RM Series Medium-sized PLC Controller User Manual

Shenzhen Rtelligent Technology Co., Ltd

Preface

Thank you for purchasing and using the RM series medium-sized PLC controller of Rtelligent.

The RM series controller is a medium-sized programmable logic controller developed by Rtelligent. It supports functions such as logic control and motion control. With the CODESYS 3.5 SP19 programming environment, support logic control and motion control functions, the process can be encapsulated and reused through FB/FC functions. Multi-layer network communication can be achieved through RS485, Ethernet, EtherCAT and CANOpen interfaces. The PLC integrates digital input and digital output functions, and supports the expansion of 8 Rtelligent IO modules.

You can choose the model that supports CODESYS softmotion function and high-speed pulse input/output according to your own needs. The RM series controller currently has three models: RM510, RM518, and RM418. Please refer to section 2.2 of this manual for specific model specifications for selection.

This manual is a comprehensive guide to the use of Rtelligent RM series controller. It describes the installation and wiring of the products, including product information, mechanical installation, electrical installation, etc. Before using this product, please read the manual carefully and perform wiring and programming debugging under the premise of fully understanding the content of the manual. Only operators with certain electrical knowledge can wire and programming debugging operations of this product. When using this product, please first confirm whether it meets the requirements and is safe. If you do not understand the use of the place, please consult our technical personnel for assistance.

Due to continuous improvements in PLC controllers, any changes to the information provided by our company will not be separately notified.

Revision History

Date	Version	Description
2024.03	V1.01	Initial issue
2024.12.17	V2.00	Add model descriptions for RM518 and RM418

Contents

1 Safety Instructions

1.1 Safety Precautions

- Be sure to disconnect all external power supplies before installing the controller. Otherwise, there is a risk of electric shock.
- After powering on the controller, do not touch the terminal, do not wire and unwire the terminal. Otherwise, there is a risk of electric shock.
- Please install and use the product under the environmental conditions specified in the specification in the manual. Do not use it in humidity, high temperature, dust, smoke, conductive dust, corrosive gas, combustible gas, and in places with vibration and impact. Otherwise, it may cause electric shock, fire, misaction, product damage, etc.
- Please design the security loop of the controller to ensure that the whole system can operate safely when the controller runs abnormally. Otherwise, there is the risk of misaction and failure.
- Please connect the DC 24V power supply to the dedicated power terminal of the controller. The wrong power supply may burn the controller.
- Do not tie the control wiring and the power wiring together, in principle, it should be 10cm apart. Otherwise, it may cause misaction and product damage.
- Do not directly touch the conductive parts of the product. Otherwise, it may cause misaction and failure.
- Please secure the product with DIN46277 rails or M3 screws and install it on a flat surface. The wrong installation may cause misoperation and product damage.
- When processing the screw hole, please do not make the cutting powder and wire debris fall into the product shell. Otherwise, it may cause misaction and failure.
- When connecting or removing peripheral equipment, expansion equipment, battery and other equipment, be sure to power off. Otherwise, it may cause misaction and failure.
- Please use a $2mm^2$ wire to ground the ground terminal of the controller for a third type, not common grounding with the strong electric system. Otherwise, it may cause failure, product damage, etc.
- When using wires to connect terminals, be careful to tighten and do not contact the conductive part to other wires or terminals. Otherwise, it may cause misaction and product damage.
- Be sure to STOP before making changes to the program in the controller. Otherwise, it may cause misaction.
- Do not disassemble or assemble this product without authorization. Otherwise, it may cause damage to the product.
- Please plug and plug the connection cable in case of power failure. Otherwise, it may cause damage to the cable and cause misoperation.
- Please never modify this product, or it may cause injury or mechanical damage.
- When the products are discarded, please treat the industrial waste or the local environmental protection regulations.

1.2 Unpacking Inspection

Items	Description
Check whether the delivered products comply with you ordered.	The packaging box contains the products you ordered. Please confirm through the nameplate model of the controller.
Check whether the products are intact.	Please check the product surface to see if the product is damaged during transportation. If any omission or damage is found, please contact our company or your supplier as soon as possible.

2 Product Information

2.1 Product Naming

RM 518 - 1616 T

① ② ③ ④

Symbol	Description
①	**Series name** RM: RM series medium-sized programmable logic controller
②	**Type code** 5: Medium-sized 500 series, support Softmotion function; 4: Medium-sized 400 series, not support Softmotion function 1: EtherCAT bus type 8: with 8-axis performance, 0 indicates not support pulse axis
③	**Input/output points** 1616: Built-in IO 16 points input and 16 points output
④	Module classification R: Relay output T: Transistor NPN output P: Transistor PNP output

2.2 Specification Description

Model	Description
RM510-1616T	Medium-sized EtherCAT bus type programmable logic controller, support EtherCAT bus 1ms cycle 8-axis synchronization, support CODESYS Softmotion, support EtherNET, CANOPEN master station, 2 RS485, built-in 16 points input 16 points transistor NPN output, can expand 8 Rtelligent IO modules.
RM518-1616T	Medium-sized EtherCAT bus type programmable logic controller, support EtherCAT bus 1ms cycle 8-axis synchronization, support CODESYS Softmotion, support EtherNET, CANOPEN master station, 2 RS485, built-in 16 points input 16 points transistor NPN output, 8 high-speed pulse output, 8 high-speed counting, can expand 8 Rtelligent IO modules.
RM418-1616T	Medium-sized EtherCAT bus type programmable logic controller, support EtherNET, EtherCAT master station, CANOPEN master station, 2 RS485, built-in 16 points input 16 points transistor NPN output, 8 high-speed pulse output, 8 high-speed counting, can expand 8 Rtelligent IO modules.

2.3 Product Structure

2.3.1 Front View

2.3.2 Side View

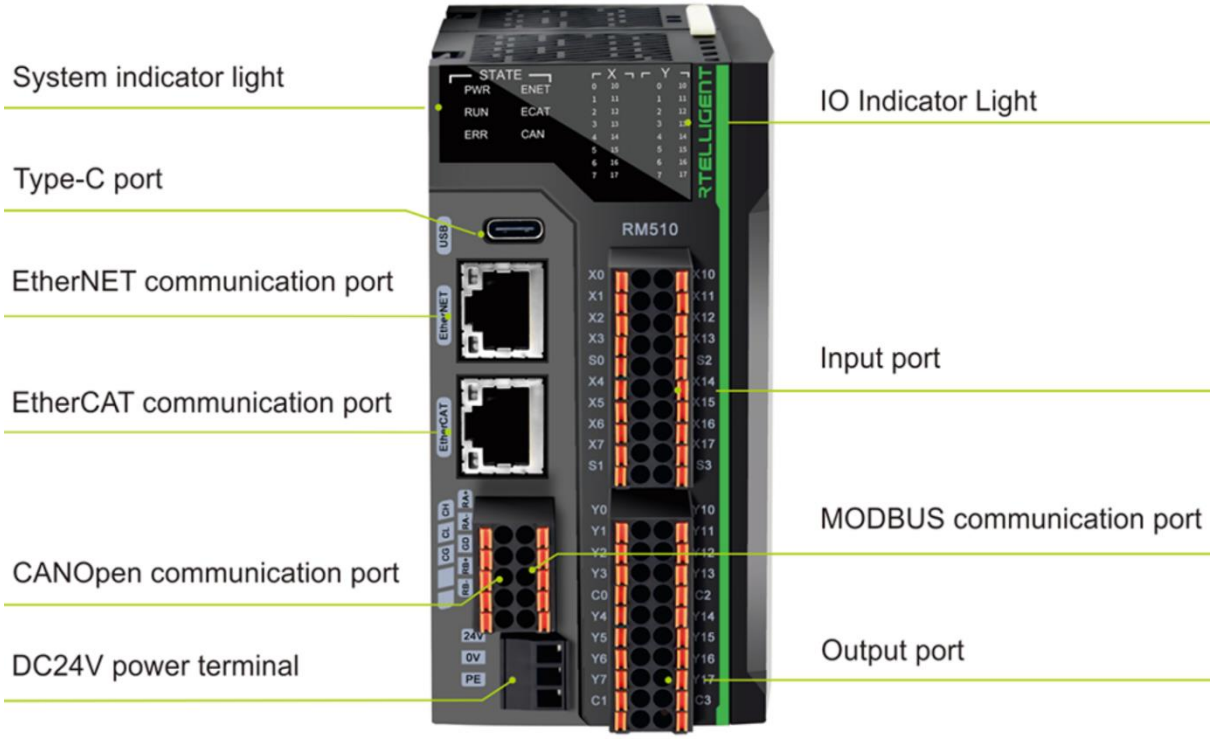
3 Specification Parameters

3.1 General Specifications

Items	Standards and specifications
Working conditions	
Temperature	0℃~55℃
Humidity	5%~95% (No condensation)
Altitude	-1000m~+2000m
Air	Dust-proof, non-corrosive, low salt spray, humid, dusty environments, SO2<0.5ppm, relative humidity<60%, no condensation H2S<0.1ppm, relative humidity<60%, no condensation
Isolation Voltage /High voltage insulation resistance	DC 500V with insulation resistance above 2MΩ/ End to ground 2200VDC, I/O interface end to other end 1500VAC, for 1 minute
Storage temperature	
Temperature	-25~70℃
Electromagnetic compatibility - immunity	
Resistance to electrical interference	Pulse width: 50ns, repetition frequency: 5kHz, 2000V voltage peak / Noise Voltage: 1000Vp-p, 1μs Pulse, 1 Minute
Electrostatic discharge (ESD) IEC61000-4-2	Contact discharge: ±4KV Air discharge: ±8KV
Electrical fast transient (EFT) IEC61000-4-4	Power cable: 2KV，5KHz Signal cable: 2KV，5KHz (I/O coupling clamp) 1KV，5KHz (communication coupling clamp)
Vibration resistance	Frequency: 10~57Hz; amplitude: 0.1mm; frequency: 57~150Hz; acceleration: 1.0g; 3D: Ten times in each direction
Impact resistance	15g, lasting 11ms, three shocks in three dimensions
Ground (FG)	The third type of grounding (cannot be connected to a common ground with a strong electrical system)

3.2 Port Descriptions

3.2.1 Port Diagram



3.2.2 Port Definition

No.	Port type	Label	Definition	Indicator color	Description
1	Running status light	PWR	Power normal	Yellow&Green	● Steady on: The power supply is normal ● Off: The power supply is abnormal
		RUN	Run normal	Yellow&Green	● Steady on: The user program is running ● Off: The user program stops
		ERR	Run error	Red	● Off: No serious error occurs ● Blinking: A serious error occurs
		ENET	EtherNet communication status indicator	Yellow&Green	● Steady on: The link is successfully established ● Off: The link is not established
		ECAT	EtherCAT communication status indicator	Yellow&Green	● Steady on: The link is successfully established ● Off: The link is not established
		CAN	CAN communication status	Yellow&Green	● Steady on: The link is

			indicator		successfully established ● Off: The link is not established
2	IO indicator light	IN/OUT	IO status display	Yellow&Green	● Steady on: The input or output is valid ● Off: The input or output is invalid
3	DIP switch	RUN/STOP	Control host running/stopping	-	-
4	Type-C port	-	For USB expansion	-	Use USB flash disk with type-c port
5	SD card slot	SD	For SD storage expansion, for storing user programs and user data	-	Use a micro SD card
6	RS485 PORT1	485A+	485 Communication signal, +	-	-
		485A-	485 Communication signal, +	-	-
		GND	485 Commuication, GND	-	RS485 PORT1 and PORT2 share the port
	RS485 PORT2	485B+	485 Communication signal, +	-	-
		485B-	485 Communication signal, -	-	-
	CAN	CAN L	CAN Low	-	-
		CAN H	CAN High	-	-
		CGND	CAN communicate, GND	-	-
7	Ethernet port	EtherNet	Ethernet communication RJ45 port	-	-
8	EtherCAT port	EtherCAT	For EtherCAT communication	-	-
9	Power port	+24V	DC 24V power supply, positive	-	-
		0V	DC 24V power supply, negative	-	-
		PE	PE	-	-
10	IO input terminal	-	16-channel input	-	Section 3.6.3 defines this in detail
11	IO output terminal	-	16-channel output	-	Section 3.6.4 defines this in detail

3.3 Performance Specifications

Items		Specifications
Basic items	Program capacity	20M Bytes
	Data capacity	20Mbyte, among them, RM510 supports 4K bytes power-off hold, while RM518/418 supports 16K bytes power-off hold
	Zone X (%I)	128Byte
	Zone Y (%Q)	128Byte
	Zone M (%M)	128KByte
	Axis performance	1ms cycle 8-axis synchronization (execution time of motion control calculation)
	Electronic CAM, interpolation	Supports
	Local expansion I/O module	Supports up to 8 local expansion modules
	Real-time clock	Button battery retention (can be replaced by oneself)
Programme	Programming platform	CODESYS V3.5 SP19
	Programming language	IEC 61131-3 programming language（LD、ST、SFC、CFC）
Communication	EtherCAT	● Transmission speed 100Mbps (100Base-TX) ● Supports protocol, EtherCAT master ● Supports up to 128 EtherCAT slave stations. Minimum synchronization period: 500μs ● Slave station supports disabling and scanning
	EtherNet	● Transmission speed 100Mbps (100Base-TX) ● Supports Modbus-TCP master/slave station: As the master station, supports up to 63 slave stations. As the slave station, supports up to 16 master stations ● TCP/UDP free protocol, supports up to 16 connections ● Socket, maximum number of connections: 4. TCP/UDP is supported ● IP address Initial value: 192.168.1.3

	CAN	● Communication baud rate: 125000bit/s, 250000bit/s, 500000bit/s, 800000bit/s, 1000000bit/s ● Supports the CANOPEN protocol ● Terminal resistance, built-in 120Ω ● Maximum transmission distance: 100m (125,000 bit/s)
	RS485	● Supported channels: 2 ● Isolation mode: No isolation ● Can be used as Modbus master or slave (ASCII/RTU) ● Number of Modbus-RTU slave stations: Supports up to 31 Modbus-RTU slave stations ● Communication baud rate: 9600bit/s, 19200bit/s, 38400bit/s, 57600bit/s, 115200bit/s ● Supports serial port free protocol ● Terminal resistance, external 120Ω ● Maximum transmission distance: 500m (9600bit/s)
	USB	● USB cable distance: 1.5m ● USB communication version: USB2.0, full speed ● USB interface: Type-C ● Master/slave: Only master, not slave
User program upgrade	EtherNet	Supports EtherNet monitoring PLC, upload & download user programs
	TF card	Downloading user programs through storage expansion cards is not supported
	Type-C	It does not support Type-C to monitor PLC, upload or download user programs

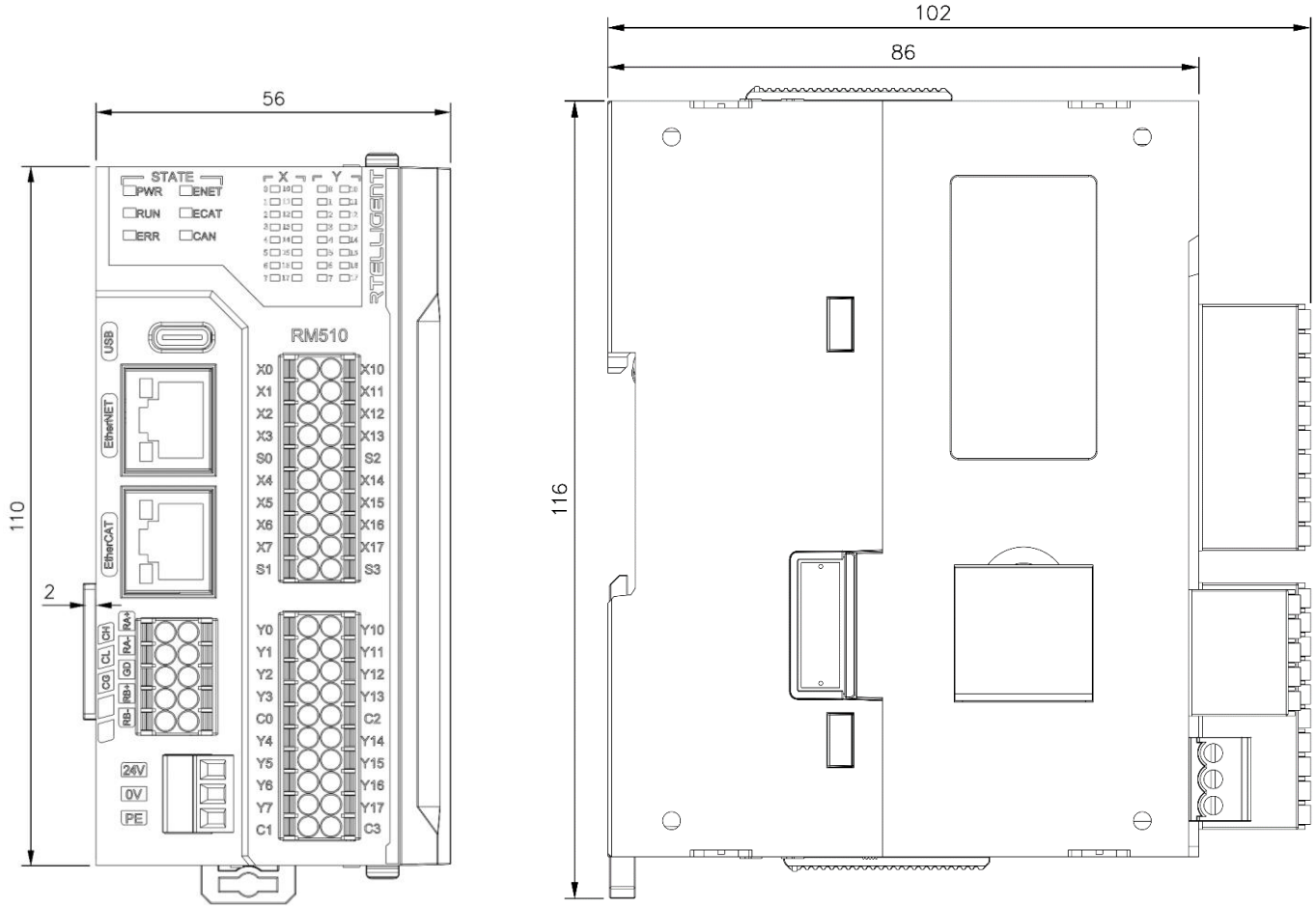
3.4 Electrical Parameters

Items	Electrical parameters
Power input voltage	DC 24V
Allowed power supply voltage fluctuation range	DC20.4V~28.8V (-15%~+20%)
24V input power protection	Supports short-circuit protection and reverse circuit protection
Host switch input	
Digital input points	16-point bipolar input
Isolation method	Photocoupling
Input impedance	2.4KΩ
Input is ON	The input current of the high-speed input is greater than 5.8mA/24V, and the input current of the common input is greater than 9.9mA/24V
Input is OFF	The input current of the high-speed input is less than 4.5mA/19V, and the input current of the ordinary input is less than 4mA/17V
Filter function	With filter function: ● Filter parameters: 1ms ~ 1000ms
High-speed pulse counting function	Without
Enter the common terminal mode	4 points/common terminal (polarity+/- of input power can be changed)
Input level	Sink type/source type, S/S connected to 24V is NPN, S/S connected to GND is PNP.
Isolation	Isolation between on-site and logical sides
Host switch output	
Digital output points	16 points NPN output
Maximum allowable current	0.5A/point
Loop supply voltage	DC 24V
Circuit insulation	Photoinsulation
ON response time	0.5ms
Output common terminal mode	4 points/common terminal (Polarity of output power: -)
Output level	Lower level NPN，COM is connected to negative
Short-circuit protection	Each circuit supports short-circuit protection and recovers after power failure and restart

3.5 Installation Specifications

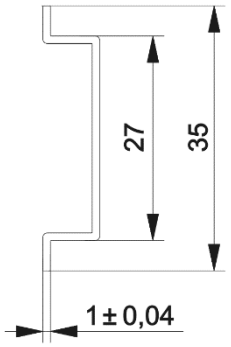
3.5.1 Installation Dimensions

The installation dimensions are shown in the following figure (unit: mm).



3.5.2 Installation Method

The DIN rail shall be installed according to IEC 60715 standard (35mm wide, 1mm thick). The size information is shown below in mm (unit: mm).



3.6 Wiring Specifications

3.6.1 Power Supply Wiring

Terminal number	Power supply wiring
1	DC 24V power supply positive
2	DC 24V power supply negative
3	PE

3.6.2 I/O Input Signal Definition

Left signal	Left terminal	Right terminal	Right signal
X0 signal input	1A	1B	X10 signal input
X1 signal input	2A	2B	X11 signal input
X2 signal input	3A	3B	X12 signal input
X3 signal input	4A	4B	X13 signal input
Input common SS0	5A	5B	Input common SS2
X4 signal input	6A	6B	X14 signal input
X5 signal input	7A	7B	X15 signal input
X6 signal input	8A	8B	X16 signal input
X7 signal input	9A	9B	X17 signal input
Input common SS1	10A	10B	Input common SS3

3.6.3 High-speed Input Interface Multiplexing Function List

Left terminal	Multiplex signal 1	Multiplex signal 2	Multiplex signal 3	Multiplex signal 4
1A	X0 signal input	Single phase high-speed count 0	AB phase counter 0, A phase	---
2A	X1 signal input	---	AB phase counter 0, B phase	---
3A	X2 signal input	Single phase high-speed count 1	AB phase counter 1, A phase	---
4A	X3 signal input	---	AB phase counter 1, B phase	---
5A	Input common SS0	---	---	---
6A	X4 signal input	Single phase high-speed count 2	AB phase counter 2, A phase	---
7A	X5 signal input	---	AB phase counter 2, B phase	---
8A	X6 signal input	Single phase high-speed count 3	AB phase counter 3, A phase	---
9A	X7 signal input	---	AB phase counter 3, B phase	---
10A	Input common SS1	---	---	---

Right terminal	Multiplex signal 1	Multiplex signal 2	Multiplex signal 3	Multiplex signal 4
1B	X10 signal input	Single phase high-speed count 4	AB phase counter 4, A phase	---
2B	X11 signal input	---	AB phase counter 4, B phase	---
3B	X12 signal input	Single phase high-speed count 5	AB phase counter 5, A phase	---
4B	X13 signal input	---	AB phase counter 5, B phase	---
5B	Input common SS2	---	---	---
6B	X14 signal input	Single phase high-speed count 6	AB phase counter 6, A phase	AB phase counter 0, Z phase
7B	X15 signal input	---	AB phase counter 6, B phase	AB phase counter 1, Z phase
8B	X16 signal input	Single phase high-speed count 7	AB phase counter 7, A phase	AB phase counter 2, Z phase
9B	X17 signal input	---	AB phase counter 7, B phase	AB phase counter 3, Z phase
10B	Input common SS3	---	---	---

3.6.4 I/O Output Signal Definition

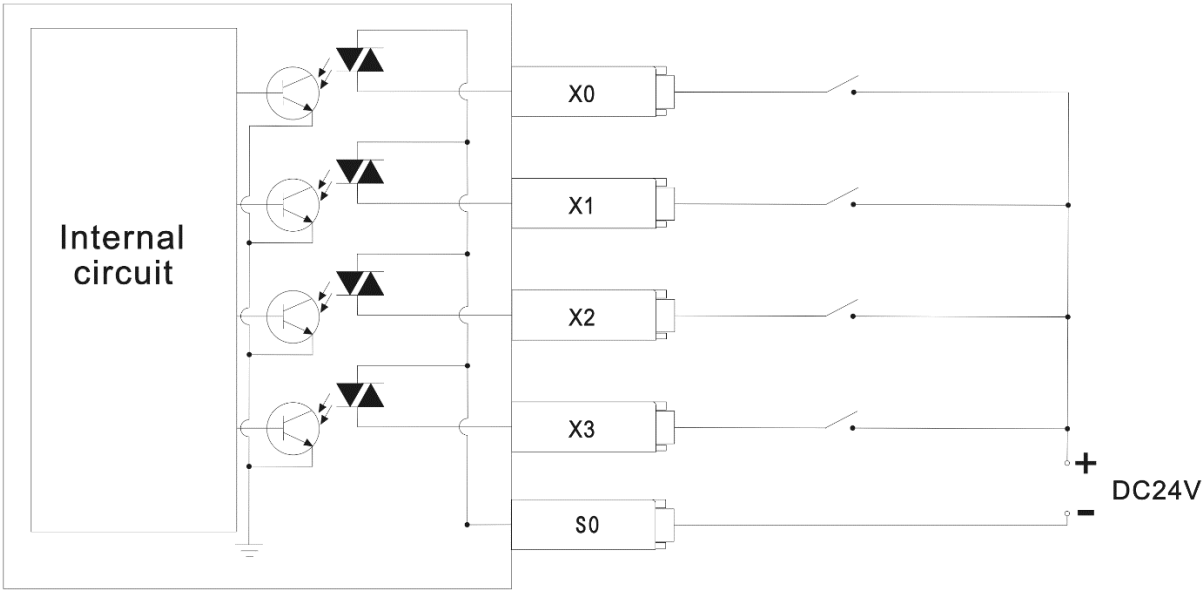
Left signal	Left terminal	Right terminal	Right signal
Y0 signal output	1A	1B	Y10 signal output
Y1 signal output	2A	2B	Y11 signal output
Y2 signal output	3A	3B	Y12 signal output
Y3 signal output	4A	4B	Y13 signal output
Output common COM0	5A	5B	Output common COM2
Y4 signal output	6A	6B	Y14 signal output
Y5 signal output	7A	7B	Y15 signal output
Y6 signal output	8A	8B	Y16 signal output
Y7 signal output	9A	9B	Y17 signal output
Output common COM1	10A	10B	Output common COM3

3.6.5 High-speed Output Interface Multiplexing Function List

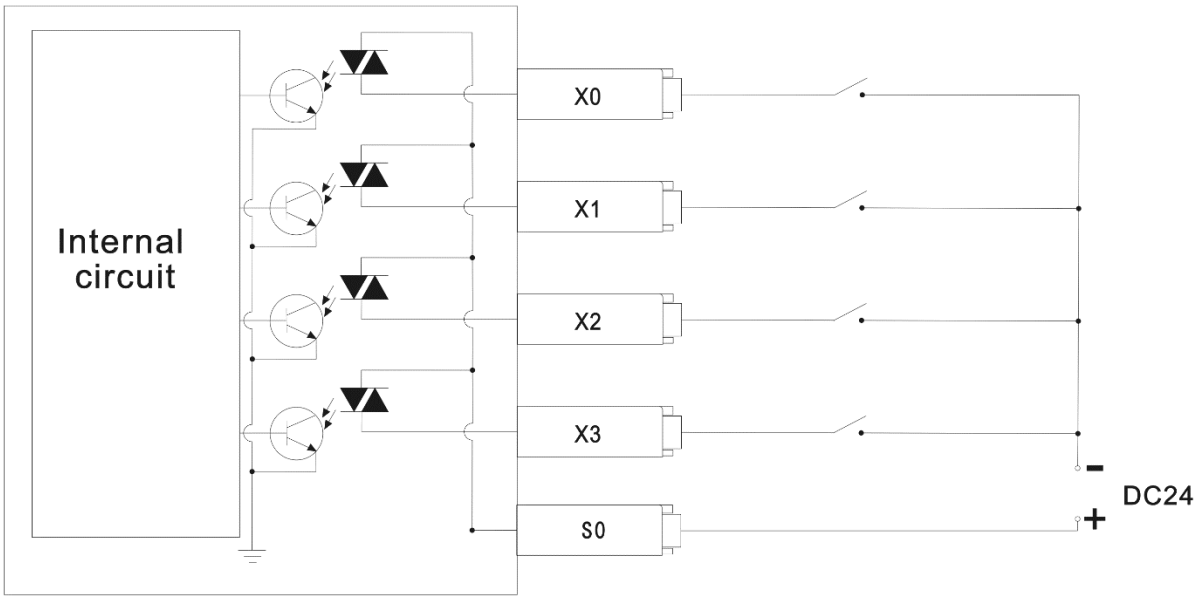
Left signal	Multiplex signal 1	Multiplex signal 2	Multiplex signal 3
1A	Y0 signal output	Axis 0 PULSE output	Axis 0 CW output
2A	Y1 signal output	Axis 0 DIR output	Axis 0 CCW output
3A	Y2 signal output	Axis 1 PULSE output	Axis 1 CW output
4A	Y3 signal output	Axis 1 DIR output	Axis 1 CCW output
5A	Output common COM0	---	---
6A	Y4 signal output	Axis 2 PULSE output	Axis 2 CW output
7A	Y5 signal output	Axis 2 DIR output	Axis 2 CCW output
8A	Y6 signal output	Axis 3 PULSE output	Axis 3 CW output
9A	Y7 signal output	Axis 3 DIR output	Axis 3 CCW output
10A	Output common COM1	---	---

Right signal	Multiplex signal 1	Multiplex signal 2	Multiplex signal 3
1B	Y10 signal output	Axis 4 PULSE output	Axis 4 CW output
2B	Y11 signal output	Axis 4 DIR output	Axis 4 CCW output
3B	Y12 signal output	Axis 5 PULSE output	Axis 5 CW output
4B	Y13 signal output	Axis 5 DIR output	Axis 5 CCW output
5B	Output common COM2	---	---
6B	Y14 signal output	Axis 6 PULSE output	Axis 6 CW output
7B	Y15 signal output	Axis 6 DIR output	Axis 6 CCW output
8B	Y16 signal output	Axis 7 PULSE output	Axis 7 CW output
9B	Y17 signal output	Axis 7 DIR output	Axis 7 CCW output
10B	Output common COM3	---	---

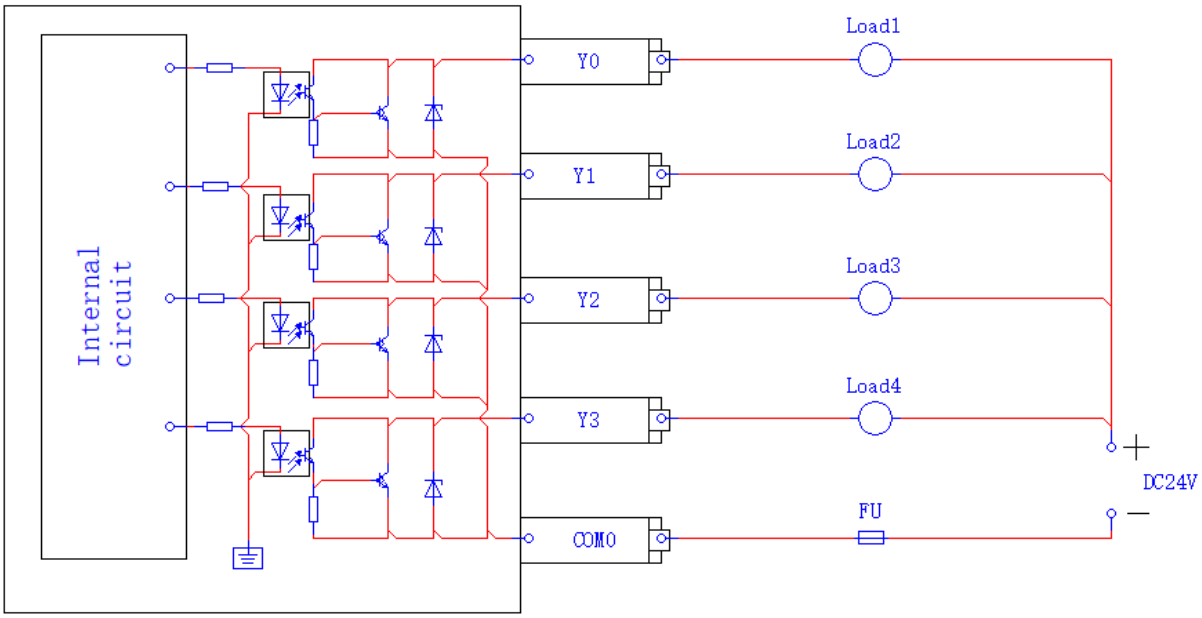
3.6.6 Sink Type Digital Input Wiring



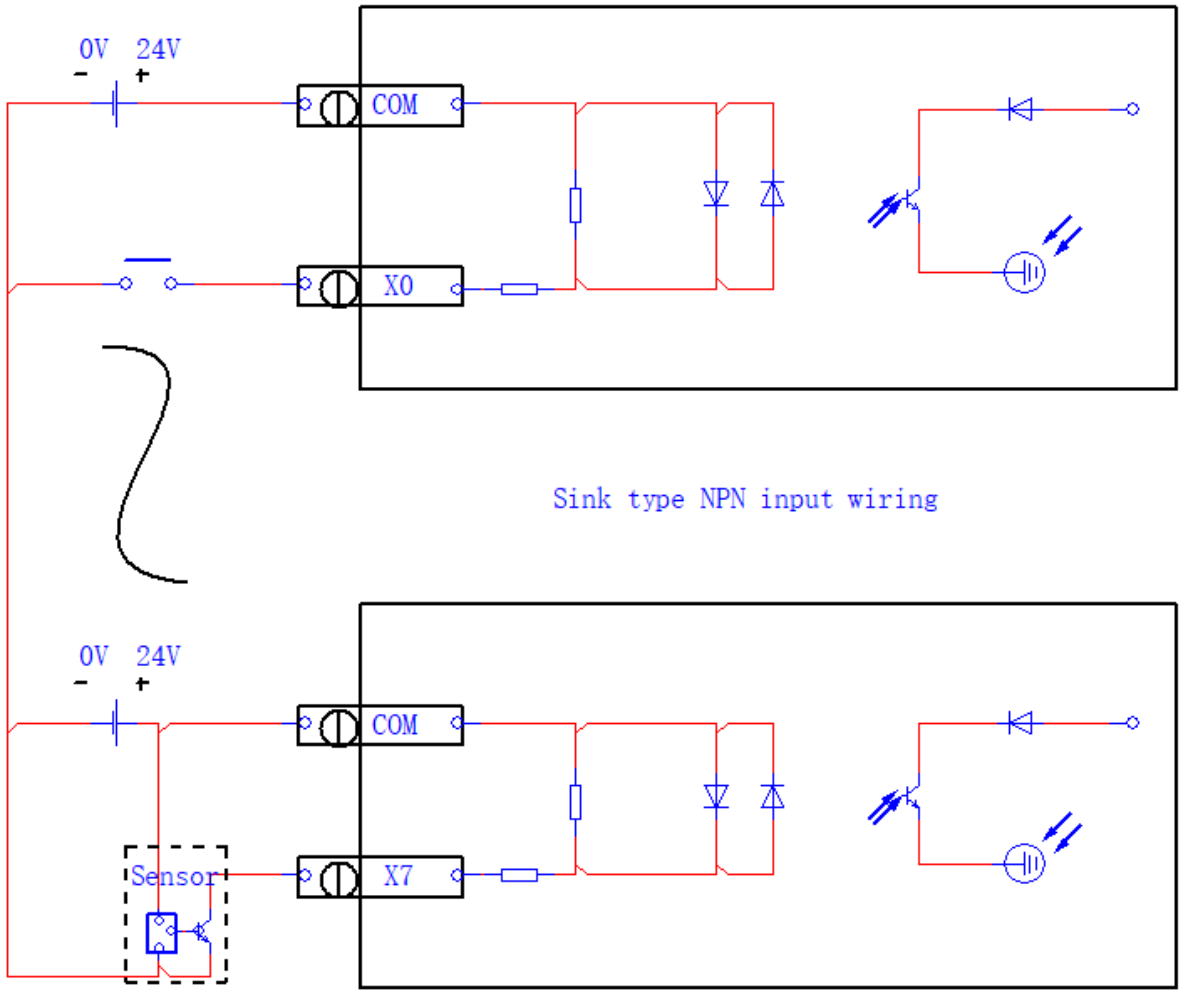
3.6.7 Source Type Digital Input Wiring

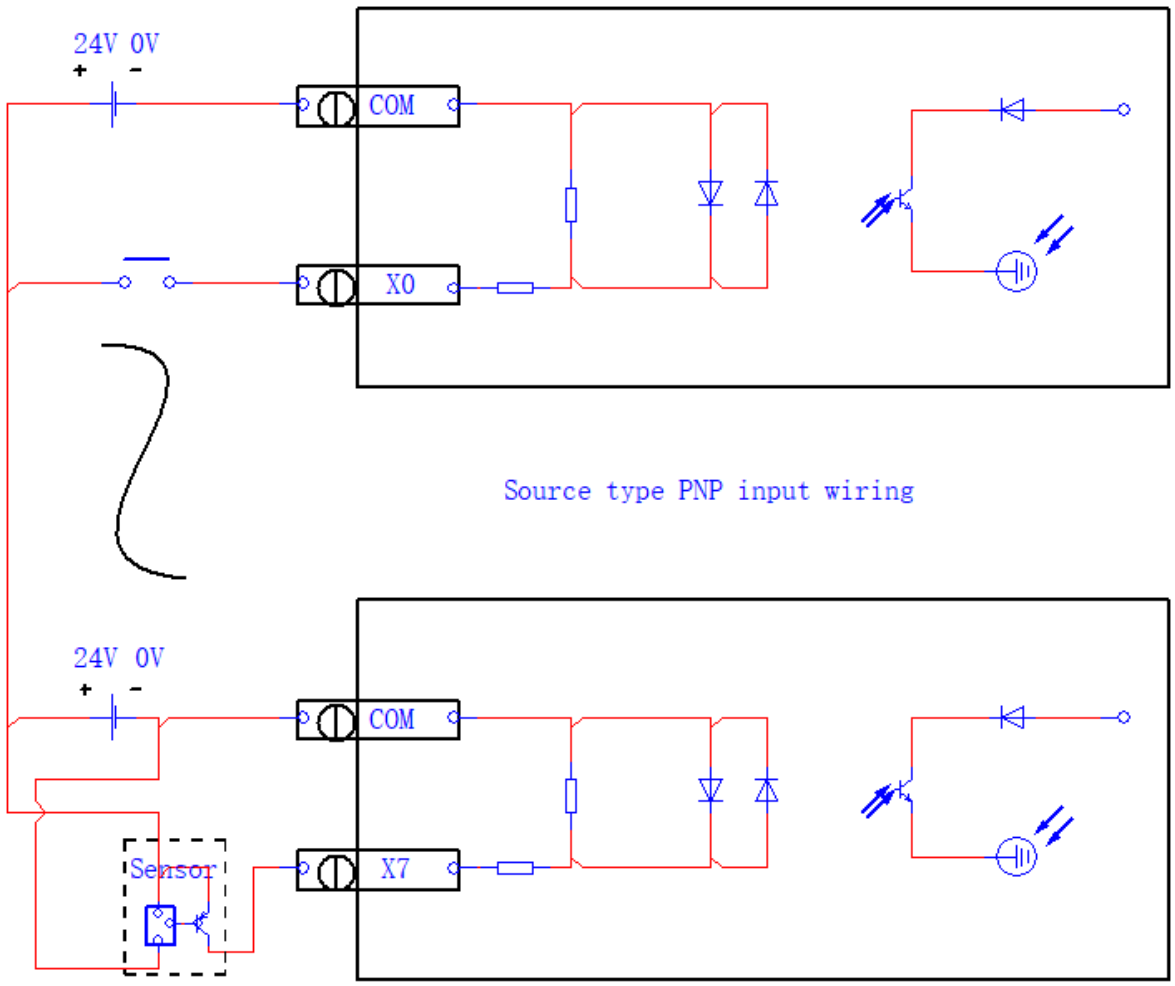


3.6.8 Digital Output Wiring

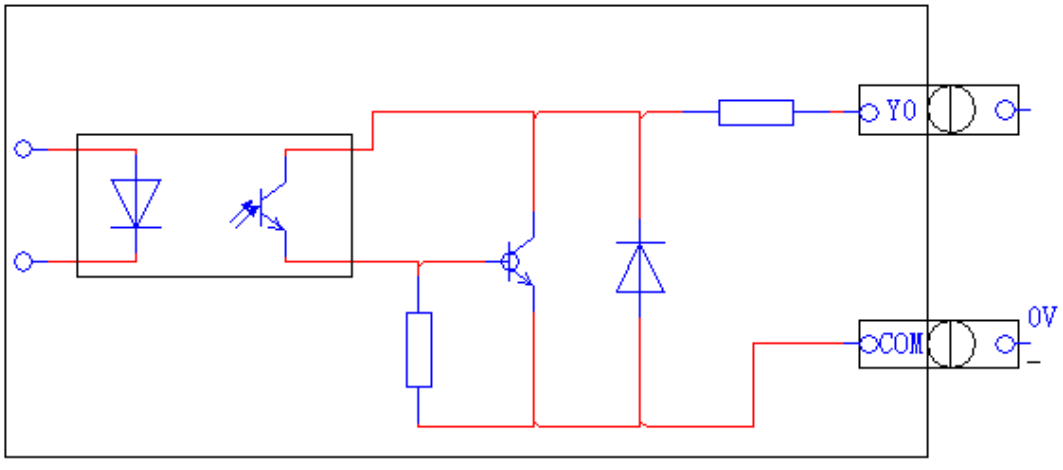


3.6.9 Digital Input Internal Diagram





3.6.10 Digital Output Internal Diagram



Source type PNP output wiring

3.6.11 RS485 & CAN Terminal Signal Definition

Left terminal	Left signal	Right signal	Right terminal
1A	CAN-H	485A+	1B
2A	CAN-L	485A-	2B
3A	CGND	GND	3B
4A	Reserved	485B+	4B
5A	Reserved	485B-	5B

3.7 Communication Specifications

3.7.1 EtherCAT Communication Specifications

Items	Descriptions
Communication protocol	EtherCAT protocol
Support services	CoE（PDO、SDO）
Synchronization method	DC-distributed clock
Physical layer	100Mbit/s (100Base-TX)
Duplex mode	Full duplex
Topological structure	Linear topology structure
Transmission media	AWG26 Type 5 shielded twisted-pair cable
Transmission distance	The distance between two nodes is less than 100m
Slave station number	Up to 128 can be carried
EtherCAT frame length	44 bytes to 1498 bytes
Process data	A maximum of 1486 bytes per Ethernet frame

4 Operation & Debugging & Maintenance

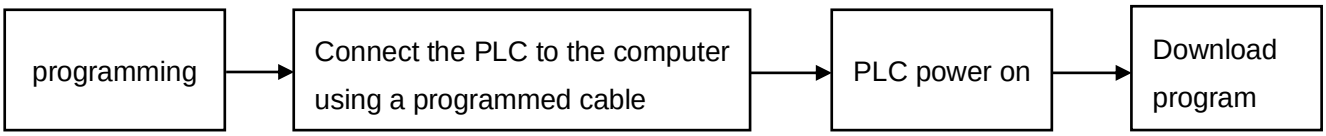
4.1 Operation & Debugging

4.1.1 Product Inspection

After getting the product, please first check whether the input and output terminals of the product are intact and whether parts are missing. Generally speaking, the PLC can be directly connected to the power cord for power-on check, and the PWR and RUN indicators should be steady on.

4.1.2 Programming and Downloading

After confirming that the product is in good condition, you can write the program for the PLC, and the program is written in the personal computer. The completed program can be downloaded to the PLC. The general operating steps are as follows:



4.1.3 Program Debugging

Ideally, the PLC is in normal operation, but if it is found that the program in the PLC is wrong and needs to be modified, it is necessary to re-write the program to the PLC in operation.

(1) Use programming cables to connect PLC and computer;
(2) Upload program from PLC;
(3) Modify the uploaded program. It is recommended to save the modified program as a separate file;
(4) Pause the operation of the PLC and download the modified program to the PLC;
(5) Monitor the PLC;
(6) If the requirements are still not met, you can continue to modify the program and download it to the PLC until the requirements are met.

4.1.4 PLC Indicator Light

- When the PLC is running normally, the PWR and RUN indicators should be steady on.
- When the indicator ERR is steady on, it indicates that there is a problem in the operation of the PLC program, please correct the program in time.
- If the indicator light PWR is off, there is a problem with the power supply and the power wiring should be checked.

4.2 Routine Maintenance

4.2.1 Regular Inspection of Products

- Although the programmable controller has a certain anti-interference and strong stability, it should also develop the habit of checking and maintaining the controller regularly. The inspection items include:
- Check if the input/output terminals and power terminals of the PLC are loose or not secure;
- Whether the communication port is in good condition; Check whether the power indicator and input/output indicator are on.
- Clean the accumulated dust outside the PLC to prevent dust and conductive dust from falling into the interior of the PLC;
- Try to ensure that the operation and storage environment of the PLC complies with the standards described in Section 3-1 of this manual.

4.2.2 About Battery

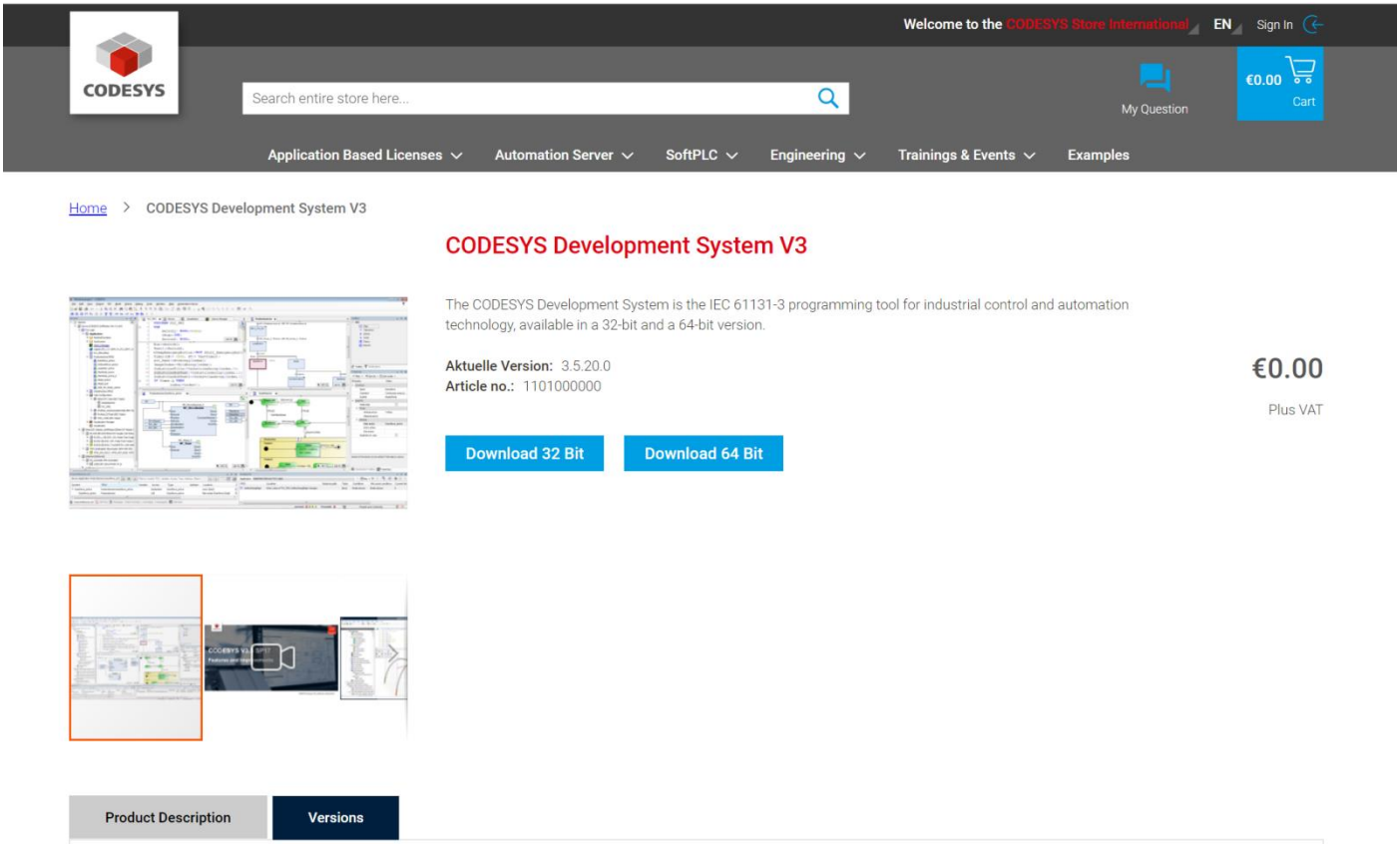
There are no components in the programmable controller that seriously shorten its life, and it can be used forever. However, if the real-time clock (RTC) function of the PLC is used, the battery needs to be replaced regularly.

- Battery life is generally 3 to 5 years.
- Replace the battery as soon as possible if the battery level drops.
- After the battery is replaced, power on the PLC as soon as possible; otherwise, the battery may run out.

5 Codesys Software Installation

5.1 Download and Install the Codesys Installation Package

(1) Find the codesys official website (https://store.codesys.com/en/codesys.html), as shown below, and download and install the codesys installation package.

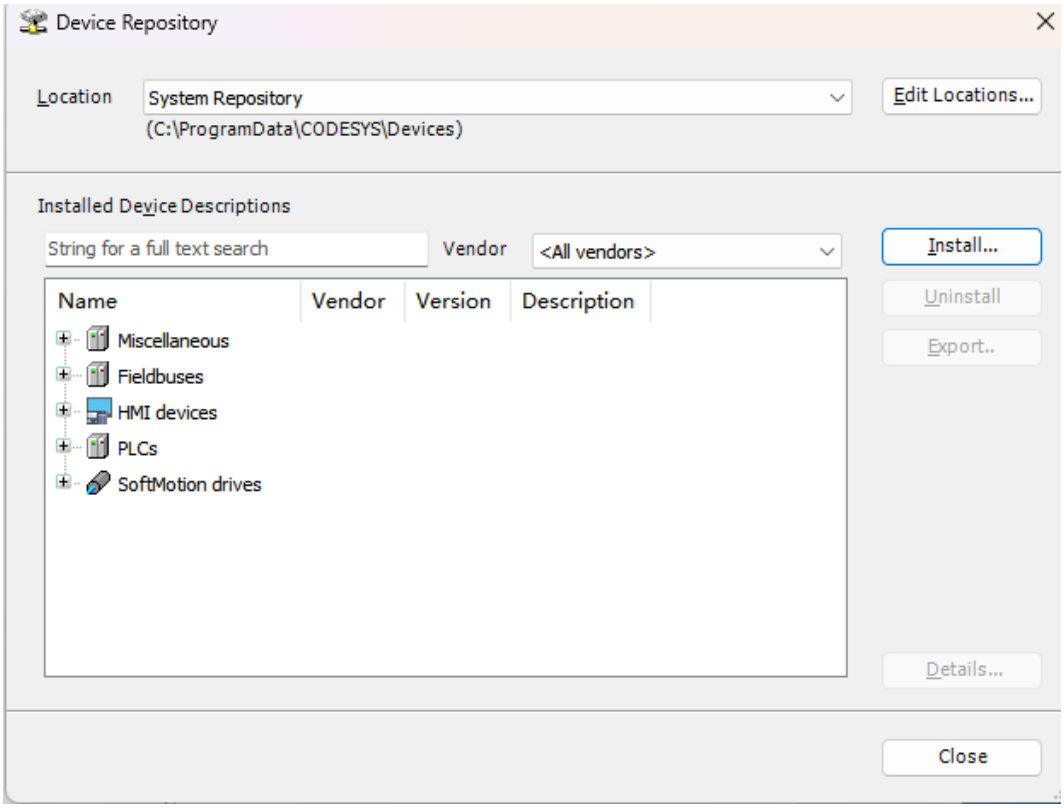


(2) After the download is completed, install the default configuration directly. Note that you need to run the file CODESYS.3.5.19.20 as an administrator to start the installation of CODESYS V3.

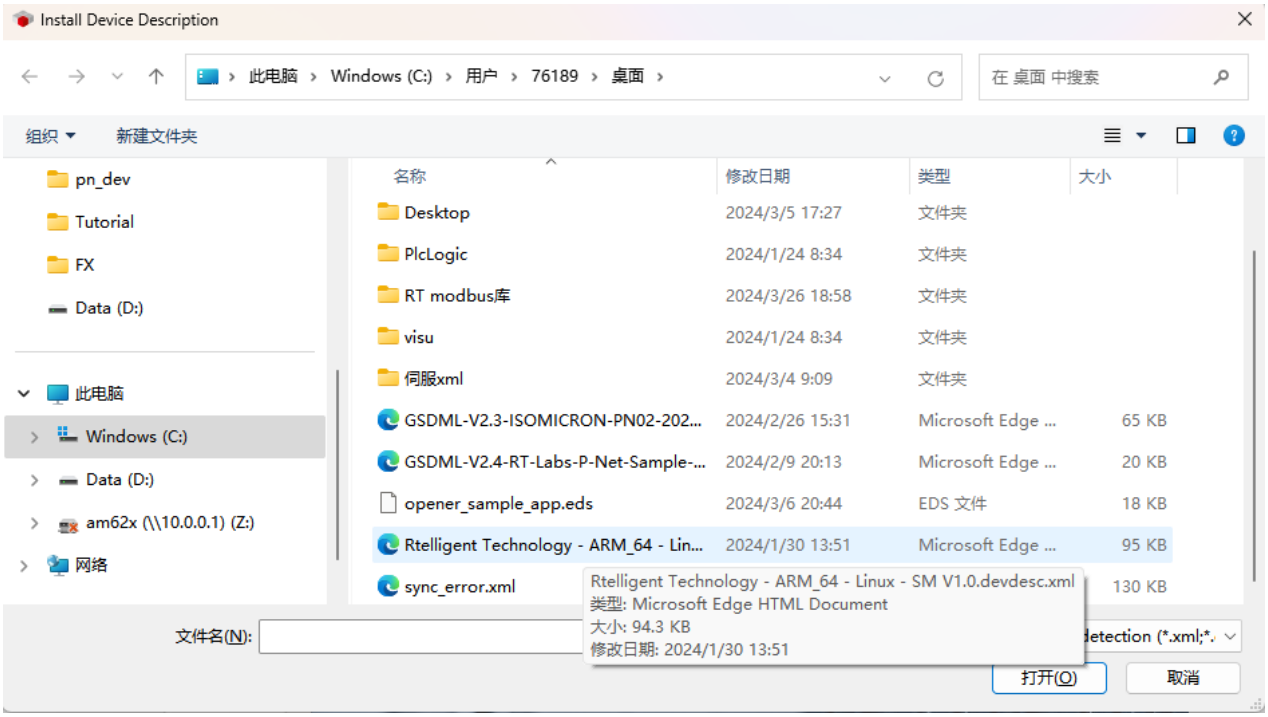
(3) After the installation is completed, the status bar will have the following three icons: permission management (which can be ignored, but must be present), gateway, and soft PLC.

5.2 Install the PLC Device Description File

(1) Open the software, open tools - Device repository - installation.



(2) Find the installation directory of the RM500 PLC device description file and select Install.



6 Create, Compile, and Run A CODESYS Project

In the previous chapter, we have completed the installation of CODESYS programming environment; This chapter details how to create, compile, and run the first CODESYS V3 project. Before doing so, reconfirm that the following requirements have been met:

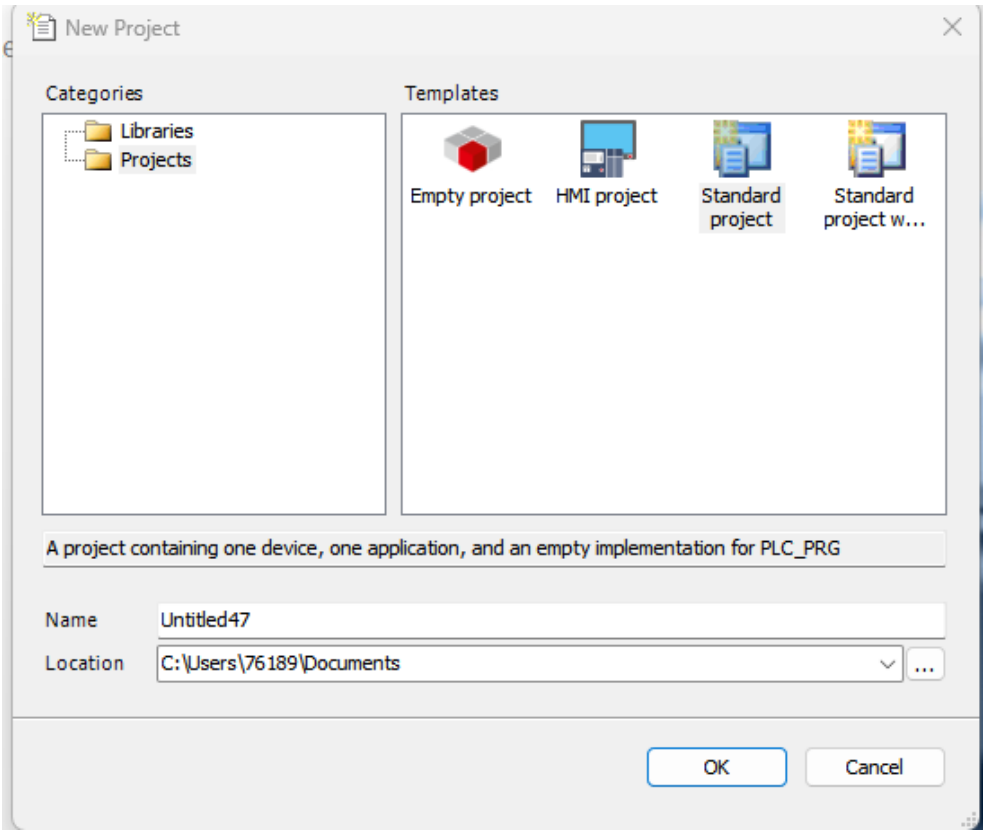
- The network settings of the controller must be configured correctly in order to access the controller through Ethernet.
- The CODESYS V3 version and device library and PLC device description files corresponding to the controller firmware version must be installed on the developer's computer.

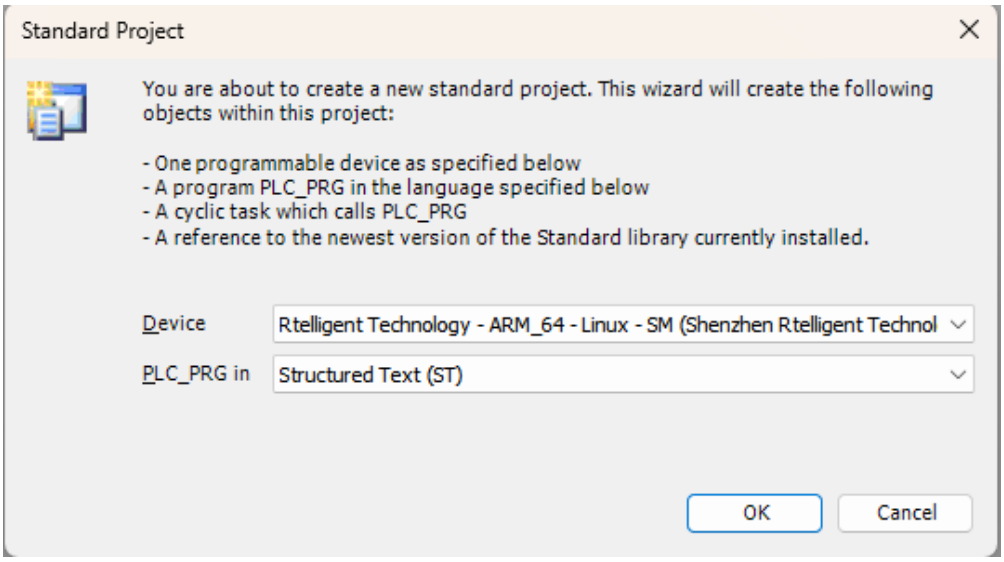
6.1 Quickly Create the First Project

After confirming the accuracy, we can gradually create a sample project for CODESYS V3 and load it onto the controller. The steps to be executed are as follows:

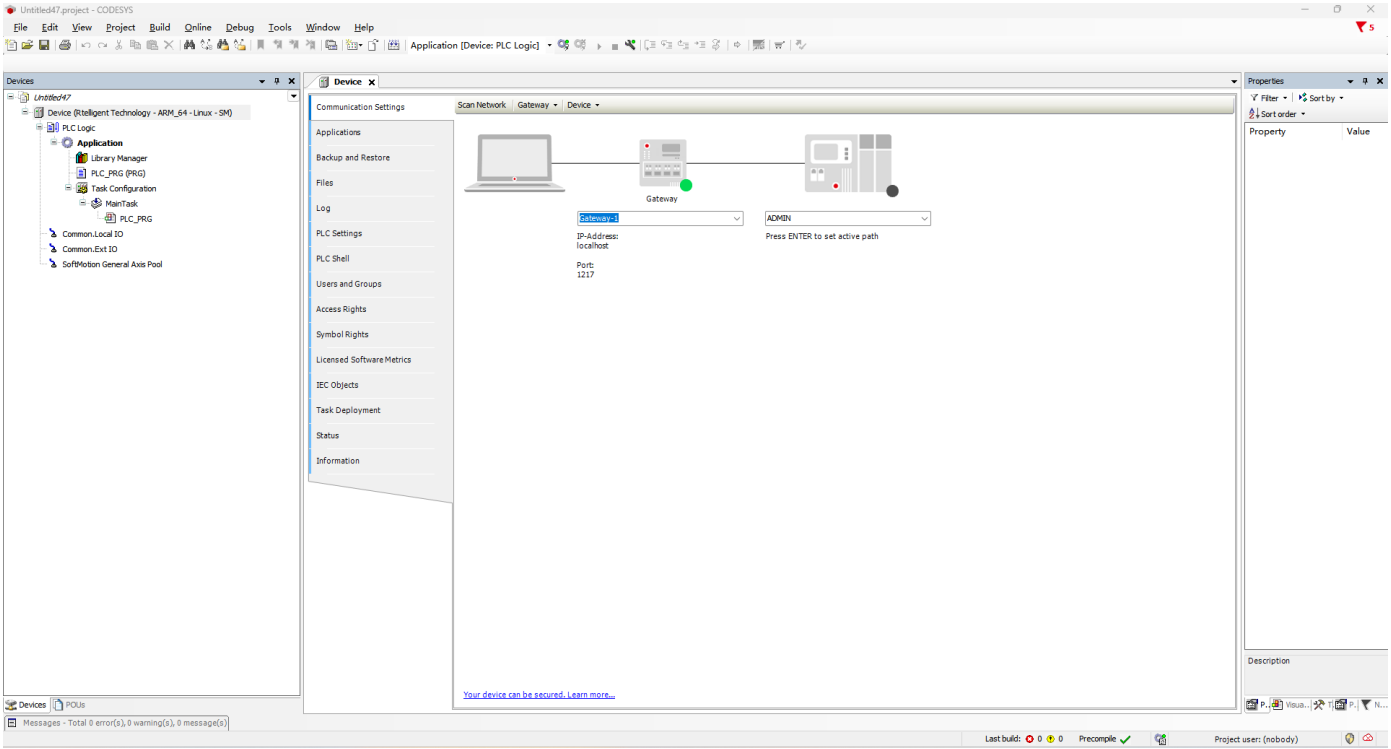
6.1.1 New Project

Click "New Project", select "Standard Project", customize the storage path and project name, pay attention to select the file device type.





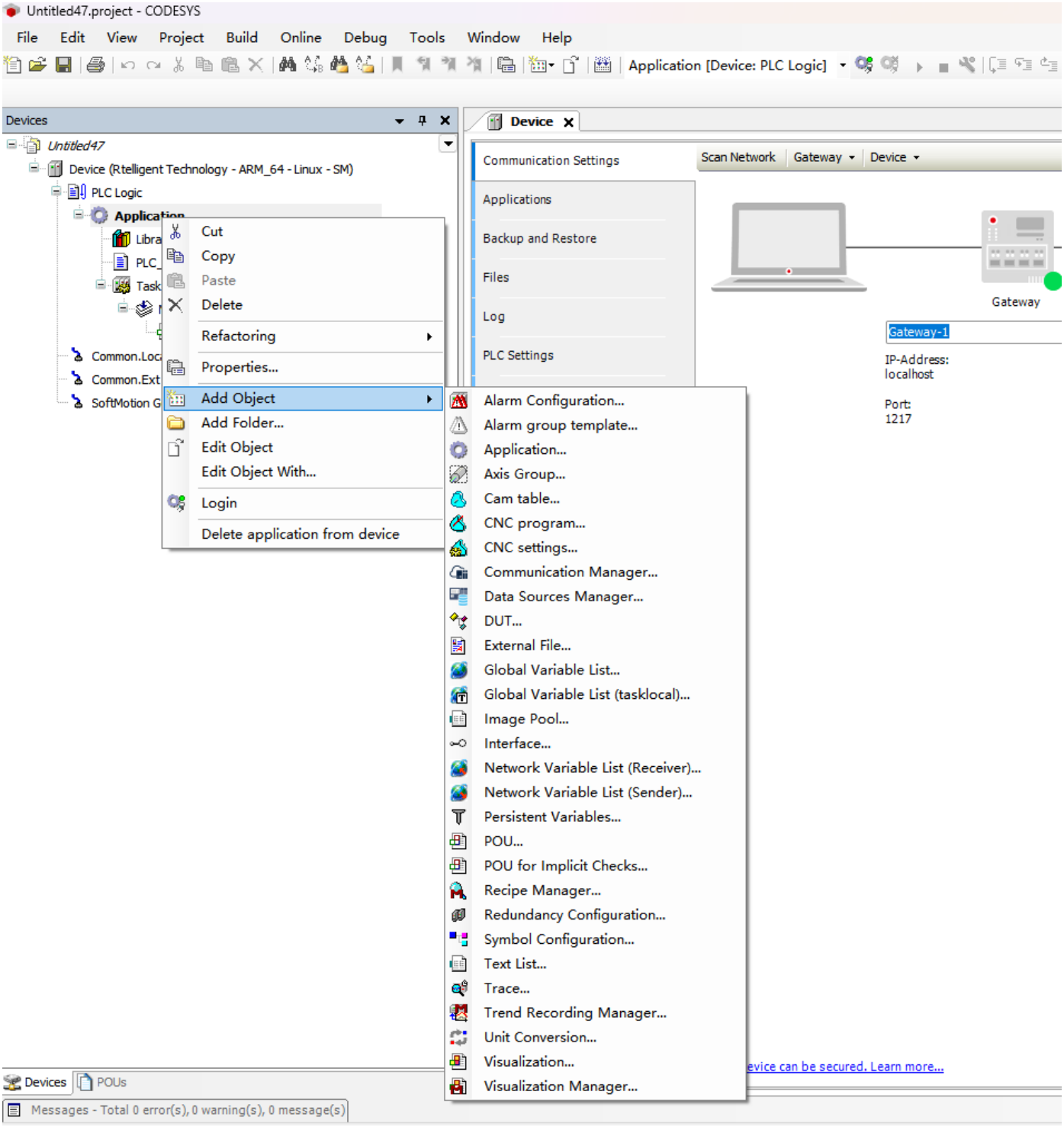
After successful creation, the project interface is shown as follows:



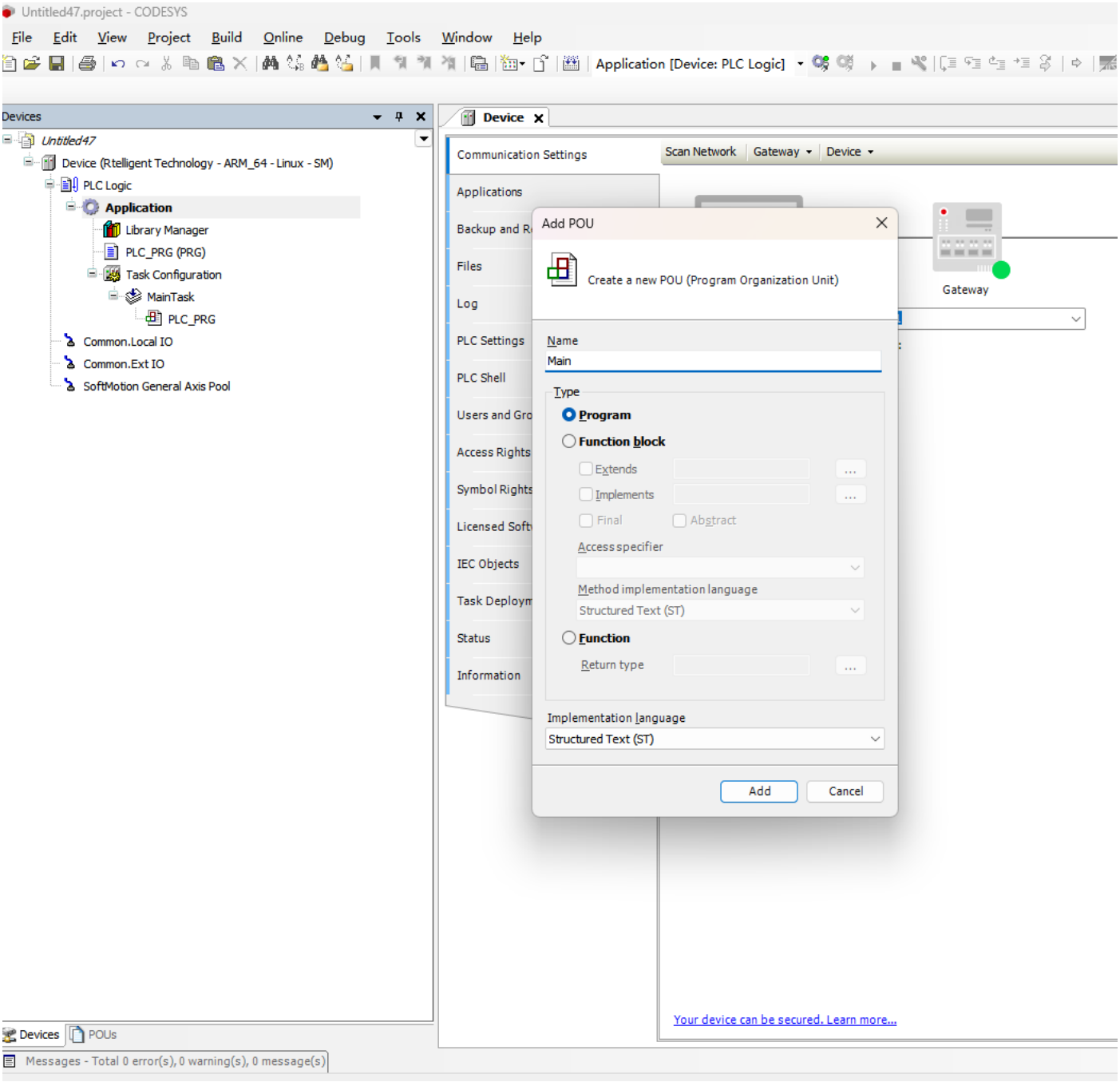
6.1.2 Create Programs and Define tasks

(1) Create programs:

Select your application object, then right-click to open the context menu, select "Add Object", and then select "POU..." in the submenu.

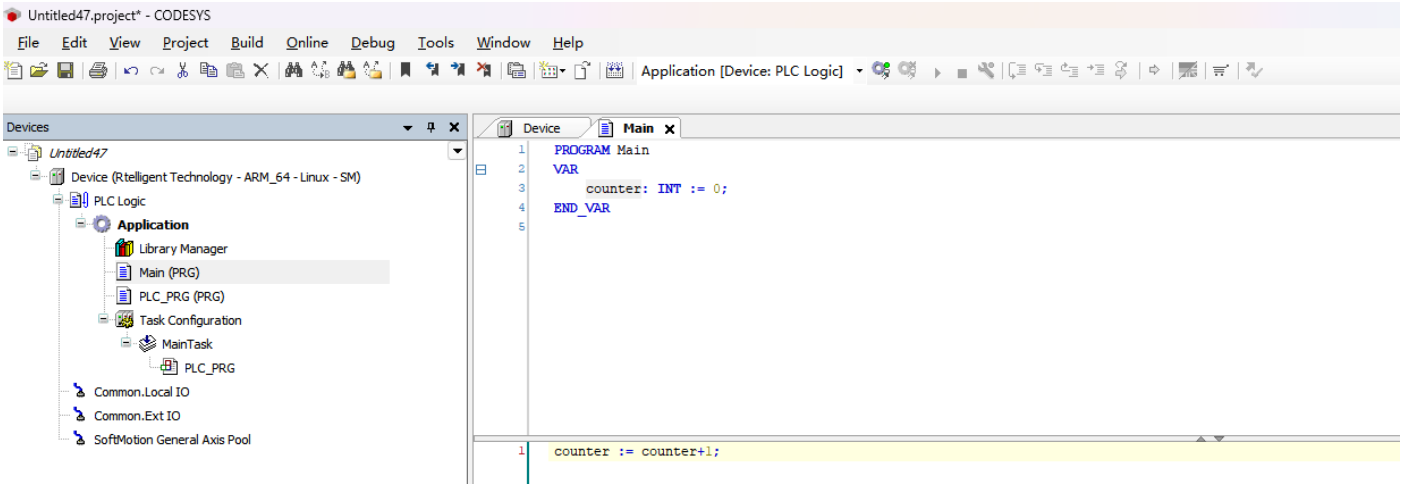


Choose the type of POU as "Program" and give the program a name. In this example, the name "Main" is used to indicate that this is the main program of the controller. The chosen implementation language is ST (Structured Text).



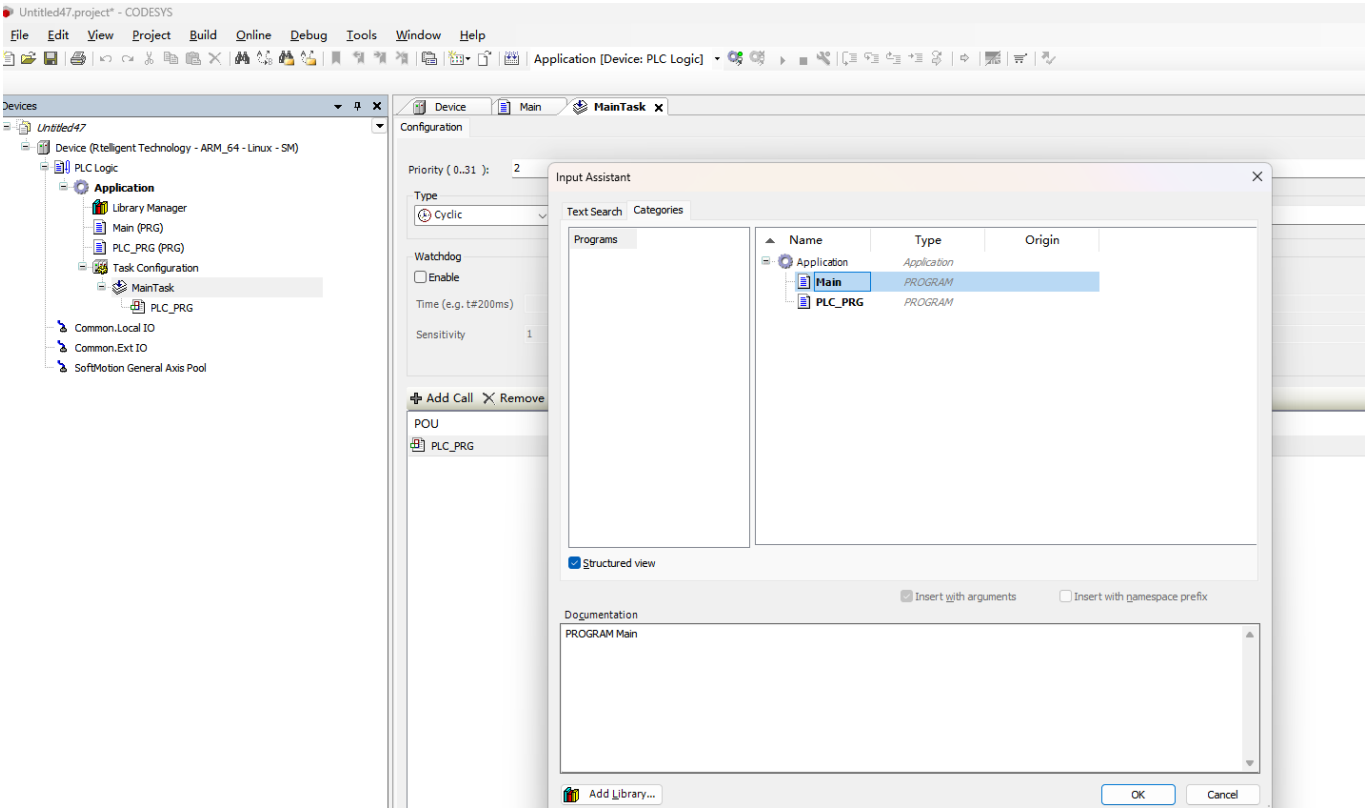
(2) Define tasks

By double clicking on the editing module, define our first variable in the upper part of the editor window, named "counter", with a data type of INT, and initialize this variable with a value of "0". We have implemented a simple program that implements+1 for the variable "counter" every time you call the "main" object.



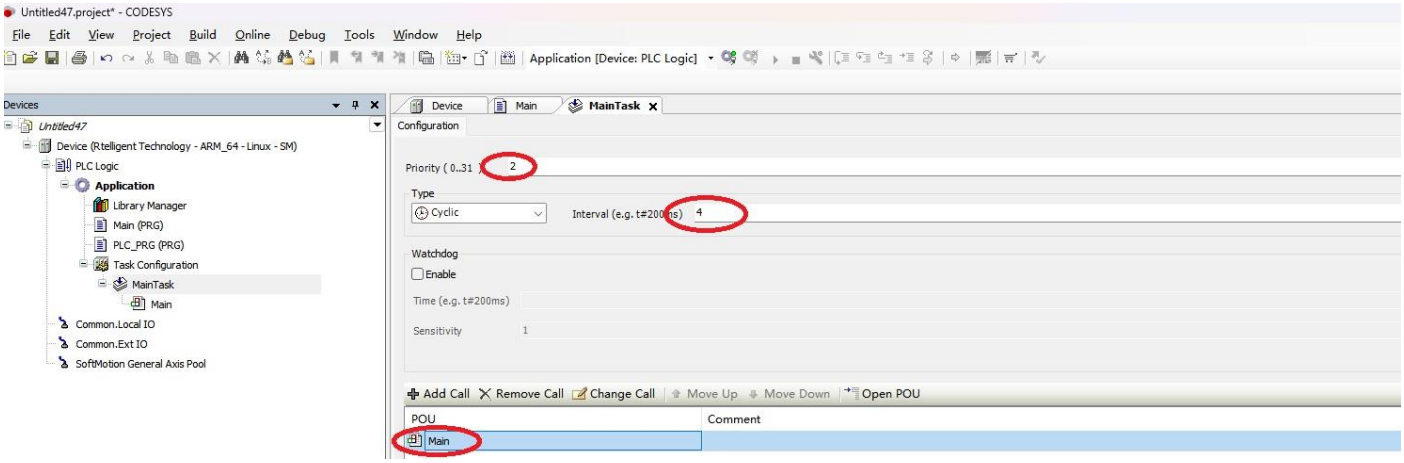
(3) Select task call object

The "Task Configuration" object will automatically create a subobject of type "MainTask". You can configure the "MainTask" object by double clicking it. Select "Add Call", then select the "Program" object "Main" that you have already created, and click OK.



(4) Task interval:

By default, the task interval is set to 4ms, which means the controller will call and run your "program" object every 4ms. For several defined tasks, when a program is executing, higher priority tasks take priority over lower priority tasks, which can interrupt the execution of lower priority programs in the same resource, slowing down the execution of lower priority programs.

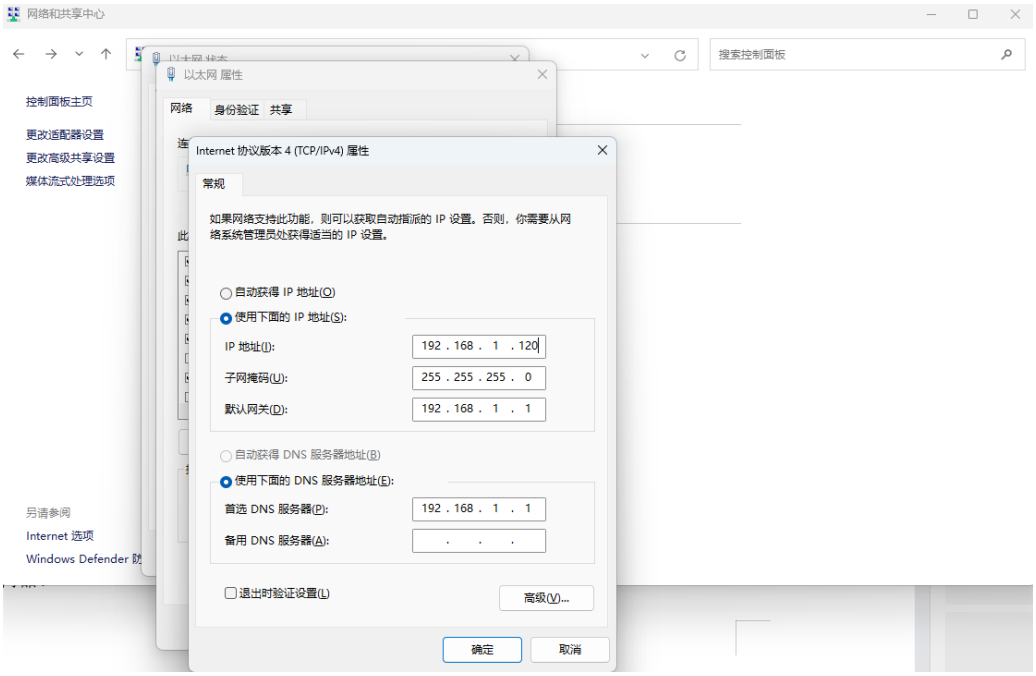


6.1.3 Log in to the Controller and Run the Project

When an application is to be loaded onto a controller, CODESYS V3 does not automatically know which controller the project should be loaded on. This requires users to assign their own controllers to the CODESYS V3 project. In addition to assigning controllers, it is also necessary to confirm that the application has no errors.

(1) Computer IP Settings

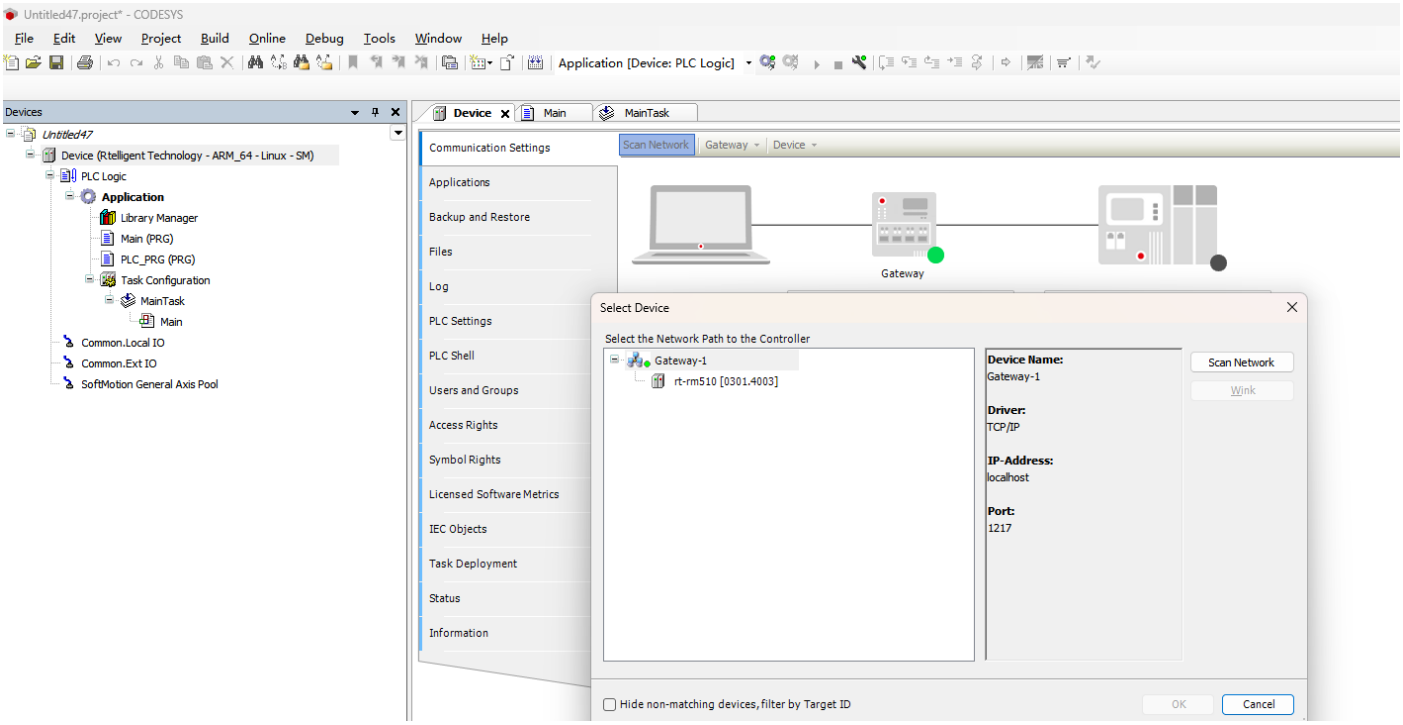
RM series PLC factory IP set to 192.168.1.3, you need to change the computer IP to the same network segment, such as 192.168.1.120 to establish communication with PLC. To change the IP address of the computer, perform the following steps: Control Panel - > Network and Sharing Center - > Ethernet - > Properties, as follows:



(2) Scan controller

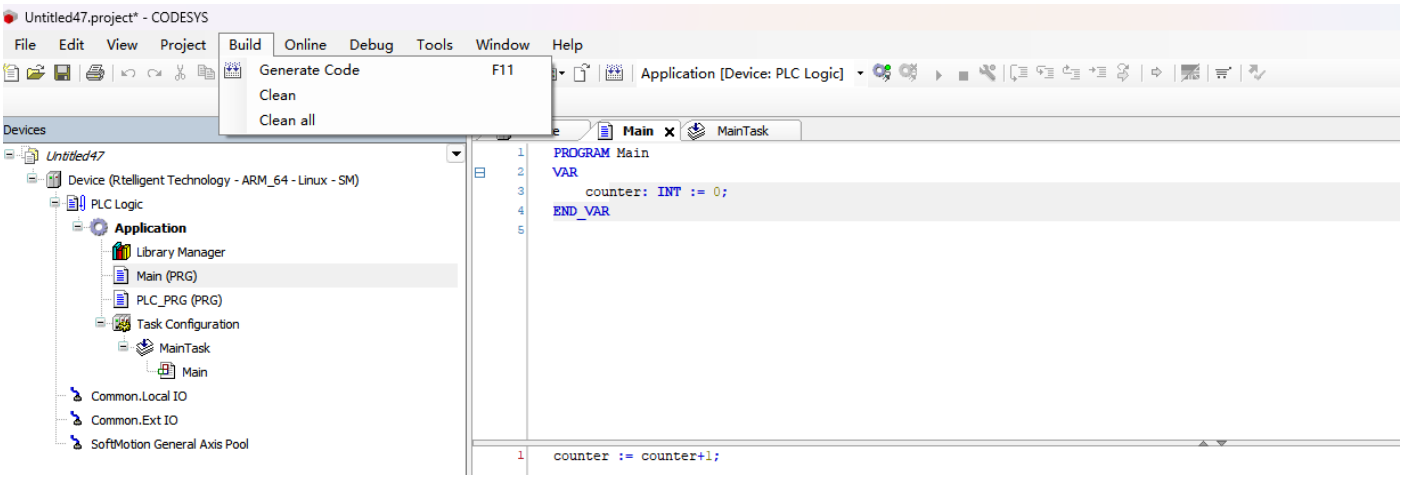
Double-click device in the device window to open the communication settings, and the device that has not been assigned is displayed.

Click the "Scan Network" button to search for connected controllers in your network.



(3) Check program error

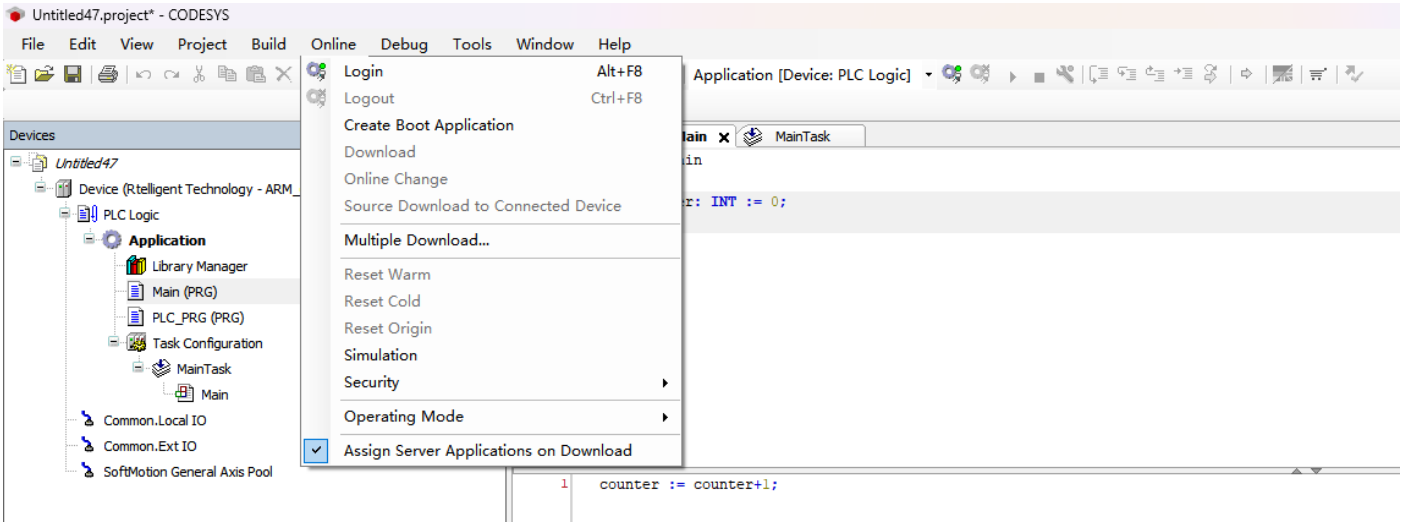
Press the " Generate Code" option under the "Compile" menu bar, or use the "F11" shortcut to build the program and check for errors in code, visualization, and settings.



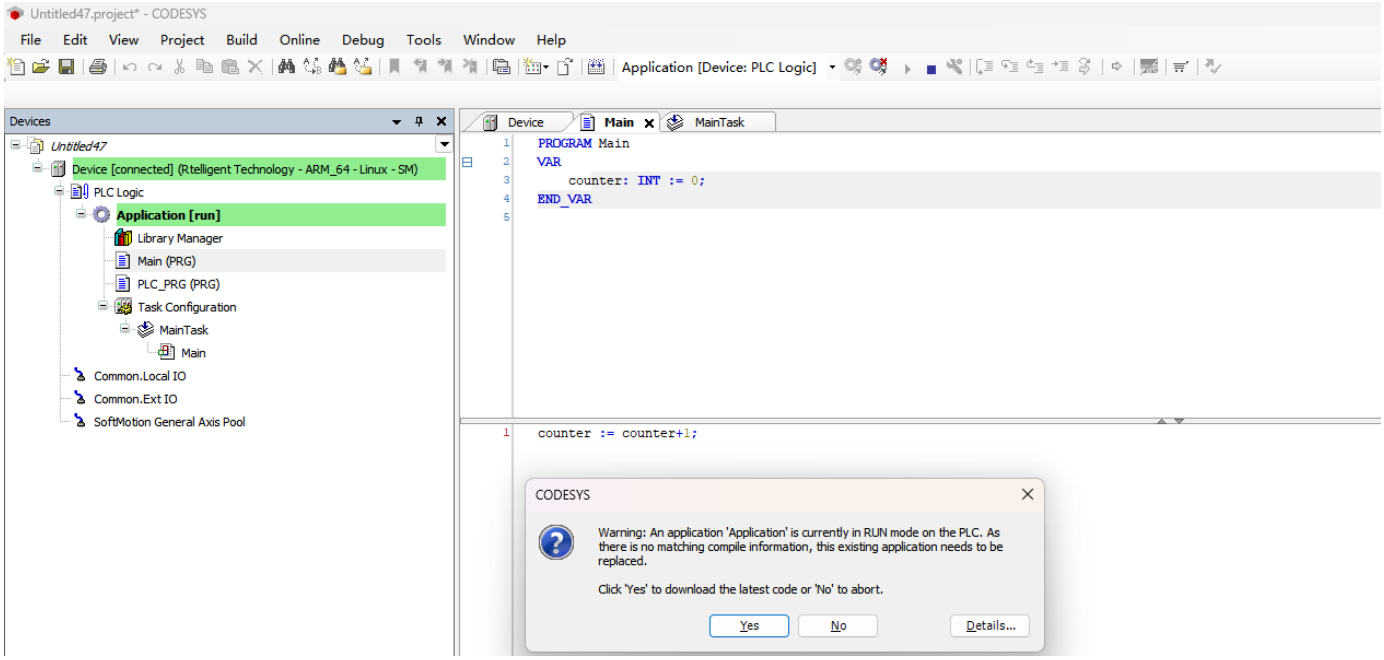
After a short wait, the results are displayed in the message window. If you did not make an error while creating this example, "0 error" and "0 Warning" should be displayed; If an error does occur, it will be displayed in a message, and by double-clicking the error message, CODESYS V3 will automatically jump to the wrong position, helping you to fix the error efficiently and easily. If the project is completely error-free and a controller has been assigned, then you can load the program onto that controller.

(4) Load application

To log in, please press the “Online -> Login” button in the menu bar.



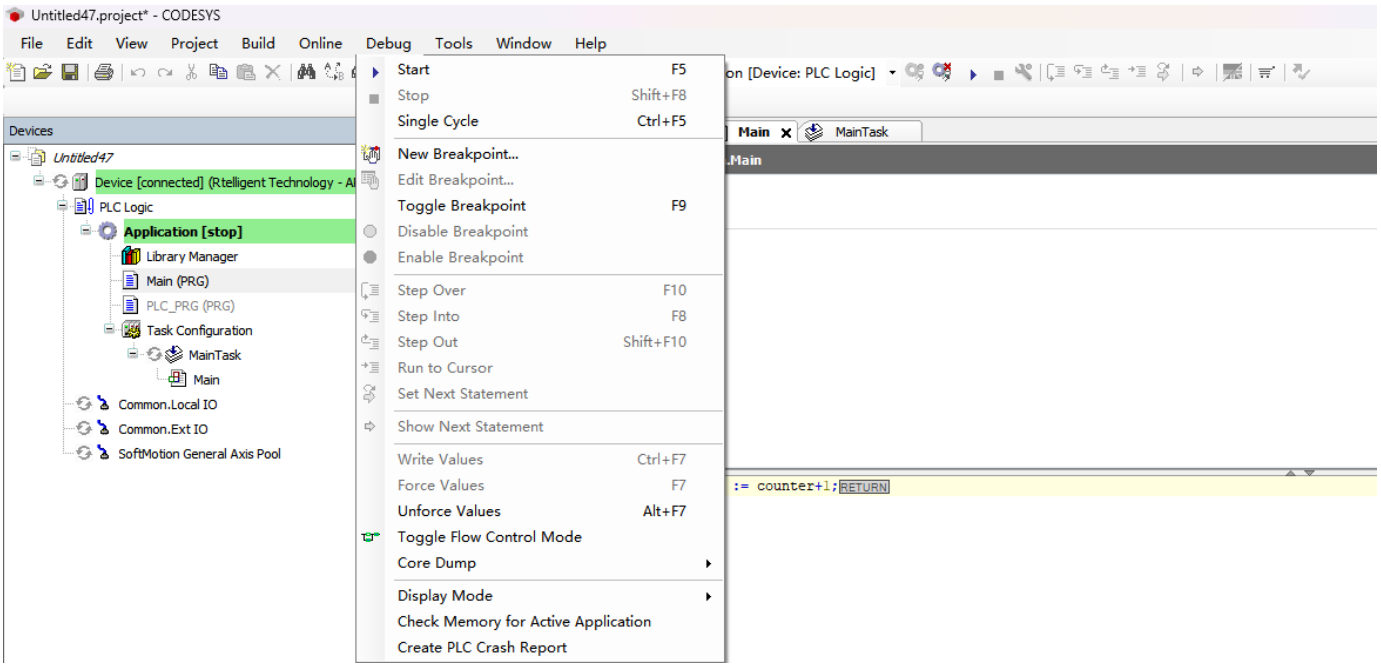
If there is no application on the controller so far, the message shown in the figure appears; If an application is already loaded on the controller, a message appears stating that there is an unknown application on the controller. This message may vary depending on whether an existing application is running.



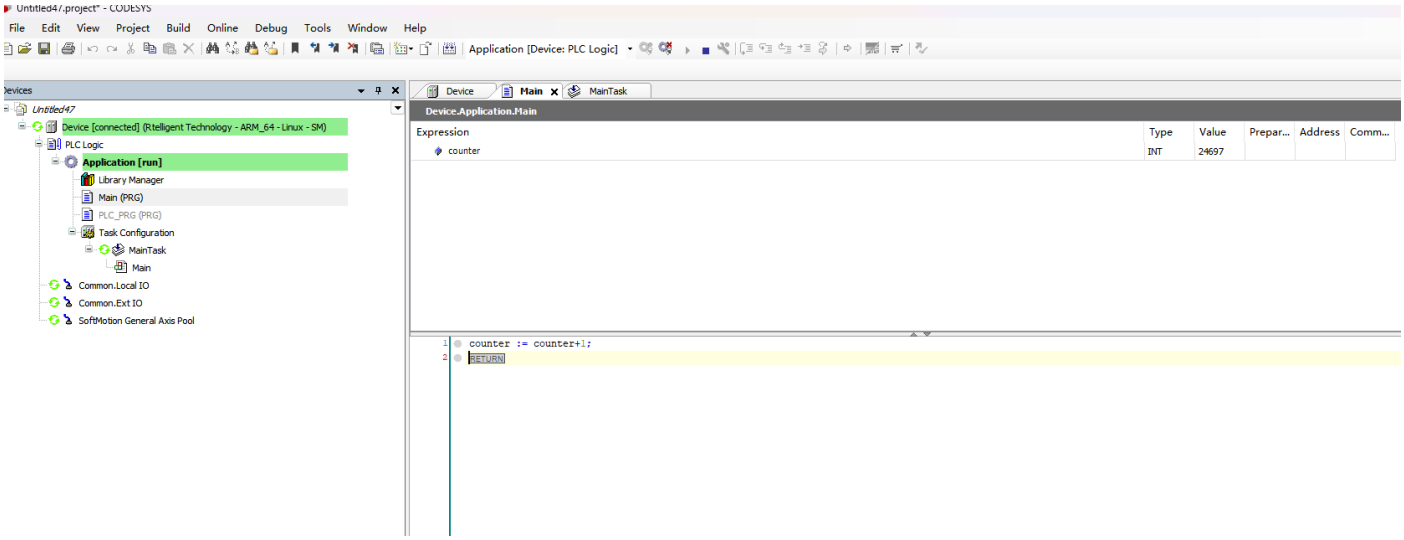
In all cases, press "Yes" to confirm. But if a message shows that there are still errors in the program, you need to cancel the login and first find the errors in the program and correct them. The application is then loaded onto the controller with CODESYS V3.

(5) Run Application

The loading process is completed when a green background appears on "devices" and "applications", followed by the words "[connected]" or "[stopped]". At this point, the application has been fully loaded onto the controller, but it is still in the Idle state and has not been run yet. To start the program, click on "Debug -> Start" in the menu bar or press the "F5" key.



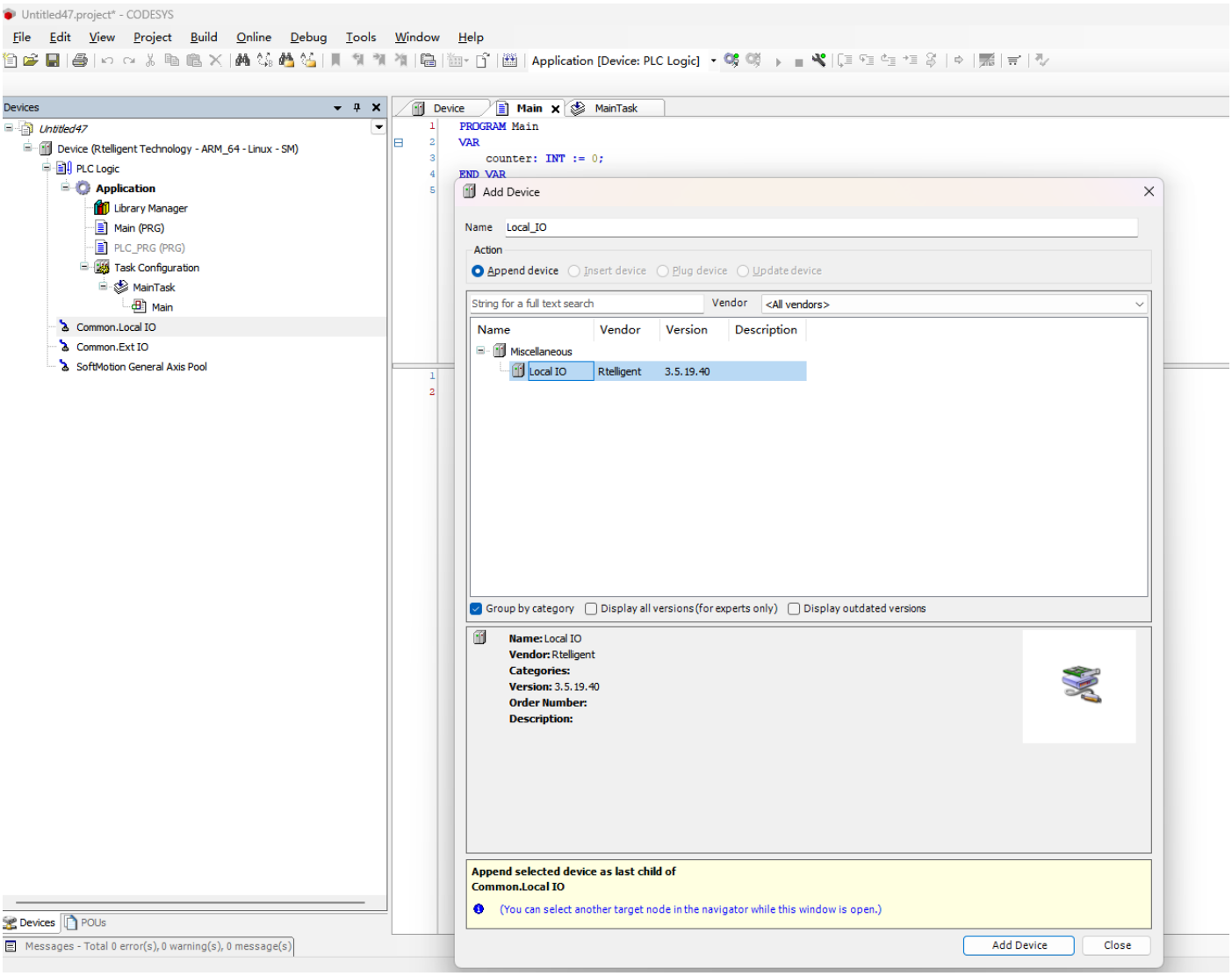
When the "Application" status in the device window changes from "[Stop]" to "[Run]", the program will be executed on the controller; At this point, by clicking on the previously created program "Main", you can see that the values in the counter "counter" are accumulating.



7I /O Configuration

7.1 PLC Local IO Configuration

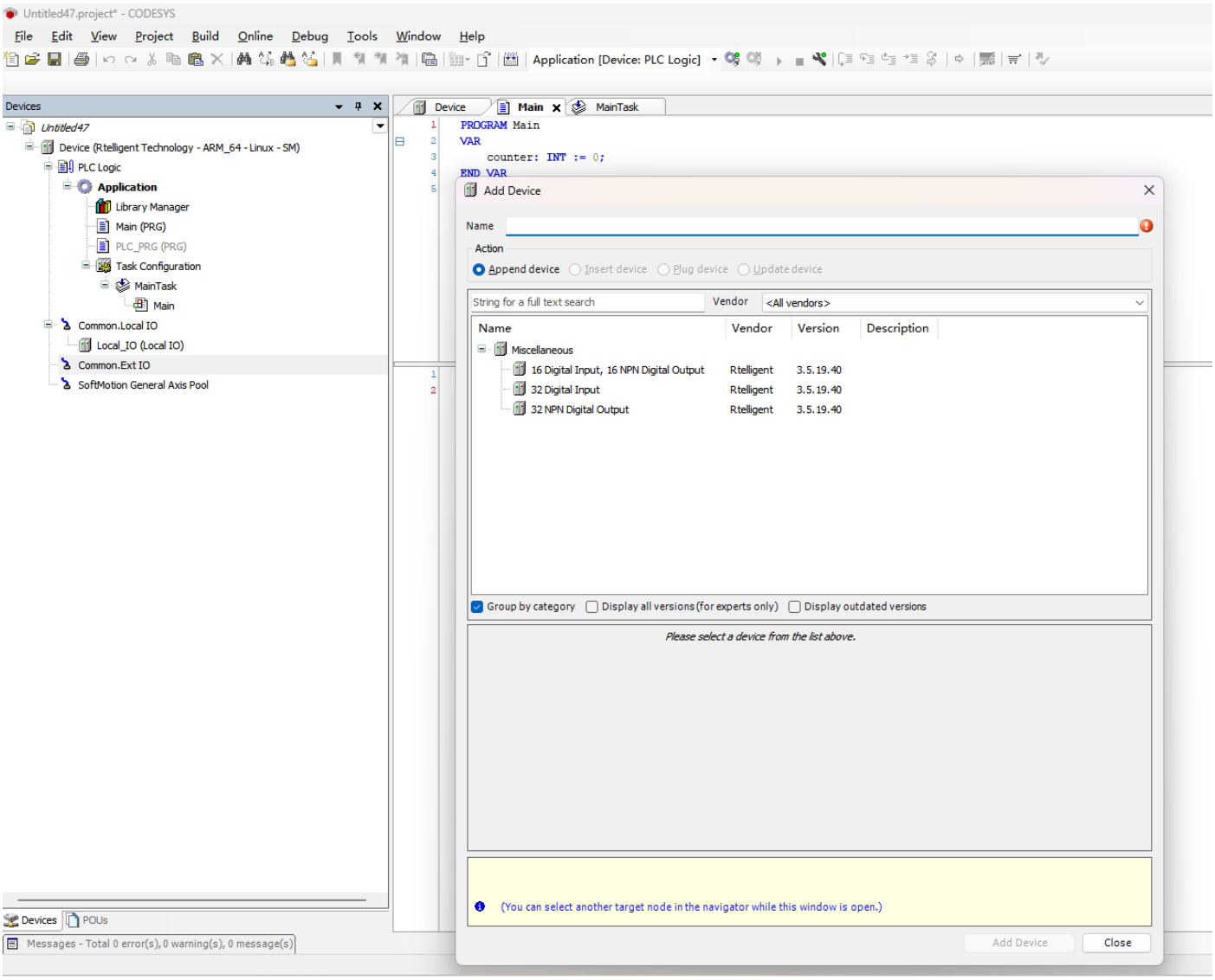
Right click on "Common Local IO" in the project, add the device, and select "Local IO" as follows:



7.2 PLC Expansion I/O Module Configuration

PLC expansion IO module needs users to buy separately, here describes how to configure and use.

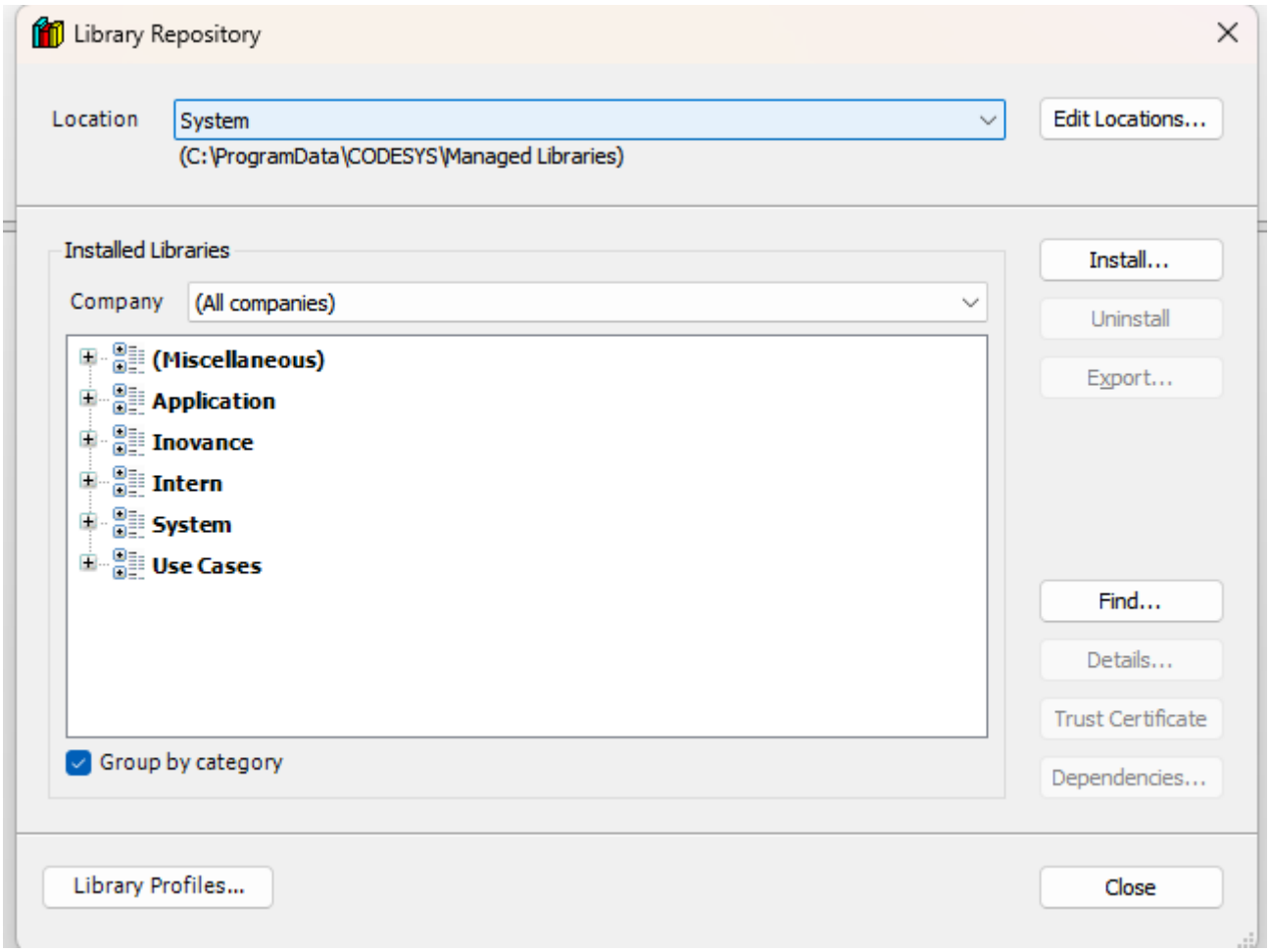
Right click on "Common Ext IO" in the project, add the device, select the extended IO module that needs to be added (up to 8 IO modules can be added), as shown below:



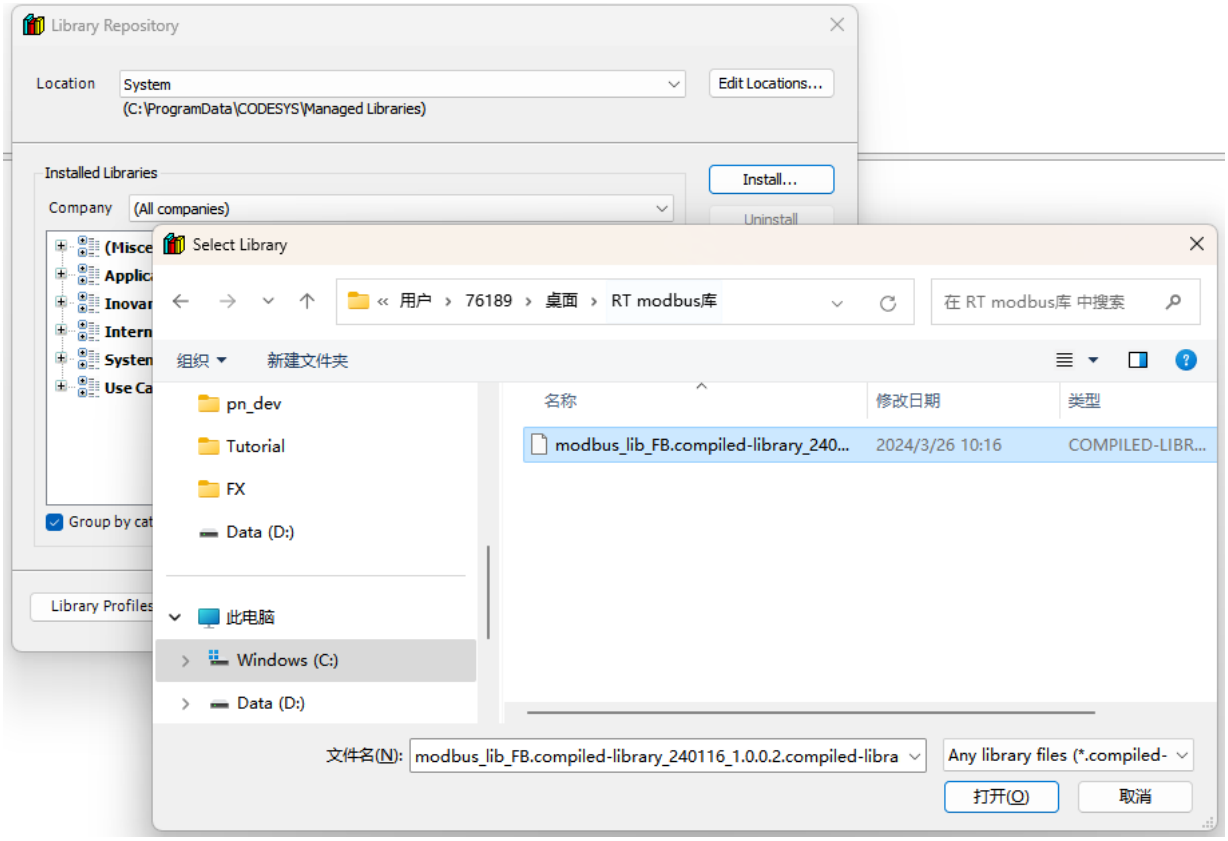
8 RT Modbus Library Installation Instructions

8.1 Library Installation

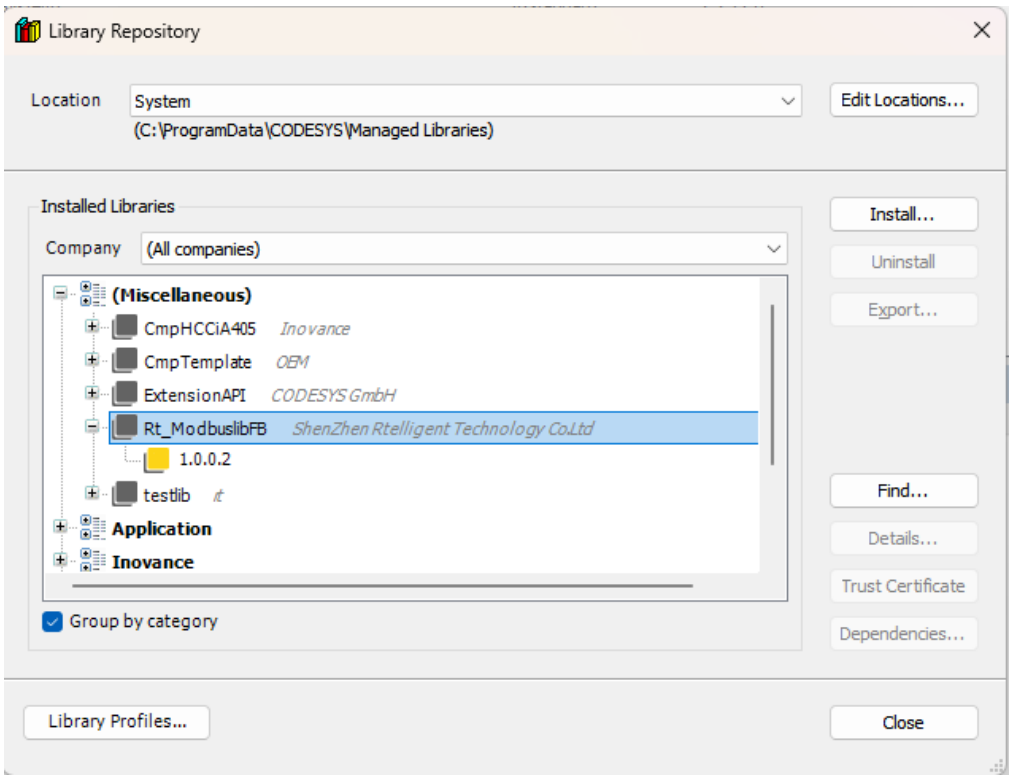
(1) Find Tools -> Library Storage above the CODESYS engineering environment, and click to bring up the following interface:



(2) Click the install button in the interface to install the provided communication library, as shown below:

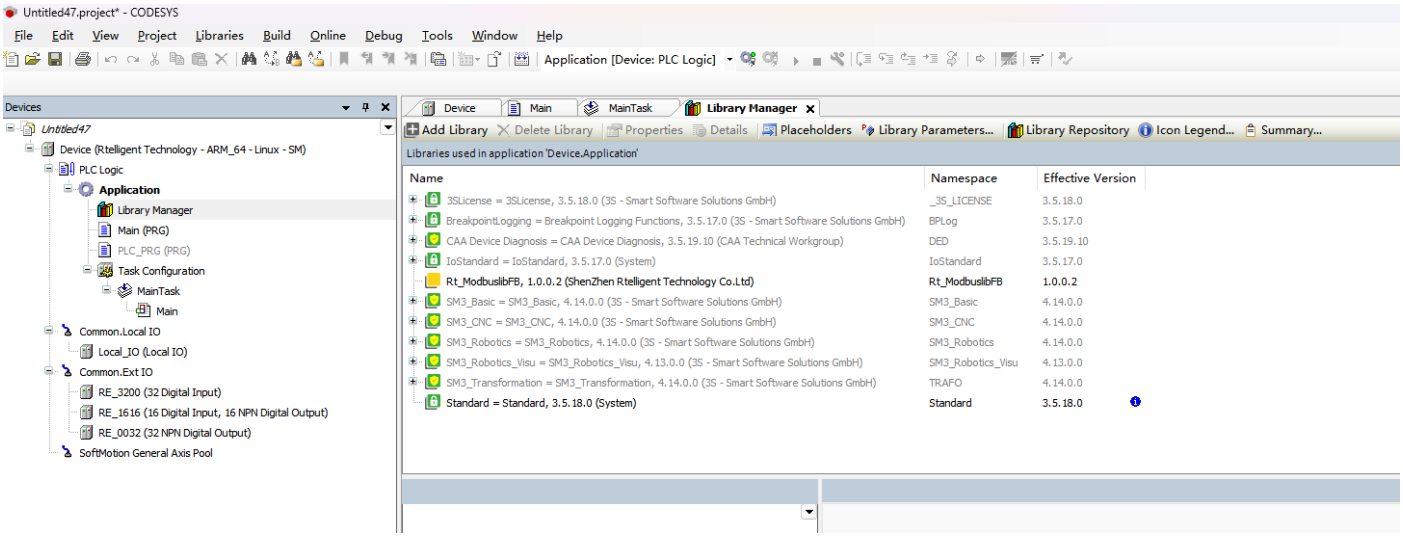


(3) When the installation is completed, the library you installed will appear:

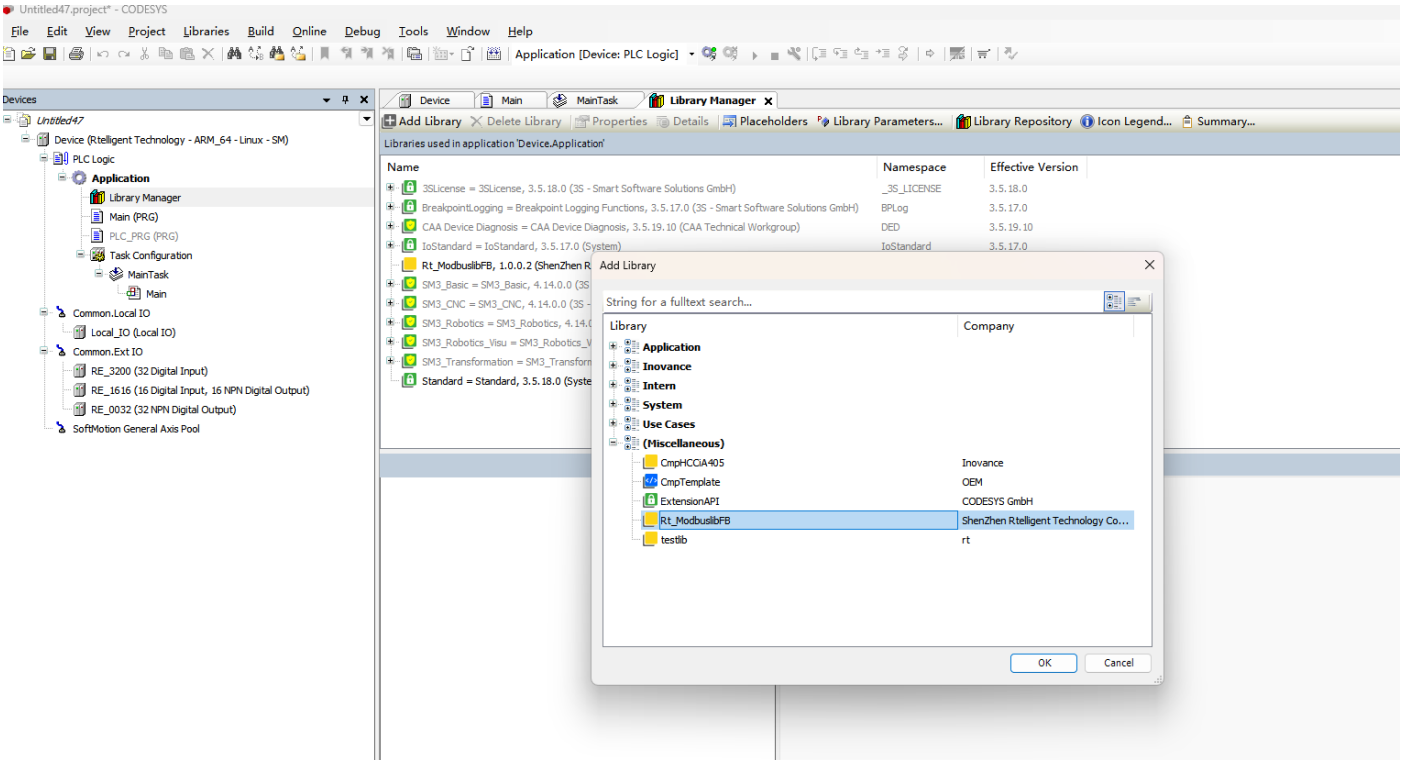


8.2 Project Usage

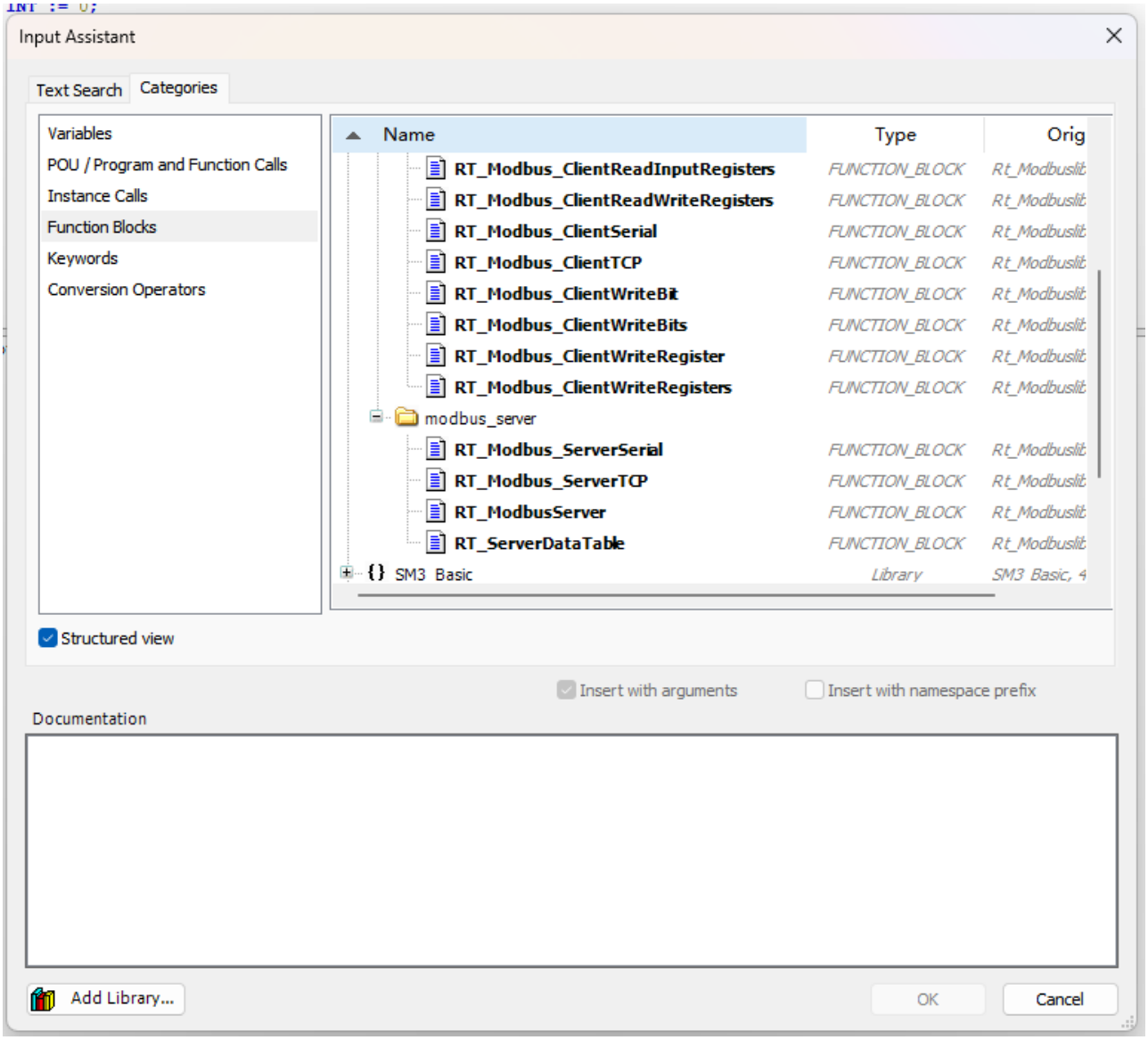
(1) Find the "Library Manager" in the project and double-click to display the following interface.



(2) Click "Add Library" to find the corresponding library in the interface that appears.



(3) At this point, the relevant library can be used in the program, and the function block in the library can be input through F2 (input assistant) in the program, as shown in the following figure.



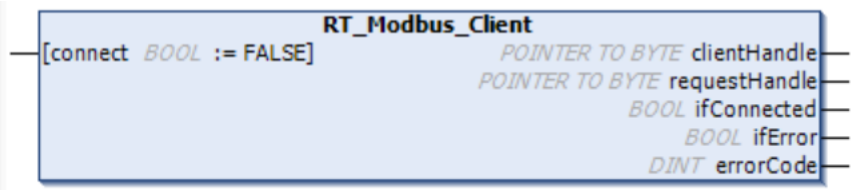
8.3 Modbus Maste Interface Description

8.3.1 RT_Modbus_Client

The Modbus master abstract base class, the parent class of the RTU and TCP master, and the subclasses share the state members of the master.

FUNCTION_BLOCK ABSTRACT RT_Modbus_Client

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	connect	BOOL			FALSE	Rising edge: Read the parameters and connect to server.Falling edge:Close connection.
OUTPUT	clientHandle	POINTER TO BYTE				modbus client handle.
OUTPUT	requestHandle	POINTER TO BYTE				modbus request handle.
OUTPUT	ifConnected	BOOL				Client is connected to server.
OUTPUT	ifError	BOOL				Error.
OUTPUT	errorCode	DINT				Error status,0 means no error.



8.3.2 RT_Modbus_ClientMaskWriteRegister

Mask write register, performs a bit operation on the specified register (function code 0x16).

This function uses ”andMask”, “orMask”

This function uses "andMask", "orMask", and the current contents of the register to modify the contents of the specified holding register. Can be used to set or clear a bit in a register.

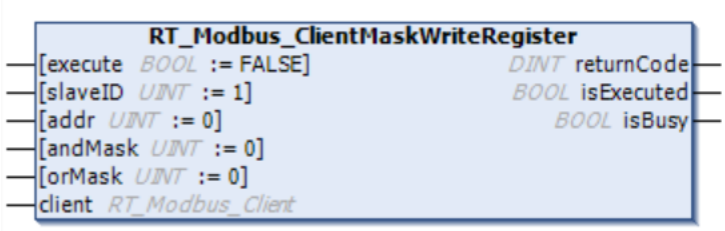
Result = (current register content && andmask) || (orMask && (~andMask))

For example:

	Hex	Binary
Current Contents=	12	0001 0010
And_Mask =	F2	1111 0010
Or_Mask =	25	0010 0101
(NOT And_Mask)=	0D	0000 1101
Result =	17	0001 0111

FUNCTION_BLOCK RT_Modbus_ClientMaskWriteRegister

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	addr	UINT			0	Register address.
INPUT	andMask	UINT			0	andMask
INPUT	orMask	UINT			0	orMask
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

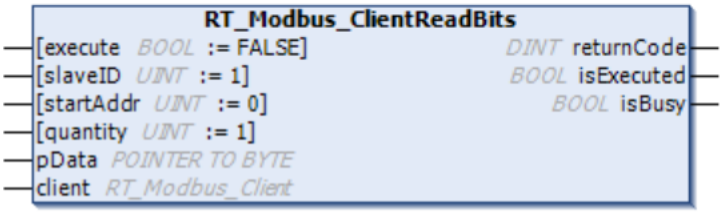


8.3.3 RT_Modbus_ClientReadBits

Read coil or discrete output status (function code 0x1).

FUNCTION_BLOCK RT_Modbus_ClientReadBits

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	startAddr	UINT			0	Start address.
INPUT	quantity	UINT			1	Number of item to read.
INPUT	pData	POINTER TO BYTE				Pointer to result data array, memory has to be provided by caller.
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

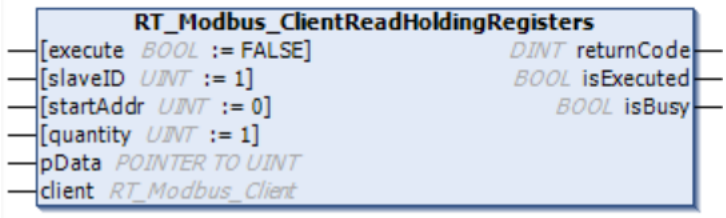


8.3.4 RT_Modbus_ClientReadHoldingRegisters

Read holding register (function code 0x3)

FUNCTION_BLOCK RT_Modbus_ClientReadHoldingRegisters

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	startAddr	UINT			0	Start address.
INPUT	quantity	UINT			1	Number of item to read.
INPUT	pData	POINTER TO UINT				Pointer to result data array, memory has to be provided by caller.
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

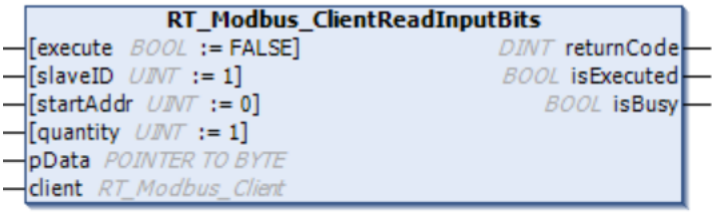


8.3.5 RT_Modbus_ClientReadInputBits

Read discrete input value (function code 0x2)

FUNCTION_BLOCK RT_Modbus_ClientReadInputBits

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	startAddr	UINT			0	Start address.
INPUT	quantity	UINT			1	Number of item to read.
INPUT	pData	POINTER TO BYTE				Pointer to result data array, memory has to be provided by caller.
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

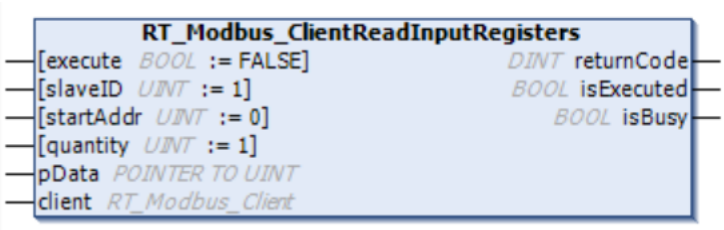


8.3.6 RT_Modbus_ClientReadInputRegisters

Read input register (function code 0x4)

FUNCTION_BLOCK RT_Modbus_ClientReadInputRegisters

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	startAddr	UINT			0	Start address.
INPUT	quantity	UINT			1	Number of item to read.
INPUT	pData	POINTER TO UINT				Pointer to result data array, memory has to be provided by caller.
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

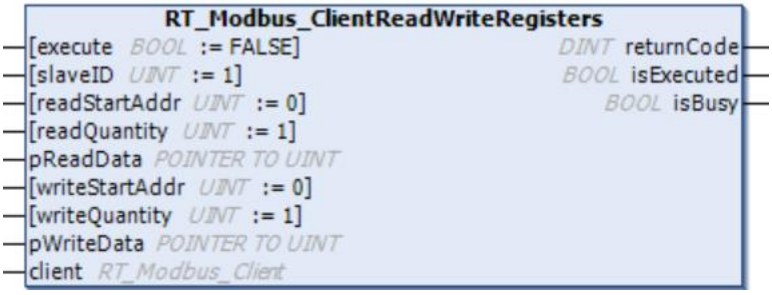


8.3.7 RT_Modbus_ClientReadWriteRegisters

Read and write multiple registers at the same time

FUNCTION_BLOCK RT_Modbus_ClientReadWriteRegisters

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	readStartAddr	UINT			0	Start address to read.
INPUT	readQuantity	UINT			1	Number of item to read.
INPUT	pReadData	POINTER TO UINT				Pointer to result data array, memory has to be provided by caller.
INPUT	writeStartAddr	UINT			0	Start address to write.
INPUT	writeQuantity	UINT			1	Number of item to write.
INPUT	pWriteData	POINTER TO UINT				Pointer to data array, memory has to be provided by caller.
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

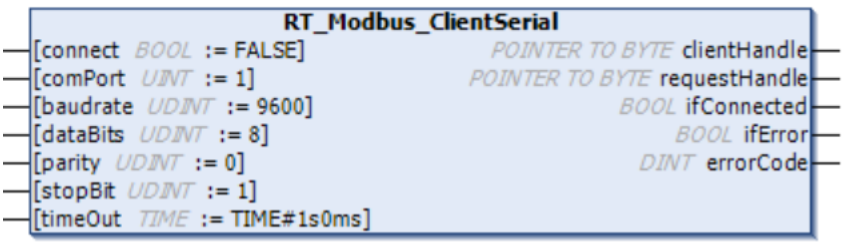


8.3.8 RT_Modbus_ClientSerial

ModbusRTU initialization connection operation; Inherited from RT_Modbus_Client.

FUNCTION_BLOCK RT_Modbus_ClientSerial EXTENDS RT_Modbus_Client

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	connect	BOOL	RT_Modbus_Client		FALSE	Rising edge: Read the parameters and connect to server.Falling edge:Close connection.
INPUT	comPort	UINT			1	Serial port, only read when rising edge on connect occurs,range[1,2].
INPUT	baudrate	UDINT			9600	Baud rate, only read when rising edge on connect occurs.
INPUT	dataBits	UDINT			8	Number of data bits/BYTE, 4-8, only read when rising edge on connect occurs.
INPUT	parity	UDINT			0	Parity, only read when rising edge on connect occurs,0=connect,1=Odd,2=even.
INPUT	stopBit	UDINT			1	Stop bits, only read when rising edge on connect occurs.
INPUT	timeOut	TIME			TIME#1s0ms	Communication timeout waiting time.
OUTPUT	clientHandle	POINTER TO BYTE	RT_Modbus_Client			modbus client handle.
OUTPUT	requestHandle	POINTER TO BYTE	RT_Modbus_Client			modbus request handle.
OUTPUT	ifConnected	BOOL	RT_Modbus_Client			Client is connected to server.
OUTPUT	ifError	BOOL	RT_Modbus_Client			Error.
OUTPUT	errorCode	DINT	RT_Modbus_Client			Error status,0 means no error.

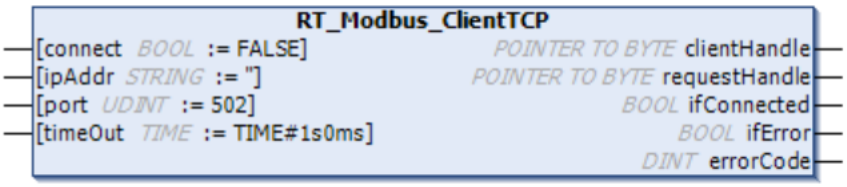


8.3.9 RT_Modbus_ClientTCP

ModbusTCP initialization connection operation; Inherited from RT_Modbus_Client.

FUNCTION_BLOCK RT_Modbus_ClientTCP EXTENDS RT_Modbus_Client

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	connect	BOOL	RT_Modbus_Client		FALSE	Rising edge: Read the parameters and connect to server.Falling edge:Close connection.
INPUT	ipAddr	STRING			''	Ip address
INPUT	port	UDINT			502	Port. The default port used by ModbusTCP is 502.
INPUT	timeOut	TIME			TIME#1s0ms	Communication timeout waiting time.
OUTPUT	clientHandle	POINTER TO BYTE	RT_Modbus_Client			modbus client handle.
OUTPUT	requestHandle	POINTER TO BYTE	RT_Modbus_Client			modbus request handle.
OUTPUT	ifConnected	BOOL	RT_Modbus_Client			Client is connected to server.
OUTPUT	ifError	BOOL	RT_Modbus_Client			Error.
OUTPUT	errorCode	DINT	RT_Modbus_Client			Error status,0 means no error.

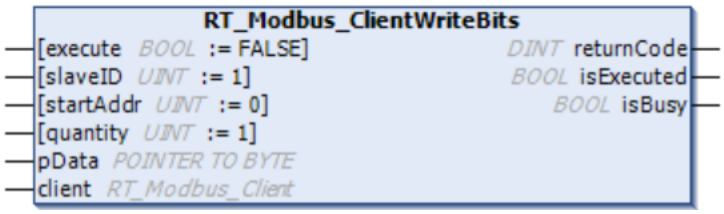


8.3.10 RT_Modbus_ClientWriteBits

Write multiple coils (function code 0xF)

FUNCTION_BLOCK RT_Modbus_ClientWriteBits

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	startAddr	UINT			0	Start address.
INPUT	quantity	UINT			1	Number of item to write.
INPUT	pData	POINTER TO BYTE				Pointer to data array, memory has to be provided by caller.
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

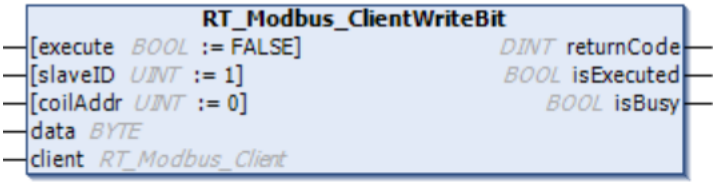


8.3.11 RT_Modbus_ClientWriteBit

Write a single coil or a single discrete quantity (function code 0x5)

FUNCTION_BLOCK RT_Modbus_ClientWriteBit

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	coilAddr	UINT			0	Address
INPUT	data	BYTE				Data
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

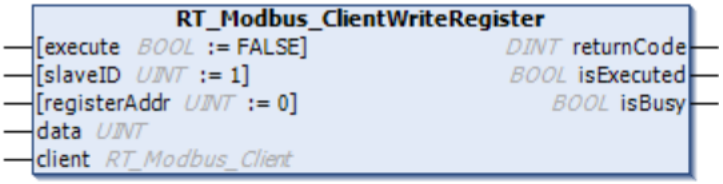


8.3.12 RT_Modbus_ClientWriteRegister

Write a single holding register (function code 0x6)

FUNCTION_BLOCK RT_Modbus_ClientWriteRegister

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	registerAddr	UINT			0	Register address.
INPUT	data	UINT				Data.
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.

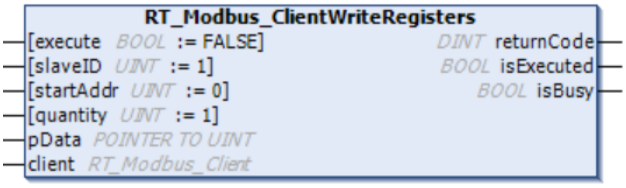


8.3.13 RT_Modbus_ClientWriteRegisters

Write multiple holding registers (function code 0x10)

FUNCTION_BLOCK RT_Modbus_ClientWriteRegisters

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	execute	BOOL			FALSE	Rising edge: Starts defined operation FALSE: Resets the defined operation after ready condition was reached.
INPUT	slaveID	UINT			1	Slave id.
INPUT	startAddr	UINT			0	Start address.
INPUT	quantity	UINT			1	Number of item to write.
INPUT	pData	POINTER TO UINT				Pointer to data array, memory has to be provided by caller.
IN_OUT	client	RT_Modbus_Client				Client handle,reference to client.
OUTPUT	returnCode	DINT				Execution return. The number of operations returned in success and error codes returned in failure.
OUTPUT	isExecuted	BOOL			FALSE	Ready condition reached.
OUTPUT	isBusy	BOOL			FALSE	Operation is running.



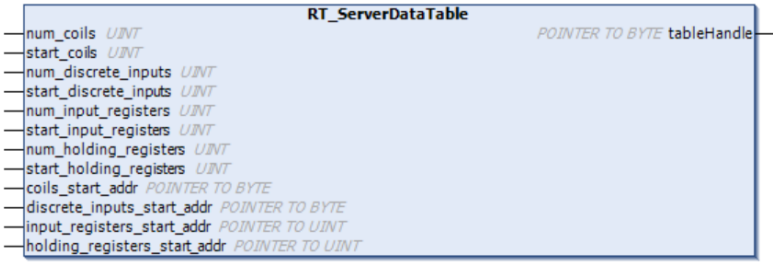
8.4 Modbus Slave Interface Description

8.4.1 RT_ServerDataTable

Modbus slave data map table, the number of registers should be equal to the length of the array to which the incoming pointer points.

FUNCTION_BLOCK RT_ServerDataTable

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	num_coils	UINT				Number of coil registers.
INPUT	start_coils	UINT				Start address of coil registers.
INPUT	num_discrete_inputs	UINT				Number of discrete input registers.
INPUT	start_discrete_inputs	UINT				Start address of discrete input registers.
INPUT	num_input_registers	UINT				Number of input registers.
INPUT	start_input_registers	UINT				Start address of input registers.
INPUT	num_holding_registers	UINT				Number of holding registers.
INPUT	start_holding_registers	UINT				Start address of holding registers.
INPUT	coils_start_addr	POINTER TO BYTE				Pointer to coil data array, memory has to be provided by caller.
INPUT	discrete_inputs_start_addr	POINTER TO BYTE				Pointer to discrete inputs data array, memory has to be provided by caller.
INPUT	input_registers_start_addr	POINTER TO UINT				Pointer to input registers data array, memory has to be provided by caller.
INPUT	holding_registers_start_addr	POINTER TO UINT				Pointer to holding registers data array, memory has to be provided by caller.
OUTPUT	tableHandle	POINTER TO BYTE				Memory space handle

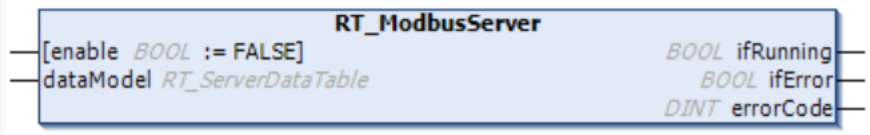


8.4.2 RT_ModbusServer

Modbus server base class, TCP and RTU servers inherit from this class. It mainly includes the running status of the server and common input parameters.

FUNCTION_BLOCK ABSTRACT RT_ModbusServer

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	enable	BOOL			FALSE	Enables the server, take over the configuration information.
IN_OUT	dataModel	RT_ServerDataTable				Parameters for running the server,memory has to be provided by caller.
OUTPUT	ifRunning	BOOL				Server is up and running, waiting for requests.
OUTPUT	ifError	BOOL				Error.
OUTPUT	errorCode	DINT				Error status.

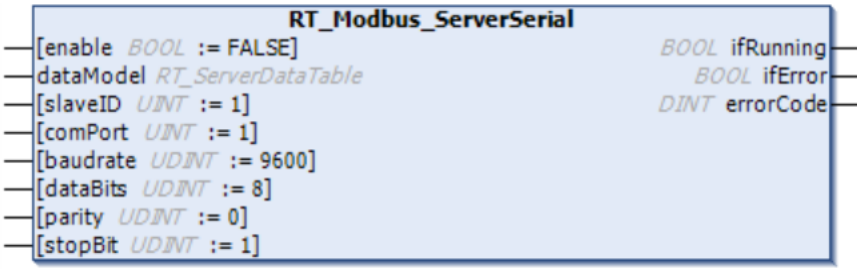


8.4.3 RT_Modbus_ServerSerial

ModbusRTU slave initialization, inherited from RT-ModbusServer.

FUNCTION_BLOCK RT_Modbus_ServerSerial EXTENDS RT_ModbusServer

	Name	Type	Inherited from	Address	Initial	Comment
INPUT	enable	BOOL	RT_ModbusServer		FALSE	Enables the server, take over the configuration information.
IN_OUT	dataModel	RT_ServerDataTable	RT_ModbusServer			Parameters for running the server,memory has to be provided by caller.
INPUT	slaveID	UINT			1	Slave id.
INPUT	comPort	UINT			1	Serial port, only read when rising edge on enable occurs,range[1,2].
INPUT	baudrate	UDINT			9600	Baud rate, only read when rising edge on enable occurs.
INPUT	dataBits	UDINT			8	Number of data bits/BYTE, 4-8, only read when rising edge on enable occurs.
INPUT	parity	UDINT			0	Parity, only read when rising edge on enable occurs,0=connect,1=Odd,2=even.
INPUT	stopBit	UDINT			1	Stop bits, only read when rising edge on connect occurs.
OUTPUT	ifRunning	BOOL	RT_ModbusServer			Server is up and running, waiting for requests.
OUTPUT	ifError	BOOL	RT_ModbusServer			Error.
OUTPUT	errorCode	DINT	RT_ModbusServer			Error status.

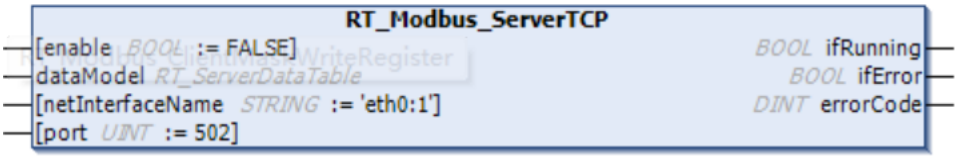


8.4.4 RT_Modbus_ServerTCP

ModbusTCP slave initialization, inherited from RT-ModbusServer.

FUNCTION_BLOCK RT_Modbus_ServerTCP EXTENDS RT_ModbusServer

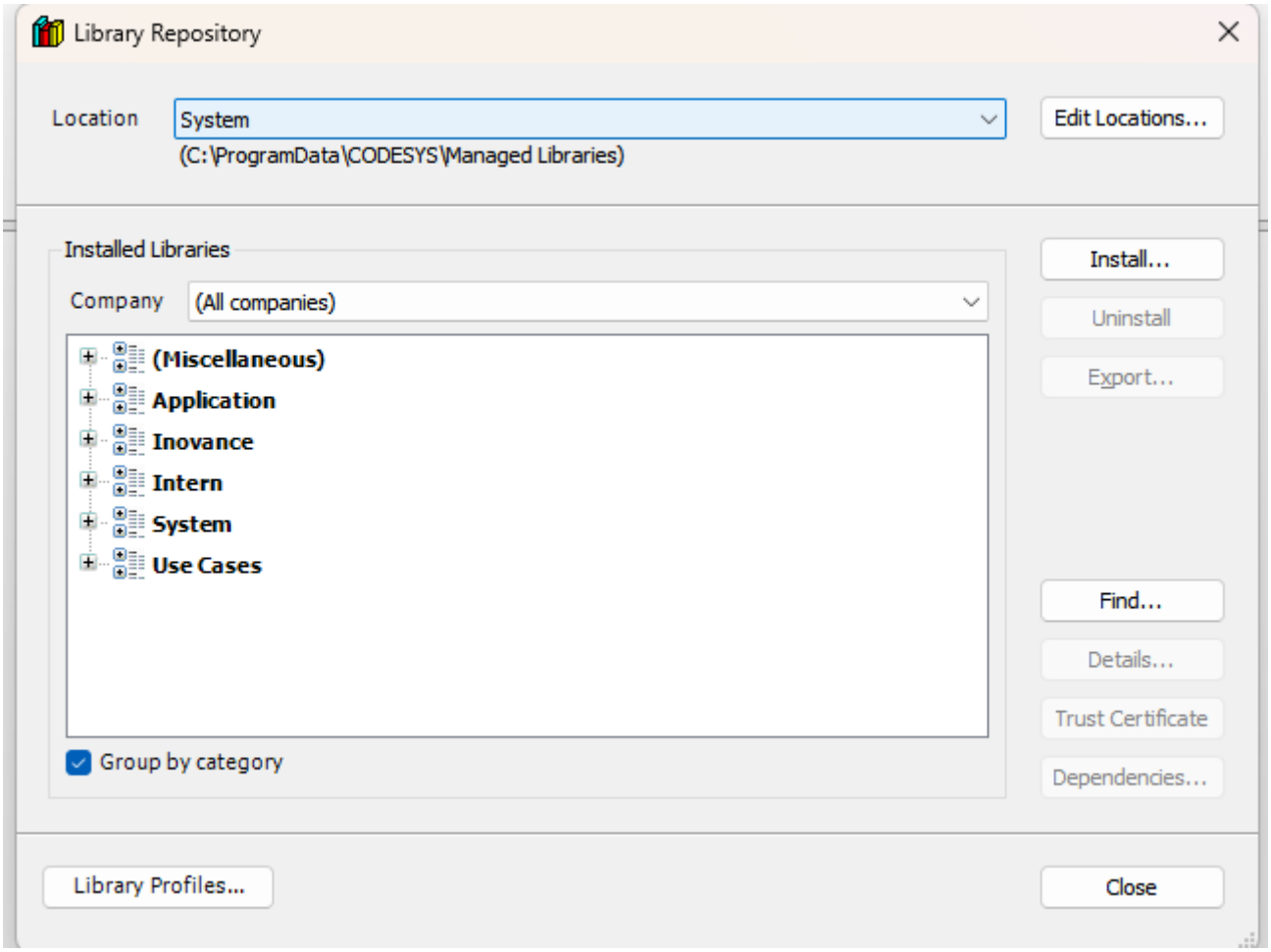
	Name	Type	Inherited from	Address	Initial	Comment
INPUT	enable	BOOL	RT_ModbusServer		FALSE	Enables the server, take over the configuration information.
IN_OUT	dataModel	RT_ServerDataTable	RT_ModbusServer			Parameters for running the server,memory has to be provided by caller.
INPUT	netInterfaceName	STRING			'eth0:1'	ETH interface name to bind to. Only read when rising edge on enable occurs.
INPUT	port	UINT			502	ETH port to use. Only read when rising edge on enable occurs.
OUTPUT	ifRunning	BOOL	RT_ModbusServer			Server is up and running, waiting for requests.
OUTPUT	ifError	BOOL	RT_ModbusServer			Error.
OUTPUT	errorCode	DINT	RT_ModbusServer			Error status.



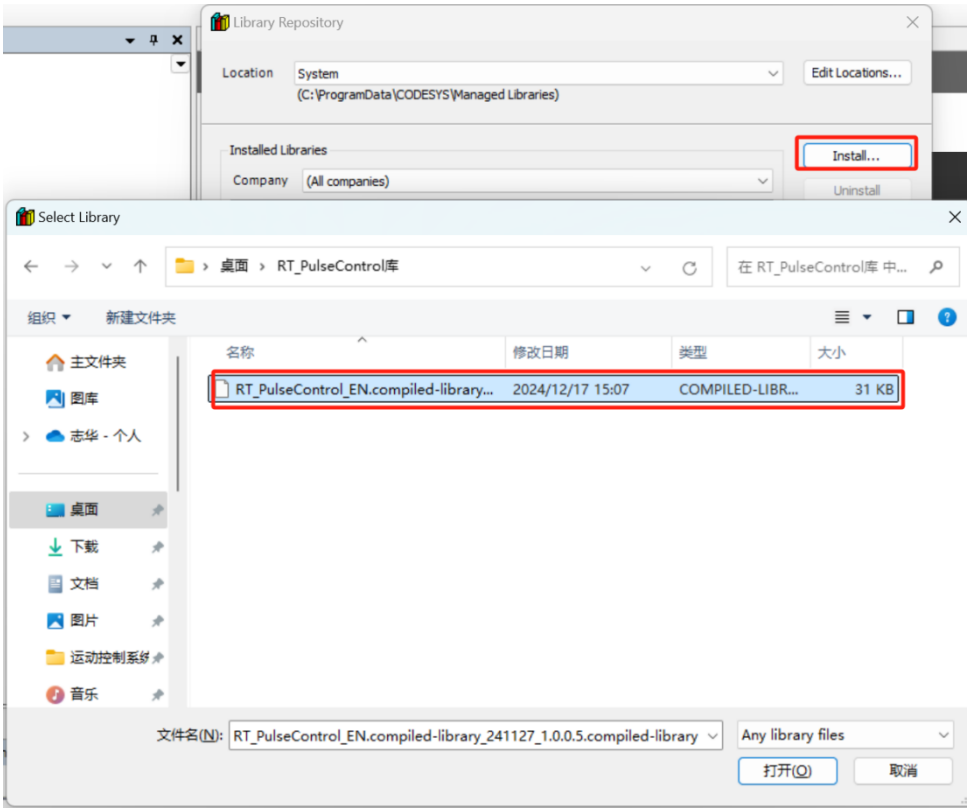
9 Rt_PulseControlFB Description

9.1 Library Installation

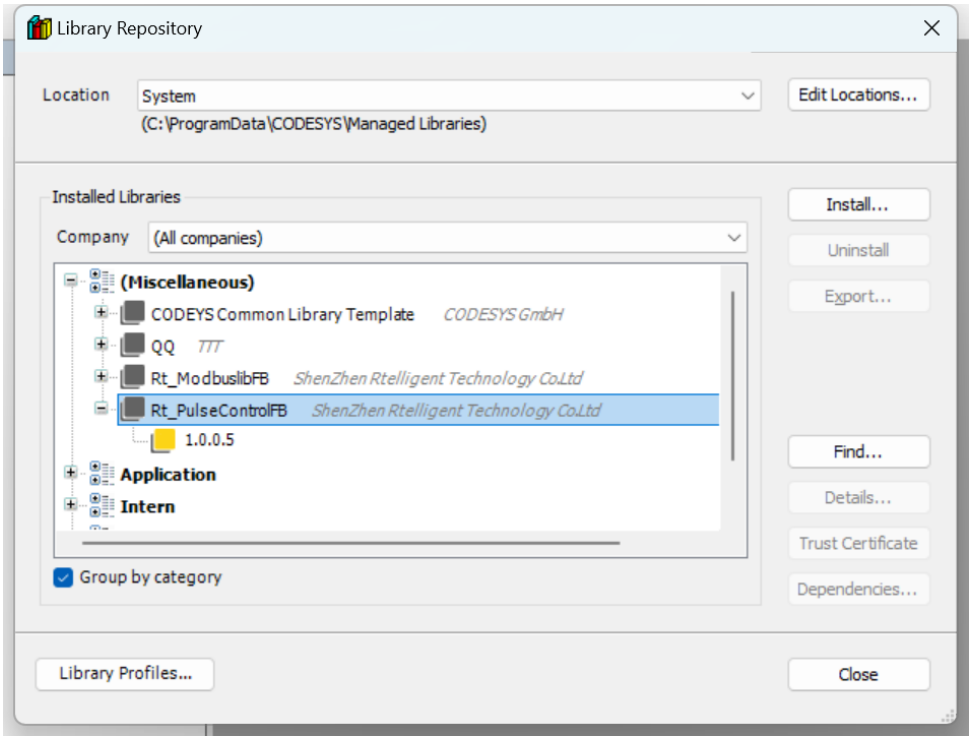
(1) Find Tools -> Library Storage above the CODESYS engineering environment, and click to bring up the following interface:



(2) Click the install button in the interface to install the provided communication library, as shown below:

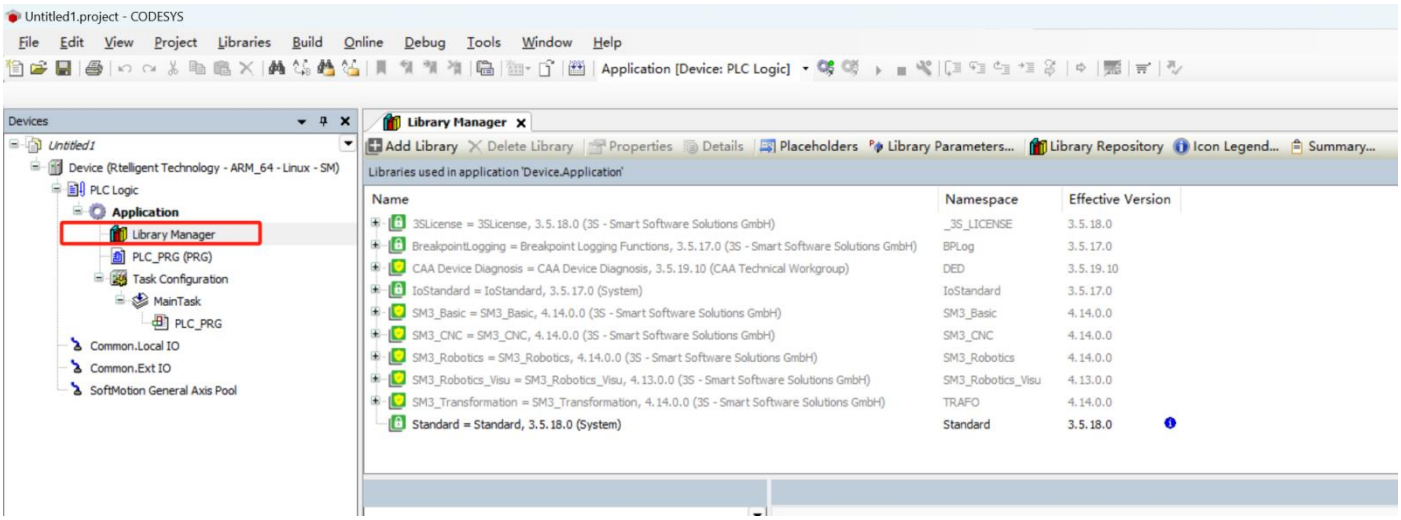


(3) When the installation is completed, the library you installed will appear:

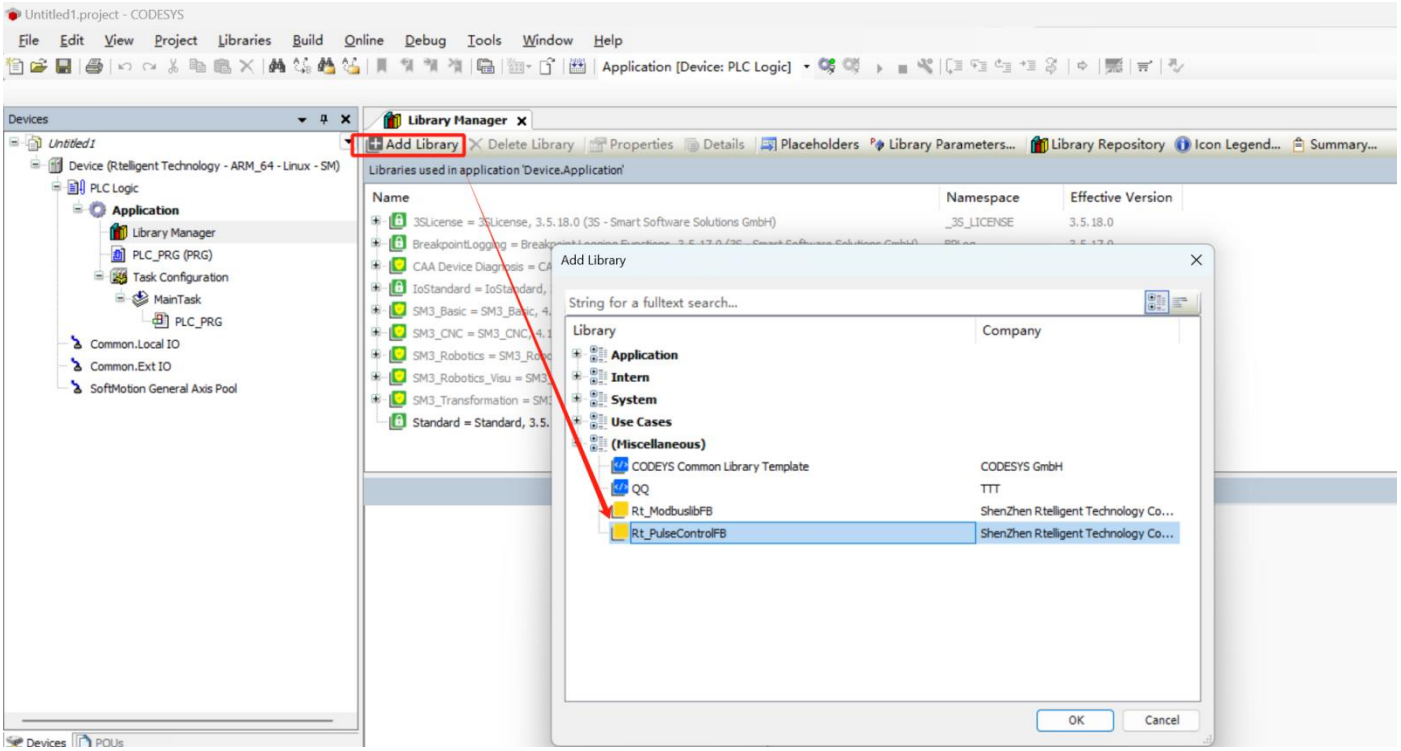


9.2 Project Usage

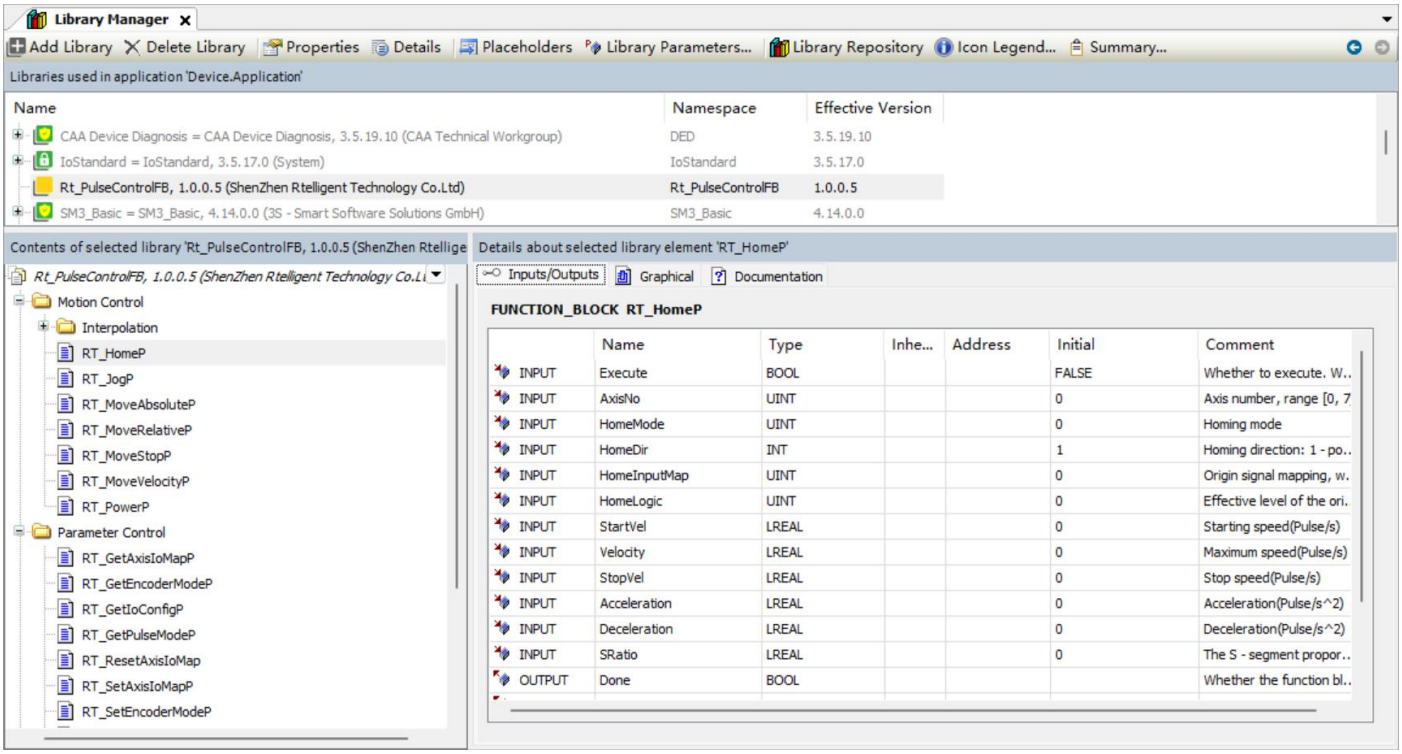
(1) Find the "Library Manager" in the project and double-click to display the following interface.



(2) Click "Add Library" to find the corresponding library in the interface that appears.



(3) At this point, the relevant library can be used in the program, and the function block in the library can be input through F2 (input assistant) in the program, as shown in the following figure.



9.3 Function Block Interface Description

9.3.1 Pulse Axis Initialization Function

9.3.1.1 RT_PulseControlInit

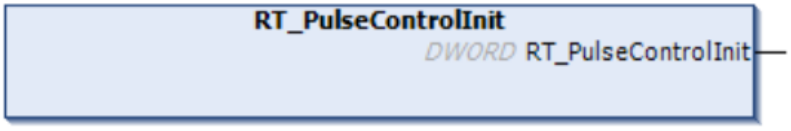
Inputs/Outputs | Graphical | Documentation

FUNCTION **RT_PulseControlInit**

Pulse module initialization
The module must be initialized before using pulse control.
A return value of 0 indicates successful initialization.
If it is called again after initialization, it will return 1801, which means the module has already been initialized.

Name	Type	Inherited from	Address	Initial	Comment
RT_PulseControlInit	DWORD				Return code

Inputs/Outputs | Graphical | Documentation



9.3.1.2 RT_PulseControlFree

FUNCTION **RT_PulseControlFree**

Release of pulse module resources.
Generally, there is no need to call it actively.

Name	Type	Inherited from	Address	Initial	Comment
RT_PulseControlFree	DWORD				Return code

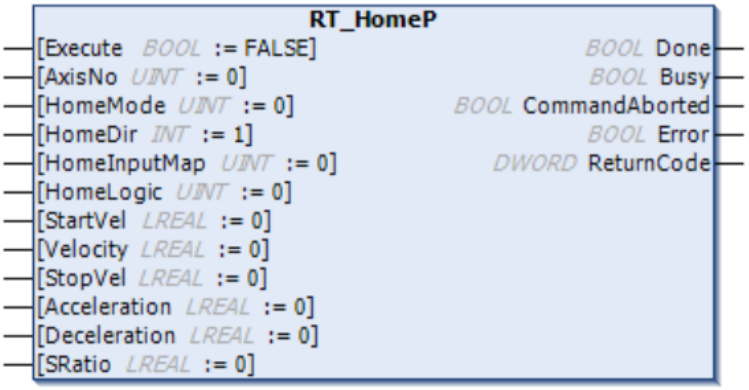
9.3.2 Motion Control Function Block

9.3.2.1 RT_HomeP

FUNCTION_BLOCK **RT_HomeP**

This instruction is used to achieve the homing of the pulse axis.
When Execute changes from False to True, read the parameters and execute once.
Homing mode: 0 - single homing.
1 - single homing + reverse search.
2 - double homing.
Origin signal mapping (HomeInputMap): The value range is [0, 15], corresponding to the general-purpose input signals Input0 - Input15.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
HomeMode	UINT			0	Homing mode
HomeDir	INT			1	Homing direction: 1 - positive direction, -1 - negative direction.
HomeInputMap	UINT			0	Origin signal mapping, with a value range of [0, 15] corresponding to general-purpose input signals Input0 - Input15.
HomeLogic	UINT			0	Effective level of the origin signal: 0 - active low level, 1 - active high level.
StartVel	LREAL			0	Starting speed(Pulse/s)
Velocity	LREAL			0	Maximum speed(Pulse/s)
StopVel	LREAL			0	Stop speed(Pulse/s)
Acceleration	LREAL			0	Acceleration(Pulse/s^2)
Deceleration	LREAL			0	Deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

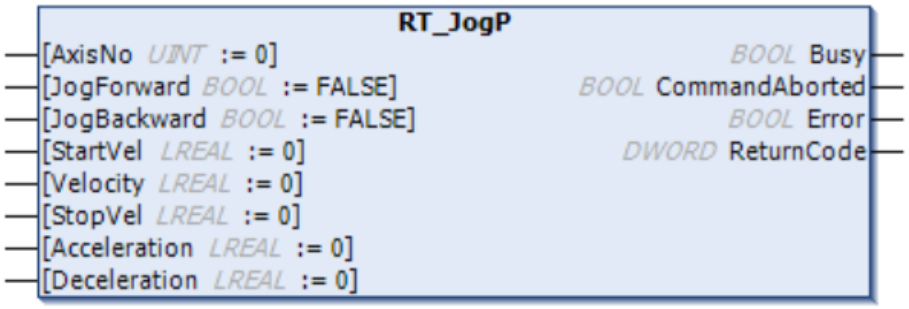


9.3.2.2 RT_JogP

FUNCTION_BLOCK **RT_JogP**

Manually control the axis to move in the specified direction.

Name	Type	Inherited from	Address	Initial	Comment
AxisNo	UINT			0	Axis number, range [0, 7]
JogForward	BOOL			FALSE	If JogForward is TRUE, the axis will move in the positive direction according to the given parameters. If JogBackward is TRUE at the same time, the axis will remain stationary.
JogBackward	BOOL			FALSE	If JogBackward is TRUE, the axis will move in the negative direction according to the given parameters. If JogForward is TRUE at the same time, the axis will remain stationary.
StartVel	LREAL			0	Starting speed(Pulse/s)
Velocity	LREAL			0	Maximum speed(Pulse/s)
StopVel	LREAL			0	Stop speed(Pulse/s)
Acceleration	LREAL			0	Acceleration(Pulse/s^2)
Deceleration	LREAL			0	Deceleration(Pulse/s^2)
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

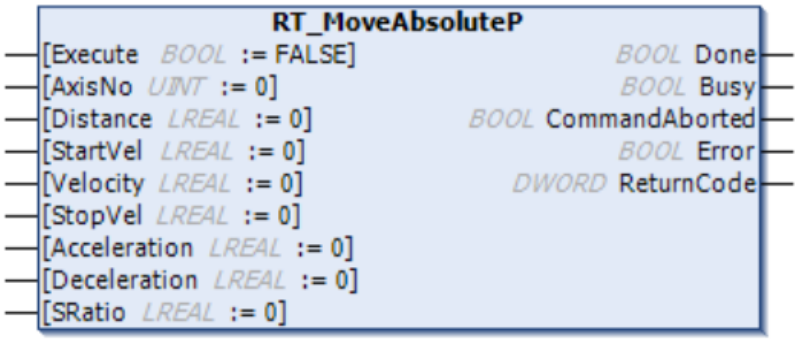


9.3.2.3 RT_MoveAbsoluteP

FUNCTION_BLOCK **RT_MoveAbsoluteP**

Pulse - Axis Single - Axis Absolute - Position Point - to - Point Motion Function Block
When Execute changes from False to True, read the parameters and execute once.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
Distance	LREAL			0	Target pos(Pulse)
StartVel	LREAL			0	Starting speed(Pulse/s)
Velocity	LREAL			0	Maximum speed(Pulse/s)
StopVel	LREAL			0	Stop speed(Pulse/s)
Acceleration	LREAL			0	Acceleration(Pulse/s^2)
Deceleration	LREAL			0	Deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

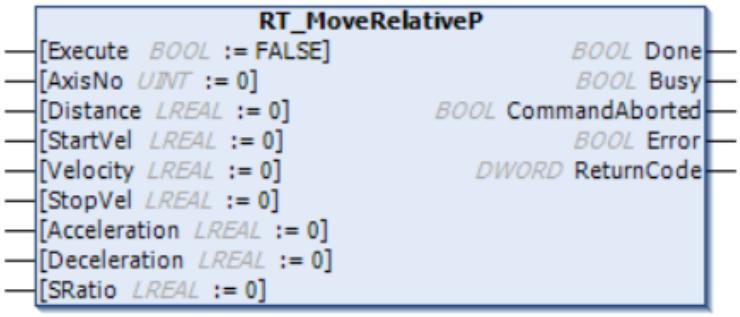


9.3.2.4 RT_MoveRelativeP

FUNCTION_BLOCK **RT_MoveRelativeP**

Pulse - axis single - axis relative - position point - to - point motion function block
When Execute changes from False to True, read the parameters and execute once.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
Distance	LREAL			0	Target pos(Pulse)
StartVel	LREAL			0	Starting speed(Pulse/s)
Velocity	LREAL			0	Maximum speed(Pulse/s)
StopVel	LREAL			0	Stop speed(Pulse/s)
Acceleration	LREAL			0	Acceleration(Pulse/s^2)
Deceleration	LREAL			0	Deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block



9.3.2.5 RT_MoveStopP

FUNCTION_BLOCK **RT_MoveStopP**

Pulse motion stop function block
When Execute changes from False to True, read the parameters and execute once.

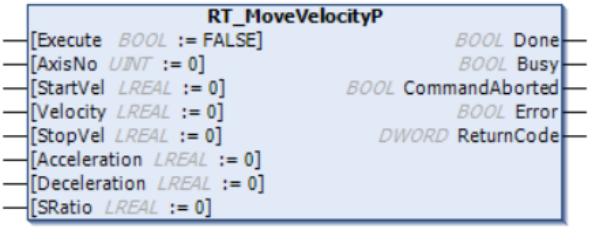
Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
StopMode	UINT			0	Stop mode: 0 - decelerate to stop; 1 - stop immediately.
StopTime	LREAL			0.1	Stop time (s), corresponding to the deceleration stop time in stop mode 0.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

9.3.2.6 RT_MoveVelocityP

FUNCTION_BLOCK **RT_MoveVelocityP**

This instruction is used to make the pulse axis move at the set maximum speed according to the set parameters.
The startup is triggered when Excuse changes from FALSE to TRUE.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
StartVel	LREAL			0	Starting speed(Pulse/s)
Velocity	LREAL			0	Maximum speed(Pulse/s)(The symbol represents the running direction.)
StopVel	LREAL			0	Stop speed(Pulse/s)
Acceleration	LREAL			0	Acceleration(Pulse/s^2)
Deceleration	LREAL			0	Deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

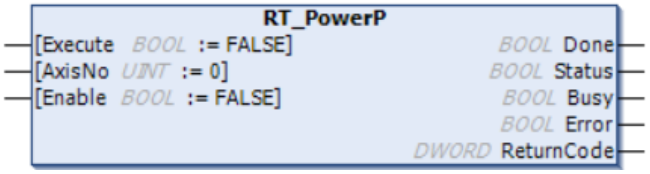


9.3.2.7 RT_PowerP

FUNCTION_BLOCK **RT_PowerP**

Pulse motor axis enable function block
When you want to control the corresponding axis pulse, you must ensure that the axis is enabled, otherwise an error code will be returned.
When Execute changes from False to True, read the parameters and execute once.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
Enable	BOOL			FALSE	Whether the pulse axis is enabled
Done	BOOL				Whether the function block execution is completed
Status	BOOL				Whether the current axis is in the enabled state
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

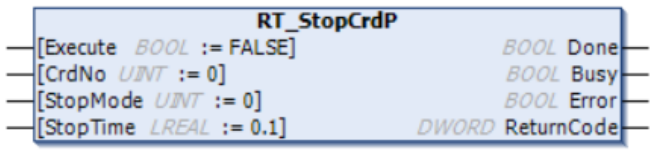


9.3.2.8 RT_StopCrdP

FUNCTION_BLOCK **RT_StopCrdP**

Stop interpolation system motion function block.
When Execute changes from False to True, read the parameters and execute once.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
CrdNo	UINT			0	The interpolation system index ranges from 0 to 3. Different interpolation systems can execute interpolation motions independently.
StopMode	UINT			0	Stop mode: 0 - decelerate to stop; 1 - stop immediately.
StopTime	LREAL			0.1	Stop time (s), corresponding to the deceleration stop time when the stop mode is 0.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

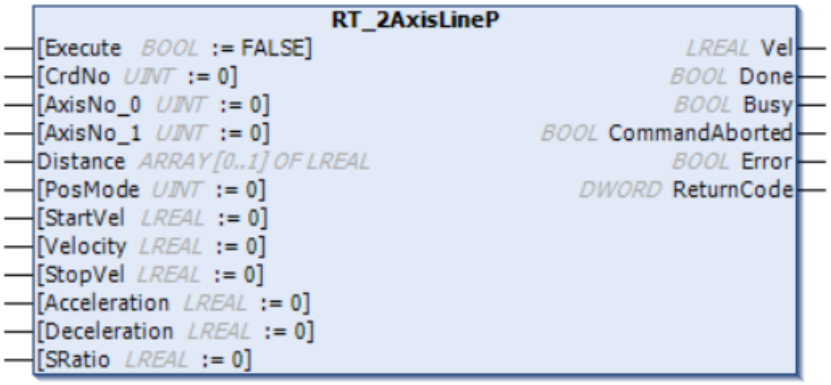


9.3.2.9 RT_2AxisLineP

FUNCTION_BLOCK **RT_2AxisLineP**

Two-axis linear interpolation function block for pulse axes.
When Execute changes from False to True, read the parameters and execute once.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
CrdNo	UINT			0	The interpolation system index ranges from 0 to 3. Different interpolation systems can execute interpolation motions independently.
AxisNo_0	UINT			0	The axis number of the 0-axis participating in interpolation, with a range of [0, 7].
AxisNo_1	UINT			0	The axis number of the 1-axis participating in interpolation, with a range of [0, 7].
Distance	ARRAY [0..1] OF LREAL				The target position of the interpolation axis.(Pulse)
PosMode	UINT			0	Position mode: 0 represents relative motion, and 1 represents absolute motion.
StartVel	LREAL			0	Interpolation start speed.(Pulse/s)
Velocity	LREAL			0	Interpolation max speed.(Pulse/s)
StopVel	LREAL			0	Interpolation stop speed.(Pulse/s)
Acceleration	LREAL			0	Interpolation acceleration(Pulse/s^2)
Deceleration	LREAL			0	Interpolation deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Vel	LREAL				The speed of the current interpolation system motion (Pulse/s).
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

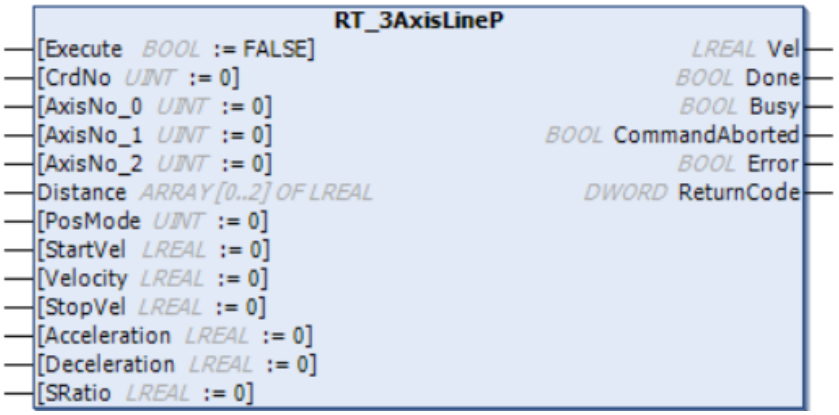


9.3.2.10 RT_3AxisLineP

FUNCTION_BLOCK **RT_3AxisLineP**

Three-axis linear interpolation function block for pulse axes.
When Execute changes from False to True, read the parameters and execute once.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
CrdNo	UINT			0	The interpolation system index ranges from 0 to 3. Different interpolation systems can execute interpolation motions independently.
AxisNo_0	UINT			0	The axis number of the 0-axis participating in interpolation, with a range of [0, 7].
AxisNo_1	UINT			0	The axis number of the 1-axis participating in interpolation, with a range of [0, 7].
AxisNo_2	UINT			0	The axis number of the 2-axis participating in interpolation, with a range of [0, 7].
Distance	ARRAY [0..2] OF LREAL				The target position of the interpolation axis.(Pulse)
PosMode	UINT			0	Position mode: 0 represents relative motion, and 1 represents absolute motion.
StartVel	LREAL			0	Interpolation start speed.(Pulse/s)
Velocity	LREAL			0	Interpolation max speed.(Pulse/s)
StopVel	LREAL			0	Interpolation stop speed.(Pulse/s)
Acceleration	LREAL			0	Interpolation acceleration(Pulse/s^2)
Deceleration	LREAL			0	Interpolation deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Vel	LREAL				The speed of the current interpolation system motion (Pulse/s).
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block



9.3.2.11 RT_2AxisCircleC

FUNCTION_BLOCK **RT_2AxisCircleC**

Two-axis circular arc interpolation (endpoint and center) function block for pulse axes.
When Execute changes from False to True, read the parameters and execute once.

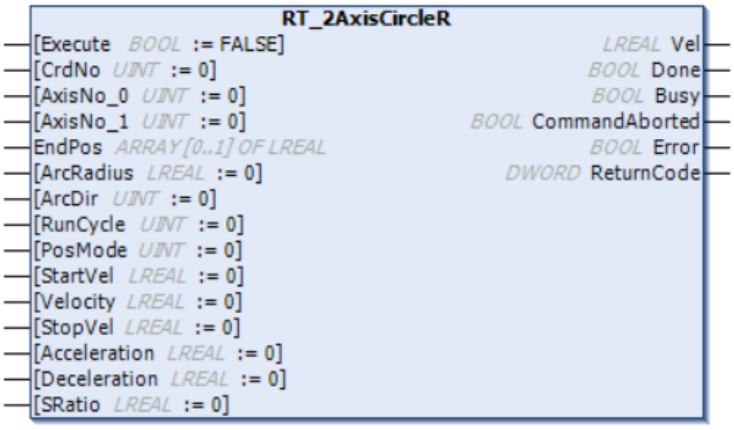
Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
CrdNo	UINT			0	The interpolation system index ranges from 0 to 3. Different interpolation systems can execute interpolation motions independently.
AxisNo_0	UINT			0	The axis number of the 0-axis participating in interpolation, with a range of [0, 7].
AxisNo_1	UINT			0	The axis number of the 1-axis participating in interpolation, with a range of [0, 7].
EndPos	ARRAY [0..1] OF LREAL				The target position of the interpolation axis.(Pulse)
CenPos	ARRAY [0..1] OF LREAL				The center position of the interpolation circular arc.
ArcDir	UINT			0	Interpolation direction: 0 - clockwise, 1 - counterclockwise.
RunCycle	UINT			0	The number of rotations of the circular arc.
PosMode	UINT			0	Position mode: 0 represents relative motion, and 1 represents absolute motion.
StartVel	LREAL			0	Interpolation start speed.(Pulse/s)
Velocity	LREAL			0	Interpolation max speed.(Pulse/s)
StopVel	LREAL			0	Interpolation stop speed.(Pulse/s)
Acceleration	LREAL			0	Interpolation acceleration(Pulse/s^2)
Deceleration	LREAL			0	Interpolation deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Vel	LREAL				The speed of the current interpolation system motion (Pulse/s).
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

9.3.2.12 RT_2AxisCircleR

FUNCTION_BLOCK **RT_2AxisCircleR**

Two-axis circular arc interpolation (endpoint radius) function block for pulse axes.
When Execute changes from False to True, read the parameters and execute once.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
CrdNo	UINT			0	The interpolation system index ranges from 0 to 3. Different interpolation systems can execute interpolation motions independently.
AxisNo_0	UINT			0	The axis number of the 0-axis participating in interpolation, with a range of [0, 7].
AxisNo_1	UINT			0	The axis number of the 1-axis participating in interpolation, with a range of [0, 7].
EndPos	ARRAY [0..1] OF LREAL				The target position of the interpolation axis.(Pulse)
ArcRadius	LREAL			0	Arc radius
ArcDir	UINT			0	Interpolation direction: 0 - represents a minor arc, 1 - represents a major arc.
RunCycle	UINT			0	The number of rotations for circular arc rotation.
PosMode	UINT			0	Position mode: 0 represents relative motion, and 1 represents absolute motion.
StartVel	LREAL			0	Interpolation start speed.(Pulse/s)
Velocity	LREAL			0	Interpolation max speed.(Pulse/s)
StopVel	LREAL			0	Interpolation stop speed.(Pulse/s)
Acceleration	LREAL			0	Interpolation acceleration(Pulse/s^2)
Deceleration	LREAL			0	Interpolation deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Vel	LREAL				The speed of the current interpolation system motion (Pulse/s).
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

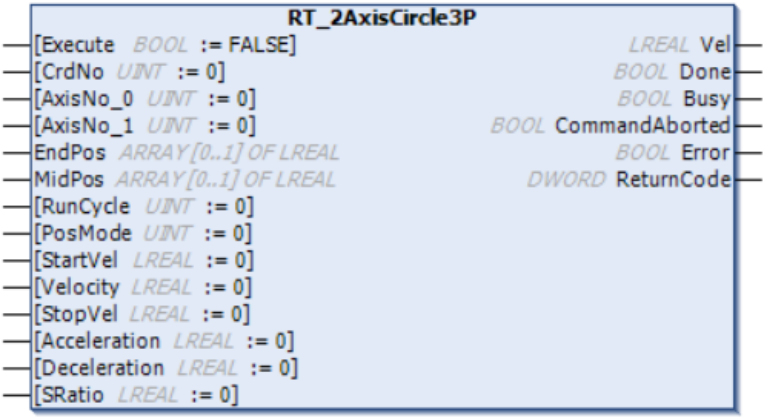


9.3.2.13 RT_2AxisCircle3P

FUNCTION_BLOCK **RT_2AxisCircle3P**

Two-axis circular arc interpolation (three-point circular arc) function block for pulse axes.
When Execute changes from False to True, read the parameters and execute once.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
CrdNo	UINT			0	The interpolation system index ranges from 0 to 3. Different interpolation systems can execute interpolation motions independently.
AxisNo_0	UINT			0	The axis number of the 0-axis participating in interpolation, with a range of [0, 7].
AxisNo_1	UINT			0	The axis number of the 1-axis participating in interpolation, with a range of [0, 7].
EndPos	ARRAY [0..1] OF LREAL				The target position of the interpolation axis.(Pulse)
MidPos	ARRAY [0..1] OF LREAL				The midpoint of a circular arc. A circular arc can be determined by three points, namely the starting point position, the midpoint position and the target position.
RunCycle	UINT			0	The number of rotations of the circular arc.
PosMode	UINT			0	Position mode: 0 represents relative motion, and 1 represents absolute motion.
StartVel	LREAL			0	Interpolation start speed.(Pulse/s)
Velocity	LREAL			0	Interpolation max speed.(Pulse/s)
StopVel	LREAL			0	Interpolation stop speed.(Pulse/s)
Acceleration	LREAL			0	Interpolation acceleration(Pulse/s^2)
Deceleration	LREAL			0	Interpolation deceleration(Pulse/s^2)
SRatio	LREAL			0	The S - segment proportion of the acceleration and deceleration section, with a range of [0,1]. When the value is 0, it is a trapezoidal acceleration - deceleration curve.
Vel	LREAL				The speed of the current interpolation system motion (Pulse/s).
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
CommandAborted	BOOL				It will be TRUE if this command has been terminated by other commands.
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block



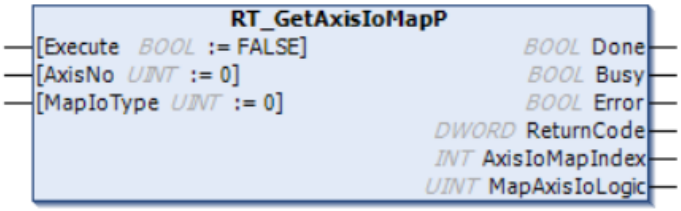
9.3.3 Pulse Axis Parameter Setting Function Block

9.3.3.1 RT_GetAxisIoMapP

FUNCTION_BLOCK **RT_GetAxisIoMapP**

This function block is used to obtain the mapping status of the dedicated signals of the pulse axis.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
MapIoType	UINT			0	IO Type: 0 - Negative Limit, 1 - Positive Limit
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block
AxisIoMapIndex	INT				Axis-specific signal mapping index, with a range of [0, 15] corresponding to the general-purpose input signals Input0 - Input15; -1 indicates that it has not been mapped.
MapAxisIoLogic	UINT			0	Axis-specific signal logic: 0 - active low; 1 - active high.

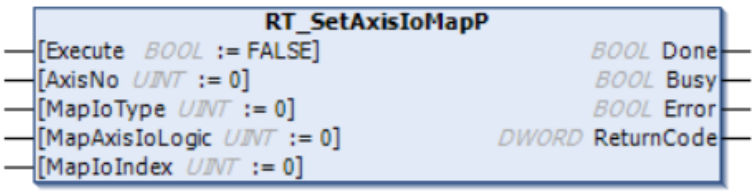


9.3.3.2 RT_SetAxisIoMapP

FUNCTION_BLOCK **RT_SetAxisIoMapP**

This function block is used to set the dedicated signal mapping of the pulse axis.
General - purpose input signals can be flexibly configured for use as limit signals of the pulse axis.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
MapIoType	UINT			0	IO type: 0 - negative limit, 1 - positive limit.
MapAxisIoLogic	UINT			0	Axis-specific signal logic: 0 - active low; 1 - active high.
MapIoIndex	UINT			0	Mapping index, with a value range of [0, 15] corresponding to the general-purpose input signals Input0 - Input15.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

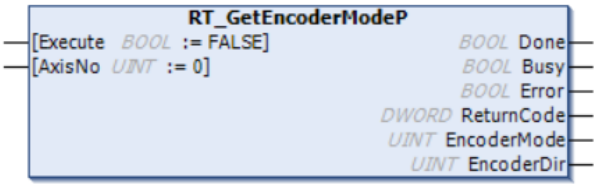


9.3.3.3 RT_GetEncoderModeP

FUNCTION_BLOCK **RT_GetEncoderModeP**

This function block is used to obtain the encoder counter mode.
Encoder mode:
0 - Quadruple frequency counting for A and B phases.
1 - Single-phase counting.
Encoder counter direction:
0 - Counting in the forward direction.
1 - Counting in the negative direction.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block
EncoderMode	UINT				Encoder mode
EncoderDir	UINT				Encoder dir

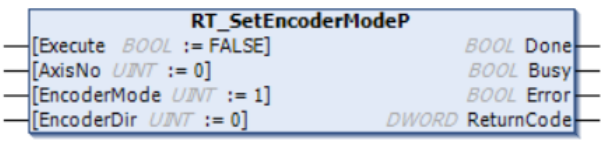


9.3.3.4 RT_SetEncoderModeP

FUNCTION_BLOCK **RT_SetEncoderModeP**

This function block is used to set the encoder counter mode.
Encoder mode:
0 - Quadruple frequency counting for A and B phases.
1 - Single-phase counting.
Encoder counter direction:
0 - Counting in the forward direction.
1 - Counting in the negative direction.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
EncoderMode	UINT			1	Encoder counting mode, with the default being 1 - single-phase counting.
EncoderDir	UINT			0	Encoder counting direction, with the default being 0 - counting in the forward direction.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block



9.3.3.5 RT_GetIoConfigP

This function block is used to obtain whether the Output port is enabled for pulse output function:

AxisOutput_0	Corresponding output ports Y0&Y1; If it is true, the output port is used for pulse output of axis 0. If it is false, the output port is used for normal output
AxisOutput_1	Corresponding output ports Y2&Y3; If it is true, the output port is used for pulse output of axis 1. If it is false, the output port is used for normal output
AxisOutput_2	Corresponding output ports Y4&Y5; If it is true, the output port is used for pulse output of axis 2. If it is false, the output port is used for normal output
AxisOutput_3	Corresponding output ports Y6&Y7; If it is true, the output port is used for pulse output of axis 3. If it is false, the output port is used for normal output
AxisOutput_4	Corresponding output ports Y10&Y11; If it is true, the output port is used for pulse output of axis 4. If it is false, the output port is used for normal output
AxisOutput_5	Corresponding output ports Y12&Y13; If it is true, the output port is used for pulse output of axis 5. If it is false, the output port is used for normal output
AxisOutput_6	Corresponding output ports Y14&Y15; If it is true, the output port is used for pulse output of axis 6. If it is false, the output port is used for normal output
AxisOutput_7	Corresponding output ports Y16&Y17; If it is true, the output port is used for pulse output of axis 7. If it is false, the output port is used for normal output

This function block is used to obtain whether the Input port is enabled for encoder counting function:

AxisInput_0	Corresponding input ports X0&X1; If it is true, the input port is used for encoder input of axis 0. If it is false, the input port is used as a normal input
AxisInput_1	Corresponding input ports X2&X3; If it is true, the input port is used for encoder input of axis 1. If it is false, the input port is used as a normal input
AxisInput_2	Corresponding input ports X4&X5; If it is true, the input port is used for encoder input of axis 2. If it is false, the input port is used as a normal input
AxisInput_3	Corresponding input ports X6&X7; If it is true, the input port is used for encoder input of axis 3. If it is false, the input port is used as a normal input
AxisInput_4	Corresponding input ports X10&X11; If it is true, the input port is used for encoder input of axis 4. If it is false, the input port is used as a normal input
AxisInput_5	Corresponding input ports X12&X13; If it is true, the input port is used for encoder input of axis 5. If it is false, the input port is used as a normal input
AxisInput_6	Corresponding input ports X14&X15; If it is true, the input port is used for encoder input of axis 6. If it is false, the input port is used as a normal input
AxisInput_7	Corresponding input ports X16&X17; If it is true, the input port is used for encoder input of axis 7. If it is false, the input port is used as a normal input

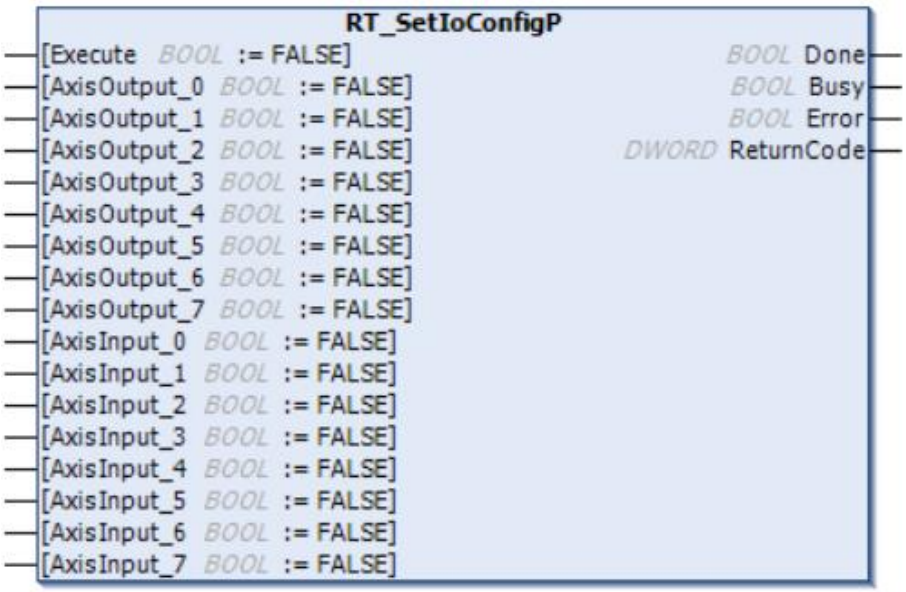
9.3.3.6 RT_SetIoConfigP

This function block is used to obtain whether the Output port is enabled for pulse output function:

AxisOutput_0	Corresponding output ports Y0&Y1; The default value is False, this output port is used for normal output, and when it is True, it is used for pulse output of axis 0
AxisOutput_1	Corresponding output ports Y2&Y3; The default value is False, this output port is used for normal output, and when it is True, it is used for pulse output of axis 1
AxisOutput_2	Corresponding output ports Y4&Y5; The default value is False, this output port is used for normal output, and when it is True, it is used for pulse output of axis 2
AxisOutput_3	Corresponding output ports Y6&Y7; The default value is False, this output port is used for normal output, and when it is True, it is used for pulse output of axis 3
AxisOutput_4	Corresponding output ports Y10&Y11; The default value is False, this output port is used for normal output, and when it is True, it is used for pulse output of axis 4
AxisOutput_5	Corresponding output ports Y12&Y13; The default value is False, this output port is used for normal output, and when it is True, it is used for pulse output of axis 5
AxisOutput_6	Corresponding output ports Y14&Y15; The default value is False, this output port is used for normal output, and when it is True, it is used for pulse output of axis 6
AxisOutput_7	Corresponding output ports Y16&Y17; The default value is False, this output port is used for normal output, and when it is True, it is used for pulse output of axis 7

This function block is used to obtain whether the Input port is enabled for encoder counting function:

AxisInput_0	Corresponding output ports X0&X1; The default value is False, this output port is used for normal output, and when it is True, it is used for encoder counting of axis 0
AxisInput_1	Corresponding output ports X2&X3; The default value is False, this output port is used for normal output, and when it is True, it is used for encoder counting of axis 1
AxisInput_2	Corresponding output ports X4&X5; The default value is False, this output port is used for normal output, and when it is True, it is used for encoder counting of axis 2
AxisInput_3	Corresponding output ports X6&X7; The default value is False, this output port is used for normal output, and when it is True, it is used for encoder counting of axis 3
AxisInput_4	Corresponding output ports X10&X11; The default value is False, this output port is used for normal output, and when it is True, it is used for encoder counting of axis 4
AxisInput_5	Corresponding output ports X12&X13; The default value is False, this output port is used for normal output, and when it is True, it is used for encoder counting of axis 5
AxisInput_6	Corresponding output ports X14&X15; The default value is False, this output port is used for normal output, and when it is True, it is used for encoder counting of axis 6
AxisInput_7	Corresponding output ports X16&X17; The default value is False, this output port is used for normal output, and when it is True, it is used for encoder counting of axis 7

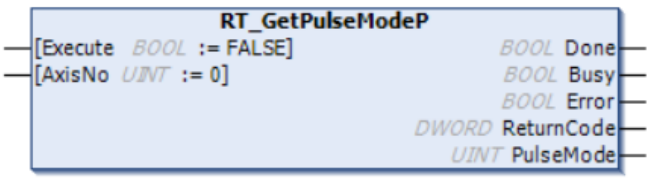


9.3.3.7 RT_GetPulseModeP

FUNCTION_BLOCK **RT_GetPulseModeP**

This function block is used to obtain the pulse mode of the pulse axis.
Pulse output mode:
0 - High pulse + High direction.
1 - Low pulse + High direction.
2 - High pulse + Low direction.
3 - Low pulse + Low direction.
4 - Double high pulses.
5 - Double low pulses.
6 - A and B phases.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block
PulseMode	UINT				The current pulse mode of the pulse - axis.

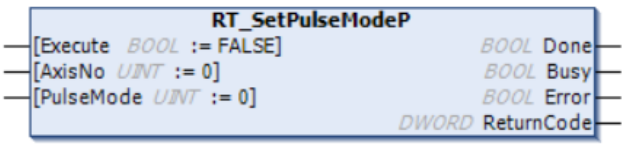


9.3.3.8 RT_SetPulseModeP

FUNCTION_BLOCK **RT_SetPulseModeP**

This function block is used to set the pulse mode of the pulse axis.
Pulse output mode:
0 - High pulse + High direction.
1 - Low pulse + High direction.
2 - High pulse + Low direction.
3 - Low pulse + Low direction.
4 - Double high pulses.
5 - Double low pulses.
6 - A and B phases.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
PulseMode	UINT			0	Pulse output mode, with the default being 0 - high pulse + high direction.
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block



9.3.3.9 RT_ResetAxisIoMap

FUNCTION_BLOCK **RT_ResetAxisIoMap**

This function block is used to reset the dedicated signal mapping of the pulse axis. After the reset, all the dedicated signals of the axis are unconfigured and invalid.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

9.3.3.10 RT_SetPositionP

FUNCTION_BLOCK **RT_SetPositionP**

This function block is used to set the current position of the pulse axis.

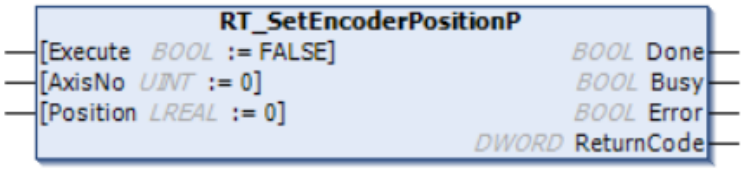
Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
Position	LREAL			0	Input position(Pulse)
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block

9.3.3.11 RT_SetEncoderPositionP

FUNCTION_BLOCK **RT_SetEncoderPositionP**

This function block is used to set the current position of the encoder.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute. When it changes from False to True, read the parameters and execute.
AxisNo	UINT			0	Axis number, range [0, 7]
Position	LREAL			0	Input position(Pulse)
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block



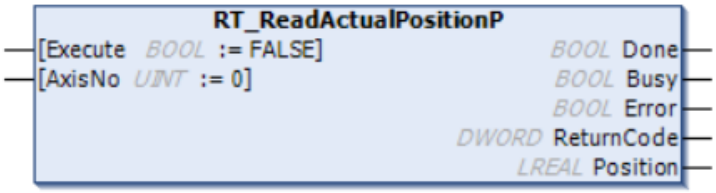
9.3.4 Pulse Axis Information Acquisition Function Block

9.3.4.1 RT_ReadActualPositionP

FUNCTION_BLOCK **RT_ReadActualPositionP**

This function block is used to read the current actual position value of the pulse axis.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute or not. When it is TRUE, the actual position of the axis will be continuously read.
AxisNo	UINT			0	Axis number, range [0, 7]
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block
Position	LREAL				The current actual position of the axis.(Pulse)

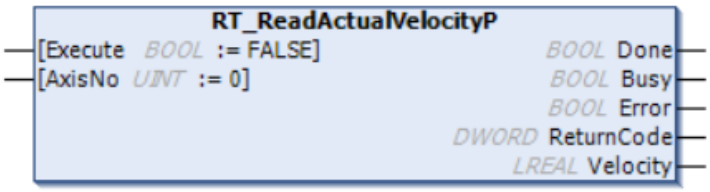


9.3.4.2 RT_ReadActualPVelocityP

FUNCTION_BLOCK **RT_ReadActualVelocityP**

This function block is used to read the actual speed value of the pulse axis.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute or not. When it is TRUE, the current speed of the axis will be continuously read.
AxisNo	UINT			0	Axis number, range [0, 7]
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block
Velocity	LREAL				The current actual speed of the axis.(Pulse/s)

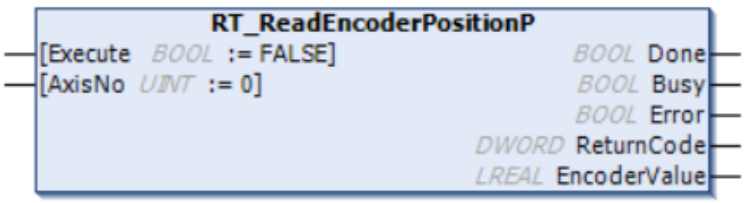


9.3.4.3 RT_ReadEncoderPositionP

FUNCTION_BLOCK **RT_ReadEncoderPositionP**

This function block is used to read the counting function of the encoder.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute or not. When it is TRUE, the count value of the encoder will be continuously read.
AxisNo	UINT			0	Axis number, range [0, 7]
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block
EncoderValue	LREAL				The count value of the current encoder.(Pulse)



9.3.4.4 RT_ReadStatusP

FUNCTION_BLOCK **RT_ReadStatusP**

This function block is used to read the status of the pulse axis.
The homing status of the axis (HomeStatus): 0 - Homing not completed.
1 - Homing completed, the origin position has been found.
2 - An error occurred during the homing process, and there is no origin signal.
3 - Two origin signals were found on the device during the homing process.
4 - The EZ signal was not found during the homing process.
5 - The direction configured for homing is incorrect.
6 - The limit was reached and the operation stopped during the homing process.
7 - It is beyond the range of the software limit.
100 - Homing was not completed normally.

Name	Type	Inherited from	Address	Initial	Comment
Execute	BOOL			FALSE	Whether to execute or not. When it is TRUE, the current status of the axis will be continuously read.
AxisNo	UINT			0	Axis number, range [0, 7]
Done	BOOL				Whether the function block execution is completed
Busy	BOOL				Whether the function block is running
Error	BOOL				Whether an error has occurred in the function block
ReturnCode	DWORD				The return value of the function block
Enable	BOOL				If the axis status is "Enable", then it is TRUE.
Running	BOOL				If the axis is in motion, then it is TRUE.
HomeStatus	UINT				The homing status of the axis.

10 Appendix

10.1 What do the three types of grounding in electricians mean?

The single point grounding methods for low-voltage electrical equipment can be divided into series single point grounding, parallel single point grounding, and multi branch single point grounding.

Series single point grounding: also known as the first grounding method. Grounding method: Connect the grounding terminals of multiple low-voltage electrical equipment to the same grounding wire near the equipment, and then connect it to the grounding device through this grounding wire. The benefits of this type of grounding are: saving manpower and material resources; The downside is that when there is an open circuit in the common grounding wire, if one device in the grounding system leaks electricity, it will cause voltage to appear on the shells of other devices, posing a threat to personnel safety.

Parallel single point grounding: also known as the second grounding method. Grounding method: Each grounding terminal of the equipment is connected to a grounding wire, and then these several wires are simultaneously connected to the grounding device. The advantage of this grounding method is that when one of the grounding devices in the grounding system experiences an open circuit, it will not cause voltage to the casing of other devices, which is beneficial for ensuring personal safety. The imperfection of this grounding method lies in the fact that if it is an electronic device or other highly sensitive electrical device to high-frequency interference, high-frequency interference from other devices (such as frequency converters, intermediate frequency furnaces, and other thyristor converter devices) will be connected from a common point, causing the equipment to work improperly.

Multi branch single point grounding: this is the third grounding method. Grounding method: Connect the grounding terminal of each device separately to the grounding device. The difference between the grounding method and the second grounding method is that the equipment has a separate grounding body (or alternatively, it is directly connected to the grounding device (or grounding source) closest to the grounding body, and the distance between each equipment on the electrical grounding circuit is relatively long (such as over 50 meters), which effectively avoids mutual electromagnetic interference between equipment. But this method is time-consuming, labor-intensive, and it may not be easy to obtain a separate grounding source.

In normal construction, if conditions permit, it is recommended to use the third grounding method. However, in fact, the second grounding method is commonly used for PLC grounding. As for electromagnetic interference, if there are multiple high-power frequency converters in the cabinet, a single-phase power filter can be installed at the front end of the PLC power supply.