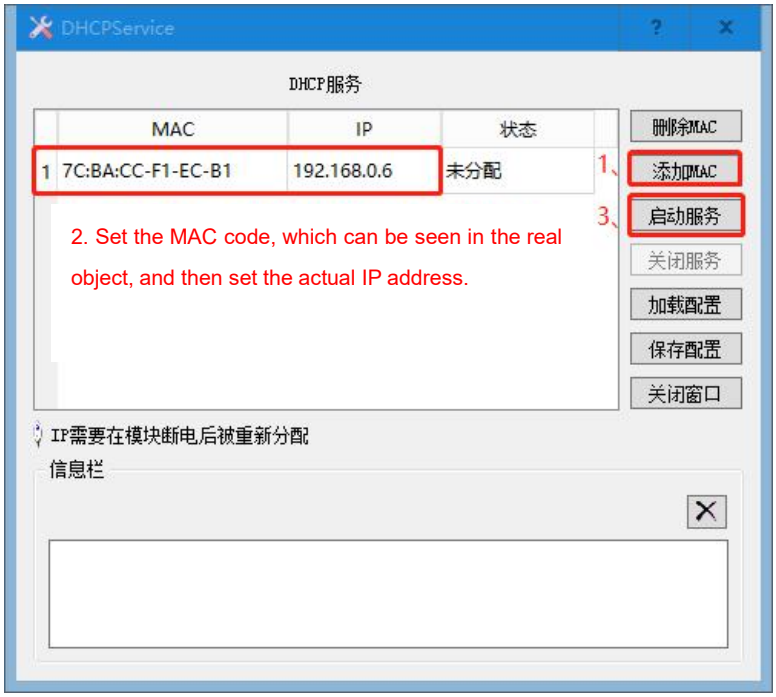
1) The IP address of R51Cx-EP should be in the same network segment as the computer network IP. After setting, click the scan icon 扫描 to display the scanned slave modules. Click "Network Device" to enter the module interface.



2) Double-click the slave module to see the actual IP address of the slave.



3) To modify the IP address of the slave, click "DHCP" on the toolbar, open DHCP Service, click "Add MAC" to add the MAC code to be assigned and set the IP address, click "Start Service" after the setting is complete, enter the assignment In the process of IP, power off and restart the module according to the prompt. The IP needs to be reassigned after the module is powered off. After the assignment is successful, the status will show that the assignment is successful.



4) Click Close Service after assignment

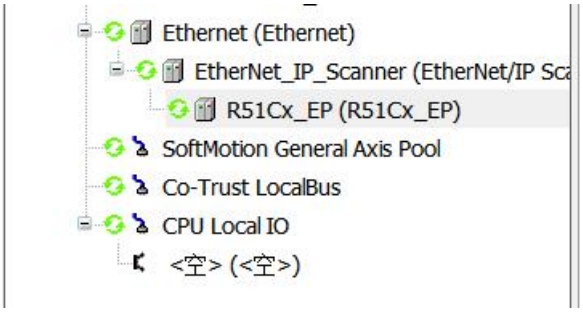
<Remark> After the R51C1-EP slave station has set the IP address and is powered on again, once the power is off again or the network cable is plugged in and unplugged, the assigned address will be lost, and the communication will be unsuccessful. To solve the problem of R51C1 dropping the address, turn off the DHCP of the router.

Step 5. Refer to 2.3 Set up communication and set the communication between C56-10 and the host computer

Step 6. Running and debugging

1) Select the menu item "Online" → "Login" to make the application establish a connection with the C56-10 and enter the online state; then, select "Debug" → "Start" to make the application in the C56-10 start operation.

2) The following is a schematic diagram of the successful connection of the master and slave stations.



7.2 High-speed counting application example

The C5X-10 motion controller CPU itself integrates 6 high-speed counters, and can also be equipped with HSC high-speed counter modules. This section will introduce the example of high-speed counter modules and CPU high-speed counters to realize the pulse signal feedback from the servo motor encoder. Counting and speed measurement.

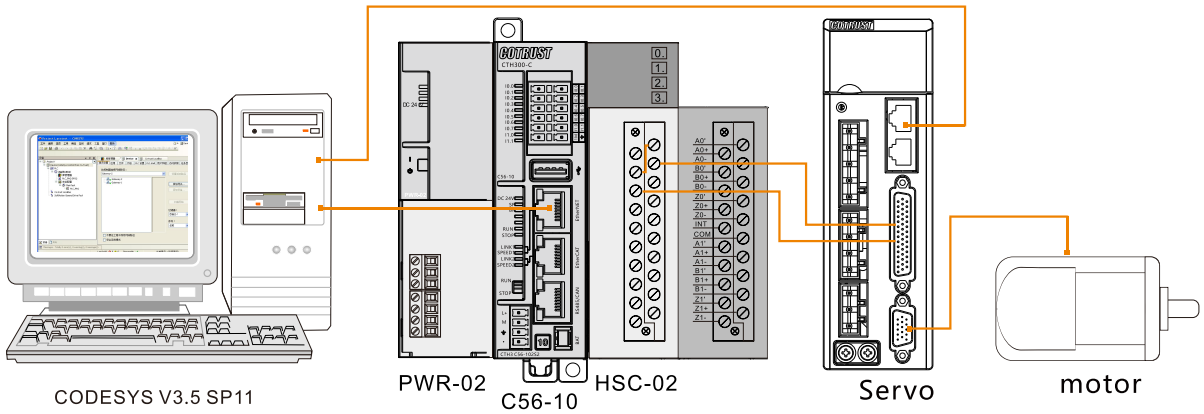
7.2.1 Application example of high-speed counting module

1. Preparation before communication

Table 7-9 Example Components

Components	Function
Programming device PG\PC	CODESYS V3.5 SP11 for configuration, programming and commissioning of the C56-10
Power Module PWR-02	Powers the C56-10 motion controller and its 24 VDC load circuit
C56-10 main control module	The CPU executes the user program, provides 5V voltage to the C56-10 backplane bus, and communicates with other modules through the Ethernet interface.
HSC-02 high-speed counting module	HSC-02 high-speed counting module is connected with C56-10 through the bus, and counts the motor speed and position.
Drive	A4S servo driver.
Servo motor	The servo motor is connected to the A4S servo driver.
Standard network cable	Connect programming device and CPU with A4S servo driver.
Encoder cable	Encoder cable connects the A4S servo driver to the motor.

2. Network connection



3. Operation steps

Step 1: Wiring

Open the front panel of the C56-10 and power module PWR-02 and wire them. Specific steps are as follows:

1) Use a standard network cable to connect the programming device and C56-10 (EtherNET communication port)

2) The C56-10 is connected with the high-speed counting module through the bus

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

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

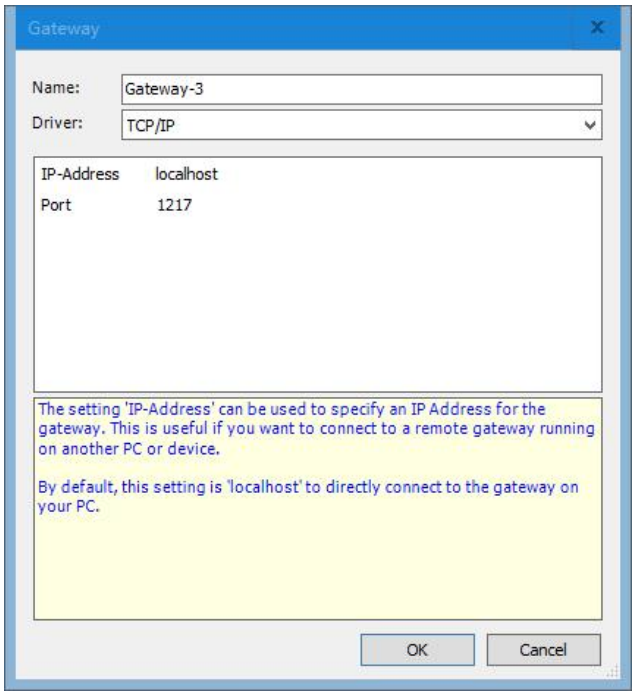
5) Wiring the high-speed counting module and the A4S driver

Step 2: Run the drive system

The motor starts to run normally by setting the parameters of the A4S servo driver. For the specific operation, please refer to the *A4S Series AC Servo Driver Instruction Manual*.

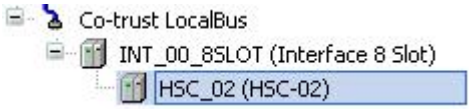
Step 3: Set up PLC communication

Refer to 2.3 Set up communication, and make communication settings in CODESYS so that the programming device can establish communication with the C56.



Step 4: Configuration in CODESYS

Right-click on "Co-Trust LocalBus" and select "Add Device", then select the supplier as "Co-Trust" in the pop-up dialog box, select "Special Device Interface 8 Slot", and finally click "Add Device" to add For this device, the successfully added device will be displayed under Co-Trust LocalBus. Finally, select the successfully added INT_00 and right-click to select "Add Device", and then select to add HSC-02 device.



<Remarks> Double-click the device HSC-02 to open its properties, and you can configure the module's mode, control word and other parameters. If you configure it here, you don't need to use the HSC_300 and HSC_SETMODE commands.

Step 5: Run the drive system

The motor starts to run normally by setting the parameters of the A4S servo driver. For the specific operation, please refer to the *A4S Series AC Servo Driver Instruction Manual*.

Step 6: Add the library file Extbus library

Click "Tools" → "Library" to select the installation library file. After the library file is installed, it will prompt that the addition is successful, and then add the successfully added library to the library manager. Please refer to the following operations:

1) Open the library manager from the device catalog: PLC Logic→Application→Library Manager

2) Then click on the library manager interface to add the library file "Co-Trust Extbus Library"

Step 7: Programming with Extbus Library

The library file of the HSC-02 module has been successfully added, and the information such as the speed, position and status of the module can be read directly through commands.

<Remark> If parameters such as control word and mode are configured in the HSC-02 module properties dialog box, there is no need to call the HSC_300 and HSC_SETMODE commands.

Step 8: Debug and Monitor the Program

1) Select "Online" → "Login" to establish a connection between the application and the C56-10 and enter the online state; then, select the "Debug" → "Start" to make the application in the C56-10 The program starts running.

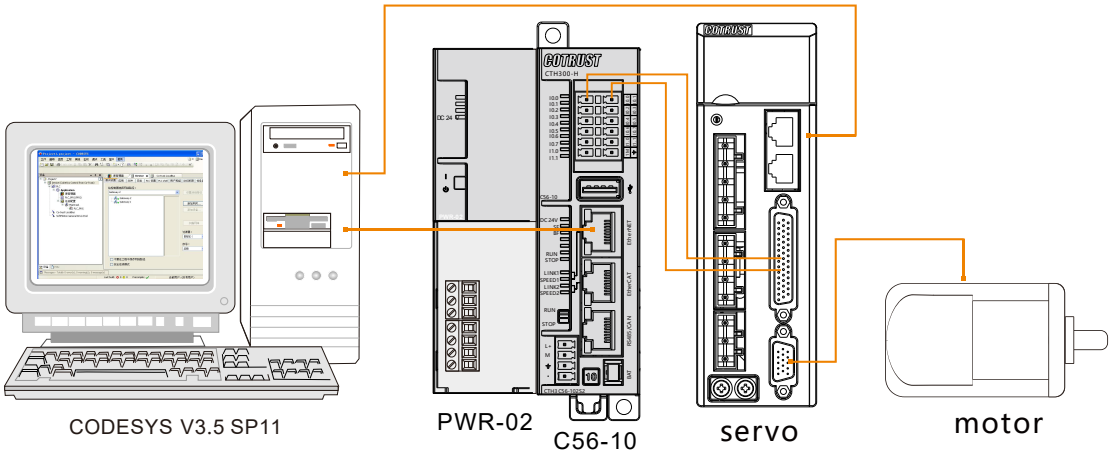
2) Debug program

Read the current position, speed and other values of the servo motor through HSC_GETCV, HSC_GETSPEED and other instructions in the program.

<Remarks> For details about the use of Extbus library, please refer to G Introduction of the ExtBus library.

7.2.2 Application example of CPU high-speed counter

Internet connection:



2. Operation steps

Step 1: Wiring

Open the front panel of the C56-10 and power module PWR-02 and wire them. Specific steps are as follows:

1) Use a standard network cable to connect the programming device and C56-10 (EtherNET communication port)

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

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

Step 2: Run the drive system

The motor starts to run normally by setting the parameters of the A4S servo driver. For the specific operation, please refer to the "*A4S Series AC Servo Driver Instruction Manual*".

Step 3: Set up PLC communication

Refer to 2.3 Set up communication and make communication settings in CODESYS so that the programming device can establish communication with the C56.

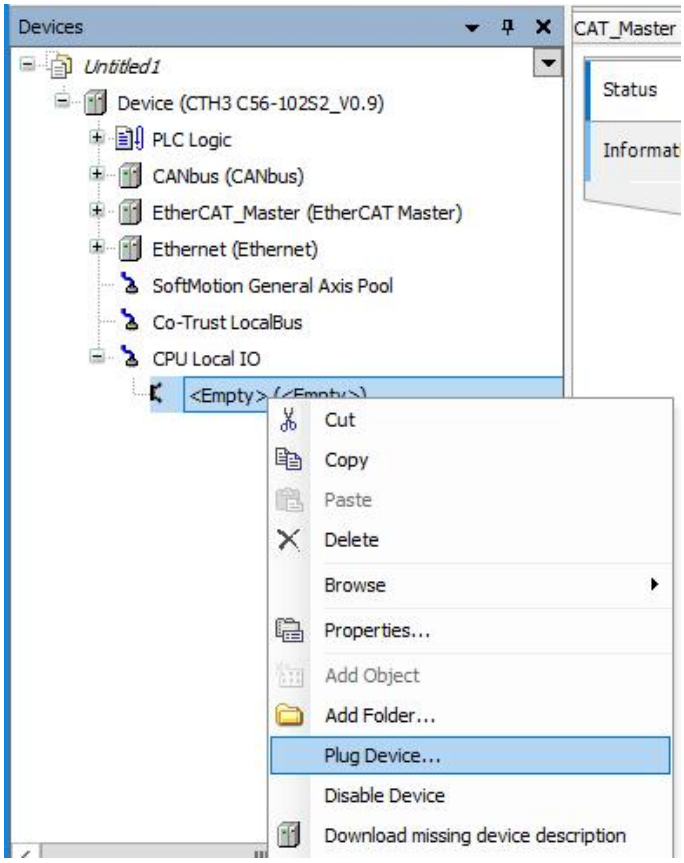
Step 4: Configure in CODESYS

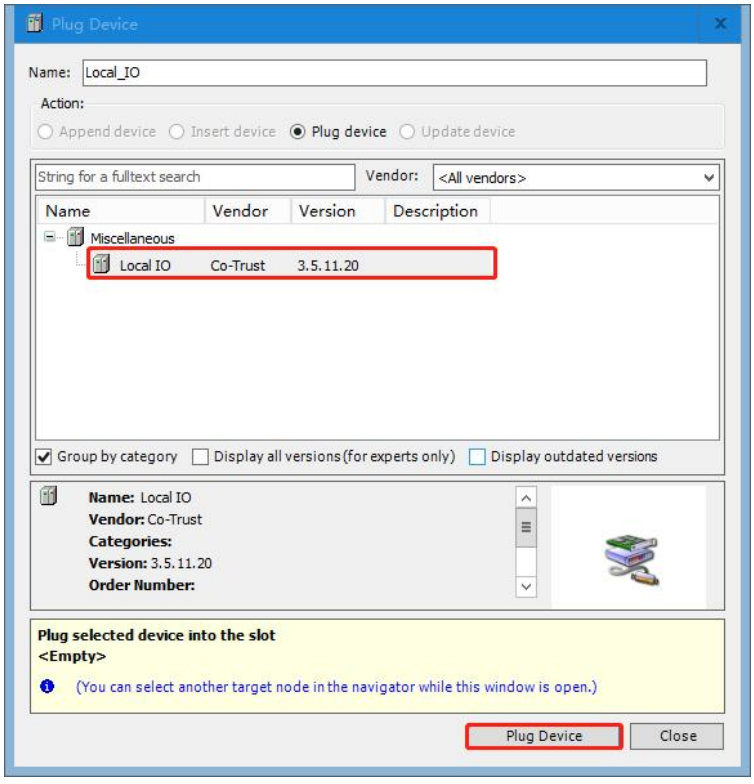
Start the CODESYS software and perform the following steps:

1) Create a new project

- Select "File" → "New Project" on the main page of DODESYS, then select "Standard project" in the pop-up dialog box and set the file name and storage directory. Then select the newly added CPU file device (CTH3 C56-10S2_V0.9) and the programming language used, and then click "OK" to complete the creation of the project.

- Right-click "<empty> <empty>" under "CPU Local IO" in the device catalog, select "Insert Device", and then select the supplier as "Co-Trust" in the pop-up dialog box, and select "Local IO". IO", and finally click the "Insert Device" to add the device, the successfully added device will be displayed under the CPU Local IO.

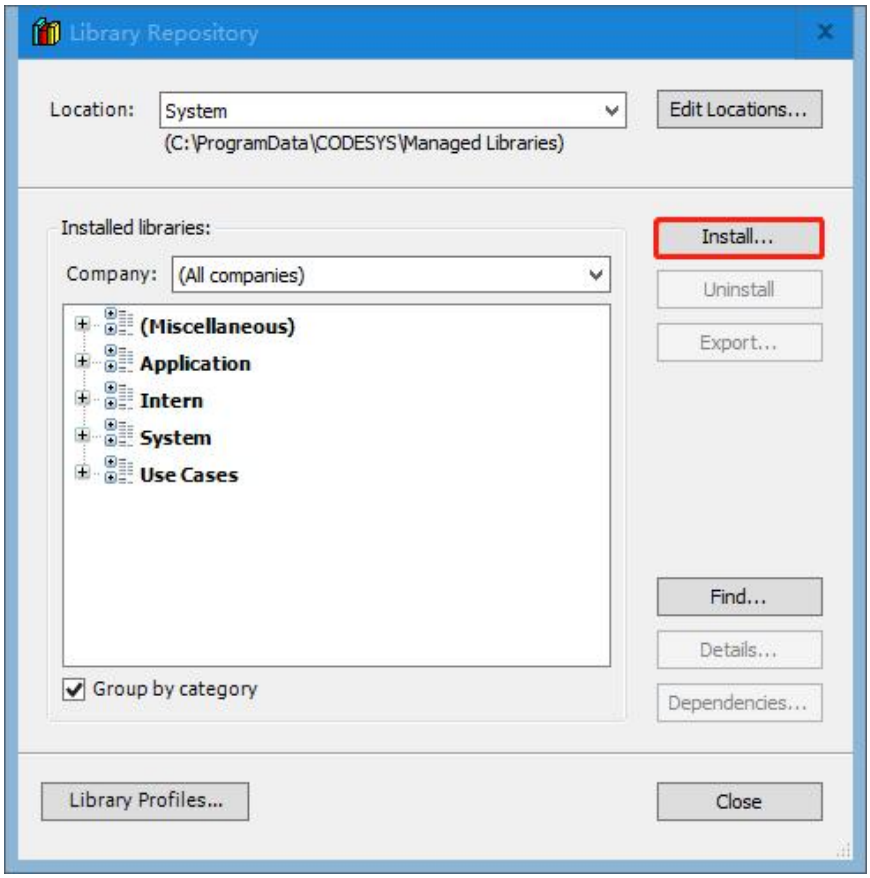




2) Install library files

After the installation is successful, the high-speed counter that comes with C5X-10 can be used in the program system. The specific installation operations are as follows:

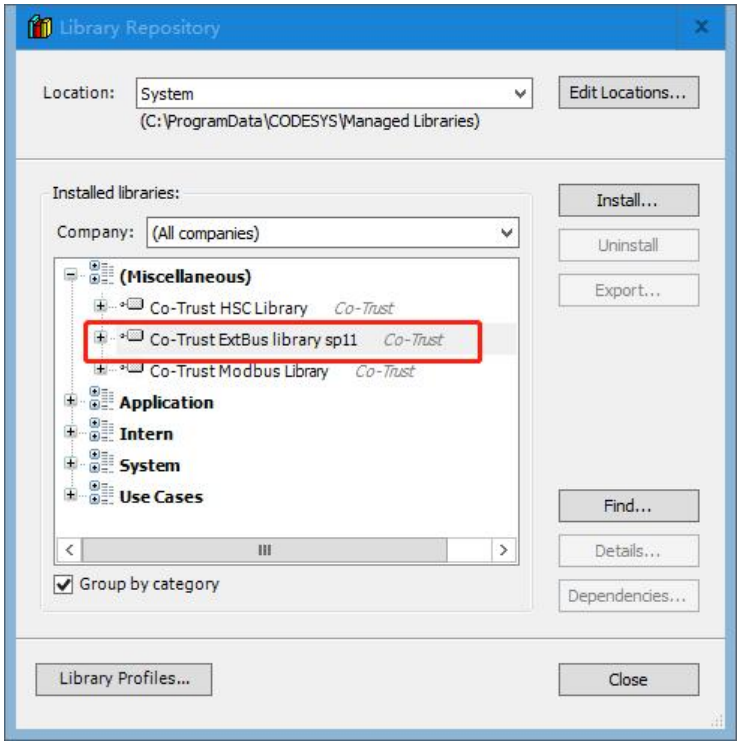
Select the "Tools" → "Library Repository" , click "Install".



Select "Co-Trust_ExtBus_SP11V1.2.library". Click "Open" to confirm the option, and the new device will be added to the device catalog in the "Device Library".

After the above operations are completed, you can refer to the following steps to e check whether the files are installed correctly:

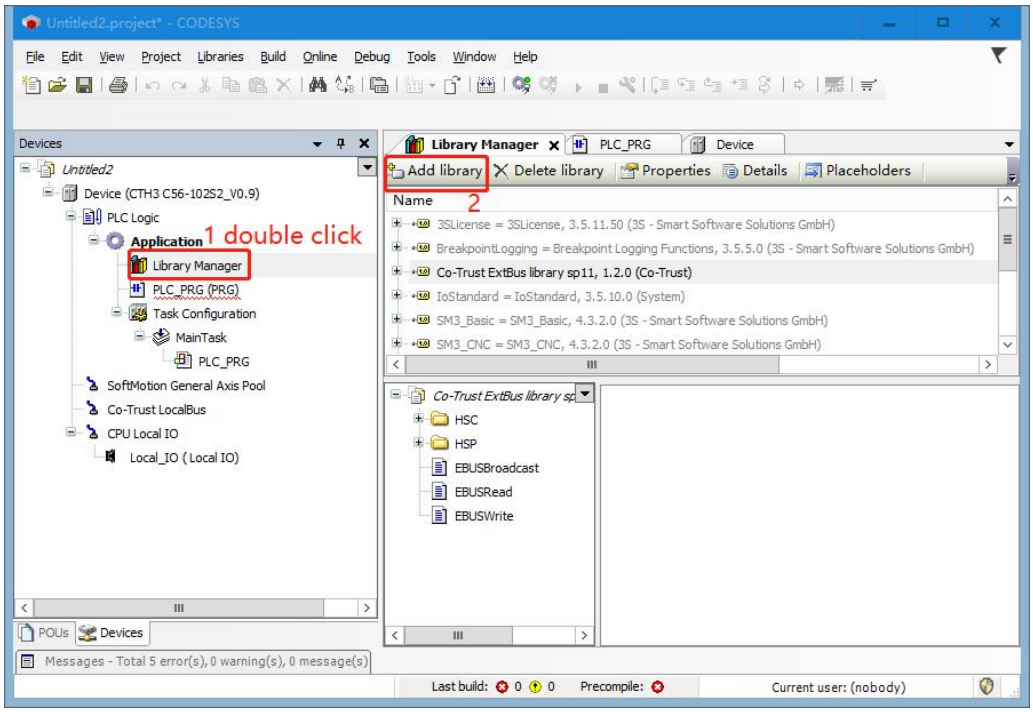
- Select the "Tools" → "Library Repository " → "Miscellaneous" to see the installed Co-Trust_ExtBus_SP11V1.2.library.



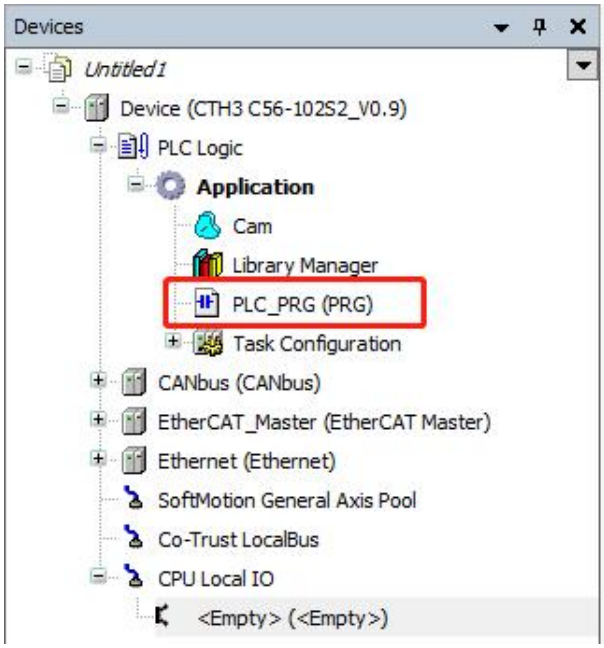
3) Call library file programming

After the library file is added successfully, the instruction of the high-speed counter can be called for programming, taking the HSC_GETCV instruction as an example.

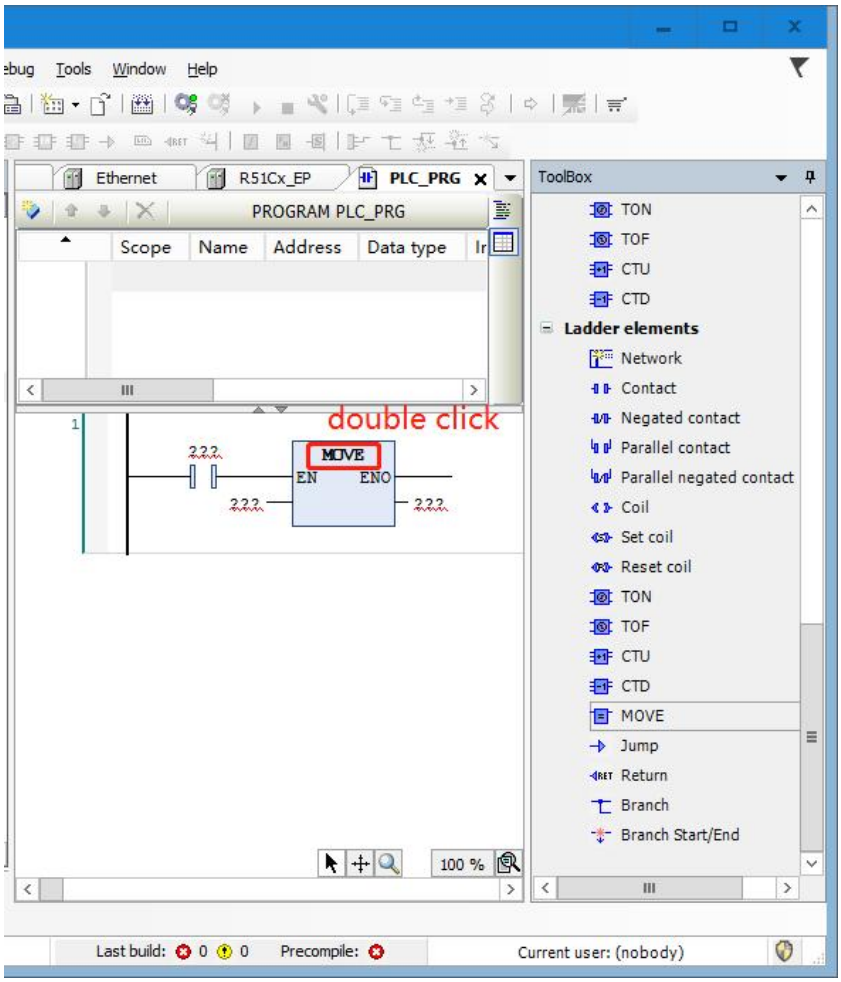
Double click “Library Manager“→“ Add Library“, add Co-Trust_ExtBus_SP11V1.2.library to this project.

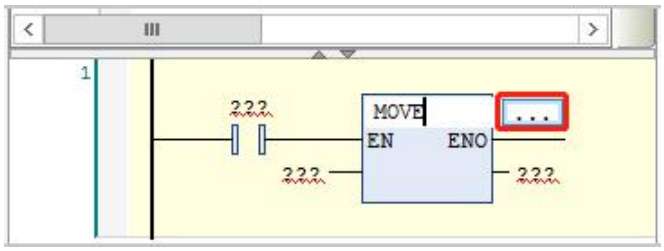


Double click PLC_PRG(PRG) to start to program.

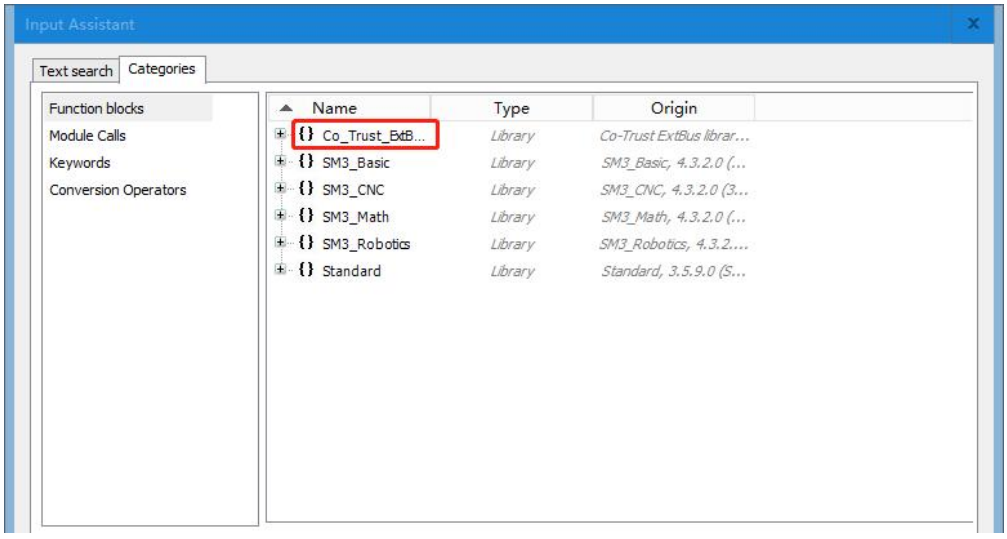


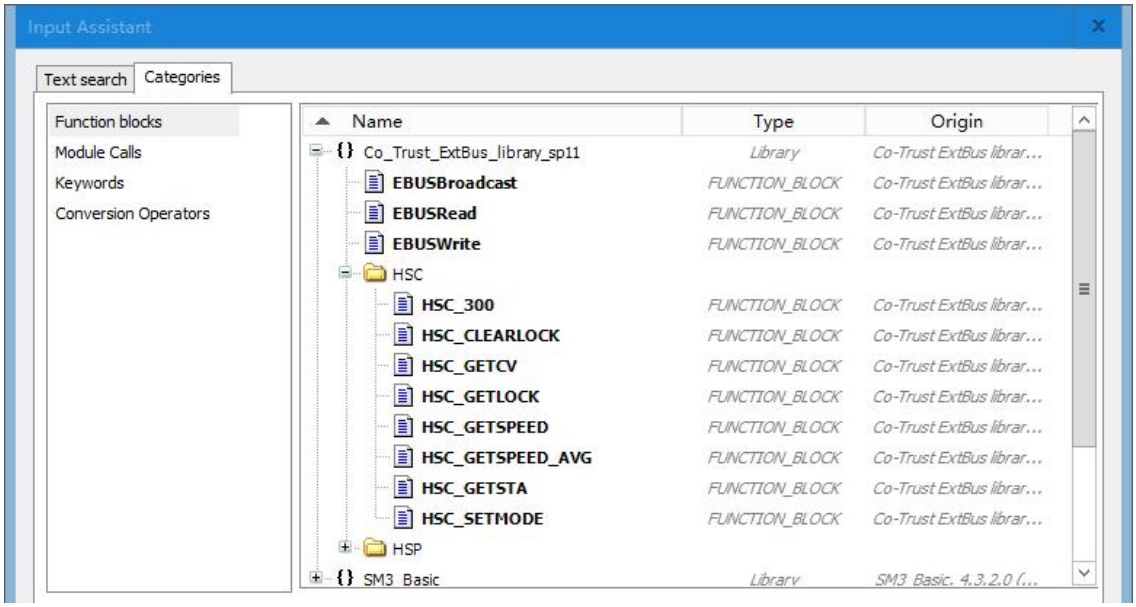
Drag a function block with EN/ENO from the "Toolbox" on the right, click the three question marks in the function block box, and an option box will appear.



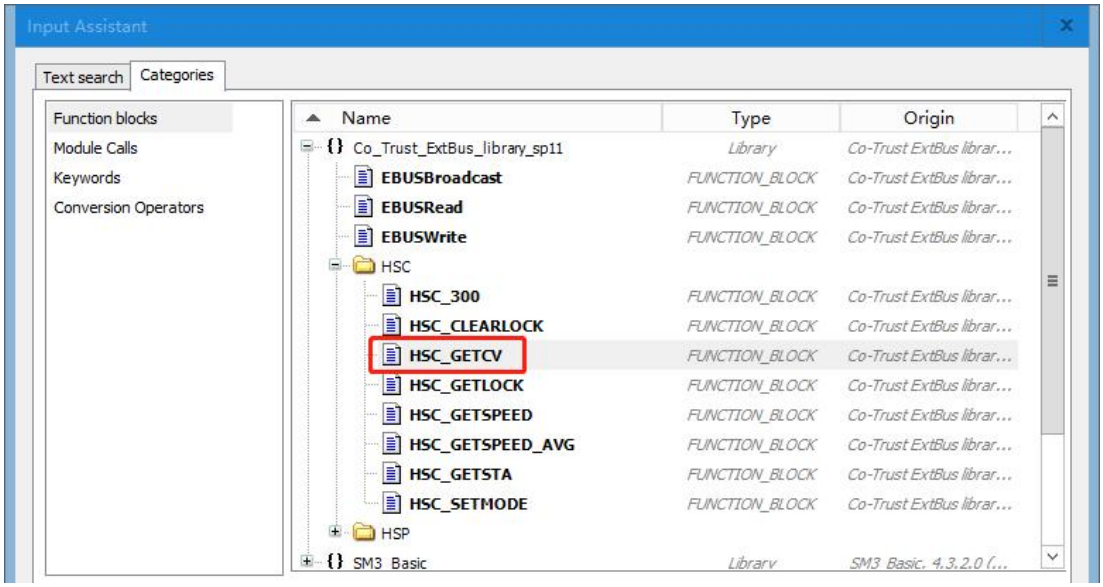


Select "Co-Trust_ExtBus" → "HSC" to see the instructions related to the high-speed counter, and click the required instruction to confirm to transfer the instruction into the program.

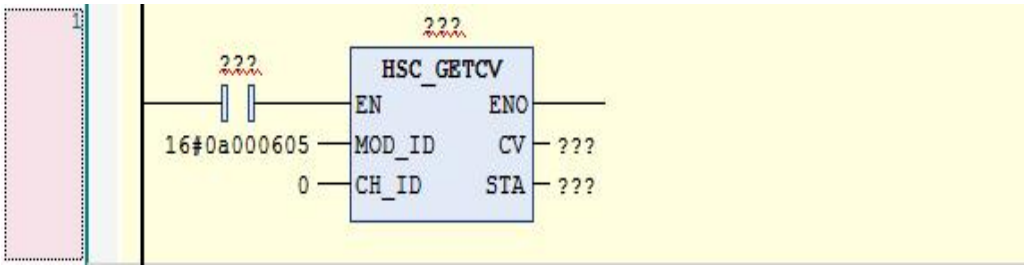




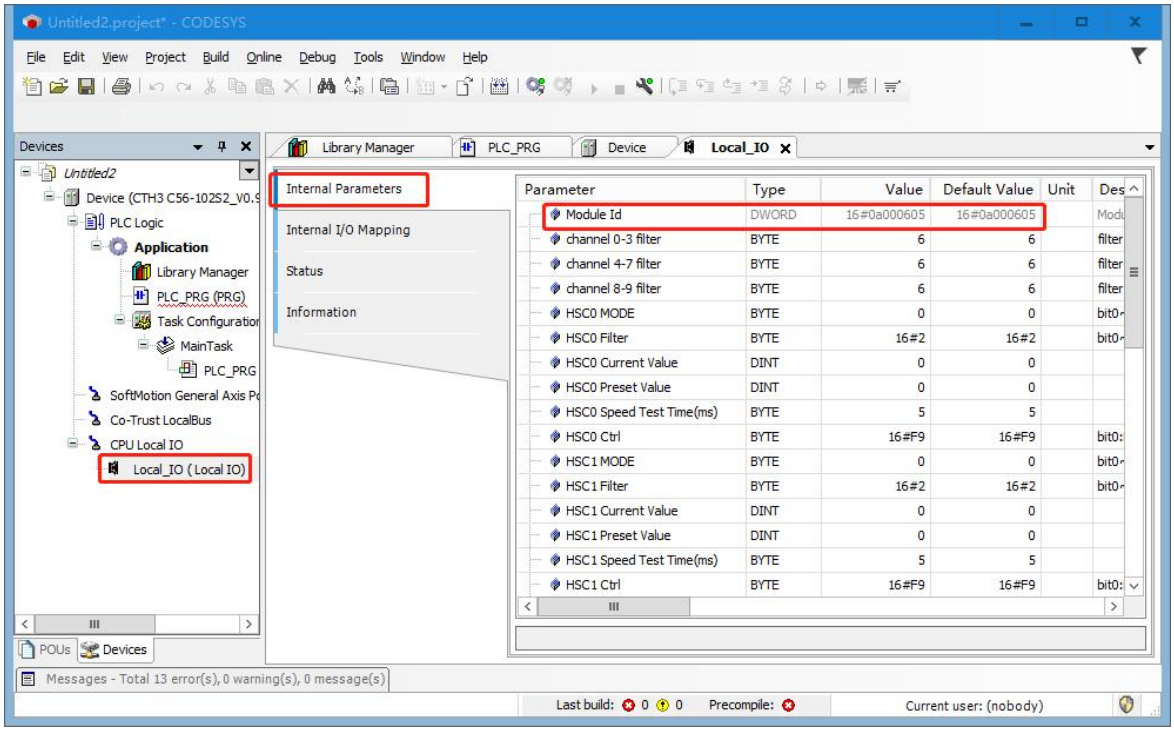
Select the HSC_GETCV instruction.



After calling this instruction, the value used by the instruction input parameter MOD_ID is the default value of MOD_ID in "Internal configuration".



Click "Local_IO" in the device catalog, select "Internal configuration", you can see the default value of MOD_ID.



Remarks: The value of input parameter MOD_ID of all HSC related commands uses the default value of MOD_ID of "Internal configuration" in "Local_IO".

Step 5: Debug and monitor the program

1) Select "Online" → "Login" to establish a connection between the application and C56-10, and enter the online state. Then, select "Debug" → "Start" to start the application in C56-10.

2) Debug program

Read the current position, speed and other values of the servo motor through HSC_GETCV, HSC_GETSPEED and other instructions in the program.

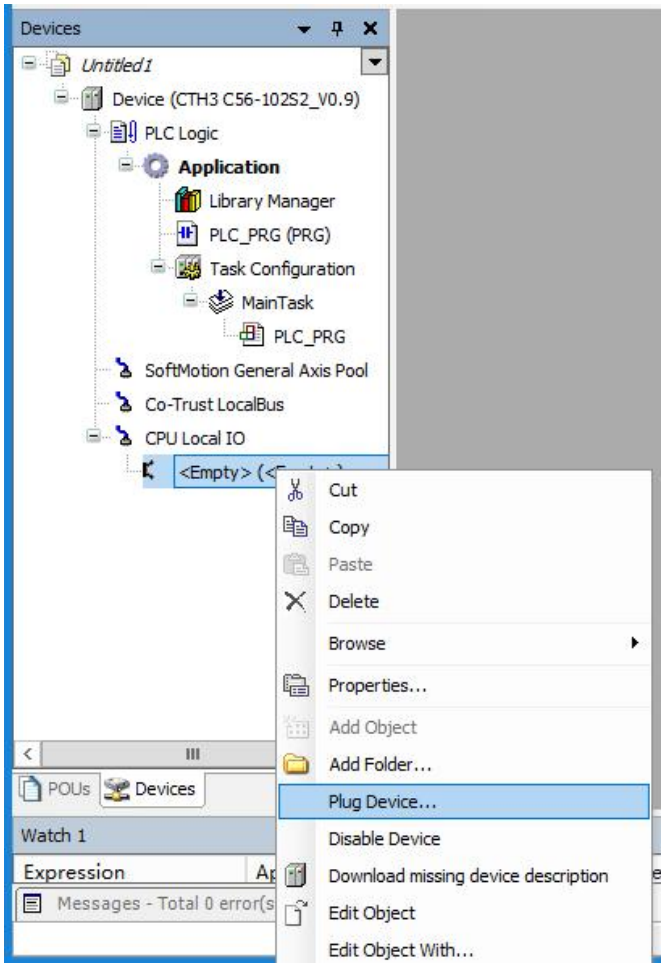
7.3 Example of HSC Interruption

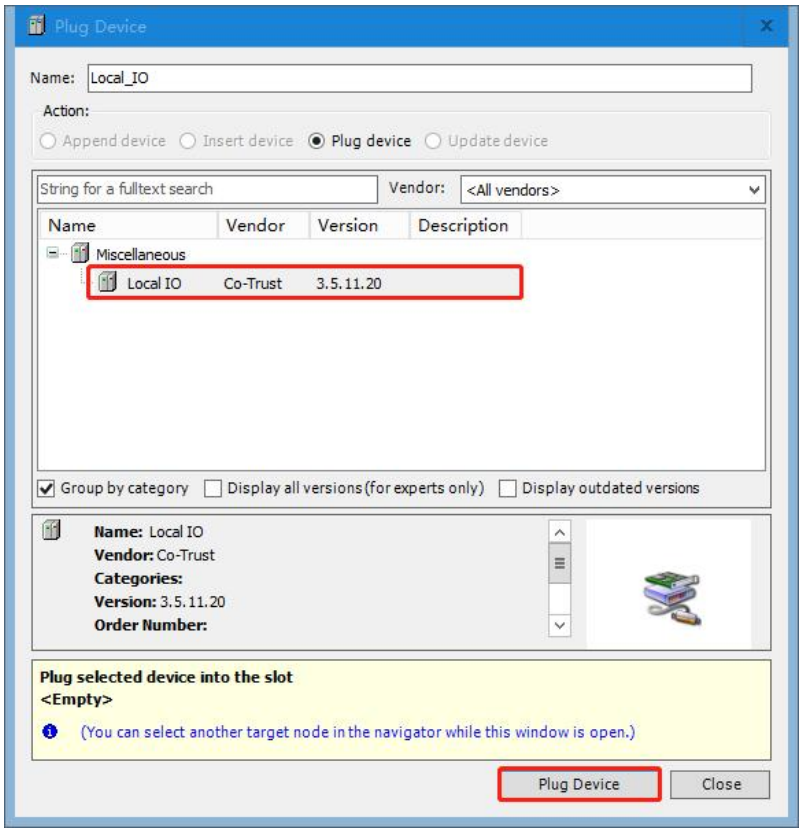
1. Configure in CODESYS

1) Create a new project

- Select "File" → "New Project", then select "Standard project" and set the file name and storage directory. Select the device (CTH3 C56-10S2_V0.9) and programming language, and then click "OK" to complete the creation of the project.

- In the device catalog, right-click "<empty> <empty>" under "CPU Local IO", select "Plug Device", then select "Local IO", and finally click " Click the "Insert Device" button to add the device, the successfully added device will be displayed under the CPU Local IO.

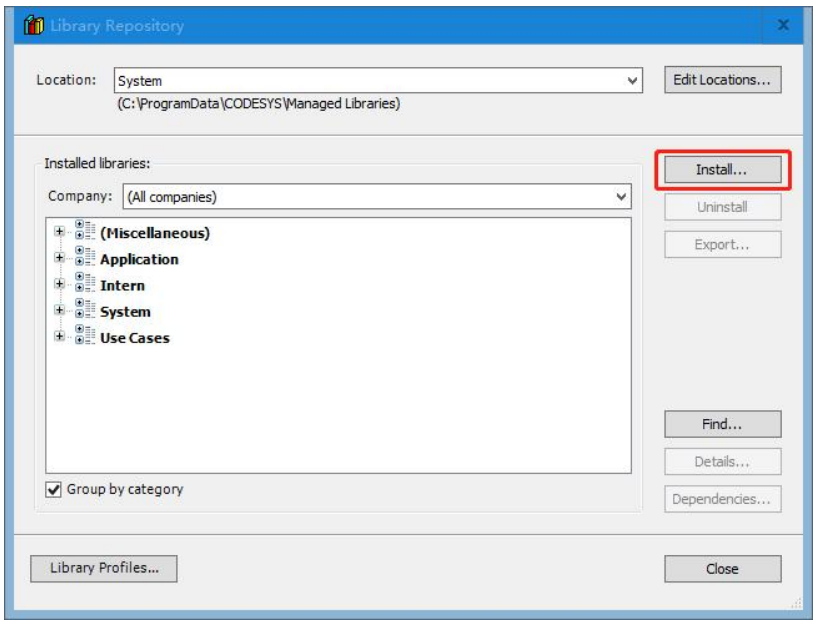




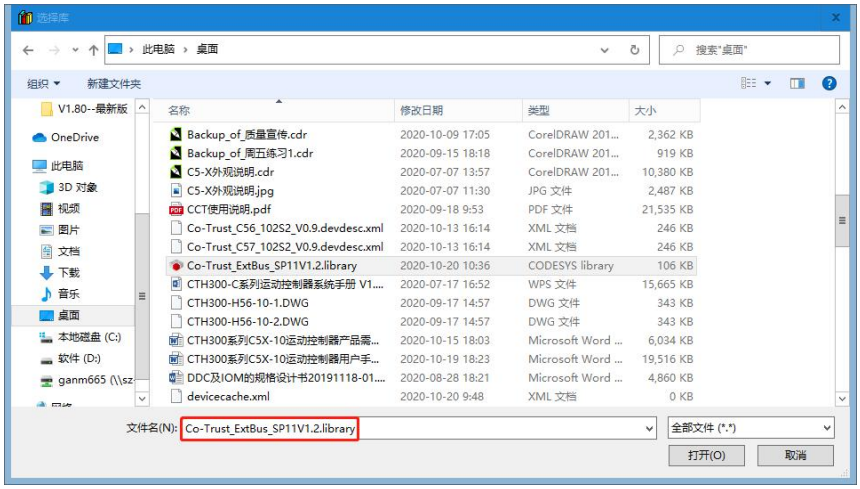
2) Install library files

Before using the C5X-10 automatic high-speed counter, you need to install the Exbus library. The specific installation operations are as follows:

Select "Tools" → "Library Repository" to open the following dialog box, click "Install".

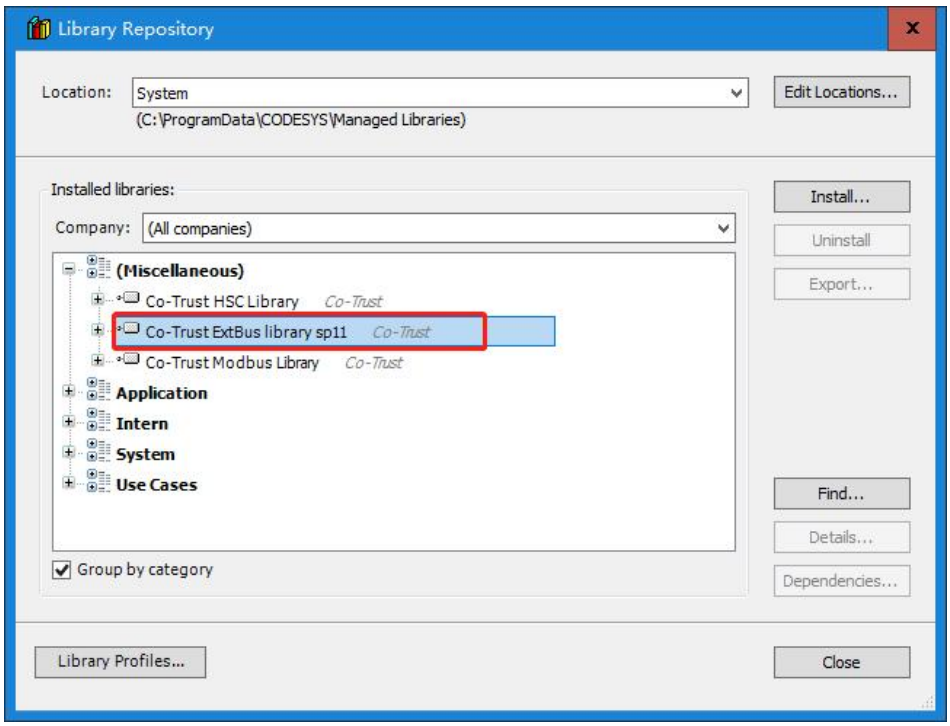


Select "Co-Trust_ExtBus_SP11V1.2.library". Click the "Open" to confirm the selection, and the new device will be added to the device catalog in the "Device Library".



Check if the file is installed:

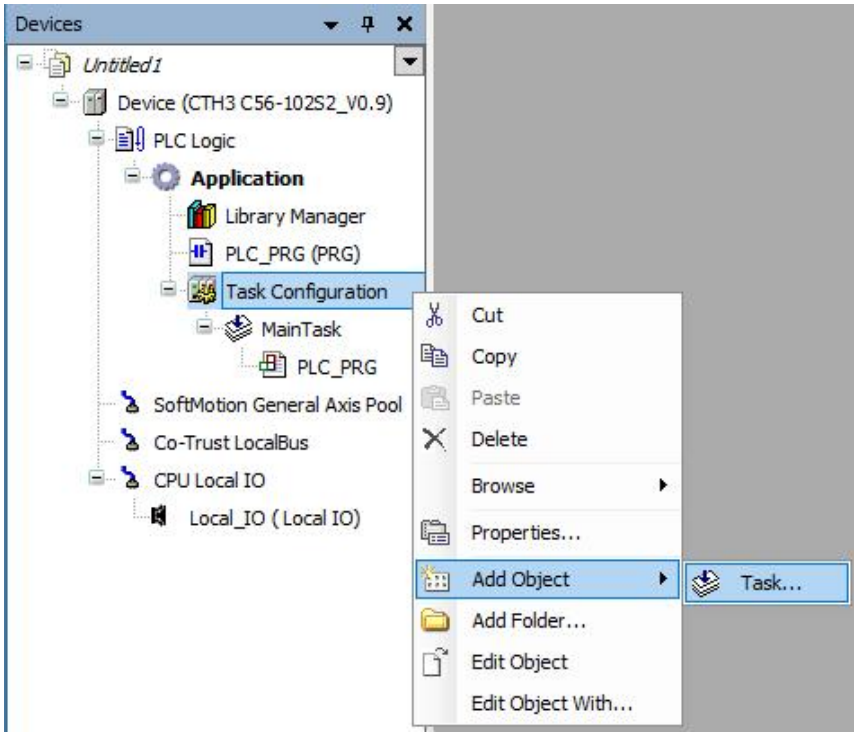
- Select "Tools"→"Library Repository "→"Miscellaneous" to see the installed Co-Trust_ExtBus_SP11 V1.2. library.



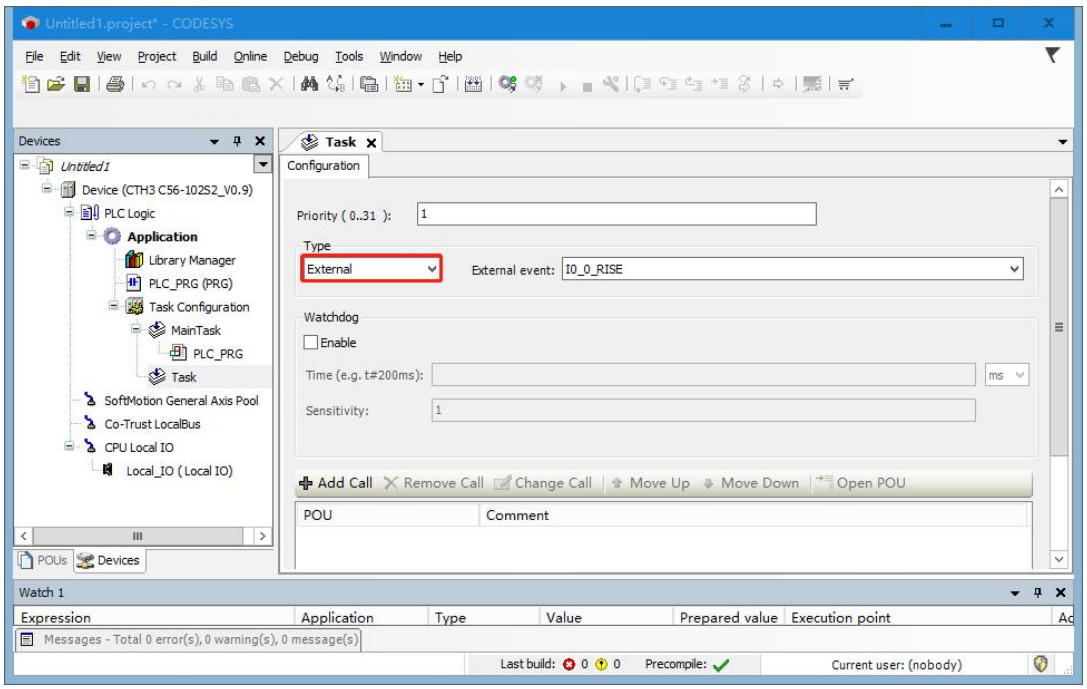
3) Call library file programming

2. Set interrupt

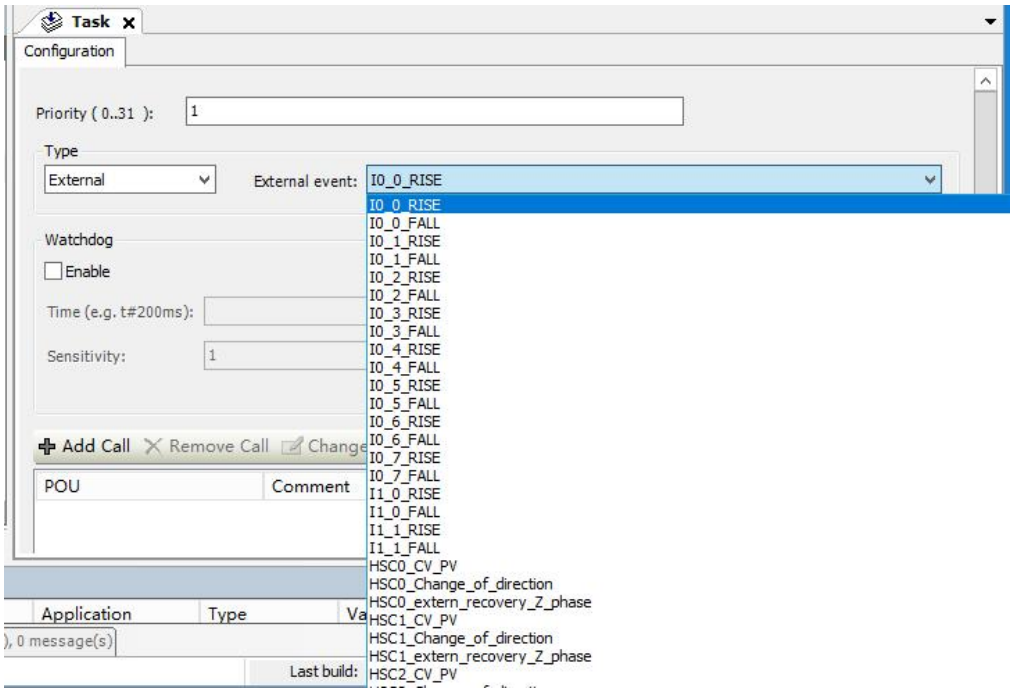
Right-click "Task Configuration", select "Add Object", and then select "Task" to create a new "Task".



Click new "task" to set the interrupt. First, select "external" as the configuration type.

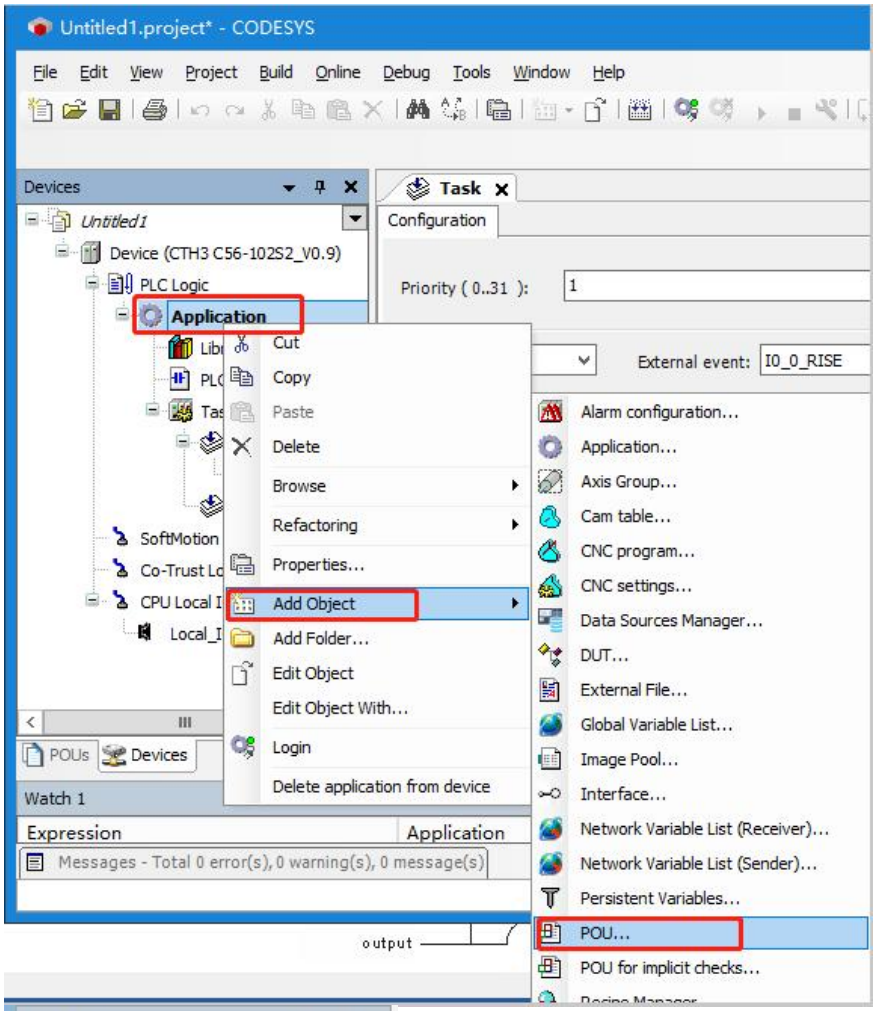


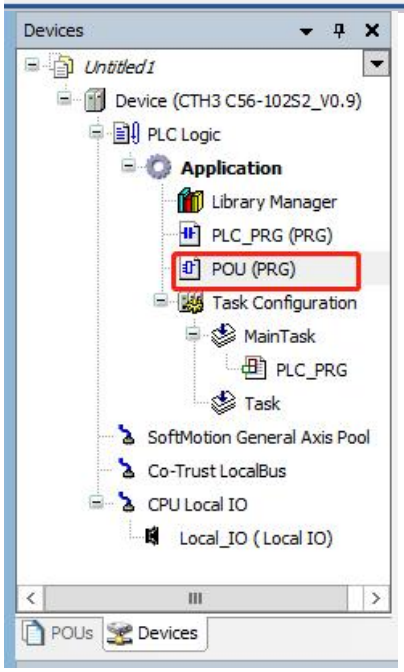
Next, select external events. There are 30 kinds of external events to choose from.



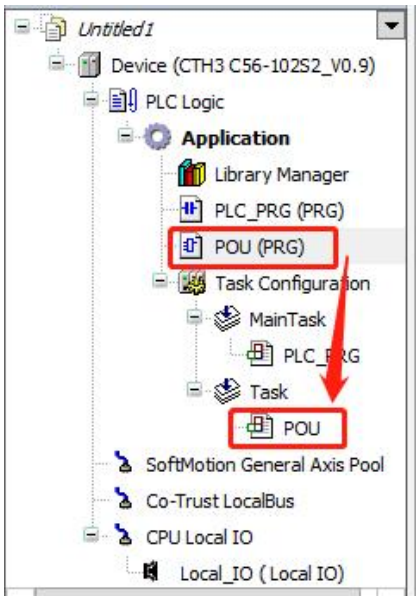
3. Connection interruption

Right-click "Application" → "Add Object" → "Program Organization Unit", click "Open" to create a new POU.



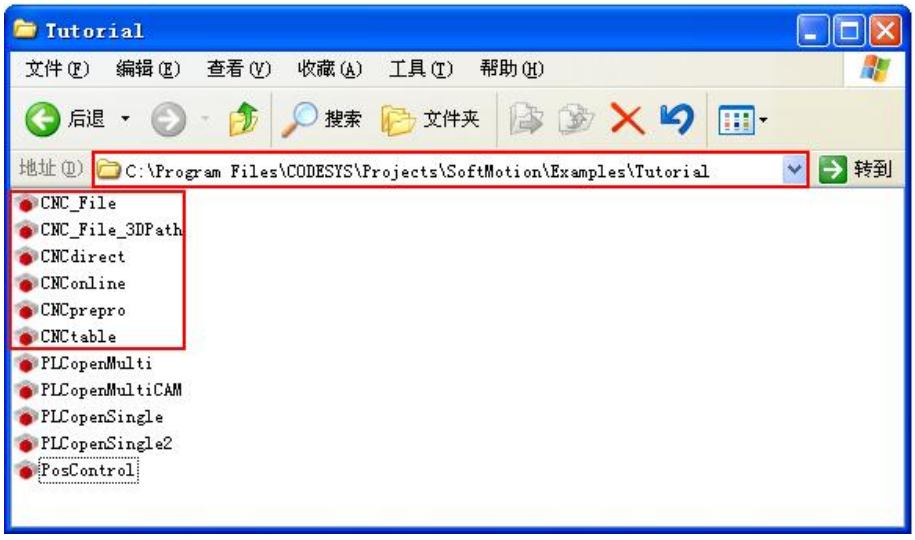


Drag the new POU to the bottom of the configured "Task" (interrupt task), and the interrupt subroutine can be written in this POU. When an interrupt occurs, the subroutine below the corresponding interrupt task will be executed once.



7.4 CNC project

Please get the project about the CNC function in the CODESYS installation directory:



Appendix

The appendix introduces various library commands, quick query information, ordering information, etc. The details are as follows:

A Terminology Explanation

Table A-1 Explanation of Abbreviated Terms

Abbreviation	Full Name
AC	Alternating Current
DC	Direct Current
ADC	Analog to Digital Converter
DAC	Digital-to-Analog Converter
CODESYS	Controlled Development System
PLC	Programmable Logic Controller
CPU	Central Processing Unit
HMI	Human Machine Interface
FB	Function Block
FC	Function
HSC	High-speed counter
IEC	International Electro Technical Commission
SDO	Service Data Object
PDO	Process Data Object
RTD	Resistance Temperature Detector
TC	Thermocouple
UDP	User Datagram Protocol
IL	Instruction List
ST	Structure Text
SFC	Sequential Function Chart
CFC	Continuous Function Chart
FBD	Function Block Diagram
LD	Ladder Diagram

B Power Budget

After selecting 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 5V DC, 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 Intermediate expansion module.

Condition 2: Confirmation of power consumption

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

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

Table B-1 5V DC bus current supply and consumption

Name	Supply Current	Current Consumption
C56-10	1600mA	--
C57-10	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

Table B-2 24V DC bus current supply and consumption

Name	Supply Current	Current Consumption
PWR-02	2000mA	--
C56-10	--	800mA
C57-10	--	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

C Expansion Module Specifications

C.1 Power supply module

The PWR-02 power supply module can provide 24V DC for C5X -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 C-1 Basic Attributes of Power Module PWR-02

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

Table C-2 General Characteristics of Power Module PWR-02

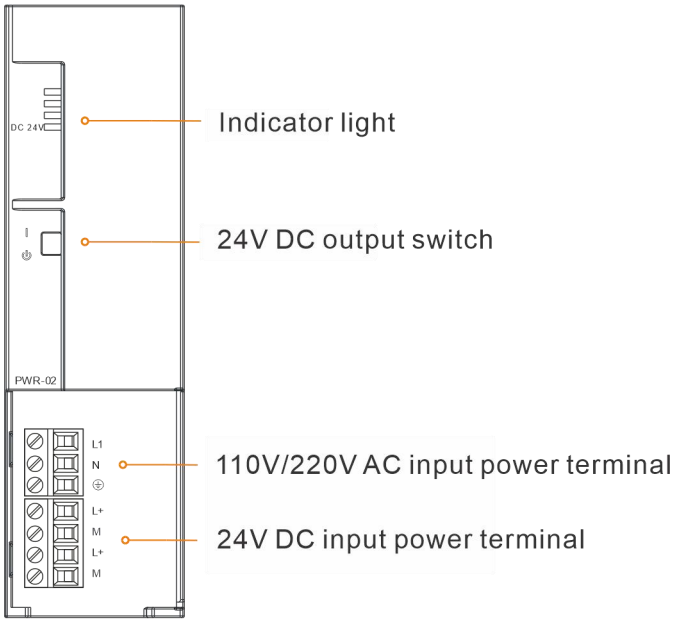
Physical characteristics	
dimensions(W×H×D)	34×115×101.6 mm
LED indicator	
24VPOWER(green)	ON: with 24V DC output, OFF: without 24V DC output
Switching	
24V DC power control switch	ON: with 24V DC output, OFF: without 24V DC output

Table C-3 Functional Characteristics of Power Module PWR-02

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
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.

Interface diagram



Interface Diagram

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

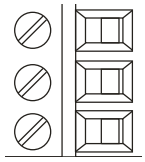
Removable terminal	Signal	Definition
	L	FireWire
	N	Zero line
		Grounded

Table C-5 Definition of the 24V DC output power port of the

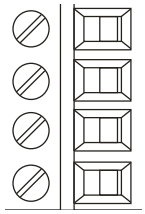
Removable terminal	Signal	Definition
	L+	24V+
	M	24V-
	L+	24V+
	M	24V-

Table C-6 DIP switch definition of PWR-02

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

C.2 Digital Module

Table C-7 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

Digital Input Module

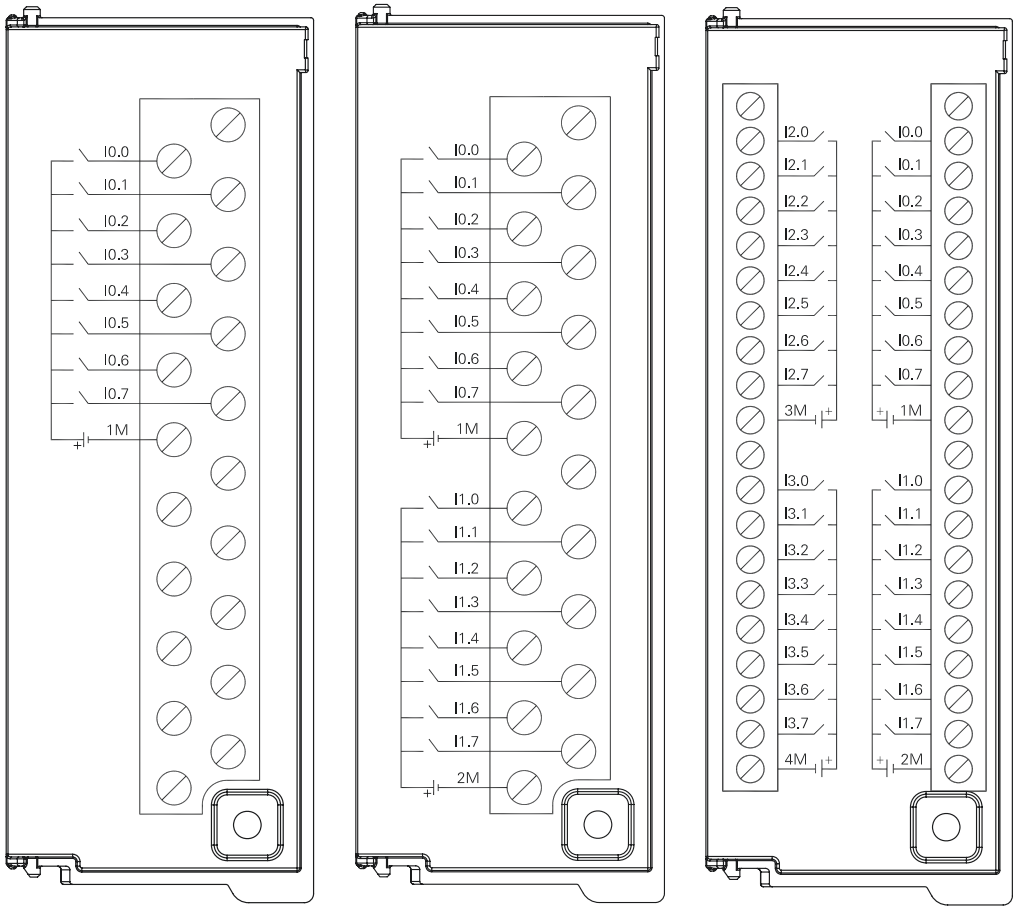
Table C-8 Specifications of Digital Input Module

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	4mA/ channel	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		1mA		

(maximum)				
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	8	8
Simultaneous ON points		8	16	32
Cable length	Shield	500m		
	Unshielded	300m		

Wiring

◆ DIT-080S1 ◆ DIT-160S1 ◆ DIT-320S1



Digital Output Module

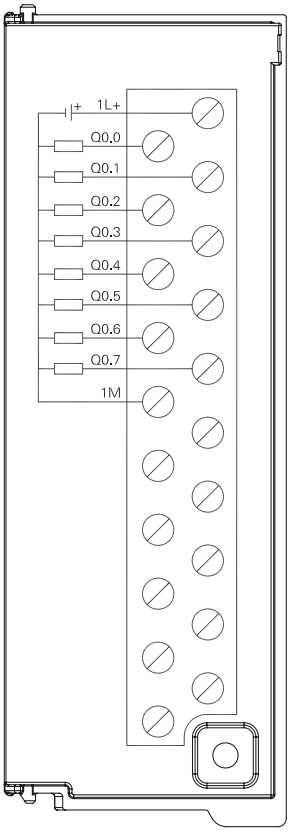
Table C-9 Digital Output Module Specifications

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	

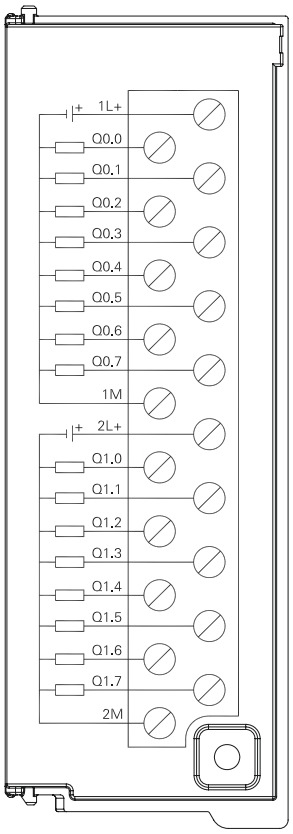
Input rated voltage		24V DC			DC: 24V,AC: 110V/220V	
Output voltage range		20.4~28.8V DC			DC: 5~30V,AC: 5~250V	
Logic 1 (min)		20V DC			-	
Logical 0 (max)		0.1V DC,10kΩ load			-	
Maximum output current		0.5A			2A	
Current at each common terminal		Maximum 4A			Maximum 16A	
Maximum output leakage current		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				

Wiring

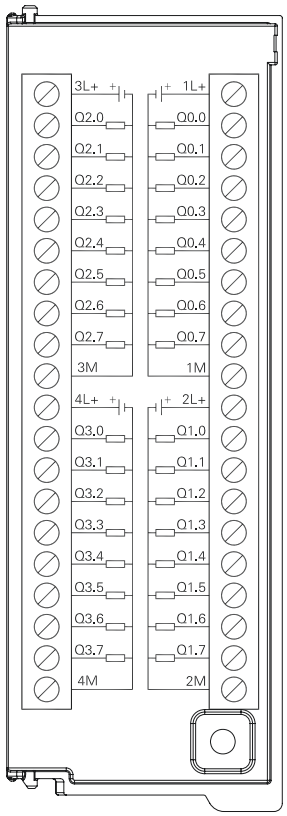
◆ DQT-080S1



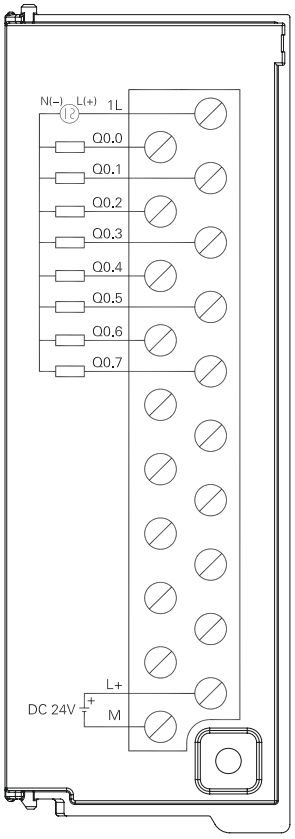
◆ DQT-160S1



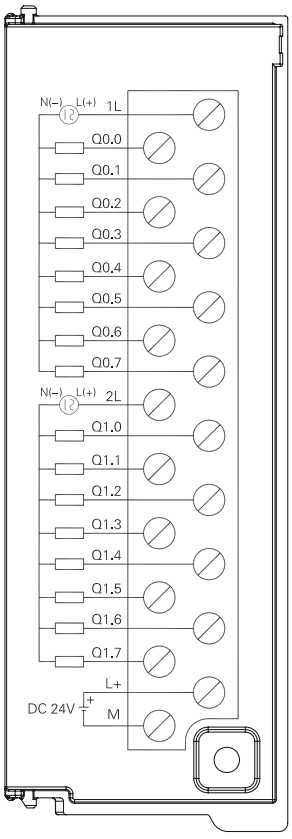
◆ DQT-320S1



◆ DQR-080S1



◆ DQR-160S1



C.3 Analog Module

Table C-10 Basic properties 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

Table C-11 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)				

C-12 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

Analog Input Module

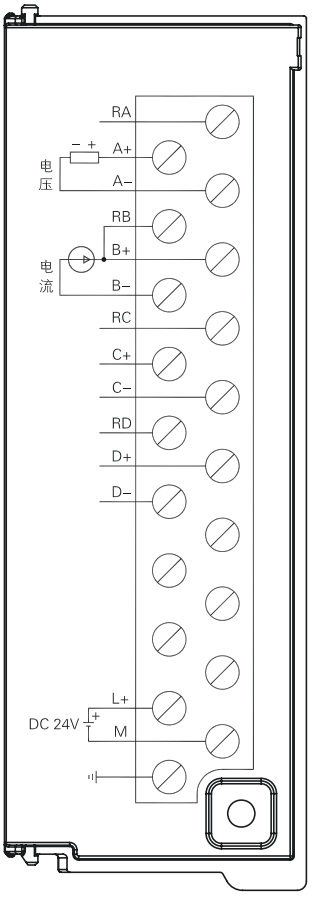
Table C-13 Analog Input Module Specifications

Item	AIS-040S1	AIV-080S1	AIC-080S1
Input type	Voltage or current (differential input)	Voltage input	Current input

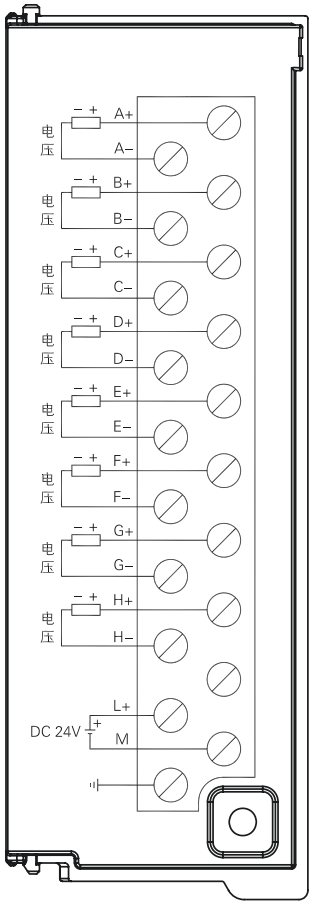
Input		4	8
Input range	Voltage	Unipolar: 0 ~ 5V, 0 ~ 10V, bipolar: ± 2.5V, ± 5V	
	Current	0~20mA, 4~20mA	
Maximum input	Voltage	30V DC	
	Current	40mA	
input resistance	Voltage	≥2MΩ	
	Current	250Ω	
Data Format	Voltage	0~+32000(unipolar), -32000~+32000(bipolar)	
	Current	0~+32000	
Input step response		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

Wiring

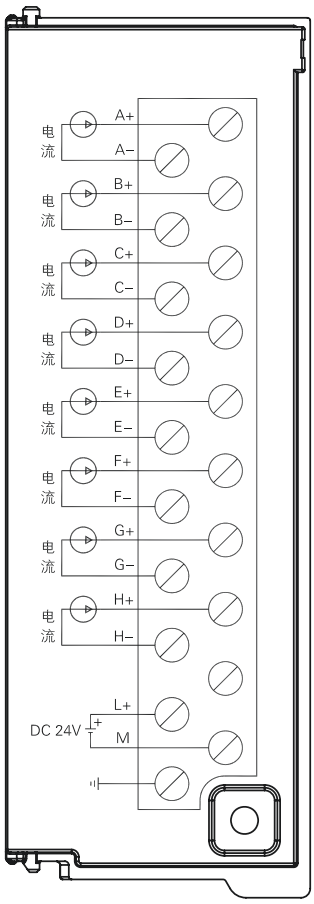
◆ AIS-040S1



◆ AIV-080S1



◆ AIC-080S1



Analog Output Module

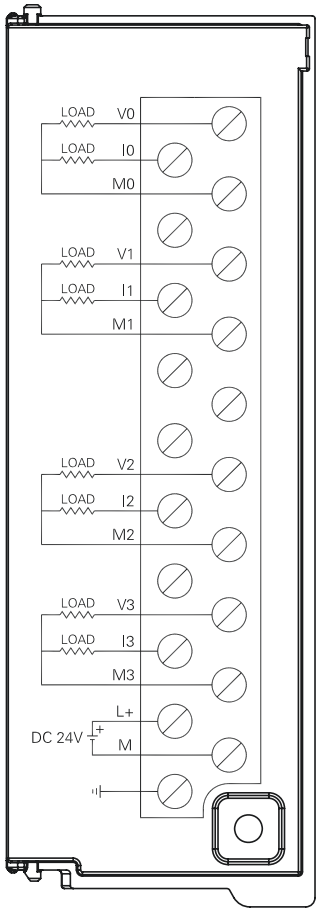
Table C-14 Analog Output Module Specifications

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	

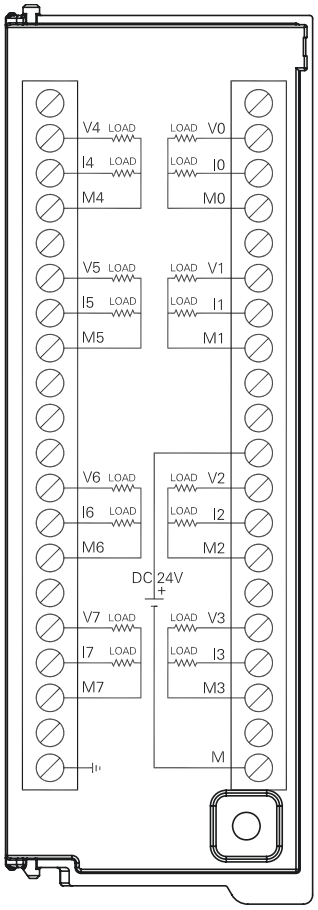
		case: ± 2% of full scale
Quarantine	Power isolation	500V AC

Wiring

◆ AQS-040S1



◆ AQS-080S1



Analog Input and Output Module

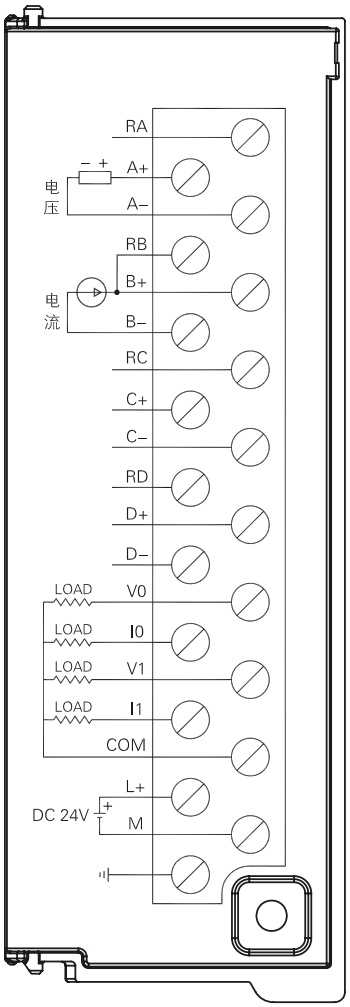
Table C-15 Specifications of analog input and output modules

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 5ms (fastest)		Input step response 4 channels 5ms (fastest)	
Module update frequency (all		4 channels support	8 channels support 50Hz, 20Hz,

channels)		200Hz, 100Hz, 50Hz, 20Hz, 10Hz configuration Default: 50Hz all channels	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		

Wiring

◆ AMS-060S1



C.4 Temperature Module

Table C-16 Basic Properties of Temperature Modules

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

Table C-17 General Characteristics of 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		

Table C-18 Functional Characteristics of Temperature Modules

functional category	function name, identifier	describe
I/O function	I/O interface	Provide 4-way/8-way temperature sensor input interface
extensions	Extensions	Provide bus expansion function
Isolation function	Signal isolation	Isolation between field and bus
	Power isolation	Isolation between external power and system power
Protective function	power protection	The power supply end provides reverse polarity protection and surge absorption
	Disconnection detection	Input provides disconnection detection
Filter function	Filter function	Combination of hardware filtering and software filtering
Power function	Power function	he module is powered by 24V DC

Table C-19 Input Characteristics of Temperature Modules

Characteristic	AIT-040S1	AIT-080S1	AIR-040S1	AIR-080S1
Input type	Suspended Thermocouple		Reference ground thermal resistance	
Enter the number of points	4	8	4	8
Wiring	-		Support 2-wire, 3-wire, 4-wire; Default: 3-wire	
Input range	Thermocouple type (choose one): S , T, R, E, N, K, J voltage range: ±80mV default: K		Thermal resistance type (choose 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)
Isolation Field to Logic Field to 24VDC 24VDC to logic	 500VAC 500VAC 500VAC	
Common Mode Rejection	＞100dB@120VAC	
Input resolution Temperature Voltage Resistance	 0.1℃/0.1℉ 15 bits + sign bit -	 0.1℃/0.1℉ - 15 bits + sign bit
Measurement principle	Sigma-Delta	
Module update frequency (all channels)	4 channels support 8Hz, 4Hz, 2Hz, 1Hz configuration, default: 2Hz for all channels	
	8 channels support 4Hz, 2Hz, 1Hz, 0.5Hz configuration, default: 1Hz for all channels	
Wire length to sensor	maximum 100m	
wire loop resistance	100Ω	20Ω, Cu type 2.7Ω
Noise suppression	85dB@50Hz/60Hz/400Hz	
Data word format	Voltage: -27648~+27648	resistance:-27648~+27648
Input resistance	＞10MΩ	＞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	
Diagnostic Disconnection No module power	 32767(positive),-32768(negative) 32766	

supply		
PID control function	None	-

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	NR
:	:								
17680	4510								
:	:								
13720	3598								
:	:								
13000	32C8								
:	:								
12000	2EE0								
:	:								
10000	2710								
:	:								
4000	0FA0								
:	:								
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								
:	:								
-2000	F830								
:	:								
-2100	F7CC								
:	:								
-2550	F60A								
:	:								
-2700	F574								
:	:								
-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	

Thermal resistance characteristics

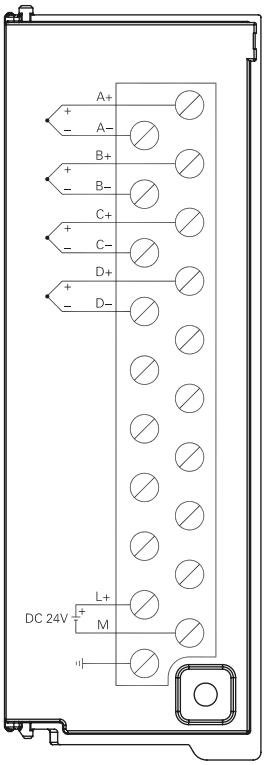
Temperature Range (°C) and Accuracy for RTD Types

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	

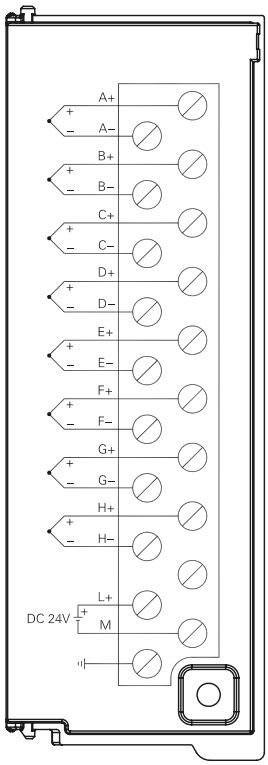
*OF=Overflow, OR=Out of Range, NR=Rated Range, UR=Under Range, UF=Underflow

Thermocouple Wiring Specifications

◆ AIT-040S1

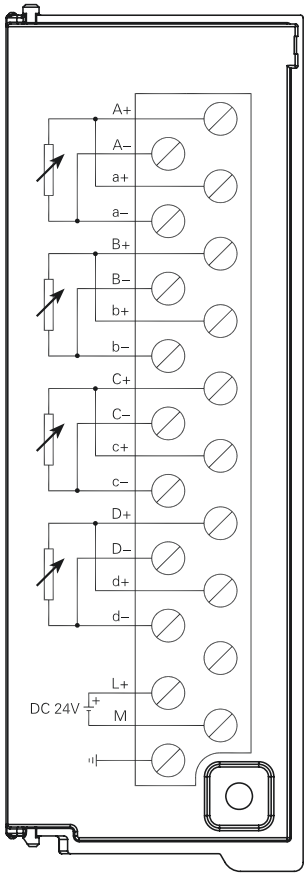


◆ AIT-080S1

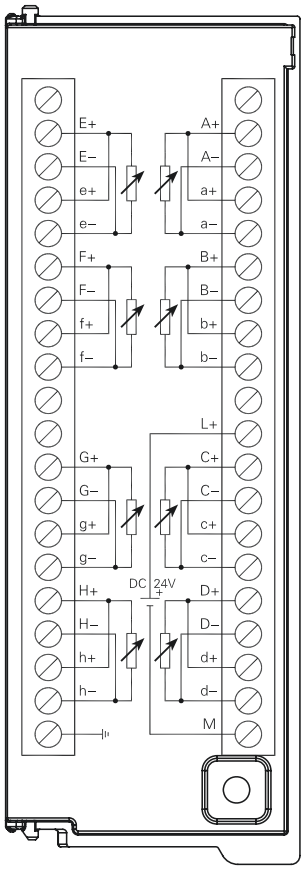


hermal resistance wiring specifications

◆ AIR-040S1



◆ AIR-080S1



C.5 High-speed counting module

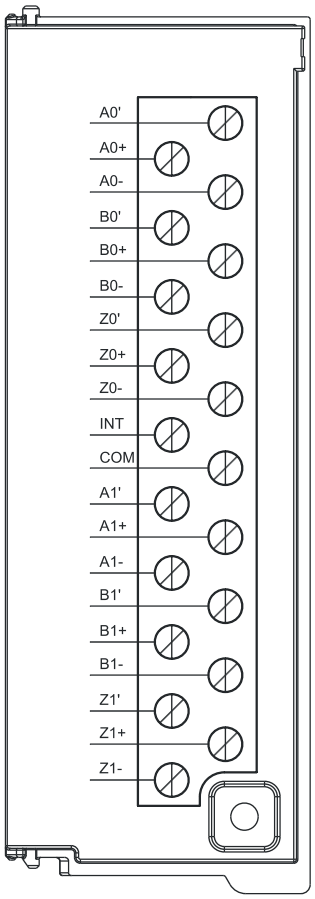
Table C-20 Basic properties of high-speed counting module

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

Table C-21 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

Wiring



C.6 Pulse Output Module

Table C-22 Basic properties of the pulse output module

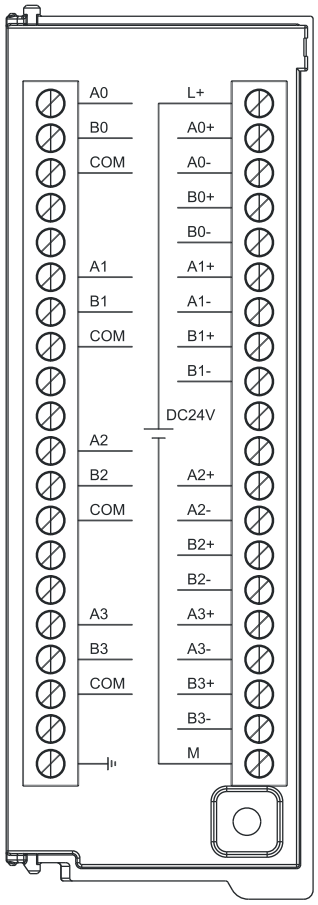
Name	Specification Description	Order number
Pulse output module	4 differential/single-ended signal outputs	CTH3 HSP-040S1

Table C-23 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	

Wiring



C.7 Intermediate expansion module

Table C-24 Basic properties of the Intermediate expansion module

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

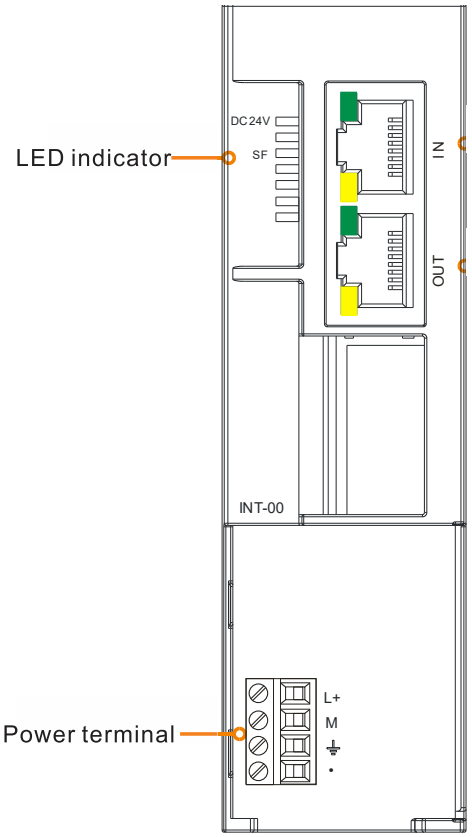
Table C-25 General characteristics of INT-00

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

Table C-26 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

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).

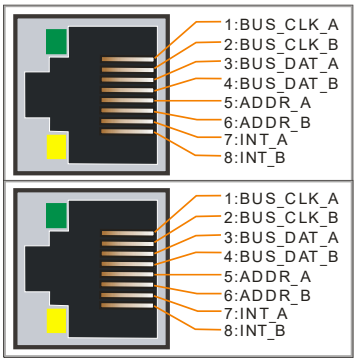
Note: The first relay module is connected to the CPU through the bus.

Interface Definition

Table C-27 Definition of power interface of INT-00

4-position removable terminal	Signal	Definition
L+ M ⏚ •	L+	24V+
	M	24V-
	⏚	the earth
	--	--

Table C-28 Definition of Dual RJ45 Interfaces of INT-00



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

C.8 EtherCAT Slave Module

Table C-29 Basic properties of the 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

Table C-30 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 type module, the communication between the slave and the master is disconnected, and the slave will not switch to safe operation.

Table C-31 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

Table C-32 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 adopts 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 a fully shielded Category 5 cable or a fully shielded super cable. Category 5 wire, 24AWG.

Table C-33 Specifications of Communication Port Cables

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

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

Interface diagram

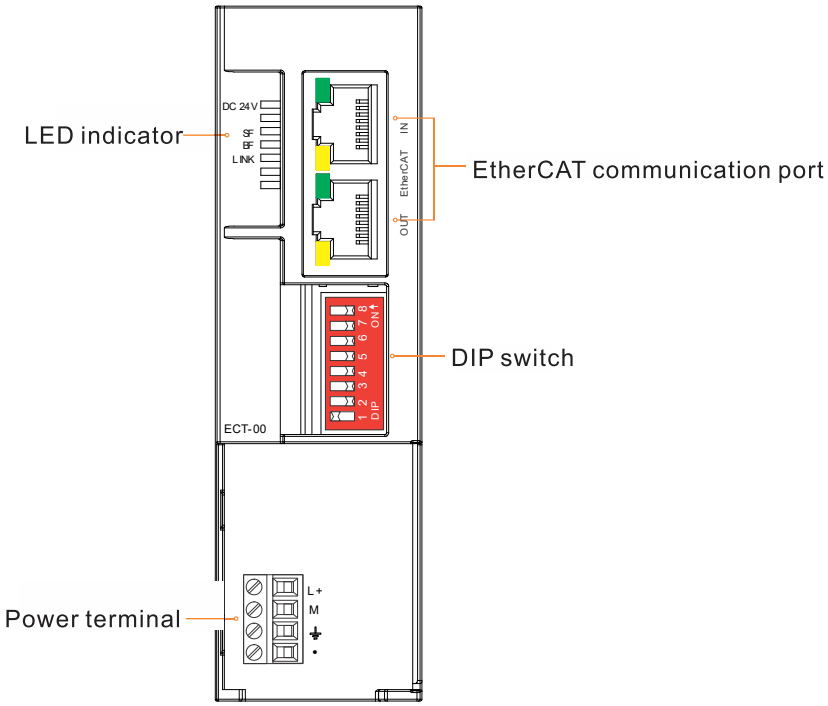


Table C-34 Power Interface Definition

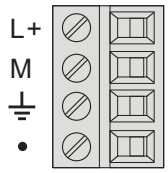
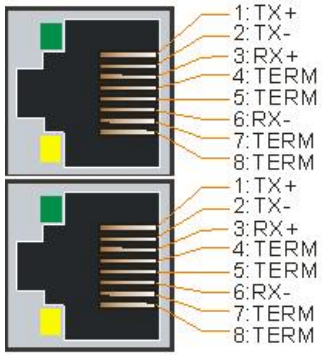
4-position removable terminal	NO	Signal	Signal definition
	1	L+	24V+
	2	M	24V-
	3	⏚	ground
	4	--	--

Table C-35 Definition 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 housing	PE	Chassis ground

D IO module channel configuration control word

D.1 Digital Input Module Channel Configuration

The configuration parameter format of each group (8 channels) is:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserve	Channel 4-7 filter time			reserve	Channel 0-3 filter time		

Filter time:

0: 0.20ms

1: 0.40ms

2: 0.80ms

3: 1.60ms

5: 3.20ms

6: 6.40ms(Defaults)

7: 12.8ms

Channel Control Byte Default Value: 16#66

D.2 Digital output module channel configuration

The configuration parameter format of each group (8 channels):

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserve	reserve	reserve	reserve	reserve	reserve	reserve	reserve

Channel Control Byte Default Value: 16#00

D.3 AI module channel configuration

Analog input a group of parameter configuration format:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
sampling period			Input type and range				
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8

reserve	reserve	reserve	reserve	reserve	reserve	reserve

Input Type and Range:

Unipolar and bipolar are distinguished by bit4, bit4 is 1 for bipolar; 0 for unipolar, the specific code is as follows:

module type	input type	Input range	Range code (BIT4~0)
AI module	Voltage	0~5V	00000
		0~10V(Defaults)	00001
		±2.5V	10000
		±5V	10001
	current	0~20mA(Defaults)	00010
		4~20mA	00011

Sampling period:

module type	Update frequency (sampling period)	Sampling period coding (bit7~5)
AI module 4 channels	200 Hz	000
	100 Hz	001
	50 Hz(Defaults)	010
	20 Hz	011
	10 Hz	100
AI module 8 channels	50 Hz	000
	20 Hz	001
	10 Hz(Defaults)	010
	5 Hz	011
	2 Hz	100

Channel default value: 16#0041 (voltage 0～10V, 4 channels, 50Hz)

D.4 TC module channel configuration

The configuration format of a set of parameters of the temperature module:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sampling period			Input Type and Range				
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Temperature Module Group Off Bit	Configuration enable bit, must be 1	Temperature Module Cold Junction Compensation	Positive and negative calibration of temperature module	Temperature module disconnection detection	Temperature Module Temperature Units	Temperature module wiring method	

Note: Group Shutdown Bit 0 - Enable

Input Type and Range:

Module type	Input type	Input range	Range code (bit4~0)
TC	TC	S	00000
		T	00001
		R	00010
		E	00011

		N	00100
		K(Defaults)	00101
		J	00110
	Voltage	±80mV	10000

Sampling period:

Module type	Update frequency (sampling period)	Sampling period coding (bit7~5)
TC 4 Channel	8 Hz	000
	4 Hz	001
	2 Hz(Defaults)	010
	1 Hz	011
TC 8 Channel	4 Hz	000
	2 Hz	001
	1 Hz(Defaults)	010
	0.5 Hz	011

Wiring	0: three-wire (Defaults) 1: Two-wire 2: Four-wire
temperature unit	0: degrees Celsius(Defaults) 1: degrees Fahenheit
disconnection detection	0: To detect disconnection (Defaults) 1: Do not detect disconnection, The temperature module needs to configure this parameter, and the AI module only needs to configure this parameter for the 4~20mA range.
Direction	0: Upscale(Defaults) 1: Downscale This parameter is required for the temperature module. For the AI module, this parameter is required only for the 4 to 20mA range.
Cold Junction Compensation	0: cold junction compensation (Defaults) 1: No cold junction compensation
Group enable	0: Enable (Defaults) 1: Disable

Channel default value: 16#4045 (K, 4 Channel, 2Hz, degrees Celsius, disconnection detection, positive calibration, cold junction compensation, enable)

D.5 RTD Module Channel Configuration

The temperature module inputs a set of parameter configuration format:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sampling period			Input Type and Range				
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Temperature Module Group Off Bit	Configuration enable bit, must be 1	Temperature Module Cold Junction Compensa	Positive and negative calibration of	Temperature module disconnection detection	Temperature Module Temperature Units	Temperature module wiring method	

		tion	temperatur e module			

Note: Group Shutdown Bit 0 - Enable

Input Type and Range:

Module type	input type	Input range	Range code (bit4~0)
RTD	RTD	Pt-100Ω(α=3850ppm) (Defaults)	00000
		Pt-200Ω(α=3850ppm)	00001
		Pt-500Ω(α=3850ppm)	00010
		Pt-1000Ω(α=3850ppm)	00011
		Pt-100Ω(α=3920ppm)	00100
		Pt-200Ω(α=3920ppm)	00101
		Pt-500Ω(α=3920ppm)	00110
		Pt-1000Ω(α=3920ppm)	00111
		Pt-100Ω(α=3850.55ppm)	01000
		Pt-200Ω(α=3850.55ppm)	01001
		Pt-500Ω(α=3850.55ppm)	01010
		Pt-1000Ω(α=3850.55ppm)	01011
		Pt-100Ω(α=3916ppm)	01100
		Pt-200Ω(α=3916ppm)	01101
		Pt-500Ω(α=3916ppm)	01110
		Pt-1000Ω(α=3916ppm)	01111
		Pt-100Ω(α=3902ppm)	10000
		Pt-200Ω(α=3902ppm)	10001
		Pt-500Ω(α=3902ppm)	10010
		Pt-1000Ω(α=3902ppm)	10011
		Pt-10000Ω(α=3850ppm)	10100
		Cu-9.035Ω(α=4720ppm)	10101
		Ni-10Ω(α=6720ppm)	10110
		Ni-120Ω(α=6720ppm)	10111
		Ni-1000Ω(α=6720ppm)	11000
		Ni-10Ω(α=6178ppm)	11001
		Ni-120Ω(α=6178ppm)	11010
		Ni-1000Ω(α=6178ppm)	11011
	Resistance	R-150Ω	11100
		R-300Ω	11101
		R-600ΩFS	11110

Sampling period:

module type	Update frequency (sampling period)	Sampling period coding (bit7~5)
RTD 4 Channel	8 Hz	000
	4 Hz	001
	2 Hz(Defaults)	010
	1 Hz	011

RTD 8 Channel	4 Hz	000
	2 Hz	001
	1 Hz(Defaults)	010
	0.5 Hz	011

Wiring	0: three-wire (Defaults) 1: Two-wire 2: Four-wire
temperature unit	0: degrees Celsius(Defaults) 1: degrees Fahenheit
disconnection detection	0: To detect disconnection (Defaults) 1: Do not detect disconnection, The temperature module needs to configure this parameter, and the AI module only needs to configure this parameter for the 4~20mA range.
Direction	0: Upscale(Defaults) 1: Downscale This parameter is required for the temperature module. For the AI module, this parameter is required only for the 4 to 20mA range.
Cold Junction Compensation	0: cold junction compensation (Defaults) 1: No cold junction compensation
Group enable	0: Enable (Defaults) 1: Disable

Channel default value: 16#4040 (Pt-100Ω(α=3850ppm), 4 Channel, 2Hz, three-wire, degrees Celsius, disconnection detection, positive calibration, cold junction compensation, enable)

D.6 Analog Output Module Channel Configuration

Analog output type configuration format:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserve		reserve	Voltage and current	Range (including polarity)			
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
reserve							

type:

0: Voltage (Defaults)

1: Current

Range:

0: ±10V(Voltage)

1: 0~20mA(current)

2: 4~20mA(current)

Defaults: ±10V(Voltage)/0~20mA(current)

channel default: 16#0000

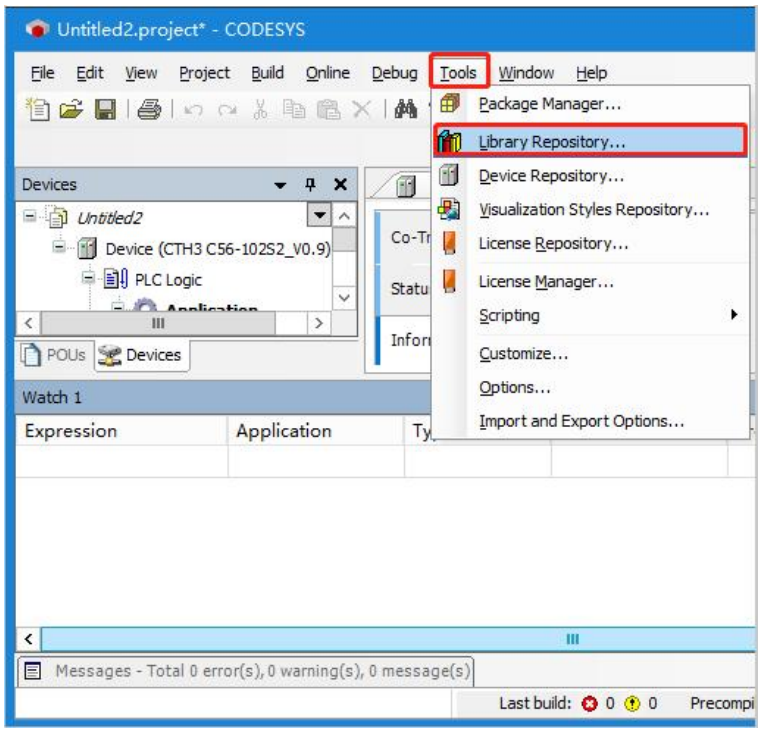
E Introduction to the use of CT_MODBUS library

E.1 Install CTModbus library files

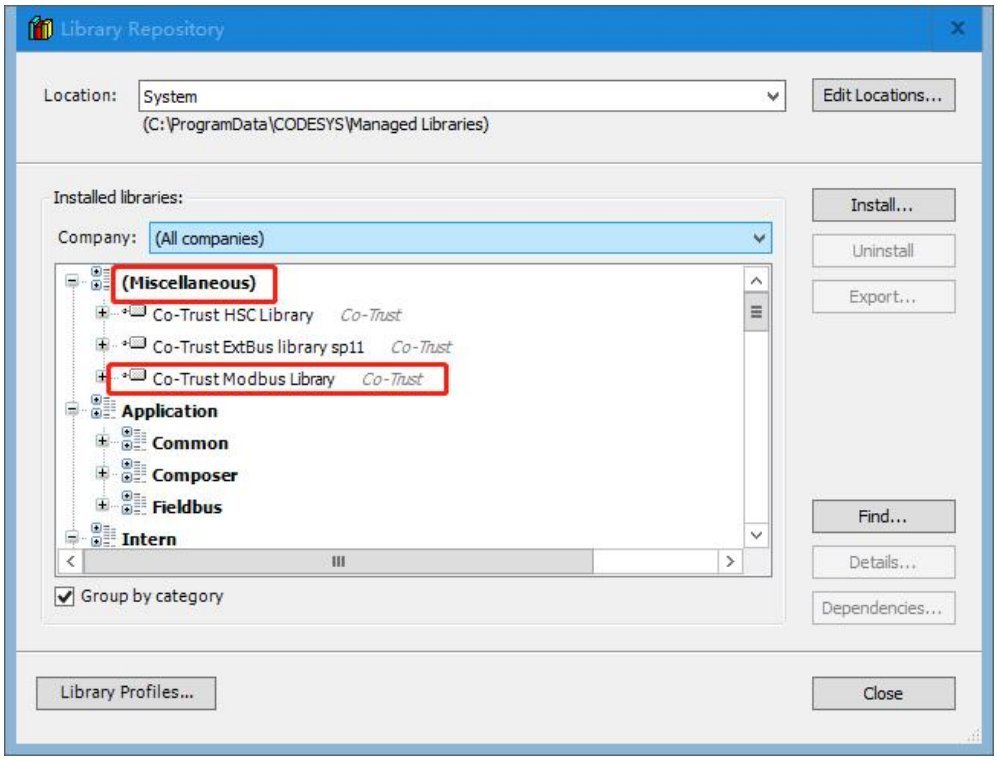
When selecting Modbus communication, please install the CT_MODBUS library file first (please download the library file from COTRUST's website: http://www.co-trust.com), and then execute "Library" to use the library command.

1) Install the library

Start the CODESYS software and select the menu item "Tools" → "Libraries":



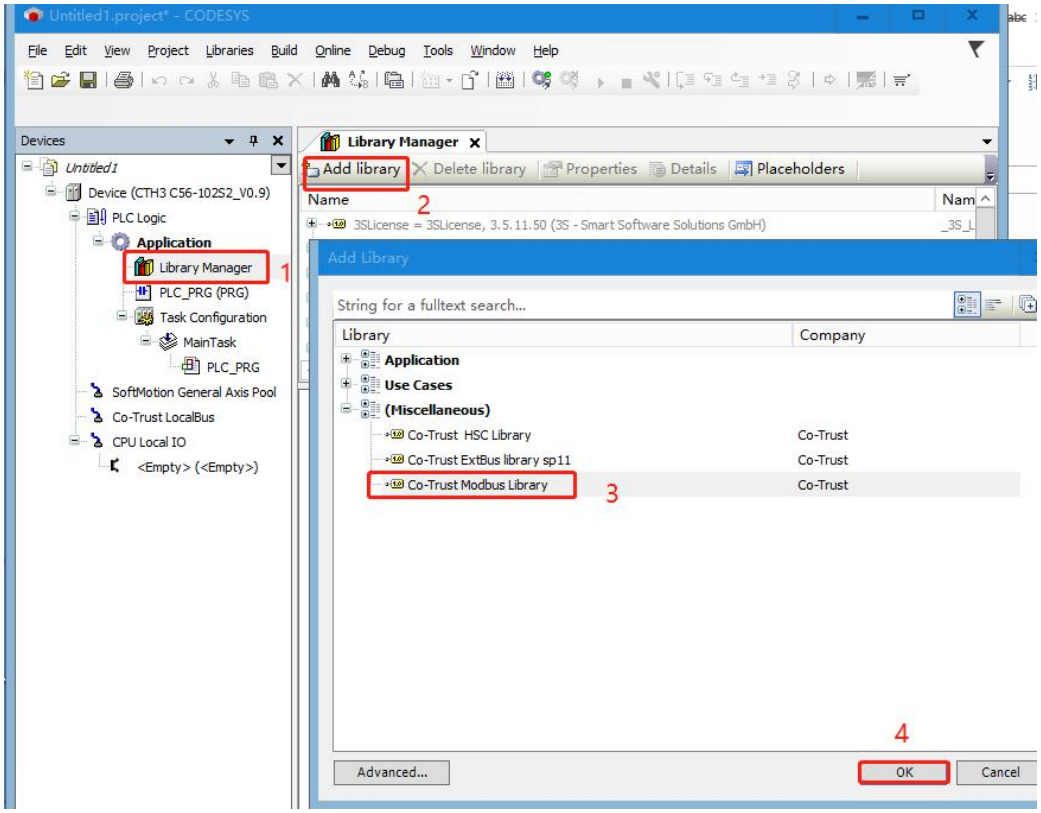
Select "Ct_Modbus.library" and click the "Open" button. That means the CT_MODBUS library has been installed successfully.



2) Add library

After the CTMODBUS library file is successfully installed, if you need to use the library in the current project, you need to execute "Add Library". The specific operation steps are as follows:

Expand "Device"→"PLC"→"Application" in the device view, double-click to open the "Library Manager", and then click the "Add Library"，then choose Co-Trust Modbus library, Modbus library is added to the project.



E.2 Modbus_RTU library file

1) Modbus_RTU master MBUS_CTRL instructions

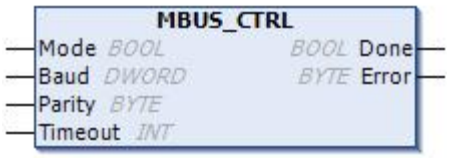


Table E-1 Parameter description of MBUS_CTRL instruction

Parameter	Description	Data type	Range
Mode	Enable switch (0: OFF 1: ON)	BOOL	
Baud	Baud rate	DWORD	1200, 2400, 4800, 9600 19200, 38400, 57600, 115200
Parity	Set check 0: no checksum 1: odd parity 2: Even parity	BYTE	
Timeout	Timeout time (the slave does not reply data within the set time, it is considered that the read and write timeout)	INT	
Done	done bit 0: instruction is being executed 1: Instruction execution completed	BOOL	
Error	error code	BYTE	

2) Master MBUS_MSG Instructions

Table E-2 Parameter description of MBUS_MSG instruction

Parameter	Description	Data type	Initialization
First	Activation bit (0: inactive 1: active), if it is not activated once, the instruction is executed once.	BOOL	0
Slave	Slave number	BYTE	0
RW	R/W read and write 0: read 1: write	BYTE	0
Addr	Read and write register addresses (eg 40001, 30001)	DWORD	0
Count	Number of read and write bytes (value range 0-120	INT	0

	words)		
DataPtr	Read and write pointer (the location where the read data is stored or the data to be written to the register)	POINTER TO BYTE	0
Done	done bit 0: instruction is being executed 1: Instruction execution completed	BOOL	0
Error	error code	BYTE	0

3) Instructions for use of slave MBUS_INIT

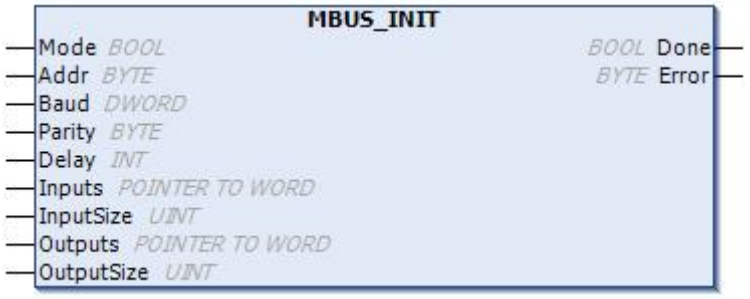


Table E-3 Parameter description of MBUS_INIT command

Parameter	Description	Data type	Initialization
Mode	Enable switch (0: off 1: on)	BOOL	1
Addr	Station address	BYTE	2
Baud	Baud rate (1200,2400,4800,9600,19200,38400,57600,115200)	DWORD	9600
Parity	Parity 0: no checksum 1: odd parity 2: Even parity	BYTE	0
Delay	Reply delay time (usually set to 0, if other communication parameters are correct but still cannot communicate, you can increase the value appropriately)	INT	0
Inputs	Pointer to Input Register	Pointer to Word	0
InputSize	Size of Input Register	Uint	0
Outputs	Pointer to the Holding Register	Pointer to Word	0
OutputSize	Size of Holding Register	Uint	0
Done	done bit (0: instruction is being executed 1: instruction execution is complete)	BOOL	0
Error	error code	BYTE	0

4) Instructions for use of MBUS_SLAVE

Table E-4 Parameter description of MBUS_SLAVE command

Parameter	Description	Data type	Initialization
Done	Completion bit (0: instruction is being executed 1: instruction execution is complete)	BOOL	
Error	error code	BYTE	

E.3 Modbus_TCP library file

1) Modbus_Tcp master MBUS_TCP_REQ instructions

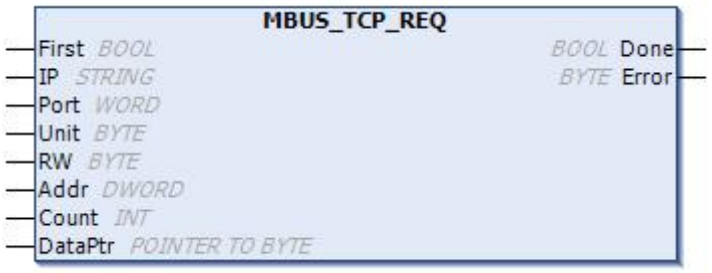


Table E-5 Parameter description of MBUS_TCO_REQ instruction

Parameter	Description	Data type	Initialization
First	Activation bit (0: inactive 1: active), activated once, the instruction is executed once.	BOOL	0
IP	Slave ip address, for example '192.168.0.1'	STRING	192.168.0.1
Port	Slave port number, e.g. 502	WORD	502
Unit	Slave unit number, e.g. 0	BYTE	0
RW	read and write flags 0: Read 1: Write	BYTE	0
Addr	Read and write register addresses (such as 40001, 30001)	DWORD	0
Count	Number of read and write bytes (value range 0-120 words)	INT	0
DataPtr	Read and write pointer (the location where the read data is stored or the data to be written to the register)	POINTER TO BYTE	0
Done	done bit 0: instruction is being executed 1: Instruction execution completed	BOOL	0
Error	error code	BYTE	0

2) MBUS_TCP_SLAVE instruction

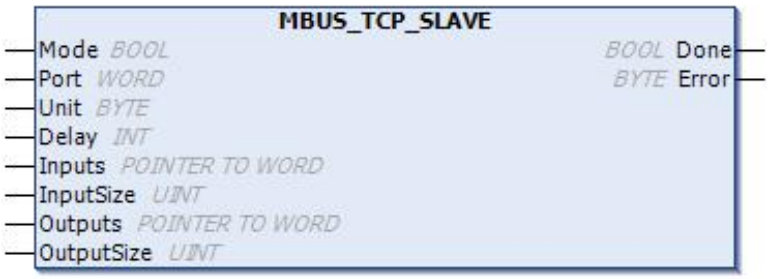


Table E-10 Parameter description of MBUS_TCP_SLAVE command

parameter	Description	Data type	Initialization
Mode	enable switch 0: off 1: on	BOOL	1
Port	Slave port, eg: 502	WORD	502
Unit	Slave unit number, eg: 0	BYTE	0
Delay	Reply delay time (usually set to 0, if other communication parameters are correct but still cannot communicate, you can increase the value appropriately)	INT	0
Inputs	pointer to Input Register	Pointer to Word	0
InputSize	Size of Input Register	uint	0
Outputs	pointer to the Holding Register	Pointer to Word	0
OutputSize	Holding Register Size	Uint	0
Done	done bit (0: instruction is being executed 1: instruction execution is complete)	BOOL	0
Error	error code	BYTE	0

F Detailed description of related operations in CODESYS

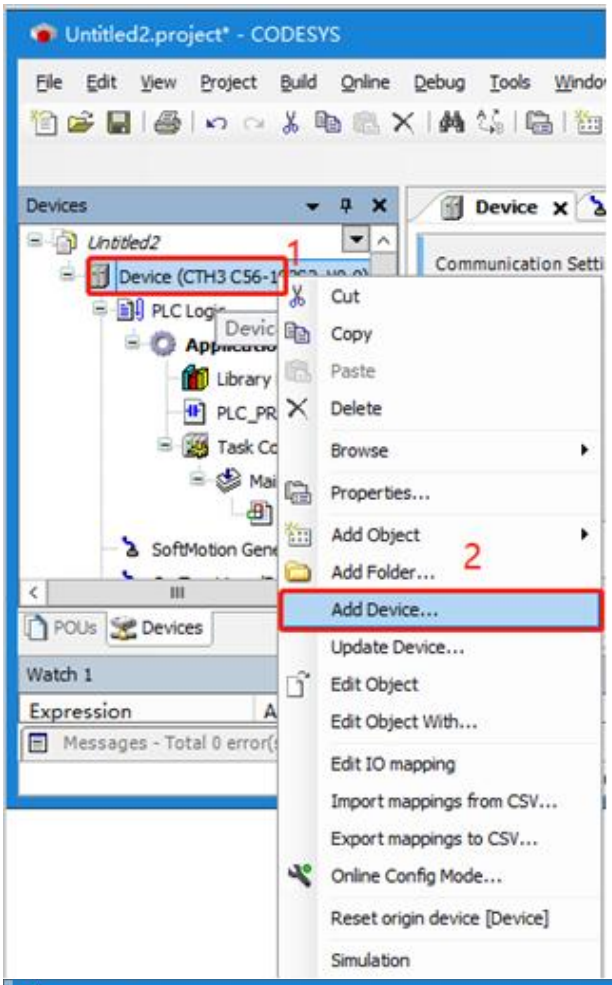
F.1 Add device

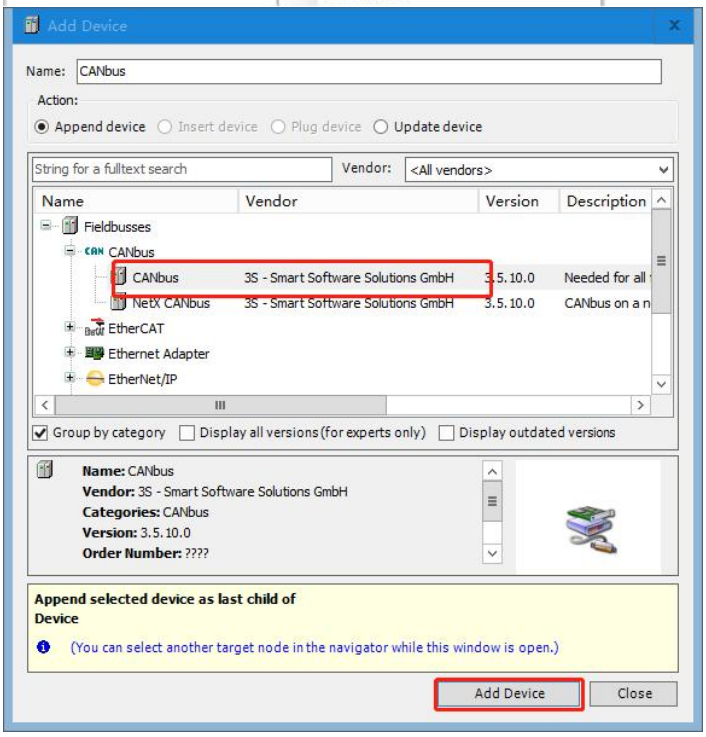
C5X-10 can communicate with CANopen, EtherCAT, Modbus_TCP/RTU on CODESYS. Modbus communication can be realized by programming through the Modbus instruction library (refer to E Introduction to the use of CT_MODBUS library). For CANopen and EtherCAT communication, it is necessary to add a communication device in the CODESYS device view to perform specified communication. The steps for adding CANopen and EtherCAT communication devices can be referred to the following descriptions.

1. Add CANopen communication device in CODESYS

1) Add CANbus

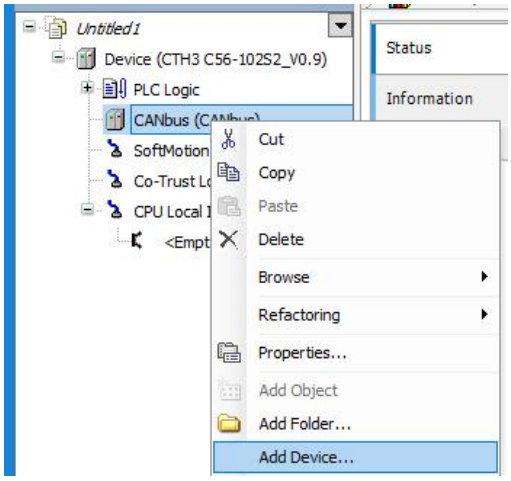
Right-click "Device(CTH3 C56-10S2_V0.9)" and select "Add Device", you can choose to add CANbus in the pop-up dialog box: Supplier select "<all suppliers>", Fieldbus select " CANbus" as the CAN bus.

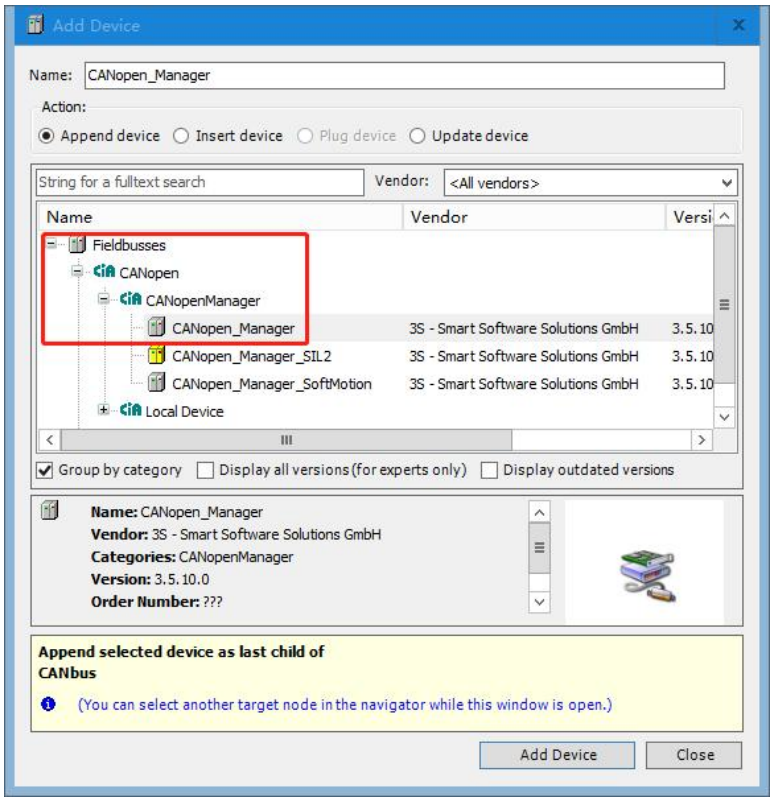




2) Add CAN master

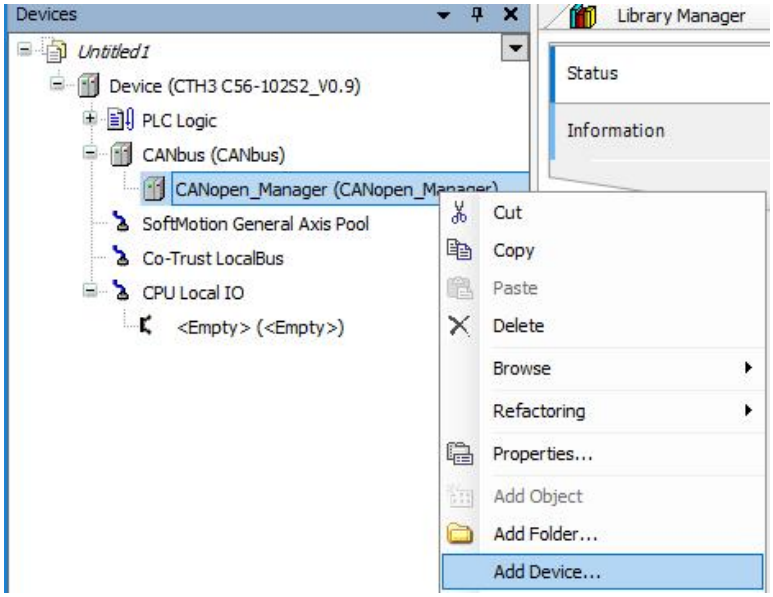
Select the "CANBUS" that has been successfully added in the device view and right-click to select "Add Device" to add a CAN master in the "Add Device" dialog: Fieldbus→CANopen→CANopen Manager→CANopen_Manager.

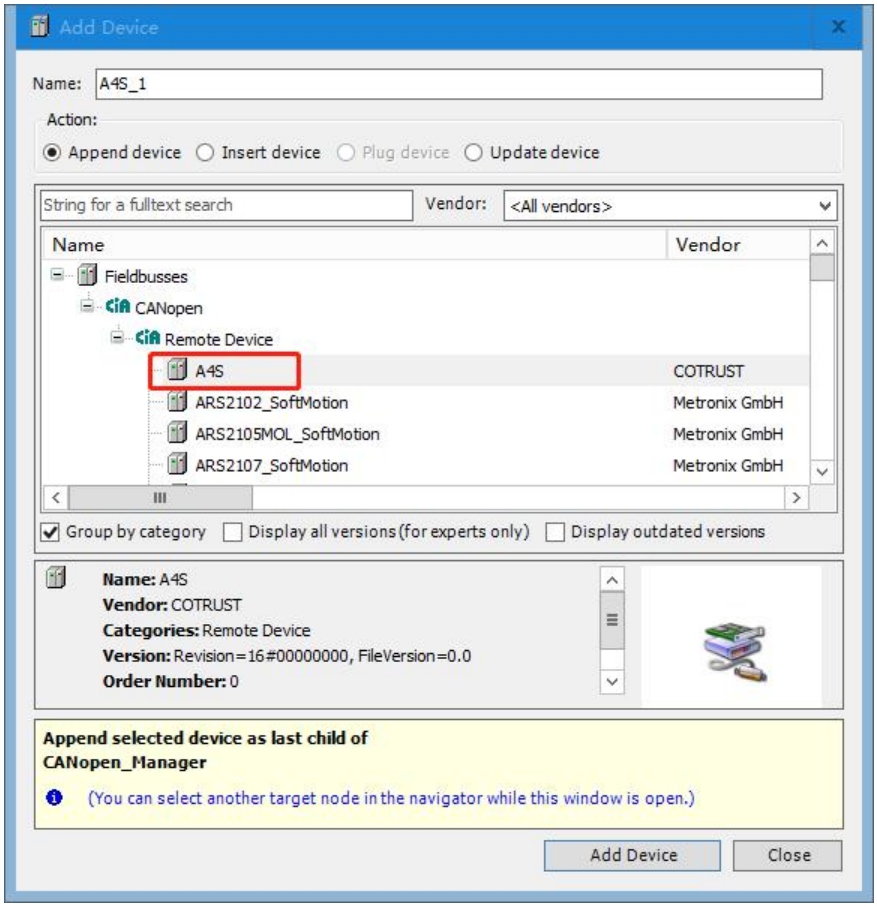


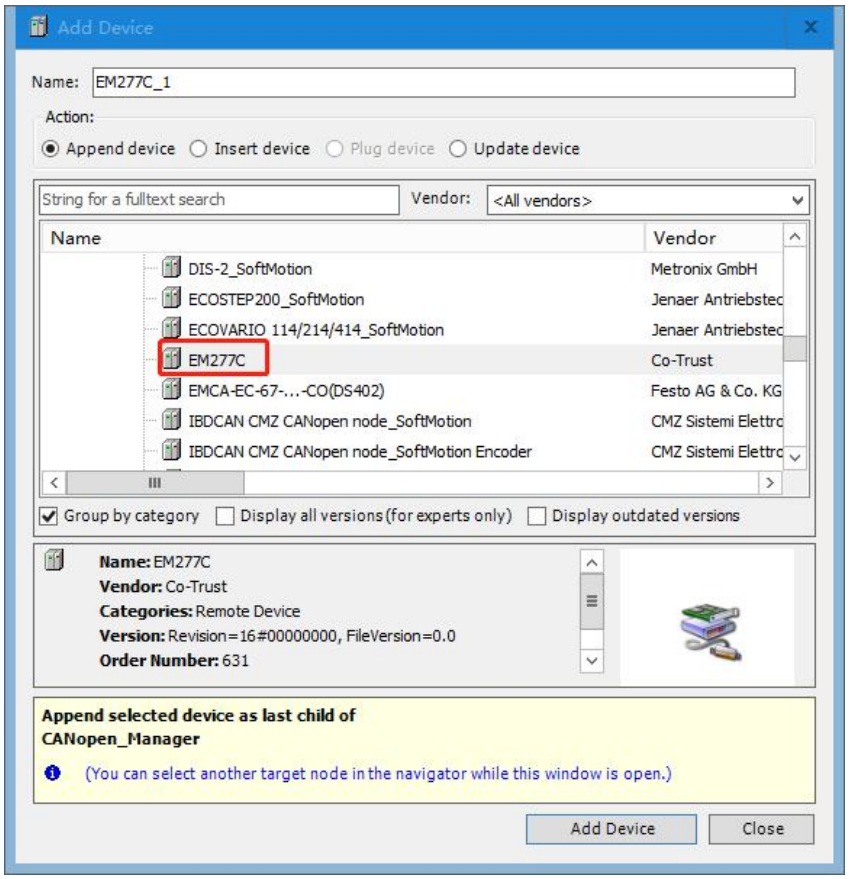


3) Add or scan CAN slaves

Select the added "CANopen_Manager" and right-click to select "Add Device" to add a CAN slave in the "Add Device" dialog: Fieldbus→CANopen→Remote Device→EM277C/A4S, or right-click "CANopen_Manager" and select " "Scan devices", the slave devices connected to the CANopen master can be displayed under the CAN master.

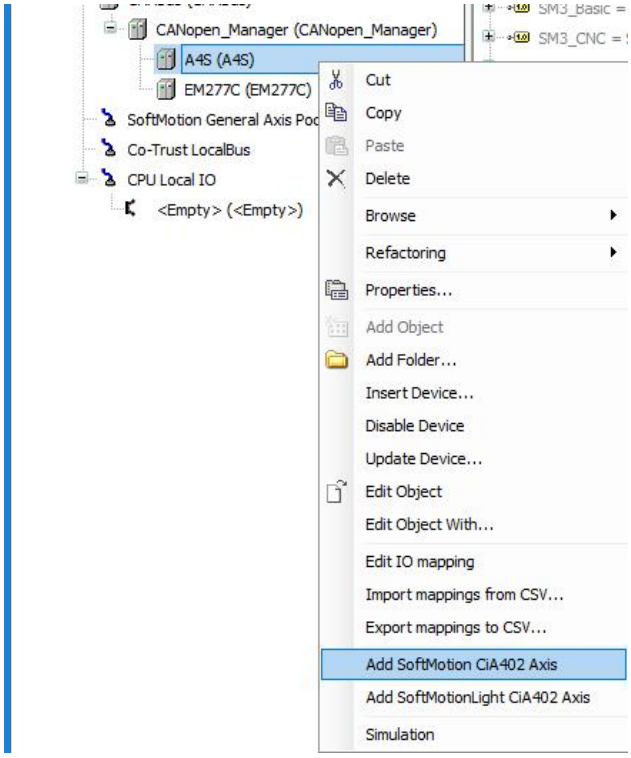

Devices
Untitled1
Device (CTH3 C56-102S2_V0.9)
PLC Logic
CANbus (CANbus)
CANopen_Manager (CANopen_Manager)
SoftMotion General Axis Pool
Co-Trust LocalBus
CPU Local IO
<Empty> (<Empty>)
Library Manager
Status
Information
Cut
Copy
Paste
Delete
Browse
Refactoring
Properties...
Add Object
Add Folder...
Add Device...

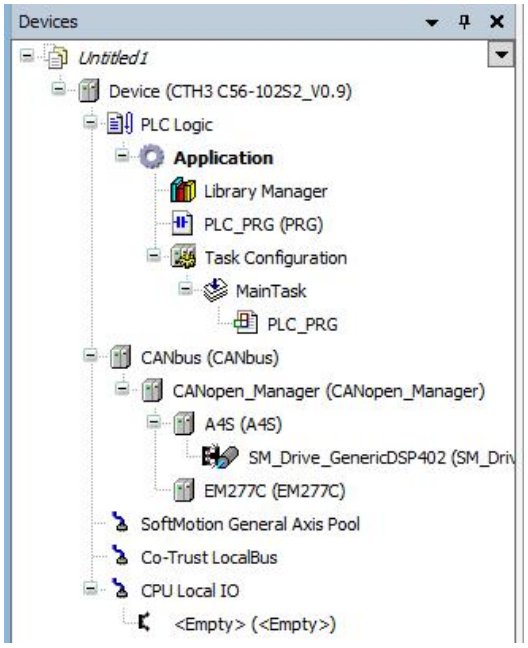

Add Device
Name: A4S_1
Action:
Append device
Insert device
Plug device
Update device
String for a fulltext search
Vendor:
<All vendors>
Name
Vendor
Fieldbusses
CANopen
Remote Device
A4S
COTRUST
ARS2102_SoftMotion
Metronix GmbH
ARS2105MOL_SoftMotion
Metronix GmbH
ARS2107_SoftMotion
Metronix GmbH
Group by category
Display all versions (for experts only)
Display outdated versions
Name: A4S
Vendor: COTRUST
Categories: Remote Device
Version: Revision=16#00000000, FileVersion=0.0
Order Number: 0
Append selected device as last child of
CANopen_Manager
(You can select another target node in the navigator while this window is open.)
Add Device
Close



4) Add driver for CANopen slave

Select the CANopen slave (A4S) in the device view and right-click, then select "Add softmotion-CiA402-axis" in the pop-up menu, that is, the driver is added successfully.

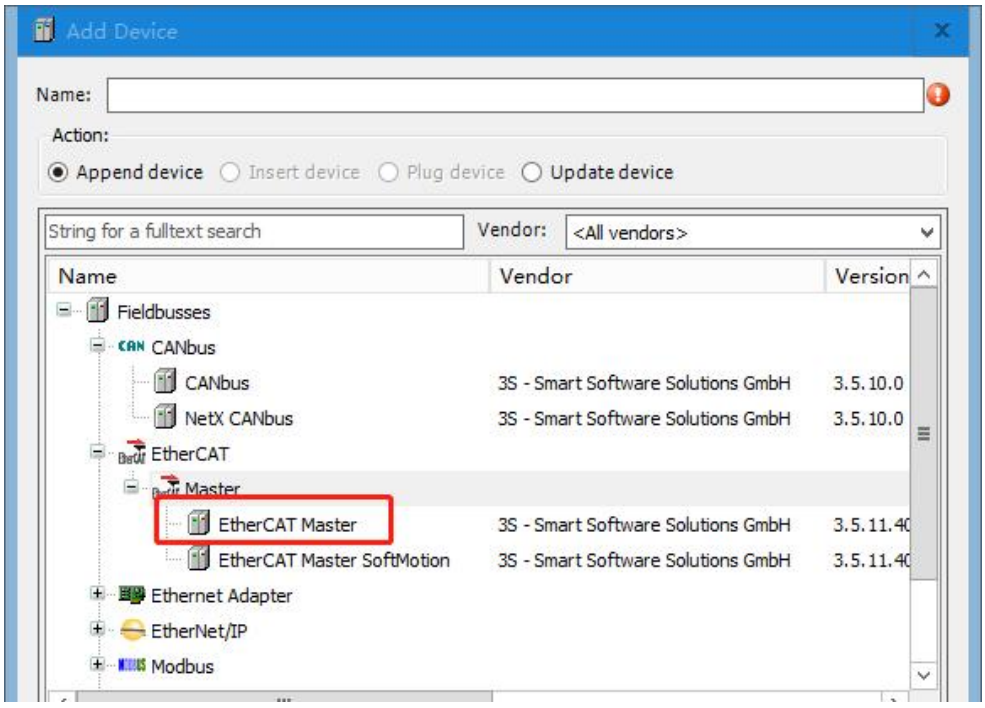




2. Add EtherCAT communication device in CODESYS

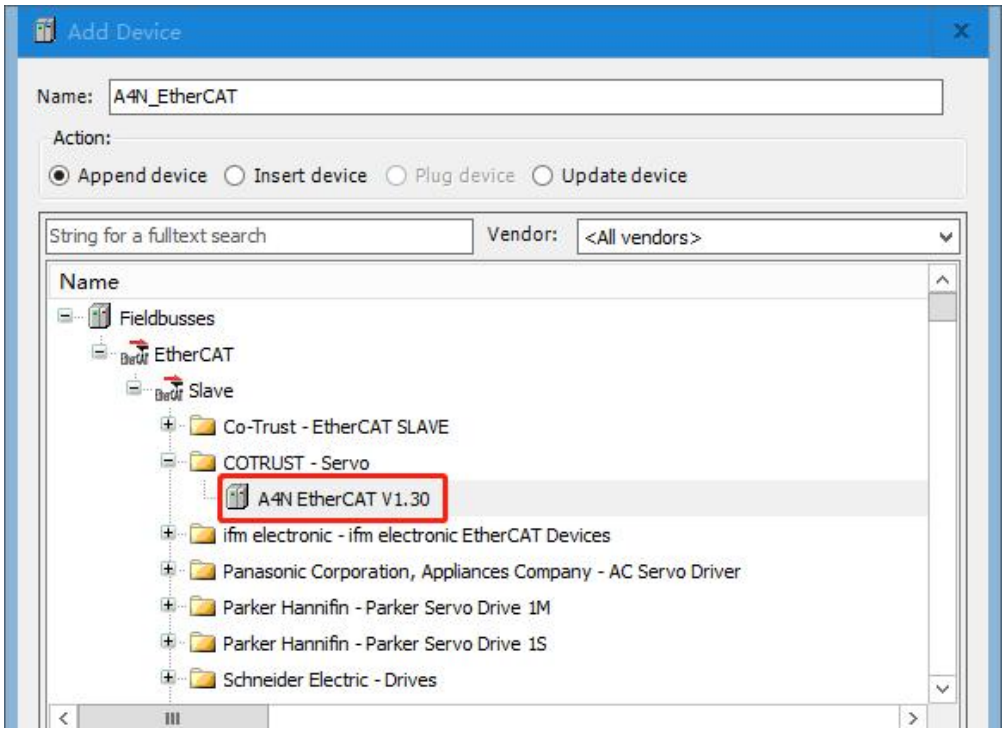
1) Add EtherCAT master

Right-click "Device(CTH3 C56-10S2_V0.9)" in the device view and select "Add Device" to add an EtherCAT master: select "<all suppliers>" for suppliers, and select "EtherCAT" → "Master" → "EtherCAT Master".



2) Add or scan EtherCAT slaves

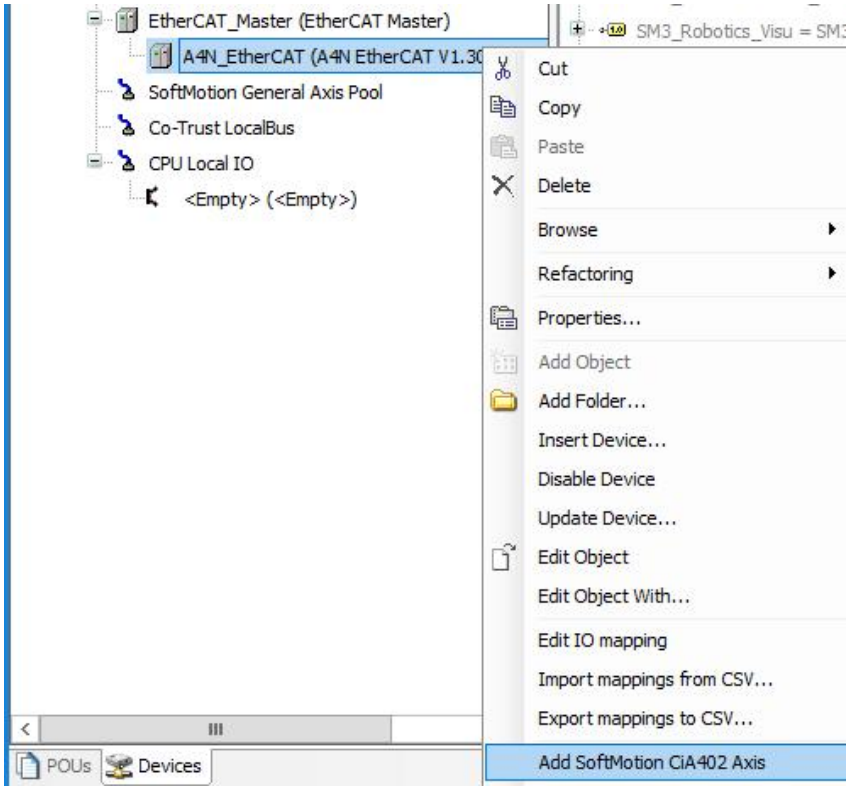
Select the added EtherCAT master and right-click to select "Add Device", then select to add EtherCAT slave: ""Ethercat" → "Slave" → "COTRUST-Servo"→“A4N EtherCAT V1.30”; or right-click on the EtherCAT master (C56-10) and select “Scan Device” to display the EtherCAT slave (A4N) connected to the C56-10 on the EtherCAT master (C56-10) below.



Note: EtherCAT slave modules cannot be added by "Scanning Devices", and can only be added manually.

3) Add the driver of the EtherCAT slave

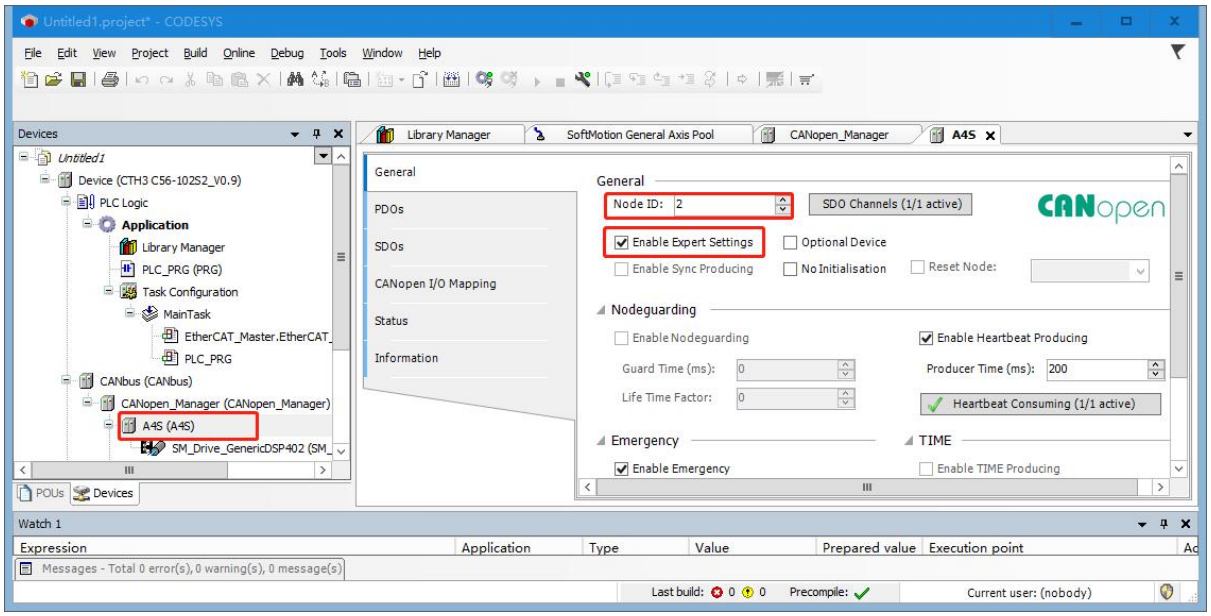
Select the EtherCAT slave (A4N) and right-click, select "Add softmotion-CiA402-axis".



F.2 Configuring CANopen Slave Devices

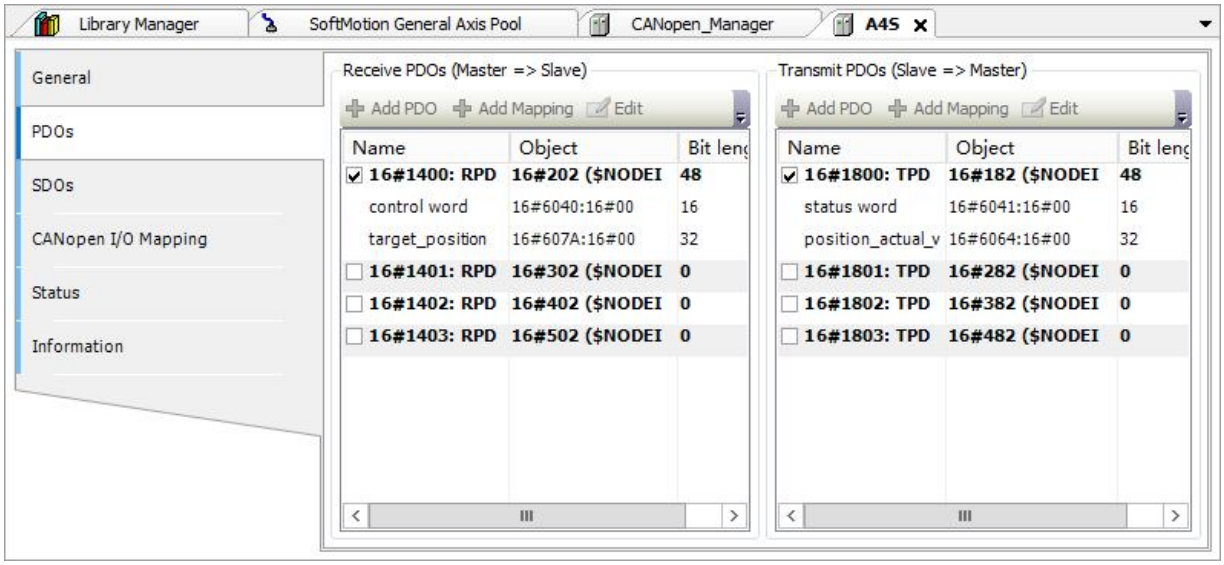
After the CAN slave device (E10/A4S/A4N/EM277C) is added to CODESYS, it needs to be configured. Double-click to open the CAN slave in the device view, and then you can configure it, refer to the following:

①"Overview" tab: Set the node ID, check "Enable Expert Settings".



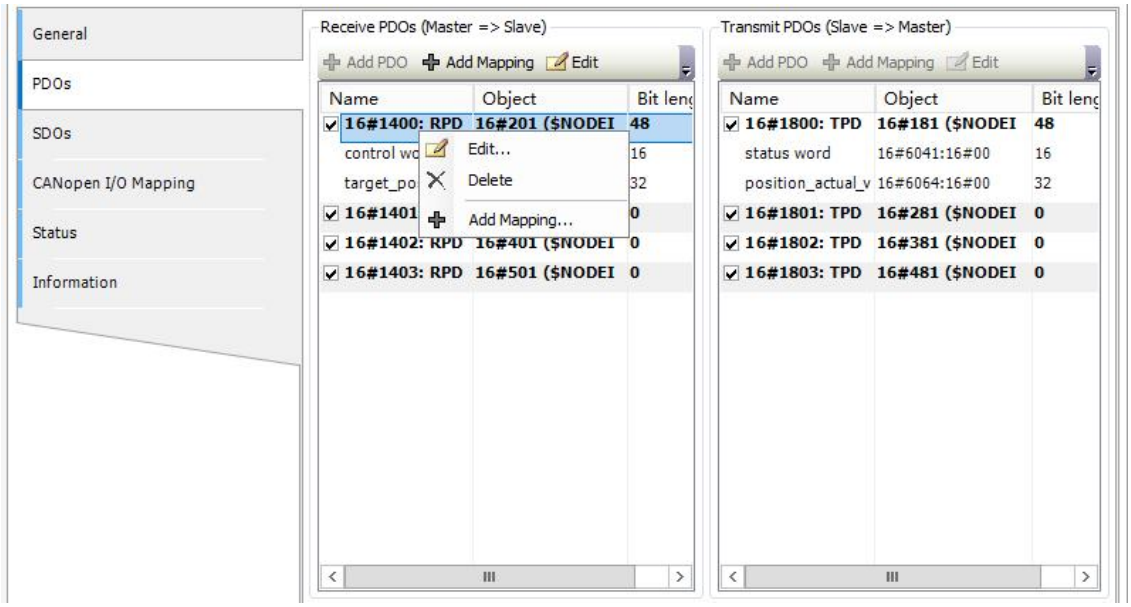
② "PDOs" tab

In the "Receive PDOs (Master => Slave)" and "Transmit PDOs (Slave => Master)" configuration, check the boxes that need to be displayed in "Receive PDOs (Master => Slave)" and "Transmit PDOs (Slave => Master)" Parameter group in the PDOs (Slave => Master)" tab.



③"Received PDOs" tab

In this tab, you can add parameter mapping for the selected parameter group, right-click to select the PDO parameter group that needs to be operated, and select "Add Mapping". The specific operation is shown in the following figure:



<Remarks> When using COTRUST servo as CANopen slave, please refer to the relevant servo instruction manual for the description of each parameter, for example: If using A4S servo as CANopen slave, you need to refer to *A4S Servo Instruction Manual*.

④ "Transfer PDOs" tab

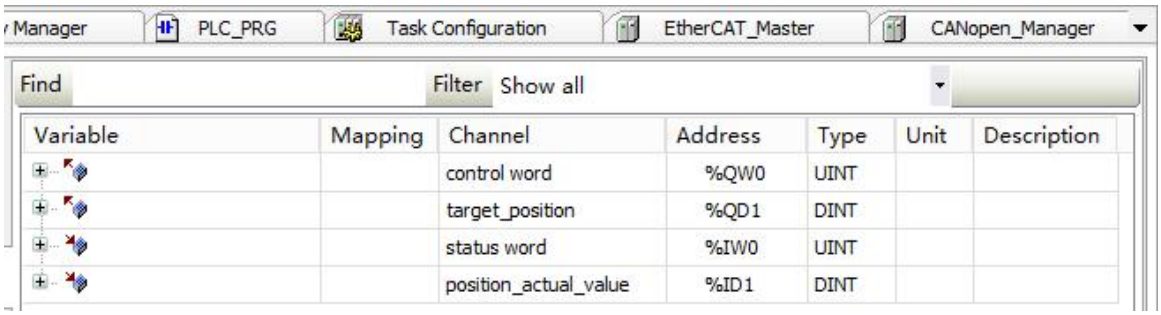
Same as the "Receive PDOs" operation, right-click to select the PDO parameter group that needs to be operated, and select "Add Mapping" to add send parameters to this group.

⑤ "Server data objects" tab

The required SDO can be configured in the service data object configuration dialog, it can define the order in which the objects need to be transferred, and it can also define the operations that will be performed during an incomplete transfer.

⑥ "CANopen I/O Mapping" tab

This dialog is used to display the CANopen I/O mapping, you can read/write the mapped data channels in this tab.



⑦ Status Tab

This dialog is used to display device status information (such as running, stopped, etc.) and device diagnostic information.

⑧ "Information" tab

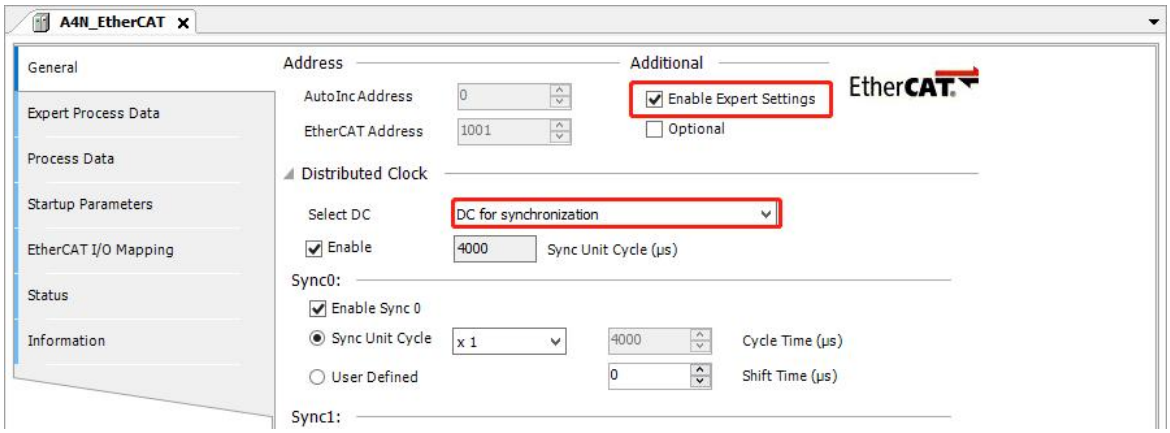
This dialog is used to display the information of the current device, such as name, supplier, type, version number, module serial number, description, etc.

F.3 Configuring EtherCAT Slave Devices

After the EtherCAT slave station device (E10 / A4S / A4n / ECT-00) is added to CODESYS, it needs to be configured.

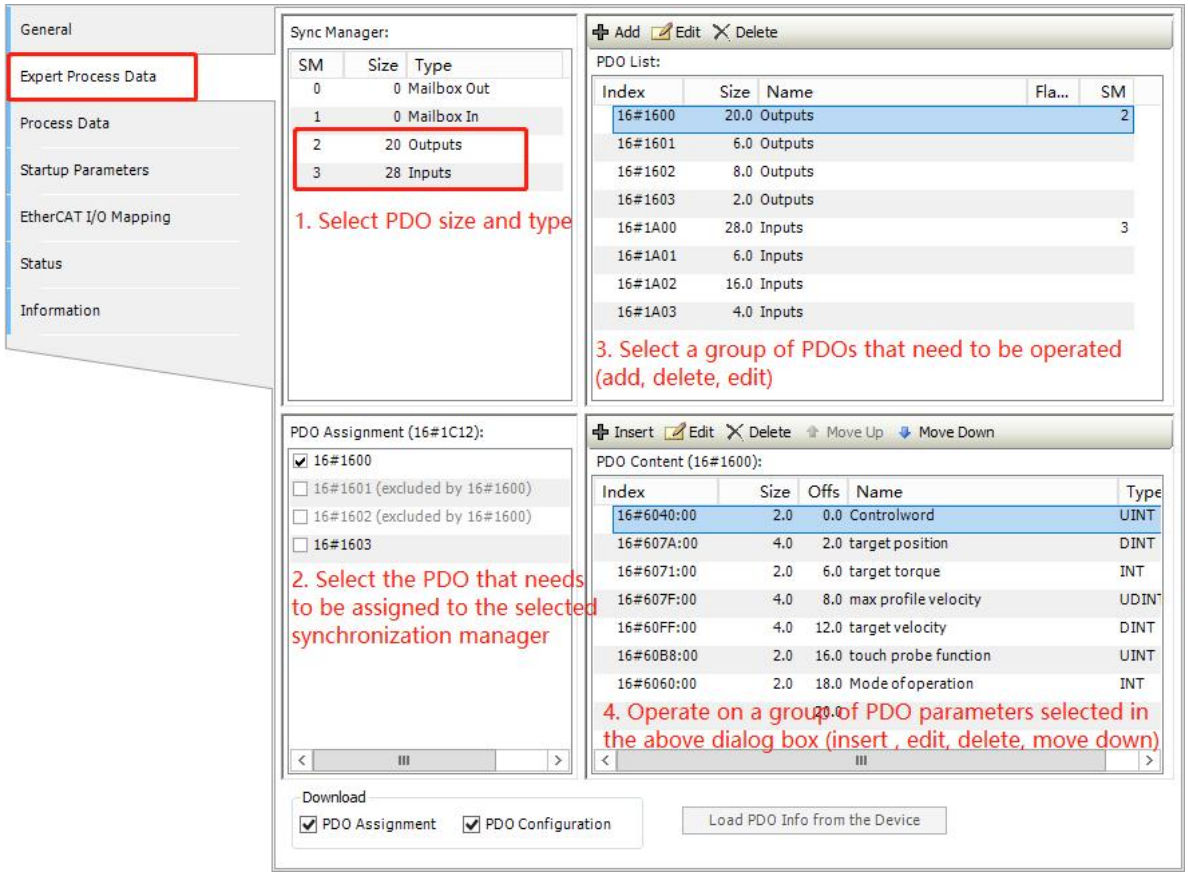
Configure E10 / A4N / A4S

① "Overview" tab: check "enable expert settings", and select "DC for synchronization" for distributed clock.



② Expert process data tab

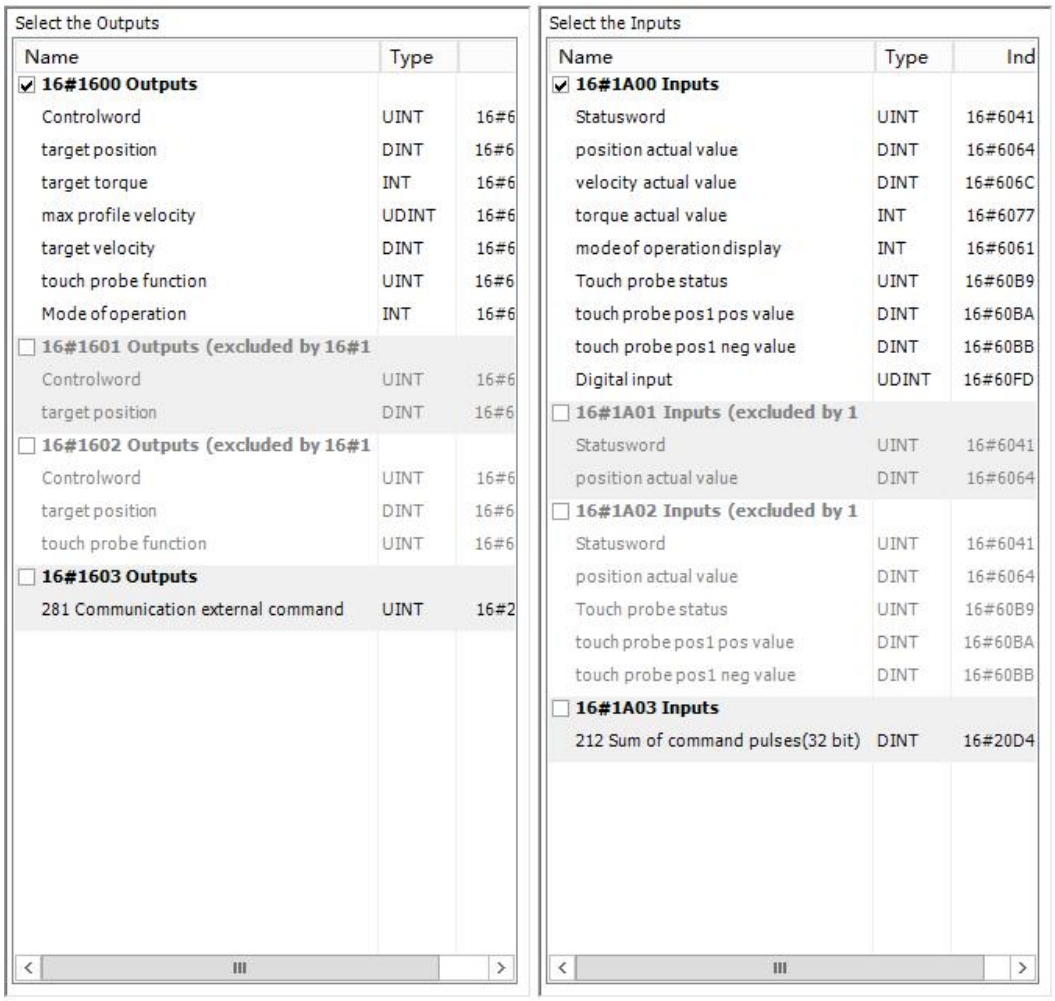
"Expert process data", please refer to the following diagram for specific operation



<Remarks> For the description of each parameter, please refer to the relevant servo instruction manual. For example, to use the A4N servo as the EtherCAT slave station, you need to refer to the *A4N Servo Instruction Manual*.

③ "Process data" tab

Display the input/output process data of the slave, which is defined by the name, type and index of the device description file.



The input (read) and output (write) parameter groups selected by the device can be used in the I/O map as inputs and outputs of the PLC (project variables may be mapped).

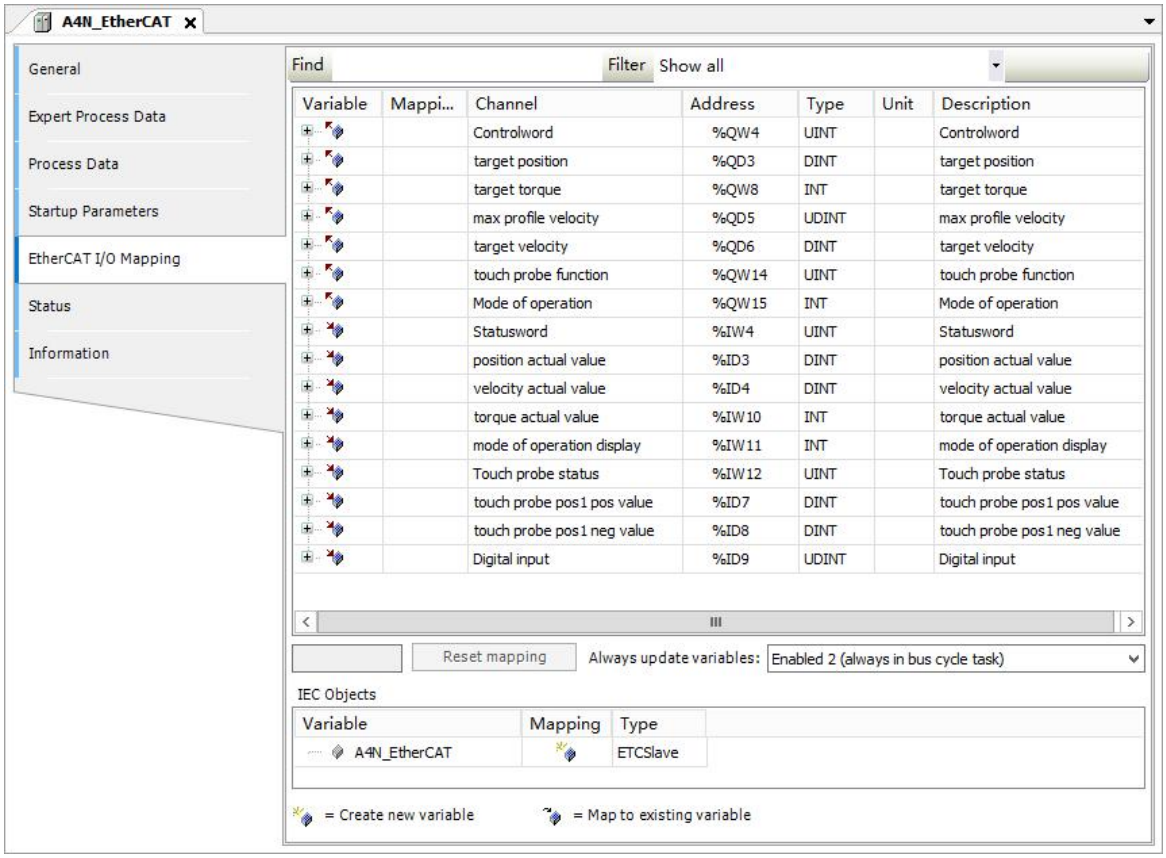
To modify the current selection, you need to cancel the selection, after which you can set another set.

④ "Startup parameters" tab

Define specific parameters for the device, which are transmitted by SDO or IDN when the system starts.

⑤ "EtherCAT I/O Mapping" tab

Displays the EtherCAT I/O mapping, and you can read/write the mapped parameters in this tab.



⑥ "Status" tab

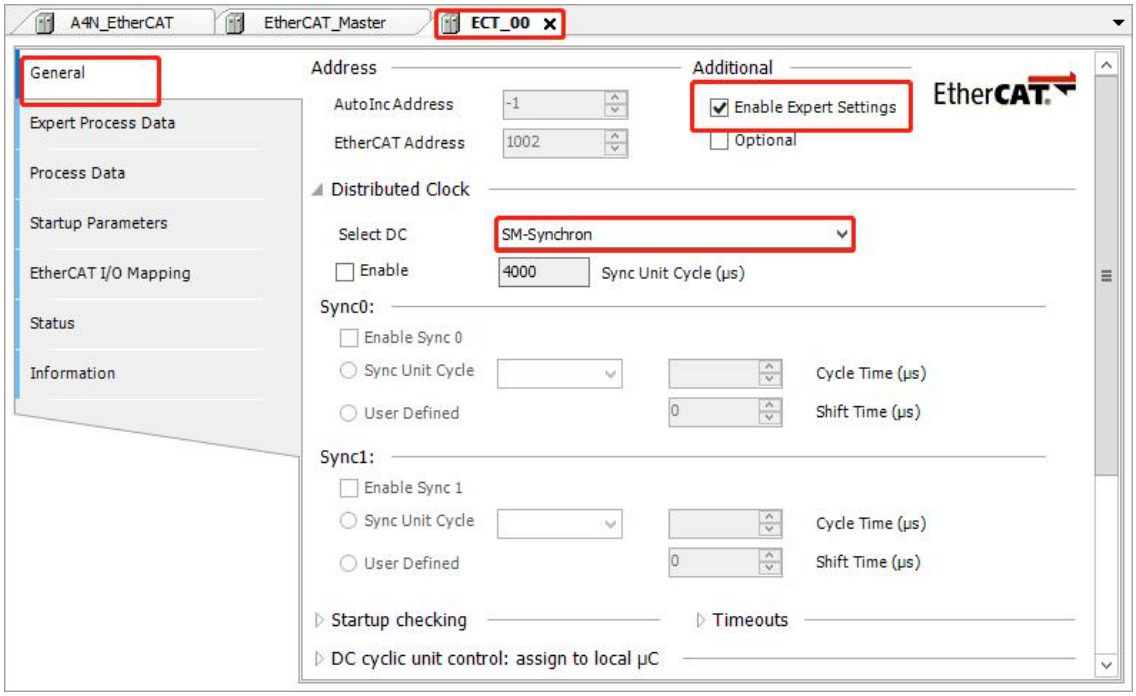
Display device status information (such as running, stopped, etc.) and device diagnostic information.

⑦ "Information" tab

Display the information of the current device, such as name, supplier, type, version number, module serial number, description, etc.

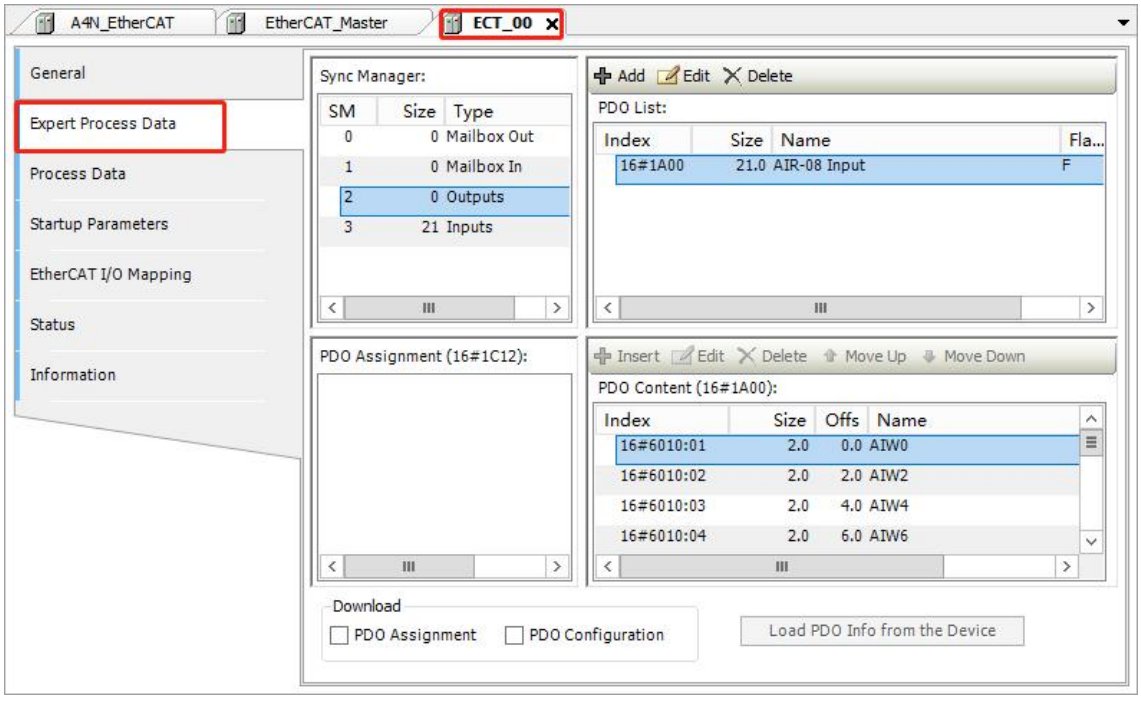
Configure ECT-00

① "Slave" tab: Check "Enable expert settings", select "DC for synchronization" for distributed clock.



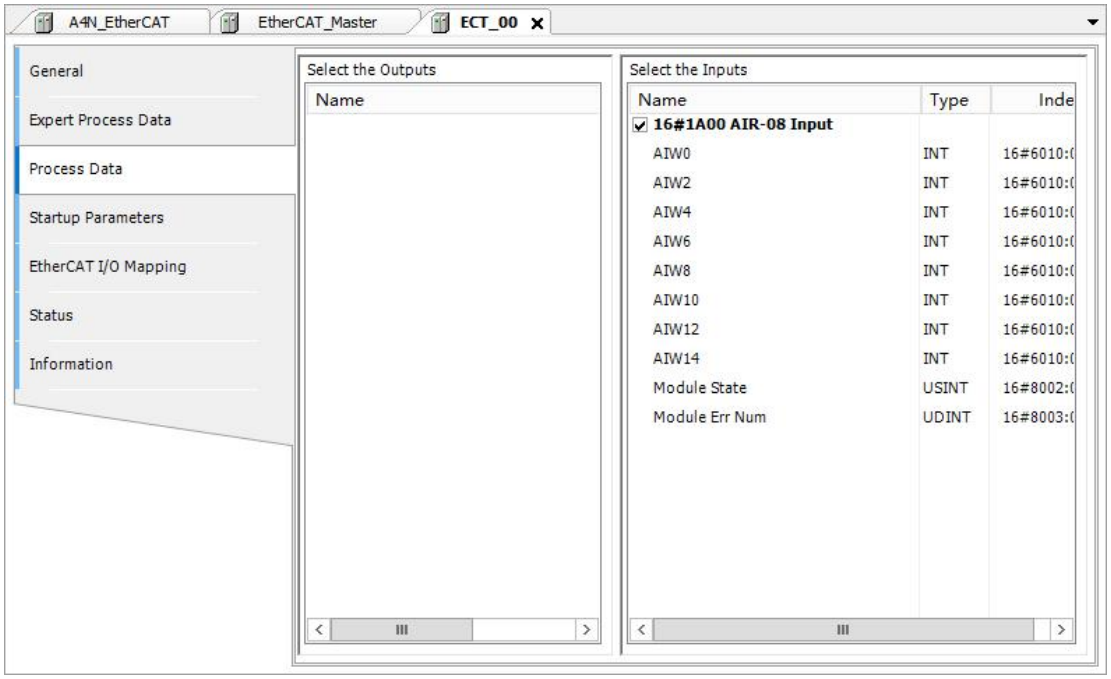
② "Expert process data" tab

Add/remove PDOs.



③ "Process data" tab

Display the input/output process data of the slave, which is defined by the name, type and index of the device description file.

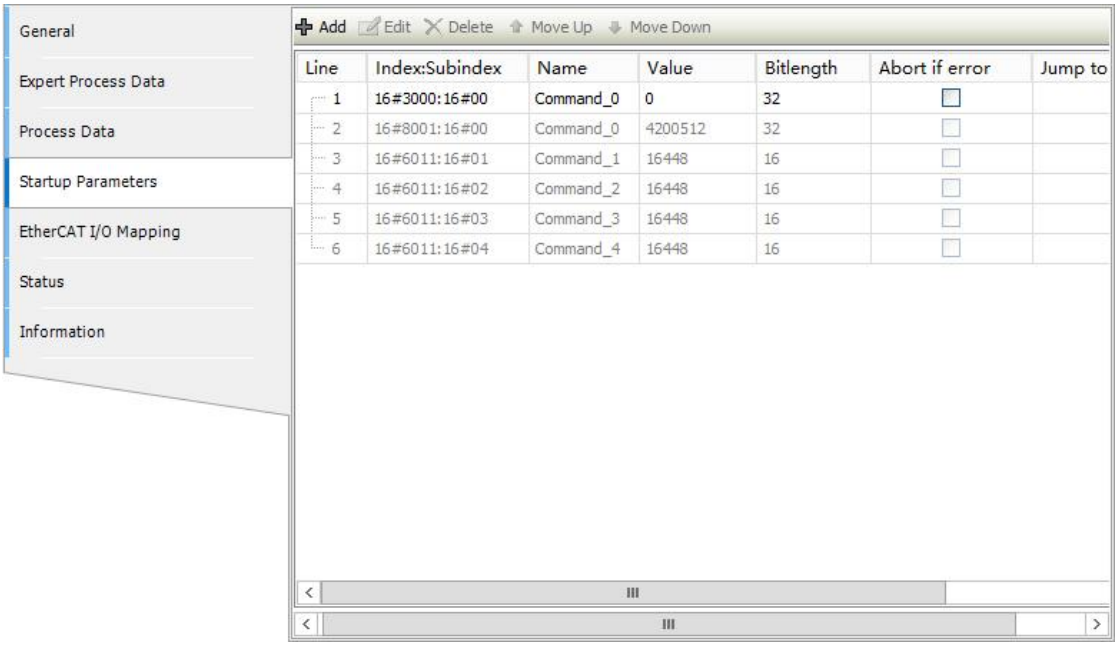


Input (read) and output (write) parameter groups can be used in the I/O map as inputs and outputs of the PLC (project variables may be mapped).

Note: The process data of the EtherCAT slave is all selected by default, please do not change it, otherwise it may cause communication failure.

④ "Startup parameters" tab

Define specific parameters for the device, which are transmitted by SDO or IDN when the system starts.

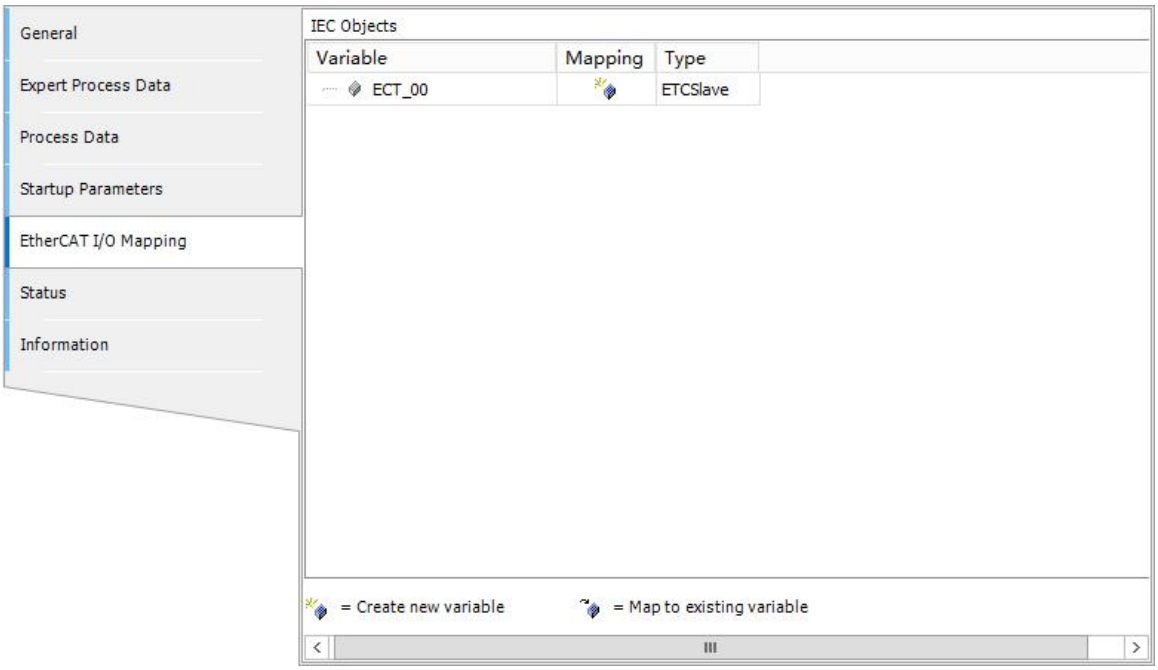


Note: The startup parameters of the EtherCAT slave cannot be modified. If you need to modify the configuration of the expansion module control word, please double-click to open the corresponding expansion module in the menu tree. After modifying the startup parameters,

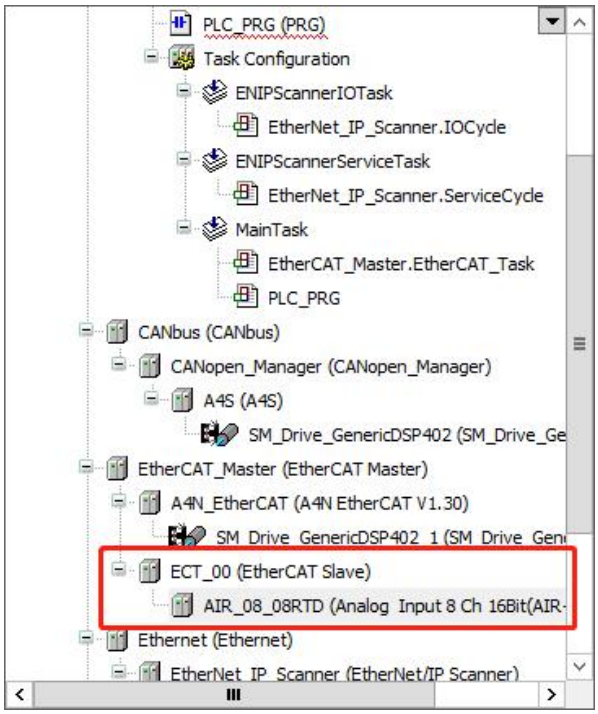
download the configuration to the CPU for operation, and the relevant configuration changes will take effect immediately. .

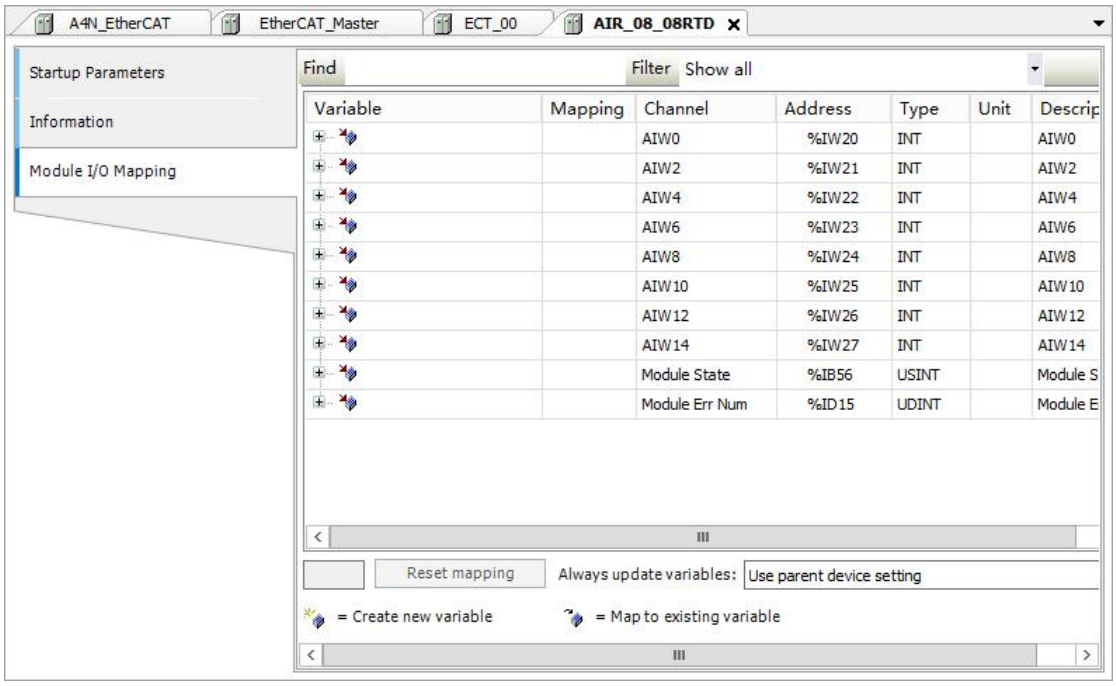
⑤ "EtherCAT I/O Mapping" tab

Displays EtherCAT I/O mapping.



To read and write the module parameters connected under "ECT_00", the operations are as follows: Debug its connection in the tab "Module I/O Mapping" of the ECT_00 slave (check "Always in the bus cycle task") module. Take the Digital Output 16 Bits digital input module as an example, click "DQ_16_16DQ (Digital Output 16 Bits)" under ECT_00, and select the "Module I/O" mapping, so that its parameters can be read and written.





Note: The addresses assigned to each expansion module are not fixed, and if modules are added/deleted, the corresponding assigned addresses will change.

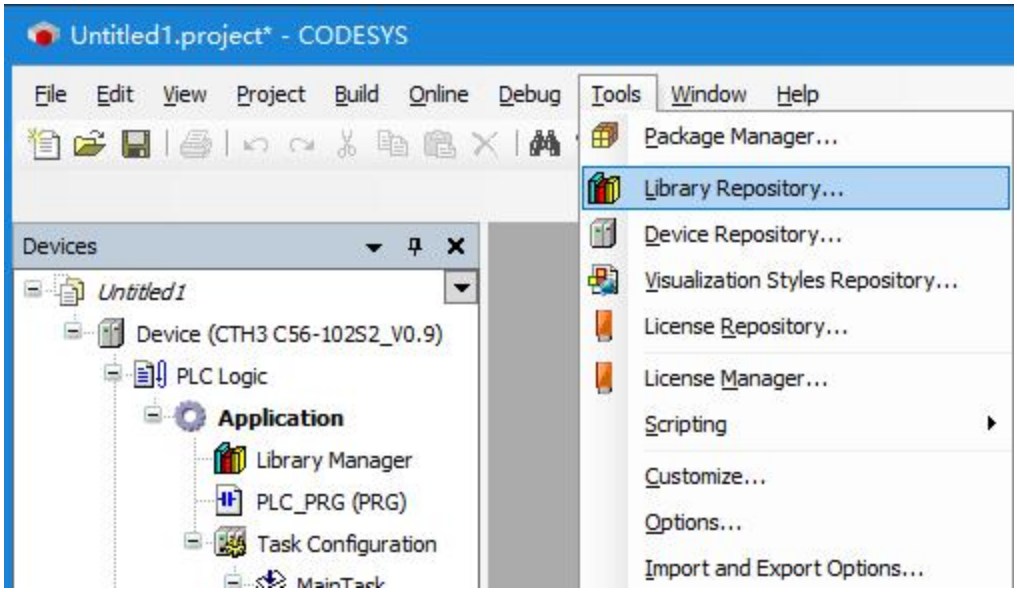
G Introduction of the ExtBus library

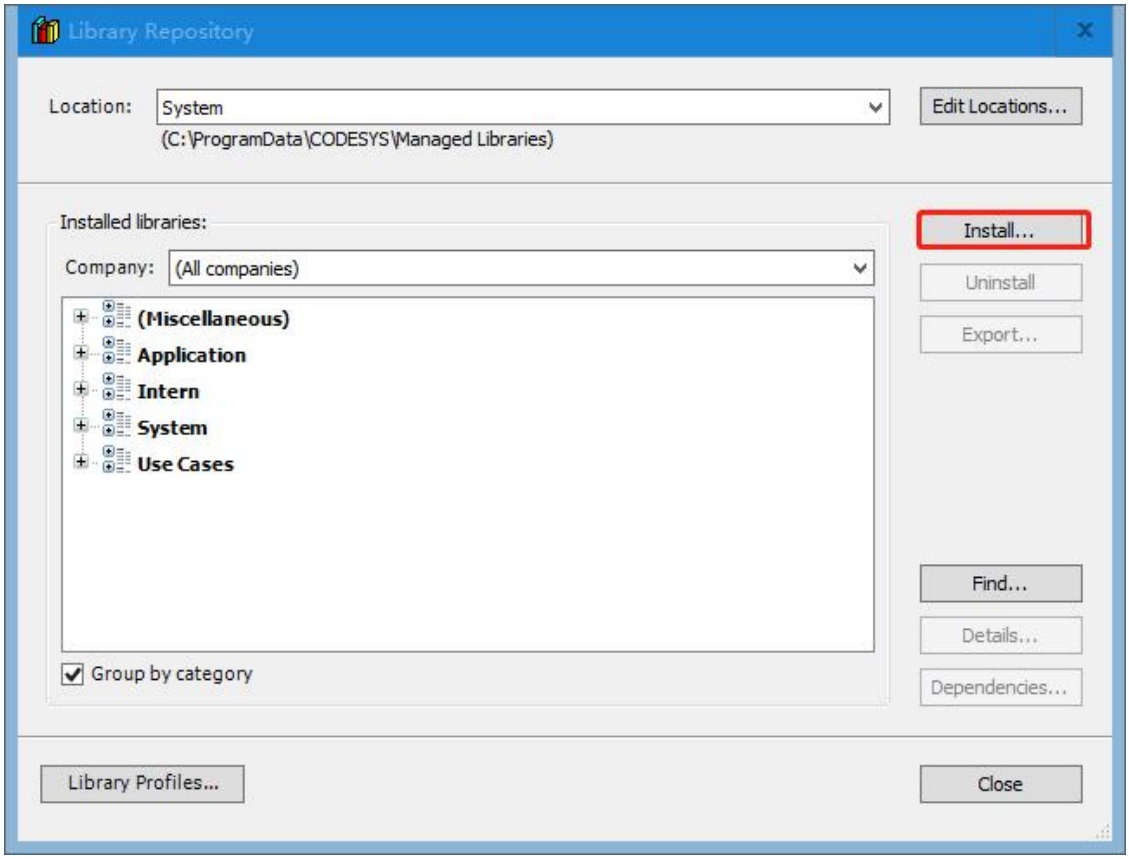
G.1 Install ExtBus library files

When using the high-speed counter, first install the ExtBus library file (please download the library file from the COTRUST website: http://www.co-trust.com), after "Add Library", you can call this library.

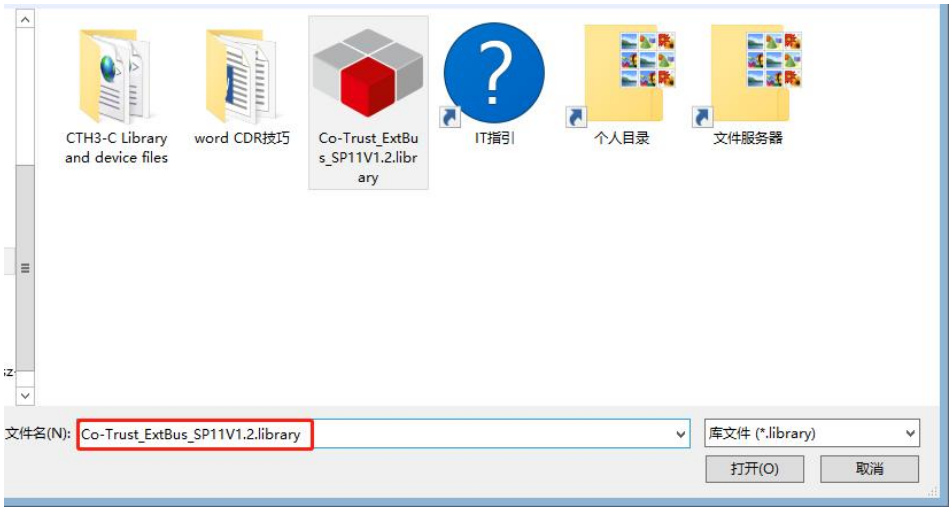
1) Install the library

Start the CODESYS software and select "Tools" → "Libraries Repository":





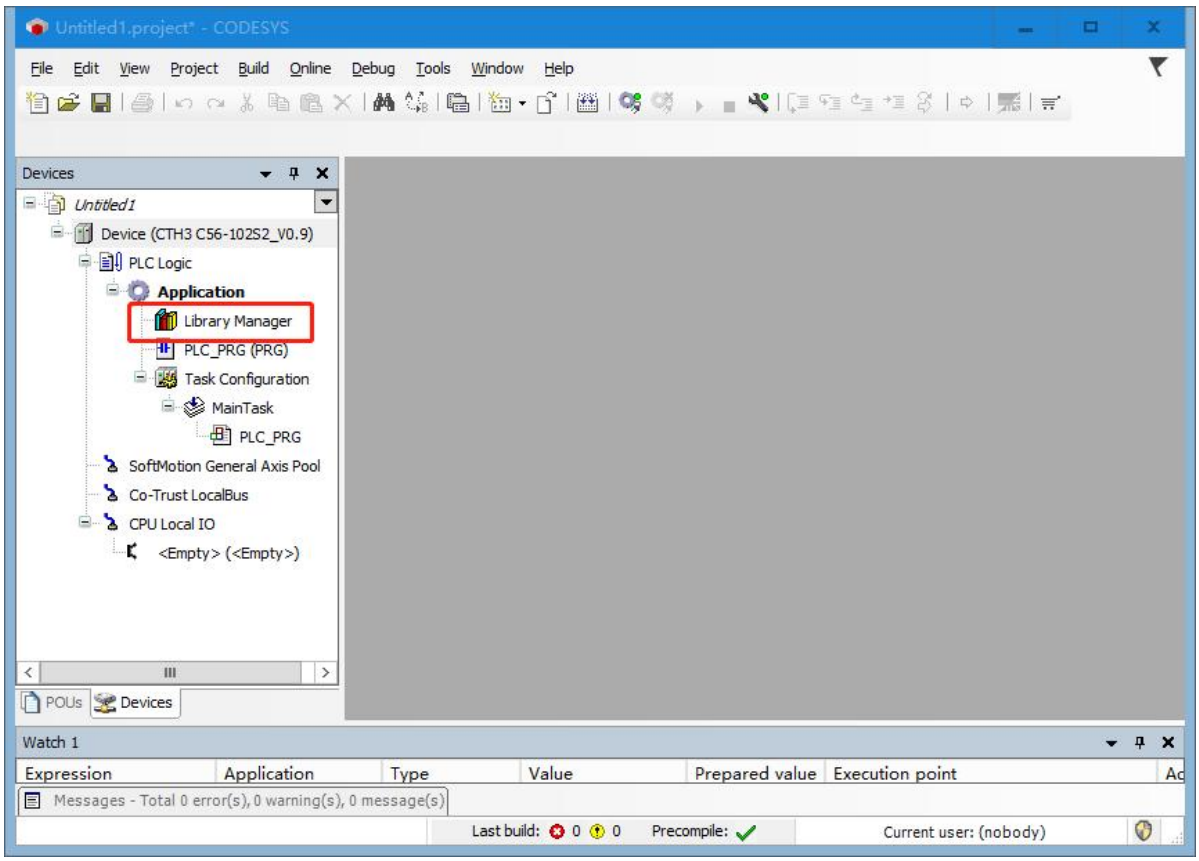
In the pop-up window, browse to the folder where "Co-Trust_ExtBus_SP11V1.2.library" is located, select "Co-Trust_ExtBus_SP11V1.2.library", and click the "Open" button. That means the Co-Trust_ExtBus_SP11V1.2.library library has been installed successfully.

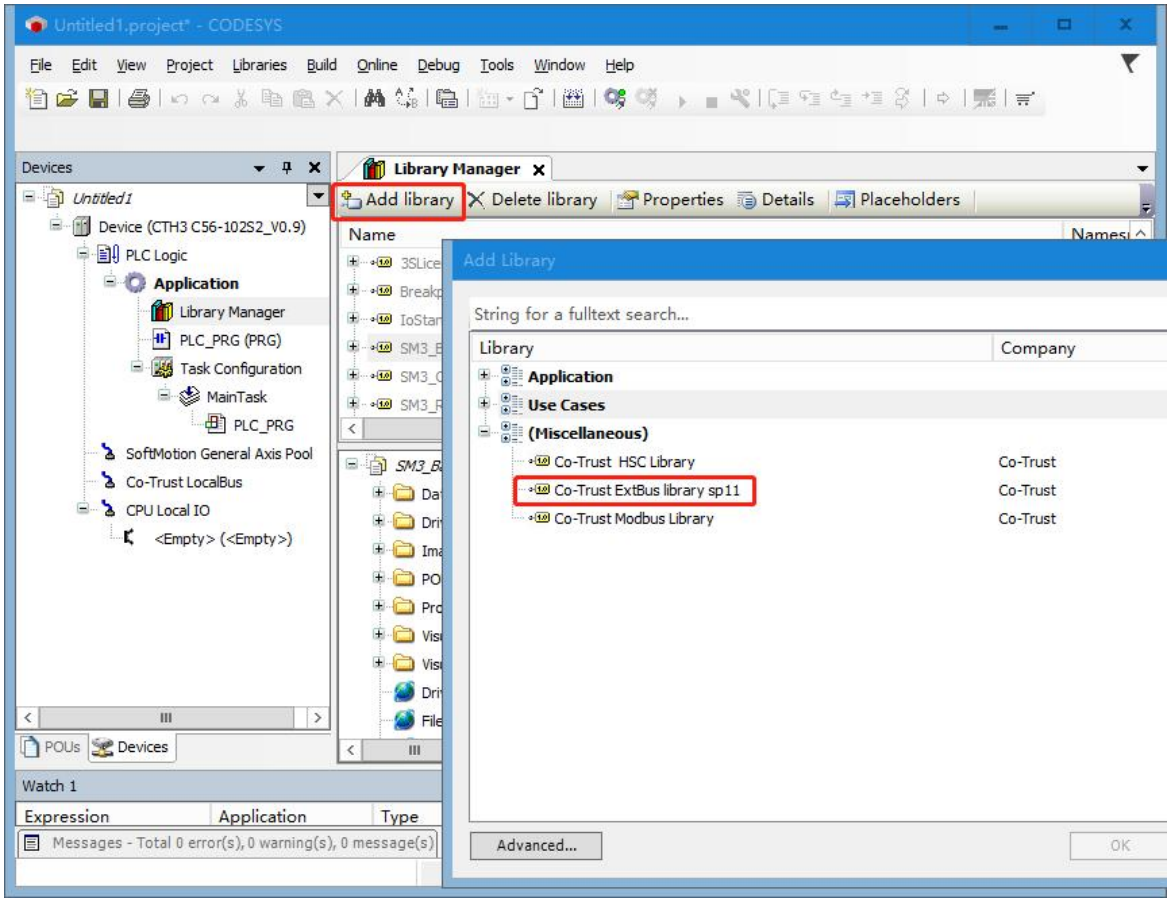


2) Add library

After the Co-Trust_ExtBus_SP11V1.2. library file is installed successfully, if the library is used in the current project, you need to execute "Add Library". The specific operation steps are as follows:

Expand "Device" → "PLC" → "Application" in the device view, double-click to open "Library Manager", and then click "Add Library" in the Library Manager dialog box to pop up the following "Add Library" dialog box:

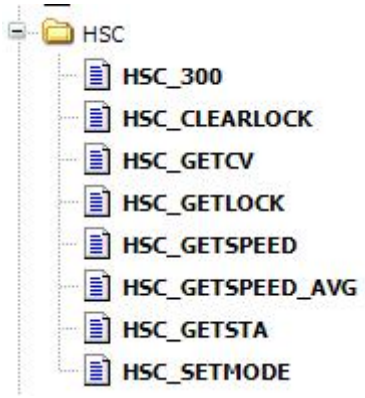




Expand "Miscellaneous", select "Co-Trust_ExtBus_SP11V1.2.library" and click "OK" to add the ExtBus library to the current project.

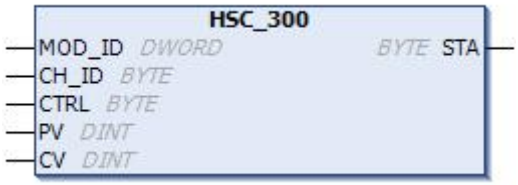
G.2 ExtBus library instruction description

The instruction Library (ExtBus) supported by the high-speed counter. Refer to the following description for the description of each instruction.



1. Set count parameter instruction

Name: HSC_300



Parameter	Input / output	Describe	Data type	Initialization	Remarks
MOD_ID	IN	Module address	DWORD	0	PLC native 16 # 0a000605, module ID of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0~5) HSC module value (0,1)
CTRL	IN	Control word	BYTE	16#F9	See the table below
PV	IN	Preset value	DINT	0	
CV	IN	Current value	DINT	0	
STA	OUT	Return status	BYTE	0	Module status word 0: OK 5: module parameter error 7: module does not respond 8: module link layer comparison error.

Control word(R/W)

7	6	5	4	3	2	1	0
Count enable	Current value update	Preset value update	Count direction update	Counting direction	Quadrature count selection		reset level

Reset level: 1-high reset, 0-low reset

Quadrature count options: 00-4x multiples, 01-2x multiples, 10-1x multiples

Counting direction: 0 - count down, 1 - count up

Counting direction update: 0-no update, 1-update

Default value update: 0-no update, 1-update

Current value update: 0 - no update, 1 - update

Counting enable: 0-disable, 1-enable

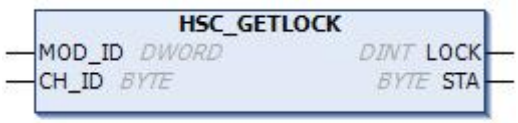
2. Clear the latched value

Name: HSC_CLEARLOCK

Parameter	Input / output	Describe	Data type	Initialization	Remarks
MOD_ID	IN	Module address	DWORD	0	PLC native 16 # 0a000605, module ID of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0~5) HSC module value (0,1)
STA	OUT	Module status word	BYTE	0	Module status word 0: OK 5: module parameter error 7: module does not respond 8: module link layer comparison error.

3. Get the current latch value

Name: HSC_GETLOCK



Parameter	Input / output	Describe	Data type	Initialization	Remarks
MOD_ID	IN	Module address	DWORD	0	PLC native 16 # 0a000605, module ID of hardware

					configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0~5) HSC module value (0,1)
LOCK	OUT	Latch value	DINT	0	Latch value
STA	OUT	Module status word	BYTE	0	Module status word 0: OK 2: Illegal parameter 5: Module parameter error 7: Module not responding 8: Module link layer comparison error.

4. Get the current count value

Name: HSC_GETCV

Parameter	Input / output	Describe	Data type	Initialization	Remarks
MOD_ID	IN	Module address	DWORD	0	PLC native 16 # 0a000605, module ID of hardware configuration
CH_ID	IN	Passageway	BYTE	0	PLC local value (0~5) HSC module value (0,1)
CV	OUT	Current count value	DINT	0	Current count value
STA	OUT	Module status word	BYTE	0	Module status word 0: OK 2: Illegal parameter 5: Module parameter error 7: Module not responding 8: Module link layer comparison error.

5. Get the current counting speed

Name: HSC_GETSPEED

Parameter	Input / output	Describe	Data type	Initialization	Remarks
MOD_ID	IN	Module address	DWORD	0	PLC native 16 # 0a000605, module ID of hardware configuration

CH_ID	IN	Channel address	BYTE	0	PLC local value (0~5) HSC module value (0,1)
SPEED	OUT	Counting speed	DWORD	0	Hz
STA	OUT	Module status word	BYTE	0	Module status word 0: OK 2: Illegal parameter 5: Module parameter error 7: Module not responding 8: Module link layer comparison error.

6. Get the current speed using the average

Name: HSC_GETSPEED-AVG

Parameter	Input / output	Describe	Data type	Initialization	Remarks
MOD_ID	IN	Module address	DWORD	0	PLC native 16 # 0a000605, module ID of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0~5) HSC module value (0,1)
BUFSIZE	IN	Average buffer	DINT	16	Average buffer greater than 0 and less than 64
DeadBand	IN	deadband	DWORD	20000	If the difference between the average value and the existing value is less than the dead zone value, the average value shall prevail, and the value greater than the dead zone average value is the existing value
SPEED	OUT	Counting speed	DWORD	0	Hz
STA	OUT	Module status word	BYTE	0	Module status word 0: OK 2: Illegal parameter 5: Module parameter error 7: Module not responding 8: Module link layer comparison error.

7. Get the current count status

Name: HSC_GETSTA

Parameter	Input / output	Describe	Data type	Initialization	Remarks
MOD_ID	IN	Module address	DWORD	0	PLC native 16 # 0a000605, module ID of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0~5) HSC module value (0,1)
HSC_STA	OUT	Count status	BYTE	0	Bit0~bit3: current mode Bit4: reserved Bit5: HSC0 current counting direction bit: 1 = count up Bit6=1: the current value is equal to the preset value bit Bit7=1: the current value is greater than the preset value
STA	OUT	Module status word	BYTE	0	Module status word 0: OK 2: Illegal parameter 5: Module parameter error 7: Module not responding 8: Module link layer comparison error.

8. Set the counter mode

Name: HSC_SETMODE

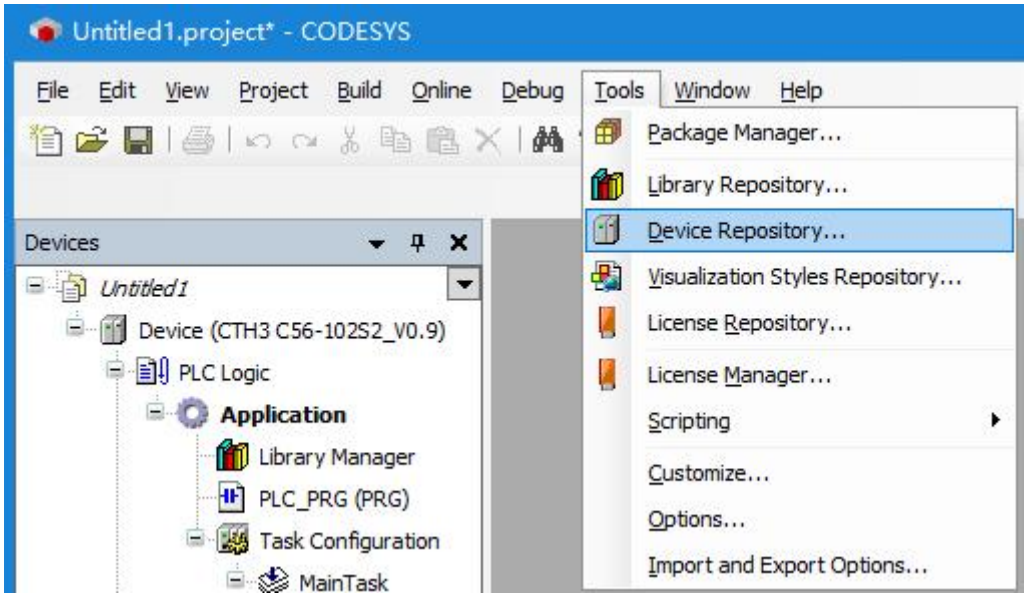
Parameter	Input / output	Describe	Data type	Initialization	Remarks
MOD_ID	IN	Module address	DWORD	0	PLC native 16 # 0a000605, module ID of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0~5) HSC module value (0,1)
MODE	IN	Counting mode	BYTE	0	Bit0~bit3: HSC counting mode (this machine supports 0,1,3,4,6,7,9,10; hsc-02 supports 0~11 mode) Mode 0~2: single phase counter with internal direction control. Mode 3~5: single phase counter with external direction.

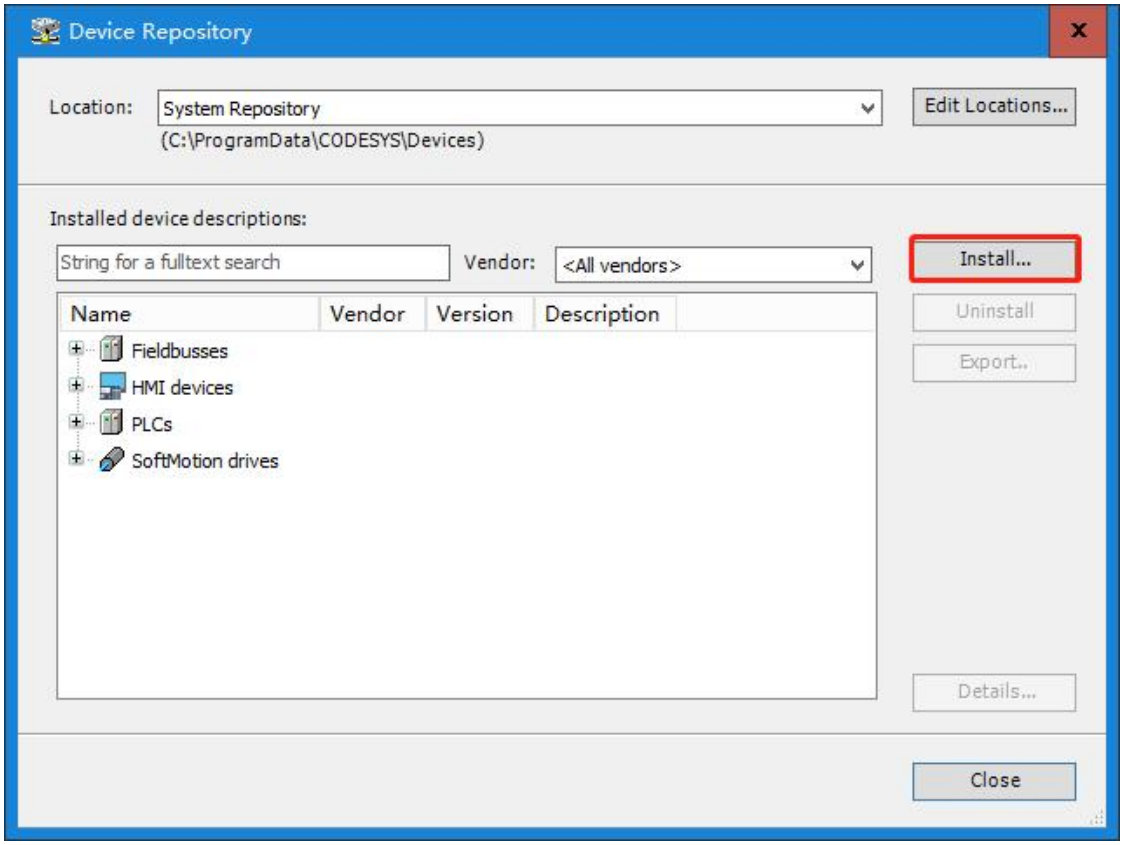
					Mode 6-8: dual phase counter with 2 clock inputs. Mode 9~11: A\B phase quadrature counter. Bit4: Z signal latching function, 0: latching, 1: not latching Bit5: Z signal zeroing function, 0: zeroing, 1: not zeroing Bit6: reserved Bit7: Latch value cleared 0: invalid, 1: valid
STA	OUT	Module status word	BYTE	0	Module status word 0: OK 2: Illegal parameter 5: Module parameter error 7: Module not responding 8: Module link layer comparison error.

H Installing device description files in CODESYS

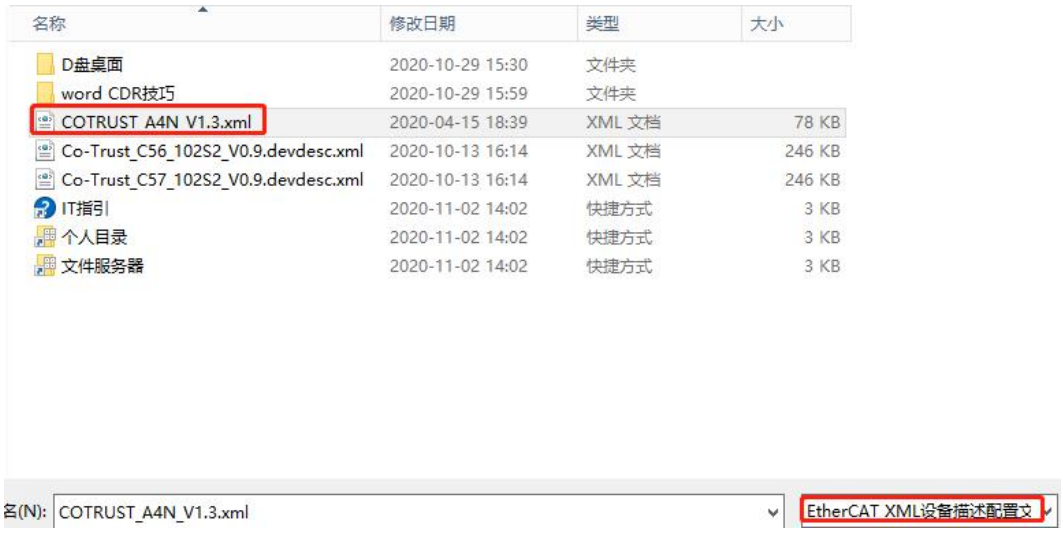
H.1 Install EtherCAT servo slave device description file

1. Open the CODESYS software, select "tools" → "Device library", and click "Install" in the dialog box.



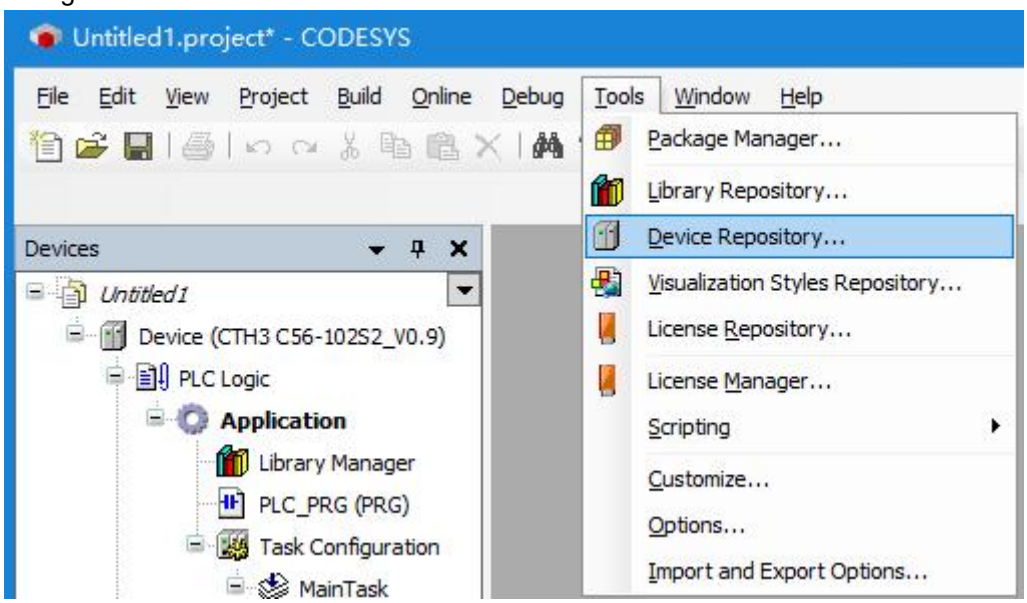


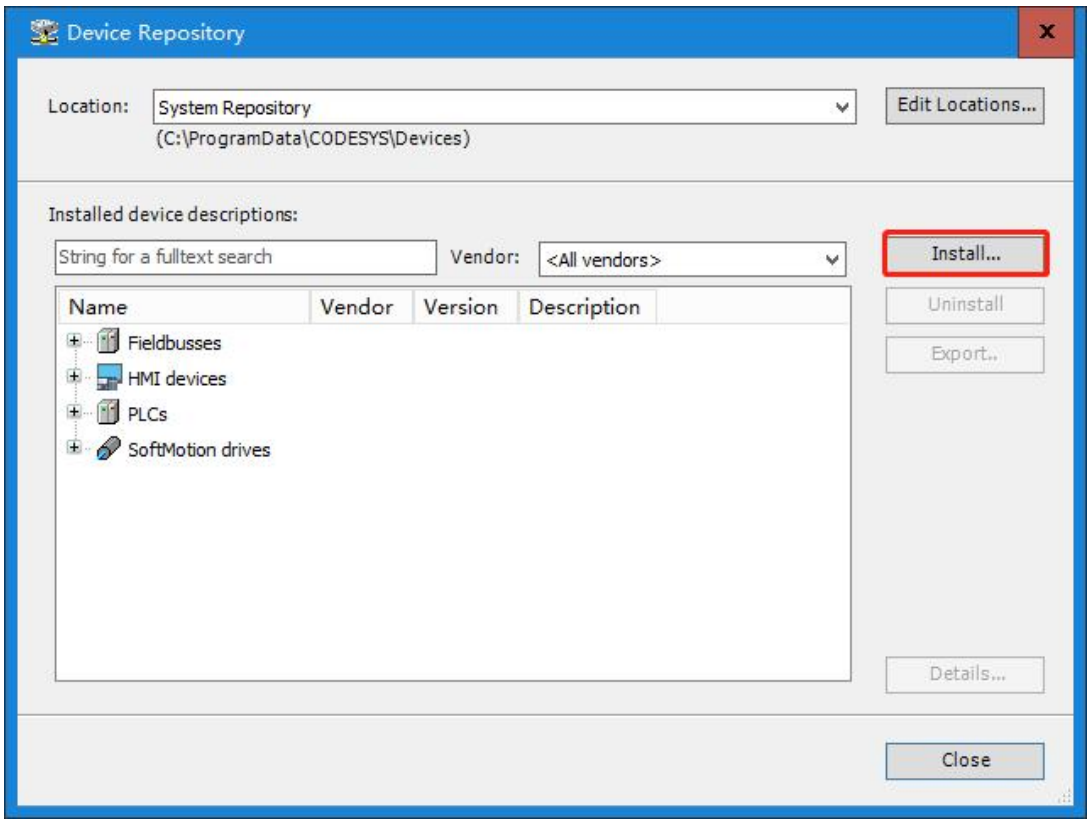
2. After performing the above operations, please select the file type "EtherCAT XML device description configuration file (*.xml)", select the file "trust_a4n \u v1.3.xml", and click "open" to install.



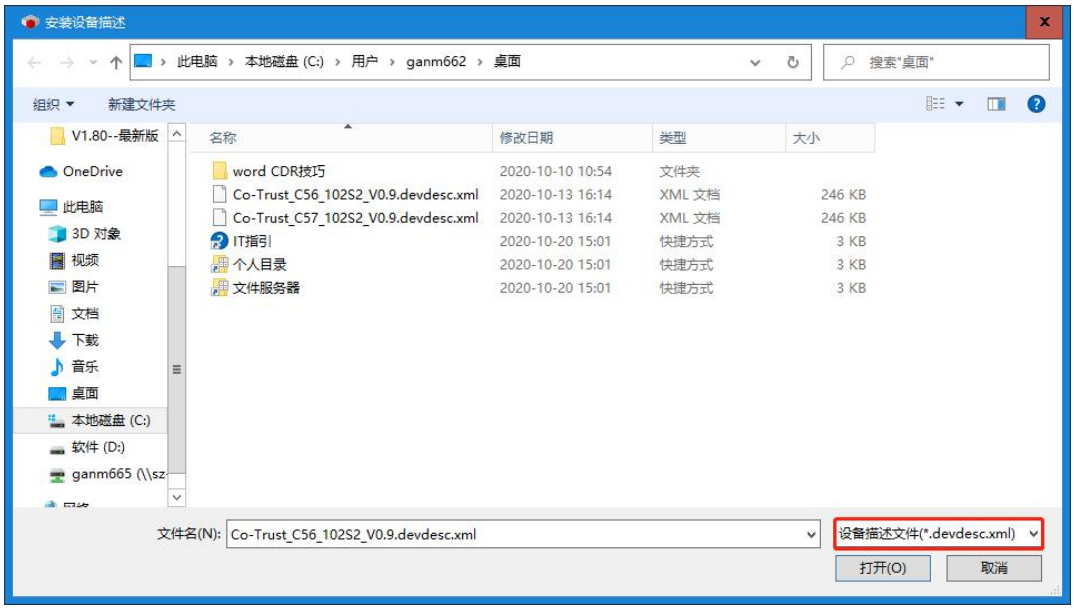
H.2 Installation of PLC / 402 axis equipment description document

1. Open the CODESYS software, select "tools" → "Device library", and click "Install" in the dialog box.



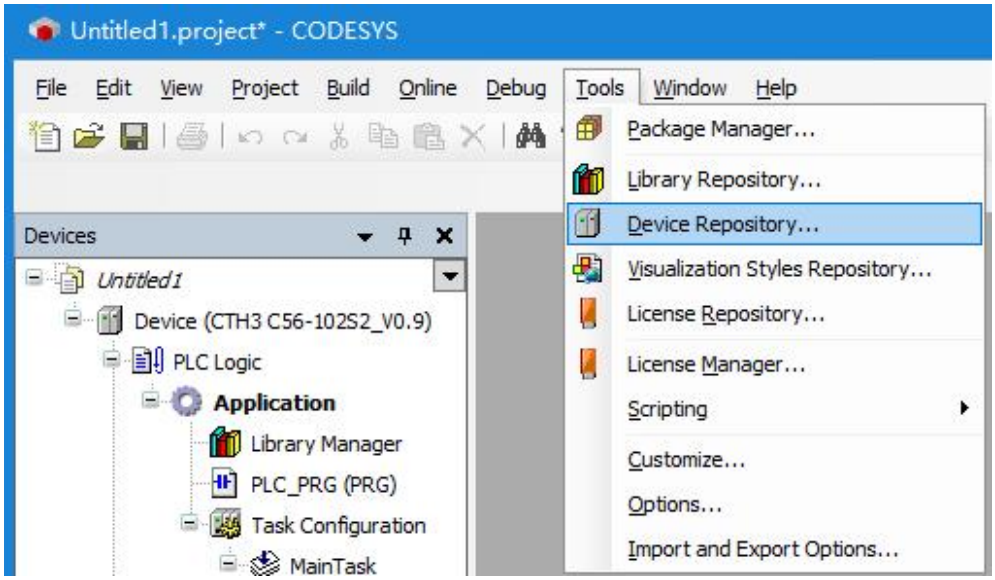


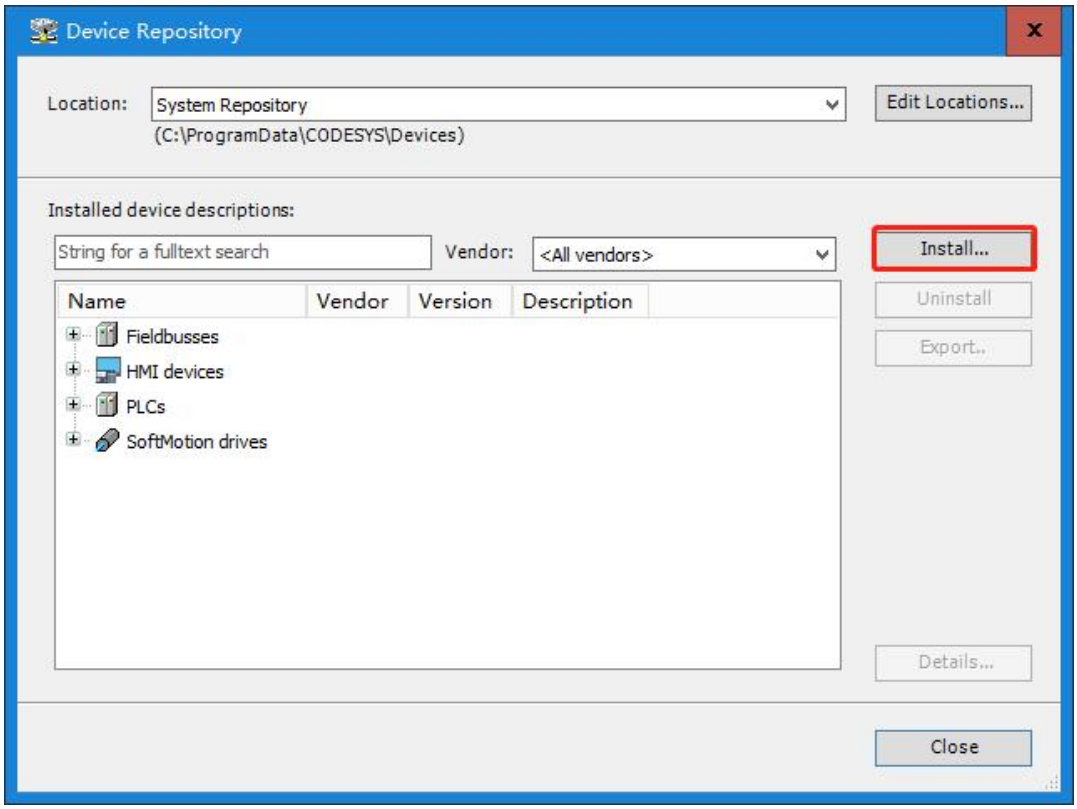
2. After performing the above operations, a dialog box will pop up as follows. Please select the file type "device description configuration file (*. Devdesc. XML)", select the file "Co trust_c56_102s2_v0.9. Devdesc. XML" or "Co trust_c57_102s2_v0.9. Devdesc. XML", and click "open" to install.



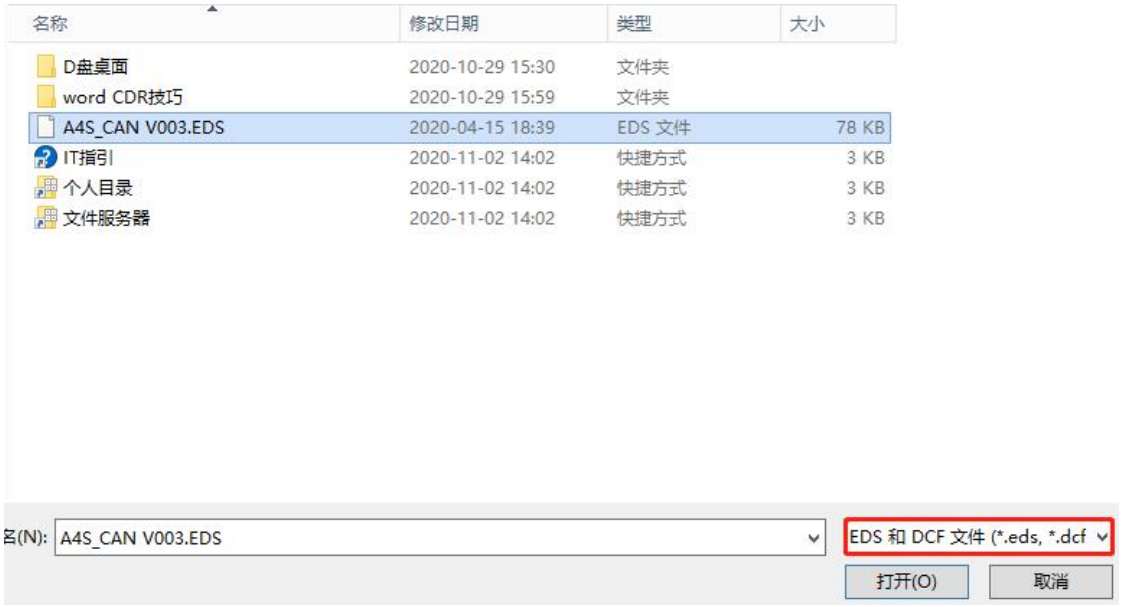
H.3 Install CANopen EDS file

1. Open the CODESYS software, select "tools" → "Device library", and click "Install" in the dialog box.





2. After performing the above operations, a dialog box will pop up as follows. Please select the file type "EDS and DCF files (*. Eds. *. DCF)", select the files with suffix. EDS such as "A4S_can_v003. EDS", and click "open" to install.


名称
修改日期
类型
大小
D盘桌面
2020-10-29 15:30
文件夹
word CDR技巧
2020-10-29 15:59
文件夹
A4S_CAN V003.EDS
2020-04-15 18:39
EDS 文件
78 KB
IT指引
2020-11-02 14:02
快捷方式
3 KB
个人目录
2020-11-02 14:02
快捷方式
3 KB
文件服务器
2020-11-02 14:02
快捷方式
3 KB
名(N):
A4S_CAN V003.EDS
EDS 和 DCF 文件 (*.eds, *.dcf
打开(O)
取消

I Instruction quick check

I.1 Operator summary in CODESYS

The following table summarizes the operators, which exist in the Standard.lib and Util.lib libraries of CODESYS respectively.

Note: The 'IL Symbol' column is displayed only for lines that use this symbol. The first set of necessary operands has been successfully loaded into the previous line (eg, LD in)

'Mod.IL' shows the modifiers that appear in the IL:

C	Modifier for the result of the preceding expression being TRUE.
N	For JMPC, CALC, RETC: If the result of the previous line is FALSE, execute this command.
N	else: negated operand (non-accumulator)
(	Operators in parentheses: The operation starts only after the parentheses appear.

Table I-1 Operator Summary Table

in ST	in AWL	Mod. AWL	Description
'			string delimiter
.. **[ ]**			Array range size
:			Separator between operand and type in declaration.
;			Description terminated.
^			Designator obsolete.
	LD var1	N	Loading of variable 1 value in buffer.
:=	**ST var1**	N	Store the actual result in variable 1.
	S boolvar		When the actual result is TRUE, set the operand boolean variable to TRUE.
	R boolvar		When the actual result is FALSE, set the operand boolean variable to TRUE.
	JMP label	**CN**	Jump directly to the label.
<Program name>	**CAL** prog1	**CN**	Call prog1。
<Instance name>	**CAL** inst1	**CN**	Call inst1。
<Fctname>(vx, vy,..)	**<Fctname> vx, vy**	**CN**	Call fctname and transmit Variables vx,vy。
RETURN	**RET**	CN	Leave POU and go back to caller
	(		The values after the parentheses are treated as operands, and the operations before the parentheses are not executed before the expressions in the parentheses.

	)		Executes the set operation now
AND	**AND**	N,(	
OR	**OR**	N,(	
XOR	**XOR**	N,(	Bit dedicated OR
NOT	**NOT**		
+	**ADD**	(	
-	**SUB**	(	subtraction
*****	**MUL**	(	multiplication
/	**DIV**	(	division
>	**GT**	(	greater than
>=	**GE**	(	greater than or equal to
=	**EQ**	(	be equal to
<>	**NE**	(	not equal to
<=	**LE**	(	less than or equal to
<	**LT**	(	less than
MOD(in)	**MOD**		Modulo division
INDEXOF(in)	**INDEXOF**		Internal exponent of POU in1
SIZEOF(in)	**SIZEOF**		Number of bytes required for known "in" data type
SHL(K,in)	**SHL**		The operator "in" bit shifts left by k
SHR(K,in)	**SHR**		The operator "in" bit shifts right by k
ROL(K,in)	**ROL**		"in" shift left by K bits
ROR(K,in)	**ROR**		"in" shifts right by K bits
SEL(G,in0,in1)	**SEL**		Binary selection between two operands in0 (G is FALSE) and in1 (G is TRUE).
MAX(in0,in1)	**MAX**		Restoration of the larger of the 2 values.
MIN(in0,in1)	**MIN**		Restoration of the smaller of the two values in0 and in1.
LIMIT(MIN,in,Max)	**LIMIT**		Limit the range (in resets to MIN, or MAX to avoid going out of range).
MUX(K,in0,...in_n)	**MUX**		Select the kth value from a set of values (in to In_n).
ADR(in)	**ADR**		The address of the operand in DWORD.
ADRINST()	**ADRINST**()		The call operator is the address of the function block operator.
BITADR(in)	**BITADR**		Bit balance of operands in DWORD.
BOOL_TO_<type>(in)	**BOOL_TO_<type>**		Type conversion of boolean operands.
<type>_TO_BOOL(in)	**<type>_TO_BOOL**		Type conversion to BOOL.
INT_TO_<type>(in)	**INT_TO_<type>**		INT operand type conversion bits to another basic type.
REAL_TO_<type>(in)	**REAL_TO_<type>**		The REAL operand type is converted to another base type.
LREAL_TO_<type>	**LREAL_TO_<type>**		The LREAL operand type is converted to

(in)			another base type.
TIME_TO_<type> (in)	**TIME_TO_<type>**		TIME operand type conversion bit another basic type.
TOD_TO_<type>(in)	**TOD_TO__<type>**		The TOD operand type casts another base type.
DATE_TO_<type>(in)	**DATE_TO_<type>**		The DATE operand type is converted to another base type.
DT_TO_<type>(in)	**DT_TO_<type>**		DT operand type conversion bits to another basic type.
STRING_TO_<type>(in)	**STRING_TO_<type>**		String operand type conversion bit another basic type, "in" must contain a valid value of the requested type.
TRUNC(in)	**TRUNC**		Convert from REAL to INT.
ABS(in)	**ABS**		Absolute value of operand “in”.
SQRT(in)	**SQRT**		Square root of operand "in".
LN(in)	**LN**		The natural logarithm of operand "in".
LOG(in)	**LOG**		The base 10 logarithm of operand "in".
EXP(in)	**EXP**		The power of operand "in".
SIN(in)	**SIN**		Sine of operand “in”.
COS(in)	**COS**		Cosine of operand “in”.
TAN(in)	**TAN**		Tangent of operand “in”.
ASIN(in)	**ASIN**		Inverse sine of operand “in”.
ACOS(in)	**ACOS**		Arc cosine of operand “in”.
ATAN(in)	**ATAN**		Arc tangent of operand “in”.
EXPT(in,expt)	**EXPT** expt		The power of the operand "in with expt".

<Remark> Please get detailed usage information in the relevant appendix of the CODESYS library.

I.2 Standard.lib

表 I-2 Standard.lib instruction

in ST	in AWL	describe
LEN(in)	**LEN**	String length function. Returns the number of characters in STR.
LEFT(str,size)	**LEFT**	Return leftmost SIZE characters of STR.
RIGHT(str,size)	**RIGHT**	Returns rightmost SIZE characters of STR.
MID(str,size,pos)	**MID**	Return LEN characters of STR, beginning at the POS-th character position.
CONCAT('str1','str2')	**CONCAT** 'str2'	Concatenation of two strings.
INSERT('str1','str2',pos)	**INSERT** 'str2',p	Insert STR2 into STR1 after the POS-th character position.
DELETE('str1',len,pos)	**DELETE** len,pos	Delete LEN characters of STR, beginning at the POS-th character position.
REPLACE('str1','str2',len,pos)	**REPLACE** 'str2',len,pos	Replaces L characters of STR1 by STR2, starting at the POS-th character position

		and returns the new string.
FIND('str1','str2')	**FIND** 'str2'	Find the character position of the beginning of the first occurence of STR2 in STR1.
SR	**SR**	Bistable function block (reset dominant)
RS	**RS**	Bistable function block (set dominant)
SEMA	**SEMA**	FB: 臂板软件(可中断)
R_TRIG	**R_TRIG**	FB: Rising edge detection
F_TRIG	**F_TRIG**	FB: Falling edge detection
CTU	**CTU**	FB: Counter Up:
CTD	**CTD**	FB: Counter Down:
CTUD	**CTUD**	FB: Counter Up Down:
TP	**TP**	FB: Pulse timer.
TON	**TON**	FB: On-Delay timer.
TOF	**TOF**	FB: Off-Delay timer.
RTC	**RTC**	FB: FUNCTION_BLOCK RTC

I.3 Util.lib

Table I-3 Util.lib instruction

BCD_TO_INT	Converts one byte in BCD format into an INT value
INT_TO_BCD	Converts an INT value into a byte in BCD code
EXTRACT(in,n)	The nth bit of DWORD in is restored as BOOL.
PACK	Up to 8 bits are pushed into a byte.
PUTBIT	Sets a bit of a DWORD value
UNPACK	Converts a byte into 8 bits
DERIVATIVE	Approximates the derivative of a given value in its time course
INTEGRAL	Integer.
LIN_TRAFO	Performs a linear transformation
STATISTICS_INT	Calculates minimum, maximum, and average of an input value
STATISTICS_REAL	Calculates minimum, maximum, and average of an input value
VARIANCE	Calculates the mathematical variance of a variable over time
PD	PD controller.
PID	PID controller.
BLINK	Simulates a blinking signal (turning on and off for specific durations)
FREQ_MEASURE	Measured frequency of the boolean input signal.
GEN	Generates periodic functions of different, given types
CHARCURVE	Linear function
RAMP_INT	Limits the slope of a value to a certain value
RAMP_REAL	Limits the slope of a value to a certain value
HYSTERESIS	Realizes a hysteresis function
LIMITALARM	Monitors if an input value is between a lower and a upper limit

I.4 SoftMotion Lib

SM3_Basic Library

This library is the base library for CODESYS SoftMotion applications, so it must be included in the SoftMotion project. It will happen automatically when a SoftMotion device is plugged in.

The library provides the following function blocks and functions:

● PLCopen function block allows both single-axis motion control and two-axis synchronous motion. In addition to the library elements for status checking, parameterization and general operation, there are function blocks for driving the axis with defined velocity and acceleration parameters. There are also modules for synchronizing master, slave and drive axis.

The function blocks are as follows:

● driver interface

Name	Function
AXIS_REF	This driver interface is a function block. It is provided by the "SM3_Basic.library" library and includes work as a driver. Each SoftMotion axis is an extension of the AXIS_REF_SM3 instance.

● data type, global variable

Name	Function
MC_DIRECTION	It is used as an input to several function blocks and specifies the direction of movement. Note that not all modes are available for all function blocks and axis types (modulo/finite).
MC_TAPPETMODE	It is used in the function block MC_DigitalCAMSwitch to determine whether the tappet value should refer to the spindle setpoint (1), or the actual position (2). In the case of automatic detection (0), the decision of the function block depends on: Whether the control state of the drive uses the setpoint (bRegulatorOn = TRUE) or the actual value (bRegulatorOn = FALSE).
SMC_BRAKESETSTATE	It is used in the function block SMC3_BrakeControl and determines how the drive sets the mechanical brake.
SMC_CAMTAPPETACTION	It determines which switching actions are performed when the cam has been passed.
SMC_CAMTAPPETTYPE	It is used to determine which direction the cam must go in order to be activated.
SMC_CONTROLLER_MODE	Describes the controlled variables of the motor.
SMC_ERROR	All error numbers that the SoftMotion function block may return.
SMC_HOMING_MODE	It defines the homing sequence when used by SMC_Homing.
SMC_INT_STATUS	It describes the state of the interpolator instance.
SMC_LANGUAGE_TYPE	It is used in the function SMC_ErrorString to determine the

	language.

● data type, structure

Name	Function
MC_CAM_ID	It is an internal data structure that describes the CAM workbench. It is generated by MC_CAMTableSelect and used as input to MC_CamIn.
MC_CAM_REF	This function block represents a generic cam.
MC_CAMSWITCH_REF	It describes the CAMSwitches table and is used as input to MC_DigitalCAMSwitch.
MC_CAMSWITCH_TR	It describes the toggle position of the follower.
MC_OUTPUT_REF	It defines the tappet output in the form of an array of 32 booleans. TYPE MC_OUTPUT_REF : ARRAY [0..31] OF BOOL; END_TYPE
MC_TRACK_REF	

● DriveBasic

Name	Function
SMC3_BrakeControl	This function block determines the behavior of the mechanical brake if supported by the drive and its SoftMotion driver.
SMC3_BrakeStatus	It reads the state of the mechanical brake.
SMC_SetControllerMode	This module can be used to switch to other controller modes if supported by the drive device.
SMC_GetMaxSetAccDec	It is used to measure the maximum value of the absolute value of the acceleration (or deceleration) of an axis.
SMC_GetMaxSetVelocity	It is used to measure the maximum value of a shaft speed.
SMC_GetTrackingError	It measures the actual and maximum hysteresis error used to compensate for dead time, which may occur during fieldbus communications and will last for many cycles (byDeadTimeCycles).
SMC_InPosition	It checks if the actual position of the axis is within a certain position value for a certain period of time.
SMC_MeasureDistance	It is used on a rotary axis to measure distance traveled (the casing is also counted).
SMC_CheckLimits	It can be used to check if the actual setpoint of the drive exceeds the maximum value configured in the controller. The result of the check will be indicated by the bLimitsExceeded output.
SMC_FollowPosition	It will write the position set point to the axis without any checks.
SMC_FollowPositionVelocity	Its use is equivalent to SMC_FollowPosition, except for possibly additionally defined speeds.
SMC_FollowSetValues	The set value is written to the axis without any checking.

SMC_FollowVelocity	Writes the set speed to the axis without any checks.
SMC_ClearFBEror	Delete the oldest function block error message.
SMC_ReadFBError	Read the oldest information on function block errors.
SMC_Homing	It performs the reference movement of the axis. As the reference switch, a boolean value is used, which is typically input by a hardware.
SMC_ChangeGearingRatio	With the help of this module, gear ratios and drive equipment types can be modified.

● Error

Name	Function
SMC_ErrorString	Depending on the input ErrorID and language, the function SMC_ErrorString will return a string representing the error.

● File

Name	Function
SMC_ReadCam	It is used to download the CAM at runtime and make it available to MC_CamTableSelect and MC_CamIn.
SMC_WriteCam	This function block writes a CAM created in the CAM editor to *.CAM-file at runtime.
SMC_AxisDiagnosticLog	It is used to cyclically write a series of parameter values belonging to an axis to a file. This output file is the basis for troubleshooting.

● PLCopen

Name	Function
SMC_ReadSetPosition	Read the current setting position of the drive unit.
SMC_SetTorque	It can be used to create a torque if the drive is in controller mode "torque".
SMC_CAMBounds	Calculate the maximum position, velocity and acceleration of the slave device. The device is connected to a master device in absolute mode. The master device moves at the specified maximum speed and acceleration/deceleration.
SMC_CAMBounds_Pos	Calculate the maximum and minimum positions of the slave device. The device is connected to a master device in absolute mode. The master device moves at the specified maximum speed and acceleration/deceleration.
SMC_CAMRegister	It represents a tappet control unit, which acts on the MC_CAM_REF structure like MC_CamIn, cancels the original track information and only reads the tappet information.
SMC_GetCamSlaveSetPosition	If the (slave) axis is connected to the motion of other (master) axes through CAM, the module calculates the current target position of the slave axis. So neither axis will be moved or affected.
SMC_GetTappetValue	It evaluates the output tappet of the MC_CamIn function block

	and contains the current tappet state.
MC_CamIn	Occupy CAM.
MC_CamOut	Immediately release the slave axis from the master axis.
MC_CamTableSelect	The CAM table is selected by setting the connection to the relevant table.
MC_GearIn	The CAM table is selected by setting the connection to the relevant table.
MC_GearInPos	The CAM table is selected by setting the connection to the relevant table.
MC_GearOut	It selects the CAM table by setting up the connection to the relevant table. It disengages the slave axis from the master axis.
MC_Phasing	Occupy CAM.
MC_AccelerationProfile	Controls a time-acceleration-locked motion distribution.
MC_Halt	Commands a controlled movement to stop. The function block will terminate the execution of all in-progress function blocks.
MC_Home	Its execution causes the axis to execute the "search home" sequence. The details of this sequence depend on the machining process and can be set by the parameters of the axis.
MC_MoveAbsolute	It is used to command controlled movement to a specified absolute position.
MC_MoveAdditive	Commands a controlled motion in a discrete motion state that adds the specified relative distance to the newly requested position.
MC_MoveRelative	It is used to command a controlled movement that, when executed, specifies a distance relative to the actual position of the axis.
MC_MoveSuperImposed	It is used to command a controlled movement that adds a specified relative distance to an existing movement. Existing movements are not interrupted, but additional movements are superimposed.
MC_MoveVelocity	It is used to command a never-ending controlled movement at a specific speed.
MC_PositionProfile	It is used to command a time-position locked motion profile.
MC_Power	It is used to control the power stage ("on" or "off").
MC_ReadActualPosition	It will return the actual position of the reference axis.
MC_ReadAxisError	It is used to describe common axis errors not related to function blocks.
MC_ReadBoolParameter	It returns a parameter of type BOOL specified by the manufacturer.
MC_ReadStatus	It returns the detailed status of the axis.
MC_ReadParameter	It returns vendor-specified parameters.
MC_Reset	It converts ErrorStop to StandStill state by resetting all internal errors about the axis - this affects the output of the function

	block instance.
MC_Stop	Its execution will cause the controlled motion to stop and set the axis to state "Stopping".
MC_VelocityProfile	It is used to command a time-velocity locked motion profile.
MC_WriteBoolParameter	It modifies factory-set Boolean-type parameters.
MC_WriteParameter	Modify the parameters set by the factory.
MC_AbortTrigger	It terminates the function block connected to the trigger event (eg MC_TouchProbe).
MC_DigitalCamSwitch	This function block is analogous to switching on a motor shaft: The function block commands a group of discrete output bits to switch in analogy to a set of mechanical cam controlled switches connected to an axis. Forward and backward movements are allowed
MC_ReadActualTorque	When Enable remains TRUE, the function block will return the actual torque or force value.
MC_ReadActualVelocity	When Enable remains TRUE, the function block will return the value of the actual speed.
MC_SetPosition	It is used to move the coordinate system of one axis.
MC_TouchProbe	It records the position of the axis when the trigger event occurs.
SMC_MoveContinuousAbsolute	It performs absolute movement, but unlike MC_MoveAbsolute, it does not reach the target position at velocity=0, but at the specified velocity.
SMC_MoveContinuousRelative	It performs relative movement, but unlike MC_MoveRelative, it does not reach the target position with velocity=0, but with the specified velocity.

● SimpleTest

Name	Function
SMC_StartupDrive	It contains a set of frequently used function blocks which are used to test and commission an axis.

SM3_CNC Library

The modules in the SM3_CNC.Library library are used (according to DIN 66025) to decode trajectories created in the CNC editor. Decoding is to transform the trajectory into a structure that can be used by IEC programs (trajectory preprocessing), or can be modified, interpolated, and transformed so that it is readable by the driving device (eg, SMC_NCDecoder, ToolCorr , SMC_AvoidLoop, SMC_SmoothPath, SMC_RoundPath, SMC_Interpolator).

Functional failure of modules allows certain components, such as decoders, to be replaced by modules designed for special needs. In addition, other modules such as certain trajectory preprocessing or optimization - can be added to the existing components without any problems, so that the compatibility between the components in the library does not need to be considered. Therefore, the program logic is completely handled by the PLC program, and only the pure action information is handled by the library module.

The data structures of the library SM3_CNC.Library (eg SMC_POSINFO, SMC_VECTOR3D) are used to describe positions, trajectory segments and vectors, and to manage GEOINFO objects (QUTQUEUE structures).

● Type of data

Name	Function
SMC_CNC_REF	This data structure manages the parsed G-code files.
SMC_GCODE_TEXT	The data source is connected to the output of the same name of the SMC_NCDecoder module. It is used to pass the decoded string of the last line of the CNC file to SMC_GCodeViewer.
SMC_GCODEVIEWER_DATA	The structure uses an array as the cache of the SMC_GCodeViewer function block, and a variable of this type stores the G code file corresponding to a curve track.
SMC_GCODE_WORD	It includes data for a certain G code word such as "N20" or "G1".
SMC_GEOINFO	This structure is designed to store track objects, that is, a segment of a programmed track, such as a line, an arc, or a spline segment.
SMC_M_PARAMETERS	It defines additional parameters for the currently active M-function, which can be counted by SMC_GetMParameters to display the additional value of each M-function.
SMC_OUTQUEUE	This structure is used to manage GEOINFO objects in a table of a defined size.
SMC_POSINFO	For a particular location point the structure describes its coordinates and the location of additional axes.
SMC_VECTOR3D	The structure describes the velocity in three dimensions.

● Global variable

Name	Function
SMC_AL_STATUS	Its value indicates the actual state of the FB instance.
SMC_DEC_STATUS	Its value indicates the actual state of the FB instance.
SMC_DIRECTION	Its value indicates the actual interpolation direction of the function block instance.
SMC_INT_STATUS	Its value indicates the actual state of the FB instance, possible states.
SMC_INT_VELMODE	Its value determines the velocity profile.
SMC_MOVTYP	It is a member of SMC_GEOINFO, which specifies the geometry of the object.
SMC_TC_STATUS	Its value represents the actual state of the FB instance.
SMC_TOOLCORRMODE	Its value represents the mode in which the SMC_Toolcorr runs.
SMC_VECTORDIR	It is used as the input variable of SMC_CalcDirectionFromVector to define the inclination, reverse or vertical of the velocity.

Function block record

●Inside the SMC_CNC_POUs\File folder

Name	Function

SMC_READNCFILE	This function block has been superseeded by SMC_ReadNCFile2 and is kept only for compatibility. It is recommended to use SMC_ReadNCFile2, which has additional features like mathematical expressions or sub programs in g-code
SMC_READNCQUEUE	This function block will read an SMC_OUTQUEUE file, which has been created by the CNC editor, from the PLC file system and provide an OutQueue structure, which typically is processed by the decoder.
SMC_VARLIST	The standard IEC 61131-3 provides no possibility to determine the value of a variable by its symbolic name represented as a string. This however is necessary, if the variable functionality, which is available for the user by SMC_CNC_REF, also should be available for reading in the CNC program from a file. This can be managed by using the structure SMC_VARLIST.

●SMC_CNC_POUs\SoftMotion CNC\Coordinate Transformation folder

Name	Function
SMC_DETERMINECUBOIDBEARING	This function block will determine the position of a cuboid (corner mark, edge alignment) in the space in dependence of 6 (3/2/1) points given
SMC_COORDINATETRANSFORMATIONS3D	It computes the coordinates of a point given in the old coordinate system and transformed to the new coordinate system. The coordinate transformation is defined by the transformation of the origin and the new unit vector (given in the old coordinate system).
SMC_INVCOORDINATETRANSFORMATION3D	This function block will calculate the coordinates of a point given in respect of the old coordinate system and transformed to the new coordinate system. For this purpose, the inverse coordinate transformation is defined by the translation of the origin and the new unit vectors (given in respect of the old coordinate system). (note: That this FB only works correct, if the vectors vX, vY and vZ are perpendicular to each other)
SMC_TEACHCOORDINATESYSTEM	A vector that assists the user in calculating the new coordinate system.
SMC_UNITVECTORTORPY	Calculate RPY-angle from the unit vector of the new coordinate system (with reference to the old coordinate system)
SMC_ROTATEQUEUE2D	When this module executes, the trajectory stored in poqDataIn will be rotated around the Z axis by the given angle dPhi[°]. A positive angle refers to a mathematically positive rotation (counterclockwise).
SMC_SCALEQUEUE3D	This function will stretch the path contained in poqDataIn by factor fScaleFactor. The input variable poqDataIn is a pointer to the structure SMC_OUTQUEUE , which describes the path to get

	scaled. At first, the initialization value FALSE of input variable bEnable will bar the scaling of the path. As soon as bEnable is set to TRUE all SMC_GEOINFO objects found in poqDataIn will be processed. When bEnable is reset to FALSE, the module will not execute any further modifications. The initialization of input variable bReset with FALSE effects, that the SMC_GEOINFO objects currently found in poqDataIn will not get rotated, but only those that will be added.
SMC_TRANSLATEQUEUE3D	This function will stretch the path contained in poqDataIn by factor fScaleFactor. The input variable poqDataIn is a pointer to the structure SMC_OUTQUEUE , which describes the path to get scaled. At first, the initialization value FALSE of input variable bEnable will bar the scaling of the path. As soon as bEnable is set to TRUE all SMC_GEOINFO objects found in poqDataIn will be processed. When bEnable is reset to FALSE, the module will not execute any further modifications. The initialization of input variable bReset with FALSE effects, that the SMC_GEOINFO objects currently found in poqDataIn will not get rotated, but only those that will be added.

● SMC_CNC_POUs\SoftMotion CNC\GCode Viewer folder

Name	Function
SMC_GCODEVIEWER	This function block produces a textual representation of the G-Code that has been processed by SMC_NCInterpreter or SMC_NCDecoder. This textual representation can be displayed in a visualization. With the input iActObjectSourceNo, typically connected with the SMC_Interpolator output iActObjectSourceNo, the function block gets the information which line has already been processed. It will remove the text of already processed lines.

● SMC_CNC_POUs\SoftMotion CNC\Posinfo Functions folder

Name	Function
SMC_POSINFO	This data type describes its coordinates and the positions of the additional axes for a particular position point.

● SMC_CNC_POUs\SoftMotion CNC\OutQueue Functions folder

Name	Function
SMC_APPENDOBJ	Returns the number of objects that are stored in the SMC_OUTQUEUE

SMC_DELETEOBJ	Removes an object from an SMC_OUTQUEUE. The n-th object will be deleted from the list (poq), counting starts with 0. If N-1 is greater than the number of SMC_GEOINFO objects stored in the list, nothing will happen and FALSE will be returned instead of TRUE.
SMC_GETCOUNT	Returns the number of objects that are stored in the SMC_OUTQUEUE (poq).
SMC_GETOBJ	Returns the n-th object of an SMC_OUTQUEUE. For N=0 the first element will be returned. If the number of elements is less than n+1, zero is returned.
SMC_GETOBJFROMEND	Returns a pointer to the n-last object (start counting from the end) of the queue (poq). For N=0 the last element in the list will be returned. task-safety: may be called from both tasks; the result however may be invalid: In case of the producer task, the output may no longer be part of the queue, in case of the consumer task, the output may no longer be the last element.
SMC_OUTQUEUEINIT	This function intiailizes the SMC_OUTQUEUE. task-safety: not task-safe; the application has to make sure that no other queue function is called while this function is active, and that no other queue function is active when this function is called.
SMC_RESTOREQUEUE	Resets the read position of the queue to zero. This function is no longer necessary. The interpolator will reset the read position when started.
SMC_SETQUEUECAPACITY	This function sets the variable nCapacity of the SMC_OUTQUEUE. task-safety: call from producer task only

● SMC_CNC_POUs\SoftMotion CNC\SoftMotion Function Blocks folder

Name	Function
SMC_AVOIDLOOP	This function block is used for trajectory preprocessing. It copies a track, but does not cut the loops it contains.
SMC_CHECKFORLIMITS	Used to check if a track leaves its specific rectangle bounds.
SMC_CHECKVELOCITIES	This module checks the orbital velocity of a particular trajectory segment.
SMC_EXTENDEDVELOCITY CHECKS	This function block is used to convert a continuous path described by SMC_GEOINFO objects into discrete path position points taking into account a defined velocity profile and time pattern. Afterwards, these position points will typically be transformed by the IEC-program (e.g. to drive-axis-positions) and sent to the drives.
SMC_INTERPOLATOR	This function block is used to transform a continuous trajectory described by the SMC_GEOINFO object into discrete trajectory position points, which take into account the velocity profile and time map.

SMC_INTERPOLATOR2DIR	This function block is similar to function block SMC_Interpolator , regarding function and allocation of inputs and outputs, but it is able to interpolate a path also in the opposite direction.
SMC_INTERPOLATOR2DIR_SLOWTASK	This FB is used for generating the reversed-path. It has been split off from SMC_Interpolator2Dir in order to be called from a task with lower priority.
SMC_LIMITDYNAMICS	The function block reduces the velocity and acceleration/deceleration of the path so that the resulting velocities, accelerations and decelerations of the cartesian and additional axes do not exceed the input values dMaxVel, dMaxAcc and dMaxDec.
SMC_LIMITCIRCULARVELOCITY	This function block limits the path velocities of circular elements depending on the radius. For a more general function block that limits the path velocity for all kinds of path elements, please see SMC_LimitDynamics. In order to limit the acceleration to a prescribed value, the velocity of the arc at the transition sqrt(a*r) must not be exceeded. The function block controls the transition between two elements (line on circular arc, circular arc on line or circular arc on circular arc) and adapts the end velocity of the first element so that the acceleration jump will not exceed the value dMaxAccJump. Furthermore, the function block limits the acceleration in X and Y to the value dMaxAcc
SMC_NCDECODER	This function block is used to convert a CNC program (Din 66025, G-code), into a list of SMC_GEOINFO objects. In each cycle, one line of the program is decoded
SMC_ROUNDPATH	The SMC_RoundPath function block is very similar to the SMC_SmoothPath module. It rounds edges, which arises at the junction of two objects, by circular arcs. The distance between the edge and the point at which the two adjacent objects are to be cut, is computed by half the length of the first object, half the length of the second object and the sum (from dRadius and D). From these three variables, the minimum is formed and assigned as cut point. Thus, the objects are cut by at most half the length of the shorter object. In addition, if the sum of input variable dRadius and G-code word is 0, then no rounding is done.
SMC_SMOOTHPATH	SMC_XInterpolator realizes a mixture of CAM and CNC. Imagine you want to cut a specified form (described by G-code) out of a workpiece`. The workpiece is moved - by another process - (e.g. along the X-axis) and the other axes (Y, Z, etc.) should be controlled according to the current position of the workpiece (X) and the target given by the

	path outline. The motion of the workpiece always follows the X-direction (other cases can be mapped on this by a rotation).
SMC_TOOLCORR	The SMC_ToolCorr is used for path preprocessing.
SMC_XINTERPOLATOR	SMC_XInterpolator realizes a mixture of CAM and CNC. Imagine you want to cut a specified form (described by G-code) out of a workpiece`. The workpiece is moved - by another process - (e.g. along the X-axis) and the other axes (Y, Z, etc.) should be controlled according to the current position of the workpiece (X) and the target given by the path outline. The motion of the workpiece always follows the X-direction (other cases can be mapped on this by a rotation).
SMC_GETMPARAMETERS	If the interpolator is waiting for an M-function, this function block can be used to query the parameters that have been set for this M-function in the G-Code (K, L, O words).
SMC_PREACKNOWLEDGEMFUNCTIONS	It is used to validate an M-function before the interpolator reaches the block. In this way, the pause on the M function can be avoided.

● SM_Trafo_POUs\Additional FBs folder

Name	Function
SMC_CALCDIRECTIONFROMVECTOR	The orientation of the tool (dAlpha) can be calculated when using it.

● SM_Trafo_POUs\Gantry Systems folder

Name	Function
SMC_TRAFO_GANTRY2	For the gantry system, no transformation can be performed, so the module can only add offsets to the x and y axes.
SMC_TRAFO_GANTRY2TOOL1	The function block is used to reverse the tool axial offset of the gantry system, that is, the axial offset relative to the Z axis, and they are approximately on a straight line.
SMC_TRAFO_GANTRY2TOOL2	The function block reverses the tool axial offset of the gantry system, that is, the axial offset relative to the Z axis, which is approximately at a right angle.
SMC_TRAFO_GANTRY3	For the gantry system, no transformation can be performed, so the module can only add offsets to the x, y, and z axes.
SMC_TRAFO_GANTRYCUTTER2	Computes an inverse transformation for a 3D gantry system being controlled and indicating an axis of rotation along the tangent of the path.
SMC_TRAFO_GANTRYCUTTER3	Computes an inverse transformation for a 3D gantry system being controlled and indicating an axis of rotation along the tangent of the path.
SMC_TRAFO_GANTRYH2	This transition module is used to provide reverse

	transitions to address H gantry systems with fixed drives.
SMC_TRAFOF_GANTRY2	The function module operates a three-dimensional gantry operating system without motion, therefore, the module only adds an offset in each of the X and Y axes. Each instance of the SMC_TRAFOF_GANTRY2 module can be linked to a visualization template named SMC_VISU_GANTRY2.
SMC_TRAFOF_GANTRY2TOOL1	The function block reversely finds the tool axial offset of the gantry system, that is, the axial offset relative to the Z axis, and they are approximately on a straight line.
SMC_TRAFOF_GANTRY2TOOL2	The function block reverses the tool axial offset of the gantry system, that is, the axial offset relative to the Z axis, which is approximately at a right angle.
SMC_TRAFOF_GANTRY3	The function module is used to operate a three-dimensional gantry operating system without motion. Therefore, this module only adds an offset to each of the X, Y, and Z axes. Each instance of the SMC_TRAFOF_GANTRY3 module can be linked to a visualization template named SMC_VISU_GANTRY3.
SMC_TRAFOF_GANTRYCUTTER2	This module will address the forward transition of a 3D gantry system with an axis of rotation pointing in a tangential direction.
SMC_TRAFOF_GANTRYCUTTER3	It will resolve the forward transition of the 3D gantry system with the axis of rotation pointing in the tangential direction of the current trajectory.
SMC_TRAFOF_GANTRYH2	This transition module is used to provide forward transitions to address H gantry systems with fixed drives.
SMC_TRAFOV_GANTRY2	An inverse transformation module is used to provide a two-dimensional gantry system that deals with the trajectory velocity and trajectory direction as a control variable for the axis.
SMC_TRAFOV_GANTRY3	An inverse transformation module is used to provide a three-dimensional gantry system that handles considering the trajectory velocity and trajectory direction as a control variable for the axis.
SMC_TRAFOV_GANTRYCUTTER2	The inverse transformation module is used to process the two-dimensional gantry system considering the trajectory velocity and trajectory direction as the control variables of the axes.
SMC_TRAFOV_GANTRYCUTTER3	An inverse transformation module is used to provide a three-dimensional gantry system that handles considering the trajectory velocity and trajectory direction as a control variable for the axis.
SMC_TRAFOV_GANTRYH2	The inverse transformation module is used to provide a two-dimensional H-gantry system that handles considering the trajectory velocity and trajectory direction as the control

	variables of the axes.

● SM_Trafo_POUs\Parallel Systems folder

Name	Function
SMC_TRAFO_BIPOD_ARM	The transformation module is provided to execute backward transformation of a bipod arm.
SMC_TRAFO_TRIPOD	The transformation module is provided to execute backward transformation of a tripod.
SMC_TRAFO_TRIPOD_ARM	The transformation module is provided to execute forward transformations of a tripod arm.
SMC_TRAFOF_BIPOD_ARM	The transformation module is provided to execute forward transformations of a bipod arm.
SMC_TRAFOF_TRIPOD	The transformation module is provided to execute forward transformations of a tripod.
SMC_TRAFOF_TRIPOD_ARM	The transformation module is provided to execute forward transformations of 3-jointed Scara systems

● SM_Trafo_POUs\Scara Systems folder

Name	Function
SMC_TRAFO_SCARA2	The transformation module is provided to execute backward transformations of 2-jointed Scara systems.
SMC_TRAFO_SCARA3	The transformation module is provided to execute backward transformations of 3-jointed Scara systems.
SMC_TRAFOF_SCARA2	Forward transformation for 2-dim. Scara system.
SMC_TRAFOF_SCARA3	The transformation module is provided to execute forward transformations of 3-jointed Scara systems

J Product Order Info.

Tablej-1 List of ordering information

Specification	Order NO.
C56-10 motion controller	
C56-10 control module, 32MB program data space, 64KB power-down retention data, 55M expansion bus, 1 RS485 communication port, MODBUS free port protocol, 1 EtherCAT interface, 1 EtherNET interface, 1 CAN port, 1 USB port, support SoftMotion	CTH3 C56-102S2
C57-10 motion controller	
C57-10 control module, 32MB program data space, 64KB power-down retention data, 55M expansion bus, 1 RS485 communication port, MODBUS free port protocol, 1 EtherCAT interface, 1 EtherNET interface, 1 CAN port, 1 USB port, support SoftMotion, CNC	CTH3 C57-102S2
Power module	
PWR-02 power module, input 85～264V AC, output 24V DC/2A	CTH3 PWR-020S1
Intermediate expansion module	

INT-00 intermediate expansion module	CTH3 INT-000S1
High-speed counting module	
HSC-02 high-speed counting module, 2-way differential/single-ended signal input	CTH3 HSC-020S1
Pulse output module	
HSP-04 pulse output module, 4-channel differential/single-ended signal output	CTH3 HSP-040S1
EtherCAT Slave Module	
EtherCAT slave module, up to 8 expansion modules can be attached	CTH3 ECT-000S1
Digital module	
DIT-08 digital module, digital input, 8*24VDC	CTH3 DIT-080S1
DIT-16 digital module, digital input, 6*24VDC	CTH3 DIT-160S1
DIT-32 digital module, digital input, 32*24VDC	CTH3 DIT-320S1
DQT-08 digital module, digital output, 8*24VDC	CTH3 DQT-080S1
DQT-16 digital module, digital output, 16*24VDC	CTH3 DQT-160S1
DQT-32 digital module, digital output, 32*24VDC	CTH3 DQT-320S1
DQR-08 digital module, digital output, 8*relay	CTH3 DQR-080S1
DQR-16 digital module, digital output, 16*relay	CTH3 DQR-160S1
Analog module	
AIS-04 analog module, analog voltage and current input, 4AI*12bit	CTH3 AIS-040S1
AMS-06 analog module, analog voltage and current input and output, 4AI x 12bit, 2AQ*12bit	CTH3 AMS-060S1
AIV-08 analog module, analog voltage input, 8AI*16bit	CTH3 AIV-080S1
AIC-08 analog module, analog current input, 8AI*16bit	CTH3 AIC-080S1
AQS-04 analog module, analog voltage and current output, 4AQ*12bit	CTH3 AQS-040S1
AQS-08 analog module, analog voltage and current output, 8AQ*12bit	CTH3 AQS-080S1
Temperature module	
AIT-04 temperature module, thermocouple input module, 4*TC, isolated 16bit accuracy	CTH3 AIT-040S1
AIT-08 temperature module, thermocouple input module, 8*TC, isolated 16bit accuracy	CTH3 AIT-080S1
AIR-04 temperature module, thermal resistance input module, 4*RTD, isolated 16bit accuracy	CTH3 AIR-040S1
AIR-08 temperature module, thermal resistance input module, 8*RTD, isolated 16bit accuracy	CTH3 AIR-080S1
Accessories	
PLC lithium battery, voltage 3.6V, capacity 2700mAh	CTH3 BAT-000S1

Service Hotline

+86 400-700-4858

深圳市合信自动化技术有限公司
COTRUST TECHNOLOGIES CO., LTD.

Address: Address: 9/F, Block A Building 6, Shenzhen International Innovation Valley, Dashi 1st Road, Nanshan District, Shenzhen 518055,China

0755-86226822

market@co-trust.com

https://www.co-trust.com

Official WeChat account