

SC series controller Software Operation Manual



Revised resume

issue date	Manual number	Revised content description	Remark
2021-06	V1.0	first edition	

SC series controller related manuals

➤ SCThe types of series manuals are as follows. Please refer to the corresponding manual according to the application.

➤ Each manual can be downloaded from our homepage<http://www.step-sigriner.com.cn>download.

NO	Manual name	content
1.	SC Series Controller Software Operation Manual	Software installation, configuration, debugging, coding, etc.
2.	SC Series Controller Software Programming Manual	Motion control programming, common programming libraries, instructions, etc.
3.	SC20 Controller Hardware Manual	SC20 related hardware interface, wiring and maintenance
4.	SC30 Controller Hardware Manual	SC30 related hardware interface, wiring and maintenance
5.	SC series controller visual interface operation instructions	Visual interface related operations and programming

Precautions



unenforceable matter;



must do




Danger

	Please do not use it in a place where it is easy to be splashed with water, in an environment with corrosive gas, in an environment with flammable gas, or near combustible materials.	May cause fire, electric shock, malfunction, damage
	Do not use it in places subject to severe vibration and shock.	May cause electric shock, injury, fire
	Do not use the wire when it is soaked in oil or water.	May cause electric shock, malfunction, damage
	Do not place around heating elements such as heaters or large coiled resistors.	May cause fire or malfunction
	Do not perform wiring or operation with wet hands.	May cause electric shock, injury, fire
	Wiring work must be carried out by professional electrical engineers.	Electric shock may result from wiring work performed by persons without relevant expertise
	Please refer to the manual for correct wiring.	Incorrect wiring may result in electric shock, injury, malfunction, damage
	The cables should be properly connected, and the energized parts must be effectively insulated by insulating materials.	Incorrect wiring, short circuit may cause electric shock, fire, malfunction
	Please do a good job of standard installation. Failure to follow the standard installation may cause fire or other personal accidents.	Failure to install may result in injury, electric shock, fire, malfunction, damage
	Please install an external emergency stop circuit to ensure that the operation can be stopped in time and the power supply can be cut off in an emergency.	
	The movement, wiring and inspection of the controller should be carried out on the premise that there is no danger of electric shock after the power supply is cut off and the charging indicator light goes out.	Electric shock may result when work is performed without turning off the power



Notice

	Do not drop it or turn it upside down during transportation and installation work.	May cause injury, malfunction
	Do not stand on the product and do not place heavy objects on the product.	May cause electric shock, injury, malfunction, damage

	Do not place obstacles that obstruct ventilation around the controller peripherals.	possible because a temperature rise that could eventually lead to a fire
	Do not block the heat release holes and do not insert foreign objects.	May cause injury, fire
	Do not subject the product to strong shocks.	may cause malfunction
	Do not turn on and off the main power of the controller frequently.	may cause malfunction
	After the power outage is over and the power supply is restored, it may restart suddenly, so do not get close to the machine. Avoid unexpected situations when restarting and ensure personal safety.	may cause injury
	Do not modify, disassemble or repair by yourself.	May cause fire, electric shock, injury, malfunction
	Please observe the specified installation method and direction.	Injuries, malfunctions may result from improper installation and settings
	The eye bolts of the motor are only used for motor transport, not for machine transport.	If it is used to transport the machine, it may cause injury or malfunction.
	Make sure that the ambient temperature of the controller is within the operating temperature and humidity range.	Injuries, malfunctions may result from improper installation and settings
	The distance between the controller and the inside of the control box and other machines should be set to the specified distance.	
	Please use the specified voltage.	Use outside the rated voltage range may result in electric shock, injury, fire
	Safety devices should be installed to deal with the built-in brake, the idling and locking of the reducer, and the leakage of the reducer grease.	Failure to install may result in damage or contamination
	When an alarm occurs, remove the cause of the alarm and ensure safety, clear the alarm state and restart.	Failure to correct the cause of the error may result in injury

For details, please refer to the hardware manual

content

SC series controller related manuals	2
Precautions	3
content	5
glossary	9
Chapter 1 Overview	11
1.1. SC Series Controller Overview	11
1.2. Overview of STEP Automation Studio	11
1.2.1. Introduction to STEP Automation Studio software	11
1.2.2. Software acquisition and installation requirements	12
1.2.3. Installation steps	13
1.2.4. Uninstalling STEP AS	17
1.3. STEP AS and hardware connection	17
Chapter 2 Quick Start	18
2.1. Start the programming environment	18
2.2. Exit the programming environment	18
2.3. Names of the parts	19
2.3.1. Menu bar	20
2.3.2. Toolbar	25
2.3.3. Navigation Bar Window	28
2.4. Engineering operations	29
2.4.1. Operation process	29
2.4.2. Project Wizard	29
2.4.3. Save the project	33
2.4.4. Open Project	34
2.4.5. Close the project	35
2.4.6. Composition of the project	36
2.4.7. Creating a backup when saving a project	36
2.4.8. Automatically save project files	37
2.4.9. Exporting and importing objects	37
2.4.10. Variable table export and import	40
2.4.11. Export and import project archives	41
2.4.12. Adding objects	42
2.4.13. Adding Devices	45
2.5. Device configuration	46
2.5.1. Overview	46
2.5.2. Toolbar	48
2.5.3. Add master	50
2.5.4. Adding Devices	51
2.5.5. Synchronization disable/enable master	53
2.5.6. Right-click function	54
2.5.7. Custom Layout	56
2.5.8. Open configuration table	56
2.6. Creating a program	56
2.6.1. The process of creating a program	56
2.6.2. Program creation interface	57
2.6.3. Creating Program Objects (POU Objects)	67
2.6.4. Types of programming languages	68
2.6.5. Variables	70
2.6.6. Functions and function blocks	80
2.7. Input Assistant	86

2.7.1. Start the input assistant	86
2.7.2. Coding Assistant	87
Chapter 3 System Configuration	89
3.1. Controller Configuration	89
3.1.1. Communication Settings	89
3.1.2. Applications	91
3.1.3. Backup and Restore	92
3.1.4. Documentation	93
3.1.5. Users and Groups	93
3.1.6. PLC Settings	94
3.1.7. Access Rights	95
3.1.8. Logs	96
3.1.9. Task Configuration	97
3.1.10. Status	97
3.1.11. Information	98
3.1.12. Display language	98
3.1.13. Version display	99
3.1.14. Online Help	100
3.2. EtherCAT configuration	102
3.2.1. Autonomous EtherCAT Master Configuration	102
3.2.2. Default EtherCAT Master Configuration	104
3.2.3. Autonomous EtherCAT Slave Configuration	108
3.2.4. Scanning Devices	115
3.2.5. Common faults of EtherCAT	115
3.3. Modbus serial port configuration	116
3.3.1. Add Modbus Device	116
3.3.2. Modbus Master Configuration	118
3.3.3. Modbus Slave Configuration	119
3.3.4. Modbus common faults	122
3.3.5. ModbusTCP Configuration	123
3.4. CANopen configuration	123
3.4.1. CANbus configuration	123
3.4.2. CANopen Master Configuration	124
3.4.3. CANopen Slave Configuration	126
3.4.4. CANopen communication failure	127
3.5. Local built-in IO configuration	128
3.5.1. Adding Devices	128
3.5.2. SC20 local IO configuration	131
3.6. LocalBus configuration	133
3.6.1. Adding Devices	134
3.6.2. localbus master configuration	134
3.6.3. localbus slave configuration	135
3.7. Pulse axis configuration	137
3.7.1. Add pulse axis master and slave device	138
3.7.2. Configuring the Pulse Axis Slave Device	139
3.7.3. Controlling Pulse Axis Slave Devices	140
Chapter 4 Programming Fundamentals	141
4.1. Direct Address	141
4.2. Variables	142
4.2.1. Variable Definition	142
4.2.2. Variable types	149
4.3. Constants	155
Chapter 5 Programming Languages	158
5.1. Introduction to programming languages supported by STEP AS	158

5.2. Structured Text Language (ST)	158
5.2.1. Expressions	158
5.2.2. ST instruction	159
5.3. Ladder Logic Diagram (LD)	166
5.3.1. Overview	166
5.3.2. Ladder Diagram Elements	166
5.3.3. LD General Settings	170
5.3.4. LD menu commands	172
5.3.5. Drag and drop operation	180
5.3.6. Graphic Display Tool	182
5.3.7. LD debugging	183
Chapter 6 Debugging and Diagnosis	187
6.1. Run/Stop	187
6.1.1. Running and Stopping the Controller	187
6.1.2. Single cycle	189
6.2. Breakpoints	189
6.2.1. Breakpoint setting	189
6.2.2. Execution point setting	190
6.2.3. Call Stack View	192
6.3. Debug operations	193
6.3.1. Writing of values and forcing of values	193
6.3.2. Monitoring	194
6.3.3. Process Control	195
6.3.4. Operating Modes	195
6.4. Monitoring functions	196
6.4.1. Monitoring variables in the declaration editor	196
6.4.2. Monitoring variables in the implementation part of the program	196
6.4.3. Monitoring variables in the watch view	197
6.5. Reset	198
6.5.1. Warm reset/cold reset/reset (PLC initialization)	199
6.5.2. Resetting the device from STEP AS	199
6.6. Device Tracking Function	200
6.6.1. Device Tracking General Features	200
6.6.2. Device tracking analysis function	202
Chapter 7 Axis Operation Control Configuration	208
7.1. Axis operation configuration	208
7.1.1. Basic axis settings	208
7.1.2. Unit Conversion Configuration	209
7.1.3. Automapping settings	210
7.2. Single axis control	210
7.2.1. Enter the axis control page	210
7.2.2. Axis operation and status	211
7.2.3. Multi-axis debugging	213
7.3. Simulating a Servo Drive	214
Chapter VIII Program Editing Method	216
8.1. Structured Text (ST) Programming	216
8.1.1. Syntax of ST programs	217
8.1.2. Commenting out ST programs	218
8.1.3. Calling function blocks	219
8.2. FBD/LD/IL programming	219
8.2.1. FBD/LD/IL Editor	219
◆ conventional components	220
8.3. Sequential Function Chart (SFC) Programming	225
8.3.1. SFC editor	225

8.3.2. Execution order	227
8.3.3. Qualifiers for Actions	227
8.3.4. SFC logo	228
8.3.5. Components "Step" and "Transform"	229
8.3.6. Element "action"	231
8.3.7. Element "Macro"	231
8.4. Continuous Function Chart (CFC)	232
8.4.1. CFC editor	232
8.4.2. Position of the break point in the CFC editor	232
8.4.3. CFC components	232
Chapter 9 Convenient Functions of STEP AS	235
9.1. Quick Access to Simulation Functions	235
9.2. Engineering Comparison Quick Access	235
9.3. 402 axis automatic addition function	235
Chapter 10 Management Library	237
10.1. Libraries to be used in project query	237
10.2. Viewing library functions	238
10.3. Adding Libraries to the Application	240
10.4. Adding Libraries to the Repository	240
10.5. Using Libraries in Programs	241
10.5.1. Graphical programming calls such as LD	241
10.5.2. ST language calls	243
10.5.3. Input Assistant	243
10.6. Development Libraries	244
10.6.1. Development library example	244
10.6.2. Library Versions	247
10.6.3. Library encryption	247
Chapter 11 Safety Functions	249
11.1. User Management	249
11.1.1. Project User Management	249
11.1.2. Creating new users and groups	249
11.1.3. Setting Operation Permissions	252
11.1.4. User login and logout	254
11.1.5. Device User Management	254
11.2. Encryption	257
11.2.1. Encrypting project files	257
11.2.2. Encryption of communication lines	258
11.3. Security function: write protection	259
11.3.1. Open as read-only	259
11.3.2. Setting the Published Flag	260
After-sales service	261

glossary

■ IEC61131

The International Standard IEC 61131 is the International Electrotechnical Commission (IEC) a worldwide standard that defines the programming language for PLCs. It defines the following 5 programming languages.

- ◆ Ladder Diagram (LD)
- ◆ Structured Text (ST)
- ◆ Sequential Function Chart (SFC)
- ◆ Function Block Diagram (FBD)
- ◆ Instruction List (IL)
- ◆ Continuous Function Chart CFC

■ EtherCAT

An industrial real-time Ethernet. Included in the IEC61784 international standard.

■ STEP AS

The full name is STEP Automation Studio. Integrated development and debugging tools for STEP controllers.

■ automatic running

It is an operation mode to automatically act upon application.

■ run manually

Refers to the operation mode that is activated during initial startup or debugging. Return-to-origin, JOG operation, and pulser operation belong to this category.

■ Return to origin

The reference position used for positioning is called origin, and moving to this position is called origin return. Move to the origin of the preset reference position, and set the coordinates there as absolute position zero. In addition, when the limit (+) input and the limit (-) input are input, the rotation of the motor is automatically reversed, the origin and near origin are searched, and the origin return operation is performed automatically.

■ JOG operation

Refers to the manual operation mode, running a single instruction or a single step to make the motor or system run.

■ Limit (+), Limit (-) input

This is the limit switch input used to set the limit during motor operation (movement). The limit (+) input indicates the limit point on the increasing side of the passing value, and the

limit (-) input indicates the limit point on the decreasing side of the passing value.

■ **soft limit**

Limits on the software can be set for the absolute coordinates managed in the SC series controller. If the software limit range is exceeded, an error will be reported, and a deceleration stop will be performed. The deceleration time can be set individually.

■ **Torque control**

The output torque of the servo amplifier can be arbitrarily limited.

■ **Linear interpolation**

In position control, in the Cartesian space of the axis group, the motion trajectory will be linearly controlled by interpolation control. Take into account the combined speed and the split axis speed limit for planning.

■ **Circular interpolation**

In the position control, in the Cartesian space of the axis group, the motion trajectory will be controlled by the interpolation control in an arc state. Take into account the combined speed and the split axis speed limit for planning.

■ **edge detection**

It is one of the detection methods of the request signal assigned by this unit, and executes various request processing using the rising edge when the request signal is ON as a trigger.

① Note: The next request cannot be received until the request signal is turned OFF.

第一章 Overview

1.1. SC Series Controller Overview

SC series controllers are modular programmable logic controller series, which can provide users with intelligent and automated solutions. It mainly includes SC20 single-machine small controller series and SC30 medium and large controller series, which adopt rack-type layout, and each rack supports local expansion modules. For details, please refer to "SC20 Controller Hardware Manual" and "SC30 Controller Hardware Manual".

1.2. Overview of STEP Automation Studio

1.2.1. Introduction to STEP Automation Studio software

STEP Automation Studio (STEP AS) is the standard software for development and application of SC series programmable controller products. It is optimized and developed based on STEP AS AP V3 platform, provides a complete configuration, programming, debugging, monitoring environment for SC series programmable controllers, and can handle the powerful IEC 61131-3 language flexibly and freely. At the same time, it provides the functions required for the solution such as UI configuration, PLCOpen Motion library, and technology package.

1. The project and equipment management can be completed through STEP AS, providing the following configuration contents for SC series products:

- ◆ Application configuration
- ◆ CPUconfigure
- ◆ local high speedI/Oor remoteI/OModule configuration
- ◆ EtherCAT/Modbus/CANOpenEqual bus configuration and connected station configuration
- ◆ Visual interface

2. It can complete the functions of program writing, downloading and debugging:

- ◆ Support standardized programming(meets theIEC61131-3standard)All six programming languages:structured text(ST), function block diagram(FBD), command list(IL), Ladder Diagram(LD), sequential function diagram(SFC)and Extended

Programming Language Continuous Function Charts(CFC)

- ◆ Flexible and comprehensive function block library, and supports user-defined library
- ◆ Offline simulation function
- ◆ Intelligent debugging and error-checking function, supports sampling tracking and curve display of variables, and supports complex operations such as time domain and frequency domain
- ◆ Precompile and compile error checking
- ◆ Diagnostics and Logs

1.2.2. Software acquisition and installation requirements

1) Software acquisition

The user programming software STEP AS of STEP SC series medium-sized programmable controller is free software, installation files and reference materials of SC series products, etc., users can obtain through the following channels:

- ◆ Obtain the software installation CD from all levels of distributors of Newstar
- ◆ Download the software installation package for free from the "Download Center" page of the official website of Step Sigriner: <http://www.step-sigriner.com.cn/>
- ◆ Contact the appropriate technician to obtain
- ◆ Since STEP is constantly improving its products and materials, it is recommended that users update the software version in time when necessary, and consult the latest released reference materials, which is beneficial to the user's application design.

2) Software installation requirements

A desktop or portable PC with the following:

- ◆ Windows7/Windows10 operating system (64-bit recommended)
- ◆ CPU frequency: 2GHz or above
- ◆ RAM: 4GB or higher
- ◆ Space: 5GB or more of available hard disk space

1.2.3. installation steps

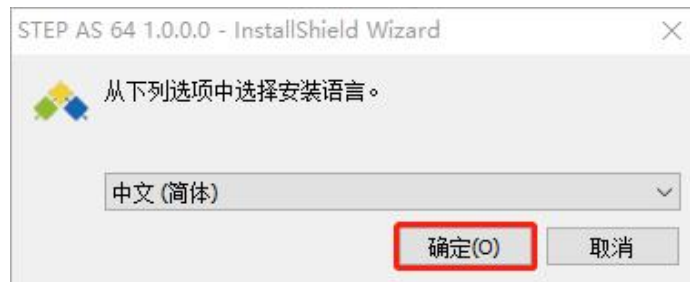
1. Preparation before installation

- a) If it is the first time to install STEP AS, please check the remaining space of the computer hard disk, confirm that the remaining space of the target disk to be installed has more than 5GB, and then install it directly.
- b) If you are upgrading and installing STEP AS, please back up your existing working files, uninstall the old version of STEP AS, restart the computer, and then start installing the new version of the software.
- c) When installing to a PC, log in to the PC with Administrator privileges.
- d) It is best to exit 360 and other anti-virus software before installation, otherwise an error will be reported during the installation process.
- e) Please avoid using Chinese installation paths.

2. Start the installation

Please double click on "STEP AS *.*.*.exe" (the * part varies depending on the version).

①The following screen will be displayed, click [OK].



②The following screen will be displayed, click [Next].



③The following screen will be displayed. After confirming the content, select [I accept the terms of this license agreement], and then click [Next].



④The following screen will be displayed. If you need to change the installation destination folder, click [Change] to specify the installation destination folder. If not, please click [Next].



①Note: The installation path cannot contain Chinese.

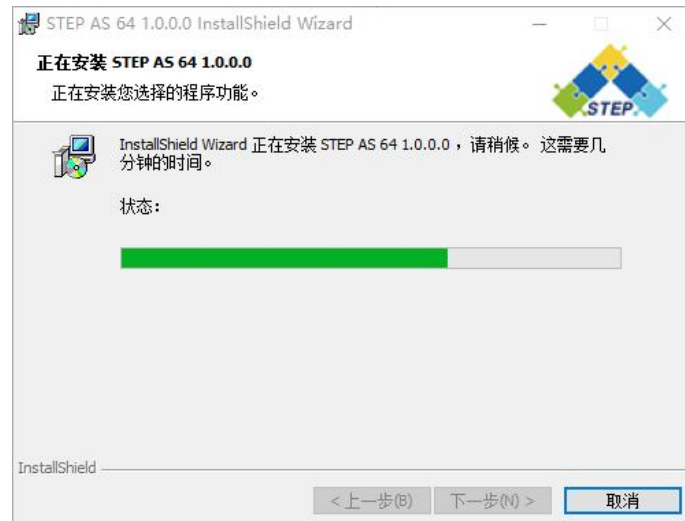
⑤The following screen will be displayed, select [Complete installation], and then click [Next].



⑥The following screen will be displayed, please click the [Install] button to start the installation.



⑦The following screen will be displayed during installation.



⑧After all installations are completed, the following screen will be displayed, please click [Finish].



1.2.4. Uninstall STEP AS

Use the standard Windows system uninstall software method to uninstall STEP AS, the specific steps are as follows:

1) Select Windows system from the start menu ► **control Panel**, click Uninstall a program.

A list of installed programs is displayed.

2) Double-click "STEP AS64 1.0.0.0".

The following screen will be displayed, click [Yes].



3) Click the [Yes] button.

STEP AS is uninstalled.

1.3. STEP AS and hardware connection

The programming device can be connected with the SC series controller through Ethernet (through a hub, switch, etc.), use STEP AS software to write and download user programs, monitor the program and control the SC series controller.

For SC20 series controllers, it can also be connected via serial port or USB interface.

第二章 Quick start

2.1. Start the programming environment

Click the [Start] button and select *STEPAutomationStudio* → *STEP AS V*.*.*.** Or double-click the STEP AS icon on the desktop. STEP AS starts.



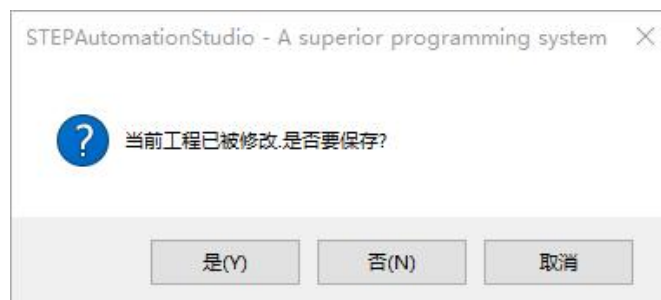
2.2. Exit the programming environment

Before exiting STEP AS, be sure to save the project file that is being edited and needs to be saved.

1. Select File in the menu bar → *quit*.

If it has not been saved, the following screen will be displayed.

- a) To exit without saving, please select [No].
- b) If you need to save, please select [Yes] to save.

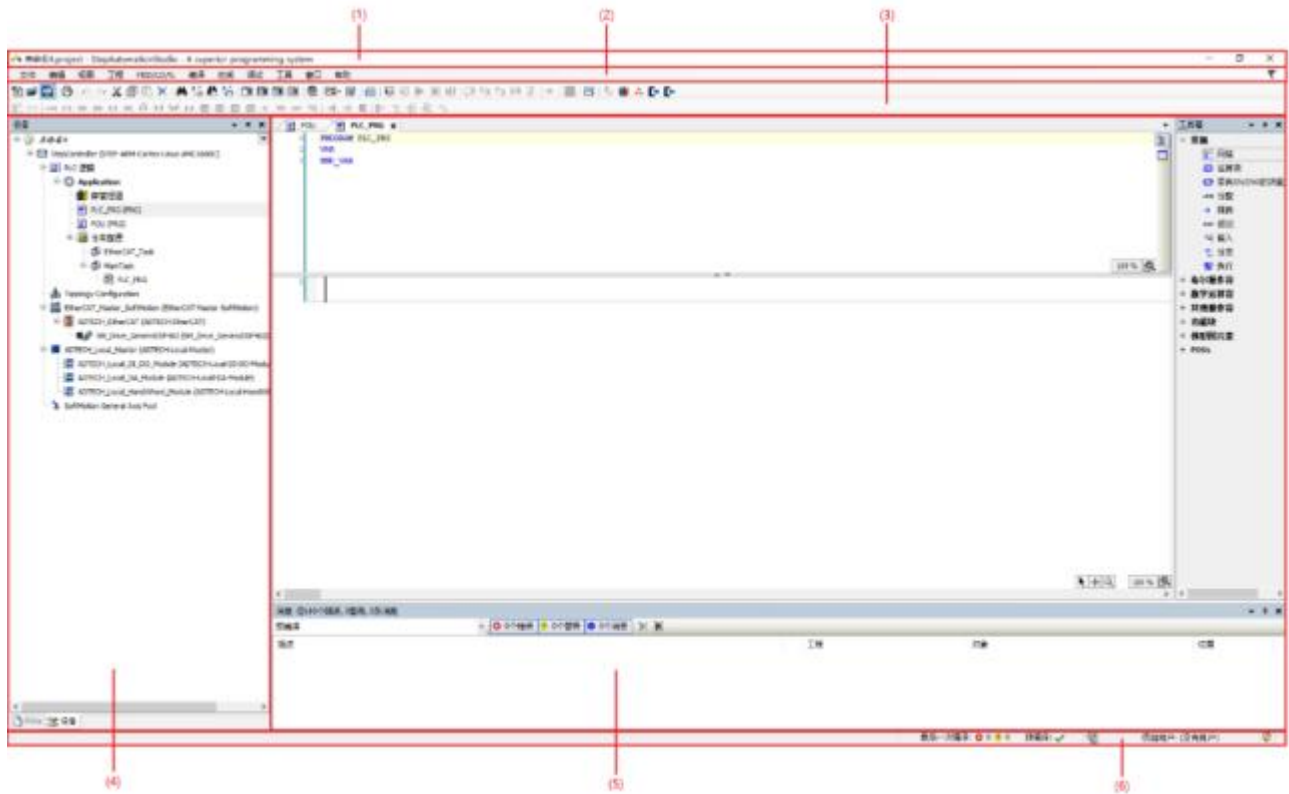


2. Click the [Yes] button, STEP AS will be closed.

①Note: Click directly on the [x] button can also exitSTEP AS.

2.3. name of each part

The names and display contents of each part of STEP AS are shown below.



No.	name	content
(1)	title	Displays the project file name, [Minimize] button, [Maximize] button, and [Close] button.
(2)	Menu Bar	Displays menu commands in a list, sorted by purpose.
(3)	toolbar	Displays instructions as icons.
(4)	navigation bar window	Views such as devices and program POU's added to the project are displayed in a tree structure.
(5)	Main window	Displays the setting screen, information, etc. of programs and functions. You can switch screens using tabs.

(6)	Status field	Displays information such as the status of the compilation, the currently logged-in user, etc.
-----	--------------	--

2.3.1. Menu Bar

The displayed contents of the menu bar are as follows.

文件 编辑 视图 工程 编译 在线 调试 工具 窗口 帮助

- document

project	Features
New Construction	New Construction.
Open project	Open the saved project.
Close the project	Close the currently viewed project.
Save the project	Overwrite and save the currently viewed project.
save project as	Save the currently viewed project with another name.
source code upload	Upload the project source code as a project archive.
Source code download	Download the project source code.
Print	Print the active editor.
Printing preview	Displays a print preview of the active editor.
page settings	Opens the Page Setup dialog for composing the print layout.
List of recent projects	Displays recently used projects.
quit	quitSTEP AS.

- edit

project	Features	
revoke	Undo the last edit.	
recover	Redo the edited content.	
cut	Cut data.	
copy	Copy data.	
paste	Paste the data.	
delete	delete data.	
select all	All Selection.	
find replace	Used to find and replace.	
	project	Features
	find	Opens the Find dialog.
	replace	Opens the Replace dialog.
	Find in project	Opens a dialog for "Find" objects within the entire project.
	In-Project Replacement	Opens the dialog to "replace" objects within the entire project.
	find next	Searches for the next one in the project starting from the selected position of the cursor.
	find next (selected)	Finds the next one within the editor, starting from the cursor's selection.

project	Features	
	find previous	Searches for the previous one in the project starting from the selected position of the cursor.
	find previous (selected)	Finds the previous one within the editor, starting from the cursor's selection.
browse	Used to browse the declaration part of the defined variable and read the usage location.	
	project	Features
	go to definition	The variable and function specified by the cursor can be moved to the position defined in the editor.
	Browse Cross References	The position where the variable at the cursor position is used can be displayed in the "Cross Reference List" view.
	Browse the call tree	The variable specified by the cursor can be called and the source of the call is displayed in the Call Tree view.
bookmark	You can move to a bookmarked location. It is used to browse the declaration part of the defined variable and read the usage location.	
	project	Features
	Toggle bookmarks	Saves the selected location of the active editor as a bookmark.
	next bookmark	Move to the previous bookmark within the active editor.
	previous bookmark	Move to the next bookmark within the active editor.
	clear bookmarks	Delete all bookmarks in the active editor.
input assistant	Variables, function blocks, operators, types, etc. that can be inserted at the cursor position can be selected from the categories and inserted.	
automatic declaration	Opens the auto-declaration dialog for help in declaring variables.	
next message	Select the next message in the message view.	
previous message	Select the previous message in the message view.	
go to source code	Moves to the location of the source code of the object that becomes the selected message in the message view.	
refactor	The position where the changed variable name is used can be displayed and all can be changed.	

- view

project	Features
---------	----------

project	Features	
equipment	Display the device view.	
POUs	showPOUview.	
module	Displays the module view.	
information	Display the message window.	
element attribute	Display element properties.	
toolbox	Show toolbox.	
monitor	The viewing window is displayed.	
	project	Features
	monitor1~4	User-defined variables are displayed at a glance for the purpose of monitoring values.
	Monitor all mandatory	All variables whose values are forced are displayed at a glance.
cross reference list	The cross reference list window is displayed.	
call tree	Displays the call tree window.	
bookmark	The bookmark window is displayed.	
breakpoint	Display the breakpoints window.	
call stack	Display the call stack window.	
start page	Display the start page.	
security fence	The Safety Fence window is displayed.	
Choose a viewing angle	Choose how the windows are laid out.	
	project	Features
	STEP ASview	according toSTEP ASLayout windows in view mode.
	HMIview	according toHMILayout windows in view mode.
full screen	Display in full screen.	
Attributes	Displays the Properties dialog.	

• **project**

project	Features
add object	Append object.
add folder	Append folder.
scan device	Scan for slave devices.
update device	Update selected devices.
edit object	Edit objects.
Edit object using	The configuration page pops up for object editing.
Online configuration mode	Delete the application downloaded in the controller, and change to the state of the connected controller.
Engineering Information	Project creator information and confirmation file information can be set.
Project settings	Project-related settings can be made
Project version information	You can browse and set project version related.
Compare	Compare the currently viewed project with the saved project.
confirm changes	Via the project on the menu bar → Compare Confirm the difference of the objects to be applied.

export	Change the object from the currently viewed project toXMLformat output file.	
import	Import objects into the currently viewed project.	
User Management	The viewing window is displayed.	
	project	Features
	User login	Log in to the currently viewed project.
	User logged out	Exit the currently viewed project.
	Authorize	Authorize the user.

- compile

project	Features
compile	Validate the syntactic structure of the object.
recompile	Perform syntactic structure verification of all objects again.
generate code	Generate application code.
clear	Removes compilation information for the application.
clear all	Same as clear, removes the compilation information of the application.

- online

project	Features	
login to	When logging in, the application generated by code generation will be downloaded to the controller.	
quit	Log out from the currently logged in device.	
download	Download the program while logged in.	
online modification	The application program can be changed without stopping the running controller.	
warm reset	to keep (RETAIN) variable and persistent (PERSISTENT) variables other than variables are initialized.	
cold reset	for continuous (PERSISTENT) variables other than variables are initialized.	
initial reset	Initialize all variables. Remove the active application from the controller.	
simulation	The login operation can be performed without connecting the controller, and the operation can be confirmed by the same operation as the actual login.	
Safety	Set user management, project encryption, etc.	
	project	Features
	Log out the current online user	Log out the user currently logged in to the device.
	Add device user	Add users who can log in to the device.
	Change device user password	Change the password of the user currently logged into the device.
	delete device user	Delete users who can log in to the device.
Operating mode	Restrictions can be made to prevent some debugging operations from being performed.	
	project	Features

	debugging	All debugging operations can be performed.	
	locked	Some operations such as new addition of breakpoints and forcing of variable values cannot be performed.	
	operational	Changes cannot be made except for the writing of variables.	

- **debugging**

project	Features
start up	Start running the application.
stop	Stop running the application.
single cycle	Can make the application execute every time/cycle.
new breakpoint	Create a new breakpoint.
edit breakpoint	Edit breakpoints.
Set or clear breakpoints	Set or delete breakpoints.
Disable breakpoints	Disable active breakpoints.
Enable breakpoints	Activates disabled breakpoints.
jump over	by1The behavioral unit executes the program. If executed at the call location of a block (function, function block), the block is executed and moved to the next line.
jump in	by1The behavioral unit executes the program. If executed at the block (function, function block) call location, it branches to the first line of the called block.
jump out	When executed within the called block, the program is executed until the calling block is returned. When executed outside the called block, the program is executed until it returns to the beginning of the program.
run to cursor	Executes the program before the line specified by the cursor.
set next statement	Makes the line specified by the cursor the next statement to execute, skipping processing until that line is reached.
show current statement	Makes the cursor jump to the next program line to be executed.
write value	Only do/secondary settings. The value can be changed later according to the program.
Mandatory value	Set the value to be changed every cycle and keep that value.
release value	Unenforce the value.
flow control	The executed and unexecuted parts of the program can be divided by color and monitored.
display mode	The display format of the variable value to be displayed can be selected from binary, decimal, and hexadecimal.

- **tool**

project	Features
package manager	Install or uninstall packages.
library	By installing the created library into the library repository, the functions and function blocks in the library can be used.

project	Features
Device repository	You can browse, install, uninstall, and export devices.
Visual type library	You can browse, install, and uninstall the visual type library.
customize	Users can customize the menu bar, toolbar and other content.
Options	can proceedSTEP ASoft of each function setting.
Import and export options	Import and export option setting files.

- window

project	Features
next editor	The next screen is displayed.
previous editor	The previous screen is displayed.
close all editors	Close all screens.
reset window layout	Sets the window's layout position to its initial state.
Create a new horizontal series group	Moves the currently selected screen down.
New vertical sequence group	Moves the currently selected screen to the right.
float	Put the currently selected screen in a floating state.
expand	Puts the currently selected screen in the docking state.
auto hide	Minimize the window.
next child window	can be found in the declarations section (section1pane) and the implementation section (p.2panes).
previous child window	can be found in the declarations section (section1pane) and the implementation section (p.2panes).
window	A list of open screens is displayed.

- help

project	Features
content	Display the help documentation and navigate to the Contents page.
index	Display the help documentation and navigate to the index page.
search	Display the help documentation and navigate to the search page.
about	Display version information.

























2.3.2. toolbar





























The display contents of the toolbar are as follows.













The specific functions are as follows:

name	icon	Features
New Construction		New Construction.
Open project		Open the saved project.
Save the project		Overwrite and save the currently viewed project.
Print		Print the active editor.
revoke		Undo the last edit.
Reply		Redo the edited content.

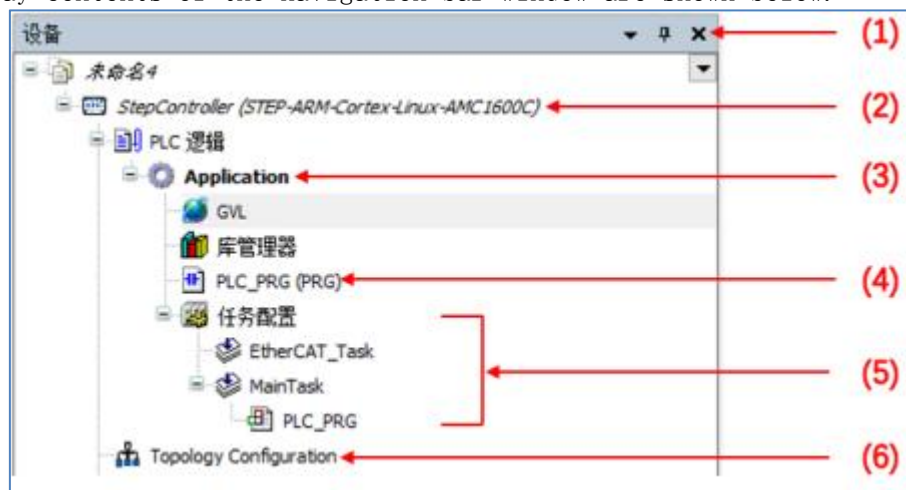
name	icon	Features
cut		Cut data.
copy		Copy data.
paste		Paste the data.
delete		delete data.
Finds a specific string that occurs within the active editor.		Finds a specific string that occurs within the active editor.
Replace a specific string appearing within the active editor with something else.		Replace a specific string appearing within the active editor with something else.
Find the specified string from the current project.		Find the specified string from the current project.
Replace the specified string from the current project.		Replaces the specified string with something else from the current project.
Toggle bookmarks		Saves the selected location of the active editor as a bookmark.
previous bookmark (activity editor)		Move to the previous bookmark within the active editor.
next bookmark (activity editor)		Move to the next bookmark within the active editor.
clear all bookmarks (activity editor)		Delete all bookmarks in the active editor.
Attributes		Display properties.
add object		Append object.
edit object		Open the object.
compile		Performs compilation of in-application objects.
login to		When you log in, the code-generated application will be downloaded to the controller.
quit		Log out from the currently logged in device.
start up		Start running the application.
stop		Stop running the application.
Online configuration mode		The downloaded application program in the controller is deleted and the controller is connected.
jump over		by1The behavioral unit executes the program. If executed at the call location of a block (function, function block), the block is executed and moved to the next line.
jump in		by1The behavioral unit executes the program. If executed at the block (function, function block) call location, branch to the first line of the called block
jump out		When executed within the called block, the program is executed until the calling block is returned. When executed outside the called block, the program is executed until it returns to the





name	icon	Features
		beginning of the program.
run to cursor		Executes the program before the line specified by the cursor.
set next statement		Makes the line specified by the cursor the next statement to execute, skipping processing until that line is reached.
show current statement		Makes the cursor jump to the next program line to be executed.
Toggle targeting		Switch to Works already in the menu bar → Engineering location Set [Switch Location] to a valid language.
confirm changes		Make sure it works in the menu bar → Compare Changes to the application object in .
Compare		Displays the Project Comparison page.
start simulation		Start the simulation.
stop simulation		Stop simulation.
export variable		export variables toExcel.
import variable		importExcelvariable table.
plug into the network		Insert empty net.
Toggle Network Annotation Status		Change the annotation status of the selected network.
Insert output		Inserts a new assignment at the specified location.
Insert the coil		Insert the coil at the specified location.
Insert set coil		Insert the set coil at the specified location.
Insert the reset coil		Insert the reset coil at the specified location.
insert contact		Insert at the specified locationacontact.
insertbcontact		Insert at the specified locationbcontact.
Insert contact (right)		Insert to the right of the specified positionacontact.
Lower parallel insert contacts		Inserted in parallel with the contact at the specified positionacontact.
Insert in parallelbcontact		Inserted in parallel with the contact at the specified positionbcontact.
Insert contacts in parallel		Insert in parallel on the upper side of the contacts at the specified positionsacontact.
insert operation block		Opens the input assistant for inserting a box at the specified location.
Insert empty operation block		Insert an empty box at the specified location.
insert tapeEN/ENOthe operation block		for inserting the attached at the specified positionEN/ENObox to open the Input Assistant.
insert withEN/ENOfunction block		Insert the attached at the specified locationEN/ENObox.
insert jump		Inserts a new jump at the specified position.
insert label		Inserts a label for the currently selected network.


name	icon	Features
insert return		Insert return at the specified position.
insert input		Append input to the specified box.
Negate		Appends "not" to the selected element.
edge detection		Adds edge detection (rising edge detection) to the selected element.
Position/reset		Convert the selected coil to a set or reset coil.
set output connection		Convert the output box to a forward output box.
insert branch		Inserts a branch to the right of the currently selected contact.
insert branch below		Inserts a new branch below the currently selected branch.
insert branch above		Inserts a new branch above the currently selected branch.
set branch start/end point		Sets the currently selected line as the branch start position.

2.3.3. navigation bar window

The display contents of the navigation bar window are shown below.



NO.	name	icon	Features
(1)	Window Position window Location		<ul style="list-style-type: none"> New Horizontal Tab Group Moves the currently selected screen to the right. New Vertical Tab Group Moves the currently selected screen down. Suspended placed in suspension. butt in the docked state. auto hide
			The navigation bar window will minimize and become hidden.
			The navigation bar window will become always on.
	Auto Hide		The navigation bar window will minimize and become hidden.

	hide		
	Close		Close the navigation bar window
(2)	device object		Set the device object.
(3)	application object		Set the application object.
(4)	program(POU) object		Set the program object (POUobject).
(5)	task object		Set the task object.
(6)	Device configuration		Click to enter the device configuration page.

2.4. Engineering operations

2.4.1. Operating procedures

Please refer to the following process when creating a project:

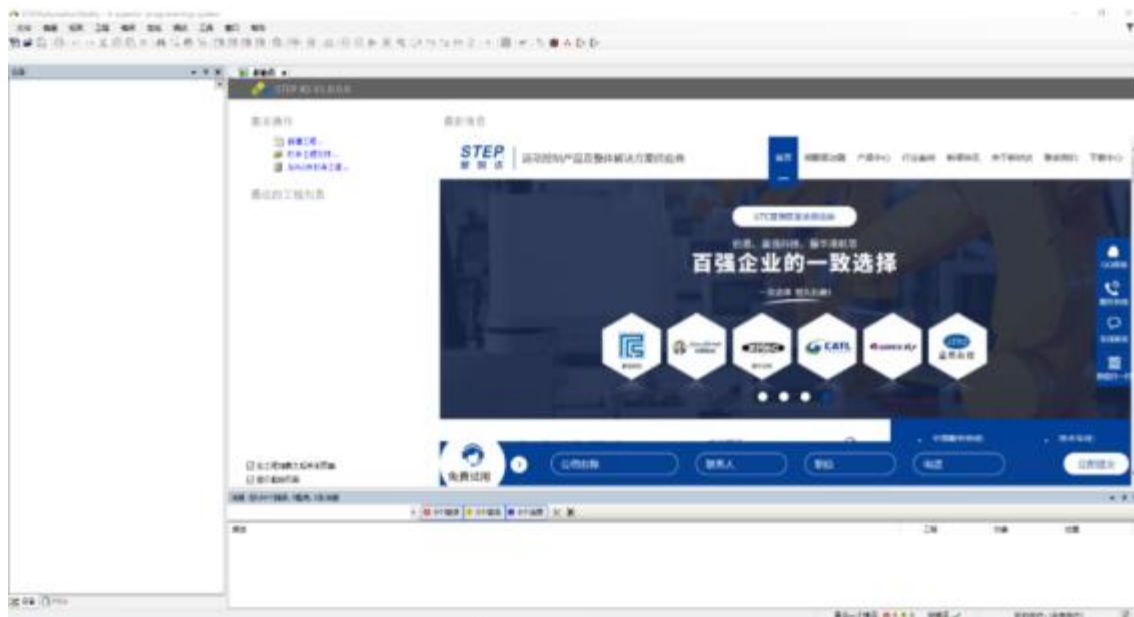
- ① To create a new project, you can create a project through the project wizard or manually (controller selection, new main program POU, etc.).
- ② Write code and configuration, write programs in five languages, and configure tasks, buses, and configuration of interconnected devices.
- ③ Connection, wiring, connect STEP AS PC and controller and controller and external equipment.
- ④ Log in to download the program.
- ⑤ Debug the program, can monitor and track variables in a single step or cycle, and display and calculate the curve.
- ⑥ Auxiliary operations, such as code encryption, user management, startup settings, etc.

2.4.2. Project Wizard

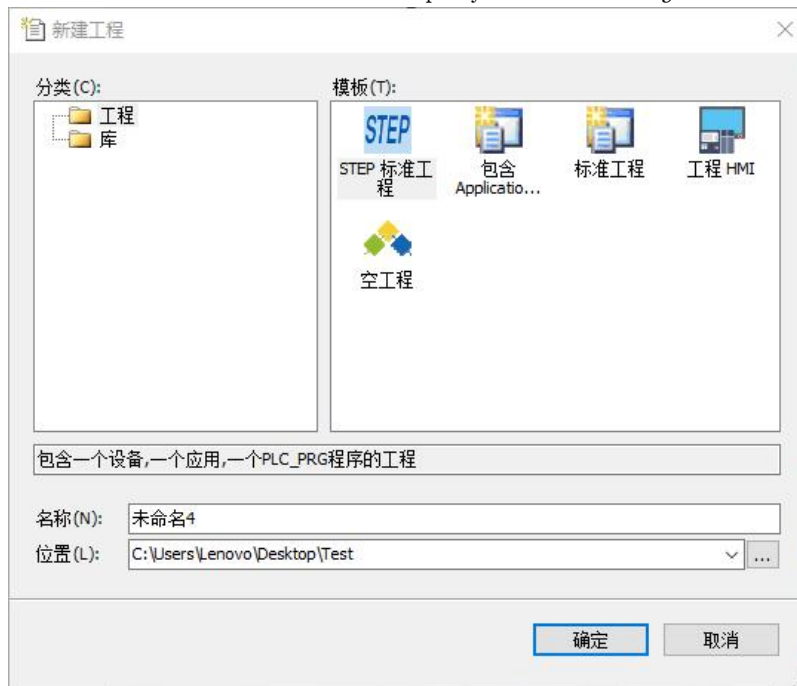
When using STEP AS to create a program for the first time, it is necessary to create a new project and set the equipment and programming language to be used. The steps for creating a new project are described below.

The following is an example of a project created using a ladder diagram (LD) for the SC30-B6H controller.

- (1) start upSTEP AS, the start page will be displayed.



(2) choose "New Construction" to display the New Project dialog box.

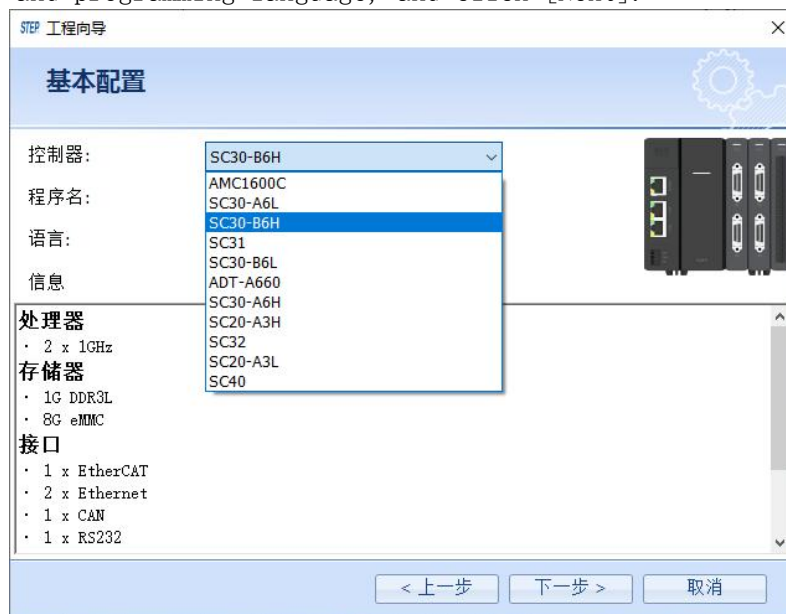


(3) Select Project → STEP Standard Project, specify the file name of the project in the Name column, and specify the save location of the project in the Location column.

(4) Click the [OK] button.
show "STEP Standard Engineering" project wizard, click [Next].



(5) The following basic configuration page appears, set the controller model, program name and programming language, and click [Next].



(6) Enter the following expansion module configuration page according to the selected controller. The expansion module configuration page includes device type selection and pulse selection. The controller does not support Axis Pool or the number of pulses supported by the controller is 0. When the pulse is not available, the number of added pulses can be configured in other cases.

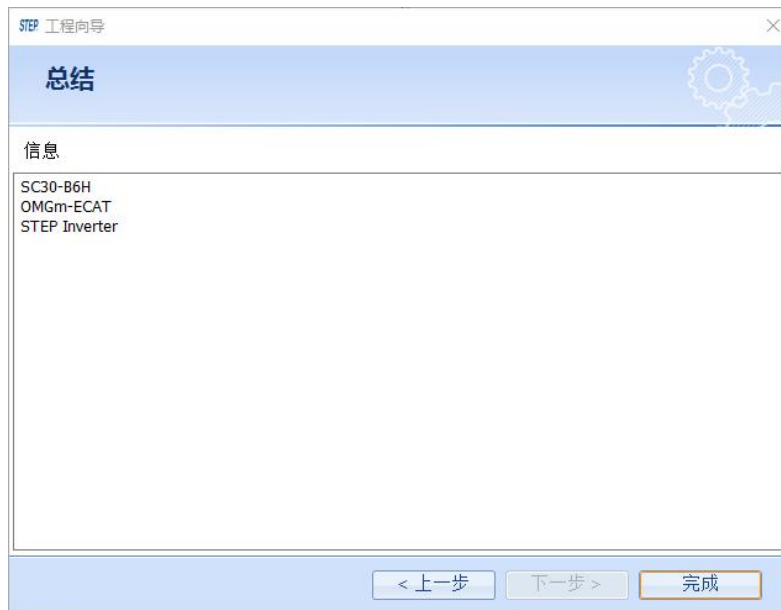


Select a device type and click the Add button next to it to add the corresponding device to the configured device list. Multiple devices can be added in the same way.

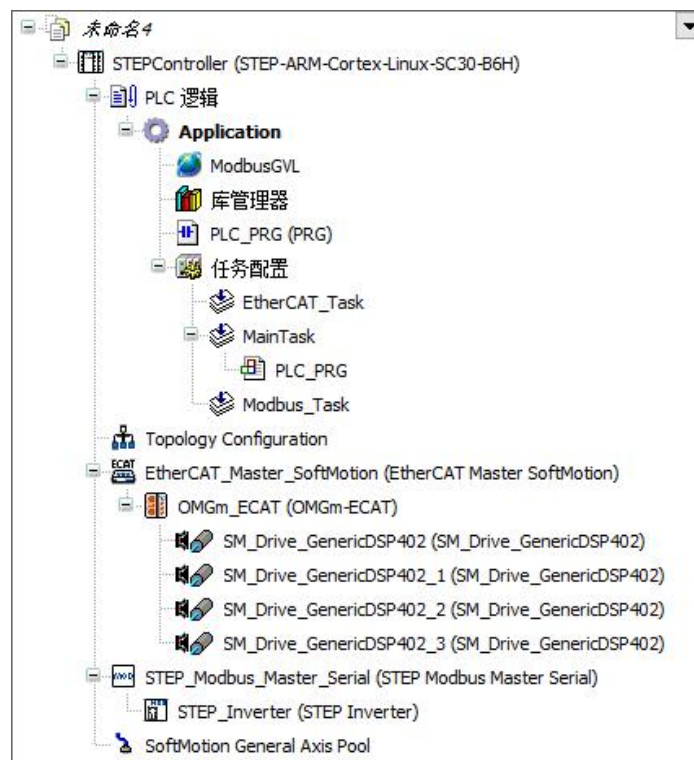


The configured device information can be deleted by pressing the clear button or double-clicking the selected device. The clear button will clear all device configuration information, and double-clicking will delete the selected settings. After the configuration is complete, click [Next].

(7) After the configuration is completed, the following summary page will appear, displaying the project configuration information, click [Finish] to end the new project.



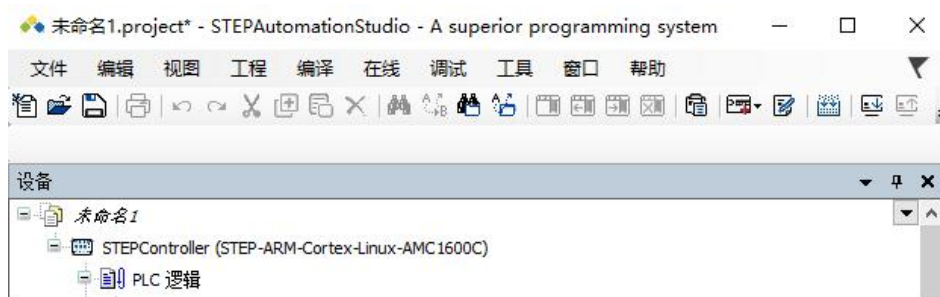
(8) After the new project is completed, the following project tree structure is generated.



① To create a new project, you can select the file in the menu bar → **New Construction** to create.

2.4.3. Save the project

1. Select File in the menu bar → **Save the project**, or press the shortcut key "Ctrl+S".



① Save the created project. The saved project will be saved as ".project" suffixed files.

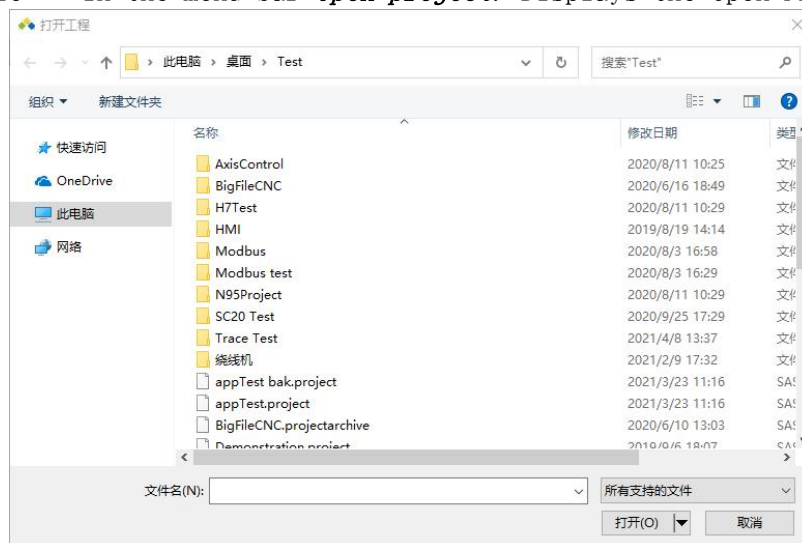
① For unsaved projects, "*" is displayed after the project file name in the title bar*".

① The project being created is saved. "*" disappear.

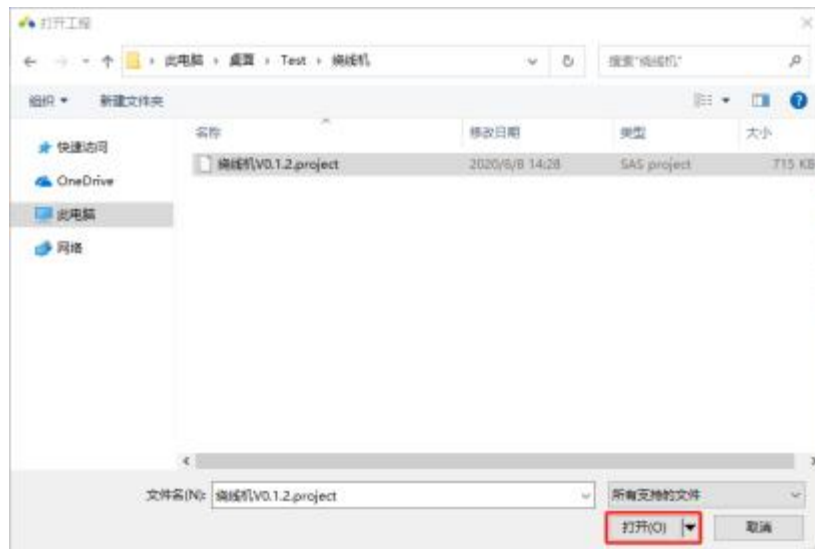
① To change the name of an existing project when saving the project, select File in the menu bar → **save project as**.

2.4.4. Open project

1. Select File → in the menu bar **Open project**. Displays the Open Project dialog box.



2. Select the project file, and then click the [Open] button. The selected project file will be opened.

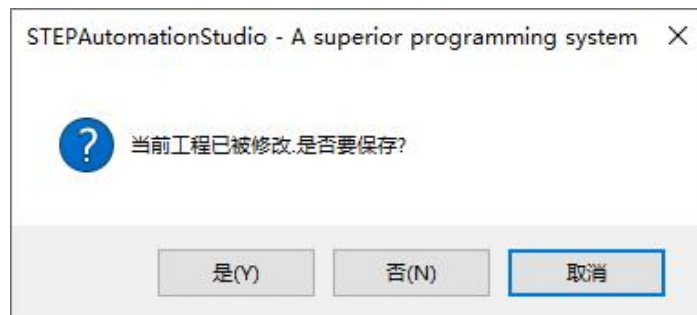


2.4.5. Close the project

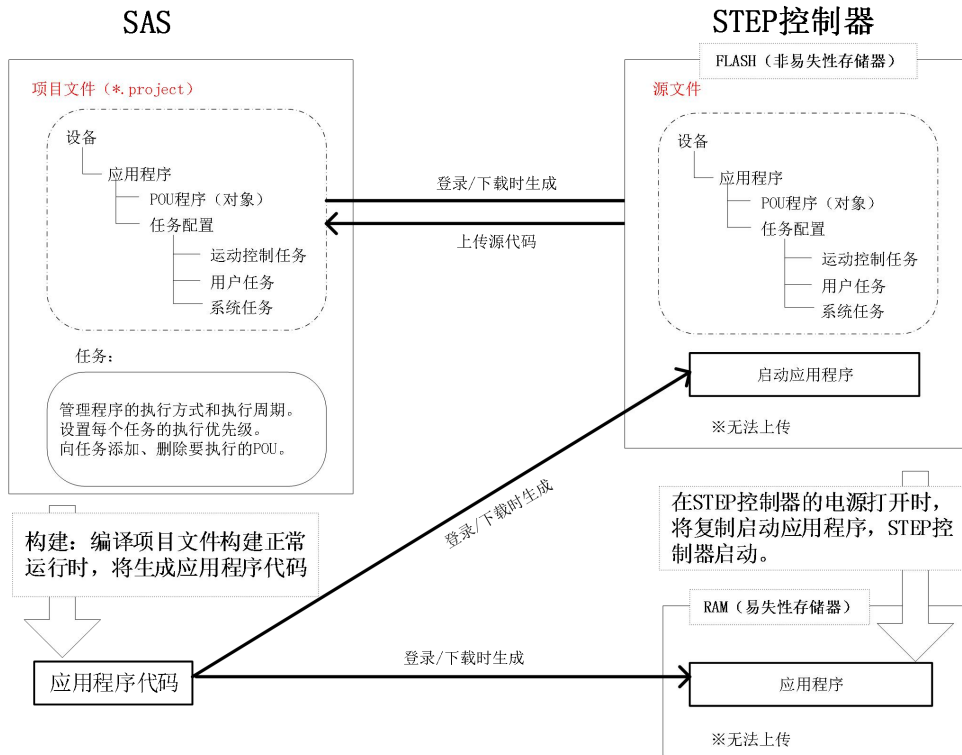
1. Select File in the menu bar → *Close the project*. Close the project being created.

❶ If you select "Close Project" without saving the updated project file, a dialog box to confirm saving the project will be displayed.

Click the [Yes] button to save the project.



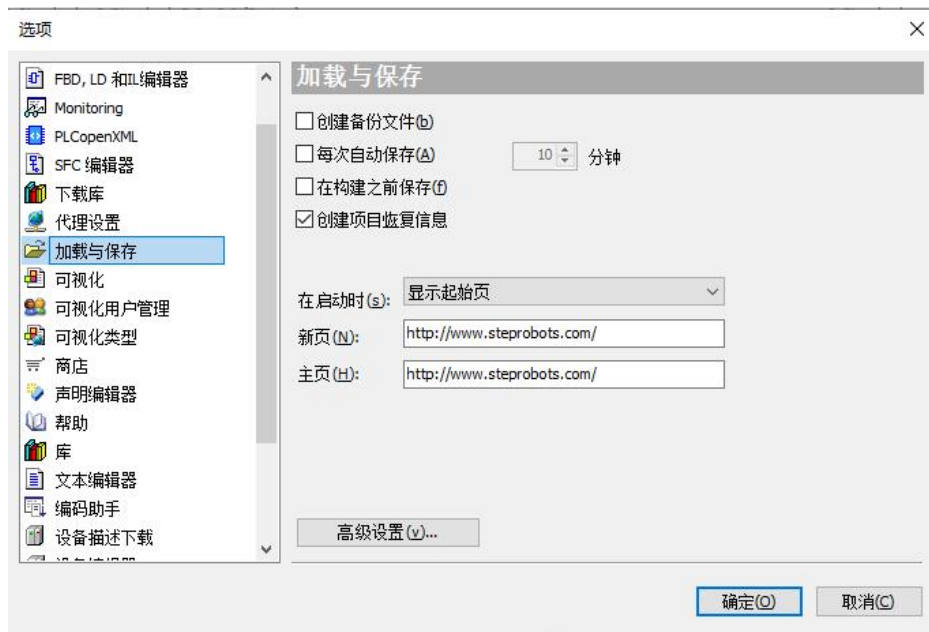
2.4.6. Composition of the project



2.4.7. Create a backup when saving a project

When saving a project, you can keep the project file before updating as a backup file. Backup files have the extension ".backup".

- (1) 1. Select Tools in the menu bar → *Options*, show "Options" dialog.
- (2) 2. Select the Load & Save category to display the Load & Save settings page.



(3) 3. Check "Create backup file", and then click the [OK] button. After that, when saving the project, the project file before the update will be saved as ".backup" document.

①Note: To restore a backed up project file, manually change the file extension from ".backup" Change to and then open the project file in STEP AS.

2.4.8. Automatically save project files

The project file being edited can be automatically saved. Even if STEP AS exits abnormally and data is lost, the file can be restored to the location where it was automatically saved. Backup files have the extension ".autosave".

1. Select Tools in the menu bar → *Options* to display the Options dialog box.
2. Select the "Load & Save" category to display the "Load & Save" setting page. Refer to the previous section for the interface.
3. Check "Auto save every time", change the interval of saving time in minutes (initial value: 10 minutes), and then click the [OK] button.
4. When set to autosave, project files are saved as ".autosave" files at specified intervals when editing a project.

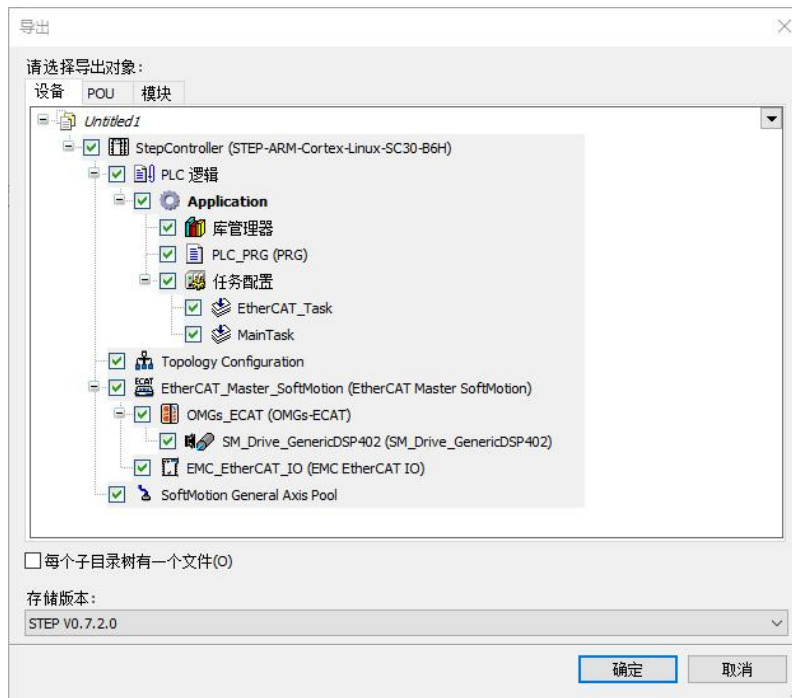
①Note: inWhen STEP AS exits abnormally and opens the project file again after closing the project file, you can select the original project file ".project" , or as an autosaved project file ".autosave" in the selection. To open the auto-saved project file, click the [Open Auto-Save File] button.

2.4.9. Export and import objects

The objects of the project can be exported to the XML format file, and the extension of the export file is ".export". Exported files can also be imported into STEP AS.

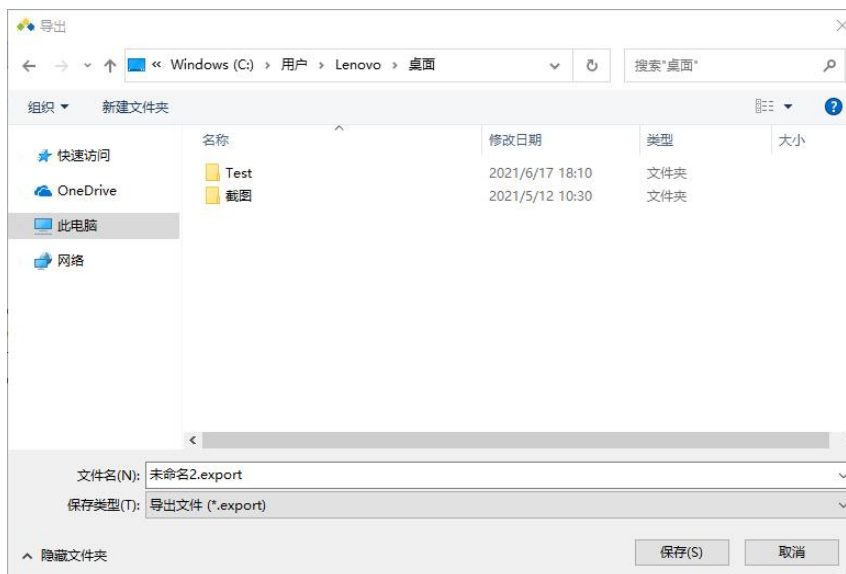
■ export object

1. Select the project in the menu bar → *export*. Displays the Export dialog.

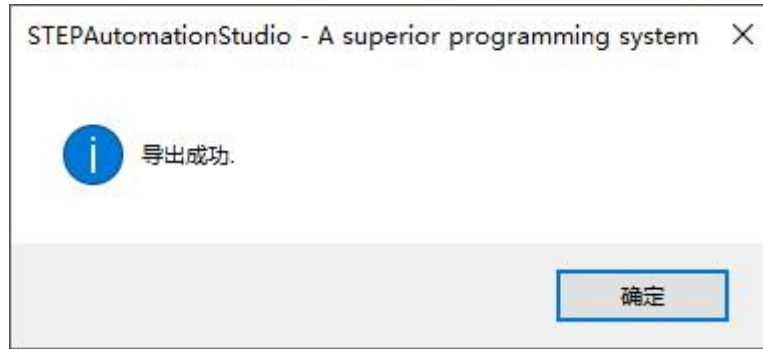


2. Select the objects to export. Usually no changes are required.

3. Click the [OK] button to display the "Export" dialog box. Change the file name and save location as needed.



4. Click the [Save] button. Execute the export, and there will be a prompt for success or not.



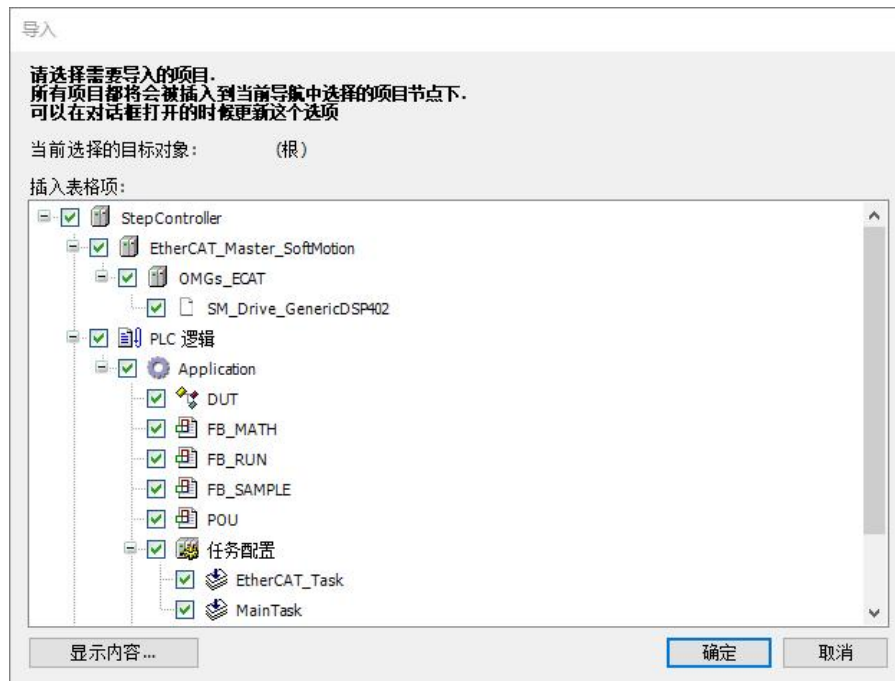
①Note: When importing in STEP AS, please select only one object under [Application] to export.

■import object

This section describes the steps for installing the exported engineering objects into STEP AS.

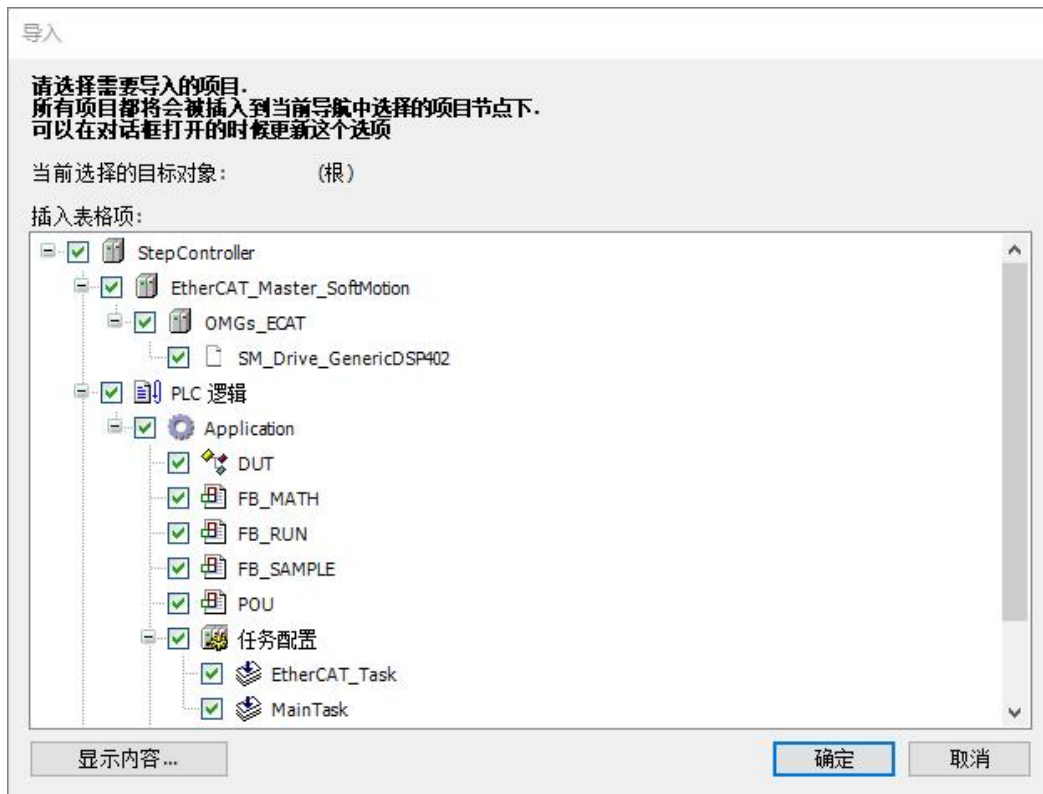
For example, to import objects under the application object, follow these steps.

1. After selecting the project in the navigation bar window, select the project in the menu bar → *import*. Displays the Import dialog.



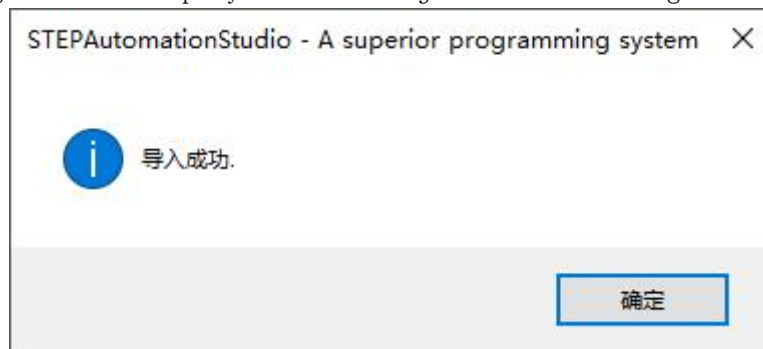
2. Select ".export" file, and then click the [OK] button. Displays the Import dialog.

Importable objects are displayed in the Insert Table Item column.






3. Uncheck the objects that do not need to be imported, and then click the [OK] button. Execute the import.

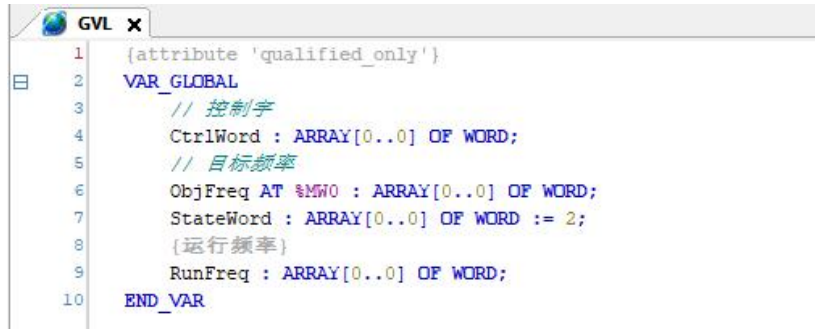
Imported objects are displayed under Projects in the Navigation Bar window.




2.4.10. Variable table export and import

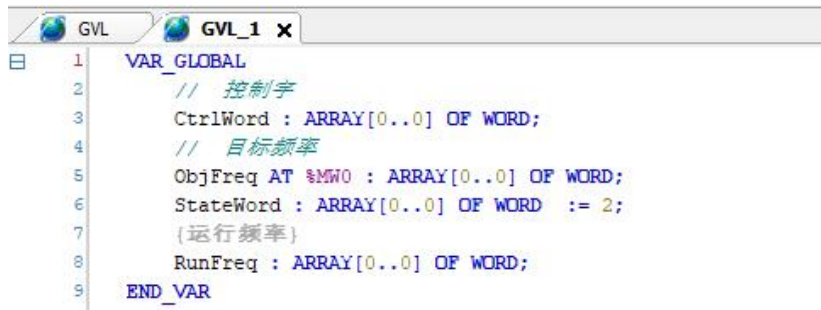
Global variables and program blocks support the export and import of variables. In these two pages, the export and import of the toolbar becomes enabled.  .

1. Click the export icon  The variable list can be exported to an Excel file.



类别	名称	地址	数据类型	初值	注释	特性
VAR_GLOBAL	CtrlWord		ARRAY[0..0] OF WORD		控制字	
VAR_GLOBAL	ObjFreq	%MW0	ARRAY[0..0] OF WORD		目标频率	
VAR_GLOBAL	StateWord		ARRAY[0..0] OF WORD	2		
VAR_GLOBAL	RunFreq		ARRAY[0..0] OF WORD			运行频率

2. Click the import icon  Excel variable tables can be imported into the program.



2.4.11. Export Import Project Archive

1. In addition to the pure interface library, ensure that the fixed version of the library is integrated in the project. To do this, open the “[library management](#)”, and check that a fixed version specification contains “*” of all items.

2. Make sure a fixed compiler version is set in the project settings. to check, choose [project](#) → [Project settings](#) → [compile options](#) category.

3. Make sure a fixed view profile is defined in the project settings. to check, choose [project](#) → [Project settings](#) → [View Profile](#) category.

4. Please make sure that the currently opened application is the same as the application currently on the PLC. This means that the “startup application” must be the same as the project in the programming system. Go to check and see the project name in the title bar of the “Programming System” window: if “*” appears after the name, which means that the project has been modified but not saved. Application and startup application may not correspond!

① Note: In this case, first create a (new) startup application. The automatic occurrence of the

download process of the application depends on PLC and application properties. For explicit creation, select command *online* → *Create a startup application*. Then execute the about command in the help *online* → *login and online* → *load* download.

5. Launch on the controller about the command *debugging* → *start ups* application.

6. Generate a project file: select *document* → *Engineering Archive* → *save/send archive*. exist "Engineering Archive" dialog box, also select the following information:

- Download info file
- library configuration file
- Referenced device
- referenced library
- View Profile

Save the project file in a place that can be accessed by PC2.



7. Log out of the controller: To do this, select *online* → *log out*. before you reconnect PC2 Before, you could stop and restart without reservation PLC.

8. Extract project files to PC2: choose *document* → *Engineering Archive* → *unzip archive*. And open the above saved document. In the Extract Archive dialog, activate the same information as described above when generating the documentation.

9. Open the project and log into PLC "xy" again.

2.4.12. add object

An application that can add objects for creating programs (POU objects) and objects with

various functions to the project.

For example, if you want to add a POU object for ST program, follow the steps below.

1. Right-click to select [Application] in the navigation bar, and select Add Object → POU from the displayed menu.



The "Add POU" dialog box is displayed, first select the type, the default is "Program", and other functions include function blocks and functions; then select the implementation language.



2. Enter the program name in the name field, select the program type in the description language field, and then click the [Open] button.



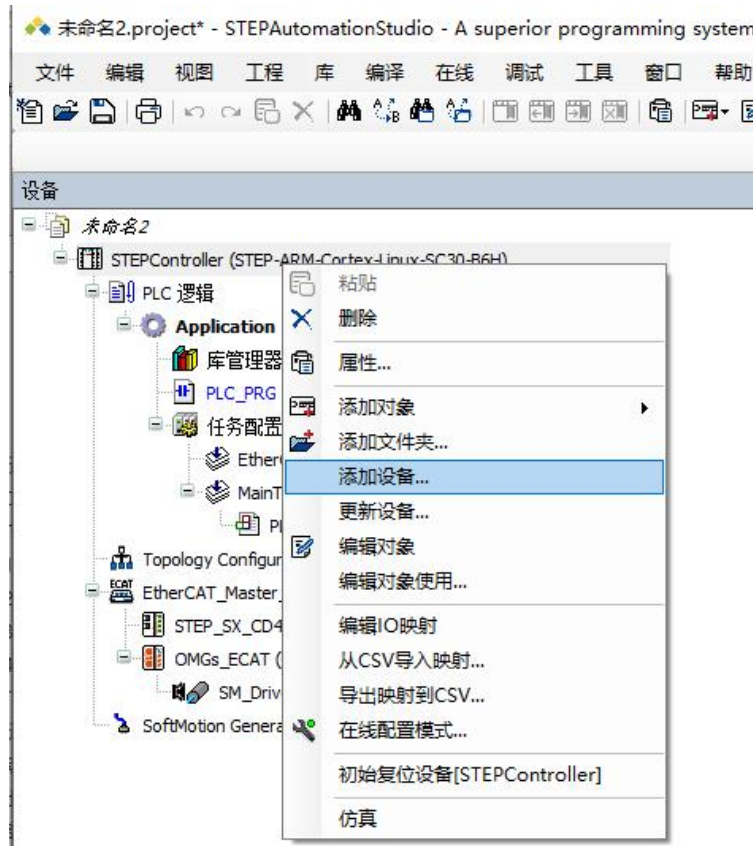
The [POU] object describing the program selected in the language bar is added to the navigation bar window.

2.4.13. Add device

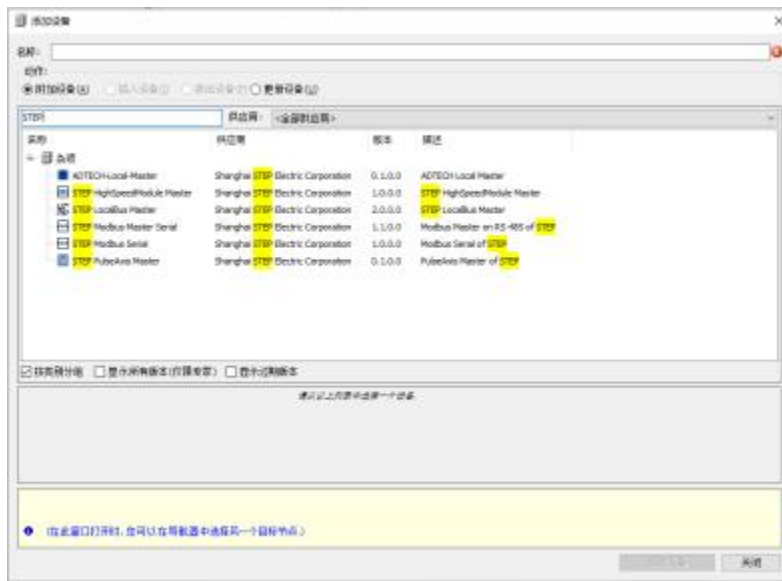
You can add devices to the devices in the project.

For example, to add a device under SC30-B6H, please follow the steps below.

1. Right-click the "SC30-B6H" device in the navigation bar window and select "Add Device" from the displayed menu.



The Add Device dialog box is displayed, and the string to filter can be typed in the text box.



2. Select the device to be added, and click the [Add Device] button to complete the addition.

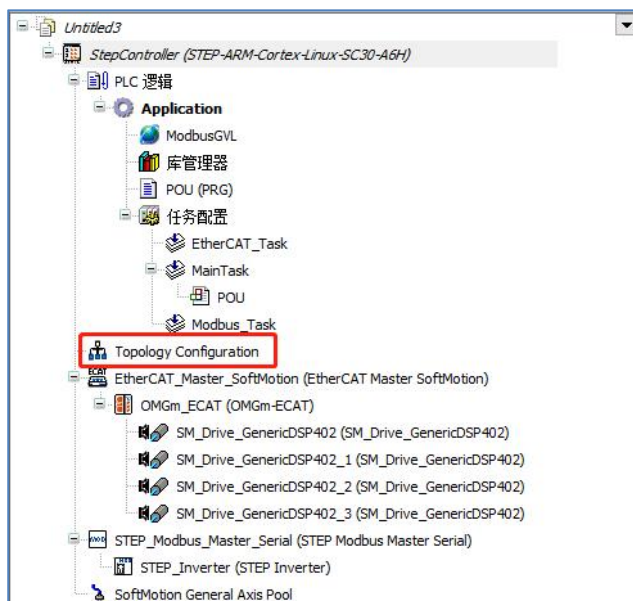
① Note: The addition of devices can also be done through graphical configuration (refer to the chapter on device configuration below).

2.5. Device configuration

STEP AS supports the visual configuration of projects.

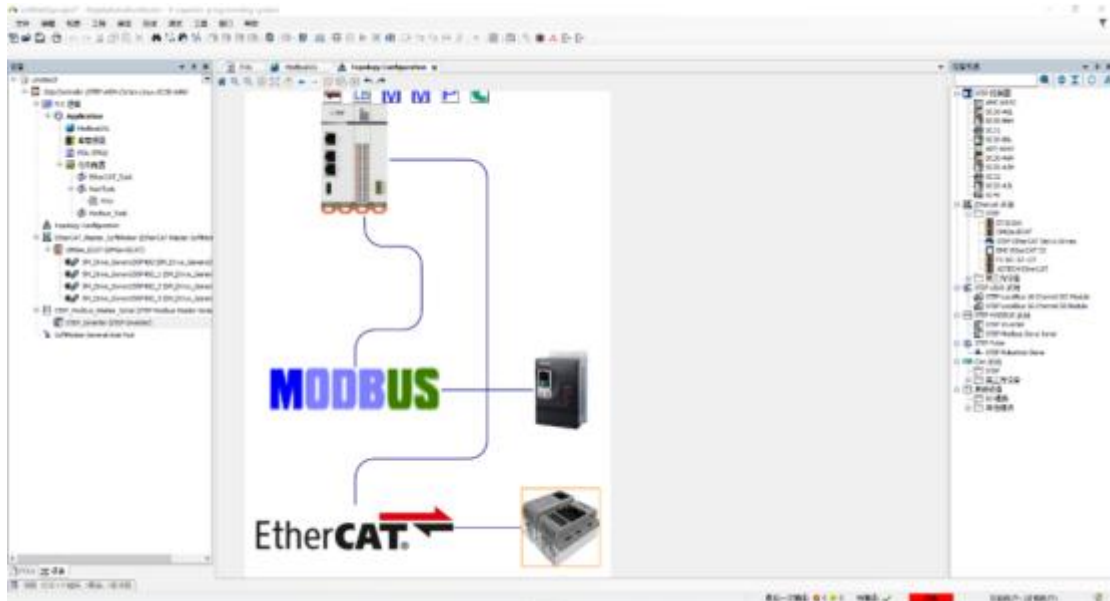
2.5.1. Overview

1. Double click on the navigation bar  icon to enter the device configuration.

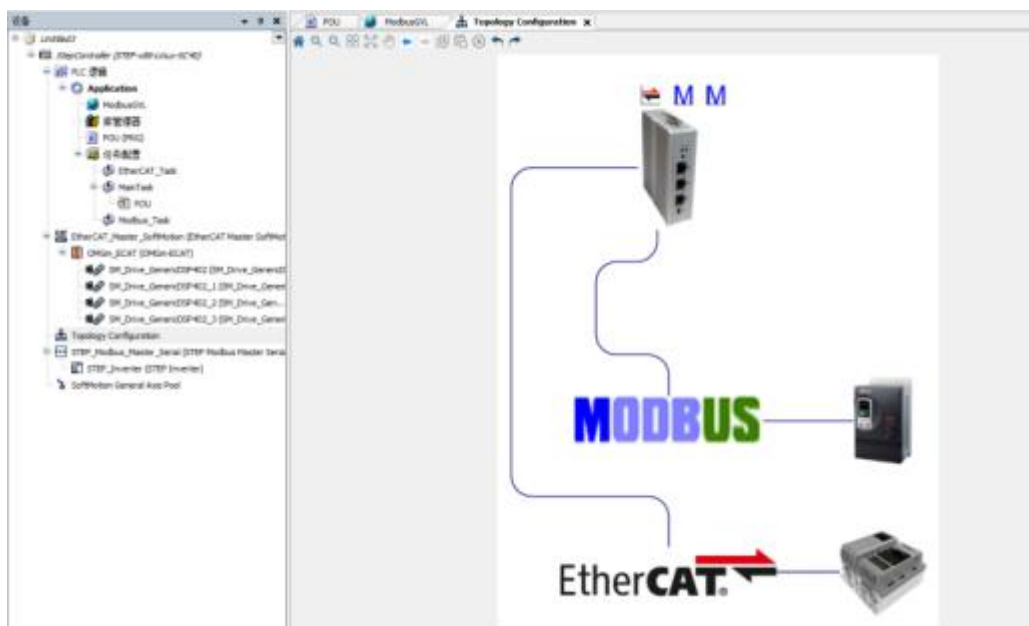


①Note: When there are multiple STEP controllers in the project, only the first added STEP controller has the visual configuration function.

2. The device configuration includes the page drawing area on the left and the device tree on the right. The drawing area of the page draws the devices that have been added in the current project, the device tree displays the slaves and IO devices in the current device library, and the device tree can be used to add devices to the project.



3. Different controllers customize the display hardware diagram.

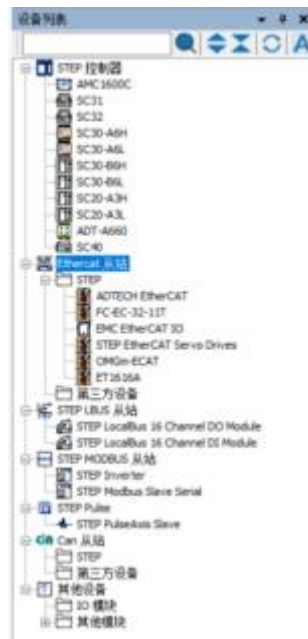


①Note: The selected device border is red, the device border the mouse is over is Orange.



4. Device tree

The device tree, on the one hand, shows all addable types of slaves supported by the current controller (Modbus/CANopen/EtherCAT/StepLbus/Local/PAll or part of the types in ulse) and integrated IO devices; on the other hand, slave devices can be added to the project.

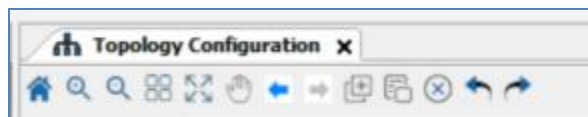


①Note: The top of the device tree is the STEP controller device in the current device library. Double-click to update the current controller.

2.5.2. toolbar











The tools of device configuration are divided into visual page tool bar and device page tool bar.

1. Visual page toolbar.



The specific functions are as follows:





name	icon	Features
global display		The drawing area fits the entire page, restoring the zoomed drawing area to tile the entire page.
enlarge		With the mouse as the center, zoom in on the drawing area.
zoom out		With the mouse as the center, zoom out the


name	icon	Features
		drawing area.
arrange again		Updates all devices under the current project and restores the default layout.
Zoom in		Press and drag the left mouse button to zoom in on the frame selection area.
drag		Click to enter the drag mode, drag the mouse to move the entire drawing area, and then click the button to exit the drag mode before other operations can be performed.
back		Used in combination with drag to return to the previous position in the drawing area.
go ahead		Used in combination with drag to return to the next position in the drawing area.
copy		To copy the selected device, only the slave station and IO can be copied. The controller and the master station cannot be copied. Only a single copy can be selected. If multiple copies are selected, only the last selected device can be copied.
paste		Paste the copied device. If you do not paste and operate other devices after copying, it may cause the paste to fail, and you need to copy it again.
delete		Delete the selected device, multiple devices can be selected at the same time. Controllers and StepLbus devices cannot be deleted (StepLbus devices can be deleted using the delete function of the right-click menu or the navigation bar).
revoke		Undo added or deleted devices.
recover		Used in combination with Undo, undoes an undone action.

2. Device page toolbar.



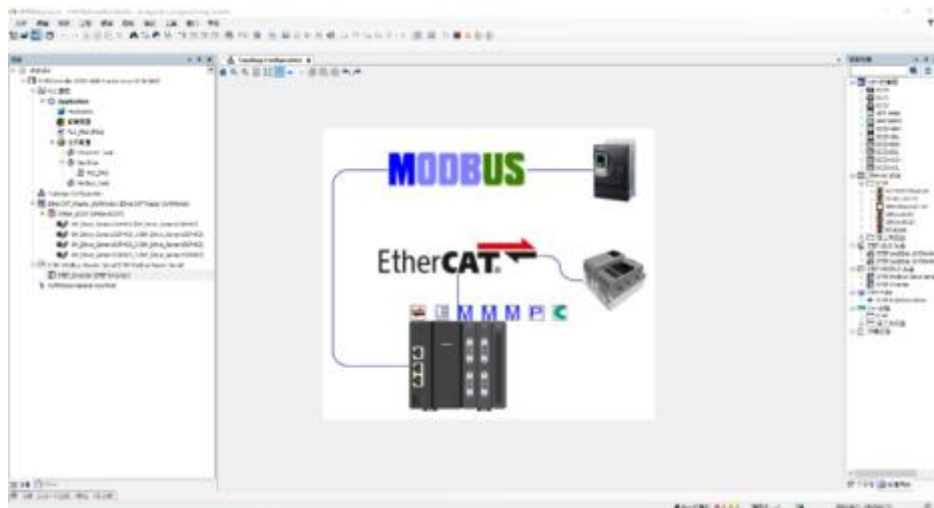
The specific functions are as follows:

name	icon	Features
search		Enter a keyword in the search box, and click the search icon to highlight the searched content in the device tree. Changing the keyword will automatically update the searched content.
expand		Click the expand icon to expand all nodes in the list.
fold		Clicking the collapse icon will collapse all nodes to the root node.
refresh		Clicking the refresh icon will refresh the

name	icon	Features
		available devices in the current development environment. It is mainly used to update the status of the device in the device tree after the device is installed/uninstalled.
Version management		Check the version management icon to display all versions of the device, and leave the version management icon unchecked to display only the latest version of the device.

2.5.3. Add master

There are multiple icon buttons on the top of the controller, which correspond to the master station types that can be configured by the current controller. Unsupported master station types cannot be added through the topology. Master station types include ethercat, canopen, modbus, StepLbus, STEPModbus, STEPHighSpeedBus, Local, Pulse. Double-click to add the corresponding master device. The device version is the same as that of the controller. If the current device does not have a consistent version, select the lower version. If there is a master station and then double-click the button, a pop-up box will prompt whether to delete the current master station.



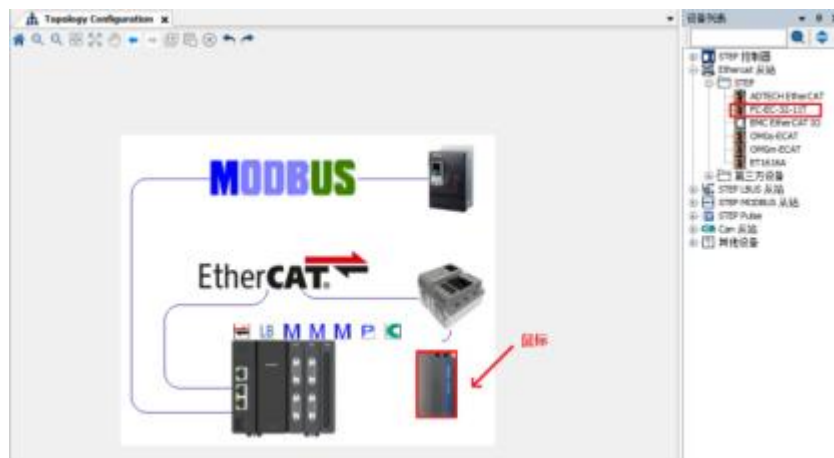


2.5.4. Add device

1. Add a slave

When adding a slave, you can select the master node first, and the device tree will automatically expand the corresponding slave list and collapse other device lists.

There are two ways to add using the device tree on the right, double-click the device or drag the device to the drawing area. When dragging to add a device, press the left mouse button after selecting the device in the device list, and drag it to the position where the device is to be placed in the drawing area.

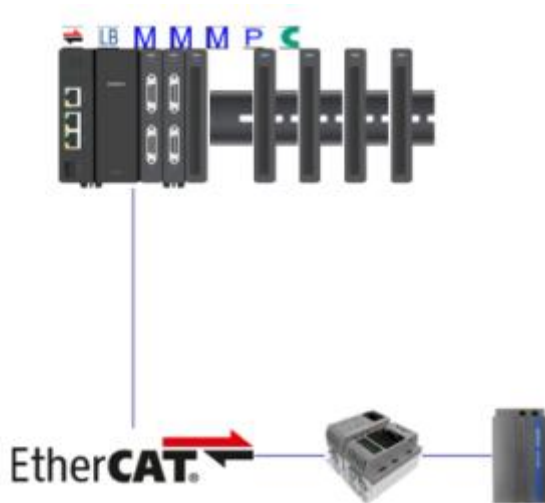


Release the left mouse button, the device will be added to the project at the appropriate position, and the device will be drawn at the mouse release position.

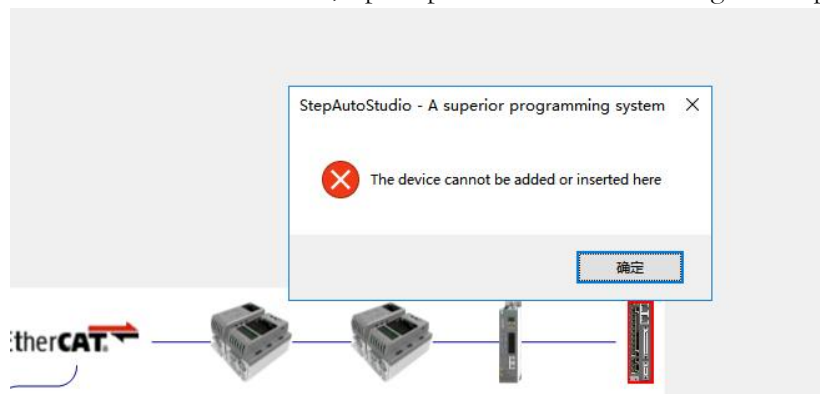
①Note: Double-click the selected device directly in the device list, and the device will be added to the right position of the last device of the same type.

2. Add LlocalBus

StepLbus is an extended IO. When adding a StepLbus slave, the device diagram will be displayed on the right side of the controller in the form of rails. The first rail diagram is the StepLbus master, and the others are the StepLbus slaves. When adding controller integrated IO, since the IO is integrated inside the controller, no new device graph will be displayed on the page.



①Note: If the device fails to be added, a prompt box as shown in the figure will pop up:

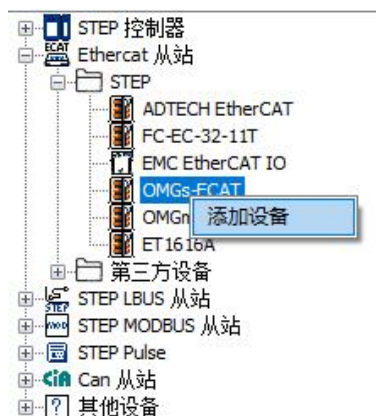


The reason for the pop-up box is that the device does not match the parent device, maybe the device XML file is incorrect and cannot be added to the current master.

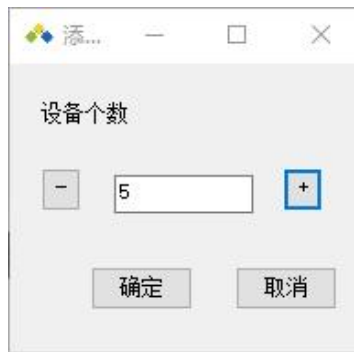
When adding other device types, including IO and other non-classified devices, identify whether there is a matching parent device node in the current project, add if there is, and prompt if not.

3. Add multiple devices

You can right-click the device to be added to the project in the device list to add multiple devices.



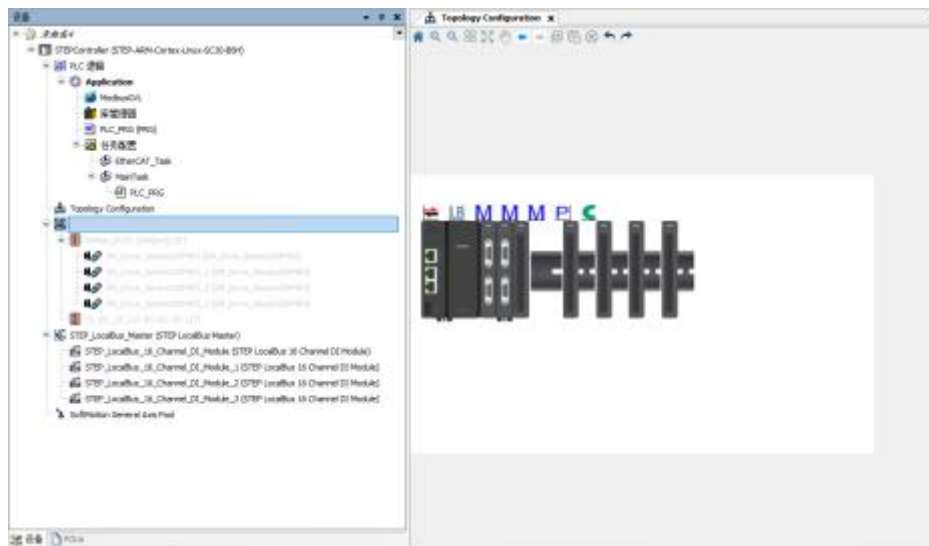
The dialog box shown in the following figure pops up, enter the number of devices to be added and click OK to add the corresponding number of devices in the topology diagram.



2.5.5. Synchronization disable/enable master

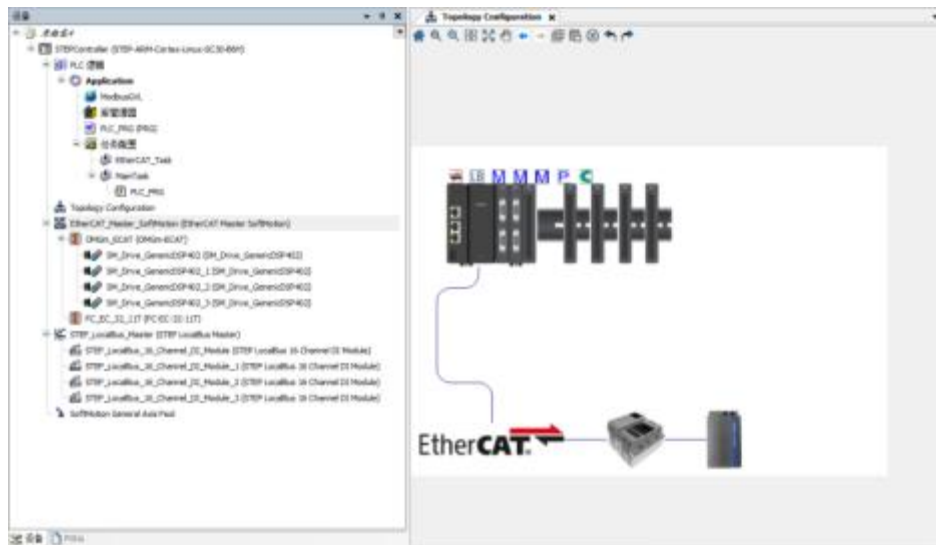
1. Disabled master station

Right-click the master station in the navigation bar and select "Disabled Device", the corresponding master station and slave station will be grayed out, and the disabled device will be hidden on the visual configuration page.



2. Enable the master station

Right-click the master station in the navigation bar and select "Enable Device", the corresponding master station and slave station are restored, and the corresponding device will be displayed on the visual configuration page.



2.5.6. Right-click function

The right-click functions of the visual configuration page include: delete device, rearrange, scan device, update device, collapse device, and uncollapse.

1. Remove the device

Click Delete Device to delete the device where the mouse is located.



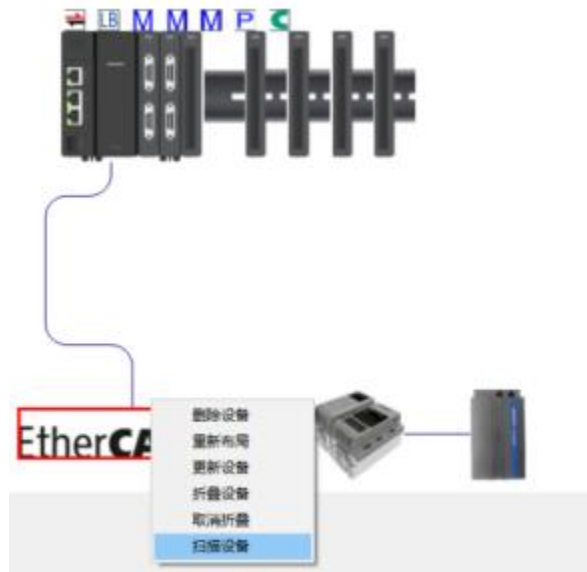
①Note: Unlike Toolbar Delete, Toolbar deletes the currently selected device.

2. Re-layout

The rearrangement function of the right-click menu is the same as that of the toolbar, synchronizing the engineering device and restoring the default layout.

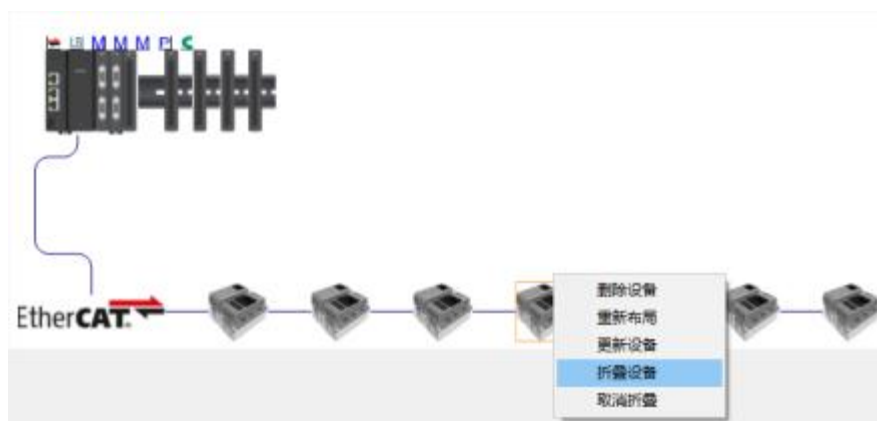
3. Scan the device

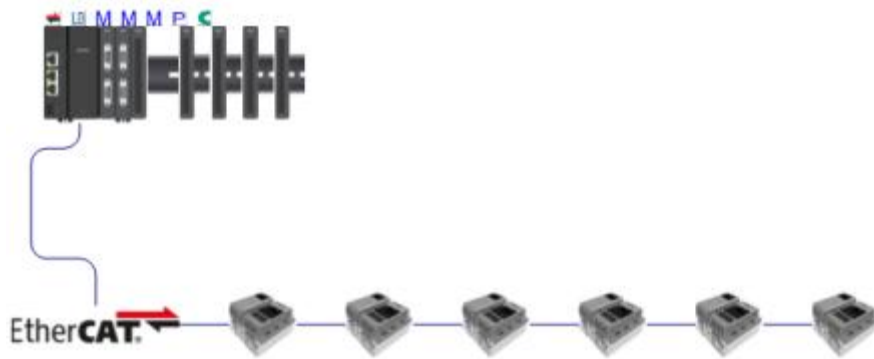
On the visual configuration page, the EtherCAT master can also scan the device. Right-click to select the scanning device to realize the same device scanning function as the navigation bar.



4. Folding the device and unfolding

When there are too many devices configured, sometimes it is not necessary to display all the devices. You only need to browse the general situation of the devices. You can right-click to select the folded device to hide some devices, and only display some devices at the head and tail. When continuing to add devices, the newly added device will be displayed behind the last device to indicate that the device is added successfully. If multiple devices are added, you can still right-click to select the folded device, hide the device, and display some devices at the head and tail after the addition.



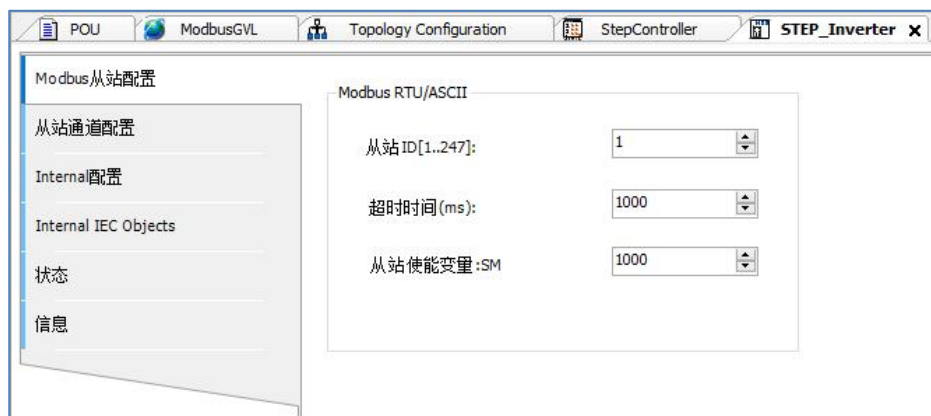


2.5.7. custom layout

Select a device for a custom layout. To select multiple devices, you can use the mouse to make a box selection, or hold down ctrl and click the devices to be selected respectively. It should be noted that the StepLbus device and the controller move synchronously, it cannot move with other devices.

2.5.8. Open configuration table

Double-click the device icon to open the device detailed configuration page shown in the figure below, and you can configure some parameters of the device.



2.6. Create a program

2.6.1. Process of creating a program

① Create a POU object

Create objects for use in programs (POU objects).

② Program input

Open the POU object, perform operations such as inputting programs, declaring variables,

etc.

③ Compile

Compile and check the program.

If there is an error, return to 2 and modify the program.

④ Register to the task

Register the POU object to be executed to the task in the controller.

① Equipped with support functions for efficient program creation.

① Programs that can create functions and function blocks.

2.6.2. program creation interface

This section describes the screens for creating programs using STEP AS.

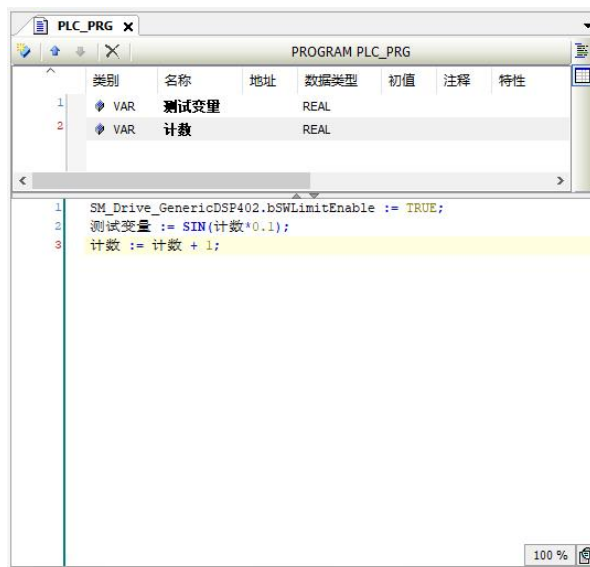
1. Main window

The upper part of the main window is the declaration section for declaring variables.

The lower part of the main window is called the implementation part and is used to describe the processing of the program. The declaration part is also sometimes referred to as pane 1, and the implementation part is referred to as pane 2.

The method of editing the implementation section varies from program to program.

Example: Main window of ST program



If it is graphical programming, there will be a corresponding programming icon on the right.



- ① You can toggle the selection state between the declaration part (pane 1) and the implementation part (pane 2). Window via the menu bar → *next child window* or "Previous Child Window" to switch.

- ① The declaration part (pane 1) or the implementation part (pane 2) can be hidden.

Select Window in the menu bar → *Pane 1* Toggle, will hide the declarations section.

Select Window in the menu bar → *2nd pane* Toggle, will hide the implementation part.

- ① With the cursor at the variable position in the implementation section, select Edit → *Browse* → *go to statement*, you can move the cursor to the declaration position of the variable.

- ① You can also declare variables of user-defined types such as structures. User-defined types must be pre-defined in the DUT object.

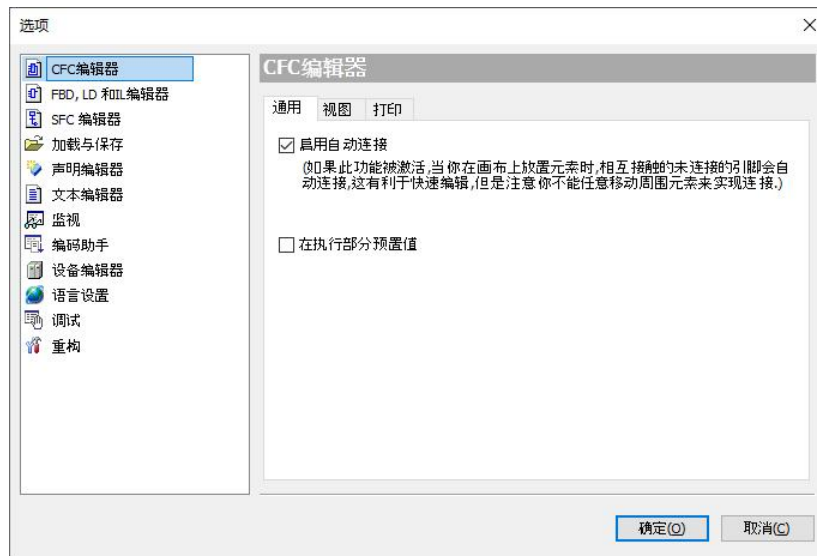
2. Declaration Editor

Declare variables in the declaration editor.

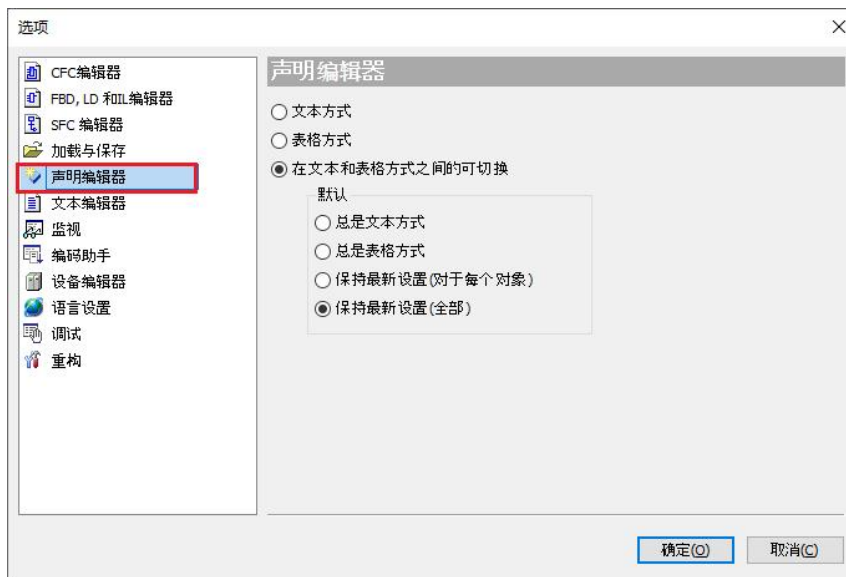
There are two display formats for declarations: form format and text format. The form format and text format can be toggled via the toggle button to the right of the declaration editor.

The display format used can be set.

Select Tools from the menu → *Options*. Displays the Options dialog box.




Select the "Declaration Editor" category in the options.






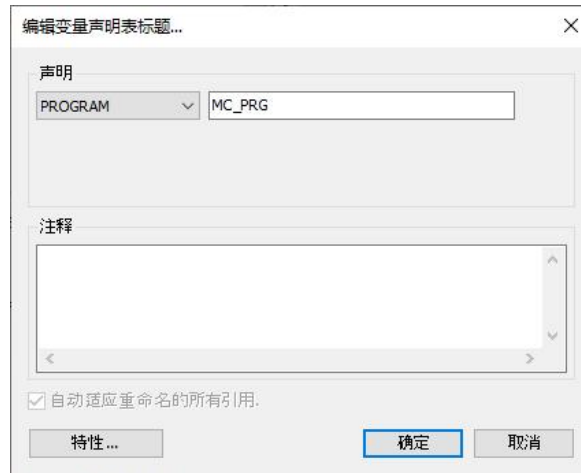
Select the format to use.

☒ form format



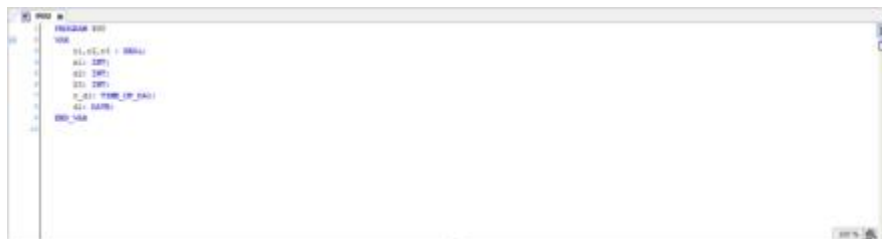
a) To add a new statement, click here  icon to add a new row. Enter the variable name in the "Name" field, double-click the other item to make the cell in the input state, and enter as needed.

- b) use   The icons (move up, move down) sort the variables.
- c) use  icon to delete a variable.
- d) To add a program name or a comment to the program name, click the Declaration Header section.



Displays the Edit Variable Declaration Table Title dialog box.

☒text format



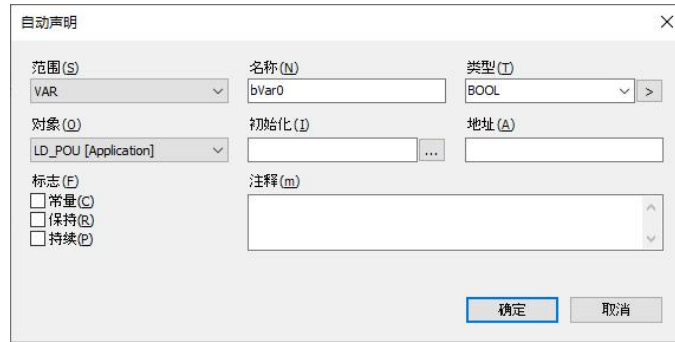
- a) Enter the variable you want to declare like a text editor.
- b) Annotations can be used with a row object (//) and with a multi-behavior object (*,*).
- c) according to "F2" key, will start the input assistant, you can select the type of variable, etc. and enter.

3. Automatic declaration

If you enter a variable in the implementation section that was not declared in the declaration section, the Auto-Declaration dialog box will appear.

Change the necessary items and click the [OK] button, the variable has been declared in the declaration section.

Example: When the variable name of the contact is entered as bVar0 in the LD program



- **Address (A)**

The address of input data and output data of the controller or expansion unit can be specified in the address bar. In this case, declare it as a variable assigned to the input data or output data corresponding to the address of the input.

- **Logo (F)**

The properties of variables can be set by checking Constant, Hold, and Persistence of the flag.

- **constant (C)**

declared as constant. Please enter an initial value.

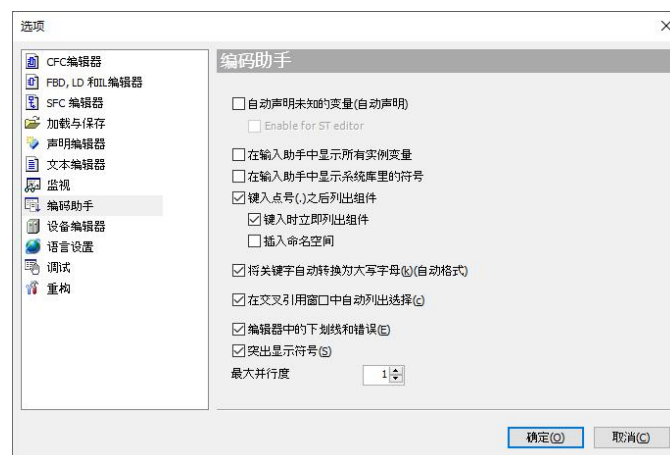
- **Hold (R)**

declared as a holding variable. Holding variables are not initialized on warm reset.

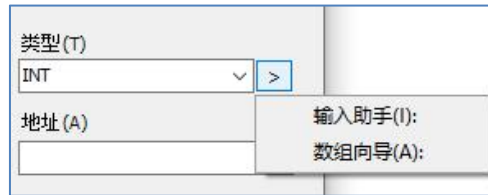
- **Continuous (P)**

declared as a persistent variable. To declare a persistent variable, keep the variable checked. Persistent variables do not initialize their value on either a cold reset or a warm reset.

① When entering an undeclared variable, it is also possible not to display the Auto-declare dialog. Select Tools from the menu bar → *Options* → *coding assistant* Category, uncheck "Auto-declare unknown variables (auto-declare)".



① Using the Array Wizard, you can declare an array by simply entering the index and primitive type. Click next to the Type column and select Array Wizard.

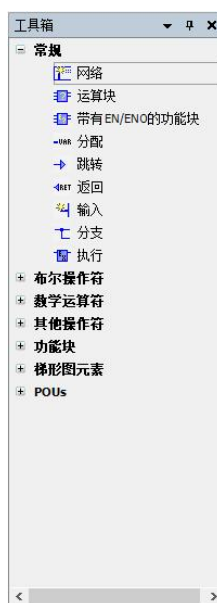


4. toolbox

Programs can be created by dragging the programming elements displayed in the toolbox.

Programming elements for programs other than ST programs are displayed in the Toolbox.

Example: Toolbox for LD programs



5. Program input screen settings

You can change the relevant settings of the text editor.

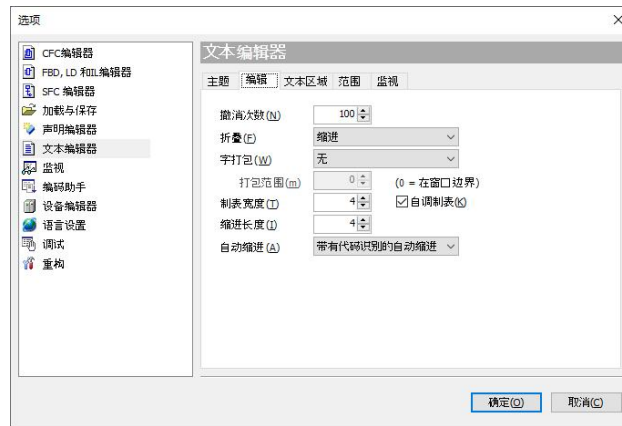
Select Tools in the menu bar → **Options** to open the Options dialog box. Select the Text Editor category of the Options dialog box and change the settings.

- theme



project name	default setting	set content
theme	Default	Sets the color scheme theme of the text editor. Default/Dark

- edit

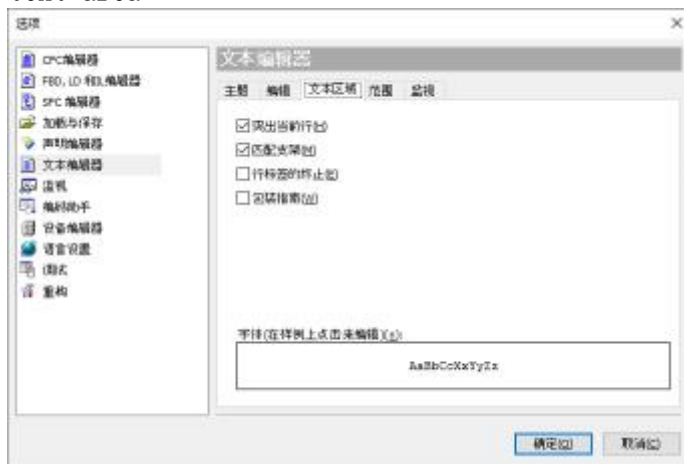


The specific functions are as follows:

project name	default setting	set content
Number of revocations	100	Set the number of times you can execute [Edit] - [Undo] in the menu bar. Setting range: 1~1000Second-rate.
fold	indentation	Specifies a way to define the structure of the code. none: Code is not folded. indentation: Combines all lines indented from the previous line into a single unit. Explicitly: Display the markup as a 1-unit code section in the comments.
word packing	none	Sets the word wrapping rule for the input string. none: Code is not folded. Soft line break: When the number of characters entered in 1 line exceeds the value of "Repeat Margin", a code continuation marker ":" is added and the line is automatically wrapped. if "Repeat margins:" input in "0", will wrap at the far right of the editor window. Hard line break: When the number of characters entered in 1 line exceeds "repeat margin" value will automatically wrap. but the code continuation marker will not be added ":". Also, if the first entered word has more than "repeat margin" value, the repetition will not be performed.
Packaging range	0	Specifies the number of characters to wrap. Setting range: 0~240

project name	default setting	set content
Tab width	4	Specifies the number of characters to convert tabs to spaces. Setting range: 1~16
self-modulation table	enable	Set to convert tabs to whitespace or to remain tabs. Enabled: Keep tabs as tabs. Disabled: Tab characters are entered as whitespace characters.
Indent length	4	"auto indent" settings selection "automatic" , "auto-encoding" , inserts a tab character of the specified width. but if "keep tabs" Set as "disabled" , whitespace characters will be inserted. Setting range: 1~16
auto indent	auto-encoding	Sets the action when auto-indenting is performed. none: Do not automatically insert indentation. yuan: Inserts an indent of the same width as the indent of the previous line when wrapping. automatic: Lines after lines containing keywords (VAR, etc.) will be "Indent width" The settings automatically insert indents. Autocode: remove "automatic" behavior, will automatically insert "END_IF" , "END_VAR" and other corresponding keywords.

- text area

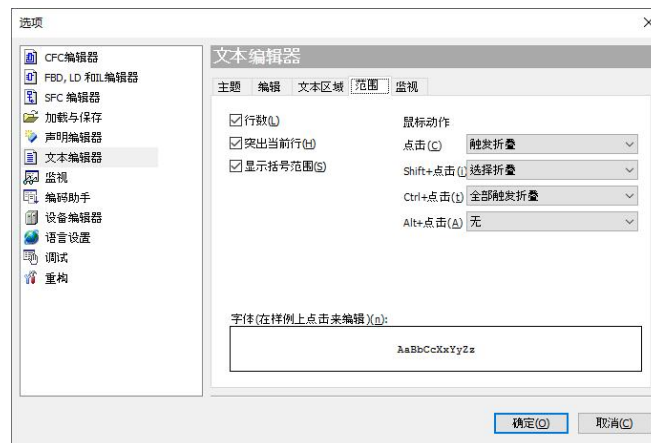


The specific functions are as follows:

project name	default setting	set content
highlight current line	enable	Highlight the line selected by the cursor. enable/disable.

matching bracket	enable	When the cursor is at the position of a parenthesis in the code, the paired parentheses are highlighted. enable/disable.
end of line label	disabled	Small dashes with the colors set in the theme". Marks the end of the line. enable/disable.
Packaging Guidelines	disabled	The vertical line display reference set in the theme is used as the basis for wrapping in the column. If "Wrap Margin" set "0" value other than the reference line is displayed. enable/disable.
font	—	Displays the Font dialog box for setting the font.

- scope

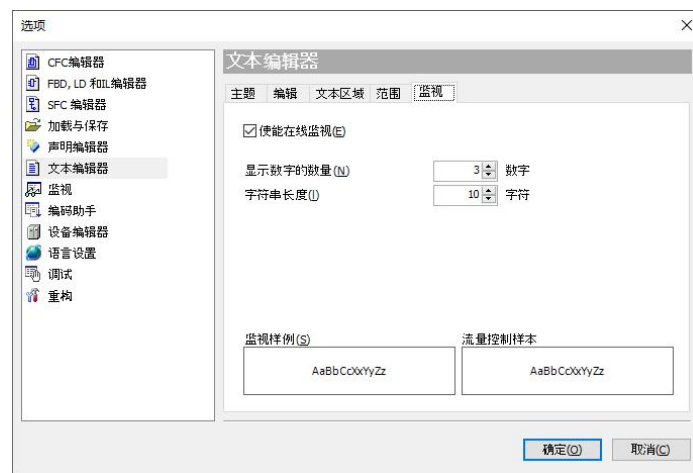


The specific functions are as follows:

project name	default setting	set content
Rows	enable	Display line numbers in the variable declaration section and the program implementation section.
highlight current line	enable	Enabled: Display line numbers. Disabled: Do not display line numbers. Change the color and highlight of the line number selected by the cursor. Disable "line numbers" is not highlighted. Enabled: Changes the color and highlighting of line numbers. Disabled: Does not change the color of line numbers.
show bracketed range	enable	Display keywords in the margin section to the left of the line number (IF~END_IFetc.) range from start to end. Enabled: Display range. Disabled: The range is not displayed.
		Assign the click to the margins section of the "+", "-" mouse action.

project name	default setting	set content
mouse action	—	None: No mouse action is assigned. Collapse selection: Selects all lines in the bracketed area. Collapse Toggle: Expand or collapse the area in parentheses. Collapse All Toggle: If there are nests, expand or collapse all nested regions.
font	—	Displays the "font" of the set font"dialog.

- monitor



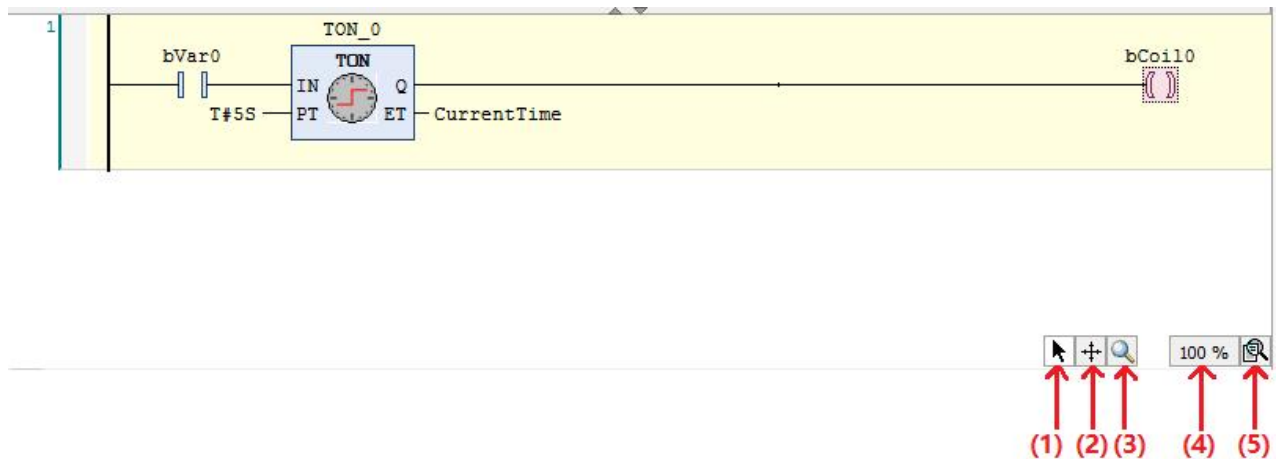
The specific functions are as follows:



project name	default setting	set content
Enable online monitoring	enable	In online mode, the program implementation section will display the monitoring fields. Enabled: Displays watch fields. Disabled: Watch fields are not displayed.
Display the number of numbers	3	Sets the number of digits displayed after the decimal point in the monitor field. Setting range: 1~20.
string length	10	Sets the maximum length of string variables in watch fields. Setting range: 1~80.

6. Window operation of the program input screen

The display size in the program screen can be enlarged by window operation.

Icons for window operations are displayed in the lower right corner of the screen.



Numbering	default setting	set content
(1)	normal mode	Clicking on an element selects that element.
(2)	mobile mode	Click and drag on the screen to move the screen.
(3)	magnifying glass tool	Window that launches the Magnifier tool. Enlarges the display content at the cursor position in the window.
(4)	current display size	Displays the display size of the current program screen.
(5)	Change display size	<p>Change the display size. Clicking the button will display the menu. Please select the changed size.</p>  <p>If you choose , will show "enlarge" dialog box where you can enter a magnification.</p>

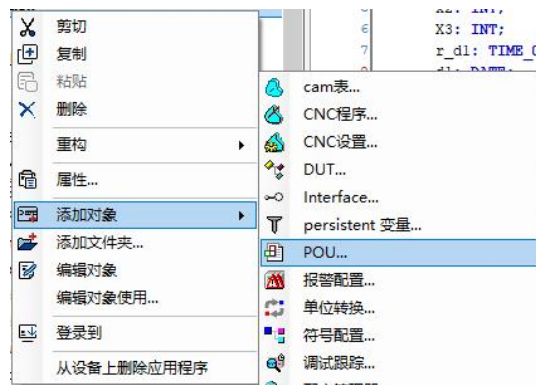
2.6.3. Create Program Objects (POU Objects)

1. Create a program

Programs are created in POU objects. Only one program can be used in one POU object. To use different programs within the project, you need to add POU objects.

2. Add to POU object

To add a POU object, right-click the [Application] object in the navigation bar window, and select Add Object → POU from the displayed menu.

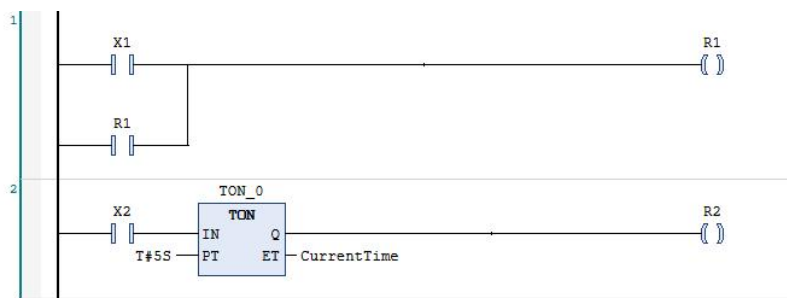


2.6.4. Types of programming languages

STEP AS supports 6 programming languages according to the PLC international standard IEC 61131-3.

1. Ladder program (LDprogram)

It is a graphical program created by placing ladder elements such as contacts and coils on a network (circuit). Functions and function blocks with various functions are also available.



2. Structured Text Programs (ST Programs)

A program created by describing expressions, conditional statements, etc. in text format. It is based on the programming language PASCAL and is suitable for processing such as numerical operations, data processing, conditional branching and repeated processing

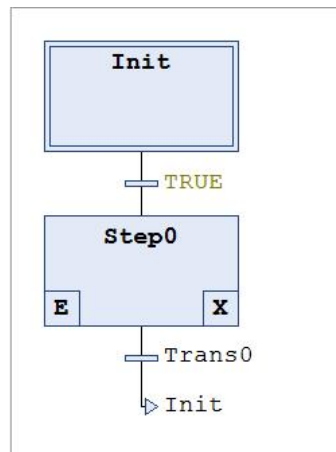
```

1  x1 := i1 + 1;
2  x2 := i2 + 2;
3  x3 := i3 - 1;
4  ADD_3(x1,x2,x3);
5  CASE x1 OF
6    1: x2 := 44;
7    2: x2 := 55;
8    3: x2 := 66;
9  ELSE x2 := 77;
10 END_CASE
11 IF (x2 = 66) THEN
12   x3 := 88;
13   ELIF (x2 = 77) THEN
14    x3 := 99;
15 END_IF
16 b1 S= b2;
  
```

3. Sequential function chart program (SFC program)

It is a graphical program created by placing elements such as steps, transitions, and

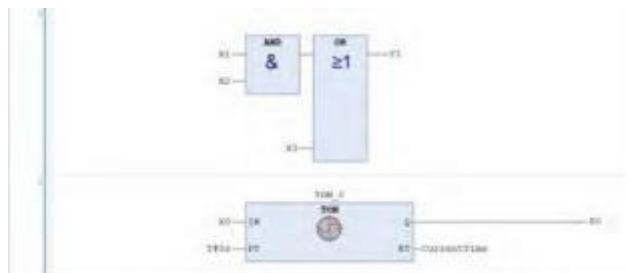
actions in a top-to-bottom order. Suitable for processing that describes state transitions.



4. function block diagram program (FBDprogram)

A graphical program created by placing functions and function blocks on a network (circuit).

Unlike ladder programs, ladder elements such as contacts and coils cannot be placed.



5. instruction list (ILprogram)

A program created by writing assembler-like instructions sequentially in text format.

Ideal for high-speed processing and when memory usage needs to be limited.

1	LD	iVar	
	ADD	3	
	ST	iResult	
	LD	bVar1	
	JMPC	mark1	
2	mark1:		
	LD	bVar2	
	S	bVar3	

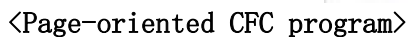
6. Continuous Function Chart program (CFCprogram/page orientedCFCprogram)

It is a graphical program created by placing elements such as function blocks on the screen.

Elements can be placed freely on the screen, and the order of execution can be specified.

There are CFC programs that create programs on one screen, and page-oriented CFC programs that create programs while switching screens called pages.

<CFC program>



1. Standard data types

The following types can be used as standard data types in STEP AS.

type	type	scope	size (bit)
true and false	BOOL	TRUE(1)and FALSE(0)	8
integer	BYTE	0~255	8
integer	WORD	0~65535	16
integer	DWORD	0~4294967295	32
integer	LWORD	0~264-1	64
integer	SINT	-128~127	8
integer	USINT	0~255	8
integer	INT	-32768~32767	16
integer	UINT	0~65535	16
integer	DINT	-2147483648~2147483647	32
integer	UDINT	0~4294967295	32
integer	LINT	-263~263-1	64
integer	ULINT	0~264-1	64
floating decimal point	REAL	-3.402823e+38~3.402823e+38	32
floating decimal point	LREAL	-1.7976931348623158e+308~1.7976931348623158e+308	64
string	STRING		(Number of characters+1)×8
string	WSTRING		(Number of characters+1)×16

type	type	scope	size (bit)
time	TIME	0~4294967295	32
time	LTIME	0~213503d23h34m33s709ms551us615ns	64
time	TIME_OF_DAY	0 (00:00:00:000) ~ 4294967295 (11:59:59 PM: 999)	32
date	DATE	0 (1970-01-01) ~ 4294967295 (2106-02-07)	32
date and time	DATE_AND_TIME	0 (1970-01-01, 00:00:00) ~ 4294967295 (2106-02-07, 06:28:15)	32

① User-defined data types of structures, enumerations, aliases, communities, etc. can also be used.

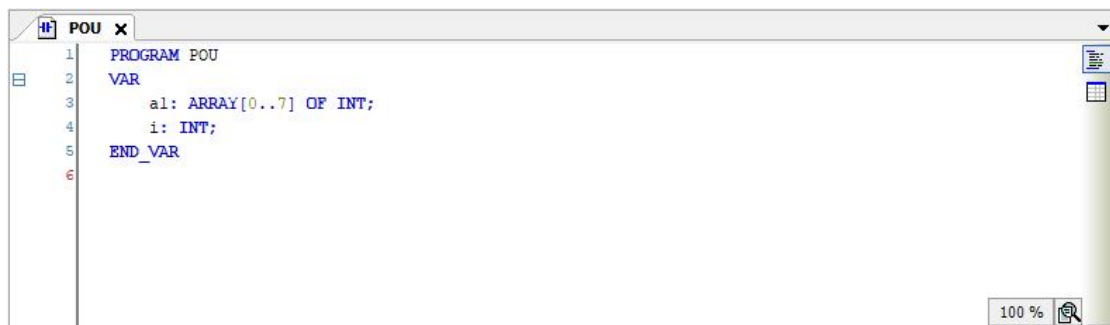
2. Array

Arrays can be used in STEP AS.

By using arrays, multiple data can be used as 1 variable.

Handy when you want to work with variables of the same type together.

Example: When a one-dimensional array a1 with 8 INT data is declared and used in an ST program



① When accessing a variable of an array, you can automatically check whether the index is within the declared range. Please use bounds-checked POUs in POUs for automatic checking.

① Using the auto-declare array wizard, you can declare variables of an array simply by entering the index and primitive type.

3. subrealm type

Subrealm types can be used in STEP AS. Subrange types can specify ranges for values of standard data types.

The following is an example of a string declaration of a variable of type subworld.



If you try to assign an out-of-scope value to a variable, you will get an error at compile

time.

Subrealm types can also be declared in form format.



① When accessing a subrange type variable of type DINT, UDINT, LINT, ULINT, you can automatically check whether the value is in the declared range.

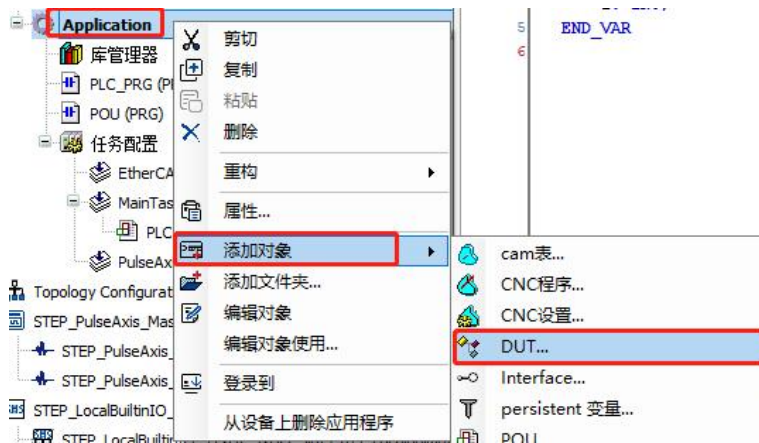
within. Please use the POU for range checking in the POU for automatic checking.

4. Structure/enumeration/alias/community

User-defined data types of structures, enumerations, aliases, and unions can be declared through DUT objects.

To use these data types, add DUT objects to the project.

① Right-click the [Application] object in the navigation bar window, and select Add Object → DUT in the menu.



Displays the Add DUT dialog. "name" is the name by which the type is accessed in the program.



②After selecting the type to be defined and entering the required information, click the [Open] button.

A DUT object that defines the type of selection is added to the navigation bar window.

Example: adding a structure



③ Select the added object and enter the definition content in the main window. The definitions and usage of each type are as follows.

a. structure

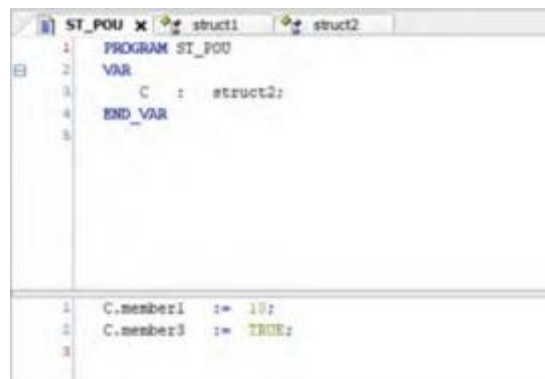
is an example of declaring structures struct1 and struct2. struct2 extends struct1. To extend the declaration, check Extends in (2), and enter the extended declaration.





① declared as struct2. The variables of the struct can be accessed1 and struct2a member of.

Example: ST program to access members of structure struct2

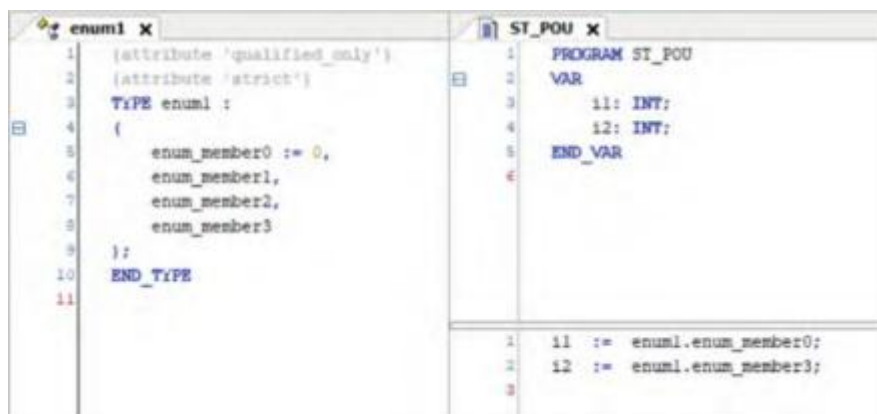


① You can use the BIT type as a member of a structure. Available values are TRUE (1) or FALSE (0). The size of the BIT type is 1 bit.

b. Enumeration

is an example of an ST program that defines a declaration of enum enum1 and accesses the members of enum1.

Variable iVar0 and variable iVar1 are assigned 0 and 3, respectively.

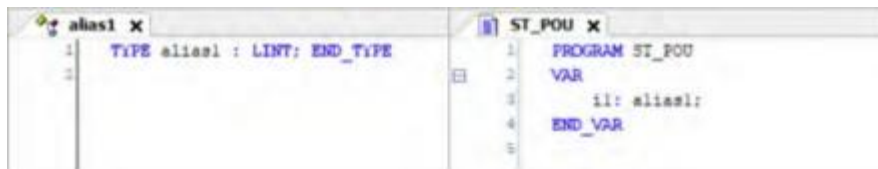


c. Aliases

Aliases allow users to assign user-defined names to type names. Please declare the variable with the defined alias in the declaration section.

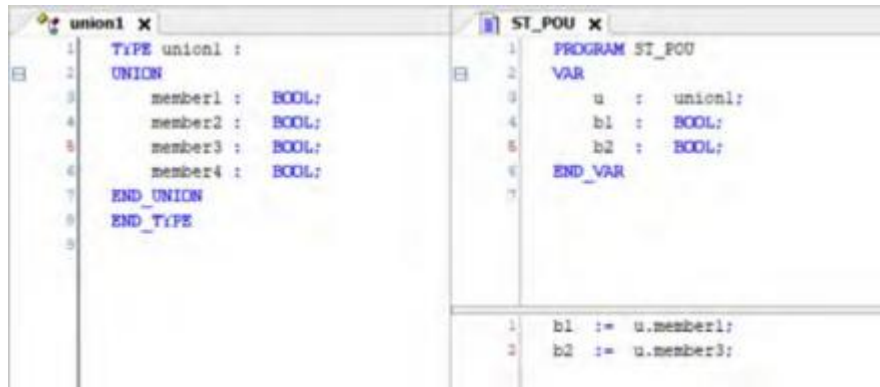
Here is an example of a declaration that defines an alias of alias1 of LINT and a declaration part that declares a variable iVar0 of type alias1.

A variable iVar0 declared with type alias1 will be treated as a variable of type LINT.



d. Community

This is an example of a declaration of a union that defines a union of union1 and an ST program that accesses members of union1.



5. Constants

Constants can be used in STEP AS.

Constants are declared using the following syntax. VAR CONSTANT

constant name:type:=initial value;

END_VAR

type	type	content
BOOL	BOOL	TRUE (1), FALSE (0)
integer	Types available for numeric values	Binary, Octal, Decimal, Hexadecimal For non-decimal numbers, describe integer constants after the base and # Examples: 14, 2#0101, 8#27, 16#34AB
Decimals and Exponents	REAL/LREAL	Decimals and Exponents Example: 1.4, 2.34e+008
time	TIME	32bit matches IEC 61131-3 time constant of Syntax: t#, T#, time#, TIME# Example: T#12ms, T#12h32m twenty fours
time	LTIME	64-bit time constant. except TIME In addition to constants, the following units can also be used microseconds: m Nanoseconds: ns Syntax: LTIME# Example: LTIME#123m456ns
time	TIME_OF_DAY	time Syntax: tod#, TOD#, time_of_day#, TIME_OF_DAY# Example: tod#12:24:20.123

type	type	content
date	DATE	date Syntax: d#, D#, date#, DATE# Example: d#2018-01-01
datetime	DATE_AND_TIME	datetime Syntax: dt#, DT#, date_and_time#, DATE_AND_TIME# Example: dt#2018-01-01-07:04:13
string	STRING, WSTRING	enclosed in single quotes Example: 'Hello World'

6. Objects for declaring global variables

Global variables that can be used throughout the project can be used in STEP AS.

- list of global variables
is the object used to declare global variables.

Variables declared using the global variable list can be accessed using the object name.global variable name of the global variable list.

Example: When accessing the variable of the object GVL of the global variable list in the ST program



- List of persistent variables
is an object used to declare global variables for persistent variables.
Select "Persistent Variable" in Add Object.

7. Global variables

You can use global variables that are common across projects.

Global variables are declared in objects in the global variable list (GVL).

This section describes how to declare global variables and how to access declared variables.

- ① Double-click the GVL object on the navigation bar window.



Displays the GVL screen in the main window.

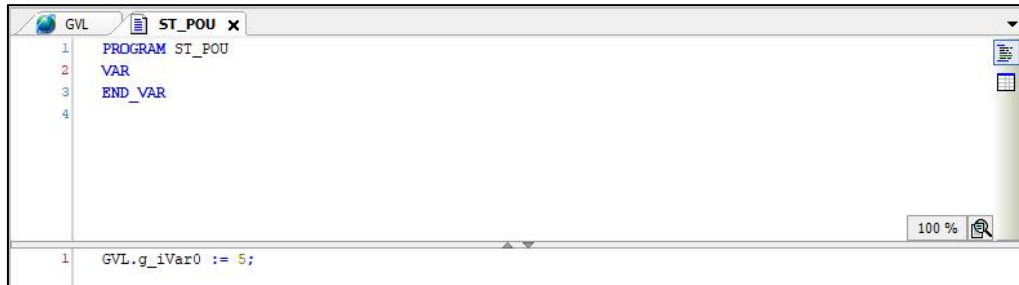
- ② Declare variables in the global variable list (GVL).

Example: declare a global variable g_iVar0 of type INT



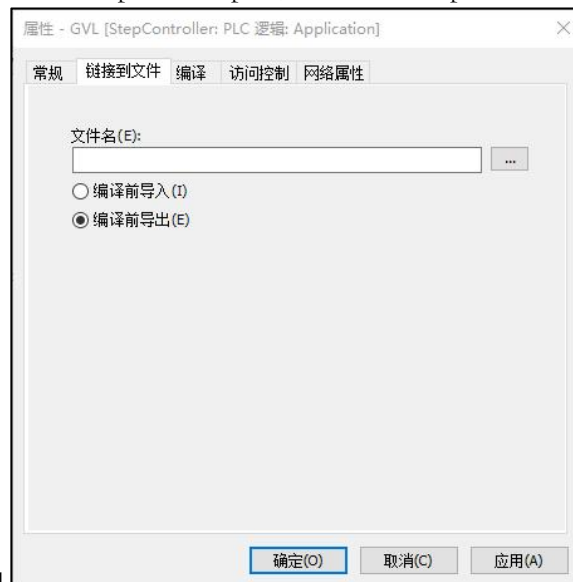
The declared variable can be accessed from the program as "name.variablename"

Example: Assign value "5" to global variable g_iVar0



① Variables declared before compilation can be imported and exported in XML format.

Right-click the object in the global variable list and select Properties. The Properties dialog box will appear, open the Link to File tab screen, select the item to import or export, and enter the path to the file to import or



export in the Filename field.

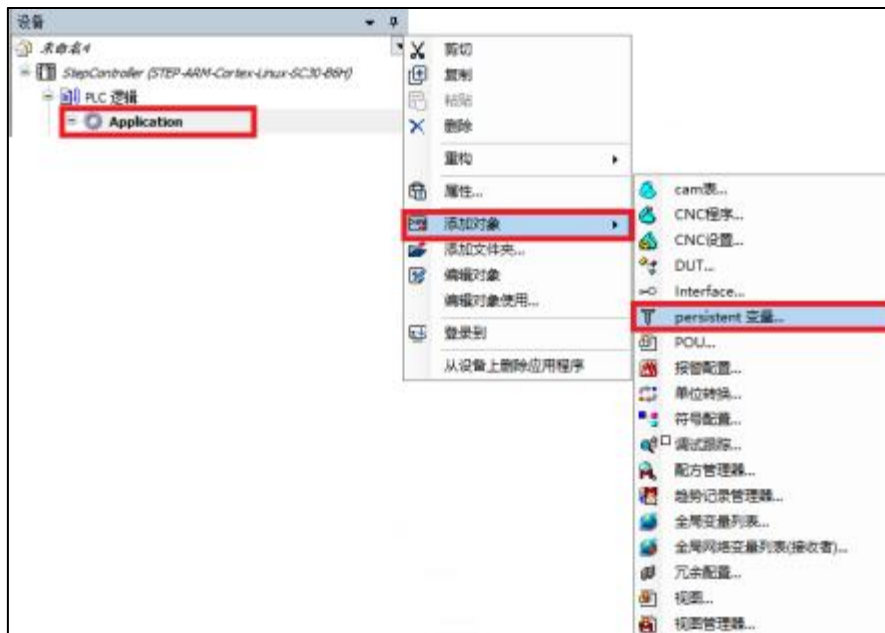
8. Persistent variables

On reset, global variables can be used as persistent variables, which can hold values without initialization.

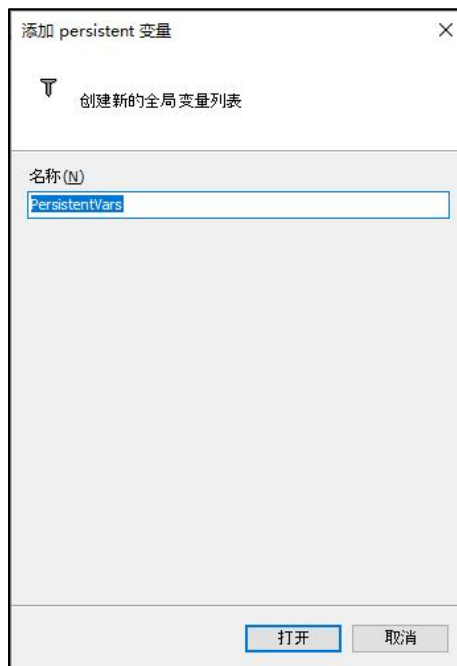
A persistent variable that can be used as a global variable is declared in the object of the persistent variable list.

Only 1 object of persistent variable list can be registered.

① Right-click the [Application] object in the navigation bar window, and select Add Object → Persistent Variable from the displayed menu.



Displays the Add persistent variable dialog.



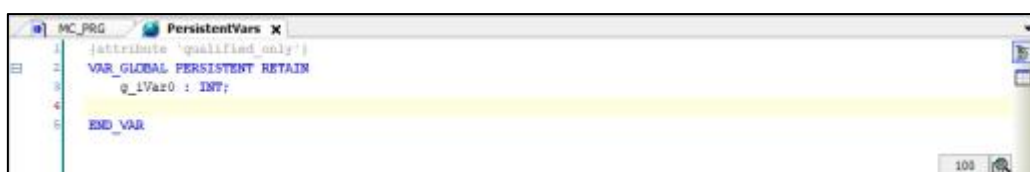
② Enter the name of the persistent variable list, and then click the [Open] button.

A "Persistent Variable List" object is added to the Navigation Bar window.

③ Declare variables in the persistent variable list.

Example: declare the global variable `g_iVar0` of the persistent variable of type `INT`

The declared variable can be accessed from the program as "name.variablename".



Example: Assign value "6" to global variable `g_iVar0` of persistent variable



① Persistent variables for local use can be declared (VAR PERSISTENT RETAIN) in the declarations section of each POU object.

➤ ① The instance paths of persistent variables declared in each POU object can be added to the persistent variable list.

① With the Declarations section of the Persistent Variables list selected, select Declarations → *add all instance paths*.

9. Shorthand format function

You can enter fewer characters for variable declarations if you use the shorthand formatting feature in the declaration section in string form.

Example: When declaring variables bVar0 and bVar1 of type BOOL using the shorthand format function

Enter the variables bVar0 and bVar1 and press <Ctrl>+<Enter>.

"bVar0, bVar1:BOOL;" is automatically entered.



The pattern of an input example using the shorthand format function is shown below.

Strings described after a semicolon (;) are processed as comments.

Enter in short form	press <Ctrl>key+<Enter>result after key
bVar0	bVar0:BOOL;
iVar0 iVar1 I 6	iVar0, iVar1: INT := 6;
strVar S 8	strVar: STRING(8)
wVar w; wVar comment	wVar: WORD; // wVar comment

2.6.6. Functions and function blocks

Functions and function blocks can be called from the program. Functions and function blocks can be created using POU objects.

Functions and function blocks have the following differences.

function(FUN)

- a) Can be used without a declaration in the declarations section.
- b) There is only 1 output. But other outputs can be defined.
- c) The values of output variables and internal variables are not saved.

function block(FB)

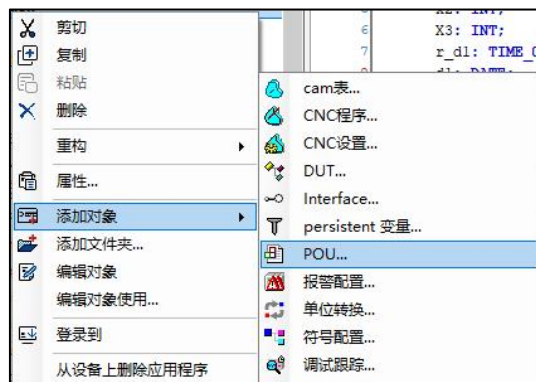
- a) It can be used by declaring the instance in the declaration section.
- b) There can be multiple outputs.
- c) Save the values of output variables and internal variables.
- d) Object-oriented definition using inheritance (EXTENDS), interface implementation (IMPLEMENTS), access modifiers is possible.

1. Function

A function performs 1 output for 1 or more inputs. Functions can be used without declaring variables.

For example, to create and call the function "ADD_SUB" that takes 3 INT-type parameters as input, calculates (1st parameter) + (2nd parameter) - (3rd parameter) and outputs, follow the steps below .

① Right-click the [Application] object in the navigation bar window, and select Add Object → POU from the displayed menu.

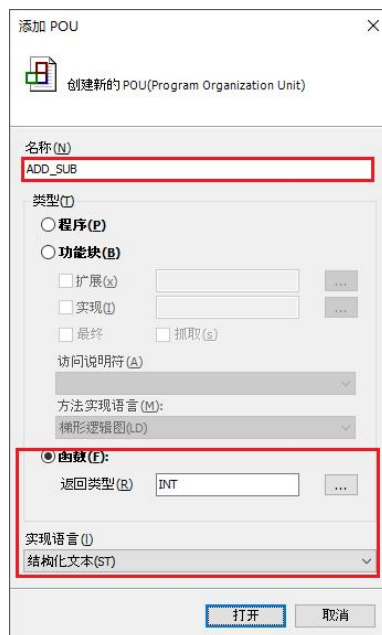


The Add POU dialog is displayed.



② Select "Function" and choose the input and description language for the name, return type.

Name selects the function name. The return type selects the return value when the function is executed. Description Language Select the programming language that describes the processing of the function.



③ Click the [Open] button. A POU object with functions added. POU objects are displayed as "Function Name (FUN)" in the navigation bar window.



④ Handling of input functions.

Open the function's POU object and create the function. Declare the input variables of the function in "VAR_INPUT". Assign the output of the function to the variable of the function name.

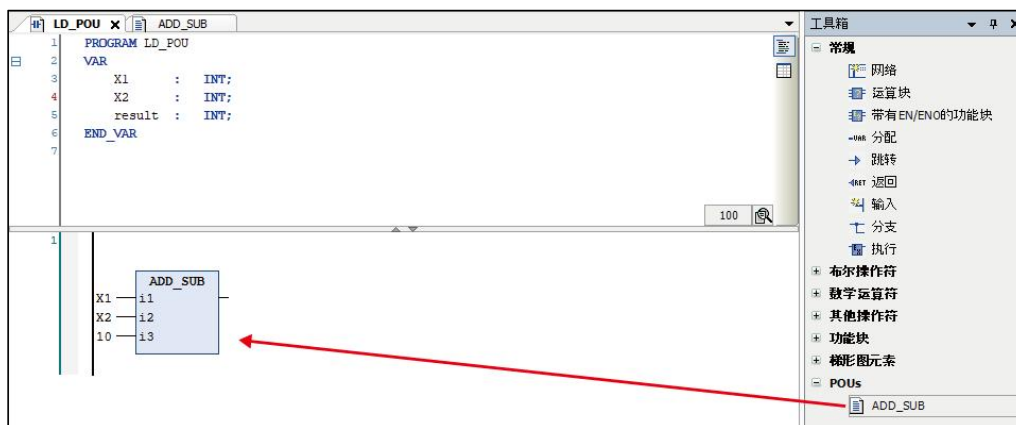


① At this point, the steps to create the function are completed. The steps to call the created function are described later.

⑤ Open the POU object where you want to call the function, and call the function.

a) Functions can be called using the function name. Calling a function does not require declaring variables.

Example: When calling from an LD program



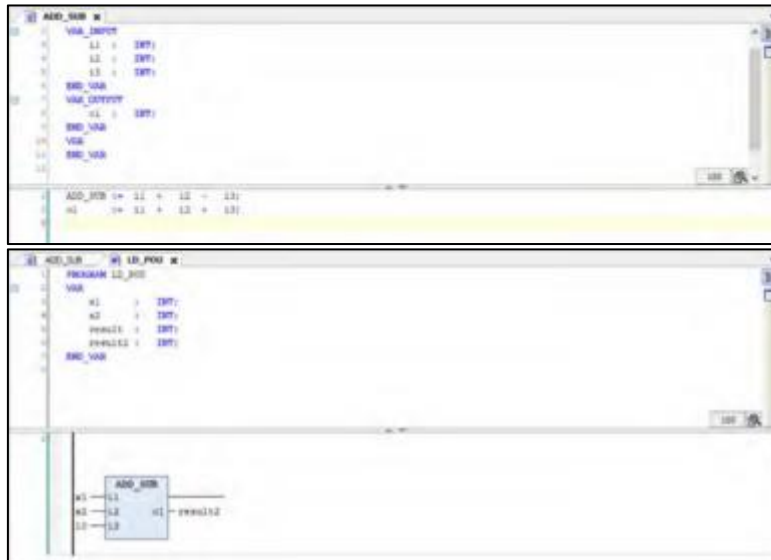
Example: When calling from ST program



b) Additional outputs can be defined for functions. Declare the variable "VAR_OUTPUT" in the declaration section of the POU object that defines the function.

Example: The definition of the function "ADD_SUB" that outputs the variable iOut that outputs the sum of the three input variables is added

(1). Call the "ADD_SUB" function in the LD program



(2). Call the "ADD_SUB" function in the ST program

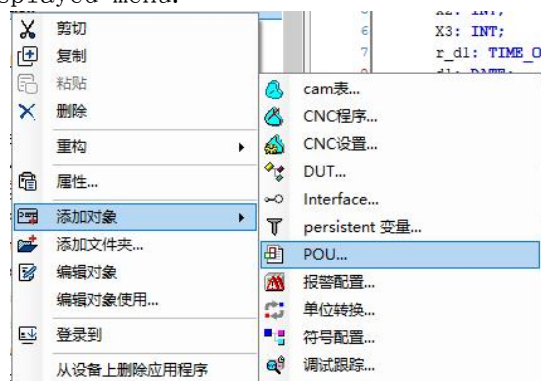


2. Function block

A function block performs 1 or more outputs for 1 or more inputs. Using function blocks requires declaring variables (instances).

For example, when creating a function block "FB_ADD" that takes 3 INT type variables as inputs and outputs the sum of 3 parameters and calls an instance, follow the steps below.

① Right-click the [Application] object in the navigation bar window, and select Add Object → POU from the displayed menu.



The Add POU dialog is displayed.



② Select Function Block, enter a name and select a description language.

Name Select the function block name. Description Language Select the programming language that describes the processing of the function block.



③ Click the [Open] button.

Added POU objects for function blocks. The POU object is displayed as "Function Block Name (FB)" in the navigation bar window.



④ The processing of the input function block.

Open the POU object of the function block to create the function block.

Declare the input variables of the function block in "VAR_INPUT". Declare the output variables of the function block in "VAR_OUTPUT".

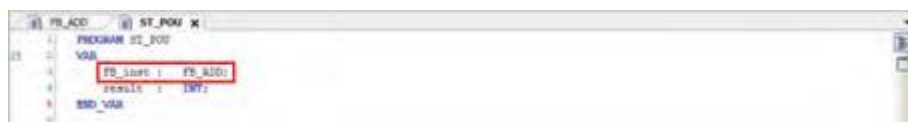


① At this point, the steps to create the function block are completed. The procedure for calling the created function block is described later.

⑤ Open the POU object of the calling source of the function block, and declare the instance of the function block in the declaration part.

Declare the instance as a copy of the function block.

Instance name: declared in the form of a function block name.



⑥ Instance of the calling function block.

When an instance of a function block is called, the processing defined in the function block is executed. Input variables and output variables can be accessed using instance.variablename.

Example: call in LD program



Example: call in ST program

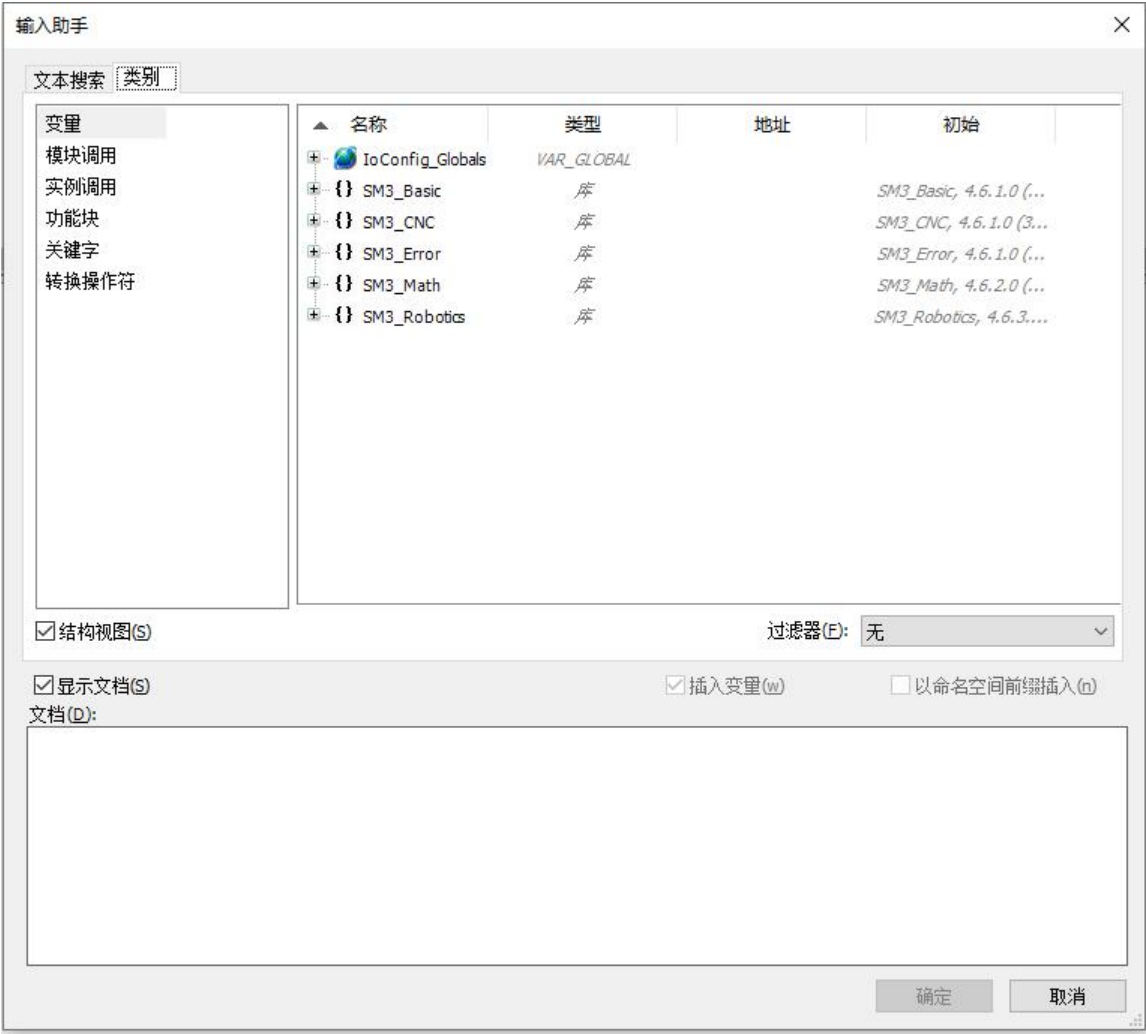


2.7. input assistant

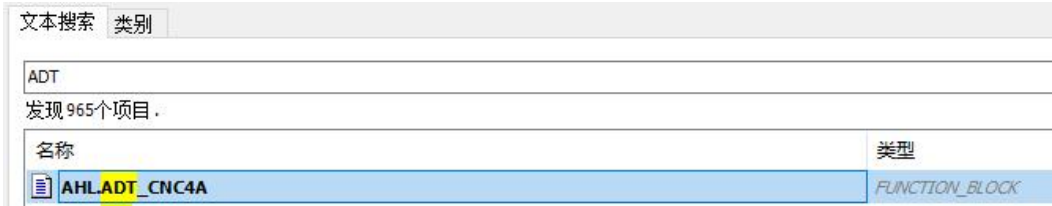
When writing code, inserting library or FB (function block), you can quickly prompt existing variables or function blocks and insert them into the code.

2.7.1. Start input assistant

When writing code, you can pass *editor* → *input assistant* or press **F2** to open the "Input Assistant" dialog box.

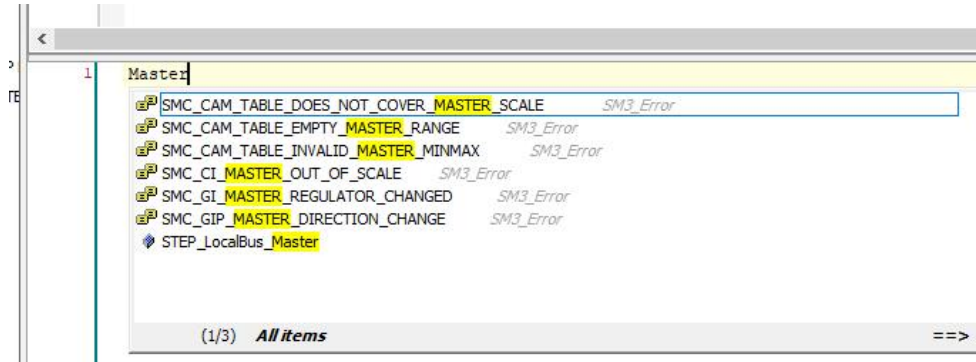


As shown in the image above, it is possible to filter by category. Users can also search directly by text:



2.7.2. coding assistant

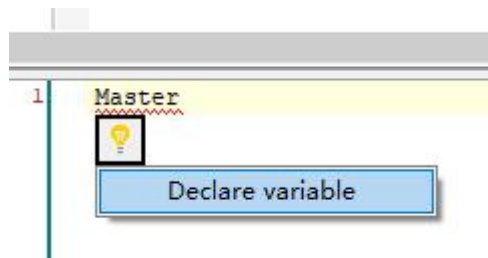
If you know the partial name of the variable, you can type it, and the system will automatically prompt, as shown in the following figure:



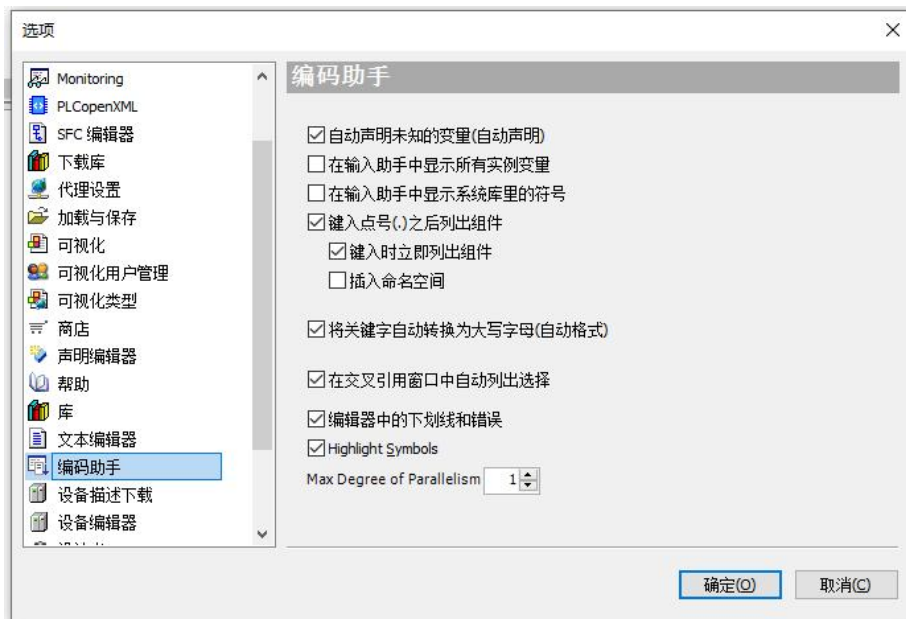
At the same time, you can type a '.' dot to prompt for global variables.

For undeclared variables, STEP AS provides a shortcut function for automatic declaration,

as shown in the following figure, click  Follow the prompts to complete the operation.



The assistant can be set through the menu Tools → Options, as shown in the following figure:



第三章 System Configuration

3.1. Controller configuration

Double-click the device node in the navigation bar or double-click the device node on the configuration page to open the configuration page of the corresponding device.

The generic device editor can include the following options:

- Communication settings: development systems and programmable devices (PLC) configuration for connections between. in pureI/O Not available under device.
- Applications: List of applications on the controller.
- Backup and Restore: "Configuration for file transfers between the host" file system and the device.
- log: PLC Display of log files.
- PLC set up: deal with I/O Configuration: Which application, behavior in stop state, update, bus cycle options, etc.
- users and groups: User management of the device at runtime.
- access permission: Access object and file permissions on the device.
- Task configuration: All input and output assignments.
- state: Specific device status and diagnostic messages.
- information: General equipment information (name, vendor, version, etc.).

3.1.1. Communication settings

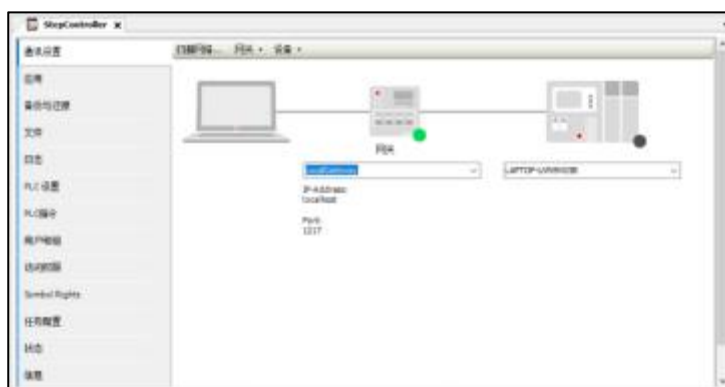


Figure 3-1 Communication settings

Scan Network: Opens the Select Devices dialog, displaying a list of gateway configurations and devices linked to those gateways. A target device can be selected from this list.

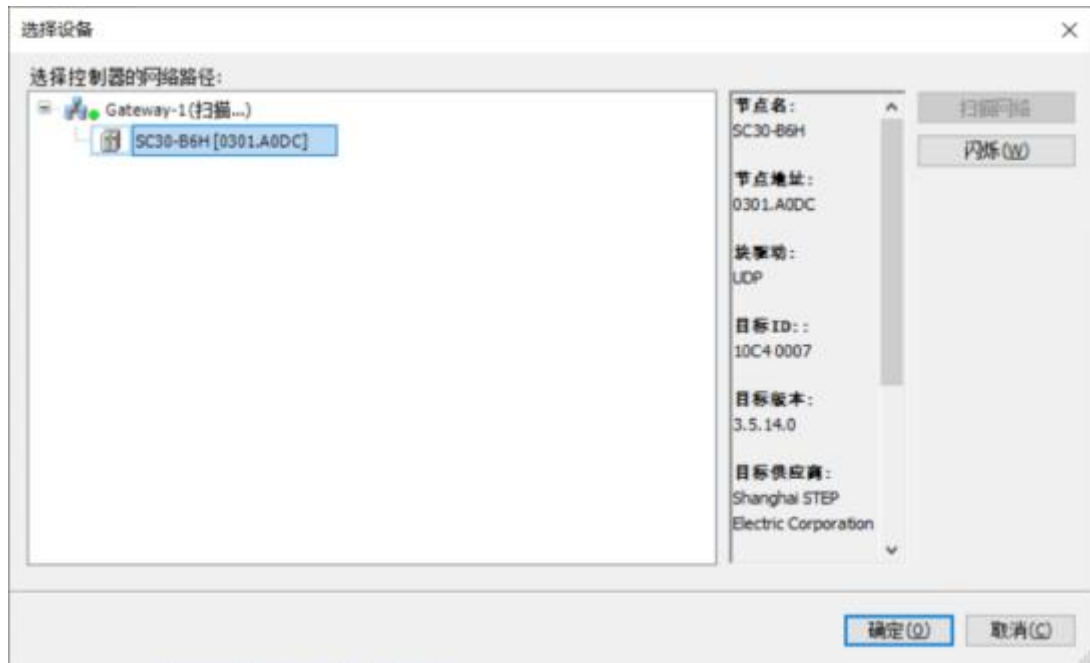


Figure 3-2 Select Device dialog box

● Gateway:



Figure 3-3 Gateway selection

- a. Add New Gateway: Opens the Gateway dialog to define a new gateway.
- b. Manage gateways: Open the "Manage Gateways" dialog box to display all gateways. Their order can be added or removed or changed.
- c. Configure the local gateway: Open the "Gateway Configuration" dialog. A block driver can be configured for the local gateway.

● equipment:

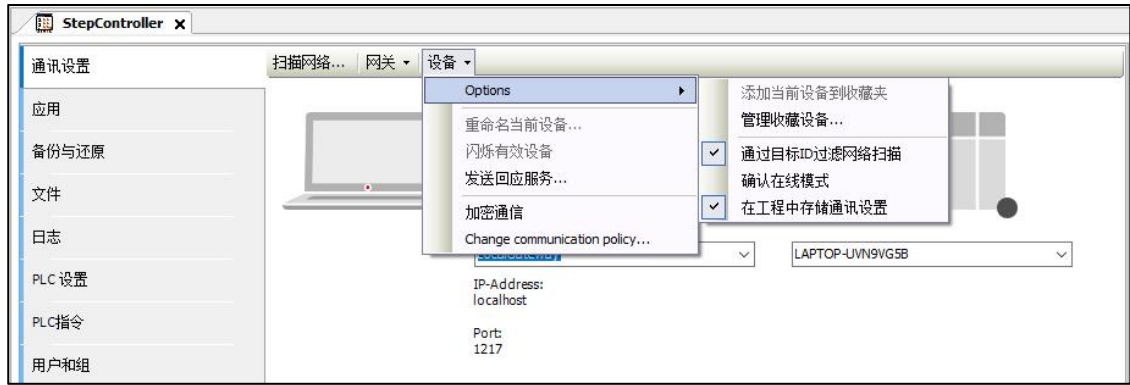


Figure 3-4 Device selection

- ✧ Add Current Device to Favorites: Add the currently set device to the list of favorite devices.
- ✧ Manage favorite devices: Open Favorites to display a list of all optimal devices. In this dialog, you can add or delete entries, or change their order. The upper device is the default.
- ✧ Rename current device: Open "Change Device Name" dialog.
- ✧ Blink valid device: Devices that support this feature will emit a flashing signal.

send response service: STEP ASSend five responses to PLC for testing network connections, similar to ping. The packet will not be sent the first time and only after that. The range of the packet depends on PLC communication buffer.

3.1.2. application

Get a list of apps on the device.



Figure 3-5 Device List

3.1.3. Backup and Restore

In the Generic Device Editor tabs, you can backup and save application-specific files on the PLC by saving and reading compressed files.

Require:The communication settings connected to the device are correct. A backup of the application is available on the PLC.

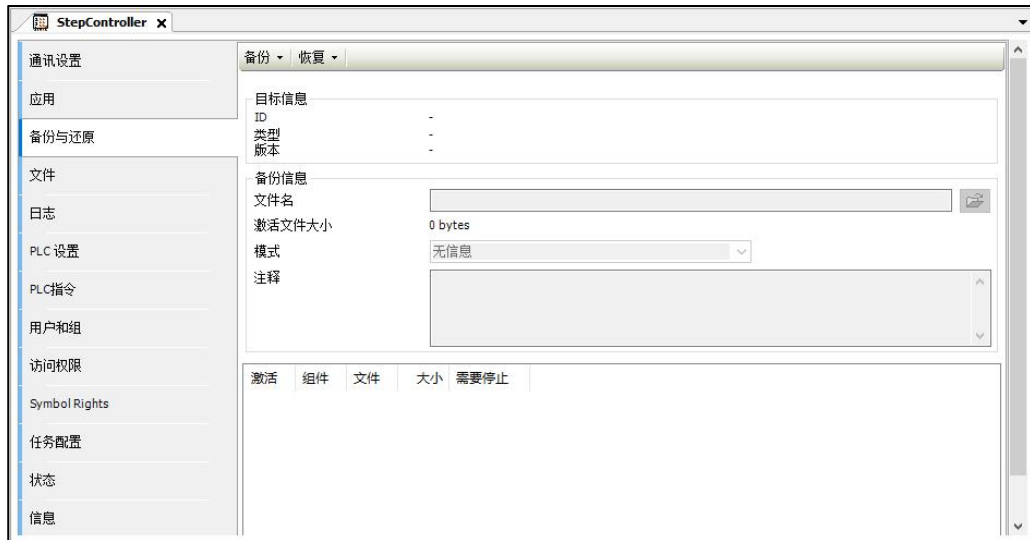


Figure 3-6 Backup and restore

Backup:You can read backup information from the device, create a backup file and save it to disk, and save the backup file to the device.

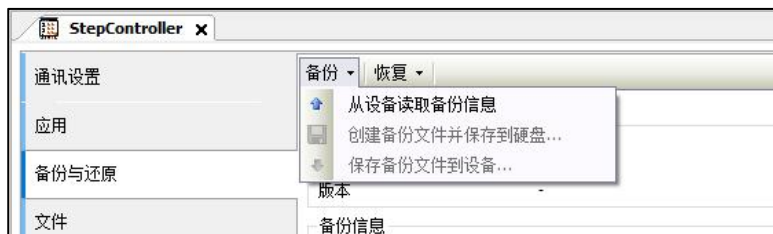


Figure 3-7 Backup file

recover:Backup files can be loaded from the hard disk or device, and backup files can be restored from the device.

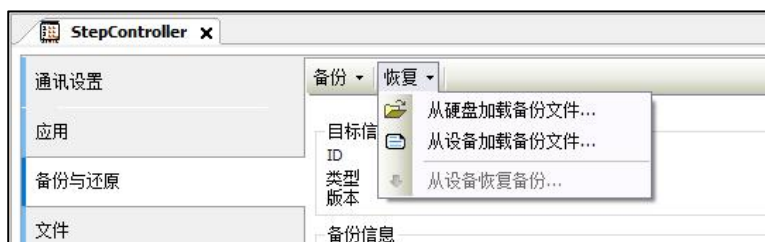


Figure 3-8 Restoring files

To restore the backup file on the device:This command is available if at least one of the components in the backup file is loaded and set to active on the current tab page. Used to

restore the state of the application on the device. The user interface is locked during the restore process. You can also cancel the operation.

3.1.4. document

Files can be transferred between this unit and the PLC. If the communication settings are correct and the PLC is online, then STEP AS will establish an automatic link with the PLC for continuous file transfer.



Figure 3-9 File

You can set a new file path, perform operations such as deletion and update, and copy the selected files and directories to other file systems. If the file does not exist in the destination folder, then the file will be created. If the file has already been created and is not write-protected, it will be rewritten.

3.1.5. users and groups

Depending on device support, user accounts and user groups can be defined. In combination with the "Acquisition rights" tab, control objects and files can be acquired in Runtime.

Require: The controller has user management and has login information to be able to log in to the controller.



Figure 3-10 Users and groups

Add, import, edit, and delete operations can be performed for both users and groups.

3.1.6. PLC settings

Basic PLC configurations can be implemented, such as handling inputs and outputs and bus cycle tasks.

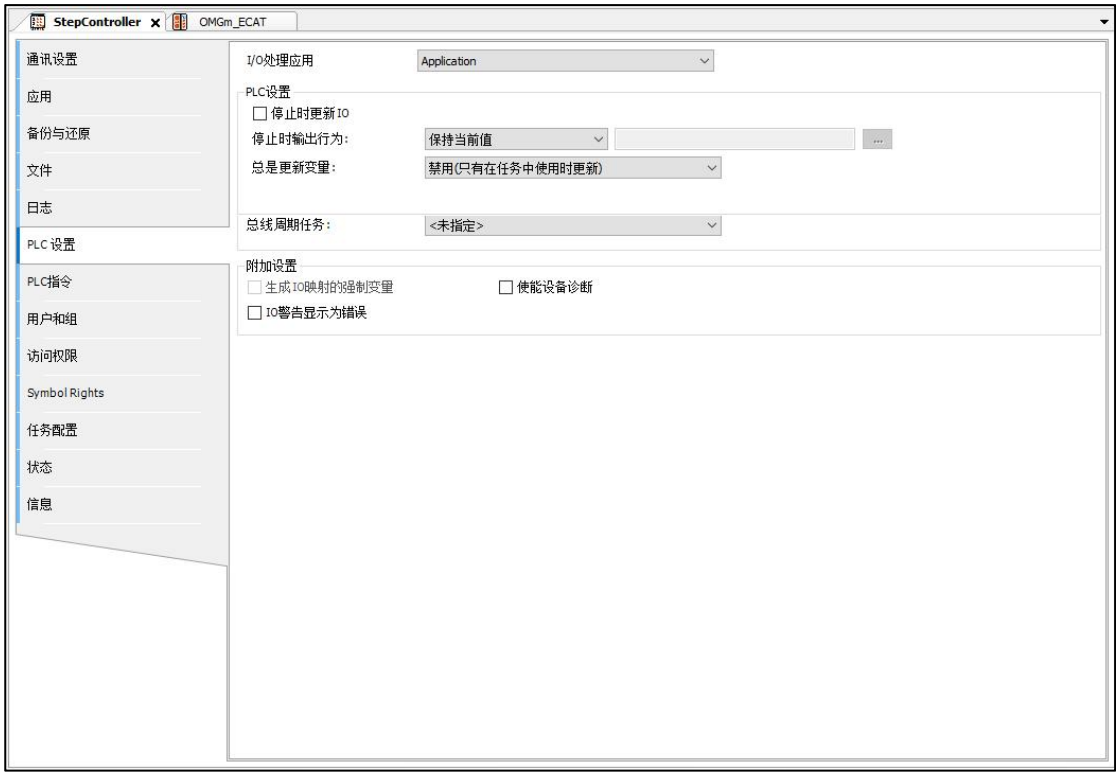


Figure 3-11 PLC configuration

I/O processing application: The application responsible for IO processing.

PLC settings:

- a. Update IO when stopped: If checked, the values of input and output channels will

be refreshed even if the PLC is stopped. If the watchdog detects a fault, the output will be set to a predetermined value. Otherwise the input and output channel values will not be refreshed.

b. Output behavior when stopped: You can keep the current value, set all outputs as default, and execute the program.

Always update variables:

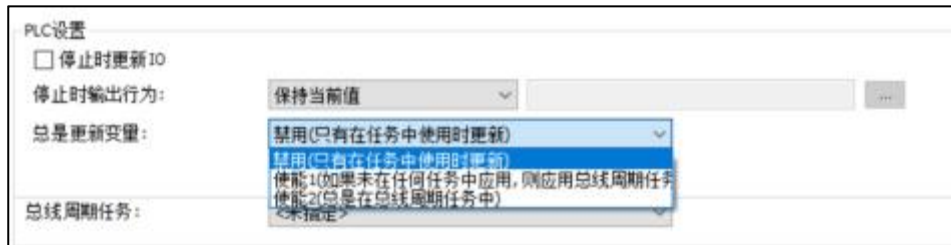


Figure 3-12 Update variable settings

The global setting defines whether or not tasks update I/O variables on bus cycles. The I/O variable settings of these modules and subordinate modules are valid as long as "invalid" is defined in the update setting.

- ▶ disabled: Will only update if used in a mission I/O variable.
- ▶ Enable1: If not used in other tasks, I/O variables are updated in the bus cycle task.
- ▶ Enable2: On every bus cycle the task updates all variables, whether they are used or mapped to input and output channels.

Bus cycle task: The task that controls the bus cycle, by default, has entered the task of the device description. By default, the bus cycle setting of the higher-level bus device (using the cycle setting of the upper-level bus) applies, and the device tree is scanned up for the next useful bus cycle task definition. Strictly heed the following tips:

(1) Before selecting the setting <unspecified> for the bus cycle task, you should pay attention to the following: ' <unspecified>' means that the default setting of the device description file takes effect. Therefore, you should check this description. Using the task with the shortest period can be defined as the default, but using the task with the longest period can also be defined

(2) For fieldbus, a fixed cyclic matrix is required to ensure a deterministic behavior. For this reason, do not use the "free run" type in a bus cycle task.

3.1.7. access permission

On the options page of the generic device editor, you can define access rights for PLC objects. Requirement: User management must be set up on the PLC.



3-13 User Privileges

3.1.8. log

The PLC log can be viewed in the Generic Device Editor, which lists events logged on the target system. Mainly include the following:

- ▶ Matters during system startup and shutdown (component loading, and versioning)
- ▶ Initiate the download and loading of the application
- ▶ custom entry
- ▶ I/ODriver's log entries
- ▶ Log entries for the data server

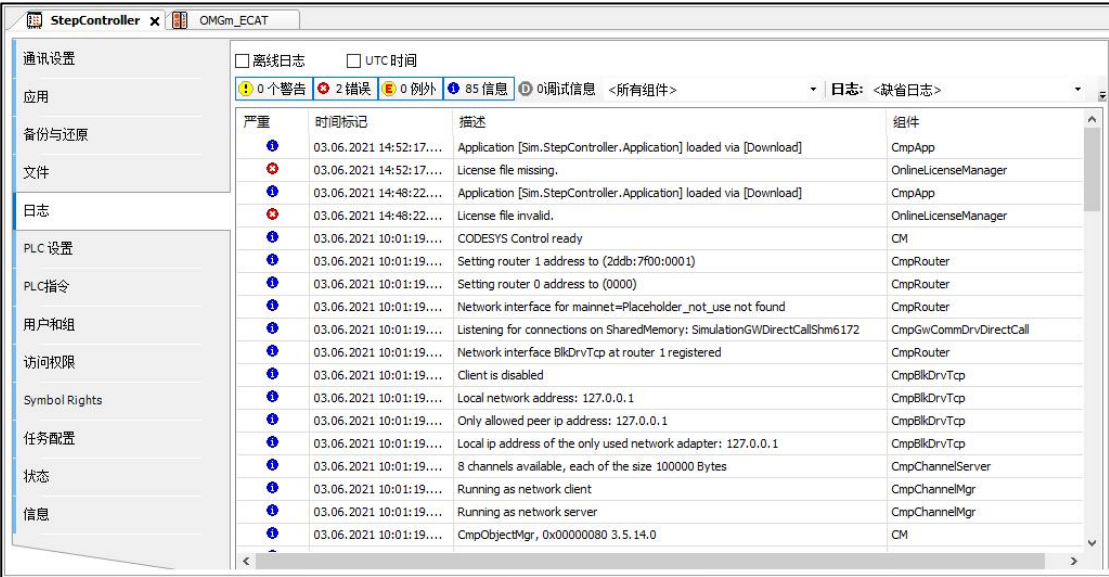
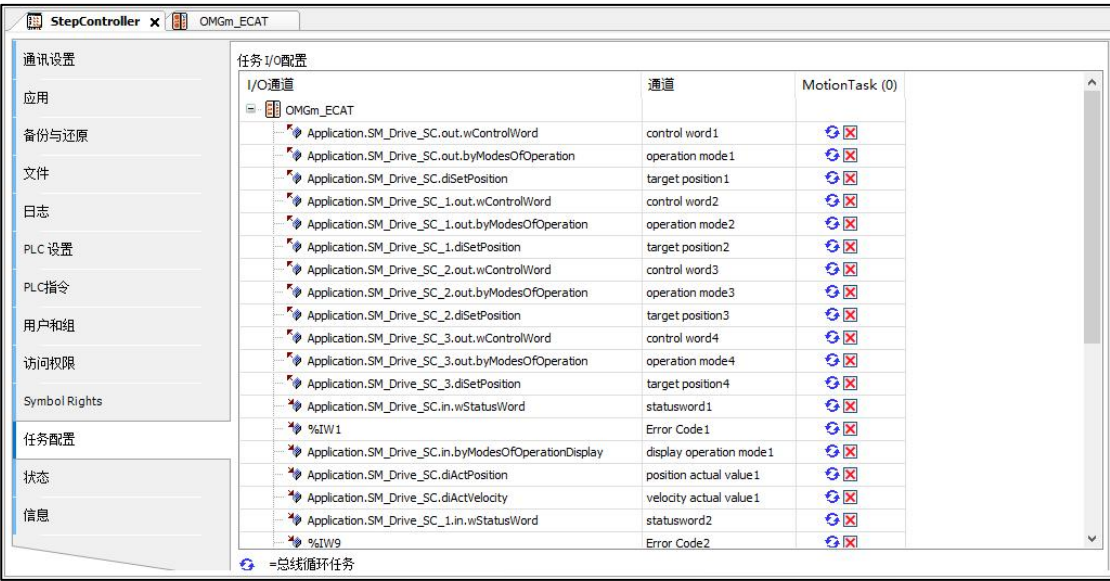


Figure 3-14 Log list

The severity of events can be divided into 4 categories: Information, Warning, Error, Exception. The toolbar above the list can be shown or hidden. The button for each category shows the number of log entries for the related category.

3.1.9. Task configuration

A sub-dialog of the device editor shows a table of inputs and outputs and their assigned tasks.



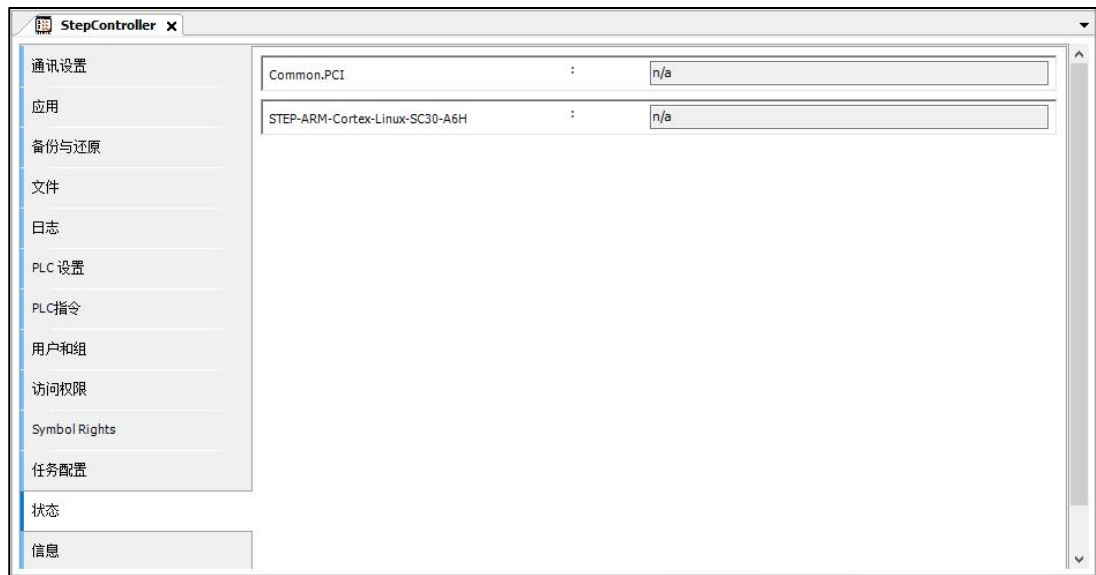
This information is only visible after the application code is compiled and generated. It is used for troubleshooting as it shows multiple input locations and output locations with different priority tasks. Using through multiple overrides can result in undefined values.

I/O Channels: Inputs and outputs of all related devices. This display corresponds to the I/O mapping in the Device Editor dialog. By double-clicking an input or output the associated I/O map editor can be opened.

Tasks: Displays the tasks defined in the task configuration. The title contains the task name and priority.

3.1.10. state

The generic device editor tabs display status information such as 'running' and 'stopped', as well as specific diagnostic messages for individual devices, as well as information used by the internal bus system.



3.1.11. information

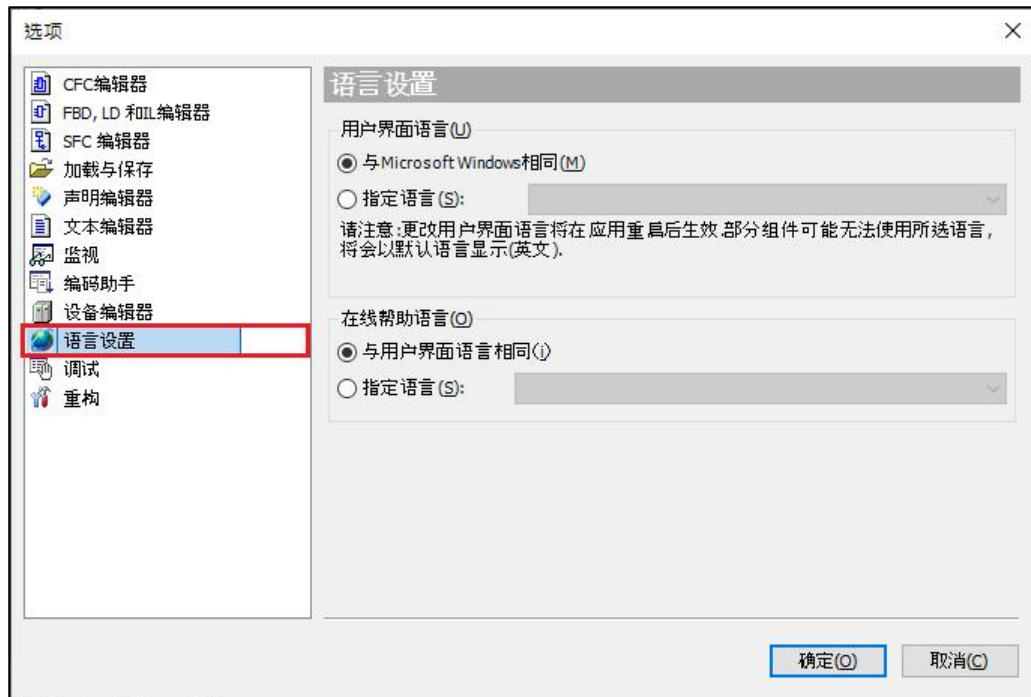
The tabs of the generic device editor show general information in the device description file, such as: name, vendor, class, version, serial number, description. The information items of the device involved in this article are the display of general information in the device description file.



3.1.12. display language

STEP AS supports Chinese and English. The language of the initial setting is the same as the language used by the OS. To use a language different from the OS, please make the display language setting. When making a language change, STEP AS needs to be restarted.

1. Select Tools-->Options from the menu. Displays the Options dialog.
2. Click "Language Settings" from the category window to display the "Language Settings" screen.



3. Select the language in the User Interface Language-->Specify Language column.

4. Click the [OK] button.

§ The "Options" dialog closes. The language will not be switched at this time. Exit STEP AS and restart STEP AS. The selected language takes effect after STEP AS is started.

3.1.13. version display

You can view the version numbers of components and devices such as software and compilers.

1. Select Help-->About in the menu bar. The "About" screen is displayed.



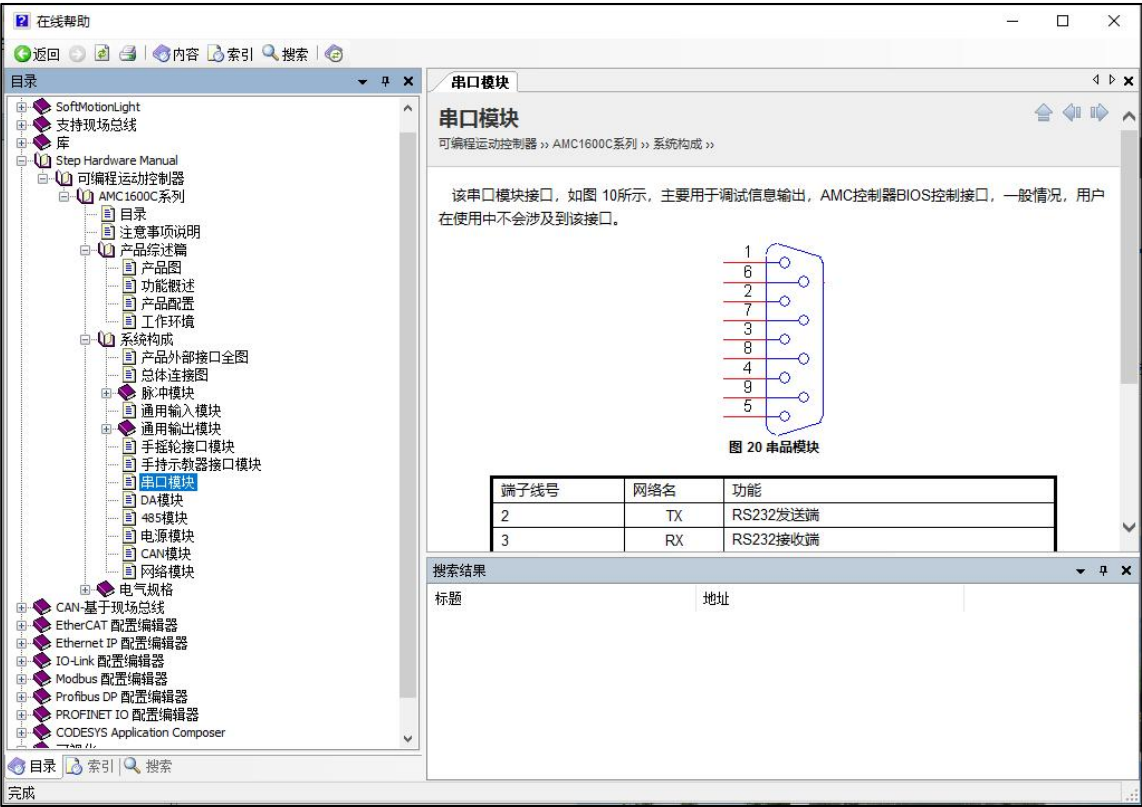
2. Double-click the device in the navigation bar and click the information on the pop-up device information page to view the device version.



3.1.14. online help

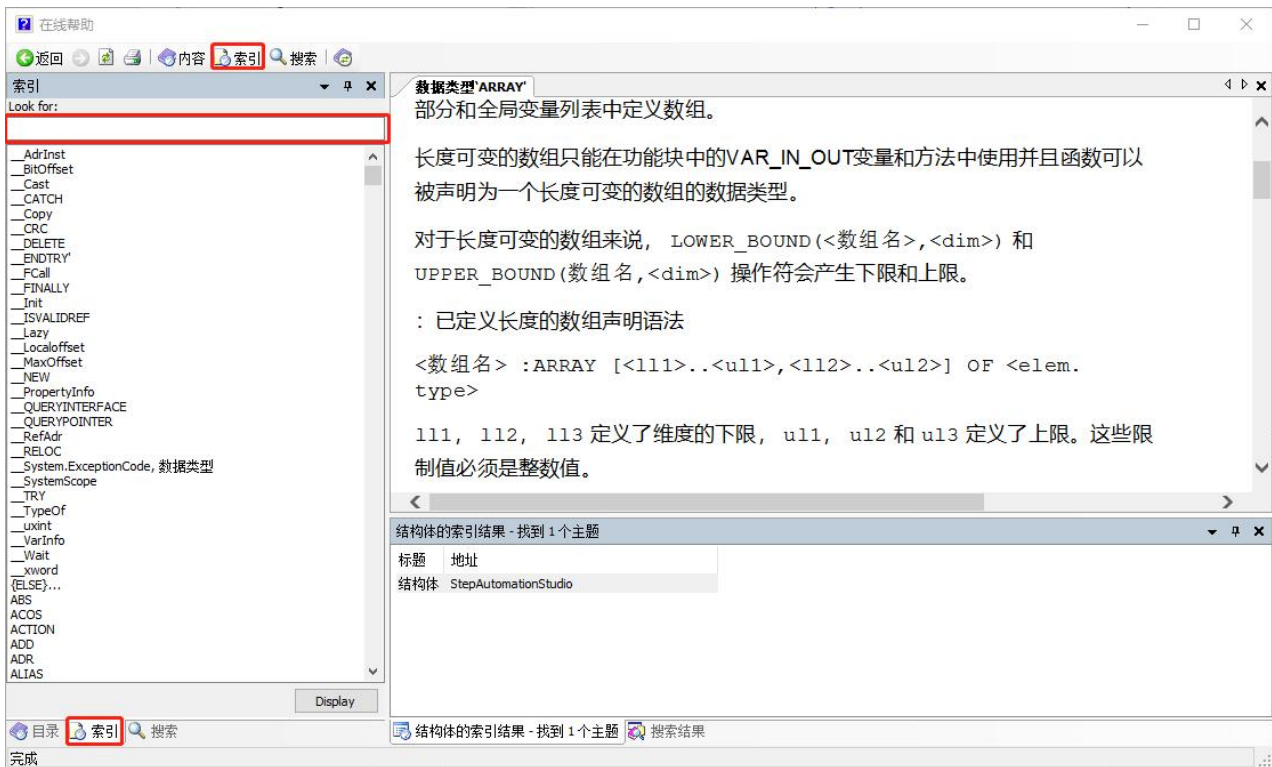
See help documentation.

- 1, Select Help --> Contents or Help Documentation in the menu bar and click the "Target" item in the lower left corner. Displays the Help Documentation Contents page.



- 2, Select Help --> Index or Help Documentation in the menu bar and click the "Index" item in the lower left corner.

Display the help document index page, double-click the keyword on the left to display the corresponding content, or search for the keyword in the search box.



3. Select Help --> Search or Help Documentation in the menu bar and click the "Search" item in the lower left corner.

Display the help document search page, and search for keywords in the search box to display the corresponding content on the right.



3.2. EtherCAT configuration

STEP AS supports two EtherCAT master stations, one is the default EtherCAT master station, which has more complete functions and is supported by the SC30 controller. One is the autonomous EtherCAT master station, which is optimized for the small controller SC20, with better performance and less resource occupation.

Introduction to EtherCAT Bus

EtherCAT is a high-performance, low-cost, easy-to-apply, and flexible topology industrial Ethernet technology that can be used in industrial field-level ultra-high-speed I/O networks, using a standard Ethernet physical layer, transmission media twisted pair or optical fiber (100Base-TX or 100Base-FX). The EtherCAT system consists of a master station and a slave station. The master station only needs a common network card, and the slave station needs a dedicated slave station control chip, such as: ET1100, ET1200, FPGA, etc. It has the following features:

- (1) One network to the end, the protocol processing goes straight to the I/O layer, and a single system can cover all devices
- (2) No need for any lower sub-bus, On the fly message transmission
- (3) Transmission rate: 100 Mbit/s (fast Ethernet, full duplex mode)
- (4) Distributed node synchronization <1us
- (5) The protocols supported above the link layer are as follows:
 - ▶ CoE (CANOpen over EtherCAT), CANOpen application protocol based on EtherCAT, can be connected to servo drive protocol (Cia 402) or IO protocol (Cia 401)
 - ▶ SoE (SERCOS over EtherCAT), servo drive profile according to IEC 61800-7-204
 - ▶ EoE (Ethernet over EtherCAT)
 - ▶ FoE (File over EtherCAT) File transfer over EtherCAT, supports firmware upgrade.

Both SC system controllers support the above features. STEP EtherCAT bus servo supports the above CoE protocol.

For SC series controllers, STEP AS has already configured the hardware of EtherCAT master by default.

For applications based on the EtherCAT CoE protocol, it is necessary to configure "process data PDO" and "service data SDO". The former PDO is arranged to be sent and received periodically, and the latter SDO is only communicated when necessary (usually used to transmit configuration parameters) .

3.2.1. Autonomous EtherCAT Master Configuration

- (1) Master station configuration



parameter	Parameter Description
sync offset	This value allows the offset of the synchronization interrupt to be corrected within the PLC cycle of the EtherCAT slave. Usually the task cycle of the plc starts 20% earlier than the slave synchronization interrupt. This means that the plc task lags 80% of the cycle time and no data is lost.
Automatically restart the slave	When this option is activated, the master will try to restart the slave immediately after the communication ends.
Master mode (first DC Slave as reference time)	Take the first DCThe system time of the slave is used as the reference time.
Slave mode (software time)	Take A in the controllerRMThe system time is used as the reference time.
Slave Mode 2(FPGA time)	Take F in the controllerPGAThe time is used as the reference time for the synchronization of the bus axis and the pulse axis.

(2) Bus cycle task

Typically, for each IEC task, the input data used is read at the beginning of each task (1), and the written output data is transferred to the I/O driver (3) at the end of the task. The implementation in the I/O driver is decisive for the further transfer of I/O data. Therefore, the implementation is responsible for the time frame and specific time that the actual transmission takes place on the corresponding bus system.

The PLC's bus cycle task can be defined globally for all fieldbuses in the PLC settings. However, for some fieldbuses, you can change this setting independently of the global settings. The task with the shortest cycle time is used as the bus cycle task (setting: not specified in PLC settings). In this task, messages are usually transmitted on the bus.

Other tasks just copy the I/O data in the internal buffer, which is only exchanged with the physical hardware in the bus cycle task.



(3) Status

Provides status information (eg 'start' 'stop') and device-specific diagnostic information on the network card and internal bus system used.



3.2.2. Default EtherCAT Master Configuration

(1) General configuration

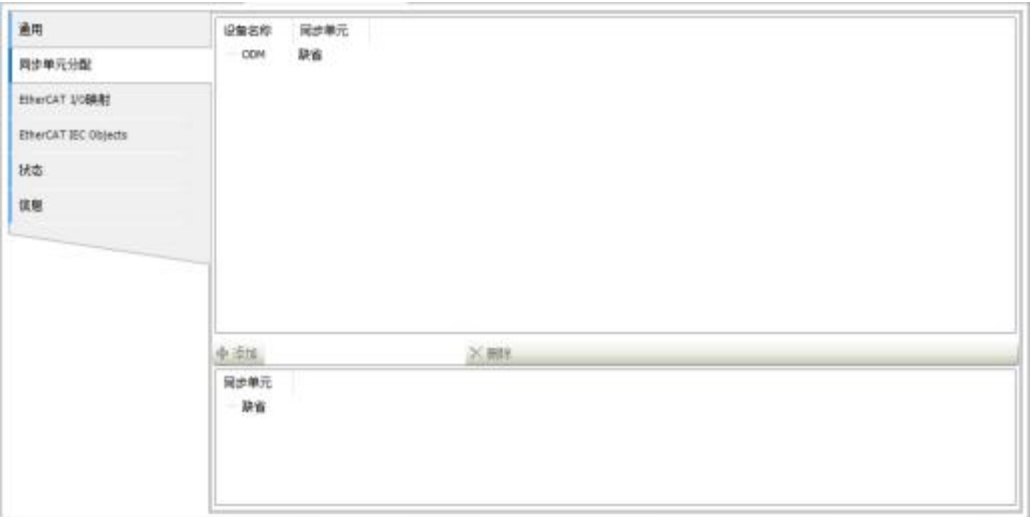
Figure 3 – EtherCATMaster Device Configuration

Classification	parameter	Parameter Description
	Automatic configuration of master/slave	If this option is activated, the main configuration of the master and slave will be done automatically according to the description in the device description file. In this case, the FMMU/SYNC setting dialog will not be displayed.
EtherCAT NICs set up	Destination address (MAC)	The MAC address of the EtherCAT network member that should accept the message. If the "Broadcast" option is checked, the address does not have to be specified.
	Source address (MAC)	The MAC address of the PLC, the name of the network card, i.e.: PLC (target system): Select one of the following options: 1. Select network by MAC 2. Select network by name
	network name	EtherCAT NICs are selected by their specific network names, not MAC addresses.
	Select network by MAC	Select which network card is used as the EtherCAT network card by the MAC address of the network card. for SC30Controller, only 1 can be selected2-34-56-78-9A-BC.
	Select network by name	Select which network card to use as EtherCAT network card by network name, for SC30Controller, only eth can be selected0.
	Enable redundancy	SC20, SC30The controller does not support redundancy.

Classification	parameter	Parameter Description
distributed clock	main cycle time	Refers to the time period after which new data packets can be transmitted. If the 'Distributed clock' function is activated, the master cycle time will be transferred to the slave clock. This enables precise synchronization of data exchange, which is especially important when synchronizing actions are required in distributed processes (eg multiple servo axes performing simultaneous linkage tasks). Therefore, a very accurate clock base with signal jitter less than 1 microsecond can be obtained within the network range.
	sync offset	This value allows the offset of the synchronization interrupt to be corrected within the PLC cycle of the EtherCAT slave. Normally, the task cycle of the PLC starts 20% later than the slave synchronization interrupt. This means that the PLC task lags 80% of the cycle time and no data is lost.
	Sync window offset	Activating this option allows monitoring of the synchronization status of the slaves.
	Sync window	The time to monitor the synchronization window. If all slave synchronizations fall within this time window, the variable xSyncInWindow (IODrvEtherCAT) is set to TRUE, otherwise FALSE.
Options	Use LWR instead of LWR/LRD	Activating this option will use combined read/write commands (LWR) instead of separate read (LRD) and separate write (LWR) commands.
	Enable messages for each task	When this option is activated, the read and write commands for processing input and output information will be completed by different tasks.
	Automatically restart the slave	When this option is activated, the master will try to restart the slave immediately after the communication ends.

(2) Synchronization unit assignment

This tab shows all the slaves plugged under a specific master and assigned to the synchronization unit.



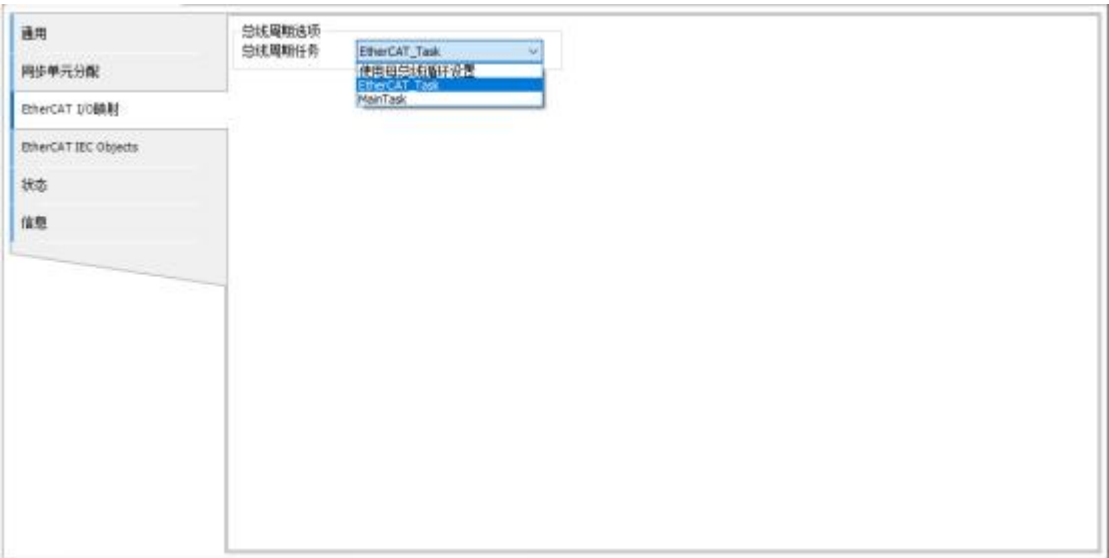
By default, the first slave station is used as the reference clock of the current network segment, which has strong real-time performance; the master station clock can also be used as the reference clock, and the DCSync of the EtherCAT master station device object can be used as the reference clock. ToMasterproperty is set to TRUE. .

(3) Bus cycle task

Typically, for each IEC task, the input data used is read at the beginning of each task, and the written output data is transferred to the I/O driver at the end of the task. When Softmotion EtherCAT is used, the reading and writing of messages will be performed at the beginning of the task.

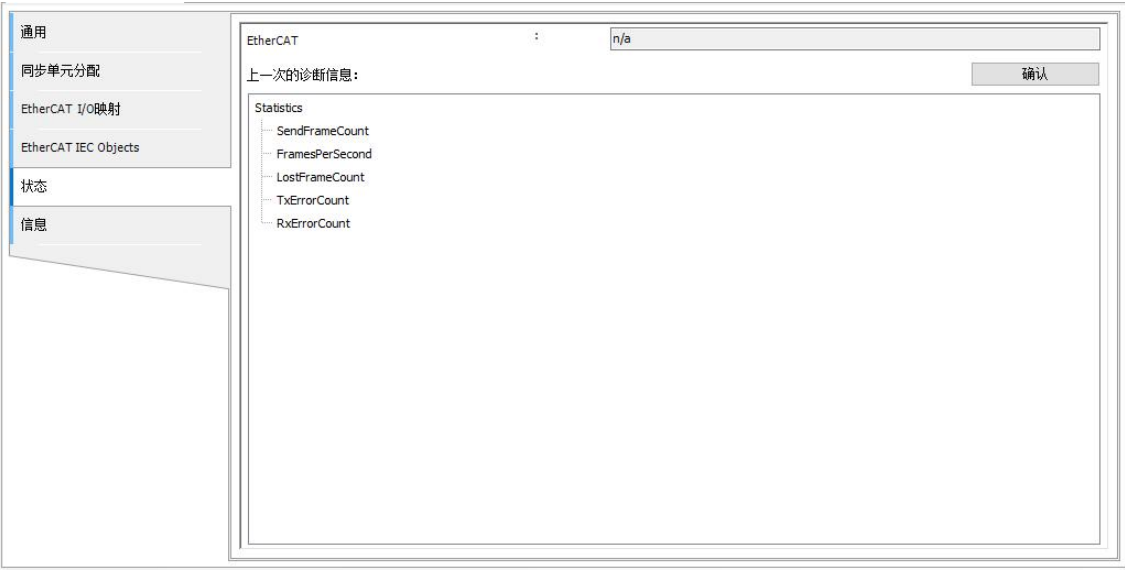
The PLC's bus cycle tasks can be defined globally for all fieldbuses in the PLC settings. This setting can also be changed independently of the global settings. Usually the task with the shortest cycle time is used as the EtherCAT bus cycle task.

Other tasks just copy the I/O data in the internal buffer, which is only exchanged with the physical hardware in the bus cycle task.



(4) Status

Provides status information (eg 'start' 'stop') and device-specific diagnostic information on the network card and internal bus system used.



3.2.3. Autonomous EtherCAT slave configuration

The basic settings of the EtherCAT slave are configured in this option. Device description files are preset to basic settings.

(1) Common settings





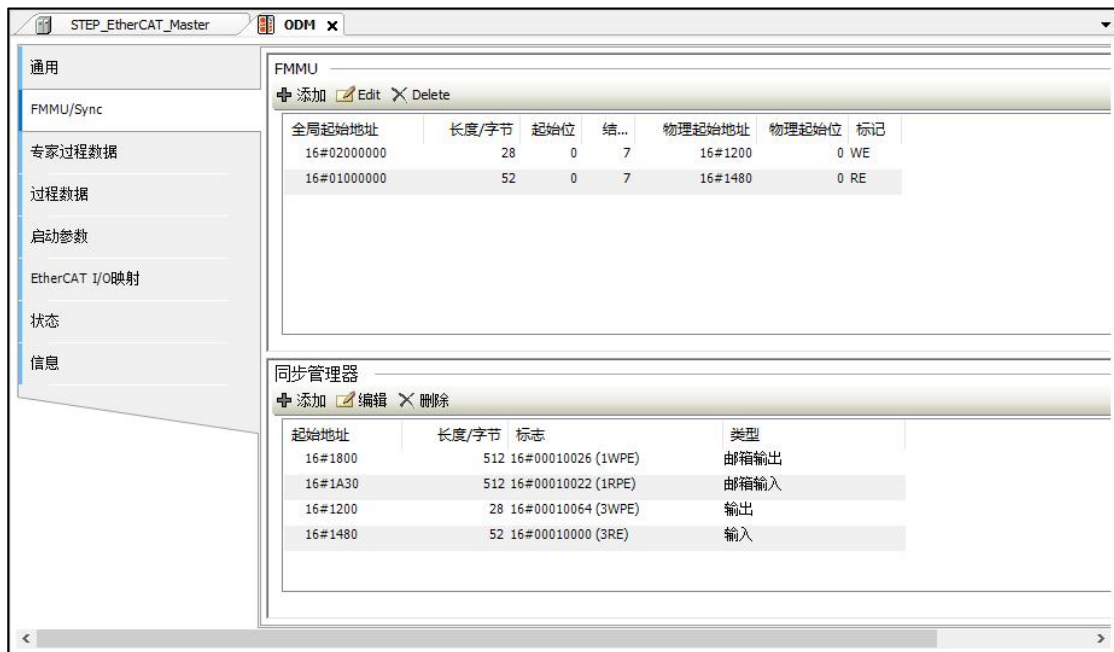
The parameters are explained as follows

Classification	parameter	Parameter Description
address	auto increment address	Auto-incrementing address (16 bits), determined by the position of the slave in the network. This address is only used at startup, when the master is assigning the EtherCAT address to the slave. For this purpose, when the first message passes through the slave, each slave that passes the message increases its auto-increment address by 1. The slave with address 0 finally receives the data. Possible input values are '-8'.
	EtherCAT address	The final address of the slave, assigned by the master at startup. This address is independent of the actual location in the network.
additional	Start expert setup	If this option is activated, additional expert settings for startup checks and timeouts will be effective and the Advanced Process Data dialog will be activated. Note, however, that for standard applications the auto-configuration mode, which is active by default on the master, is sufficient, so it is not necessary to use the 'advanced settings'.
	optional	If a slave device is defined as 'optional', no error message will be created in case the device does not exist in the bus system. To activate this option, a station address must be stored in the slave device, so the 'station alias' address must be defined and written in the EEPROM. And this option is only valid if the 'Auto-configure master/slave' option in the EtherCAT master settings is activated and if the EtherCAT slave supports this function.
distributed clock	select DC	The drop-down menu provides all the settings for distributed clocking provided by the device description file.
	Enable	If the "Distributed Clock" function is

Classification	parameter	Parameter Description
		activated, it is displayed in the "Sync Unit Cycle" (μs) The data exchange cycle time of the area will be determined by the master station cycle time. Therefore, the master clock can synchronize the data exchange within the network.
Sync0	Enable Sync0	If this option is activated, the synchronization unit (Beckhoff) of 'SYNC0' is used. A synchronization unit describes a series of synchronously exchanged process data.
	sync unit cycle	If this option is activated, the master cycle time multiplied by the selected factor will be used as the slave synchronization cycle time. "Cycle Time(μs)" field displays the currently set cycle time.
	User defined	If this option is activated, the desired time in microseconds can be entered in the "Cycle Time (μs)" field.
Sync1	Enable Sync1	If this option is activated, the synchronization unit (Beckhoff) of 'Sync 1' is used. A synchronization unit describes a series of synchronously exchanged process data.
	sync unit cycle	If this option is activated, the master cycle time multiplied by the selected factor will be used as the slave synchronization cycle time. "Cycle Time(μs)" field displays the currently set cycle time.
	Custom	If this option is activated, the desired time in microseconds can be entered in the "Cycle Time (μs)" field.
start check		By default, when the system starts, the vendor ID and product ID are automatically checked against the current configuration settings. If a mismatch is detected, the bus stops and no further action is taken. This setting is to avoid downloading the wrong configuration. This option can be deselected here to turn off checking.
time out	SDO access	When the system starts, the SDO list is sent.
	I -> P	Transition from 'initialization' to 'pre-operational' mode.
	P -> S / S -> O	Transition from 'Pre-Operation' to 'Safe Operation' mode or from 'Safe Operation' to 'Operation' mode.
DC cycle unit control		Select the option defined for the distributed clock function that should be assigned to the local microprocessor. The control functions are already done in register 0x980 of the EtherCAT slave: possible settings: cycle unit, latch unit 0, latch unit 1.
watchdog	set multiplier	The watchdog PDI and SM get their cycles from the local terminal and are accepted by the watchdog.
	Set PDI watchdog	The watchdog is triggered if the PDI (Process Data Interface) communication time with the EtherCAT slave exceeds the set and activated PDI watchdog time.

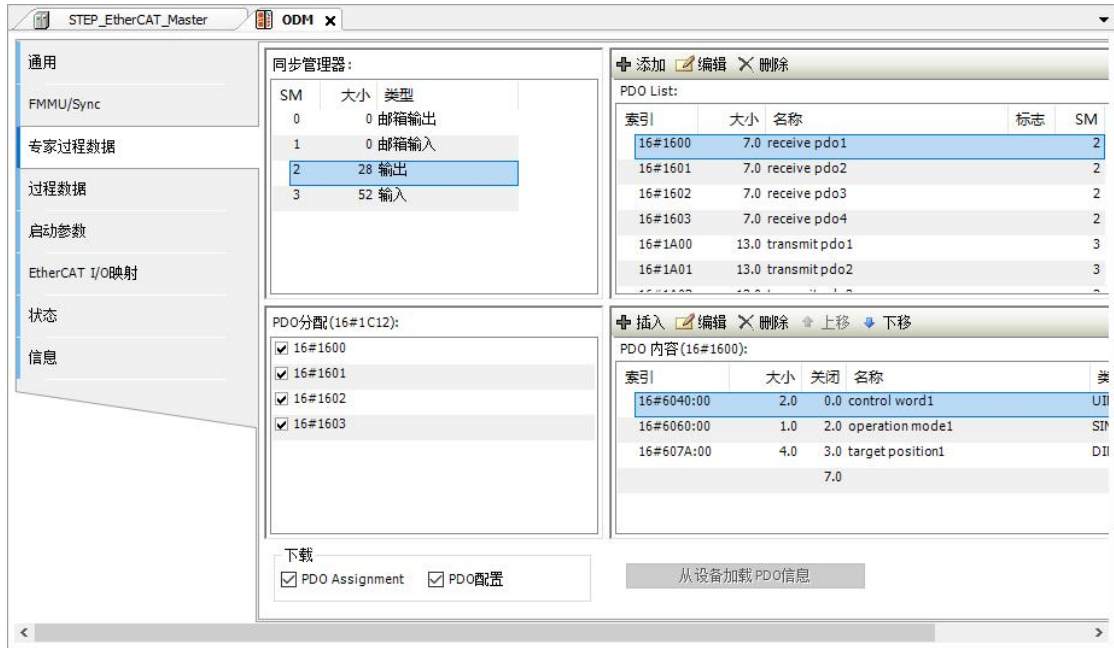
Classification	parameter	Parameter Description
	Set SM watchdog	If the cyclic EtherCAT process data communication is longer than the set and active SM (Synchronous Management) watchdog time, the watchdog will be triggered.
station alias	activation	If the setting "Optional" is not activated, this setting is only valid if it is explicitly supported by the slave device description file. It allows direct assignment of alias addresses for slave addresses independent of their physical location on the bus. This checkbox is disabled if the option 'optional' is activated.
	writeEEPROM	This command is only visible in online mode. It allows to write the defined address to the slave's EEPROM. If the slave does not support it, this command has no effect and the slave does not work as 'optional slave'.
	real address	This column is only visible in online mode and displays the actual address of the slave. It can be used to check 'writeEEPROM' whether the command was successful.

(2) FMMU/Sync



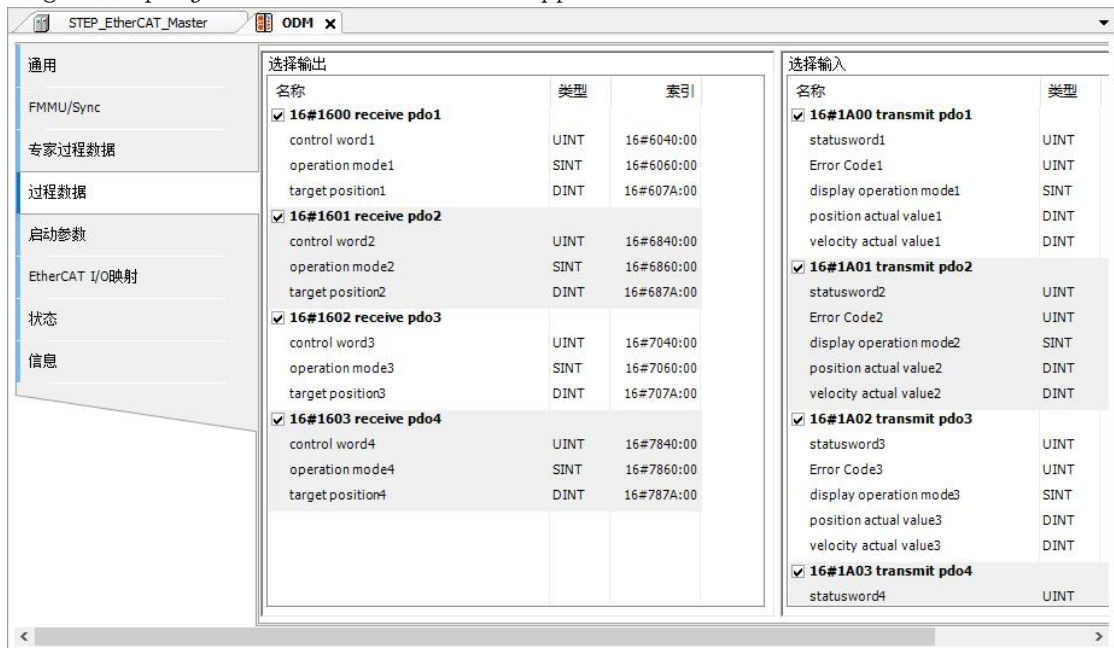
(3) Expert process data

Only used in the EtherCAT Slave Configuration Editor, and when the Slave Options Expert Setup Mode is activated. In addition, PDO configuration can also be performed here.



(4) Process data

Displays process data for the inputs and outputs of slaves, each of which is defined by its name, type and index in the device description file. The selected inputs (readable) and outputs (writable) in the device will be available as PLC outputs and inputs in the "I/O Mapping" dialog. PLC project variables can be mapped.

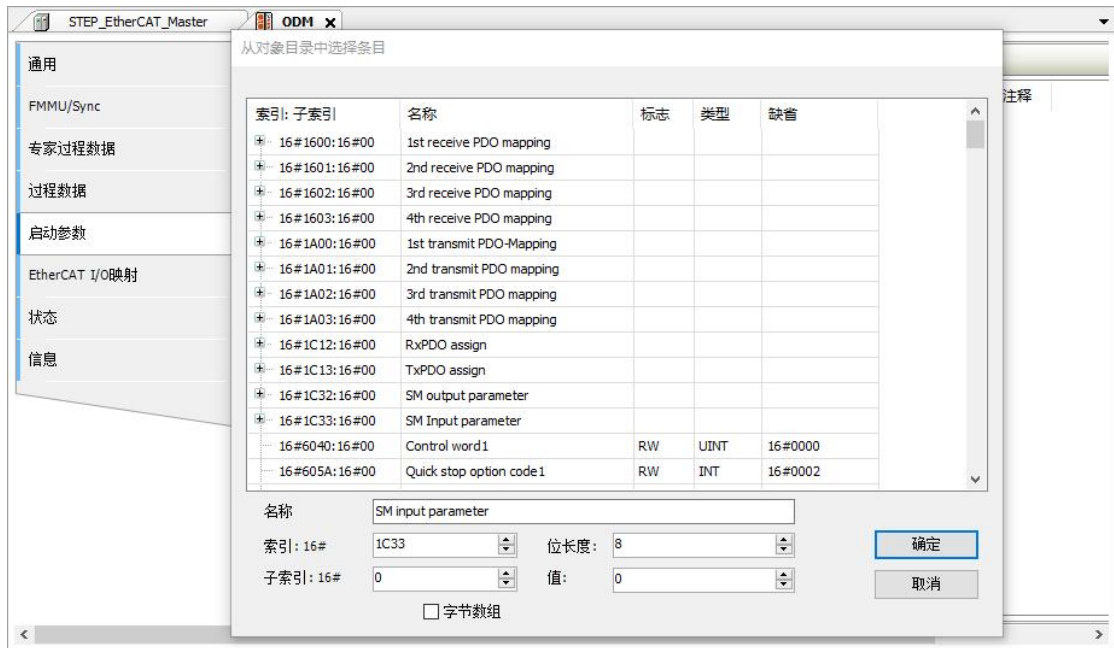


In order to modify the current selection, the mouse must be clicked on the check box in front of the currently selected data to cancel the selection, after which other items can be selected

(5) Startup parameters

Here it is possible to define device parameters for special slaves, which can be transferred by SDOs (Service Data Objects) or IDNs at system startup. This dialog only appears if the

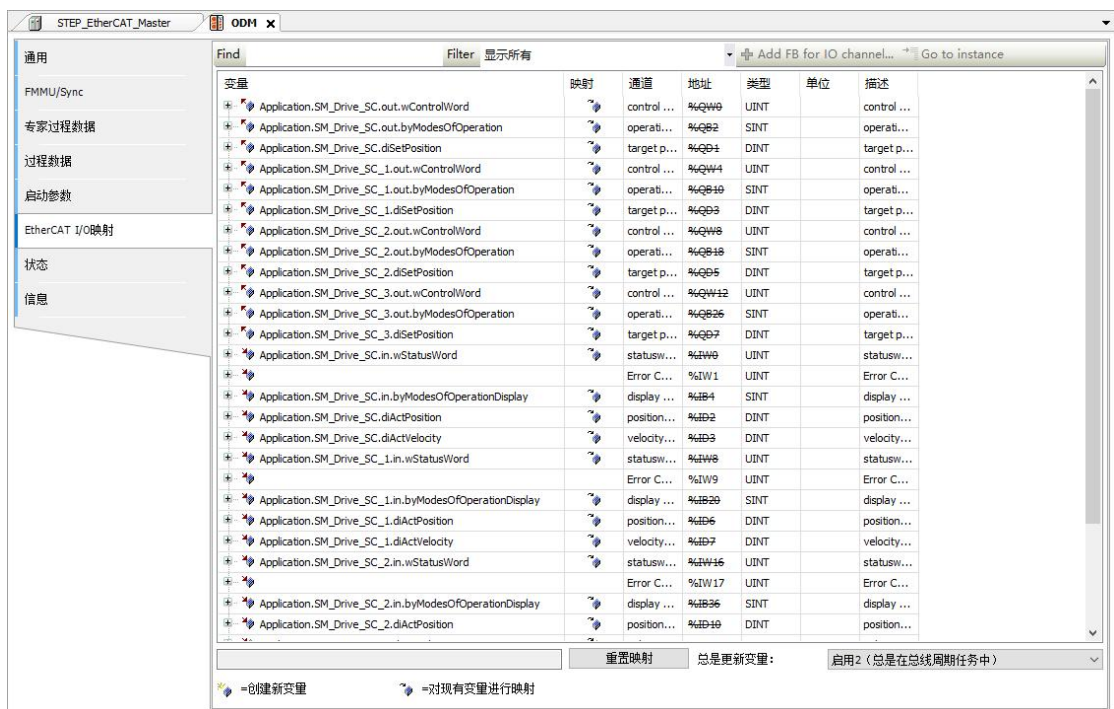
device supports 'CAN over EtherCAT' or 'Servodrive over EtherCAT.'. The object dictionary containing the necessary data is provided by the EtherCAT XML description file, or by the EDS file referenced by the EtherCAT XML description file.



(6) IO mapping

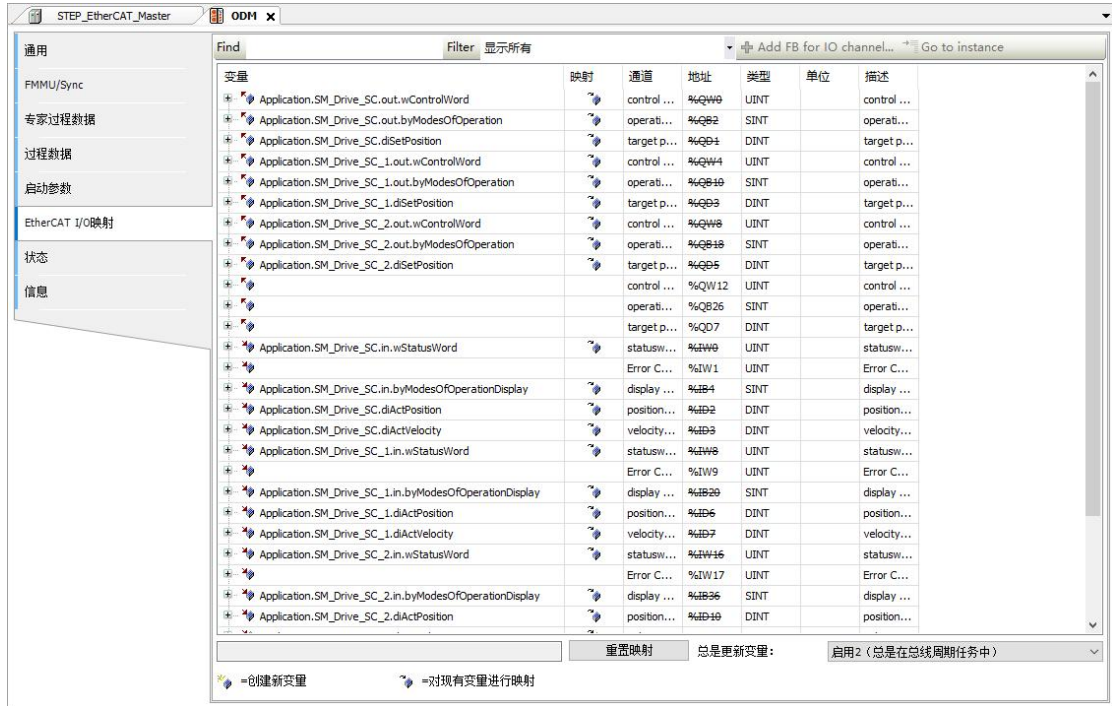
For each EtherCAT slave, an implicit instance of type "ETCSlave" is established as soon as the device is inserted into the device list. The instance name is exactly the same as the device name used in the device list. Valid information for this instance is also displayed in the I/O Mapping Dialog dialog box. The application can use the slave instance to obtain, switch and check the slave status at runtime.

Added slaves are automatically mapped to variables.



IO mapping provides the ability to specify engineering variables for EtherCAT input or output. Thus the PLC connected to the EtherCAT slave can be controlled by the application

When adding or deleting 402 axes, the variable mapping will be updated automatically.



(7) Status

Provides status information (eg 'start' 'stop') and diagnostic information for a specific device.

(8) Online CoE

This dialog is only available from the EtherCAT Slave Configuration Editor tab if the expert settings for the slave are enabled and the application is logged into the device. It shows the status information of slaves and function blocks transferring files to slaves using the EtherCAT bus.

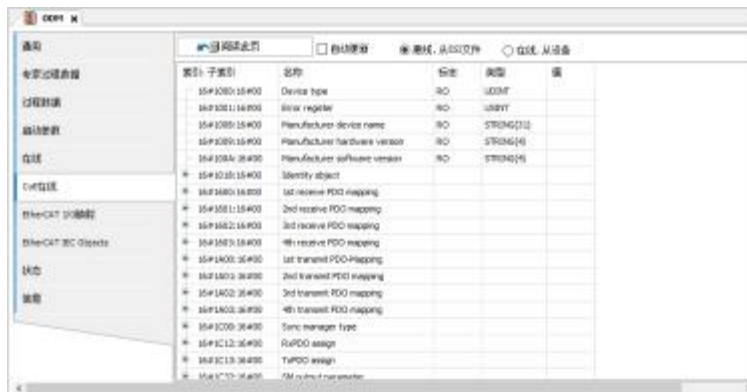
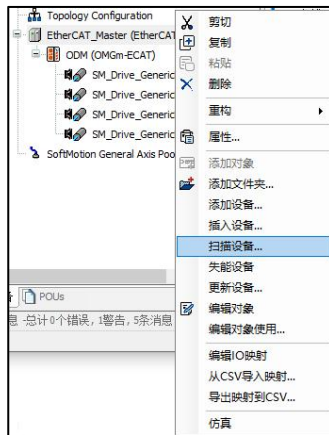


Figure CoE

3.2.4. scan device

Devices can be scanned using the "Scan Devices" in the context menu of the navigation bar.



Scan Devices, only available if the application is logged in, initiates a scan of the hardware environment currently connected to the PLC. This means detecting and viewing the hardware configuration in one dialog and enabling the user to map this configuration directly into the device tree in the project.

In each case, a scan is automatically established before the connection to the PLC, and is automatically closed after the scan is complete. Therefore the gateway connection must be properly configured and the PLC must be running before the scan can take place. If the scan requires library functionality, log in at least before the first scan in order to get the library download.

If an EtherCAT master module has been added, scanning for devices will result in a list of all valid EtherCAT slaves. As shown in the figure below, the device can be copied to the project.



Figure scanning equipment

3.2.5. EtherCAT common faults

Common faults of EtherCAT master:

1. Fieldbus synchronization lost SMC_DI_FIELDBUS_LOST_SYNCRONICITY
 - a) The EtherCAT master distribution clock offset setting is not reasonable enough.

Can be set to 40%~50%.

b) The controller and servo drive use common network cables. It is recommended to use Category 5e twisted pair shielded cable.

2. Stop SMC_FB_WASNT_CALLED_DURING_MOTION during axis motion

a) The task attached to the EtherCAT master and the task attached to the program that controls the axis motion are not the same task

b) The function block that controls the axis movement is not called periodically in the task

3. After the program starts, the EtherCAT master device displays a red triangle and cannot enter the OPERATIONAL state

a) The name of the network card selected by the EtherCAT master is incorrect. For the SC30 controller, only eth0 can be selected as the EtherCAT network card

b) The network cable is inserted wrongly or there is no network cable inserted

4. There is an error in configuring the servo or remote IO module during the startup of the EtherCAT master

a) SDO configuration PDO process error, please refer to 3.4.4 CANopen communication failure

b) Other error reasons can be analyzed according to STEP ASlog log

3.3. Modbus serial port configuration

3.3.1. Add Modbus device

STEP AS supports independent modbus communication, the controller can be used as a master station and a slave station, and general slave stations, STEP frequency converters and controller slave stations can be added in the new project wizard.



The number of Modbus buses supported by different controllers is different, and the number of supported buses is displayed on the controller configuration page.



The controller uses different buses when it acts as a master station and a slave station, so when adding a Modbus device, the program will judge the number of buses currently occupied. If the number of buses supported by the controller exceeds the number of buses supported by the controller, a warning that the serial port is occupied will be given.



The project structure after adding one general modbus slave station and two STEP inverters is shown below.



Modbus serial communication supports the standard ModbusRTU protocol, can be configured as the master station, and supports 9600, 115200 and other baud rates.

The range of variables that the master can access is defined as follows:

- (1) All bit variable operations (01 02 05 0f) can read and write %MX0.0-%MX8191.7 a total of 65536 bit variables;
- (2) All register variable operations (03 04 06 10) can read and write %MW0-%MW65535, a total of 65536 register variables

3.3.2. Modbus master configuration

Double-click the master station to enter the Modbus parameter configuration interface, including the settings of communication port, baud rate, parity bit, data bit, stop bit, transmission mode, and frame interval.

Modbus master configuration parameters:

configuration item	Features
The port number	The master physical connection selects the port.
baud rate	rate of communication.
parity	The verification method of communication frames.
data bits	The actual data bits contained in the communication frame.
stop bit	Identifies the last bit of a single packet when communicating.
transfer mode	RTU.

frame interval	The time interval that the master station waits between receiving the last response data frame and the next request data frame.
----------------	---

3.3.3. Modbus slave configuration

Double-click the slave station to enter the communication parameter configuration of the slave station, including the slave station address, timeout time, and slave station enable variable configuration.



Modbus slave configuration parameters:

configuration item	Features
Slave station number	Identifies the slave station number, range 1~247.
overtime time	After the master station sends, if the time exceeds this time, the master station reports the receiving timeout.
Slave Enable Variable	Program to enable the slave station, start sending communication frames to the slave station, 0Nefficient.

Configure the slave channel, including adding, deleting, and editing operations. Each channel represents an independent Modbus request.



After clicking the "Add" button, the channel setting page will pop up. After the page setting is completed, click "OK" to add the corresponding channel. Click the "Cancel" button to terminate the channel establishment.

从站通道设置

通道
名称: Channel 05

访问类型: 读保持寄存器(功能码03)

触发器: 周期 周期(ms): 1

重发次数: 1

内容:

读寄存器
偏移: 0x0000

长度(WORD): 1

错误处理: 保持上次值

写寄存器
偏移: 0x0000

长度(WORD): 1

确定 取消

configuration item	illustrate	
access type	Read coil status (function code 01) Read input status (function code 02) Read holding register (function code 03) Read input register (function code 04) Write a single coil (function code 05) Write a single register (function code 06) Write multiple coils (function code 15) Write multiple registers (function code 16)	
trigger	Loop Execution: Periodically Triggered Requests	Cycle Time: Set the time to execute again
	Level Trigger: Trigger when programming is changed	trigger variable (SM): set trigger SMelement, After the trigger is successful, the element is automatically reset
number of retransmissions	This time a communication failure occurs and the frame returned by the slave station is not obtained, and the retransmission is performed according to the number of retransmissions.	
Notes	A short text area that can describe the data	
read register		
starting address	Start position of the read register	
length	Number of registers read	
error handling	keep last value: keep the data at the last valid value set to 0: zeros all values	
write register		
starting address	Write register start position	
length	write register length	

"The valid range of the "length" parameter depends on the following function codes:

function code	type access	Number of registers
01	Read coil status	1~2000
02	read input status	1~2000

03	read holding register	1~125
04	read input register	1~125
05	write a single coil	1
06	write a single register	1
15	write multiple coils	1~1968
16	write multiple registers	1~123

The slave channel interface after adding the channel is as follows:

名称	序号	访问类型	触发变量	读偏移	长度	重发次数	内容
Channel 01	1	读保持寄存器(功能码03)		0x0010	1	1	读寄存器
Channel 02	2	读输入状态(功能码02)		0x0000	1	1	读输入状态

☐ 使用十进制

添加... 删除 编辑...

- Click the "Edit" button to edit the selected channel. Click the "OK" button to update the channel settings or click the "Cancel" button to keep the original settings.
- Click the "Delete" button to delete the selected channel.
- When creating STEP inverter, the system will automatically add several commonly used channels.

Modbus从站配置

从站通道配置

Internal配置

Internal I/O映射

Internal IEC Objects

状态

信息

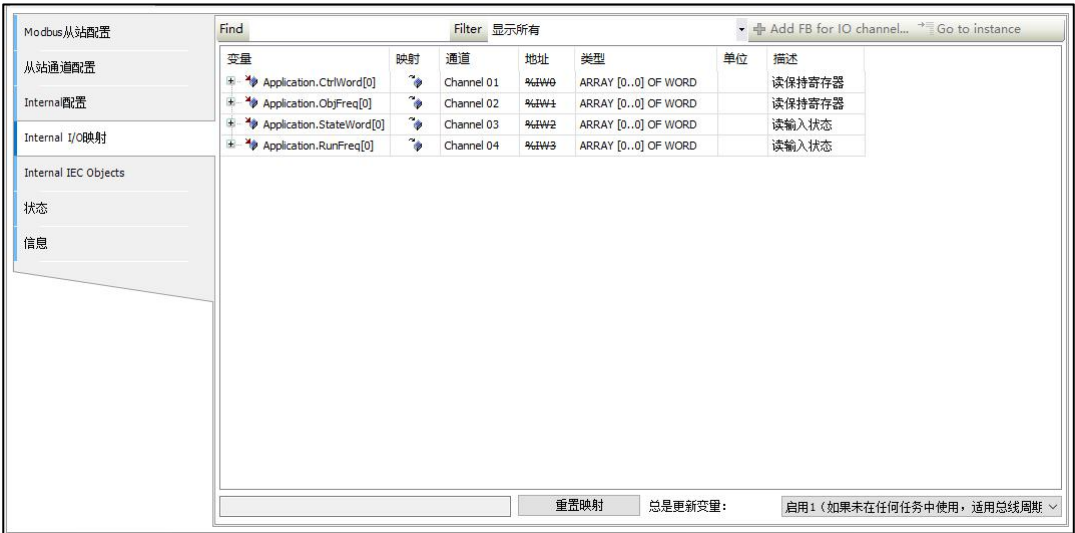
名称	读写类型	读偏移	长度	写偏移	长度	重发次	注释
Channel 01	读输入寄存器(功能码04)	0x0000	1			2	
Channel 02	读保持寄存器(功能码03)	0x0030	4			1	
Channel 03	读输入状态(功能码02)	0x0150	7			1	

☐ 使用十进制地址

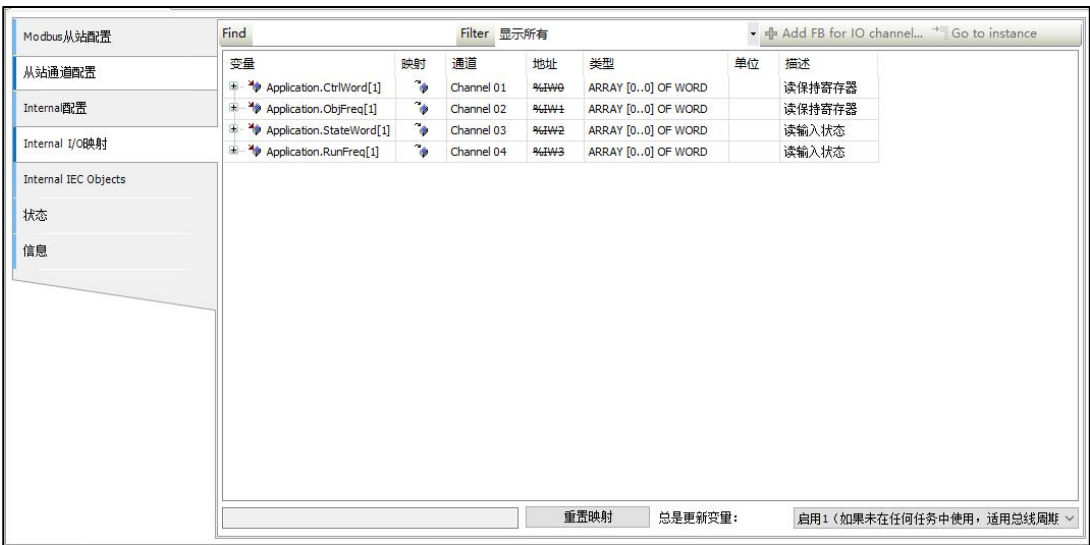
添加... 删除 编辑...

Graph channel configuration item

In addition, when using the project wizard to create a new project, the system will automatically map the added channel to the corresponding global variable in the ModbusGVL file. The mapped variables are consistent with the order added by the inverter. For example, CtrlWord[0] corresponds to the slave _STEP_inverter Map variable; CtrlWord[1] corresponds to the map variable of slave _STEP_inverter_1.



Graph mapping variables



Graph mapping variables

3.3.4. Modbus common faults

- Major failure of Modbus master connecting to Modbus slave:
- (1) The configuration of the Modbus master station and the Modbus slave station are inconsistent, resulting in the failure to establish the communication between the master station and the slave station
 - (2) The Modbus master station accesses the illegal address of the Modbus slave station and returns an error response.

(3) The Modbus master station operates the Modbus slave station to write the register, but the Modbus slave station only supports read and not write operations, and the Modbus master station will receive the error response returned by the Modbus slave station.

❶error response frame

Error response: slave address + (command code + 0x80) + error code + CRC check

This error frame applies to all operation command frames.

serial number	Data (byte) meaning	number of bytes	Number of registers
1	slave address	1 byte	value 1~247
2	Command code+0x80	1 byte	wrong command code
3	error code	1 byte	1~4

3.3.5. Modbus TCP configuration

Visual configuration is not currently supported.

The ModbusTcpSlave function block provided by our company can be used for communication.

3.4. CANopen configuration

The configuration of CANopen mainly includes the configuration of CANbus, CANopen master and CANopen slave. CANbus is added by default when adding a CANopen master device using the wizard and configuration. For details on how to add devices, refer to the chapter on device configuration.

SC30 controller supports CANopen communication protocol standard DS301.

The relationship between CANopen transmission rate and transmission distance is as follows:

baud rate (bps)	Maximum bus length (m)
1M	40
500K	110
250K	240
125K	500
100K	1300
50K	3300
20K	6600
10K	13000

3.4.1. CANbus configuration

CANbus is the top configuration of every CAN bus configuration in the device tree. A CANopen master can only be inserted under a CAN bus node and then a slave, ie a CANopen remote device, can be added.

Open the CANbus configuration page, as follows:



Figure CANbus configuration page

parameter	Parameter Description
The internet	Represents CAN peripheral index, default CAN0
Baud rate (kbits/s)	Communication baud rate

Notice:

To prevent the loss of CAN frames, make sure that the cycle time is set correctly by the following variables: the baud rate currently in use, the number of bus frames, and the heartbeat time setting, node guarding or synchronization. These times should be integer multiples of the cycle time!

3.4.2. CANopen master configuration



Classification	parameter	Parameter Description
Overview	Node ID	The node ID provides the CANopen manager with a one-to-one correspondence to the group number of modules, and the ID value is between 1 and 127. ID must be a decimal number.
	Automatic start of CANopen manager	If this option is activated, CANopen management starts automatically when all slaves are ready (to

Classification	parameter	Parameter Description
		OPERATIONAL mode). If the option is not activated, management must be initiated through the application, using the CiA405 NMT function block to handle this.
	Optional slave polling	If the slave does not answer while the script is queued then it will send every minute until it responds successfully. Permanent polling of slaves will result in an increase in the bus cycle time which may affect the application (especially for motion applications). To avoid this the behavior can be stopped. If a polling is stopped the slave will only be activated after sending a start telegram.
	start slave	This option is used in CANopen management to start slaves. If stopped, the slave needs to be restarted via the CiA405 NMT function block in the application.
	NMT start all (if possible)	If the option "Start slaves" is activated the CANopen management will start all slaves with the command "NMT Start All". The "NMT Start All" command does not start when the slave is not ready. In this case the CANopen management will start each slave individually. "NMT Start All" is only guaranteed to start if there are no optional slaves in the project.
	NMT misbehavior	restart the slave: If a guard event occurs all slaves will be automatically restarted by the stack (NMT reset + SDO configuration + NMT start) stop slave: If a guard event occurs all slaves will stop. The slave must be restarted through the application by using the CiA405 NMT function block.
heartbeat	Enable heartbeat generation	If this line selection is activated, the master will continuously send heartbeats according to the internally defined "heartbeat time". If a new slave heartbeat function is added, their heartbeat action will be automatically activated and configured, that is, the node-IDIt is automatically set in the management configuration, and the heartbeat interval is automatically multiplied by a factor1,2. ifCANopenThe heartbeat creation in the management is not activated, then the node protection will be activated in the slave(with life time factor10and a100msguard time). NoticeCANopen (Slaves)The device is happy to create a configuration as a heartbeat.
	Node ID	A unique identifier (1 - 127) generated by heartbeats on the bus.
	Generation time (ms)	Defines the internal heartbeat time in milliseconds.
Synchronize	Enable synchronous generation	If this option is enabled (default: disabled), the CANopen manager will send synchronization messages.
	COB-ID(Hex)	The communication object identifier, which identifies the synchronization message. Possible values: [1, 2047].
	Cycle period (μs)	How many microseconds to send sync messages. Possible values: [100, 232-1].
	Window length (μs)	Synchronized PDO's with time window length in microseconds [1,232-1] or 0 If not applicable, synchronized PDO's will be sent concurrently after the synchronization message.
	Start synchronous consumption	If this option is enabled (default: disabled), another device will send a sync message and CANopen management will receive it.

Classification	parameter	Parameter Description
time	Enable time generation	If this option is enabled (default: disabled), CANopenManager will send TIME messages.
	COB-ID (Hex)	The communication object identifier, which identifies the time message. Standard value: [0, 2047], default value is 100.
	Generation time (ms)	The millisecond interval at which timestamp messages are sent. Must be an integer multiple of the task cycle time. Possible values: [0, 65535].

Note: The runtime system must support high resolution time, otherwise an error message will be generated

3.4.3. CANopen slave configuration



Classification	parameter	Parameter Description
Overview	Node ID	The Node ID is used to define a unique CANopen node and is between 1 and 127 according to the module's local setting. ID must be a decimal number.
	SDO channel (1/efficient)	This button opens a dialog where the SDO channel can be defined. Service data types (SDOs) allow access to arbitrary entries in the CANopen object dictionary. An SDO creates a channel peer-to-peer data transfer between two devices (SDO server and client).
	Enable expert settings	Display the "SDO Channel" configuration item.
	optional equipment	If this option is activated then the slave device is optional and not forced to start by the CAN network.
	Uninitialized	If this option is activated (visible in the dialog depending on the target system), the master will activate the node immediately without sending configuration SDOs. (However, the SDO data will be created and saved by the controller).
	reset node	Before downloading the configuration or the slave configuration, the CANopen communication configuration parameters of the slave will be reset to default values.

Node guard	Enable Node Guard	If this option is activated, a message will start with "Protection Time" The interval to be sent to the module (microseconds, default 100). If the module does not follow the given "Protect COB-ID" (communication object definition) to send data, then the CANopen management will "Life Time Factor" Resend or until the module responds. If no module responds, the module will be marked as "unavailable".
	Enable heartbeat generation	If this option is activated, this module will "Heartbeat generation time (ms)" The definition in send heartbeat, which defaults to 10 if there is no other default setting in the device configuration file or if it defaults to 0.
	Heartbeat consumption (1/efficient)	This button opens a dialog to define the variables that the slave needs to protect, which can be defined.
Emergency situations	enable emergency	If this option is activated, the module will send an emergency message through the COB-ID interval until an internal error occurs. This information can be recovered by the functions provided by the CiA405 library (RECV_EMCY_DEF, RECV_EMCY) function library.
	COB-ID	Communication object definition, defining urgent messages.
time	Enable time generation	If this option is enabled (default: disabled), CANopenManager will send TIME messages.
	COB-ID (Hex)	The communication object identifier, which identifies the time message. Standard value: [0, 2047], default value is 100.
	Generation time (ms)	The millisecond interval at which timestamp messages are sent. Must be an integer multiple of the task cycle time. Possible values: [0, 65535].
check on reboot		If this option is activated then information will be received from the CANopen slave firmware and compared with the information in the EDS file. If a configuration does not match then the configuration will stop and the slave will not start.

①Note: If a device with heartbeat function is plugged in, then its heartbeat settings will be automatically set according to the master's configuration

3.4.4. CANopen communication failure

- 1) General troubleshooting steps
 - 1) Check the wiring
 - 2) Check the baud rate
 - 3) Check the matching resistance
 - 4) Other
- 2) Communication fault code

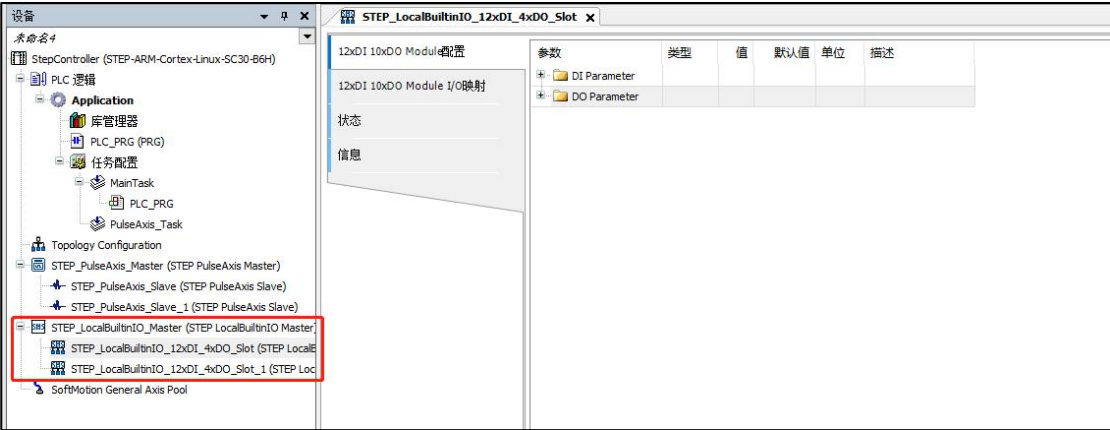
Code ID	Code function description
0503 0000	Trigger bits are not alternately changed

0504 0000	SDO protocol timeout
0504 0001	Illegal or unknown Client/Server command word
0504 0002	Invalid block size (Block Transfer mode only)
0504 0003	Invalid serial number (Block Transfer mode only)
0503 0004	CRC error (Block Transfer mode only)
0503 0005	memory overflow
0601 0000	Object does not support access
0601 0001	Attempt to read write-only object
0601 0002	Attempt to write read-only object
0602 0000	Object does not exist in object dictionary
0604 0041	Object cannot be mapped to PDO
0604 0042	The number and length of mapped objects exceeds the PDO length
0604 0043	General parameter incompatibility
0604 0047	General device internal incompatibility
0606 0000	Object access failed due to hardware error
0606 0010	Data type mismatch, service parameter length mismatch
0606 0012	Data type mismatch, service parameter length is too large
0606 0013	Data type mismatch, service parameter length is too short
0609 0011	subindex does not exist
0609 0030	Out of range of values for parameter (on write access)
0609 0031	The write parameter value is too large
0609 0032	Write parameter value is too small
0609 0036	The maximum value is less than the minimum value
0800 0000	general error
0800 0020	Data cannot be transferred or saved to the app
0800 0021	Data cannot be transferred or saved to the application due to local control
0800 0022	Data cannot be transferred or saved to the app due to the current device state
0800 0023	The object dictionary generates an error dynamically or the object dictionary does not exist (for example, the object dictionary is generated from a file, but an error occurs due to a corrupt file)

3.5. Local built-in IO configuration

3.5.1. Add device

The local built-in IO master and slave devices can be added by using the wizard, topology configuration or right-clicking to add devices. The effect after adding is as follows:



(1) Local built-in IO slave configuration

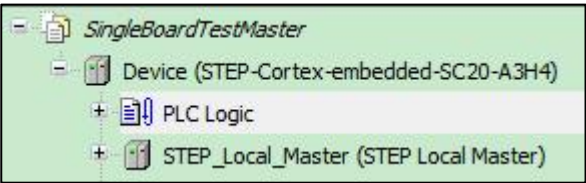
STEP_LocalBuiltInIO_12xDI_4xDO_Slot_SC308_ X						
12xDI 4xDO Module Parameters						
12xDI 4xDO Module I/O Mapping						
Status						
Information						
Parameter	Type	Value	Default Value	Unit	Description	
Fault	BOOL	FALSE	FALSE		Fault State	
DI Parameter						
Level	WORD	4095			Reverse level	
Bit0	BOOL	TRUE	TRUE			
Bit1	BOOL	TRUE	TRUE			
Bit2	BOOL	TRUE	TRUE			
Bit3	BOOL	TRUE	TRUE			
Bit4	BOOL	TRUE	TRUE			
Bit5	BOOL	TRUE	TRUE			
Bit6	BOOL	TRUE	TRUE			
Bit7	BOOL	TRUE	TRUE			
Bit8	BOOL	TRUE	TRUE			
Bit9	BOOL	TRUE	TRUE			
Bit10	BOOL	TRUE	TRUE			
Bit11	BOOL	TRUE	TRUE			
DO Parameter						
Level	USINT	0			Reverse level	
Bit0	BOOL	FALSE	FALSE			
Bit1	BOOL	FALSE	FALSE			
Bit2	BOOL	FALSE	FALSE			
Bit3	BOOL	FALSE	FALSE			

Parameters page

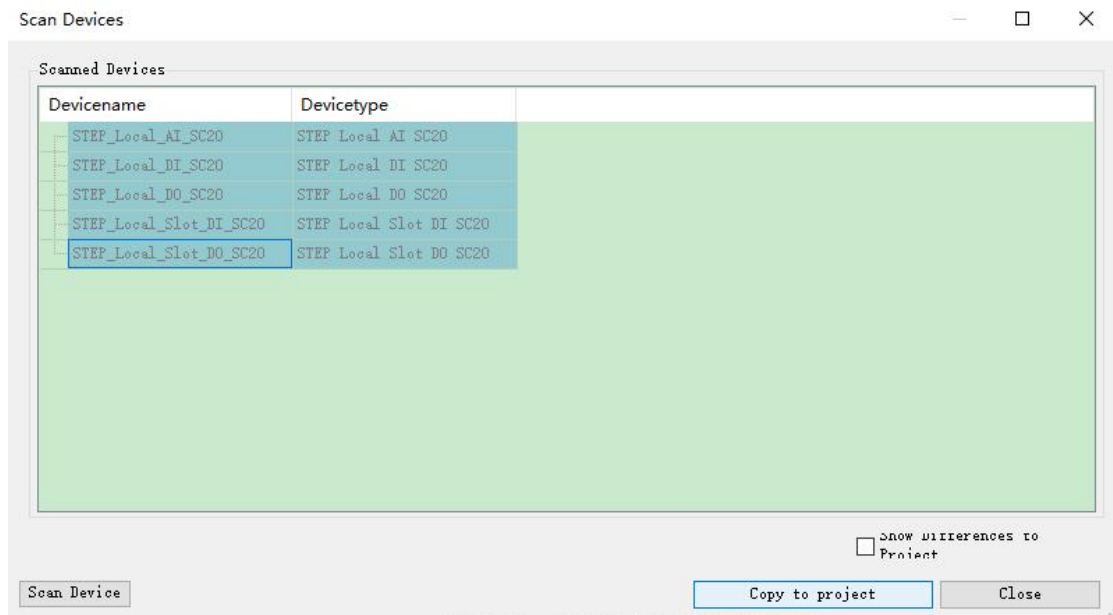
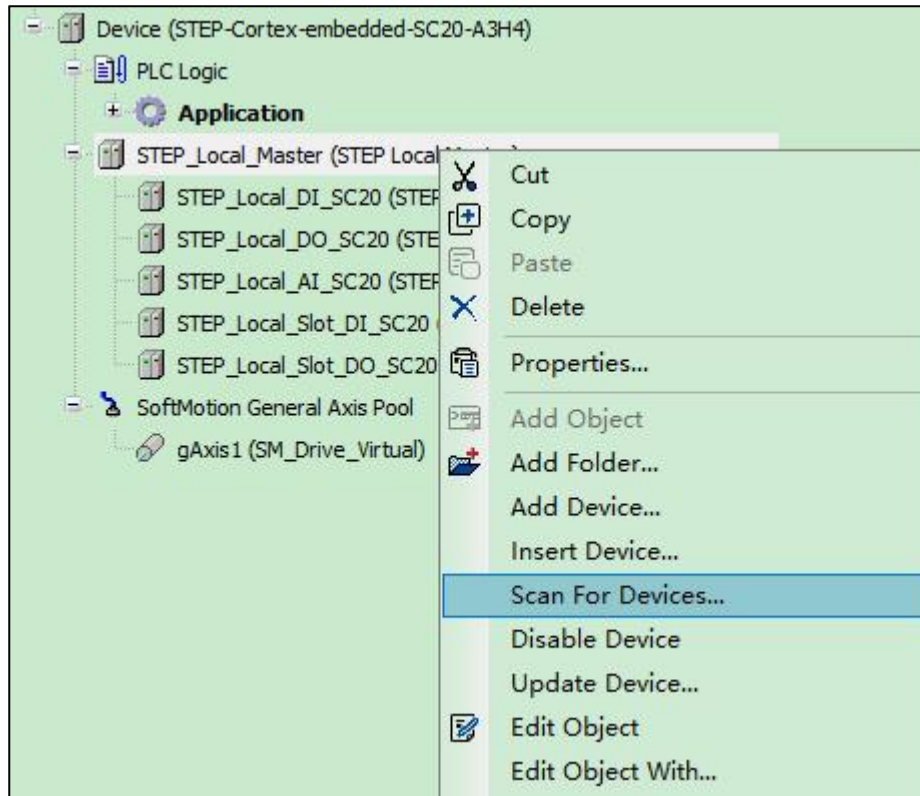
parameter	Parameter Description
Fault	Whether the module is in Fault state
Level	Bit flips, not flipped by default

(2) SC20 local IO master

SC20 local IO control, need to add STEP_Local_Master master station



Once the connection to the controller is established, right-click on the device and click "Scan for Devices..." can scan slave devices



Click "Copy all devices to the project" to add the scanned slave devices to the master station.



Of course, you can also add it manually, double-click STEP_Local_Master to view the error message

STEP Local Master Parameters		Parameter	Type	Value	Default Value
Status		Error string	STRING	"	"

You can assign IO devices to different tasks, such as assigning AI devices to PulseTask. If not assigned, the first task will be used by default. For applications that require fast response, the corresponding equipment can be assigned to tasks with shorter cycle times.

STEP Local AI SC20 Parameters

STEP Local AI SC20 I/O Mapping

Status

Information

Find Filter Show all

Variable	Mapping	Channel
		AI
Application.IO_TEST.wLocalAi[0]		AI1
Application.IO_TEST.wLocalAi[1]		AI2
		AI3

= Create new variable = Map to existing variable

Bus cycle options




Bus cycle task PulseTask

3.5.2. SC20 local IO configuration










(1) 8 digital inputs

Configuration parameters

Bit inversion configuration, not inversion by default, generally do not need to be modified

Parameter	Type	Value	Default Value	Unit	Description
 Periph Type	BYTE	1	1		
 Config Parameter					
 Bit Inversion	BYTE	0	0		位反转,bit0-bit7 bit: 0 不反转 1 反转




I0 mapping

Variable	Mapping	Channel	Address	Type	Unit	Description
		DI	%IB12	BYTE		
		X0	%IX12.0	BOOL		
		X1	%IX12.1	BOOL		
		X2	%IX12.2	BOOL		
		X3	%IX12.3	BOOL		
		X4	%IX12.4	BOOL		
		X5	%IX12.5	BOOL		
		X6	%IX12.6	BOOL		
		X7	%IX12.7	BOOL		






(2) 4 digital outputs

Configuration parameters

Bit inversion configuration, not inversion by default, generally do not need to be modified

Parameter	Type	Value	Default Value	Unit	Description
 Periph Type	BYTE	2	2		
 Config Parameter					
 Bit Inversion	BYTE	0	0		位反转,bit0-bit3 bit:0 不反转 1 反转

I0 mapping

Variable	Mapping	Channel	Address	Type	Unit	Description
		DO	%QB0	BYTE		
		Y0	%QX0.0	BOOL		
		Y1	%QX0.1	BOOL		
		Y2	%QX0.2	BOOL		
		Y3	%QX0.3	BOOL		

(3) 2 analog inputs

2 analog inputs, 1 RTC under-voltage detection, 16-bit resolution.

Configuration parameters

Upper and lower limit calibration value, filter coefficient (using first-order lag filter)

Parameter	Type	Value	Default Value	Unit	Description
Periph Type	BYTE	3	3		
Config Parameter	ARRAY [1..2] OF CfgAI				
Config Parameter[1]					
UpperLimit	WORD	0	0		电压上限，0-65535，用于校准电压最大输入
DownLimit	WORD	0	0		电压下限，0-65535，用于校准电压最小输入
Filter	BYTE	96	0		滤波系数，0-255，系数越大，滤波效果越明显





Calibration method: Set the initial upper and lower limits to 65535 and 0, input the maximum voltage (10V) and the minimum voltage (0V) respectively, and record the corresponding sampled AD value, and then use the sampled AD value as the upper limit and lower limit respectively, you can Complete the calibration; the upper and lower limits are only related to calibration, and the AD value range collected after calibration is still 0-65535. To recalibrate, set the upper and lower limits to 65535 and 0, and then repeat the above steps. Usually no modification is required.

When the upper and lower calibration values given by the user are both 0, the calibration value saved in the controller is used; when the upper calibration value specified by the user is greater than the lower calibration value, the calibration value specified by the user is used; when the upper calibration value specified by the user is less than When the lower limit calibration value is used, the controller uses the default calibration value.

The last channel is the RTC voltage, which can be used for RTC under-voltage detection to remind the user to replace the battery.

①Note: In online mode, parameter value modification can only be achieved through the Write Parameters button. Parameters modified online can be saved after power-off, and do not need to be modified in offline mode.

Write Parameters

Parameter	Type	Current Value	Prepared Value	f...	>...	Unit	Description
 Periph Type	BYTE	16#03		3	3		
 Config Parameter	ARRAY [1..4] OF CfgAI						
 Config Parameter[1]							
 UpperLimit	WORD	16#0000		0	0		电压上限，0-65535，用于校准电压

IO mapping

Variable	Mapping	Channel	Address	Type	Unit	Description
		AI	%IW0			
Application.IO_TEST.wLocalAi[0]		AI1	%IW0	WORD		0-65535
Application.IO_TEST.wLocalAi[1]		AI2	%IW1	WORD		0-65535
		AI3	%IW2	WORD		RTC电压，0-65535

3.6. LocalBus configuration

LocalBus is a bus used to expand local IO, which is fast, stable and scalable. There are currently three localbus slave devices:

serial number	device name	illustrate
1	STEP_LocalBus_16xDI_Module	16digital input module

2	STEP_LocalBus_16xDIO_Module	16digital output module
3	STEP_LocalBus_4xAI_2xAO_Module	4 analog inputs and 2 analog outputs

3.6.1. Add device

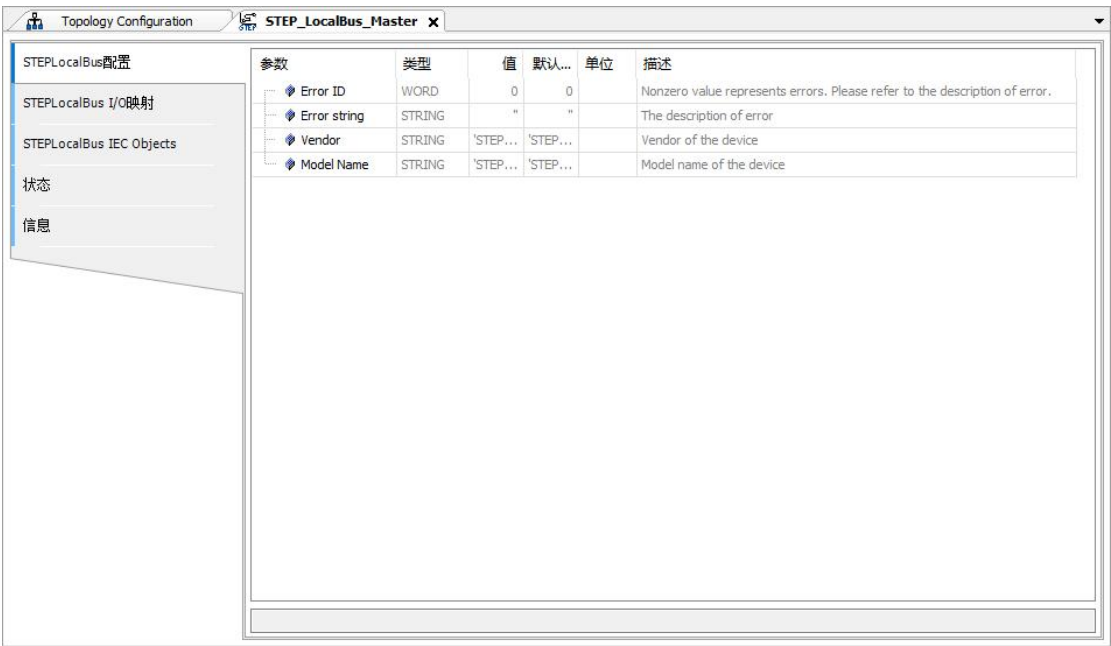
Localbus devices can be configured using wizards or configuration. For detailed configuration, refer to the chapter on device configuration.



Figure localbus device diagram

3.6.2. localbus master configuration

The Localbus master does not need to be configured, and the configuration parameters are read-only.



Users can view specific error information through the ErrorID and Error String parameters:

Error ID	Error description
0	No error, enter the periodic data communication state
1	LocalBusinitialization error
2	Local in STEP ASBusThe expansion module device is not added under the master device
3	No expansion modules have been added to the controller
4	The number of expansion modules configured in STEP AS is inconsistent with the actual number of expansion modules
5	The order of expansion module types configured in STEP AS is inconsistent with the actual expansion module type order
6	Error when the extension module state machine switches from enumeration state to configuration state
7	Error configuring expansion module parameters
8	Error when the expansion module state machine switches from the configuration state to the periodic loop state
9	The expansion module has an error in the cyclic data communication state

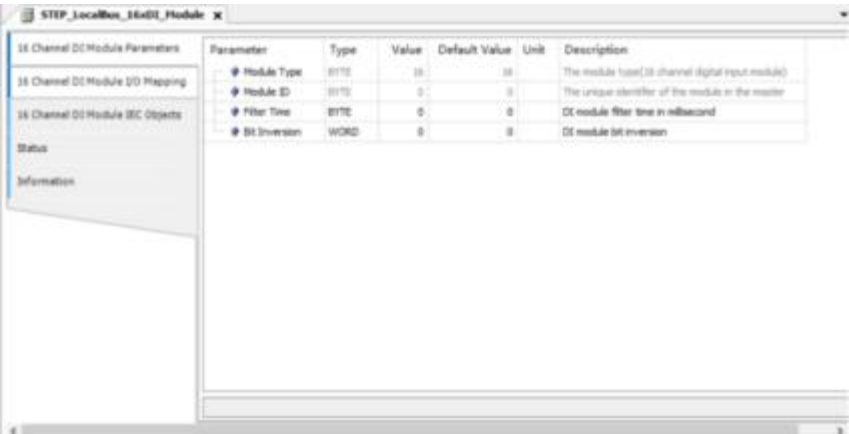
Figure master station information

The Localbus master station does not need to be configured, the configuration parameters are read-only, the user can view the specific error information through the error ID and error string parameters: The expansion module device has a baud rate parameter, the default communication rate is 6Mbps, the user does not need to use it Modified, except in special circumstances.

3.6.3. localbus slave configuration

For different Localbus slaves, the configuration parameters are also different.

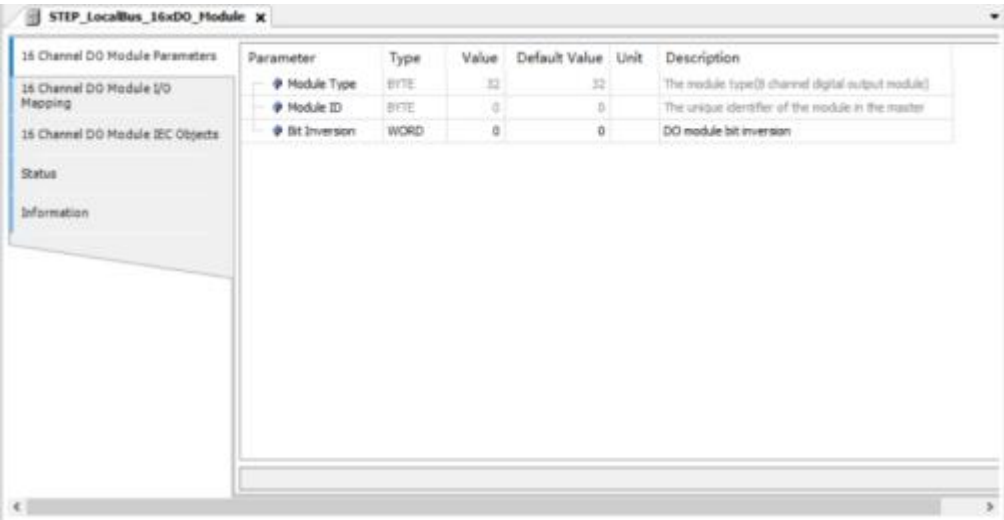
1. For the 16-channel digital input module, the configuration page is as follows:



The configuration parameters are as follows:

Configuration parameters	Parameter Description
Ffilter Time	Define the filter time in ms
BitInversion	Bit flip

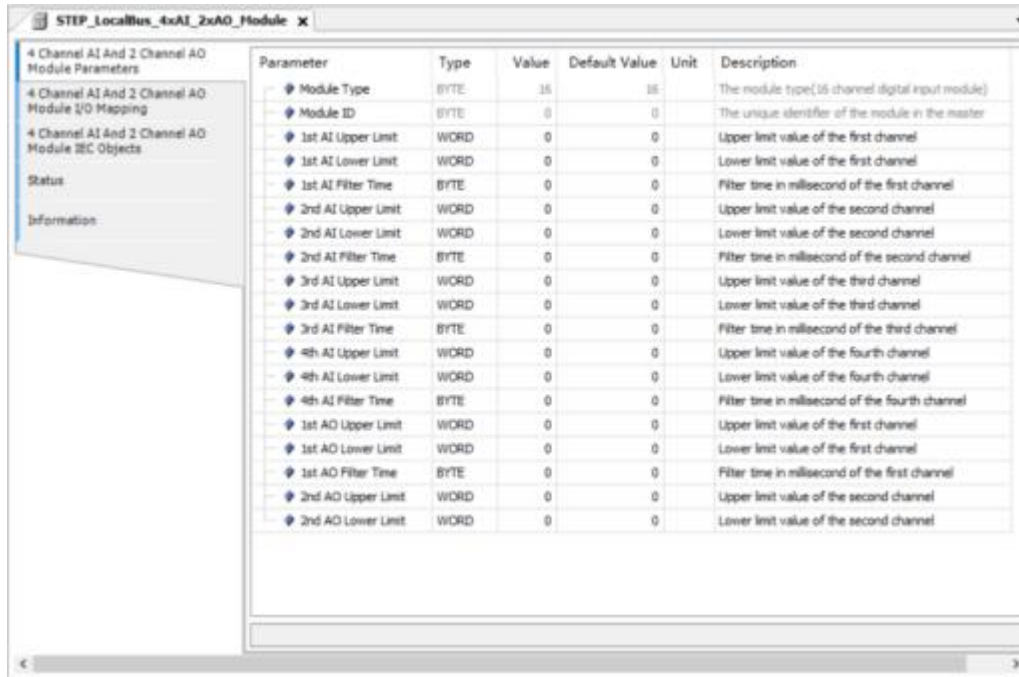
2. For the 16-channel digital output module, the configuration page is as follows:



The configuration parameters are as follows:

Configuration parameters	Parameter Description
BitInversion	Bit flip

3. For 4-channel analog input and 2-channel analog output module, the configuration page is as follows:



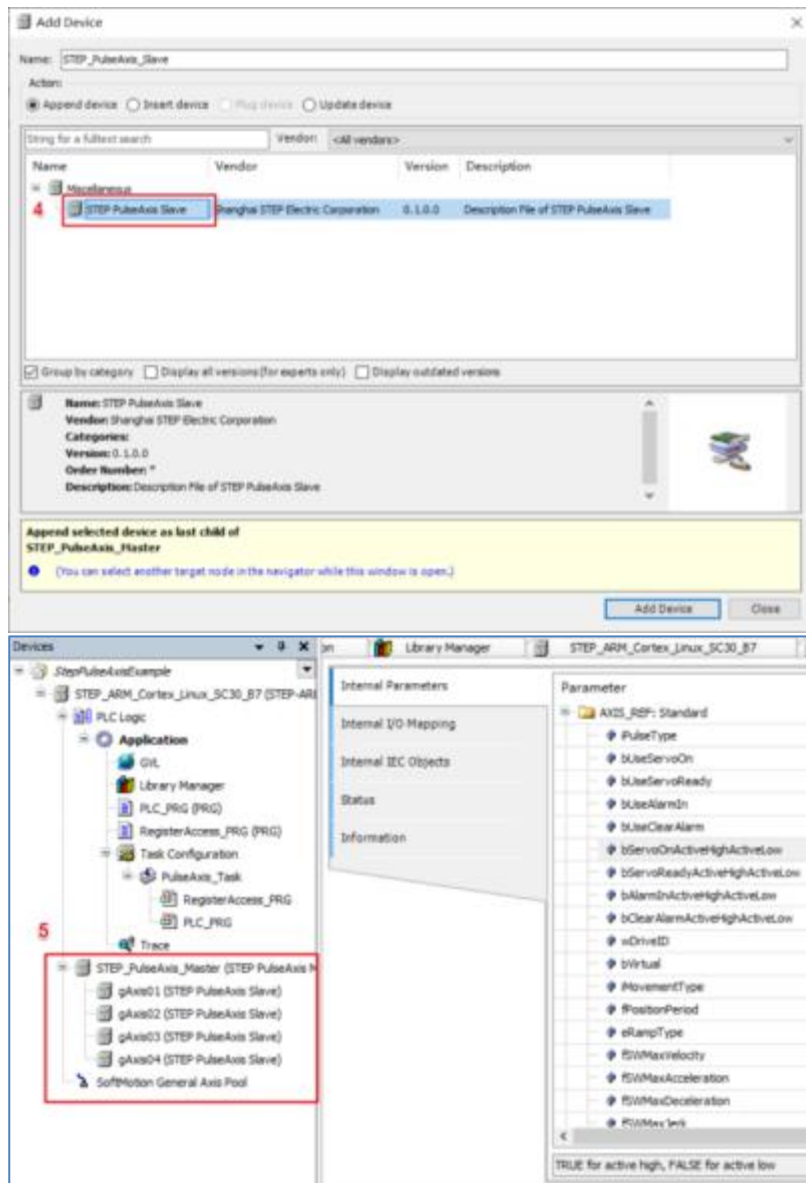
The configuration parameters are as follows:

Configuration parameters	Parameter Description
UpperLimit	Defines the upper limit of the input value.
LowerLimit	Defines the lower limit of the input value.
FilterTime	Defines the filter time in ms.

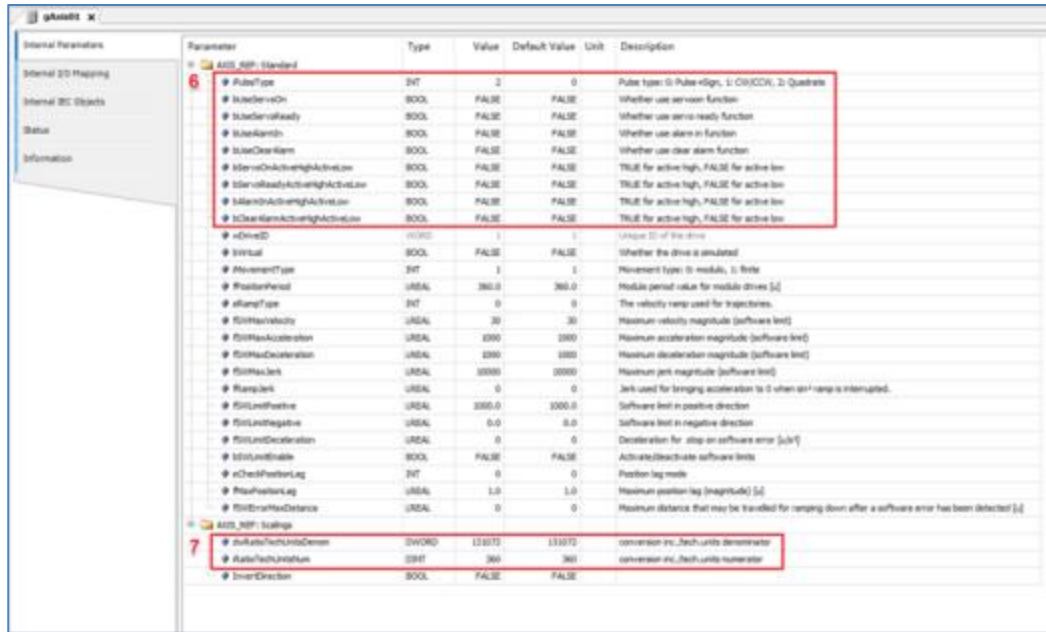
3.7. Pulse pulse axis configuration

For SC30-B6H controller and SC20-A3H, it can support 4 high-speed pulse outputs and ABZ encoder input. The pulse output type supports pulse plus direction, positive and negative pulse and quadrature pulse.

Using the configuration topology to add the pulse servo master and slave station equipment, the control of the pulse servo can be realized by using the motion control standard interface provided by CODESYS. Similarly, you can use the project wizard to add pulse axes when creating a new project. The number of configurable pulse axes is limited by the number of pulse axes supported by the controller where you are located.



3.7.2. Configuring the Pulse Axis Slave Device



parameter	Parameter Description
iPulseType	Pulse type. 0: pulse + direction 1: Positive and negative pulses 2: Quadrature pulse
bUseServoOn	Whether to use the servo enable pin on the interface
bUserServoReady	Is the pin ready using the servo on the interface
bUseAlarmIn	Whether to use the servo alarm input pin on the interface
bUserClearAlarm	Whether to use the clear servo alarm pin on the interface
bUseServoOnActiveHighActiveLow	Servo enable active high or active low
bUserServoReadyActiveHighActiveLow	Servo ready active high or active low
bUseAlarmInActiveHighActiveLow	Servo alarm input is active high or active low
bUserClearAlarmActiveHighActiveLow	Clear Servo Alarm Active High or Active Low
dwRatioTechUnitsDenom	Denominator for the ratio of user units to pulse units
iRatioTechUnitsNum	Numerator of the ratio of user units to pulse units
InvertDirection	Movement Reverse

3.7.3. Control pulse axis slave device

After the pulse axis slave device is configured, the pulse axis slave device can be controlled like an EtherCAT bus axis or an axis conforming to the PLCOpen standard, thereby supporting single axis control, electronic cam, CNC, Robotics, etc.

第四章 programming basics

Operands are objects of operator, function, function block or program operation in the user program, which can be used as input, output and intermediate storage of results. In STEP AS, common operands include direct addresses, constants and variables.

Similar to other high-level languages, STEP AS There are also concepts of constants and variables. A constant is a number whose value does not change. Variables are identifiers defined by the user. The storage location of variables can be specified by the user as the specific address of the %I area, %Q area, and %M area, or it can be assigned by the system without specifying the address, and the user does not need to pay attention to the storage location of these variables.

4.1. direct address

This type of fixed address is also called a direct variable, which is directly mapped to the specific address of the PLC device. The address information includes the storage location of the variable in the CPU, the storage size and the offset corresponding to the storage location.

Syntax: %<memory area prefix><size prefix><number>.<number>

The programming system supports the following store prefix

- 1) I: input, physical input, "sensor"
- 2) Q: output, physical output, "actuator"
- 3) M: storage location

The programming system supports the following size prefixes:

- 1) X: Bit , one.
- 2) B: Byte, a byte
- 3) W: Word, a word
- 4) D: Double Word, two words (4 bytes)
- 5) L: four characters (8 bytes)

The first number is the offset address of the variable corresponding to the memory prefix, ". "The number after the variable is BOOL When type, the number of bits after the offset address.

Example:

%QX7.5 The output area is offset by 7 bytes, the sixth bit (bit5).

%QX17 Output area offset 17 bytes

%IW215 Input area offset by 215 words

%MD48 memory area offset 48 double words

iVar AT %IW10: WORD;//iVar variable is word type, mapped to the input area offset 10

words s position

- ◆ The data type represented by the size prefix is X type variable is BOOL type, and the offset address should be specific;
- ◆ The size prefix matches the data type. A variable with a size prefix of type B should be declared as a byte data type, such as BYTE, SINT, USINT; a variable with a size prefix of W type should be declared as a word data type, such as WORD, INT, UINT; Variables with a size prefix of D type should be declared as a double-word data type, such as DWORD, DINT, UDINT.

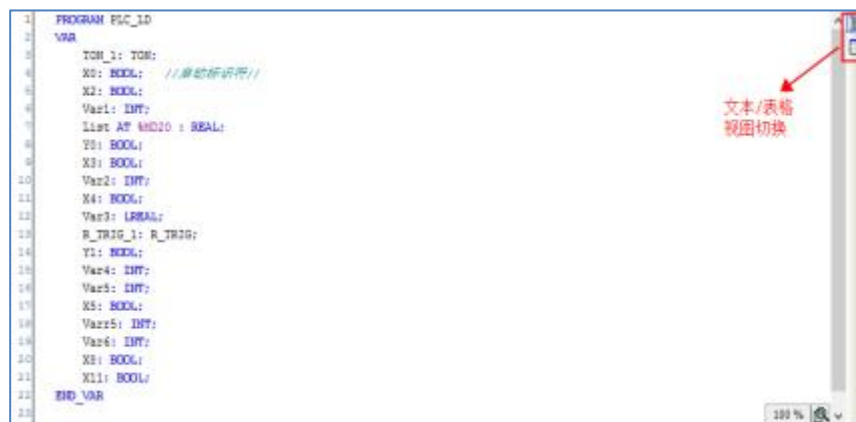
4.2. variable

Variables can be defined in the definition part of the POU or through the automatic declaration dialog, or in the DUT or GVL editor. The variable type is identified by the variable type keyword, such as VAR and END_VAR to identify the variable defined between it as local variable.

Variable types include local variables (VAR), input variables (VAR_INPUT), output variables (VAR_OUTPUT), input and output variables (VAR_IN_OUT), global variables (VAR_GLOBAL), temporary variables (VAR_TEMP), static variables (VAR_STAT), configuration variables (VAR_CONFIG).

4.2.1. Variable Definition

Variables can be defined in the declaration editor. The declaration editor has two display forms: text view and table view. The text view declaration editor of the POU is as follows:



The table view declaration editor is as follows:



Definition syntax: <identity> {AT <address>};<data type> {:=<initial value>}; The optional

part enclosed in curly brackets {}.

① logo

The identifier is the name of the variable. Variable naming should pay attention to the following:

- 1) Cannot contain spaces or special characters
- 2) cannot be a predefined keyword
- 3) Names are not case sensitive
- 4) There is no limit to the length of the name
- 5) The name cannot be defined repeatedly

The name of the defined local variable can be the same as that of the global variable. By default, the local variable is used. This variable can be used to represent the global variable, or the full path variable name can be used to specify the specific variable. E.g:

Local variable iVar:=1; global variable.iVar:=2; full path variable GVL.iVar:=3;

Some naming suggestions should be considered when naming: if the variable name should be prepared to express its meaning and data type, the variable should preferably use the Hungarian notation (variable name = attribute + type + object description).

② AT address

AT addresses are direct addresses.

③ type of data

Data types are divided into standard data types and user-defined data types

1) Standard data types

Standard data types are divided into boolean, integer, floating point, string, time type

type	keywords	scope	used internal memory
boolean type	BOOL	TRUE, FALSE, 0, 1	8Bit
BIT type	BIT	TRUE, FALSE, 0, 1, can only be used in structures or function blocks	1Bit
integer	BYTE	0 - 255	8Bit
	WORD	0 - 65535	16Bit
	DWORD	0 - 4294967295	32Bit
	LWORD	0 - 264-1	64Bit
	SINT	-128 - 127	8Bit
	USINT	0 - 255	8Bit
	INT	-32768 - 32767	16Bit
	UINT	0 - 65535	16Bit

	DINT	-2147483648 - 2147483647	32Bit
	UDINT	0 - 4294967295	32Bit
	LINT	-263-263-1	64Bit
	ULINT	0 - 264-1	64Bit
floating point type	REAL	1.401e - 3.403e+38	32Bit
	LREAL	2.2250738585072014e-308 - 1.7976931348623158e+308	64Bit
string	STRING	Only ASCII characters are supported (Chinese characters are not supported). The default maximum length is 80 characters. If the maximum length is exceeded, it will be removed. You can declare a maximum length in characters, Such as <code>str:STRING(35):=' This is a String' ;</code> String functions support a maximum of 255 characters.	Strings are stored in ASCLL form, and a byte is used to store the terminator
	WSTRING	Only UNICODE characters are supported (Chinese characters are supported). The default maximum length is 80 characters. If the maximum length is exceeded, it will be removed. You can declare a maximum length in characters, Such as <code>swstr:WSTRING(35):=' This is a WString' ;</code>	Store the string in UNICODE form and use two bytes to store the terminator
time	TIME		
	TIME_OF_DAY(DT)	time frame of a day	
	DATE	From January 1, 1970	
	DATE_AND_TIME(DT)	From January 1, 1970	

2) User-defined data type

User-defined data types include arrays, structures, enumerations, unions, aliases, subsets, references, and pointers. In the programming software STEP AS, it can be applied by right-clicking→**add object**→**DUT**To add structure\enumeration\union\alias 4 custom data types.

a) array

grammar:

<Array_Name> : ARRAY[<Il1>..

Il1, Il2, Il3 define the lower limit of the area, ul1, ul2, ul3 define the upper limit, the value must be an integer, and elem.Type is the data type of each array element.

initialization and example

```
Card_game: ARRAY [1..13,1..4] OF INT;
arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5];
```

```
arr2 : ARRAY [1..2,3..4] OF INT := [1,3(7)]; (* array value 1,7,7,7*)
arr3 : ARRAY [1..2,2..3,3..4] OF INT := [2(0),4(4),2,3]; (* array 0,0,4,4, 4,4,2,3*)
arr1 : ARRAY [1..10] OF INT := [1,2]; (* array is partially initialized, no elements are
initialized to default value 0*)
```

Array structure initialization example

```
Structure definition:
TYPE STRUCT1
STRUCT
  p1 : INT;
  p2 : INT;
  p3 : DWORD;
END_STRUCT
END_TYPE

Array structure initialization:
arr1 : ARRAY[1..3] OF STRUCT1 :=
  [(p1:=1,p2:=10,p3:=4723), (p1:=2,p2:=0,p3:=299), (p1:=14,p2:=5,p3:=112)];
```

Access union element syntax:

<Array-Name>[Index1,Index2].

Example:

```
Card_game[9,2]
```

b) structure

grammar:

TYPE <structurename> | EXTENDS DUTTYPE:

STRUCT

<declaration of variables 1>

...

<declaration of variables n>

END_STRUCT

END_TYPE

<structurename> is a type that can be used as a data type. EXTENDS DUTTYPE is optional, indicating that the members of DUTTYPE are inherited, and the members of DUTTYPE can be accessed through the structurename type variable. Here DUTTYPE can be a struct type, a union type or an alias.

initialization and example

Polygonline type structure definition:

```

TYPE Polygonline:
STRUCT
  Start: ARRAY [1..2] OF INT;
  Point1: ARRAY [1..2] OF INT;
  Point2: ARRAY [1..2] OF INT;
  Point3: ARRAY [1..2] OF INT;
  Point4: ARRAY [1..2] OF INT;
  End: ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE

```

initialization:

```

Poly_1 : polygonline :=
(Start:=[3,3], Point1:=[5,2], Point2:=[7,3], Point3:=[8,5], Point4:=[5,7], End:=
[ 3,5]);

```

Access structure element syntax:

<structurename>.<variable>

Example :

```
Poly_1.Start
```

c) enumerate

An enumeration type is composed of several strings of constants, which are called enumeration type values.

grammar:

TYPE <identifier>:

(<enum_0> ,<enum_1> , ...,<enum_n>) |<base data type>;

END_TYPE

identifier: custom enumeration type. enum_n: The constant value corresponding to the enumeration type. Each constant can declare its corresponding value. If it is not declared, the default value is used. base data type The corresponding data type of the enumeration constant, which can be omitted, and the default value is an integer.

initialization and example

```

TYPE TRAFFIC_SIGNAL:
  (red, yellow, green:=10); (* red initial value 0, yellow initial value 1, green initial
value 10 *)
END_TYPE

TRAFFIC_SIGNAL1 : TRAFFIC_SIGNAL;
TRAFFIC_SIGNAL1:=0; (* the value corresponding to this enum variable is red *)

FOR I := red TO green DO

  i := i + 1;

```

```
END_FOR;
```

d) joint

grammar:

TYPE <unionname>:

UNION

<declaration of variables 1>...<declaration of variables n>

END_UNION

END_TYPE

<unionname> is a type and can be used as a data type. All variables in the union have the same storage location, and the size of the space allocated for the variable of the union type is the size allocated by the variable that occupies the largest space in it

Example:

```
TYPE union1:
UNION
  a : LREAL;
  b : LINT;
END_UNION
END_TYPE
```

Access array element syntax:

<unionname>.<variable>

Example:

```
union1.a
```

e) alias

A data type is represented by an alias.

grammar:

TYPE <aliasname>: basetype END_TYPE

aliasname is the alias type name, used as the data type. basetype can be a standard data type or a user-defined data type.

Example:

```
TYPE
  alias1 : ARRAY[0..200] of byte;
END_TYPE
```

The initialization and access methods are consistent with their corresponding basic types

f) Subset

The subset data type is a subset of the basic data type defined by it. You can add a subset type by adding DUT, or you can directly declare a variable as the subset type.

DUT object syntax:

TYPE <name> :

<Inttype> (<ug>..og>**)**

END_TYPE;

Name : valid IEC identifier

Inttype: is the data type SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, one of LWORD).

Ug : is a constant that must conform to the lower bound range corresponding to the base type. The lower bound itself is contained within this range within.

Og : is a constant that must conform to the upper bound range corresponding to the base type. The upper bound itself is contained within this range within.

Example of DUT object declaration:

```
TYPE
    SubInt : INT (-4095..4095);
END_TYPE
Variable direct declaration example
VAR
    i : INT (-4095..4095);
    ui : UINT (0..10000);
END_VAR
```

g) quote

A reference is an alias for an object, and manipulating the reference is like manipulating the object.

grammar:

<identifier> : REFERENCE TO <data type>

identifier: reference identifier. data type: The data type of the referenced object.

Example and initialization:

```
ref_int : REFERENCE TO INT;
a : INT;
b : INT;
ref_int REF= a;      (* ref_int refers to a *)
ref_int := 12;       (* a value is 12 *)
b := ref_int * 2;     (* b value is 24 *)
ref_int REF= b;      (* ref_int reference b *)
ref_int := a / 2;     (* b value is 6 *)
```

➤ The BIT type cannot be referenced, that is, the definition of ref1:REFERENCE TO BIT is not allowed

h) pointer

The pointer holds the address of an object, and the pointer can point to any data type

(except the BIT type)

grammar:

<identifier>: POINTER TO <data type>;

identifier: pointer identifier. data type: The data type pointed to by the pointer. Pointers are manipulated through address operators. Address operators include ADR (get variable address) and ^ (value corresponding to variable address)

Example and initialization

```
VAR
  pt:POINTER TO INT;          (* declares pointer pt to type INT*)
  var_int1:INT := 5;
  var_int2:INT;
END_VAR

pt := ADR(var_int1);          (* address of variable var_int1 is assigned to pointer pt *)
var_int2:= pt^;               (* Get the value corresponding to the pointer through the
^ address operator*)
pt^:=33;                       (* Assign value to the variable var_int1 corresponding to
the pointer*)
```

④ initial value

The default value of variable initialization is 0, and the user can add a custom initialization value through the assignment operator "==" when the variable is declared. The initialization value is a valid ST expression. ST expressions are composed of operators, operands, and assignment expressions. Operators mainly include addition (+), subtraction (-), multiplication (*), division (/), etc.; operands mainly refer to constants, variables and Function; assignment expression refers to the assignment operator "==" to a variable in an ST expression. Therefore, initialization can be a constant, variable or function, just make sure that the variable used is already initialized.

Example:

```
VAR
  var1 : INT := 12;            (* Integer variable initial value 12*)
  x : INT := 13 + 8;           (* constant expression defines initial value*)
  y : INT := x + fun(4);        (* initial value contains function call*)
  z : POINTER TO INT := ADR(y); (* pointer is initialized by address function ADR*)
END_VAR
```

- The global variable list (GVL) is generally initialized before the definition of the POU local variables;
- When the pointer is initialized at the time of definition, if the default is modified online, the pointer will not be initialized (the pointer still points to the variable before the online modification)

4.2.2. Variable types

The main variable types include: local variables (VAR), input variables (VAR_INPUT), output variables (VAR_OUT), input and output variables (VAR_IN_OUT), global variables (VAR_GLOBAL), temporary variables (VAR_TEMP), static variables (VAR_STAT) and configuration variable (VAR_CONFIG).

Variable type declaration syntax:

<type_key> | attribute_key

variable1;

variable2;

...

END_VAR

type_key: Type keyword, including VAR (local variable), VAR_INPUT (input variable), VAR_OUTPUT (output variable), VAR_IN_OUT (input and output variable), VAR_GLOBAL (global variable), VAR_TEMP (temporary variable), VAR_STAT (static variable), VAR_CONFIG (configuration variable).

attribute_key: attribute keywords, including RETAIN, PERSISTENT, CONSTANT, used to clarify the scope of variables. The attribute keywords are described in detail below.

◆ RETAIN variable (reserved variable)

The RETAIN variable can continue to retain the original value after the PLC is powered off and restarted or warmly reset. RETAIN variables are stored in a specific RETAIN storage area. A practical example would be a counter on a production machine; after a power failure, it will recount where it left off.

Example

Define the RETAIN variable in the program:

```
PROGRAM PLC_PRG
  VAR RETAIN
    iRem1 : INT;
  END_VAR
```

Define the RETAIN variable in the global variable table:

```
VAR_GLOBAL RETAIN
  gvarRem1 : INT;
END_VAR
```

- If the RETAIN variable is declared in the program, only this RETAIN variable is saved in the RETAIN storage area;
- If a RETAIN variable is declared in a function block, the entire function example data will be saved in the RETAIN storage area, but only this RETAIN variable is treated as a reserved variable;
- If a RETAIN variable is declared in a function, this variable declaration has no effect.

◆ PERSISTENT variable (permanent variable)

The definition of VAR PERSISTENT is always the same as the definition of VAR PERSISTENT RETAIN or VAR RETAIN PERSISTENT, which means that permanent variables have cold-reset retention and program download in addition to the characteristics of RETAIN variables (power-down retention and warm-reset retention). Reserved value properties. Persistent variables are only initialized when the initial value is reset. An example of a common permanent variable is a program runtime counter that can continue to count after a power failure and continue to count after the program is re-downloaded.

Example

Example of permanent variable table:

```

VAR_GLOBAL PERSISTENT RETAIN
  iVarPers1 : DINT;
  bVarPers : BOOL;    // Add the permanent variable instance path to the right-click
                        menu of the permanent variable table editor
  PLC_PRG.PERS: INT;  (*Persistent variable PERS defined in the PLC_PRG program *)
END_VAR

```

- An application has only one permanent variable table, and the permanent variable table can only be added by right-clicking the application - adding object - permanent variable.
- You can add permanent variables in the program through the PERSISTENT property, and then in the permanent variable editor, through the right-click menu - add all instance paths, add all the permanent variables in the program to the permanent variable table.

The following table lists whether a variable retains its original value or is initialized after a reset, power failure, etc.

X := keep original value - := value is initialized

action	Var	VarRETAIN	VAR PERSISTENT or VAR PERSISTENT RETAIN or VAR RETAIN PERSISTENT
power down	-	X	X
warm reset	-	X	X
cold reset	-	-	X
Initial value reset	-	-	-
Program download	-	-	X
online modification	X	X	X

illustrate:

- 1) RETAIN variables and PERSISTENT variables are reserved variables, and they are reserved in the same reserved variable area of the programming system.
- 2) Direct variables mapped to %M addresses can be declared as reserved variables, while direct variables mapped to %I and %Q A variable cannot be declared as a reserved variable. (Reserved variables cannot be declared as direct variables when automatically declared, So the %M direct variable can only be entered manually).

① local variable

The variables between VAR and END_VAR inside the POU are local variables and cannot be accessed externally.

Assignment format:

local variable := value

Example

```

VAR
  iLoc1:INT; (* local variable*)
END_VAR

```

② input variable

The variables between VAR_INPUT and END_VAR in the POU are all input variables, which can be assigned values at the calling position.

POU call format:

local variable := caller input value

Example

```
VAR_INPUT
  iIn1:INT; (*input variable*)
END_VAR
```

- *Input variables can also be modified within the POU, even if the CONSTANT attribute is added*

③ output variable

The variables between VAR_OUTPUT and END_VAR inside the POU are output variables. Output variables can be returned to the caller when called, and the caller can do further processing.

POU call format:

output variable => caller match type variable

Example

```
VAR_OUTPUT
  iOut1:INT; (* output variable*)
END_VAR
```

- *For functions (FUNCTION) and methods (METHOD) in addition to the return value, there can be additional output variables, but the caller must be assigned to receive variables at the time of the call. For example fun(iIn1 := 1, iIn2 := 2, iOut1 => iLoc1, iOut2 => iLoc2);*
- *For function blocks, the function block output variable can be assigned to the caller after the call.*

④ input and output variables

The variables between VAR_IN_OUT and END_VAR inside the POU are input and output variables. Input and output variables can not only be passed into the called POU, but also can be modified inside the called POU. The variable that is actually passed to the called POU is a reference to the caller's variable.

Example

```
VAR_IN_OUT
  iInOut1:INT; (*input/output variable*)
END_VAR
```

- *Because the variable passed to the called POU is the reference of the caller variable, the input and output variables in the function block instance cannot be directly accessed, that is, <FBinstance><InOutVariable> cannot be used directly, because the input variable is already the caller variable. citations, which have changed;*
- *Input and output variables cannot be constants and direct variables of Bit type (eg xBit0 AT %I2.0:BOOL). If you need to declare input and output constants, you can add the CONSTANT attribute (VAR_IN_OUT CONSTANT). If you need a direct variable of Bit type, you need to add an intermediate variable as an input and output variable, and then assign the value of the intermediate variable to the direct variable of Bit type.*

Examples of direct variables of type Bit:

```
VAR_GLOBAL
```

```

    xBit0 AT %MX0.1 : BOOL;      (* declare a direct variable of type Bit*)
    xTemp : BOOL;               (* Intermediate variables*)
END_VAR

// function block with input and output variables (xInOut)
FUNCTION_BLOCK FB_Test
VAR_INPUT
    xIn : BOOL;
END_VAR
VAR_IN_OUT
    xInOut : BOOL;
END_VAR

IF xIn THEN
    xInOut := TRUE;
END_IF

// Call the function block in the program
PROGRAM Main
VAR
    xIn : BOOL;
    I1 : FB_Test;
    I2 : FB_Test;
END_VAR

// Use the direct address variable of type Bit, compile an error
//I1(xIn:=xIn, xInOut:=xBit0);
// Pass the value of xBit0 to the function block through the intermediate variable xTemp,
// and then assign the intermediate variable to xBit0
xTemp := xBit0;
I2(xIn:=xIn, xInOut:=xTemp);
xBit0 := xTemp;

```

Input and output constants (VAR_IN_OUT CONSTANT) can only be read but not written, and input variables can be modified in the current version, even if constant attributes are added, so input and output constants can be used to make variable attributes unmodifiable.

Example of input and output constants:

```

PROGRAM PLC_PRG
VAR
    sVarFits : STRING(16);
    sValFits : STRING(16) := '1234567890123456';
    iVar: DWORD;
END_VAR
POU(sReadWrite:= '1234567890123456',          scReadOnly:= '1234567890123456',
iVarReadWrite:=iVar);
//POU(sReadWrite:=sVarFits, scReadOnly:=sVarFits, iVarReadWrite:=iVar);
//POU(sReadWrite:=sValFits, scReadOnly:=sValFits, iVarReadWrite:=iVar);
//POU(sReadWrite:=sVarFits, scReadOnly:= '23' , iVarReadWrite:=iVar);

FUNCTION POU : BOOL
VAR_IN_OUT
    sReadWrite : STRING(16); (* This string is readable and writable within this POU*)
    iVarReadWrite : DWORD;   (* This variable is readable and writable in this POU*)
END_VAR
VAR_IN_OUT CONSTANT
    scReadOnly : STRING(16); (* within this POU this string is read only*)
END_VAR
sReadWrite := 'string_from_POU';
iVarInPOU := STRING_TO_DWORD(scReadOnly);

```

⑤ global variable

Variables defined between VAR_GLOBAL and END_VAR are global variables. General variables, constants, and reserved variables can be declared as global variables.

existSTEP ASIn the programming software, you can right-click Apply > Add Object>Add **global variable table** to add the global variable table, and then add the global variable in the global variable table.

Example

```
VAR_GLOBAL
    iGlobVar1:INT; (* global variable*)
END_VAR
```

- If a local variable has the same name as a global variable, when the variable name is directly operated, it means that the operation is a local variable. You can add the global scope operator (.) before the variable name to operate the global variable, such as .iGlobVar1;
- Global variables are always initialized before local variables.

⑥ Temporary variables

Variables defined between VAR_TEMP and END_VAR are temporary variables that are initialized on each call.

Example:

```
VAR_TEMP
    iTemp1:INT; (*temporary variable*)
END_VAR
```

- Temporary variables can only be declared in programs and function blocks;
- Temporary variables can only be used in the declared program or function block.

⑦ static variable

Variables defined between VAR_STAT and END_VAR are static variables. Static variables are initialized on the first call and after each call

After this POU, the variable value remains.

Example:

```
VAR_STAT
    iStat1:INT; (* static variable*)
END_VAR
```

- Static variables can only be declared in function blocks, functions and methods, not in programs;
- Static variables can only be used within declared POU's.

⑧ configuration variable

Variables defined between VAR_CONFIG and END_VAR are configuration variables. Configuration variables are direct variables, generally mapped to indeterminate address direct variables defined by function blocks. A variable with an indeterminate address can be defined in the function block. The address of this variable is represented by "*" to represent an indeterminate address (any address), and then a configuration variable table (by adding a global variable table) is added to put all The indeterminate address variables in the function block instance are added to the configuration variable table, and all indeterminate addresses are clarified in this variable table, so that the indeterminate address variables in all function blocks can be managed centrally.

Function block undefined address variable definition syntax:

<identifier> AT %<I|Q|M>* : <data type>

The final determination of the address is done in the "variable configuration" of the global variable list;

Example:

```
FUNCTION_BLOCK locio
VAR
    xLocIn    AT %I* : BOOL := TRUE;
    xLocOut   AT %Q* : BOOL;
END_VAR
```

Two I/O-variables are defined here, a local input variable (%I*) and a local output variable (%Q*).

Then add the "Global Variable List" object (GVL). Enter the specific address of the instance variable declaration between the keywords VAR_CONFIG and END_VAR, where the instance variable refers to the complete instance path including the POU, and the specific address corresponds to the undefined specified address in the function block (%I*, %Q*). In addition, the data type must be consistent with the declaration of the function block.

Configuration variable definition syntax:

<instance variable path> AT %<I|Q|M><location> : <data type>;

Example

```
PROGRAM PLC_PRG
VAR
    locioVar1: locio;
    locioVar2: locio;
END_VAR
VAR_CONFIG (*correct variable configuration table*)
    PLC_PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;
    PLC_PRG.locioVar1.xLocOut AT %QX0.0 : BOOL;
    PLC_PRG.locioVar2.xLocIn AT %IX1.0 : BOOL;
    PLC_PRG.locioVar2.xLocOut AT %QX0.3 : BOOL;
END_VAR
```

- Generally, there is no need to configure variables, because for I/Q address input/output, variables can be mapped to I/Q addresses through the input assistant (or directly input instance variable path) in the I/O mapping interface of the corresponding module;
- Configuration variables are generally mapped to indeterminate address variables in function blocks, and it can also map indeterminate address variables in programs;
- If there are only indeterminate address variables or only configuration variables, an error will be reported during compilation. The two are used together.

4.3. constant

In PLC programming, some parameters with constant values will be used, such as timer time, conversion ratio, etc. These parameters with constant values are called constants.

Constant declaration syntax:

VAR CONSTANT

<identifier>:<type> := <initialization>;

END_VAR

Example:

```
VAR CONSTANT
  c_iCon1:INT:=12;
END_VAR
```

STEP ASSupports constants of multiple data types, common constants include boolean, integer, time, string, etc. The specific constants are shown in the following table:

type	describe	Example
boolean type	There are two values TRUE and FALSE (1 and 0 can also be used), 1 equals TRUE, 0 equals FALSE	TRUE, FALSE, 1, 0
BIT type	Similar to the Boolean type, it can only be used in a structure (occupancy bits) or a function block (map the direct address of the BOOL type)	TRUE, FALSE, 1, 0
integer	The value of an integer constant can be binary, decimal, octal, and hexadecimal. If the integer value is not a decimal value, it can be represented by "base" plus the sign "#" before the integer value. 10 to 15 in decimal are A to F in hexadecimal	decimal : 66 binary : 2#101 Octal : 8#72 Hexadecimal: 16#3A Type constants: INT#22, BYTE#204
floating point type	Floating point constants are represented by decimal fractions and exponents, following standard scientific notation format	7.4, 2.3e+9, REAL#3.12
ASCIIstring	ASCII string constants are between two single quotes and can contain spaces and special characters. One character is represented by one byte, only ASCII characters are supported (Chinese characters are not supported). The default maximum length is 80 characters. If the maximum length is exceeded, it will be removed. You can declare the maximum length of characters, such as str:STRING(35):='This is a String'; String functions support a maximum of 255 characters.	\$ as an escape character Example: '\$30': 0, character 0, ASCII character corresponding to hexadecimal 30 \$\$: \$, dollar character \$': 'apostrophe
UNICODE string	The UNICODE string constant is between two double quotation marks, and one character stands for two bytes. Only UNICODE characters are supported (Chinese characters are supported). The default maximum length is 80 characters. If the maximum length is exceeded, it will be removed. You can declare the maximum length of characters, such as wstr:WSTRING(35):=" This is a WString" ;	"Unicode string"
time	Time constants are generally used to manipulate time, consisting of "T#" (or "t#") plus "time value". The unit of time value includes days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms)	T#12h34m15s;
time	The time range of one day, syntax: TOD# time	TOD#15:36:30.123

type	describe	Example
	value.	
date	Since January 1, 1970, syntax: d# date.	D#2015-02-12
date time	Date constants and time constants are combined into date constant constants, starting from January 1, 1970, syntax: dt# date.	DT#2004-03-29-11:00:00

- *Except for BOOL, BIT and string types, other types can use the keyword # constant value to represent a type constant*

第五章 Programming language

5.1. Introduction to programming languages supported by STEP AS

The programming background supports a total of 6 PLC programming languages:

- a) Ladder Logic Diagram (LD) Ladder Logic Diagram
- b) Function Block Diagram (FBD) function block diagram
- c) Structured Text (ST) structured text
- d) Instruction list (IL) instruction list
- e) Sequential Function Chart (SFC) Sequential Function Chart
- f) Continuous Function Chart (CFC) Continuous function diagram

Among them, LD, FBD, ST, SFC, IL are based on the IEC 61131-3 standard, and CFC is an extension of the IEC 61131-3 standard.

No matter which language the user chooses, the basic editing method in the programming interface is universal, which brings great convenience to programming.

Support common Windows Text editor features such as support for "copy" (Ctrl+C) "paste" (Ctrl+V) and "delete" (Del) and other shortcut keys;

Support standard Windows superior <Ctrl>, <Shift> Press the button to make multiple selections;

Support function keys <F2> Start the input assistant, and the system provides corresponding input prompts or choices according to the specific environment.

- *FBD and IL are temporarily not supported by default. For details on SFC and CFC, please refer to the help document.*

5.2. Structured Text Language (ST)

Structured text is a textual high-level language similar to PASCAL and C. Program code consists of instructions, and instructions consist of keywords and expressions. Unlike the IL language, ST statement loops can contain numerous statements, allowing the development of complex structures.

E.g:

```
IF Value < 7 THEN
  WHILE Value < 8 DO
    Value := value +1;
  END_WHILE;
END_IF;
```

5.2.1. Expressions

An expression is a structure that, when evaluated, can be used in an instruction.

Expressions consist of operators and operands. An operand can be a constant, variable,

function call or other expression.

E. g:

constants, for example: 20, t#20s, '22231 test'

variables, for example: iVar, Var1[2, 3]

Function call, the value is the return value of the call, for example: Fun1(1, 2, 4)

Other expressions: 10+3, var1 OR var2, (x+y)/z, iVar1:=iVar2+22

Evaluation of expressions evaluates operands by operator, in an order defined by specific operator precedence. The operator with the highest precedence in the expression shall be evaluated first, followed by the next lower precedence operator, etc., in order from highest to lowest. Operators with equal precedence shall be performed in the left-to-right order as written in the expression.

example:

If A, B, C, and D are of type INT and have values 1, 2, 3, 4, respectively, then A+BC*ABS(D) should be equal to -9, and (A+BC)*ABS(D) should be equal to 0.

When an operator has two operands, the leftmost operand should be evaluated first. For example, in the expression SIN(A)*COS(B), the expression SIN(A) should be evaluated first, then COS(B), and finally the product.

The following table documents the operators of ST language:

operate	symbol	priority
brackets	(expression)	Highest
function call	function name (parameter list, separated by commas)	
exponentiation	EXPT	
Negative value	-	
make up	NOT	
take	*	
remove	/	
remainder	MOD	
add	+	
reduce	-	
Compare	<, >, <=, >=	
equal	=	
not equal to	<>	
logical and	AND	
logical XOR	XOR	
logical or	OR	lowest

5.2.2.ST instruction

The entire ST program consists of instructions, which are separated by semicolons ";". These instructions consist of keywords and expressions. The ST instructions are as follows:

keywords	illustrate	Example
:=, S=, R=	assign, set, reset	A:=B; C:=SIN(X); b1 R=cond1
	Function block calls and outputs	CMD_TMR (IN:=%IX5, PT:=300); A:=CMD_TRM.Q
RETURN	Return (exit the current POU)	

IF	choose	<pre> D:=B*B; IF D<0.0 THEN C:=A; ELSIF D=0.0 THEN C:=B; ELSE C:=D; END_IF; </pre>
CASE	multiple selection	<pre> CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE; </pre>
FOR	FOR loop	<pre> J:=101 FOR I:=1 TO 100 BY 2 DO IF ARR[I] = 70 THEN J:=I; EXIT END_IF; END_FOR; </pre>
WHILE	WHILE loop	<pre> J:=1; WHILE J<100 AND ARR[J]<>70 DO J:=J+2; END_WHILE </pre>
REPEAT	REPEAT loop	<pre> J:=-1; REPEAT J:=J+2; UNTIL J=101 OR ARR[J]=70 END_REPEAT </pre>
EXIT	exit the loop	EXIT;
CONTINUE	Continue the loop for the next execution	CONTINUE
JMP	jump	<pre> label: i:=i+1; JMP label </pre>
;	empty statement	;

(1) assignment instruction

The assignment instruction is used for variable assignment, that is, the left side of the assignment keyword is the variable, and the right side is the value to be assigned, which is assigned through the assignment keyword.

E.g:

```
Var1 := Var2 * 10;
```

After completing the execution, Var1 is 10 times the value of Var2. The assignment keywords include three kinds of ":", "S=", and "R=".

➤ ":=": For general assignment, the rvalue is directly assigned to the lvalue, and the lvalue and the rvalue are equal.

➤ "S=": Assigns a value to set, indicating that if the rvalue is TRUE, the lvalue variable becomes TRUE (set) until called R=early order

Initiate.

➤ "R=": Assigns an assignment to reset, indicating that if the rvalue is TRUE, the lvalue variable becomes FALSE (reset). for reset S=command set bit variable.

E. g:

```
a S = b;
```

Once b is TRUE, a remains TRUE, even after b becomes FALSE.

(2) function block call

grammar:

```
<FB instance name>(FB input variable:=<value or variable>|, <more FB input
variable:=<value or variable>|...
```

More FB input variables. In the example below, a delay function block (TON) is called and parameters IN and PT are assigned. Then the result variable Q is assigned to variable A. The delay FB is instantiated with "TMR:TON".

Example:

```
<FB instance name>, <FB variable>:
TMR(IN := X0, PT := T#300ms);
A:=TMR.Q;
```

(3) RETURN instruction

The RETURN instruction means to leave this POU when the precondition is TRUE.

grammar:

```
RETURN;
```

Example:

```
IF b=TRUE THEN
    RETURN;
END_IF;
a:=a+1;
```

If b is TRUE, the statement "a:=a+1;" will not be executed and the POU will be returned immediately.

(4) IF instruction

Through the IF keyword, the execution condition can be judged, and the corresponding instruction can be executed according to the execution condition.

grammar:

```
IF <boolean expression1> THEN
    <IF_instruction>
```

```

{ELSIF <boolean expression2> THEN
    <ELSIF_command 1>
.
.
ELSIF <Boolean expression n> THEN
    <ELSIF_command n>
ELSE
    <ELSE_command>}
END_IF;

```

Parts inside { } are optional.

If <boolean expression1> is TRUE, then only <IF_instruction> is executed, others are not executed, otherwise, starting from <boolean expression2>, the Boolean conditional expressions are evaluated one by one until one of the expressions evaluates to the value TRUE, then execute the instruction corresponding to this expression, if no expression is TRUE, execute the instruction corresponding to <ELSE_instruction>.

Example:

```

IF temp <17 THEN
    heating_on := TRUE;
ELSE
    heating_on := FALSE;
END_IF;

```

(5) CASE instruction

Using the CASE instruction, you can list locations according to a condition variable according to its corresponding multiple values. the corresponding command. Condition variables can only be integers.

grammar:

```

CASE <Var1> OF
    <value1>:
        <Instruction 1>
    <value2>:
        <Instruction 2>
    <value3, value4, value5>:
        <Instruction 3>
    <value6 .. value10>:
        <Instruction4>
    ...
    <value n>:
        <Instruction n>
    ELSE
        <ELSE Instruction>
END_CASE;

```

CASE instructions are processed according to the following flow:
if variable<Var1>value of<valueI>, So<Instruction I>will be executed.

if<Var1>does not match any of the values, then<ELSE Instruction>be executed.

If the same instruction is executed with several variable values, then these values can be written one after the other, separated by commas, and thus executed together.

If the same instruction will be executed in a variable scope, you can write the initial value and the end value, separated by two dots.

Example:

```
CASE State OF
  2:
    Var1 := Var1 + 1;
  1..6:
    Var1 := Var1 - 7;
  7..20:
    Var1 := Var1 + 5;
END_CASE
```

(6) FOR loop

With FOR loops, it is possible to write repetitive processing logic.

grammar:

```
FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <Step size>} DO
  <instructions>
END_FOR;
```

Parts within {} are optional.

INT_Var is a counter, which is an integer type. As long as the counter <INT_Var> is not greater than <END_VALUE>, <Instructions> will be executed. Check the condition first before executing <Instructions>, if <INIT_VALUE> is greater than <END_VALUE>, <instructions> will not be executed.

When <Instructions> is executed once, <INT_Var> automatically increases <Step size>. <Step size> can be any integer value. If this parameter is not written, the default value is 1. The loop stops when <INT_Var> is greater than <END_VALUE>.

Example:

```
FOR Counter:=1 TO 5 BY 1 DO
  Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

Assuming the default value of Var1 is 2, after the FOR loop, its value is 32.

(7) WHILE loop

The WHILE loop can be used as a loop processing like the FOR loop, but unlike the FOR loop, the loop condition can be any Boolean expression. Once the loop condition is met, the loop executes, otherwise it exits the loop.

grammar:

```
WHILE <boolean expression> DO
  <instructions>
END_WHILE;
```

When the value of <Boolean_expression> is TRUE, the <Instructions> instruction starts to execute until the value of <Boolean_expression> is FALSE. <Boolean_expression> is FALSE the first time, then <Instructions> will never be executed. If <Boolean_expression> will never be FALSE, then <Instructions> will be executed repeatedly, which is called an infinite loop. Make sure not to have an infinite loop when programming.

Example:

```
WHILE Counter<>0 DO
    Var1:= Var1*2;
    Counter := Counter-1;
END_WHILE
```

In a sense, WHILE and REPEAT loops are more powerful than FOR loops because there is no need to count the number of loops before executing the loop. Therefore, in some cases, both the WHILE loop and the REPEAT loop are sufficient. However, a FOR loop is better if the number of loops is known.

(8) REPEAT loop

A REPEAT loop is different from a WHILE loop because the loop condition is checked after the loop instruction is executed. This means that the loop executes at least once, regardless of the loop condition value.

grammar:

```
REPEAT
    <instructions>
UNTIL <Boolean expression>
END_REPEAT;
```

The execution logic is: <Instructions> is executed until the value of <Boolean expression> is TRUE. If <Boolean expression> evaluates to TRUE the first time, then <Instructions> is executed only once. If the value of <Boolean_expression> is never TRUE, then <Instructions> will be executed forever, resulting in an infinite loop.

Example:

```
REPEAT
    Var1:=Var1*2;
    Counter:=Counter-1;
UNTIL Counter=0;
END_REPEAT;
```

(9) CONTINUE statement

The CONTINUE instruction is used in FOR, WHILE and REPEAT loops to end the current loop early and restart the next loop.

Example:

```
FOR Counter:=1 TO 5 BY DO
    INT1:=INT1/2;
    IF INT1=0 THEN
        CONTINUE;          (*provides division by zero*)
    END_IF;
    Var:=Var1/INT1;         (*executed,if INT1 is not 0*)
END_FOR;
```



```
Erg:=Var1;
```

(10) EXIT statement

EXIT is used in FORWHILE or REPEAT loops, to end the loop, regardless of other aborting conditions.

(11) JMP statement

The JMP instruction is used to perform an unconditional jump to a program marked by a jump label line.

grammar:

<label>:

JMP <label>;

<label> The label name is at the beginning of the program line, and the JMP instruction must have a jump target, which is a predefined label. After reaching the JMP instruction, the program will jump to the specified label to start execution.

Example:

```
aaa :=0;
_label11:aaa:=aaa+1;
(*instructions*)
IF (aaa < 10) THEN
    JMP _label11;
END_IF;
```

The variable aaa is initially 0, as long as it is less than 10, the program will jump to label11 and re-execute, so it will affect the repeated execution of the program between the JMP instruction and the label.

Such functionality can also be implemented with a WHILE or REPEAT loop. Jump instructions should generally be used sparingly because they reduce code readability.

(12) Notes

There are two ways to write comments in structured text.

use "(*" start"*)" Finish. This allows comments to be commented across lines. E.g: "(*This is a comment.*)"

Single-line comments, use "//" Begins a comment until the end of the line. E.g: "// This is a comment."

Comments can be anywhere in the declaration or implementation section of the ST editor.

Nesting of comments: Comments can be placed inside other comments

Example:

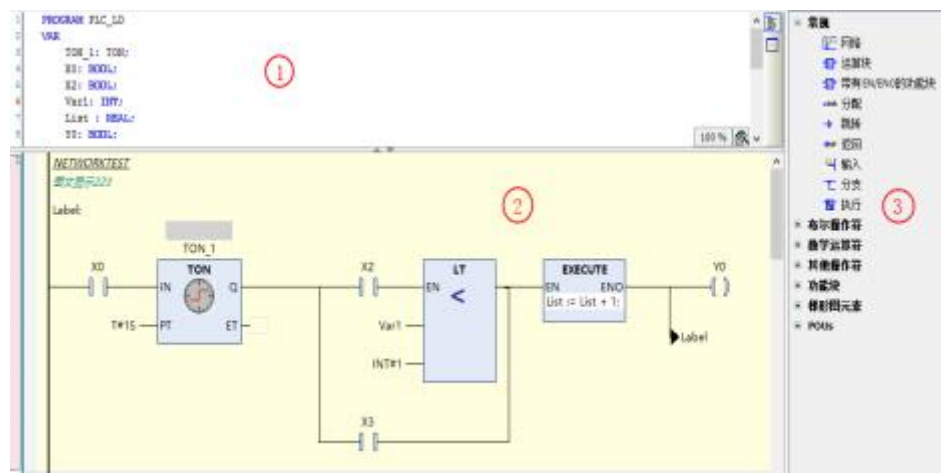
```
(*
a:=inst.out; (*to be checked*)
b:=b+1;
*)
```

5.3. Ladder Logic Diagram (LD)

5.3.1. Overview

Ladder diagram is a graphical programming language, similar to the structure of circuit diagram. A ladder diagram consists of a series of networks (also called sections, hereinafter collectively referred to as "networks"), each starting with a vertical line (power rail, energy flow line) on the left. A network consists of contacts, coils, optional POU (function, function blocks, programs), jumps, labels, and connecting lines.

The bus on the left side of the network is an energy flow line, and its state is always TRUE. After the bus, elements such as contacts, operation blocks, and coils will be connected. Boolean variables are assigned to each contact. If the value of the variable is TRUE, which is equivalent to the switch being closed, the condition is passed from left to right along the connection line, otherwise the switch is open. The coil on the right side of the network receives an "ON" or "OFF" signal from the left side, and the corresponding TRUE or FALSE is written to the Boolean variable associated with the coil. The ladder diagram editing interface is as shown below.



➤ **Description:** 1 - Variable definition area; 2 - Ladder diagram programming area 3 - Toolbox

The main elements of the ladder diagram include contacts, coils, operation blocks, branches, comments, etc. Add these elements to the network by inserting, dragging, scribing, copying and pasting to form a ladder diagram to execute logic. The font, operand and comment display of the ladder diagram interface can be set through [Tools>Options>FBD/LD Editor].

The ladder diagram supports online debugging functions such as monitoring, writing values, forcing values, and breakpoints.

Description of branches: branches are divided into closed and non-closed ones, closed ones are called parallel branches, and non-closed ones are called branches directly.

5.3.2. Ladder Diagram Elements

Ladder diagram elements include networks, contacts, coils, operation blocks, execution blocks, branches, jumps, labels, and returns.

Contact, coil, operation block input and output are associated with operands, operands can be variables, constants (TRUE, FALSE, 1, 2, etc.), addresses, see the variable definition

for details.

The LD element is located in the toolbox (menu command [View>Toolbox]), as shown in the following figure. In addition to the general elements, ladder diagram elements and IEC standard operators (such as Boolean operators and mathematical operators), the toolbox also includes function blocks and POUs defined in the current program.



1 network

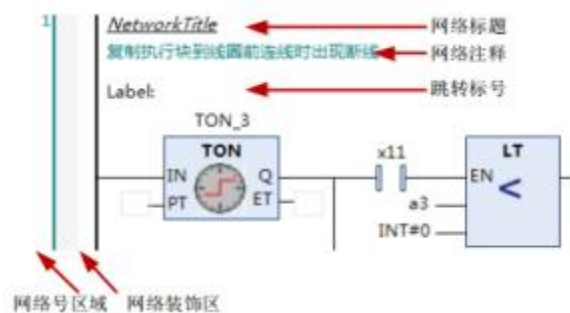
Icon -

A ladder diagram consists of a series of networks within which all other ladder elements are located. Each network is indicated by the network sequence number on the left. Networks can insert titles (summary descriptions of networks) and network notes (more detailed descriptions of networks) through the menu [Tools > Options > *FBD/LDeditor>conventional*] to enable the display of network title and network comment content.

Selecting a net inserts a label, located below the net title and net comment, as jump targets.

The network can be enabled or disabled through the menu command [Switch Network Annotation Status].

There is an area between the network serial number and the network content called the network decoration area, which is used to display the breakpoint mark and bookmark location.



2 contacts

Icon - 

Contacts are divided into normally open contacts and normally closed contacts. The contact transmits ON (TRUE) and OFF (FALSE) values, the contact is a BOOL variable, if the variable value is TRUE, the normally open contact transmits ON (TRUE) to the right, otherwise it transmits OFF (FALSE), the normally closed contact Passing the value is the opposite.

The contact can add the function of delay signal, select the right-click menu command [Edge Detection] of the contact, you can Turn the contact into a rising edge trigger contact (when the variable value of the contact changes from FALSE to TRUE, the contact will pass to the right) or a falling edge trigger contact (when the variable value changes from TRUE to FALSE, the contact will pass on to the right) .

3 coils

Icon - 

The coil is at the end of the network. The result of the logic operation on the left is assigned to the coil variable. Coil variables can only be BOOL type, TRUE means (ON), FALSE means (OFF). Coils only support up or down insertion of parallel coils.

Coils are divided into coils, inversion coils, set coils, and reset coils. You can switch between the 4 coil types through right-click menu commands or shortcut keys.

coil : The result of the logic operation on the left is directly assigned to the coil variable.

Invert coil: Invert the result of the logical operation on the left and assign it to the coil variable.

Set Coil: If the status value on the left side of the coil is ON(TRUE), then set the value of the coil variable to ON(TRUE), and a

remain in this state until the next time the variable is reset by the reset coil to OFF(false).

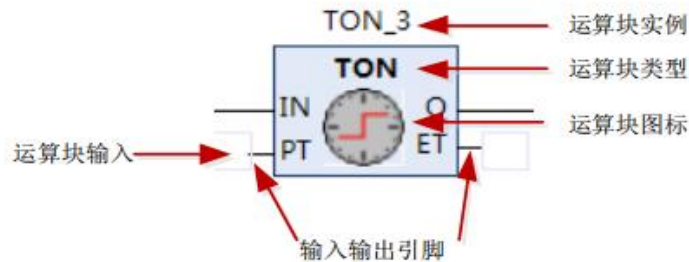
Reset Coil: Reset the set coil.

4 operation blocks

Icon - 

Operation blocks can be operators, functions, function blocks, programs, actions, and methods. If it is a function block type, an edit box will be added above the operation block box to display the function block instance.

An operation block contains at least one input and one output. The operation block mainly consists of the following figure:

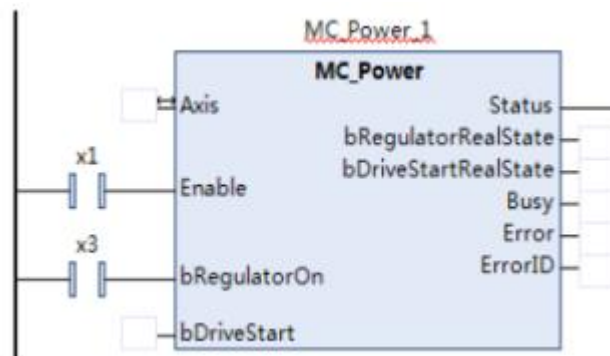


Operation blocks are divided into ordinary operation blocks and En/Eno operation blocks.

EN/ENOType operation block: In addition to including the input and output of the operation block itself, it also adds ENinput and ENOoutput. EN/ENOThe execution logic of the operation block is: when ENforTRUEWhen the operation block logic is executed, after the execution is completed ENOforTRUE, if ENforFALSE, do not execute the operation block, ENOforFALSE. Notice: EN/ENOOperate block input wires can only be connected in ENpin, output wiring can only be connected at ENO pin.

Operation block input and output pins: BOOLType input pins can add inversion, rising edge, and falling edge signals. The output connection pin of the operation block can add a negation signal.

Multi-input wiring operation block: As the name implies, there are multiple inputs and multiple inputs are connected to the energy flow line. The following figure shows a multi-input wiring operation block with two inputs connected. Since the multi-input wired operation block has multiple connections and power lines connected, the multi-input wired operation block can no longer be in parallel branches, and can only be in the first branch.



5 Execute block

Icon -

An execution block is a block in which an inline ST can be inserted, and an ST statement can be edited in the block. The execution block can be enlarged or reduced, the maximum is 1000*400.

6 branches

Icon -

The branches form a non-closed parallel logic.

7 Label

Icon -

The label indicates the jump position, which is located at the head of the network, and jumps to the label position through the jump element. The jump label is a string, and the naming rules for symbol identifiers are required.

8 Jump

Icon – 

When the input to the left of the jump element is TRUE, the jump to the specified label position is executed. The jump element is on the far right of the network.

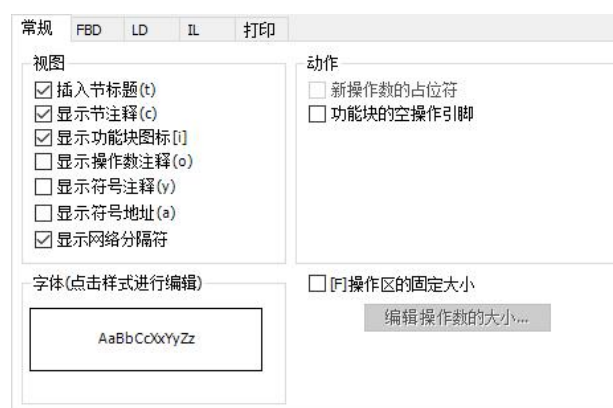
9 Back

Icon – 

When the input to the left of the return element is TRUE, the current program immediately exits execution. The returned element is at the far right of the network.

5.3.3.LD General Settings

The editor options are opened via the menu command [Tools > Options > FBD, LD and IL Editors]. It is divided into five tab settings: General/FBD/LD/IL/Print. This section mainly explains the general setting interface as shown in the figure:



view

Insert Network Title: If this option is activated, each network in the ladder diagram can insert and edit the title. If a title has been inserted, a row will be added to the top of the current network to display the title. If there is no title, the title row will not be displayed. Inserting a title is done via a menu command.

Show Net Comments: If this option is activated, the net comments can be edited for each

net in the ladder diagram. If a net comment is added, add a line under the net title to display the net comment. If the net comment does not exist, no blank line will be generated to display the net comment. Editing web annotations is done via menu commands.

Display function block icons-If this option is activated, if the operation block defines an icon, the operation An icon is displayed in the middle of the block. Standard operators (such as ADD, SUB) and function blocks such as TON, TOF) have defined icons, user-defined functions, function blocks or programs can add pictures by right-clicking [Object > Properties > Bitmap > Click here to select the bitmap related to the project] to form the operation block icon.

Show Operand Comments: If this option is activated, operand comments can be edited and displayed on each operand in the ladder interface. Operand is a programming concept, such as variables, constants, addresses are operands. Since variables such as constants or addresses are not necessarily used in ladder diagrams, they can be annotated through operand annotations. Editing operand comments is accomplished by selecting the operand string and then right-clicking the menu.

Display variable comment: If this option is activated, the variable declaration comment will be displayed on the variable in the ladder interface. Variable comments come from variable declarations and cannot be edited.

font

Click the sample text to pop up the editor font selection box to set the font of the ladder diagram text. The default font is Microsoft Yahei, which uses a small five-point font. The font range mainly includes operand characters and comments. Execute block fonts using text editor fonts (ST text, variable declaration text).

action

placeholder for new operator : not implemented

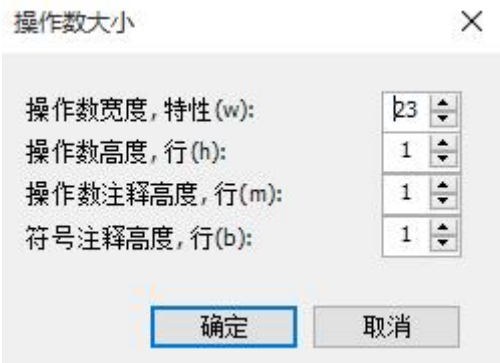
Default empty pins when adding an operation block: If this option is activated, when a new operation block is added, the input and output pins of the operation block use null characters,

If this option is not activated, operation block inputs and outputs citation“???” character.

Operand fixed size setting : If this option is activated, operand fixed width, operand annotation height and variable can be set

Annotation height.

As shown below:



Operand width : Set the number of fixed characters of the operand, the default is 23;
Operand Comment Height: Set the number of lines of fixed height of the operand comment,

default10K;


Variable annotation height : Set the variable comment fixed height line number, default10K;


5.3.4.LD menu commands

Menu commands Ladder diagram commands that can be executed through the right-click menu or the LD toolbar menu.

1 Plug in the network

Contains two menu commands: Insert Network Command and Insert Network (below). You can insert a network by dragging "Network" in the toolbox.


Insert Network: Icon- hot key:Ctrl + I, which means to insert an empty network above the selected network.

Insert Network (below): Icon- hot key:Ctrl + T, which means insert an empty net below the selected net.

Command execution conditions: first select a network, and insert a new network above/below the selected network.

2 Toggle network annotation status

In the commented state, the code of the entire network is invalid, the code will not be executed, and the execution block cannot be edited.


icon- hot key:Ctrl + O, which switches a network between the annotated state and the non-annotated state.

Command execution condition: select network


3 Insert network header information

The network header mainly contains the network title, network comments and labels.


The commands related to inserting net headers include inserting labels, editing net titles, and editing net notes.

Edit Network Title: Icon-, the title of the editor's choice network.

Command execution condition: select a network, and "display network title" is enabled in the options

Editing Web Notes: Icon-, to edit the annotation for the selected network.

Command execution condition: select a network, and enable "Display Network Comments" in the options

Insert Label: Icon-, insert a jump label into the selection network as the jump position of the jump element.

Command execution condition: select network

4 Insert an operation block


Including insert operation block, insert empty operation block, insert operation block

with EN/ENO, insert operation block with EN/ENO Empty Function Block and Insert Parallel Operation Block (below) 5 menu commands for inserting operators, functions, function blocks and programs. You can also drag the "operation block" or "with EN/ENO operation block" to insert the operation block.

The first four commands are used to insert concatenated operation blocks, and the last command is used to insert and select parallel operation blocks of elements.


Insert position:

- 1) Select the horizontal wire and insert an operation block at the horizontal wire.
- 2) Select the vertical wire (parallel brackets), and for the left bracket, insert it into the wire on the left side of the bracket. If it is a right bracket, insert it into the right side of the bracket.
- 3) Select an element and insert an operation block to the left of the selected element.

insert operation block: icon- hot key: Ctrl + B to pop up the input assistant to select an operation block to be inserted.

Insert empty operation block: icon- hot key: Ctrl + Shift + B, inserts an empty operation block without popping up the input assistant.


Operand block types can be entered at Operand Block Type.

insert tape EN/ENO operation block: icon- hot key: Ctrl + Shift + E, pop up the input assistant to select an operation block to be inserted, this operation block has EN/ENO input and output. bring EN/ENO operation block, when EN for TRUE the operation block is executed only when FALSE does not execute when EN0 and EN same result.

insert with EN/ENO The function block of : insert an empty operation block, the input assistant will not pop up, this operation block has EN/ENO input and output.

Insert Parallel Block (below): Inserts an empty block below the selected element. Selection elements can be contacts, operation blocks.


5 Insert execution block

Insert execution block: Icon - , insert a series execution block at the current selection position, or insert it by dragging the "execution block" in the toolbox.

- 1) Select the horizontal connection, and insert an execution block at the horizontal connection;
- 2) The choice is vertical connection (parallel bracket), for the left bracket, insert it to the left side of the bracket; for the right bracket, insert it to the right side of the bracket;
- 3) Select an element and insert an operation block to the left of the selected element.

The execution block is a block that can edit the ST statement, one-key text area to edit; the execution block only has EnEno input and output.

6 Insert input

Insert input: icon -  Shortcut key: Ctrl + Q, to add input to the variable input operation block.


Variable input arithmetic blocks: ADD, +, MUL, *, SEL, AND, &, OR, |, XOR, MAX, MIN, MUX.

Insertion position: When an input pin is selected, an input is added before the current input pin; when an operation block is selected, the added input pin is at the last position.

7 Insert the coil

Contains three menu commands: Insert Coil, Insert Set Coil and Insert Reset Coil. You can also insert coils by dragging the "Coil", "Set Coil" and "Reset Coil" elements in the toolbox.

Command execution conditions: The selection position cannot be located in the parallel branch, nor can it be located at the input position of the multi-input wiring operation block.

Insert Coil: Icon  hot key: Ctrl + Shift + A, output at the current position a coil

Insert position:

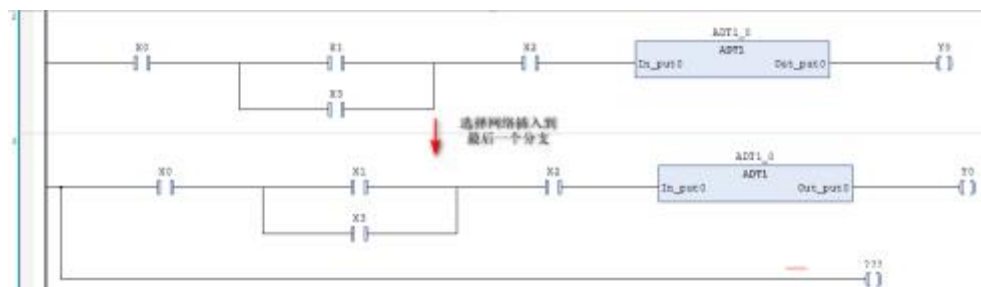
1) Select the horizontal connection, insert a coil at the horizontal connection, and the coil and connection are processed through non-closed branches.



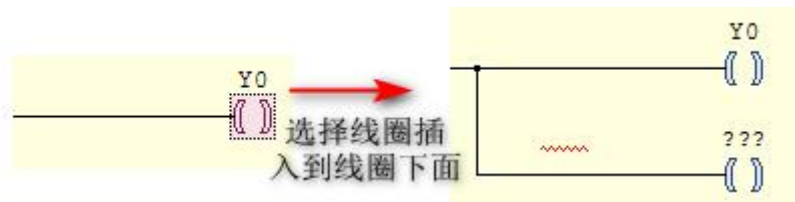
2) The choice is vertical connection (parallel bracket), for the left bracket, insert it to the left side of the bracket; if it is the right bracket, insert it to the right side of the bracket.





3) Select the network and insert the new coil at the end.



4) If a coil, return or jump is selected, a new coil is inserted below the currently selected element.




The default variable of the inserted coil is "???". You need to input the required variable or constant. You can use the input assistant to directly select the input from the variable list.

- ◆ Insert Set Coil: Icon - , which means to insert a set coil at the current position. Operation and the above "Insert Coil" is the same.
- ◆ Insert reset coil: Icon - , indicating that a reset coil is inserted at the current position. Operation and the above "Insert Coil" is the same.

8 Insert contacts




It includes four menu commands: Insert Contact, Insert Normally Closed Contact, Insert Parallel Lower Contact, and Insert Parallel Upper Contact. You can also insert contacts by dragging the "Contact" and "Normally Closed Contact" elements in the toolbox. point.

Insert Contact: Icon -  Shortcut key: Ctrl + K, which means to insert a normally open contact in series before the current position.

Insert position:

- 1) Select the horizontal connection line and insert a contact at the horizontal connection line;
- 2) Select the vertical connection (parallel bracket), for the left bracket, insert it into the left side of the bracket; if it is the right bracket, insert it into the bracket Right;
- 3) Select a network, then the new contacts are inserted at the end;
- 4) Select the element, and the new contact is inserted to the left of the element.


The contact default variable name is "???". Click on the text to enter the desired variable or constant, and you can use the input assistant to select the input directly from the variable list.

- ◆ Inserting a normally closed contact: Icon - , indicating that a normally closed contact is inserted in series at the current position. Operation and the above "Insert Contact" is the same.
- ◆ Insert parallel lower contact: Icon - , shortcut key: Ctrl + R, which means to insert a constant in parallel under the selected element Open contacts. The selection element can be a contact or an operation block.
- ◆ Inserting Parallel Upper Contacts: Icon - , shortcut key: Ctrl + P, which means to insert a parallel on the selected element Normally open contact. The operation method is the same as the above "inserting

the lower contact in parallel”.

9 Insert branch

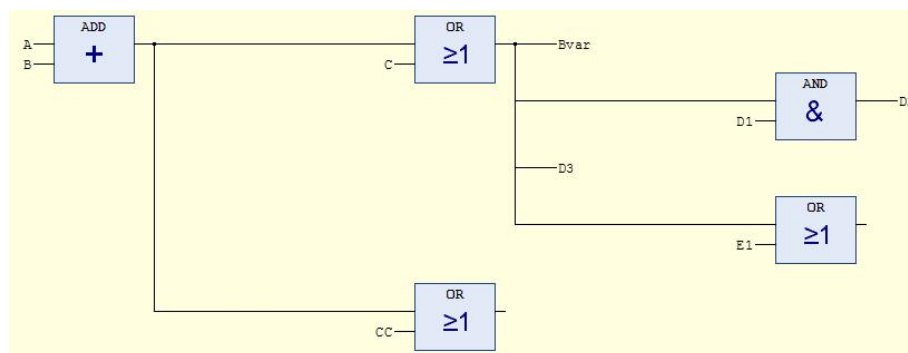
It includes three menu commands: insert branch, insert branch above, insert branch below, or insert branch by dragging the “Branch” element in the toolbox. A branch is a non-closed line, unlike a parallel branch.



- ◆ Insert branch: icon , shortcut key: Ctrl + Shift + V, which means to insert a branch at the selected connection position.
- ◆ Command execution conditions: The selection position cannot be located in the parallel branch, nor can it be located at the input position of the multi-input wiring operation block.

Insert position:

- 1) Select the wire and insert a branch below the wire.
- 2) Select contacts or coils to insert before selecting elements.
- 3) Select the left bracket and insert it on the left side of the bracket; select the right bracket and insert it on the right side of the bracket.

As shown in the figure below, each selection position represents a branch.




- ◆ Insert branch below: icon , which means to add a branch below the selected branch.
- ◆ Command execution condition: select branch connection.
- ◆ Insert branch above: icon , which means to add a branch above the selected branch. Only when branch connection is selected can execute.
- ◆ Command execution condition: select branch connection.

10 Jump and Return


Contains two menu commands: insert jump command and insert return. Jump and return belong to the program execution sequence control commands. The normal program is executed sequentially from top to bottom and from left to right according to the network sequence. You can insert a jump element by dragging “Jump” in the toolbox or “return” in the toolbox to insert a return element.

Jump and return are the same as coils and must be on the far right, so the rules for inserting


jumps and inserting returns are the same as those for inserting coils. For details, please refer to the inserting coil commands.

◆ Insert Jump: Icon -  Shortcut key: Ctrl + L, which means to insert a jump element and jump to the specified label position.

The jump location is the label in the network, that is to say, you can jump from one network to another network. The jump can only be executed when the input conditions before the jump are satisfied.

◆ insert returns: icon - , which means inserting a return element. When the input conditions are met, the current POU executes the return back, back to the POU that called it

11 Negate

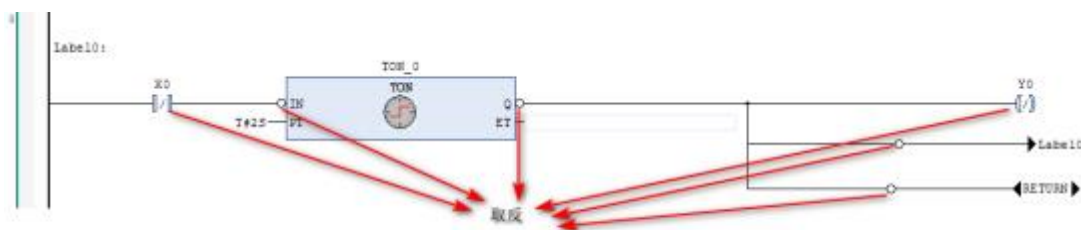
Icon -  Shortcut key: Ctrl + N, negate operation block input, operation block output, jump condition, return condition, contact value or coil.

The negation command can be executed in two places:

1) Inversion of elements: mainly contacts and coils. A slash (/) is added to the contacts and coils after negation.


2) Connection inversion: mainly includes operation block input connection, operation block output connection, coil input connection, jump input connection, and return input connection. After inversion, a circle is added to the connection.

The inverse position is as shown below:





When the negation command is executed again, the negated state switches back.

12 Edge Detection



Icon -  Shortcut key: Ctrl + E, which means to add edge trigger function to contact, operation block input connection, coil input connection, jump element input connection, and return element input connection.

The rising edge detection is equivalent to the R_TRIG function block, and the falling edge is equivalent to the F_TRIG function block.


Edge detect commands can be executed in two locations:

1) Contact edge detection: Select the contact to execute the edge detection command, and the contact adds the edge detection function.  Indicates rising edge; 



Indicates a falling edge.

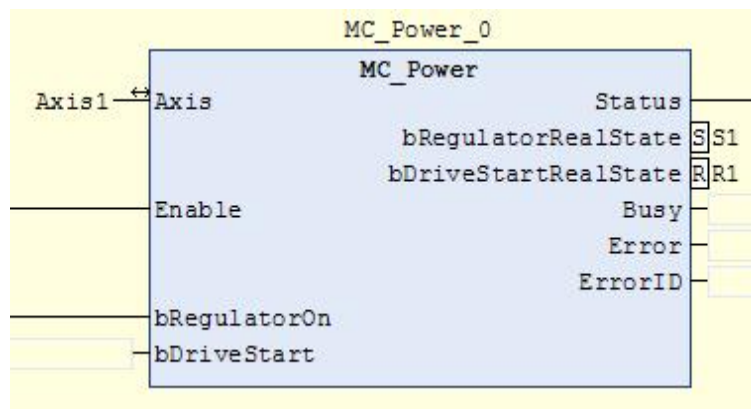
- 2) Added edge detection for connections: operation block input connection, coil input connection, jump element input connection, return element input
Connect the line, execute the block edge detection command, add the extension signal symbol to the connection line, and the rising edge detection symbol is ; the falling edge detection symbol is . Only BOOL type input wiring can add edge detection function.

13 Set/Reset


Icon -  Shortcut key: Ctrl + M, this command is used to increase the set or reset output function. The set output is displayed as "S" and the reset output is displayed as "R". This command can be executed multiple times to toggle between set, reset and normal output.

Set/Reset commands can be executed in two locations:

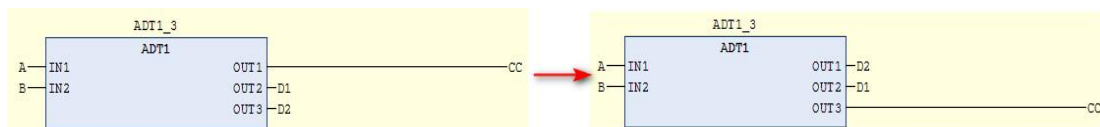
- 1) Select the coil, execute this command to set and reset the coil. Set coil:  .
Reset coil:  .
- 2) Select the BOOL output connection of the operation block (not the main output), and set the reset function, as shown in the figure below:



14 Set output connections


Icon -  Shortcut key: Ctrl + W, when the operation block has multiple outputs, the main output can be modified output pin (an operation block has only one main output, and the main output is connected to the subsequent elements). As shown below:

Select the output pin to be modified, execute this command, modify the output connection




15 Change the display of input and output pins

Contains two menu commands: Update Parameters and Delete Unused FB Call Parameters.

Update parameters: icon -  Shortcut key: Ctrl + U, which means to update the output

of the selected operation block input and output parameters. If the input or output parameters of the operation block change, by executing the "Update Parameters" command, update the input and output parameters of the operation block

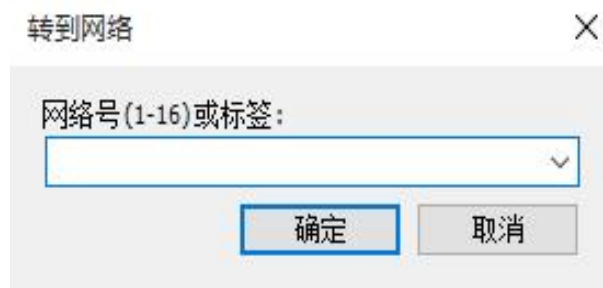
Delete unused FB call parameters: Icon - , delete the unused input pins and output pins of the operation block, that is, when the input or output of an operation block is "???" or empty, these inputs and outputs will no longer be displayed.

16 Convert to LD language

Displayed as Ladder Logic: Shortcut: Ctrl + 2. Convert FBD/IL to LD language; since FBD and IL are no longer supported temporarily, old projects can use this command to convert FBD/IL to LD language for display.

17 Jump Network

Go to...: Jump to the specified network. The jump network input box pops up, and specify the jump network number.



18 Editing Operand Comments

Edit Operand Comment: Edit the comment of the selected operand.

Command execution conditions:

- ◆ In the options FBD/LD, activate the option "Show operand comments"
- ◆ Requires selection of operand string

Operands are logical concepts. Input variables, constants, and addresses are all operands, such as operation block input variables, contact-related variables, coil-related variables, and operation block instances.

Select the operand string, execute this command to display the Edit Operand Comment dialog box, and edit the operand comment, as shown in the following figure:

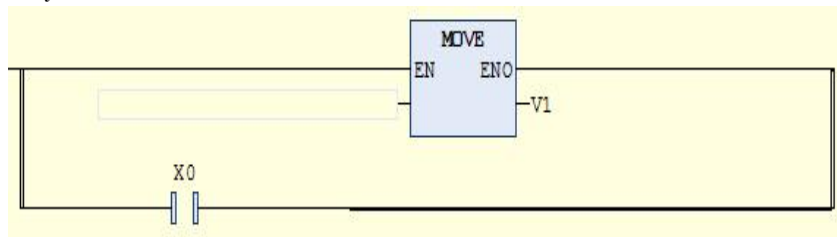


19 Parallel mode switching

Toggle Parallel Mode: Toggle parallel branch parallel mode. The parallel mode is divided

into sequential type parallel branch and short-circuit type parallel branch.

◆ The sequential parallel parallel support uses a single line, and the branch output result is a single branch output OR operation, as shown in the figure below, the branch result is formed by OR.

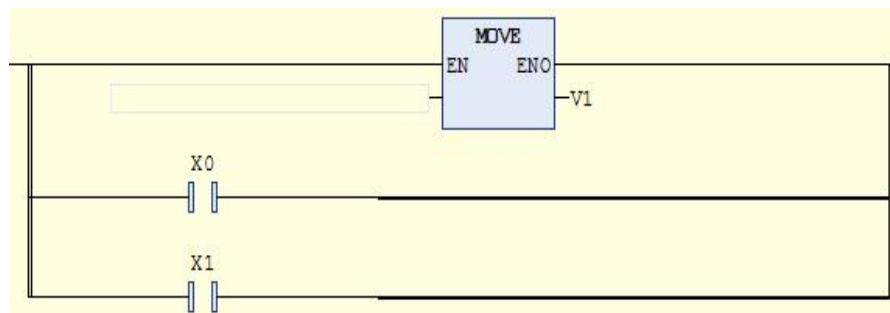


◆ The parallel support of short-circuit parallel connection uses two wires, and the output result of the branch needs to consider whether each branch contains a non-operation block.

The branch of the non-operation block is used as a condition. If the branch of the non-operation block has a result of True, the branch with the operation block will not be executed (it can be understood as a contact short-circuit operation block). As shown in the figure below, only the result of the X1 branch and the X2 branch is not TRUE, the first Move branch instruction is executed.

Non-operation block branches need to meet the following conditions at the same time:


- 1) This branch contains only contacts or operator blocks.
- 2) Contacts cannot have delay signals
- 3) The operator operation block cannot be of type EnEno, and the input wiring of the operator operation block cannot contain negation or delay signals
- 4)





5.3.5. Drag and drop operation


Ladder diagrams can drag and drop elements, including dragging elements from the toolbox to the network, dragging elements in the ladder diagram interface, and dragging across interfaces.

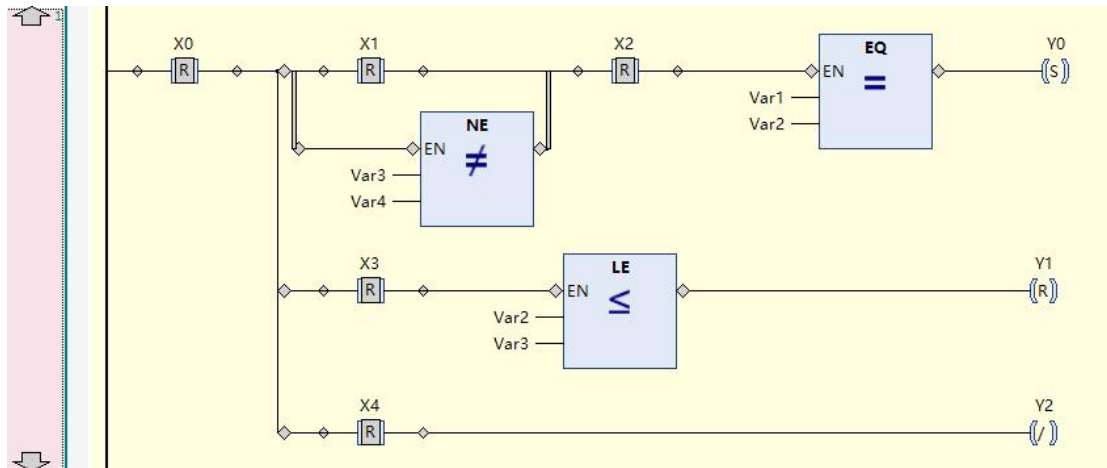
When dragging an element, the ladder interface will display the draggable position. The draggable position has three display forms:

➤Diamond Display: Use a diamond , indicating that it can be dragged to the current position to insert in series.

➤ Up and down triangle display , which means to insert a parallel element above or below the current element.

➤ Up and down arrow display: use the up and down arrows , which means adding a network up or down.

When dragging the element to the insertion position, the inside of each graphic will turn green, such as , indicating that you want to insert at this position. The drag and drop display is shown below.



1 Element box tool drag and drop

Elements in the toolbox can be dragged into the ladder editor.

Toolbox elements mainly include common elements, ladder elements, common BOOL operators, math operators, other common operators, common function blocks and POUs.

Conventional and ladder elements are commonly used elements, including networks, operation blocks, execution blocks, contacts, coils, branches, and more.

Boolean operators are mainly AND and OR operators.

Mathematical operators are mainly commonly used mathematical operators such as ADD, SUB, MUL, DIV, GE, EQ, LE, and LT.

Other operators are mainly two-choice, multiple-choice, type conversion, and assignment commonly used operators.

The function blocks mainly include commonly used function blocks such as TON, TOF, R_TRIG, F_TRIG, and RS.

POUS mainly includes programs, function blocks, functions, methods, and actions defined in the current project. The maximum display cannot exceed 200. If there are more than 200 in the project, in order to prevent the display from being confused, the POUS content will no longer be displayed.

When dragging, each element has a draggable position. The draggable rules are as follows:

➤ Contacts can be dragged to contacts, operation blocks (including execution blocks) in parallel, and dragged to connections in series;

➤ Operation blocks can be dragged to contacts, operation blocks (including execution blocks) in parallel, and dragged to connections in series;

➤ The coil can be dragged to the non-parallel branch, non-multi-connection operation block input connection line, and can be dragged to the coil, return, jump above or below;

2 Edit interface element drag and drop

In the ladder diagram interface, you can drag and drop the selected element from one position to another. Drag and drop elements can be in this editing interface, or drag and drop to other ladder diagram editing interfaces.

The dragged element is the selected element (see the section on element selection), which can be multi-selected or single-selected.

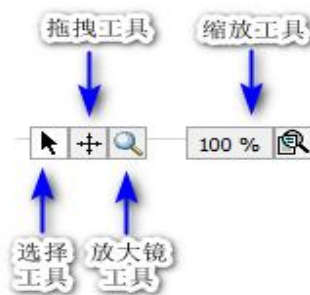
Drag includes normal drag and copy drag (press Ctrl, drag). For normal drag and drop, after the selected element is dragged over, the selected element will be deleted; for copy-drag, after the selected element is dragged over, the selected element is retained.

The drag and drop function is implemented according to the standard operation method.

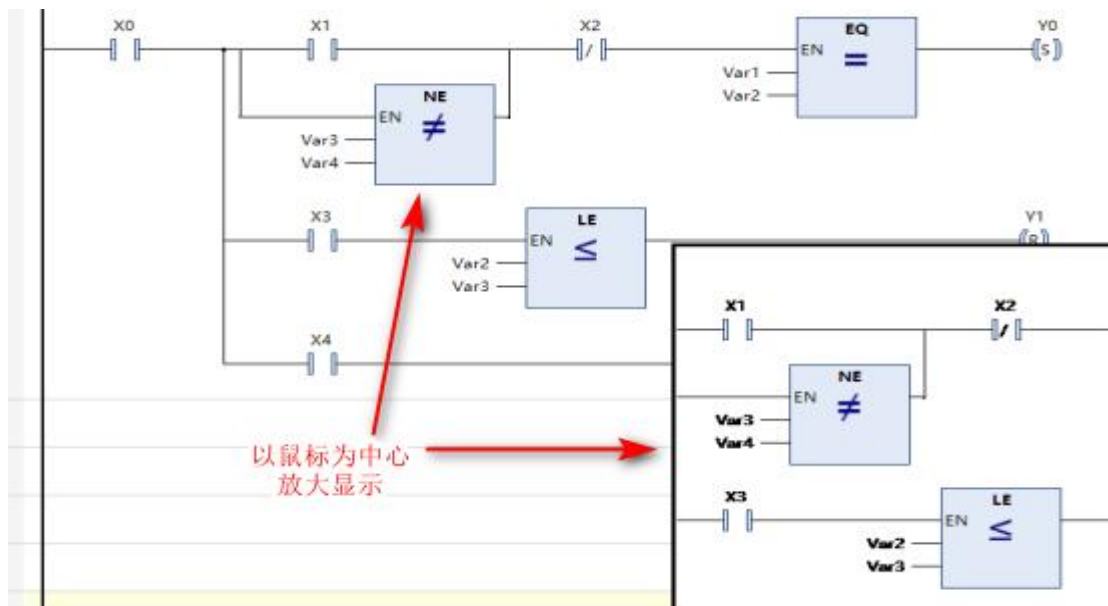
The drag and drop rules for multiple selection or single selection are consistent with the paste of standard editing commands

5.3.6. Graphic Display Tool

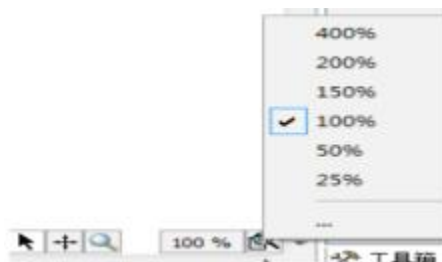
The ladder diagram graphics display tool is used to control the ladder diagram display mode, including selection tool, drag and drop tool, magnifying glass tool and zoom tool. The default ladder diagram is the selection tool mode. The graphic display tool is located on the lower right side of the ladder interface, as shown in the figure below:



- ◆ Selection tool, the selection tool is the default display tool, in the selection tool mode, the mouse style is, you can select the element element to perform editing operations.
- ◆ Drag tool, in drag tool mode, you can drag and drop the area to display
- ◆ The magnifying glass tool, in the magnifying glass mode, the mouse style is, with the mouse as the center, to zoom in and display, there is a magnifying glass as the center. use. As shown below:



◆ Zoom tool, the zoom tool can display the current interface zoom ratio, or set the zoom ratio, as shown in the following figure:



In addition, click "...", the zoom ratio setting dialog box will pop up, enter the desired zoom ratio, as shown in the following figure:

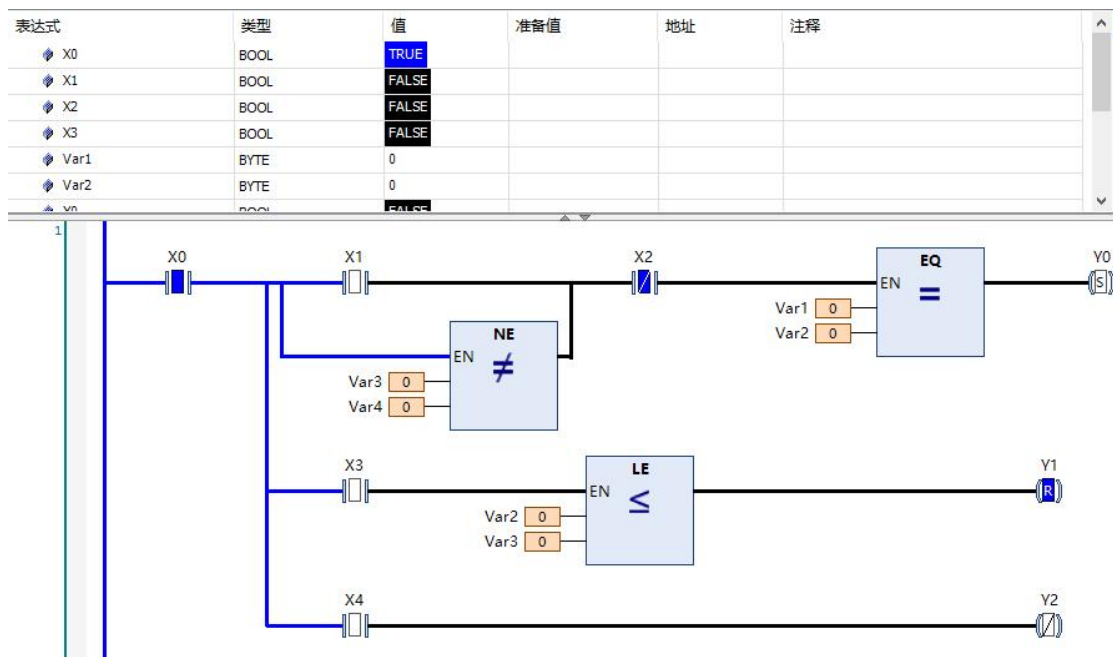


5.3.7.LD debugging

The ladder diagram provides powerful debugging functions. In addition to the existing monitoring table monitoring, the ladder diagram also provides monitoring, operand writing, forced value writing, breakpoint and single-step debugging functions in online mode.

1 Monitoring

In the online mode, the connection lines, elements, operand variables, etc. in the ladder diagram interface express the execution result in a specific form. As shown below:



1) Monitor the connection

➤ For BOOL type value connection, when it is turned on (the value is TRUE), it will display a thick blue line, and when it is not connected, it will display a thick black line.

➤ Non-BOOL values are wired (operation block input, integer variables in output, time type variables, floating point variables, etc.), use a thin line, and use a thin black line when the value is zero; use a thin blue line when not zero.

2) Monitoring elements

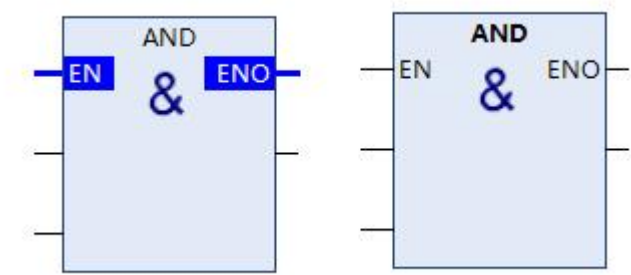
➤ When the contact is on, the normally open contact displays or normally closed contact display ; When the contact is not conducting, the normally open contact displays or

normally closed contact display .

➤ When the coil is turned on, the normal coil displays Or negate the coil display ; When the coil is not conducting, the normal coil displays Or negate the coil display .

➤ For the EnEno operation block, since the EnEno operation block only has En is True, the logic of the operation block itself is executed.

The EnEno operation block itself can know at a glance whether the operation block is executed (whether the operation block is enabled), and distinguishes the text display of the operation block type. If the operation block is executed (En input is True), the operation block type is displayed in black font. No execution is displayed in gray font (operation block is disabled), as shown below:



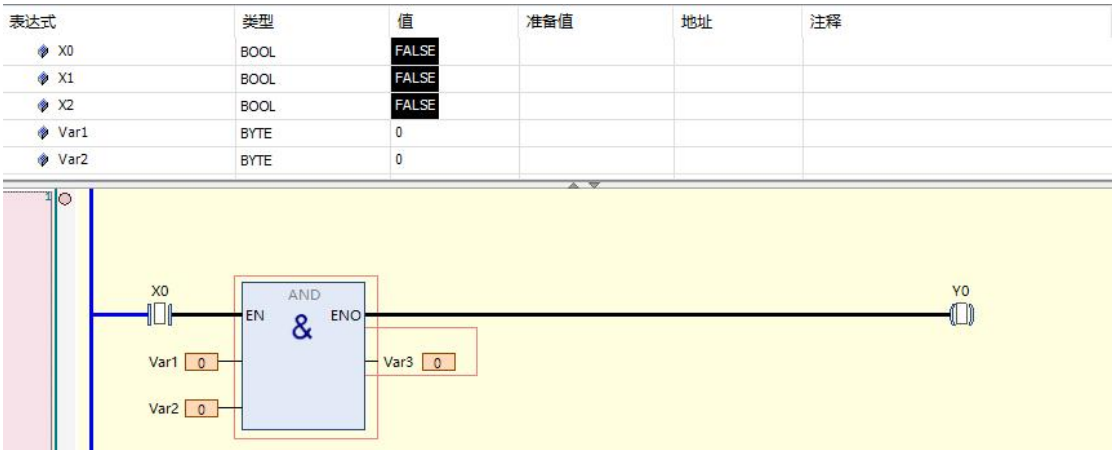
3) Monitor variables

- Monitor variables display different widths according to different types to reduce the space occupied. For variable lengths, such as strings and enumerations Type (display enumeration name), the default length is 12 characters, if the display is not complete, use ... to replace it, and display it completely through the information prompt; for fixed-length, such as integers, floating-point numbers, etc., display according to the maximum length.

➤ You can drag and drop monitoring variables to the monitoring variable list.

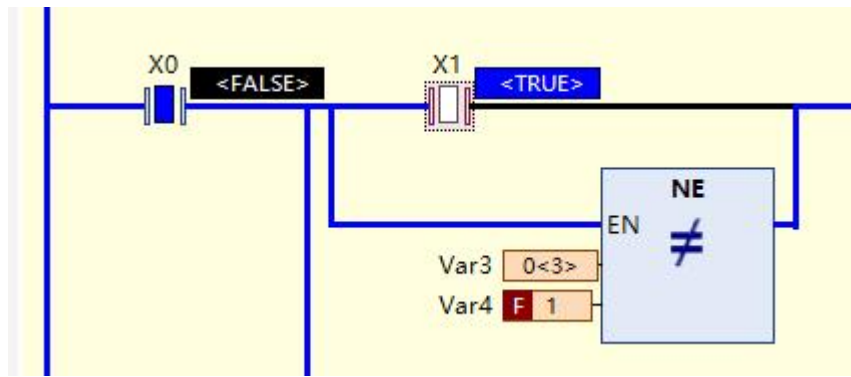
➤ You can change the variable display mode and execute the menu command: menu **【Debug>display mode】**.

Since functions and methods are executed immediately and only have temporary data, after logging in, methods and functions cannot be directly monitored. If you need to monitor functions and methods, you need to add breakpoints to the functions and methods to interrupt execution, as shown in the following figure.



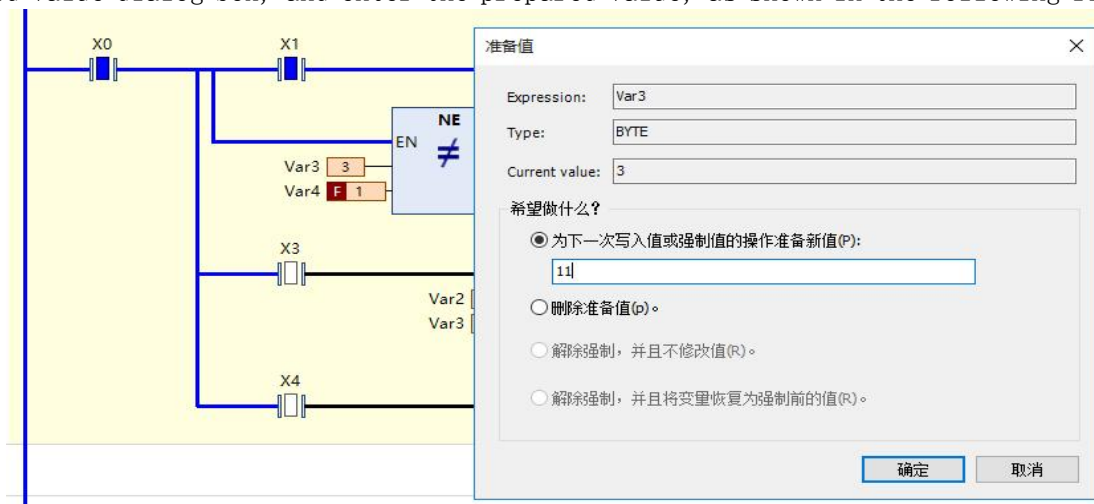
2 Write and force

Ladder contacts, coils and variables can be written to prepare values, and then execute the "Write Value" command or "Force Value" command under the Debug menu to write a value or force value to the variable. Before writing the value or forcing the value, you need to write the prepared value, as shown in the following figure:



➤ For contact, coil and BOOL type variables, switch between TRUE and FALSE ready values by double-clicking the element position or the variable value position. If you double-click the contact or the middle position of the coil, the ready value is switched.

➤ For non-BOOL type variables, double-click the variable value position to pop up the prepared value dialog box, and enter the prepared value, as shown in the following figure:



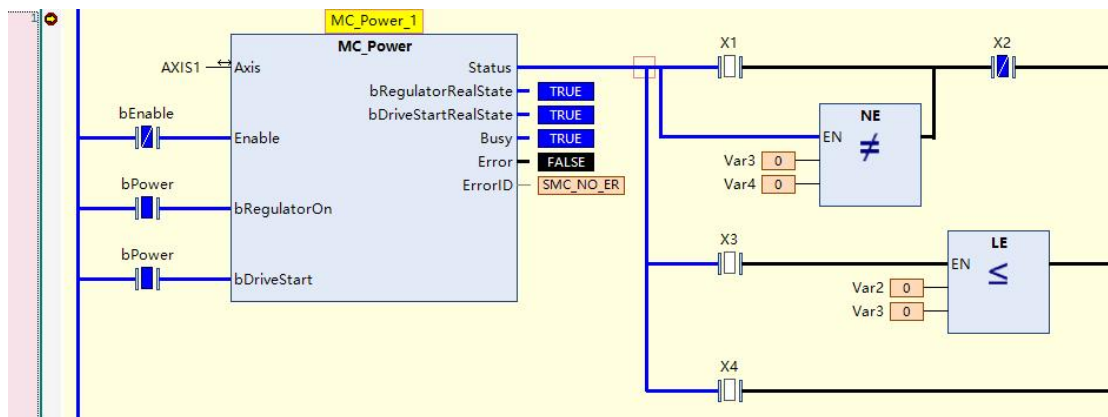
◆ After writing the forced value, the front of the value increases **F** Identifier that this value is mandatory.

◆ To release the forced value, use the menu command **【Online>Release Value】**.

3 breakpoints

LD supports breakpoint function. If a breakpoint is added, the program execution will be automatically interrupted at the breakpoint position, and program debugging can be performed. It supports operations such as jumping in, skipping, jumping out, and running to the cursor position.

After adding a breakpoint, the position (element) of the added breakpoint is represented by a light red rectangle. When the execution reaches the breakpoint, the position of the currently executing breakpoint is represented by a yellow rectangle; if there is a breakpoint in the network, the dots in the network decoration area, as shown below:



Since the ladder diagram is graphical, and breakpoints can only be added where there are logic statements, in order to optimize the performance of the ladder diagram, not all places have logic statements, that is, not all places can add breakpoints, such as touch Breakpoints cannot be added at point positions, non-EnEno operator block positions.

Breakpoints are generally located at places where variable values may change, at program branches, or at places where another POU is called, such as POU, output variable assignment, and so on. You can open the Breakpoints dialog (menu [View>Breakpoints]) to view all possible breakpoint locations.

Breakpoints can mainly be added to the following locations:

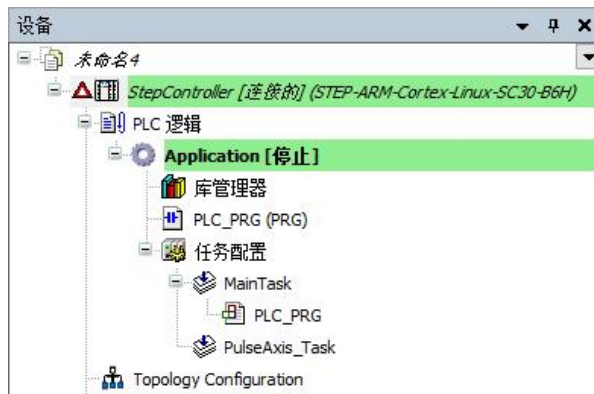
- The starting position of the network, indicating the first possible breakpoint position in the network. When a breakpoint is added to the network, it will automatically increase to the first one.
breakpoint location.
- The operation blocks of non-EnEno operators, such as FBs, actions, program calls, execution blocks, etc., are not included.
- Coil, return, jump element position.

第六章 Debug and Diagnostics

6.1. run/stop

6.1.1. Running and Stopping the Controller

1. After logging in, select Debug→Start in the menu bar, or press <F5>. Start the operation of the downloaded application in the controller.



2. Select Debug in the menu bar→**stop**, or press <Shift>+<F8>. Application stopped.

表达式	类型	值
CurrentTime	TIME	T#0ms
R1	BOOL	FALSE
R2	BOOL	FALSE
X1	BOOL	FALSE
X2	BOOL	FALSE
TON_0	TON	FALSE
X3	BOOL	FALSE

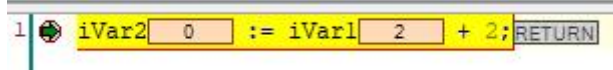
3. During the debugging operation, the current value of the variable can be confirmed in the declaration part and the implementation part.

- ❶ Click on the toolbar start Stop running.
- ❷ available from 2system, 10system, 16Select the display format of the variable value displayed in the decimal system. Select the display form from Debug→Display Mode on the menu bar.
- ❸ If using safe online mode, a confirmation message can be displayed before performing a run, stop.

6.1.2. single cycle

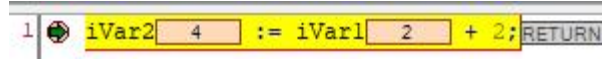
The application can be executed for 1 loop in simulation mode to confirm that the created program performs as expected.

1) Open P after loginOU.



2) Select the menu bar *Debug* → *Single Cycle*, or press <Ctrl+F5>.

The opened POU is in the state of executing one cycle.



6.2. breakpoint

By setting breakpoints at specific locations in the program, execution can be forcibly stopped and variable values can be confirmed.

All programming languages support breakpoints.

6.2.1. breakpoint settings

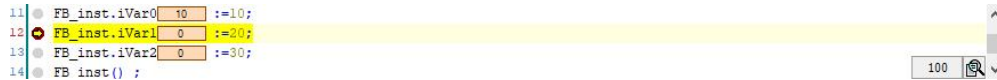
1. Select the location where you want to set the breakpoint, then select Debug→Set or Clear Breakpoint in the menu bar, or press <F9>.

Breakpoints are set to enabled.

Example: When a breakpoint is set on line 12 in ST



After starting the operation, it will stop when the breakpoint is reached.



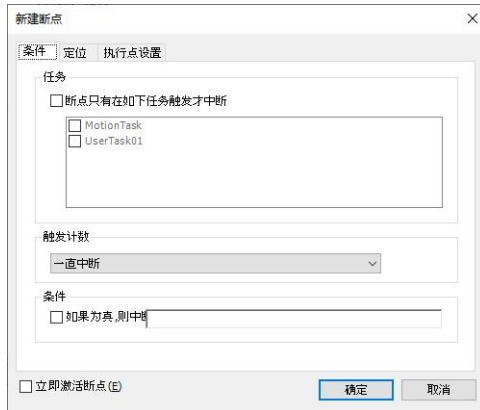
The following debug operations can be performed from the stopped state.

Select the following menu under "Debug" in the menu bar.

menu	hot key	toolbar icon
jump over	<F10>	
jump in	<F8>	
jump out	<Shift>+<F10>	
run to cursor	none	
set next statement	none	
show current statement	none	

To cancel the set breakpoint, select Debug in the menu bar again→*set up* or clear the breakpoint, or press <F9>.

① You can set a condition to stop the action at a breakpoint. Select Debug in the menu bar>Create a new breakpoint, which will show "new breakpoint" dialog. choose "condition" tab, enter the condition to stop at the breakpoint.



The list of breakpoint settings can be confirmed in the breakpoint view. The position of the breakpoint, the break condition, and the number of times reached can be confirmed. Breakpoints can also be added, removed, enabled and disabled.



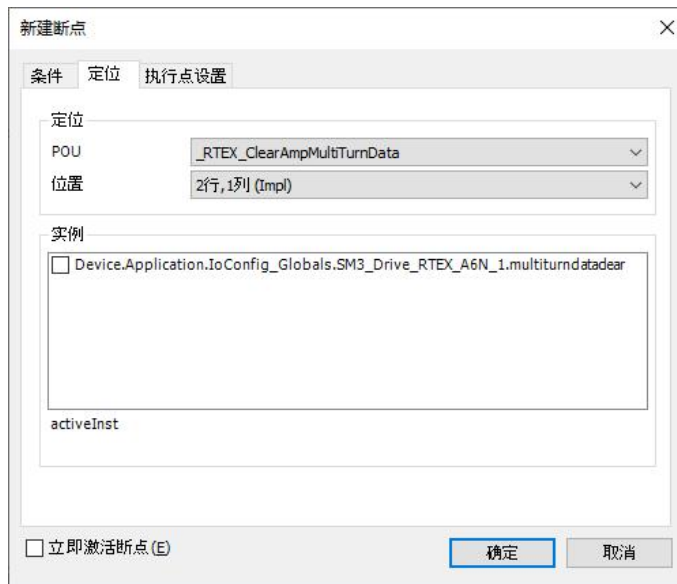
To display the breakpoint view, select View in the menu bar→**breakpoint**.

In the process of debugging the program, you can also set a breakpoint there by double-clicking the first column of the program, and double-clicking again can clear the breakpoint. Double-clicking anywhere other than the first column has no effect on setting or clearing breakpoints.

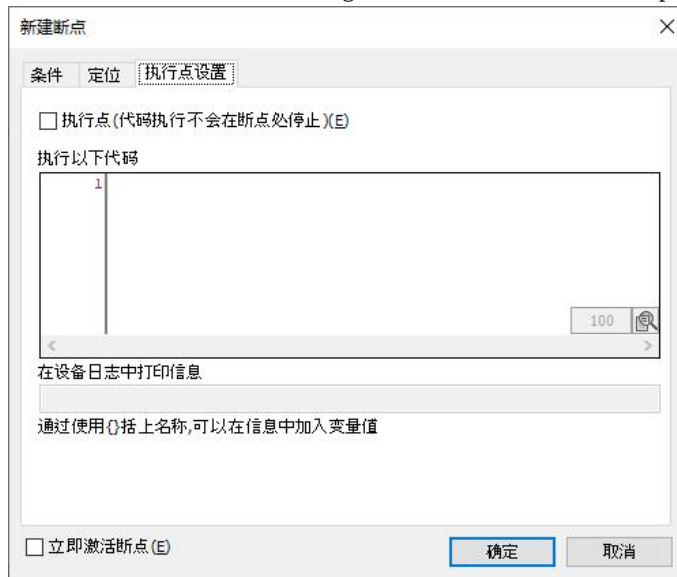
6.2.2. execution point setting

After the execution point is set, the pre-specified processing can be executed when the set position of the execution point is reached, and output to the log of the SC series controller. The application does not stop at the set position of the execution point.

1. Select the location where you want to set the execution point, and then select Debug in the menu bar→**new breakpoint**. Displays the New Breakpoint dialog box.



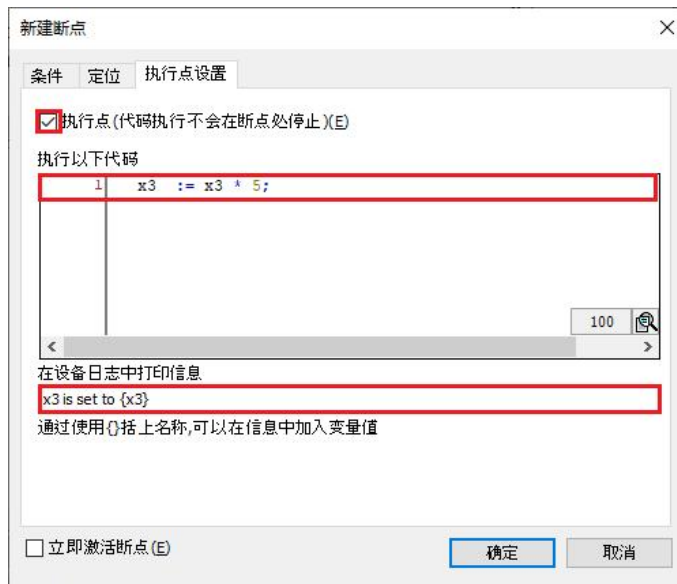
2. Click the "Execution Point Settings" tab. The execution point setting screen is displayed.



3. Check "Execution Point", enter the code to be executed at the execution point, and the message output to the log.

Use the ST program to describe the execution code in the "Execute the following code" column, and enter the message to be output to the log in the "Print information in the device log" column.

Example: When multiplying the value of "x3" by 5 and outputting the value to the log



4. Click the [OK] button.

An execution point has been set. When the execution point is enabled, it will be displayed on the execution point.

①To output messages to the log at the execution point, select Project in the menu bar → **Project settings**, exist "Project settings" dialog box select "compile options" category. Enable settings → **Enable logging in breakpoints** setting.

6.2.3. call stack view

In the call stack view, you can confirm the stop position when entering the stop state due to breakpoints, etc. The position can also be confirmed if it is called from another block.

1. Select View > Call Stack in the menu bar.

Displays the call stack view.



2. Set breakpoints to stop running the application.

Displays the POU that calls the stop position and the POU of the stop position.

Example: Stop at line 1 of "ADD_3" application, "ST_POU" calls "ADD_3"



6.3. debug operation

This section describes the procedure for debugging operations such as writing and monitoring of values.

6.3.1. Writing of Values and Forcing of Values

The variable values of the controller can be changed. Value change methods include value write and value coercion.

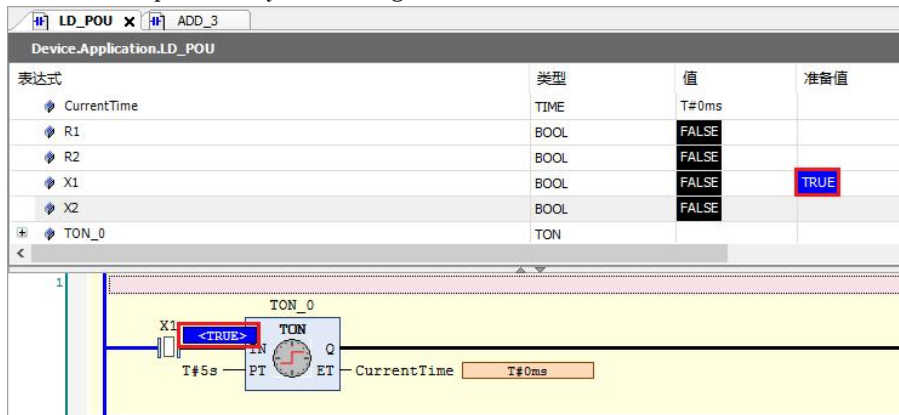
Writing of value: Set the value you want to change only once. The value can then be changed by the program.

Forcing of values: Set the value you want to change every cycle, and keep that value.

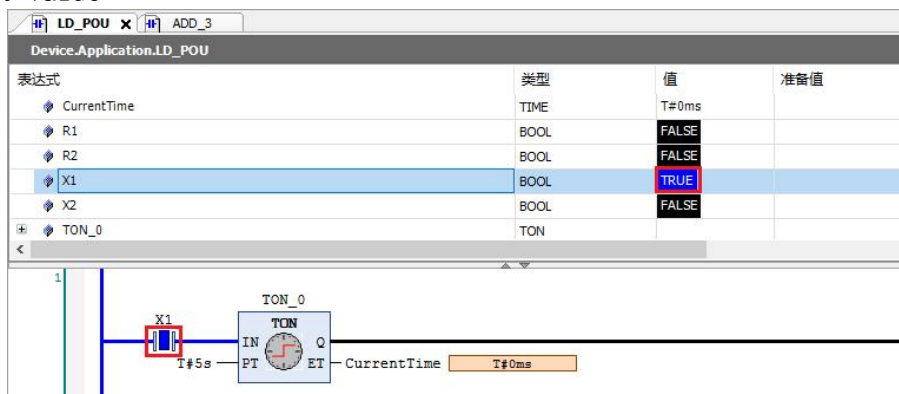
For example, to change the value of the BOOL type variable "x1" from FALSE to TRUE by writing the value, follow the steps below.

1. Double-click the element whose value you want to change in the implementation section. Change the preset to the changed value.

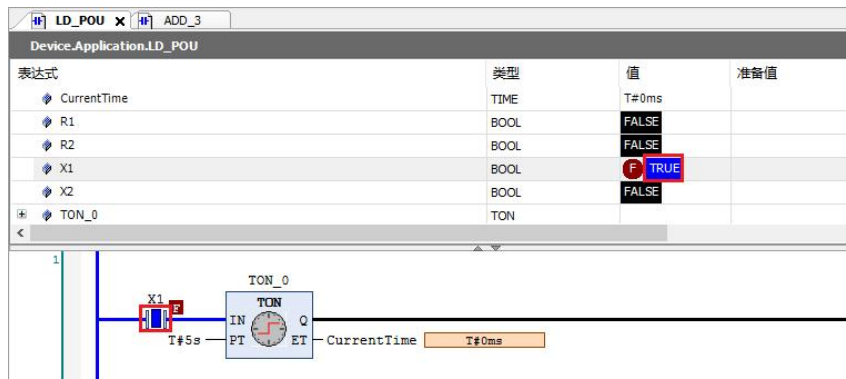
Values can also be preset by clicking on the "Set Values" cell in the Declarations section.



2. Select Debug → Write Value in the menu bar, or press <Ctrl> key + <F7> key. write preset value



Select the menu bar *Debug* → *Force Values*, or pressing the <F7> key will force the variable value to change. The value of the variable changed by force of the value will be displayed F, the value will not be updated by the executor after that.



Selecting Debug→Release Value in the menu bar, or pressing <Alt> key + <F7> key, will release the forced value.

6.3.2. monitor

You can manage variables such as registering variables, checking variable values, and changing values in the monitor view.

Up to 4 monitor views (Monitor 1 to Monitor 4) can be used.

For example, to register the variable "x1" in the watch view of watch 1, follow these steps.

1) Select View→Monitor→Monitor 1 in the menu bar.

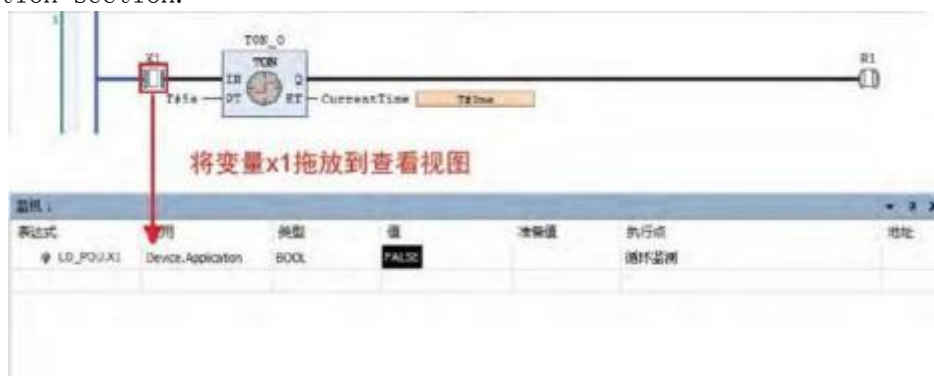
Displays the monitor view of monitor 1.



2) Drag and drop the element of the variable "X1" of the implementation section to the monitoring view.

The variable "x1" is already registered in the watch view.

Variables can also be registered to the watch view by dragging and dropping variables from the declaration section.



At this point, the step of registering the variable to the monitor view is complete. The variable value can be confirmed in the value cell.

- exist "Monitor all mandatory" In the view, variables that perform value coercion are automatically registered.

Select Display→Monitor→Monitor All Forced from the menu bar.

- If an execution point is set, the timing displayed on the monitoring view can be set

to the time when the execution point is reached. exist"execution point"Select the set execution point in the column of .

6.3.3. Process control

Flow control can distinguish executed and unexecuted parts of the program by color and monitor. Flow control can be used for LD programs, ST programs, and FBD programs.

1) Once logged in, select Debug→Toggle Flow Control Mode in the menu bar.

Displays the notification dialog for flow control.

2) Click the [OK] button.

Switch to the display of flow control.

The executed part is shown in green and the unexecuted part is shown in white.

Example: Flow Control Display in LD Program



Example: Flow Control Display in ST Program

```

● X1 2 := 2;
● CASE (X1 2) OF
  1:
  ● X2 0 := 2;
  2:
  ● X3 3 := 3;
  3:
  ● X4 0 := 4;
● END_CASE RETURN

```

① If using secure online mode, you can display a dialog box with a confirmation message before executing flow control.

6.3.4. operating mode

By using the Operational Mode feature, some debugging operations can be restricted from being performed. As a result, it is possible to prevent the controller from malfunctioning in the event of an incorrect operation.

The current operating mode is displayed by an icon in the status bar.

a) debugging(🔧)

Unlimited.

b) locked(🔒)

Cannot perform run/stop, set new breakpoints, force variables. Single loop, variable writing, and value forcing can be executed.

c) operational (🔒)

Only variable writing can be performed. Run/Stop, setting of new breakpoints, forcing of variables, single loop, and forcing of variables cannot be performed. To set this mode, the following conditions must be met.

- application is running
- no active breakpoints
- no mandatory variables
- The application of STEP AS is identical to the boot application of

the controller

After logging in, select Online→Work Mode→Locked in the menu bar.

The operating mode is changed from debug mode to locked mode.



6.4. Monitoring function

When logging in to the controller, the current values of program variables and device parameters can be checked in real time (monitoring).

6.4.1. Monitoring variables in the declaration editor

Variables declared in the declaration editor can be monitored.

The front of the value that was changed by force of the value will be displayed F.

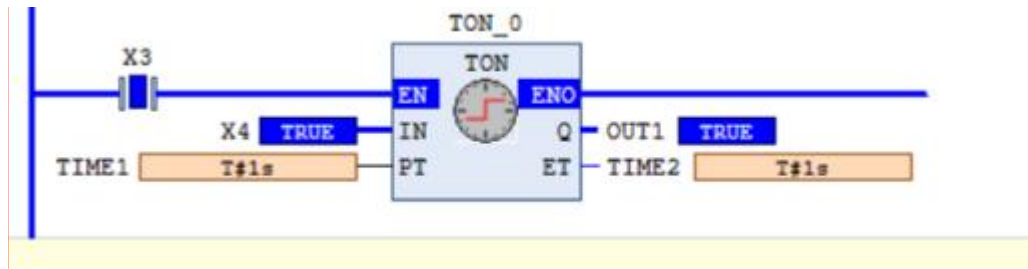
表达式	类型	值	准备值
🔗 CurrentTime	TIME	T#0ms	
🔗 R1	BOOL	TRUE	
🔗 R2	BOOL	FALSE	
🔗 X1	BOOL	F TRUE	
🔗 X2	BOOL	FALSE	TRUE

当前值

6.4.2. Monitoring variables in the implementation part of the program

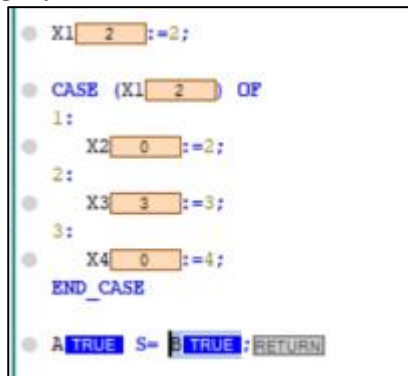
- Variables can be monitored in the implementation part of the program (inline monitoring).
- next to the variable will be like True, generally displays the current value.
- Contacts, coils, and connecting wires are displayed in blue when the current value is TRUE.

<Inline monitoring of LD programs>



<STInline monitoring of programs >

Inline monitoring can be disabled. Startup Options (Tools → Options), in "text editor" category "monitor" Cancel in tab "Enable online monitoring" tick.



6.4.3. Monitoring variables in the watch view

Variables can be registered and monitored in the watch view.

Up to 4 monitoring views can be used, and dedicated views that automatically register variables that perform value coercion.

表达式	应用	类型	值	准备值	执行点	地址
LD_POLUX1	Device.Application	BOOL	TRUE		循环监测	
LD_POLUX2	Device.Application	BOOL	TRUE		循环监测	
LD_POLUCur...	Device.Application	TIME	T#0ms		循环监测	

当前值

Reset device "Device" using hard switch	×	×	×	×	×	×	×	○	○
---	---	---	---	---	---	---	---	---	---

6.5.1. Warm reset/cold reset/reset (PLC initialization)

Select and execute warm reset, cold reset, reset (PLC initialization) from "Online" in the menu bar. Take warm reset as an example to illustrate the execution steps.

1, Select Online→Warm Reset in the menu bar.

Example: Execution steps for warm reset

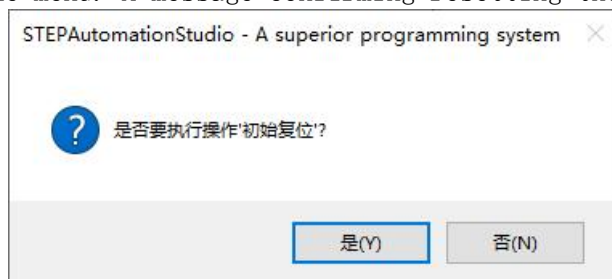


2, Click the [Yes] button. Perform a warm reset.

6.5.2. Reset the device from STEP AS

To reset the device from STEP AS, please right-click on the navigation bar window and execute from the displayed menu.

1) Right-click the device object in the navigation bar window and select "Initial Reset" from the menu. A message confirming resetting the device is displayed.



2) Click the [Yes] button. Perform a device reset. You will be logged out when you perform a device reset.

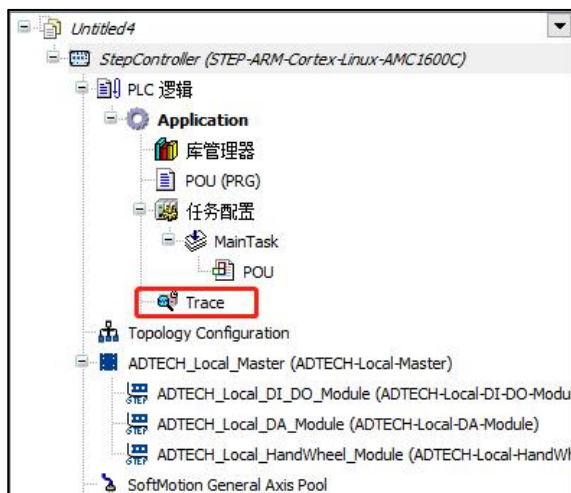
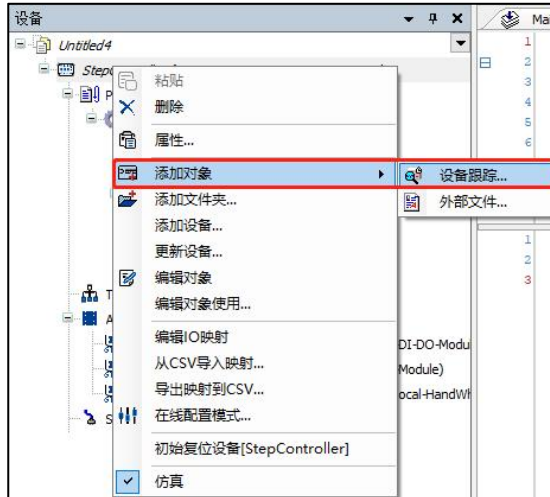
Right-click on the navigation bar window[Application]object, select "Remove the app from the device" to delete the selected application.

6.6. Device tracking feature

6.6.1. Device Tracking General Features

Using STEP AS's device tracking, you can monitor the STEP controller's data waveforms.

1, Right-click on the device and select Add Object→Device Tracking...

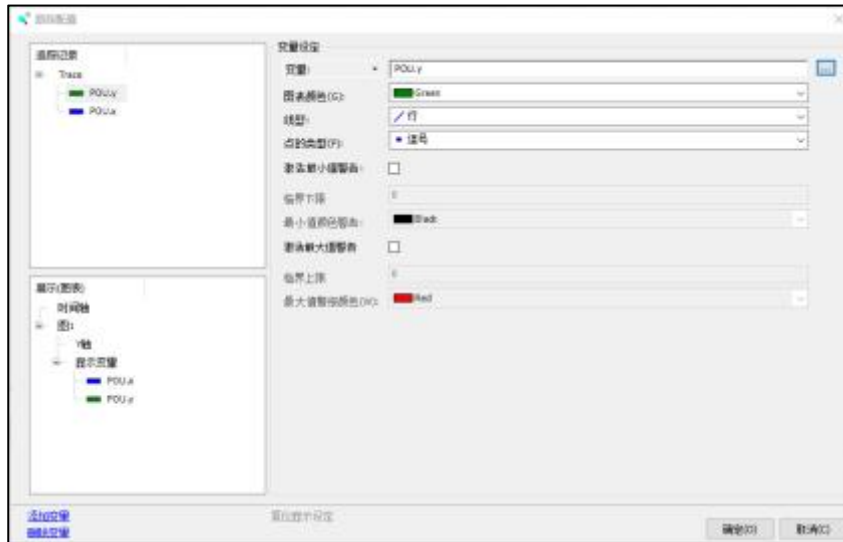


2, is in a state of being logged into the device.

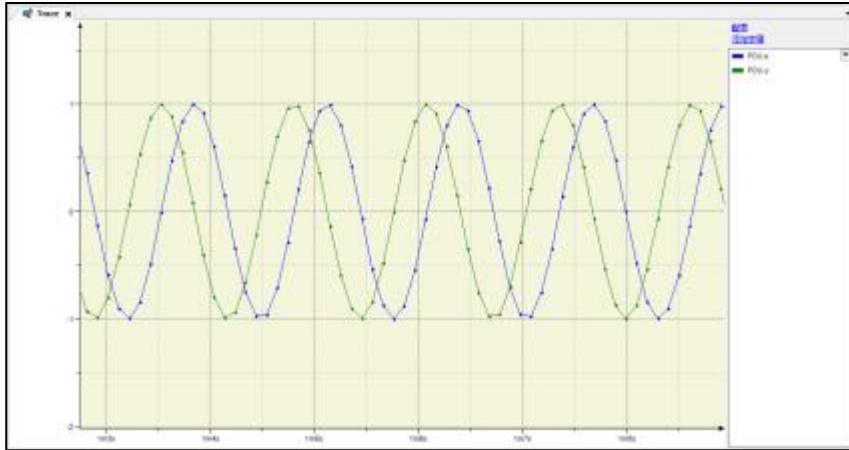
3. Double-click the added Trace to open the "Trace" window.



4. Click "Configure" or "Add Variable" to open the trace configuration window to add variables and make related settings.



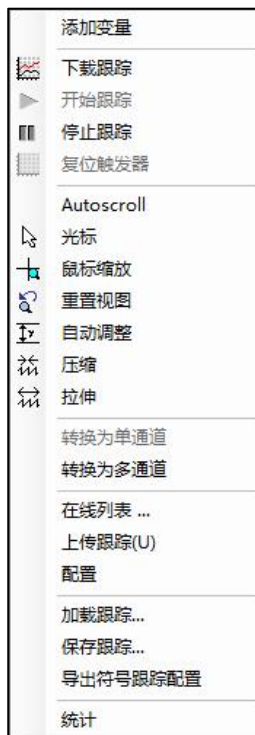
5. Graphical operations
 - Drag: move the timeline
 - Ctrl+drag: Move the Y axis
 - Scroll: Zoom the timeline
 - Ctrl+Scroll: Scale Y axis



6. right-click menu

Save Trace: You can save the information drawn in the graph to a file.

Load Trace: Trace files saved with "Save Trace" can be read on the graph.



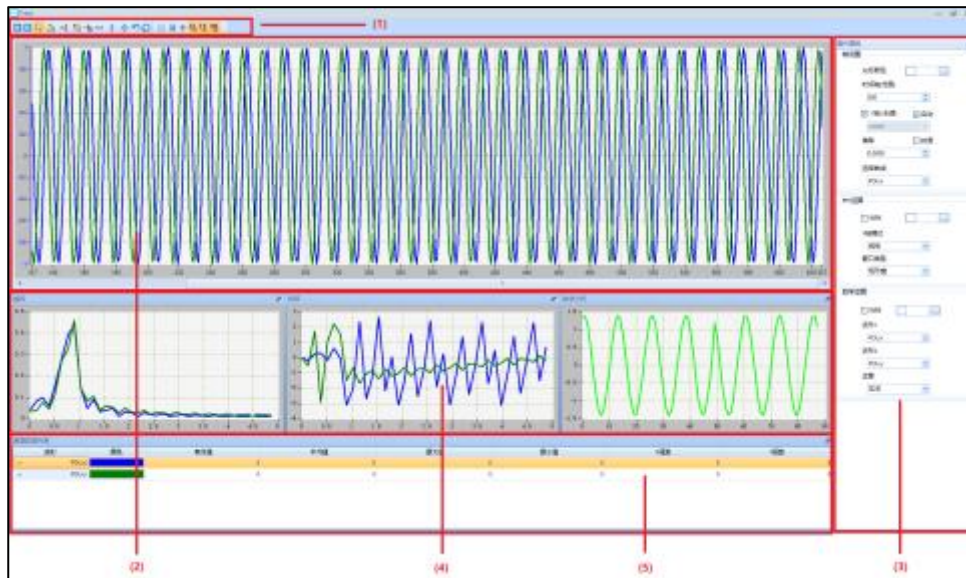
6.6.2. Device tracking analysis function

STEP AS also provides the waveform data analysis function. Click the STEP Trace icon in the device trace tool bar to enter the STEP debug trace page. This icon is valid only after the login is successful.



Enter the STEP debugging trace page to see the following picture, the curve is the variable

added in the trace configuration.
















No.	name	content
(1)	toolbar	Displays buttons for play, stop, mode, coordinates, math analysis, etc.
(2)	Waveform display area	The area where the waveform is displayed in real time.
(3)	Operation panel	Configure the cursor, coordinate range, and calculation method for the waveform display.
(4)	Mathematical analysis display area	F to display the waveformFTCalculations, basic mathematical operations curves.
(5)	Channel data list	Displays the effective value, average value, maximum value, etc. within the range of the curve cursor.

1) toolbar

The display content of the toolbar is as follows:

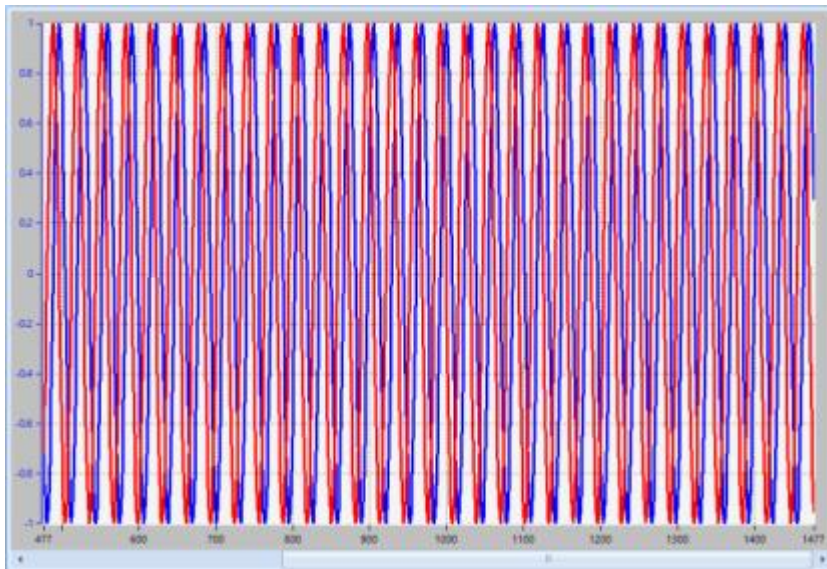


name	icon	Features
play		Click to display the curve in real time.
stop		Click to stop the curve display.
normal mode		Normal mode when selected.
X-axis zoom mode		When selected, it is the X-axis magnification mode.
Y-axis zoom mode		When selected, it is the Y-axis magnification mode.

name	icon	Features
XYShaft zoom mode		X when selectedYAxis magnification mode.
Coordinate scaling mode		When selected, it is the point zoom mode.
X-axis translation mode		When selected, it is the X-axis translation mode.
Y axis translation mode		When selected, it is the Y axis translation mode.
XYaxis translation mode		X when selectedYAxis translation mode.
revoke		Undo the operation.
reset		reset operation.
cursor		Check to display coordinate guides.
fixed cursor		Check to lock the guideline distance.
point coordinates		Check to display point coordinates.
Amplitude		Check to display the amplitude-frequency curve.
Phase frequency		Check to display the phase-frequency curve.
Mathematical analysis		Check to display math analysis curves.

2) Waveform display area

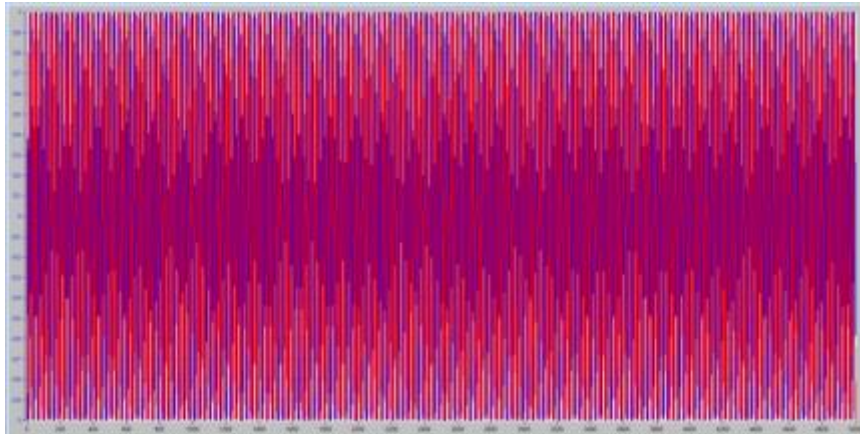
Click the play button, the waveform display area will display the variable curve, and its sampling frequency is the task cycle in the debug trace configuration, and the display effect is as follows.



3) Operation panel

The operation panel can configure the waveform display, including axis setting, FFT operation, and mathematical operation. The axis setting is mainly to configure the displayed waveform; the FFT operation configures the FFT waveform display; the mathematical operation configures the display of the mathematical analysis waveform.

Axis setting: The time axis is used to set the number of points on the X axis in the display area. The effect after setting 5000 points is as follows.

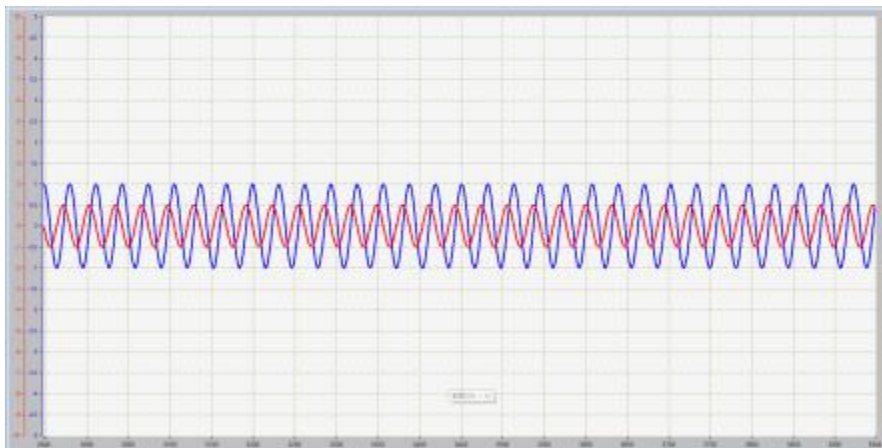


Select Curve is used to select the curve for which the current setting is valid.

Y-axis (scale) is used to set whether the Y-axis of the current curve is visible.

Automatic is used to select the Y-axis scale mode of the corresponding curve. When selected, it is the automatic mode, and the coordinate range automatically changes with the range of the curve value. When not selected, the coordinate scale unit is the scale value set in the setting box in the following line.

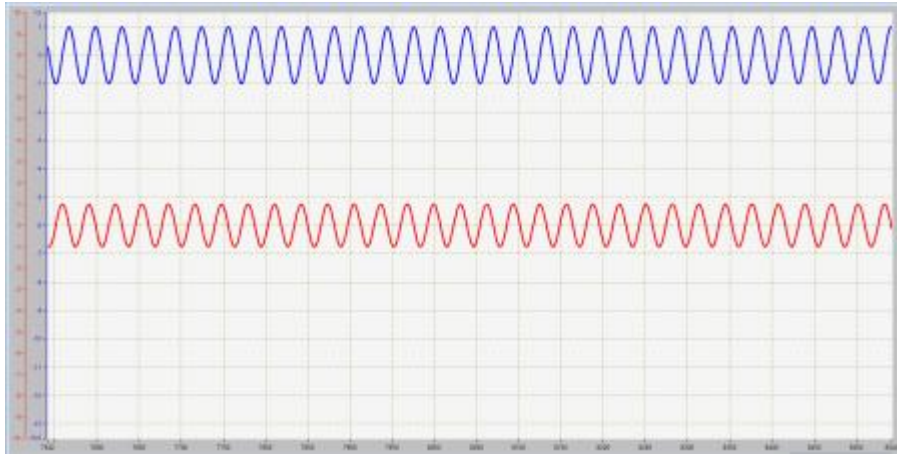
The Y-axis scale setting box is only valid when the scale mode is not automatic. The effect of setting the scale on the Y-axis is as follows.



Offset is used to set the offset position of the Y axis.

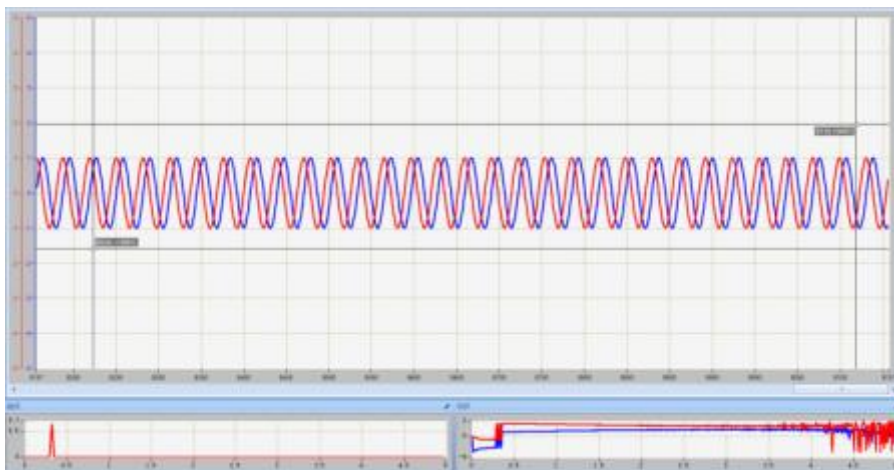
The scale is used to set the mode of the current offset.

The offset setting box is used to set the offset distance. A positive value indicates an upward offset, and a negative value indicates a downward offset. When the scale is selected, the offset distance is the setting value * the scale unit. When the scale is not selected, the offset distance is the actual setting value. , the effect after setting the offset is shown below.

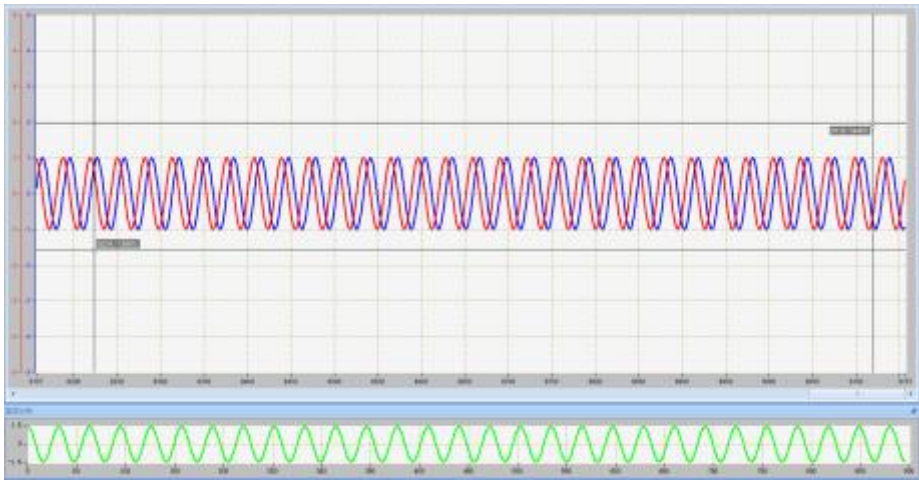


4) Data analysis display area

- FFT operation: Y-axis mode is used to select linear or logarithmic mode for Y-axis in the FFT window; window type is used to select the window function of FFT. The effect of the FFT operation is as follows, and the waveform is the FFT analysis result in the cursor area.



- Mathematical operation: Waveform 1 is the waveform before the mathematical analysis operator; Waveform 2 is the waveform after the mathematical analysis operator; Operation can select the type of mathematical operation, including addition, subtraction, multiplication and division. The following is the effect of the addition operation.



The cursor is used to select whether to display the coordinate guide line of the corresponding form; the cursor color is used to set the color of the coordinate guide line.

5) Channel data list

Display the mathematical analysis values in the cursor selection area, including effective values, maximum and minimum values, etc. The display effect is as follows.

通道名称	单位	物理量	平均值	最大值	最小值	有效值	有效值
PLC_PLC1		0.7071067811865475	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
PLC_PLC2		0.7071067811865475	-0.0000000000000000	0.0000000000000000	-0.0000000000000000	0.0000000000000000	0.0000000000000000

宽度	250
高度	150
文本	
文本	Control Cabinet2
工具提示	

a)

a) 文本属性修改



b)

b) 显示效果

图 9.X 组块工具

第七章 Axis run control configuration

7.1. Axis running configuration

7.1.1. Axis Basic Settings

Including virtual axis/real axis mode selection, linear/rotation mode setting, CNC limit parameters, etc.

NOTE:

(1) Two modes are supported: virtual axis mode and real axis mode. When "Virtual axis mode" is checked, it means that the virtual axis mode is selected. If it is not checked, it means that the real axis mode is selected (the default is not checked), as shown in the figure below.

(2) No matter which mode it is in, the "Linear Mode" or "Modal Mode" setting is supported, as shown in Figure 2 below

category	Function description
virtual axis mode	It can run the mode without physical servo and motor, and can simulate the operation to obtain the required parameters. Not disturbed by the external environment.
real axis mode	It must run with a servo motor, and some parameters must be obtained in the real axis mode, such as online CoE, real axis mode, there will be external interference, such as in TRACE, it will affect the display effect.



Graph Linear Mode



Graph Modal Pattern

- ① Virtual axis mode: If checked, it is virtual axis mode; if not checked, it is real axis mode.
- ② Linear mode: the parameters of the axis increase or decrease in a linear manner;
- ③ Software error handling: only valid in linear mode, mainly used for the response method of the software to the error when the axis is in error.
- ④ Software limit: used for negative and positive limit in linear mode
- ⑤ Modal mode: Axis parameters are executed repeatedly in rotation cycles.
- ⑥ Dynamic limit: mainly used for axis setting.
- ⑦ Speed acceleration and deceleration type: mainly used for the speed running track of the axis.
- ⑧ Identifier: axis for external ID
- ⑨ Position following error: How to manage the movement of the axis after position lag.

7.1.2. Unit Conversion Configuration



Chart unit conversion

- ① Display unit: Also known as user unit, it uses a common daily unit of length for customer display.
- ② Reverse Direction: If checked, the axis rotates in the opposite direction.
- ③ The number of motor command pulses: the encoder resolution of the motor, the number of pulses required for the motor to rotate once, the default value is ()
- ④ Working stroke: the distance that the mechanical end moves when the motor rotates once (user unit). If there is a transmission device (such as pulley, mechanical gear, reducer, etc.) between the motor and the mechanical end, check "Use reducer", otherwise, select "Do not use reducer".
- ⑤ The travel distance of one rotation of the working gear: the distance moved by the mechanical end of one rotation of the working gear (user unit).

7.1.3. Automatic mapping settings



Graph automatic mapping



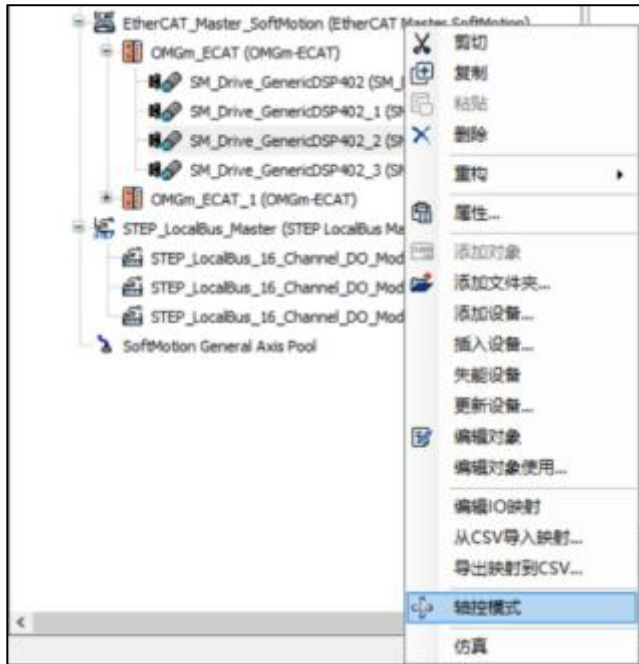
graph information

As shown in the figure, you can see the axis type, version number and other information.

7.2. Single axis control

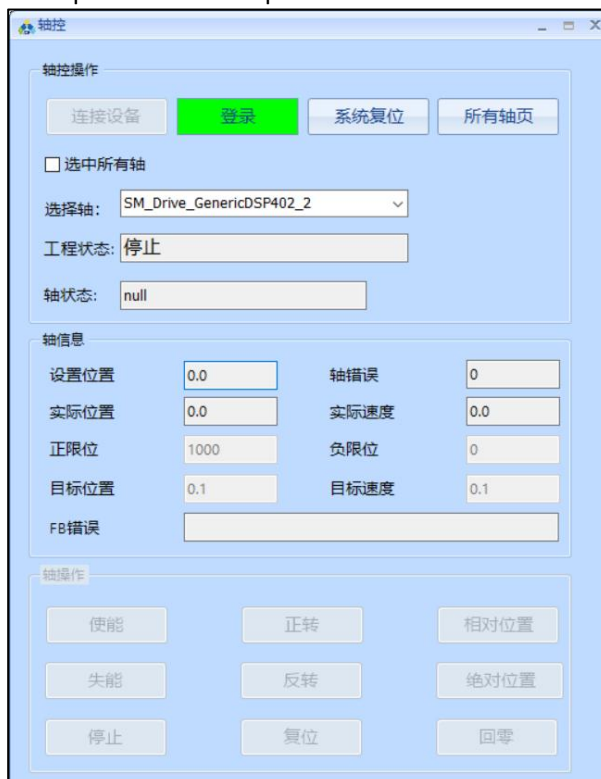
7.2.1. Enter the axis control page

The user configures the axis to be debugged and the connected controller device in the project, and can enter the axis control mode by right-clicking the 402 axis in the navigation bar.



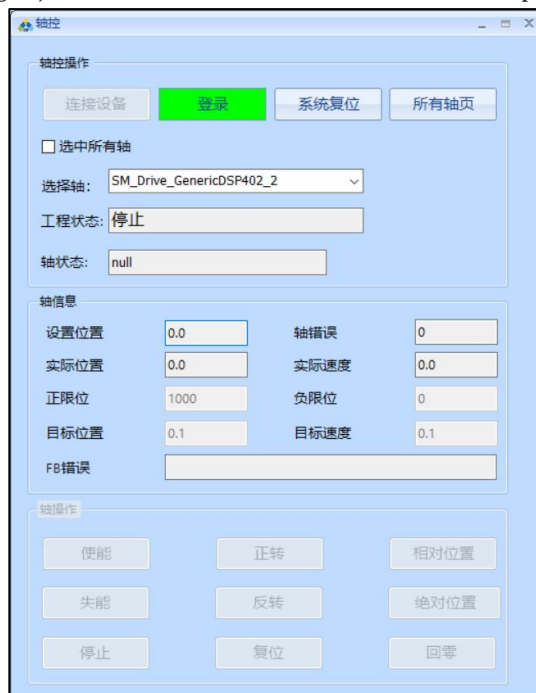
7.2.2. Axis operation and status


After entering the axis control page, click the login connection axis in the axis control page; when the project status is displayed as normal, you can click the button in the axis operation to control the axis. If the project status is displayed as stopped, you can click System Reset to reset. If it is still invalid for a long time, click logout, and then click login again to repeat the above operations.

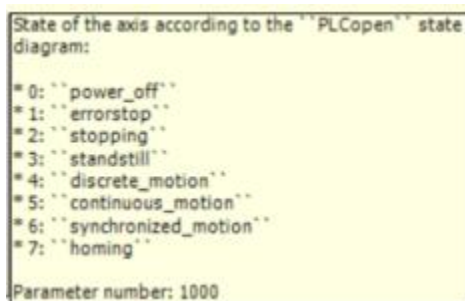


①Note: If the project is connected to the device, in the axis control mode, you can switch between the device and the simulation by clicking the connected device. If the axis control mode entered in the form of simulation, it can only be the simulation mode.

Use the drop-down box to switch to different axes for debugging. After selecting the axis to be debugged, the information on the current page is switched to the selected axis.



The following figure shows the axis status display, which is used to display the axis status and communication status of the current axis; the abnormal display of the communication status, the normal display .



In the axis control mode, the axis can be enabled, forward and reverse, jog and other operations. The page displays the parameter information and error information of the currently selected axis (including axis error and FB error). After running, you can configure the software limit of the axis, the running target speed and the target position during jogging. The position information during jogging is relative. Location. Speed and location information needs to be logged in to take effect.

Errors can be reset using a system reset, clearing the error message, returning to the default state, or using an axis reset operation to clear the error.



7.2.3. Multi-axis debugging

In addition to debugging a single axis, it is also possible to tune all axes simultaneously by selecting all axes. It should be noted that during on-site debugging, it is necessary to confirm whether other issues such as limit need to be considered, and then operate all axes.

In order to facilitate debugging of multiple axes, you can open all axis control pages to display the information of all axes in a list. If you do not enter the axis control mode, there will be no data display.



After entering the axis control mode, you can see the list of all the currently debuggable axes, but the information of the axis will not be displayed, and the operation of the axis is also unavailable.



The information of the axis will be updated to the page only after logging in. After logging in, you can set the limit of the axis, set the speed and position and other information.



Click the enable column to enable or disable the axis, and there are buttons on the lower side to enable/disable all axes. Other buttons are also available for all axes, and these operations can be used to debug a single axis, multiple axes or all axes together.

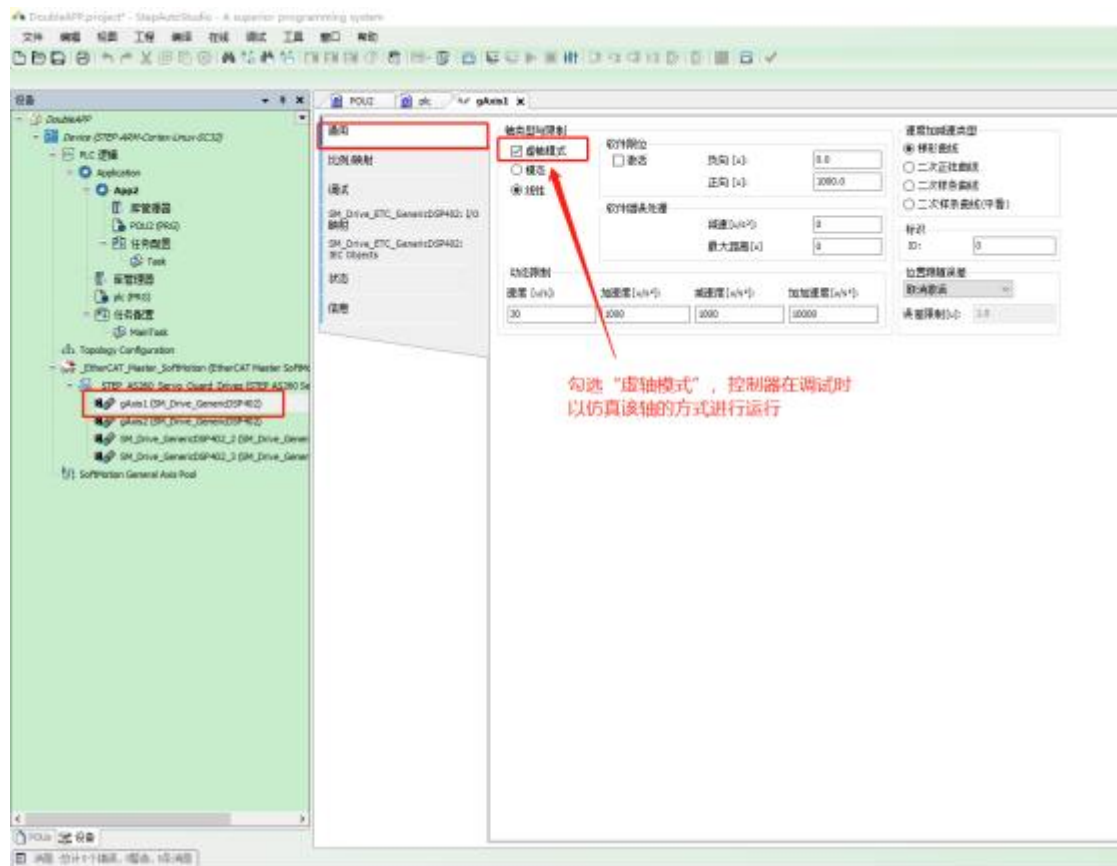


When all axes are selected, the configuration of all axes can be modified by modifying the speed, position or limit value of one axis.



7.3. Simulate Servo Drive

When programming and debugging the MC operation control application program, the programmer has SC30 controller at hand, but no servo driver, or there is not enough servo driver. If you want to debug and debug the user program, you can use the "virtual axis" method to replace the servo The real axis of the drive, as shown in the following figure:



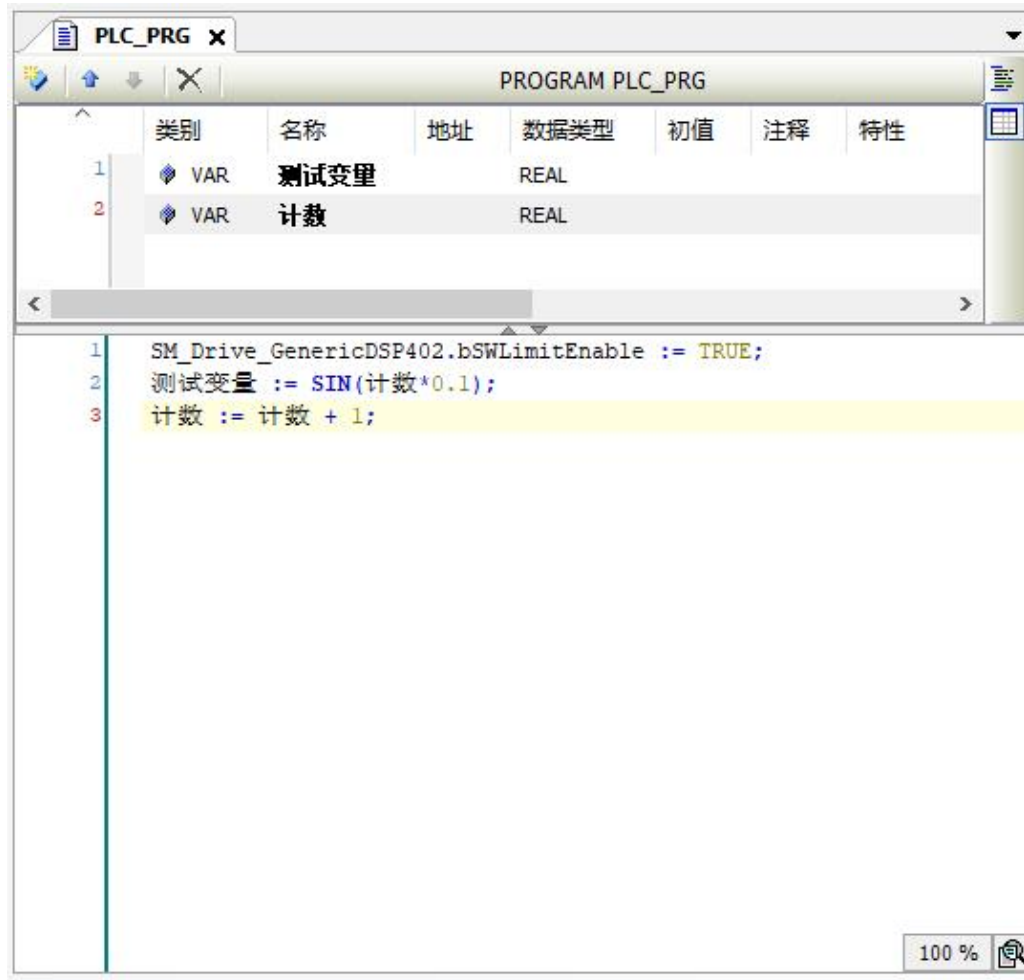
During programming and debugging, if the number of connected servo axes is different from the number configured in the user program, the system will alarm and cannot be debugged normally. The way the servo operates. The “running” state of the axis can be seen visually, verifying the correctness of our MC control program.

The imaginary axis is also the axis. Although it is a “virtual axis”, the operation logic of the state of the axis still needs to be programmed according to the state transition logic in the PLCopen specification. For example, MC_Power must be run before running, MC_Reset should be run after an error occurs, etc., which is convenient for us to debug and rule out logic errors in the user program.

If the actual servo axis is connected, just cancel the “virtual axis mode” of the corresponding axis in the above figure, and then it can run normally.

第八章 How to edit the program

STEP AS supports the PLC international standard IEC 61131-3. The program structure of the six languages is shown in the figure below. The upper part is the variable declaration, and the lower part is the program implementation. The following figure is an example of ST language.



①Tip: STEP AS supports Chinese programming.

8.1. Structured Text (ST) Programming

This section describes the procedure for creating a program (ST program) using structured text.

- Creating an ST program requires the ST program's POU object. Set the object setting language to ST program.

- ST programs consist of a combination of expressions and instructions. Expressions and instructions can also be executed in conditionals or loops. Each instruction must end with a semicolon (;).

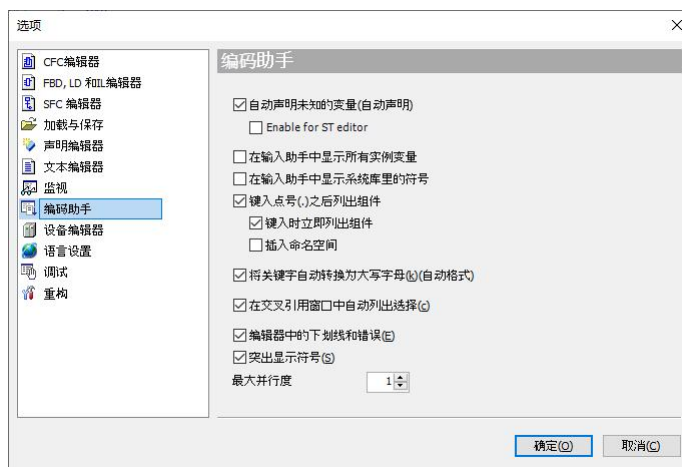
8.1.1. Syntax of ST program

The following syntaxes can be used in ST programs.

project	example
assignment statement:=	Assign the value on the right to the left. example: iVar1 := 4;
set assignment statementS=	If the right side is TRUE, set the left side to TRUE. Once left becomes TRUE, left remains TRUE even if right becomes FALSE. Do not insert spaces between S and =. example: Var0S=Var1;
reset assignment statementR=	If the right side is TRUE, it is set to FALSE on the left side. Once the left side becomes FALSE, the left side remains FALSE even if the right side becomes FALSE. Do not insert spaces between R and =. example: VarR=Var1;
IF instruction	Judging the conditions, execute the instruction according to the judgment result. example: IF temp < 17 THEN heating_on := TRUE; ELSIF temp > 25 THEN open_window := TRUE; ELSE heating_on := FALSE; END_IF;
FOR instruction	Repeats the instruction the specified number of times. example: FOR iVar0 := 1 TO 10 BY 1 DO iVar1 := iVar1 + 1; END_FOR;
WHILE command	Judging the conditions, repeating the execution of the instruction when the conditions are met. example: WHILE (iVar0 <> 0) DO iVar1 := iVar1 * 2; END_WHILE;
CASE instruction	Judging the conditions, execute the instruction according to the judgment result. example: CASE iVar0 OF 1 : iVar1 := iVar1 / 2; 2 : iVar1 := iVar1 / 4; ELSE iVar1 := iVar1 / 8; END_CASE;

project	example
REPEAT command	Judging the conditions, repeating the execution of the instruction when the conditions are met. example: REPEAT iVar0 := iVar0 + 1; UNTIL iVar0 = 100 END_REPEAT;
EXIT command	The EXIT instruction is used to end the loop within the FOR instruction, WHILE instruction, and REPEAT instruction.
RETURN instruction	The RETURN instruction is used to end the POU. Instructions in the POU following the RETURN statement are not executed.
JMP instruction	The JMP instruction unconditionally moves to the line where the JMP label is located. example: iVar0 := 0; Label1 : iVar0 := iVar0 + 1; IF (iVar1 = 5) THEN JMP Label1; END_IF;
CONTINUE command	The CONTINUE instruction is used to move to the beginning of a loop within a FOR instruction, WHILE instruction, REPEAT instruction.

Entered keywords are automatically converted to uppercase letters (auto format).
tool→OptionsIn the "Automatically convert keywords to uppercase (k) (auto format)" category, uncheck the auto format settings.



8.1.2. Comment ST procedure

Comments can be made in ST programs. The commented part cannot be executed.

type of annotation	content
single line	Beginning with // to the end of the line becomes a comment. example: bVar1 := 2; // single line comment

Multi-line	<p>The part from (* to *) becomes a comment. You can also enter (* *) in (* *). example:</p> <pre>(* multi-line comment1 multi-line comment2 *)</pre>
------------	---

8.1.3. call function block

Function blocks of all IEC libraries can be called, regardless of the original programming language.

Examples are as follows:






TMR:TON;

TMR (IN:=%OX5, PT:=T#300ms);
varA:=TMR.Q;

8.2. FBD/LD/IL programming

8.2.1. FBD/LD/IL Editor (Editor)

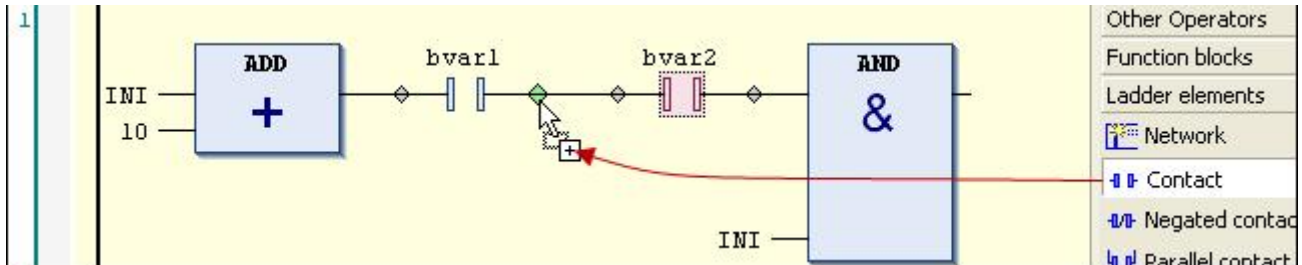
The three programming languages FBD/LD/IL share an editor, so the operation is similar. The toolbar in the lower right corner of the editing area is shown in the following table:

	Return to normal editing mode. The mouse pointer changes to the shape of the default arrow. You can select and edit components in Edit View.
	Pan Tool: The mouse pointer changes to the shape of 2 arrows. You can click and drag in the editor view FBD / LD / IL editor or move the CFC chart towards the visible area.
	Zoom in tool: In the lower right corner of the editor view, the mouse pointer changes to the position of a cross, and a zoom window can be opened. When you move the mouse pointer over your graph, the zoom in tool will show up at the intersection at 100% zoom. Please note: If you click on the view, the magnification tool will close and a portion of the graph will be displayed at 100% magnification. If you want to keep the zoom factor set, then you should use the default arrows () to return to the default edit mode.
	Zoom In Tool: This will open a drop down list to select a zoom factor. Clicking More Options (...) will open the zoom in dialog for entering more values. The current zoom factor is always displayed to the left of the symbol.

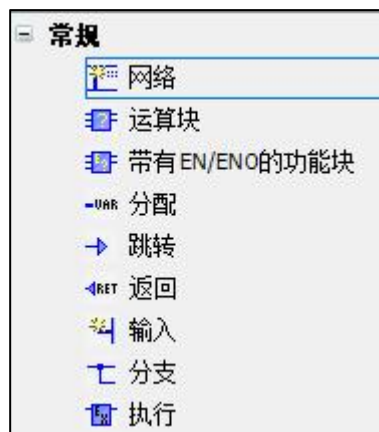
Zooming with the wheel: Hold down the [Ctrl] key and move the wheel, you can change the zoom factor in 10% steps.

It is also possible to use the mouse to move components in the editor.

Each graphical editor has its own toolbox view, by default, to the right of the editor view. A component contained in the toolbox that can be dragged to the insertion point in the editor view. STEP AS highlights the insertion point with a grey position marker in the shape of a diamond, triangle or arrow. These flags are green when you move the mouse pointer. Release the mouse button to insert the component at the selected location. As shown below.



◆ conventional components



◆ Element "Network"

symbol:

The network is the basic unit of an FBD or LD program. existFBD/LD/ILIn the editor, networks are assigned to a list. Each net is provided with a continuous net number on the left, which includes: logical and arithmetic expressions, program/function/function block calls, jump or return statements.

An IL program contains at least one network. The network can include all the IL statements of the program.

You can provide titles, annotations or labels for each network. Available in options (categoryFBD, LD, and IL, you can define whether the delimiter between net titles, comments and individual nets should be displayed in the editor.

◆ Element "Operation Block"

symbol:


A box and its calls can represent additional functions such as IEC function blocks, IEC functions, library function blocks, operands.

A box can have any number of inputs and outputs.

If the box also provides an image file, the box icon will be displayed inside the box. Available in Software Options, CategoryFBD, LD and ILactive displaysymbol boxoptions.

If you change the box interface, you can use [FBD/LD/IL → update parameters](#). The command updates the parameters of the box without reinserting the box.

◆ Component "assignment"/assign"

symbol: 

The FBD editor displays the newly inserted assignment in a straight line followed by 3 question marks. The LD editor shows the newly inserted assignment as a coil with 3 question mark markers above the coil.

Once inserted, you can replace the placeholder ??? with the name of the variable to where the left generated signal is assigned. You can use the input assistant to do it.

◆ element with EN/ENO function block

symbol: 

This element is only available in the FBD and LD editors.

Boxes typically correspond to FBD/LD/IL elements [frame](#); however, the box also includes an EN input and an ENO output. EN and ENO data types are BOOL.

Functions with EN input and ENO output: If the value of EN input is FALSE when the box is called, the operation defined in the box will not be executed. However, if the value of EN is TRUE, these operations will be performed. The ENO output has the same value as the EN input.

◆ Element "input"

symbol: 

The maximum value entered depends on the type of box.

Newly added inputs are first marked with ????. You can replace ??? strings with variables or constants.

◆ Component "label"

In FBD and LD, labels are optional identifiers in the network that you can use to specify as a jump target.

If you insert a jump label in the network, it will be added as an editable area label in the network.

① see also

◆ element "jump"

symbol: 

In FBD or LD, the current cursor position determines whether to insert jumps directly before the input, directly after the output, or at the end of the network.

You can enter a jump label directly after the jump element as the jump target.

In IL, you can edit a jump with the JMP instruction.


◆ element "return"

If the input to the RETURN element becomes TRUE, this element immediately interrupts the execution of the box.

In an FBD or LD network, you can place return instructions after or in parallel with preceding elements.

The RET instruction in IL is given to you for exactly this purpose.

◆ element "branch"

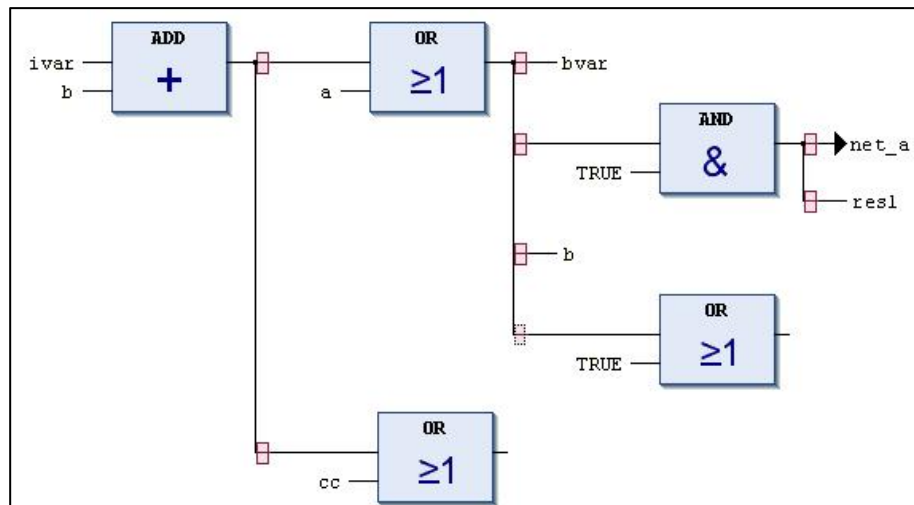
symbol: 

This element is available in the LAD and FBD editors and represents an open line branch. A branch splits the processing line from the current cursor position into 2 subnets, the operation is performed consecutively from top to bottom. You can further split each subnet and finally build a multi-branch structure in one network.

Each subnet is given a marker symbol (rectangle) at the branch point, which you can select

to execute further commands.

❗ Notice: *copy*, Cut and Paste commands are not available for subnets.



In order to delete a subnet, you must first delete all components in the network, and then delete the marker symbol for the subnet.



◆ Element "Execute"

symbol: 

The element is a box that allows you to enter ST code directly in the FBD and LD editors.

You can drag actuators from the tool view to the actuators of your POU by dragging them with the mouse. If you click Enter ST code here..., the input field will open and you can enter multiple lines of ST code.


◆ LDelement "touch point"

symbol: , in the editor 

This element is only available in the LD editor.



Contacts pass left to right on the TRUE (ON) or FALSE (OFF) signal until the signal finally reaches the coil on the right side of the network. For this purpose, a Boolean variable containing this signal is assigned to the contact. To do this, replace the '???' placeholder above the contact with the name of a boolean variable.

You can arrange some series or parallel contacts. In the case of two parallel contacts, only one needs to get a TRUE value for ON to be passed to the right. If the contacts are connected in series, all of them must get a TRUE value for ON to be passed to the right of the last contact in the connection. Therefore, you can program the electronic type parallel and series connections with the LD.

If the variable value is FALSE, the negative contact  forward pass TRUE signal. you can [FBD/LD/IL → negative](#) negates an insertion contact with the help of the command or [tool Insert](#) a negative contact into the view.

If you place the mouse pointer on the selected contact of the network where the left mouse button is pressed, then [Convert to coil](#) The button will appear in the network. If you now move the mouse pointer to the button, still press the button with the mouse, and release the mouse button on this button, STEP AS converts the contact to a coil.


◆ LDelement "coil"

symbol: , in the editor 





This element is only available in the LD editor.

The coil takes the value provided from the left and is saved in a boolean variable assigned to that coil. Its input value can be TRUE (ON) or FALSE (OFF).

Multiple coils in a network can only be arranged in parallel.

in an invalid coil , the negated value of the input signal is stored in a Boolean variable assigned to that coil.

◆ set coil, reset coil

symbol: , , in the editor: , 

Set Coil: If the value TRUE reaches a set coil, the coil retains the value TRUE. As long as the application is running, the value will no longer be overridden here.

Reset Coil: If the value TRUE reaches the reset coil, the coil regains the value FALSE. As long as the application is running, the value will no longer be overridden here.

you can **FBD/LD/IL** → **set up/reset** Define an insert coil with the help of the command as a set or reset coil or from **tool** Insert the coil in the view as **set coil** and **reset coil** element.

◆ LD element "Branch Start/Stop"

symbol: 


This element is used to close line branches.

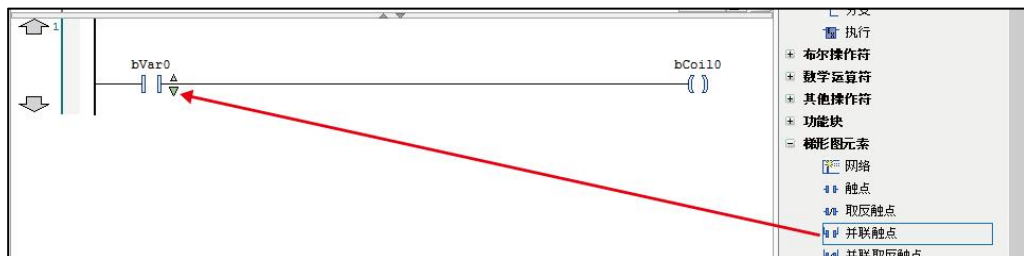
◆ LD Closed Line Branch

A closed line branch is only available in LD and contains a start and end point. It is used for the execution of parallel analysis of logic elements.

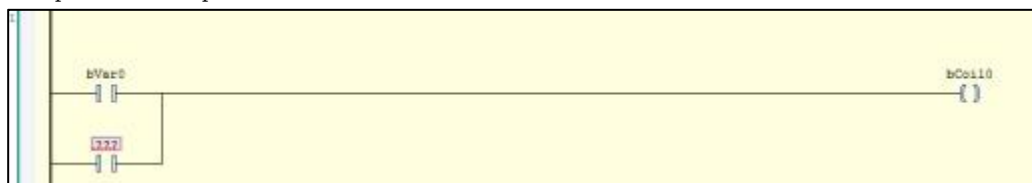
◆ LD programming operation

(1) placed in parallel operation of the contacts

Select in the toolbox **Ladder Diagram Element** → **Parallel Contact**, drag and drop to the  position.

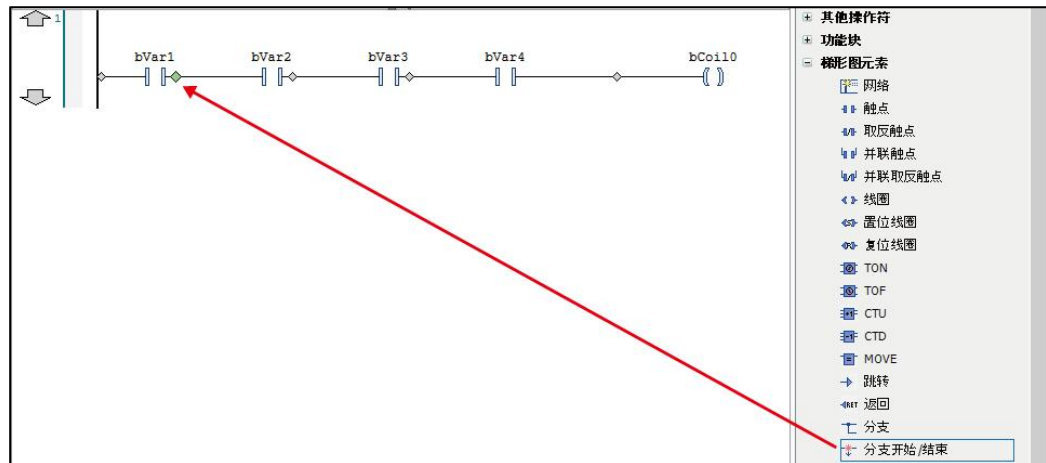


A contacts are placed in parallel.



(2) Insert/end branch at specified position

① Select in the toolbox **Ladder Elements** > **Branch Start/End**, drag it to the display next to the contact bVar1 in the main window.

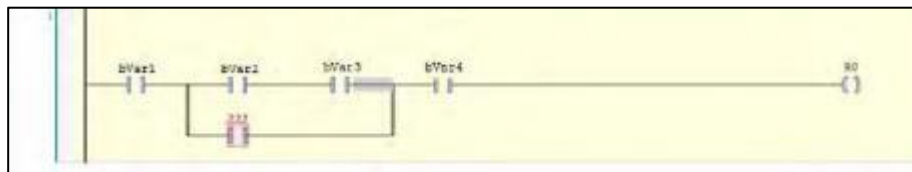


When the drag is complete, a red rectangle marking the start of the branch is displayed between the contacts bVar1 and bVar2. The blue rectangle marks are candidates for where the branch ends.



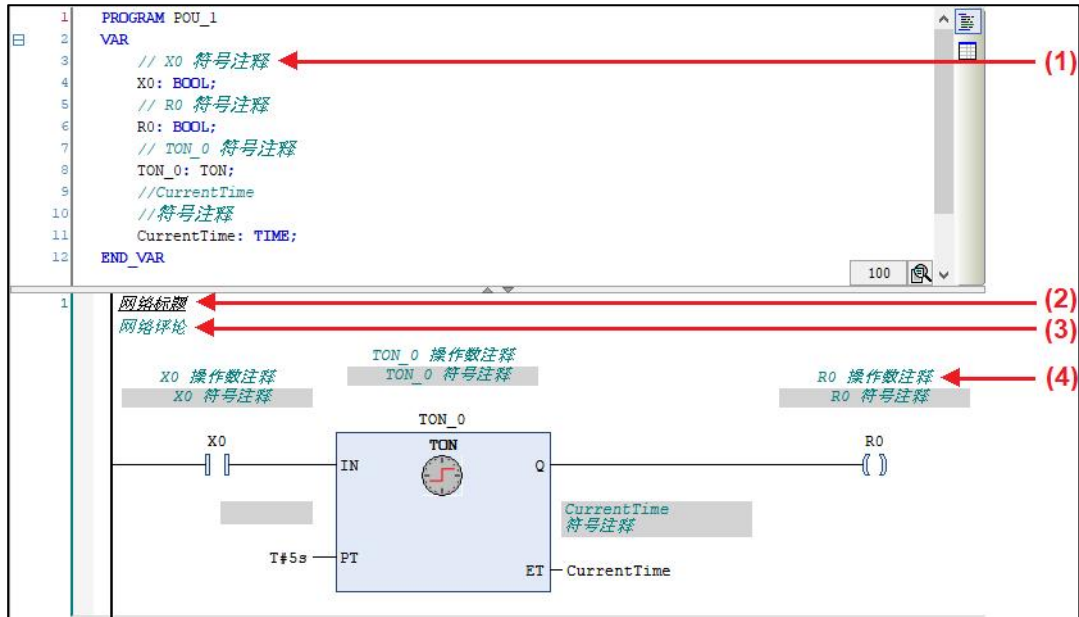
- ② Click on the blue rectangle marker between contacts bVar3 and bVar4.

A branch is inserted from between contacts bVar1 and bVar2 to between contacts bVar3 and bVar4.



- (3) Enter a title and comment (LD)

In the ladder language, the following 4 titles and comments can be entered. Display examples of titles and comments are shown in the figure below.



NO.	project	content
(1)	Symbol annotation	is a comment on the declared variable. Display the same annotation for the same variable. Please enter a comment for the variable in the declaration section. Comments appear in cells with a black background.
(2)	web title (circuit title)	Captions can be added for each net (circuit). Click on the upper left corner of the network (circuit) and enter a title.
(3)	web annotation (Circuit Notes)	Notes can be added for each net (circuit). Click on the upper left corner of the net (circuit) to enter a comment.
(4)	Operand Comments	is a comment on the variable. Different comments can be added to the same variable. Click on the top of each variable in the Implementation section to enter a comment.

To display titles and comments, options need to be set. Launch Options (Tools → Options), select the General tab of the "FBD, LD, and IL Editors" category, and select the items to display in the display bar.

8.3. Sequential Function Chart (SFC) programming

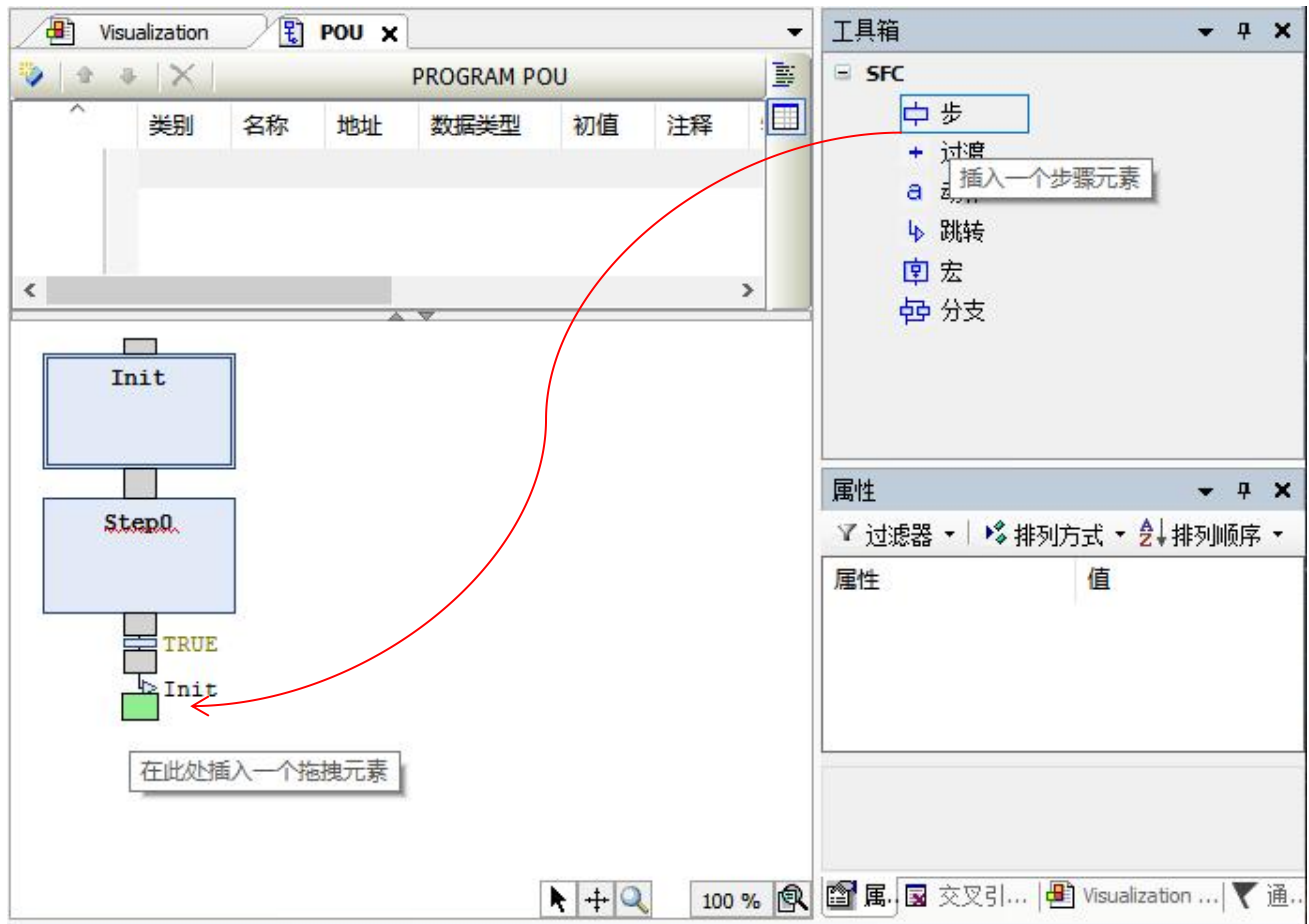
SFC is a sequential graphical programming method, and its program/POU usually includes an initial step program and a subsequent transition step program.

8.3.1. SFC editor


The SFC editor is a graphical editor. In the "SFC" editor, you can insert individual components into the diagram via the main menu, context menu and toolbox view. When the editor

is activated, SFC components are available in the SFC toolbox.

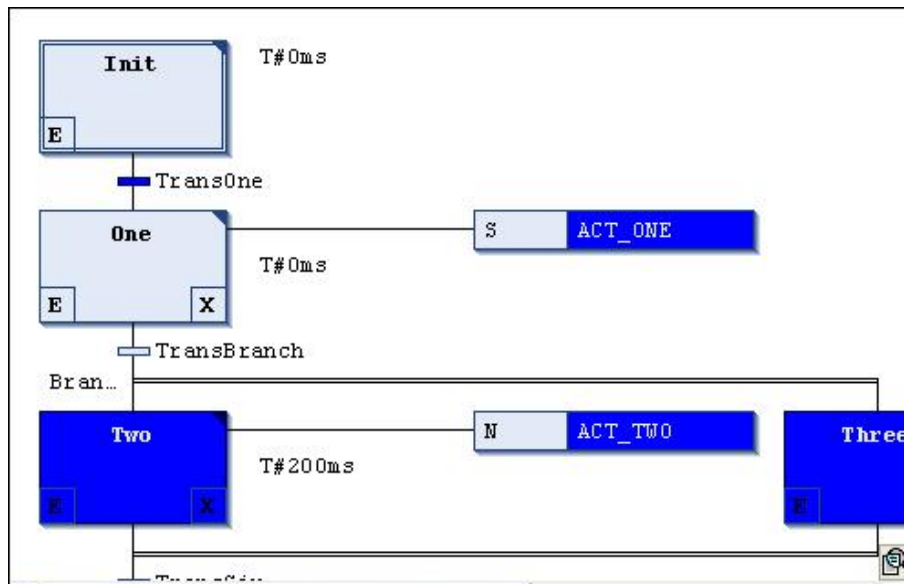
Insert Component: Select an available component in the editor toolbox. Drag the component to be inserted, STEP AS will prompt the insertable position, just drag into the target position. As shown in the figure below, STEP AS marks all possible insertion points with a gray box. If you move the mouse over a gray box, the color of the box will change to green. When the mouse button is released, the object is inserted at that location.



If inserting multiple Action and Transition branches, select these objects in batches.

If a branch  Drag into the diagram, the start and end of the branch must be set using the mouse pointer. Set the start of the branch by releasing the mouse button at the insertion point. Then the color of the box turns red, click "Insert Point", and the setting is finished. STEP AS inserts a branch between the start and end markers.

In online mode, STEP AS displays active steps in blue.



8.3.2. execution order

When performing actions such as exit and input detection, the order of verification is as follows: from top to bottom, and from left to right on the SFC layout.

time check/execute step action

STEP AS performs the following checks for each step to obtain the layout of the SFC:

- ◆ STEP ASCopy the time spent in the active step into the corresponding autonomous step variable<step name>.t. (not yet implemented)
- ◆ If a timeout occurs, then, STEP ASSet the respective error flags. (not yet implemented)
- ◆ For non-standard steps:STEP ASPerform a single step operation.

8.3.3. action qualifier

Specify a qualifier for the step that describes how the single-step action will be processed.

Qualifier preprocessing is done in the SFCActionControl function block in IecSfc.library. This library is automatically integrated into the SFC plugin project.

The available qualifiers are shown in the following table:

N	non-prestored	As long as this step is activated, the action is also activated.
R0	Complete reset	Disable action
S0	set (storage)	STEP AS This action is executed as long as the step is active. This action is performed until a reset signal is received, even if the step has been terminated.
L	limited time	STEP AS This action is executed as long as the step is active. The action is

		performed until the step is stopped or the given time elapses.
D	delay	STEP AS activates the step only after a given delay time and starts executing the action while the step is still active. Execute the action until the step is deactivated
P	pulse	STEP AS executes this action once as long as this step is activated.
SD	Storage and Latency	Only after a given time delay, STEP AS starts the operation and the step becomes active. The action will continue until a reset signal is received.
DS	Latency and Storage	STEP AS activates the step only after a given delay time and starts executing the action while the step is still active. The action will continue until a reset signal is received.
SL	Storage and limit time	STEP AS This action is executed as long as the step is active. Executes this action until the given time elapses, or it receives a reset signal.

① Notice: You must enter the qualifiers multiple times in the format L, D, SD, DS, and SL as time constants.

8.3.4. SFC logo

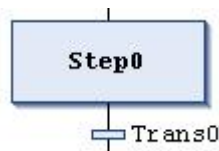
name	data type	describe
SFC initialization	BOOL	TRUE:STEP The initial step of the AS sequence reset. The flags of other SFCs will also be reset (initialized). Need to reset againSFCInitforFALSE.
SFC reset	BOOL	This function is similar toSFCInit. However, STEP AS continues processing after initial step initialization. For example, in the initial step, you can just resetSFC resetmarked asFALSE.
SFC error	BOOL	If a timeout occurs in the SFC diagram asTRUE. If a second timeout occurs in the program, please reset before confirmingSFCErrorvariable. statementSFC erroris a requirement for the function order control of the other flag variables (SFC error step,SFC error POU,SFC exit error).
SFC limit enable	BOOL	use in stepSFC errorIn the timeout control, in particular use activation (TRUE) and the statement (FALSE). If you declare and activate this variable (SFC setting), then you must set it toTRUE, so thatSFC errorWork. If not, the timeout is ignored. This is useful, for example, in startup or manual control. If you do not declare this variable, thenSFC errorwill work automatically. needSFC errorstatement of.
SFC error steps	string	Stores the name of this step, which causes a timeout, which passesSFC erroris registered. needSFC errorstatement of.
SFC error POU	string	Store the function block name when a timeout occurs, then passSFC erroris registered. needSFC errorstatement of.
SFC exit error	BOOL	As long as this boolean variable isTRUE, STEP AS suspends the processing of the SFC diagram and any timeout in the variable,SFC erroris reset. If you







name	data type	describe
		reset the variable toFALSE, then the active step is reset the first few times. needSFC errorstatement of.
SFC suspended	BOOL	as long as this variable isTRUE, STEP AS suspends the processing of the SFC diagram.
SFCTrans	BOOL	If the transition is activated, thenTRUE.
SFC current step	string	Displays the name of the active step, ignoring time monitoring. In parallel branches, the name of the step on the rightmost branch line is always stored.
SFCTip, SFC prompt mode	BOOL	Controls the prompt mode of the SFC function block. if you useSFCTipMode=TRUEenable this flag, then you can simply setSFCTipforTRUE, which activates the next step. whenSFCTipModeset asFALSE, the transition is continued to be used, and the implementation continues to activate.

8.3.5. Components "Step" and "Transform"

step sign , conversion flag 

Typically, STEP AS inserts a combination of steps and transformations to be used. Inserting a transition with no oversteps or without steps will result in a compile-time error.



      Step names must be unique within the scope of the parent block. This will be specially considered when the used action is executed in the SFC.

You can switch to the initial step by clicking on the initial step or modify the respective properties in the SFC property settings.

All steps are defined by step properties, which can be displayed and edited in the properties view, depending on the setting options.

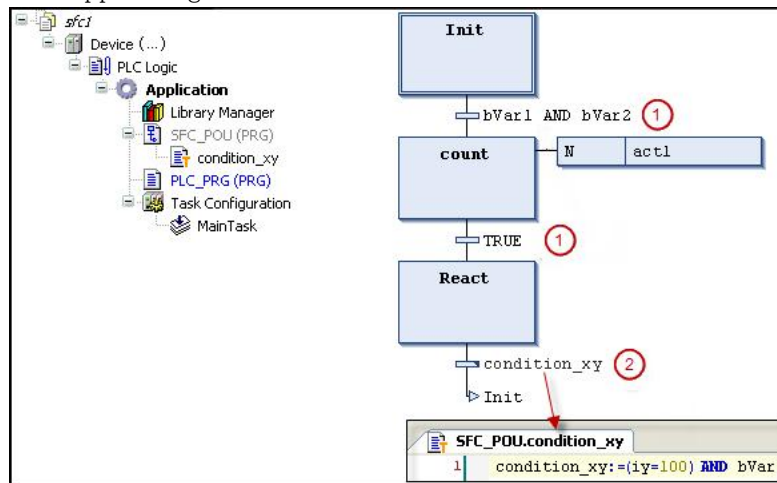
When a step is activated, you must attach these actions to the step.

The transition must include conditions for subsequent steps, as soon as possible set the conditions toTRUE. In this way, the transition condition must be set toTRUEorFALSE. It is defined in one of two ways:

(1) Online state (direct): you replace the default transformation name with a boolean variable, boolean value, boolean variable or boolean result, e.g. (i<100) AND b.

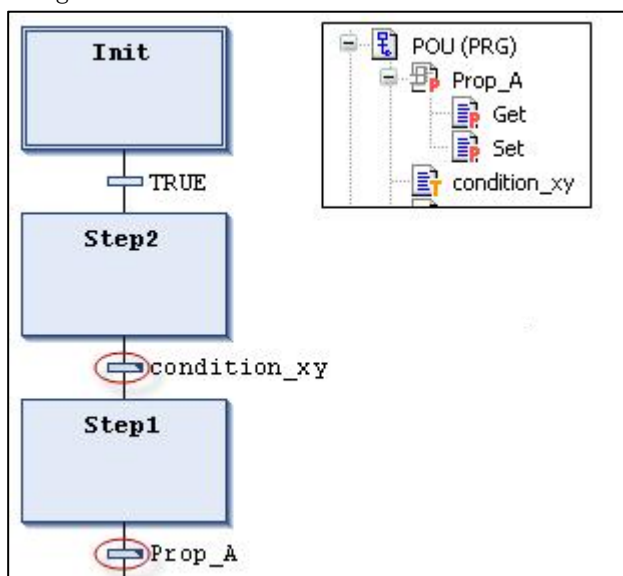
(2) Condition for comprehensive use (individual transition or object property): You replace the default transition name with the name of the object transition or property (📄, 📄). You can click the project → **add object** Create these objects. This allows multiple transformations to be used, such as "condition_xy" below the number. For example an internal condition, an object can contain a declaration of a boolean variable, a boolean address, a boolean constant, and a boolean ending. Additionally, it can contain multiple declarations of arbitrary code.

Transitions consisting of transitions and object properties are marked with a small triangle in the upper right corner of the transition box.




📄 📄 📄 📄 📄 If the transition consists of multiple statements, it is the user's responsibility to assign the necessary expression transition variables.

Transitions consisting of transitions and object properties are marked with a small triangle in the upper right corner of the transition box.




8.3.6. Element "action"

meets the: 

Actions consist of declaring a series of statements in a valid execution language. You can assign actions to steps.

You must create all actions in the project as POUs when they are used in SFC steps.

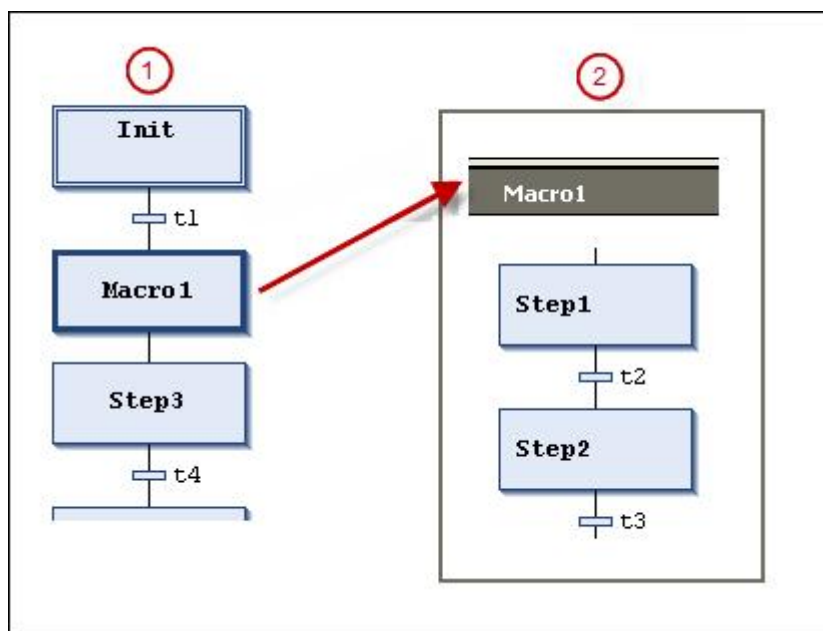
8.3.7. Component "Macro"

meets the: 

A macro contains part of an SFC diagram, but does not display details on the main editor view.

The use of macros does not affect the processing flow. Macros are used to hide specific parts of the plot, for example to increase overall clarity.

By double-clicking the macro box or clicking **SFC** → **zoom in macro** Open the macro editor. You can program here as in SFC in the main view of the editor. click **SFC** → **shrink macro**, closes the macro.



① Main view of the SFC editor

② Macro editor view for macro 1

A macro can also contain other macros. The title of the macro editor always shows the path to open the macro.


E.g:




8.4. Continuous Function Chart (CFC)

8.4.1. CFC editor.

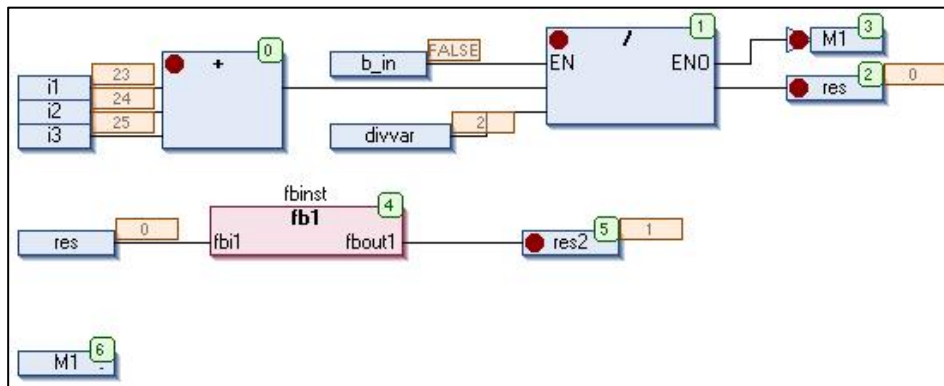
can be obtained from [toolbox](#) Drag components in the view to add them to the CFC editor. In addition, it can also be used in [CFC](#) commands in the menu.

You can connect a component's input and output pins by dragging a connector. STEP AS automatically generates connecting lines (auto-routing) between components with the shortest distance. When you move a component, the connection line automatically adjusts. If you move an element to a position that causes the routing element's connections to overlap, then a conflict can arise. This is done by connecting symbols () is indicated by a red outline around it. Overlapping connections are indicated by red connection lines in the editor.

Select the pointer from the toolbox (), the shape of the cursor changes to an arrow. You can then select the inserted components in the editor view to position and edit them.

8.4.2. Position of the break point in the CFC editor

Following the usage rules, you can set a breakpoint in any block where a variable can be changed, and locate it in a program flow branch or another called block. In the image below, red circles indicate possible breakpoint locations.



8.4.3. CFC components

◆ CFCelement "page"

symbol: 

The element inserts a new page into the editor. It is only available in the page-oriented CFC editor. The number of pages is automatically allocated based on their location. You can enter a name and a description of the page to generate an orange title. pass [Edit page size](#) command to resize the page.


◆ CFC element "Control Point"

symbol: 

Use a control point to secure the connection point before you adjust the path of the line. This is done by dragging the component to the desired position on the connector. Connector lines with control points are no longer auto-routed.


◆ CFC element "input"

symbol: 

STEP AS inserts a default input element with the text "???". You can edit this field directly by clicking and entering a constant value or variable name. Alternatively, you can click  Open the input assistant to select a variable.


◆ CFC element "output"

symbol: 

STEP AS inserts a default output element with the text "???". You can edit this field directly by clicking and entering a constant value or variable name. Alternatively, you can click  Open the input assistant to select a variable.

◆ CFC component "box"

symbol: 


Use this element to insert an operator, a function block or a program. By default STEP AS inserts components with the name ???. You can edit this field directly by clicking and entering the function block name. Alternatively, you can open the input assistant and click  to select a function block.

In the case of a function block, STEP AS additionally displays an input field (???) above the function block symbol. You must replace this name with the name of the function block instance. If you instantiate a function block with constant input parameters, the function block element displays the "Parameters..." area in the lower left corner. You can edit parameters by clicking on this area.

In order to replace an existing box, you just need to replace the current insertion identifier with the desired new name. When doing so, please note that STEP AS will take the number of input and output pins defined by the POU and existing assignments may therefore be deleted.

◆ CFC element 'jump'

symbol: 

Use this element to define a position where program execution continues. You must define the target location with a label. To do this, enter the name of the flag in the input field ???. If the corresponding label has already been inserted, you can also select it via the input assistant ().

◆ CFC component "tags"

symbol: 

With the help of jump elements, a label defines where program execution jumps to.

In online mode, STEP AS automatically inserts a return flag at the end of the CFC function block.

◆ CFC element "return"

symbol: 

Use this element to exit the function block.

Note that return elements in CFC editing in online mode are automatically inserted before the first line and after the last element. In single stepping, STEP AS will automatically jump to the return element at the end before exiting the function block.

◆ CFC component "manager"

symbol: 

Manager elements are used to handle structural components. The components of the structure are available to you as an input. To do this, you need to name the manager element (replacing ???) just like the structure.

The manager element is a copy of the selection element.

◆ CFC component "selector"

symbol: 

Selector cells are used to process structural components. The various components of the

structure are available to you as an output. For this purpose, you need to name the selector cell (replacing ???) just like the focus structure.



The selector element is a copy of the manager element.

◆ CFC element "annotation"

symbol: 

Enter comments using components in the CFC editor. Replaces placeholder text in components with annotation text. Use the shortcut key [Ctrl] + [Enter] to insert a newline.

◆ CFC Component "Connection Flag - Source/Sink"

symbol: , 

Instead of connecting lines between components, you can use connection markers. This helps you display complex graphics more clearly.

To achieve a valid connection, you need to connect the output of a component with [Connection Flags - Source](#) element and connected with the input of another element [connection sign - sink](#) element. Both flags must have the same name. Names are not case sensitive.

① Naming Considerations

- ① The standard name for a connection tag is C-<nr>.<nr> is from the serial number 1 started.
- ② You can rename standard names. To do so, you have to make sure to connect the tag-Source and connection flags-Shen has the same name.
- ③ If you change the connection tag-The source name, the target name are automatically renamed.
- ④ if you change the connection mark-sink, the source name is reserved.

◆ CFC component "input pin"

symbol: 

Depending on the type of function block, you can add more inputs to the inserted function block element. To do this, you must select the function block element and drag the input element of the function block to the body of the function block.

① NOTE: You can press [Ctrl] key-drag an input or output to connect to another location on the function block.

◆ CFC component "output pin"

symbol: 

Depending on the type of function block, you can add more outputs to the element inserted into the function block. To do this, you must select the function block element and drag the output element of the function block to the body of the function block.

① NOTE: You can do this by pressing [Ctrl] key-drag an input or output to connect to another location on the function block.

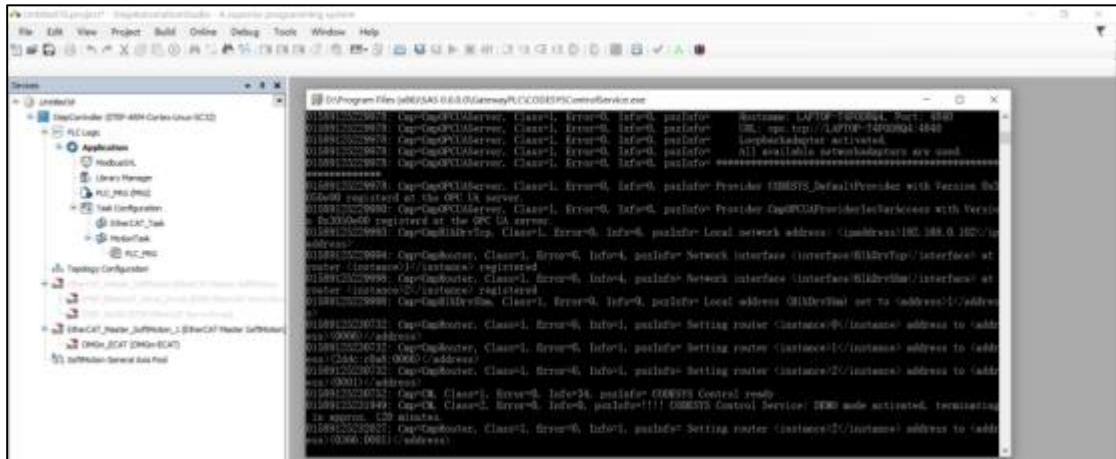
第九章 Convenient features of STEP AS

9.1. Quick access to simulation functions

Start/stop softmotion simulation in the toolbar



Icon is red when not started , green after startup .

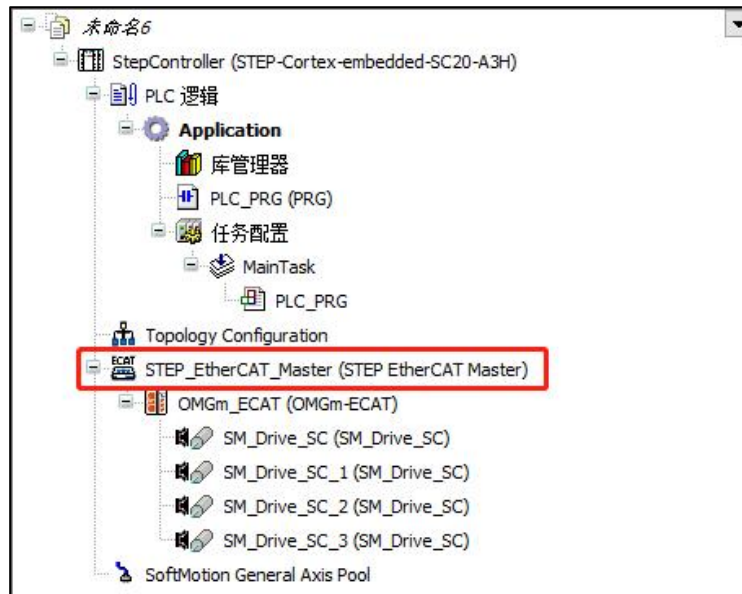


9.2. Engineering Comparison Quick Access

When the user downloads the program, the system will automatically back up the downloaded program, and when the user clicks the icon on the toolbar, the project comparison detailed information page will pop up.

9.3. 402 axis automatic addition function

STEP AS can automatically add 402 axes to the EtherCAT slave station, and add the number of supported axes according to the xml of the slave station. SC20 and A660 controllers only support the autonomous ethercat master station, and the axis device added under the slave station is also an autonomous xml axis.



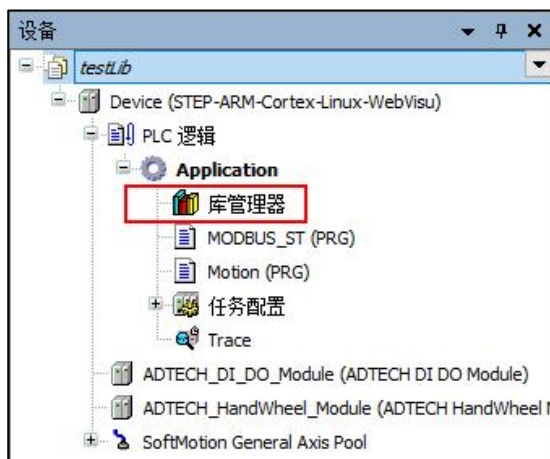
第十章 management library

Library management includes installation, editing, and deletion of library operations. The newly added library can be installed in the storage location of the system library or in the project. In the corresponding dialog, you can define – the storage location of the newly added library, and licenses can be installed and uninstalled. Some libraries cannot be used until a license is installed.

10.1. Query the library that needs to be used in the project

The currently installed libraries are displayed in the library manager. Users can click the corresponding library to know its data type, FB, and function information.

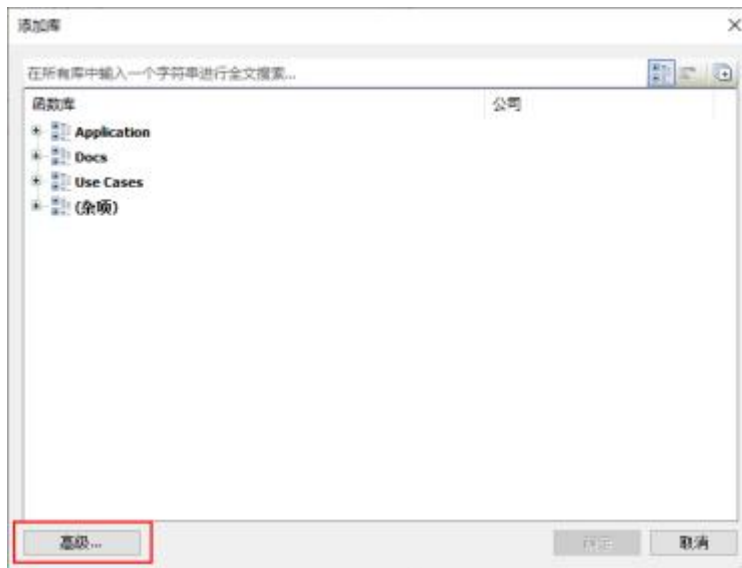
In the navigation bar window, double-click to open the library manager



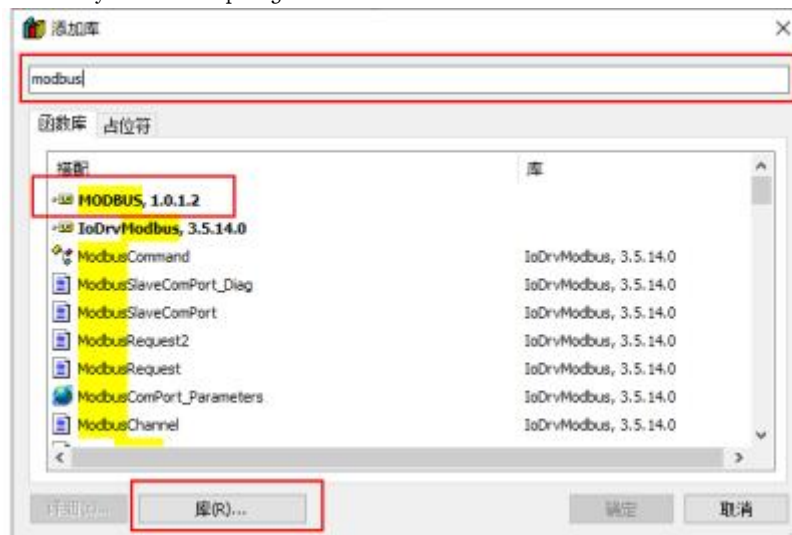
In the library manager, select Add library



In Add Library, select Advanced



Add the library to the project in the window

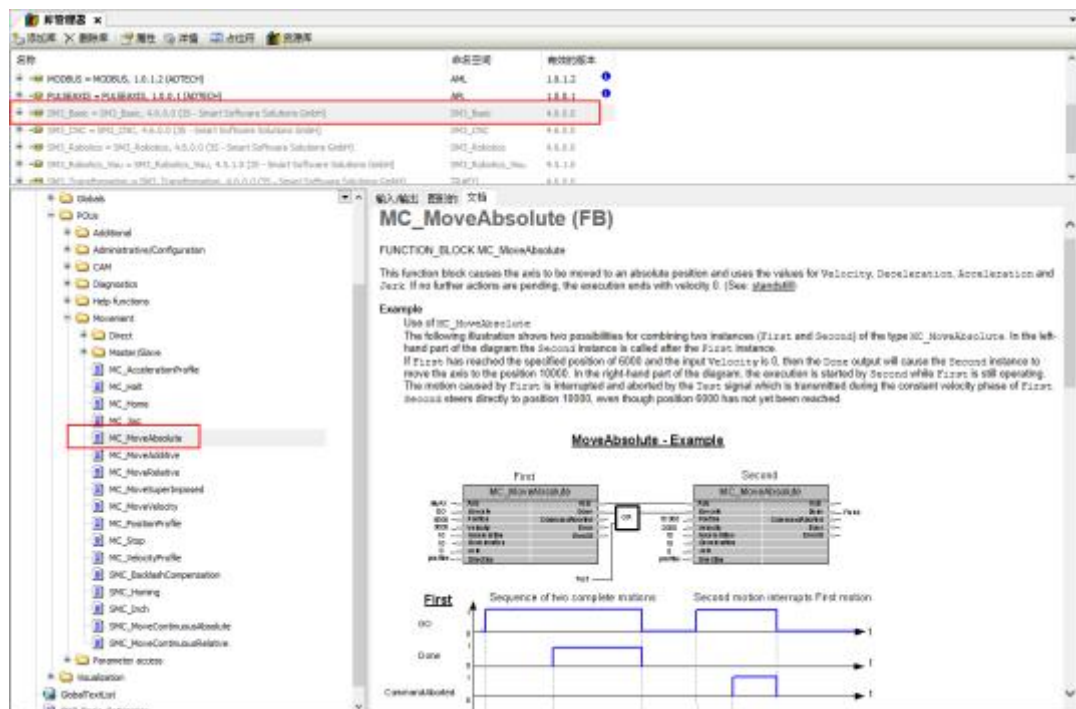


You can enter the keyword of the library in the edit box in the window, and then select the desired library in the list box to add.

Among them, the button 'Library (R)...' can open the library installation window, which is convenient for instant installation of the library.

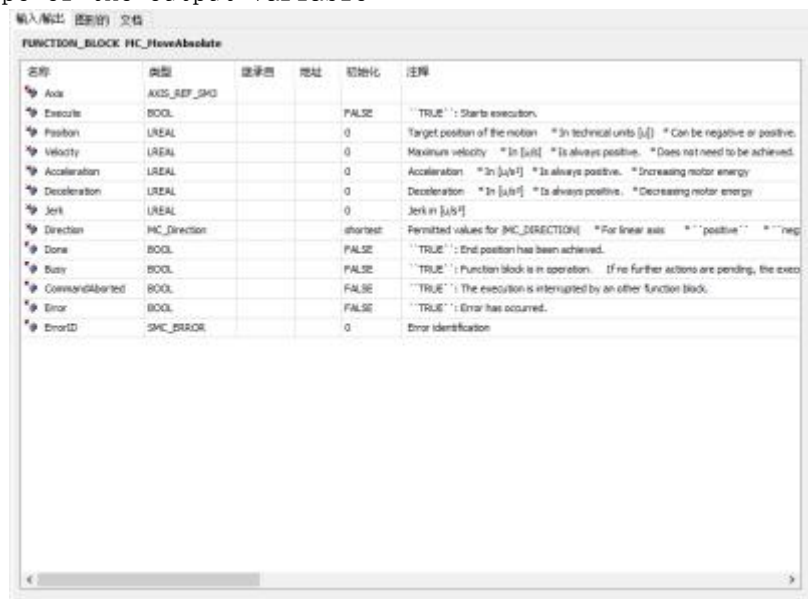
10.2. View the library's capabilities

You can select the corresponding library in the library manager, and find the corresponding function on the left side of the lower window and select it. At this time, you can view the relevant description of the function on the right side.

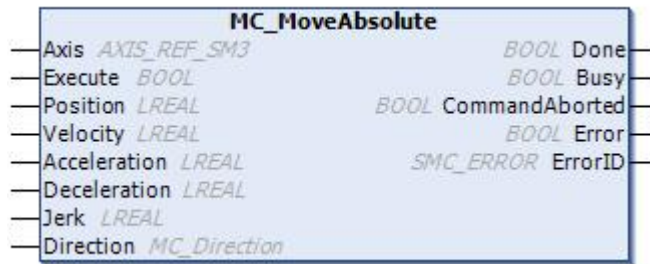


There are 3 specific instructions:

- ① enter/output Here, you can view the input of the function block/The associated type of the output variable



- ② graphics You can view the input and output pin definitions of this function block



- ③ Documentation If the function block in the library has a description document, you can view some usage instructions of the function block in this window (as shown in the above library manager, the right window).

10.3. Add library to application

Below, explain how to add the Util library to the application.

1. Select library manager, Open **project** → **edit object**.
⇒ Open the library editor in the editor.
2. click **library** → **Add library**
⇒ Open Add library dialog.
3. Browse this library by entering the string 'util' in the input field.
⇒ UtilThe library appears in the library view
4. Select the Util library, then close the dialog by clicking OK.
⇒ add in library managerUtillibrary.

10.4. Add library to repository

Below, the instructions describe how to install a library in the repository.

1. choose **tool** → **repository**.
⇒ Open repository dialog.
2. Click the Install button.
3. Select the library you want to install. You can set file filters.

Click to open.

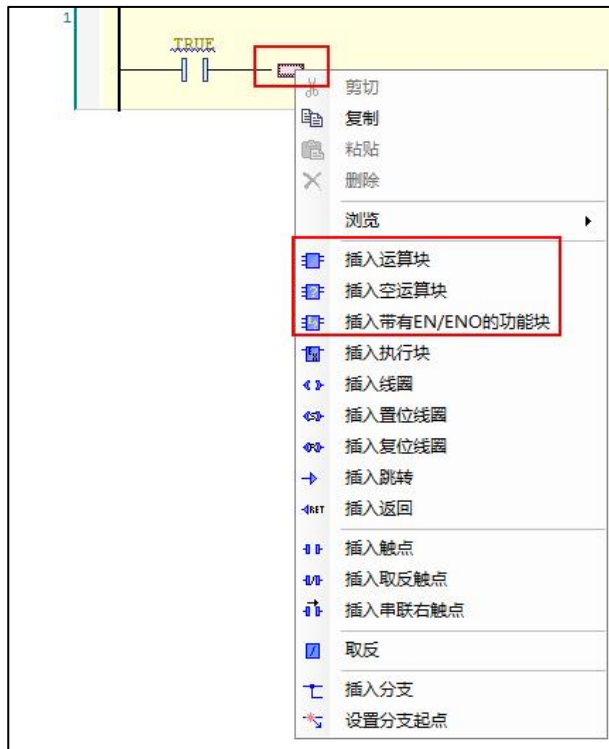
⇒ This library is added to the repository. The library can now be added in the library manager.

10.5. Use the library in your program

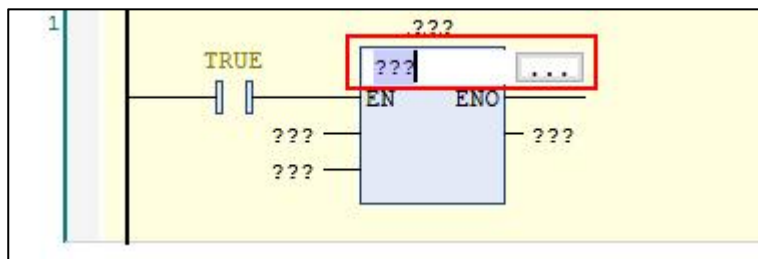
10.5.1. Graphical programming calls such as LD

Right-click, where you need to insert, and select several insertion methods as shown in the figure below

❶ If, choose Insert empty operation block, insert with EN/ENO function block

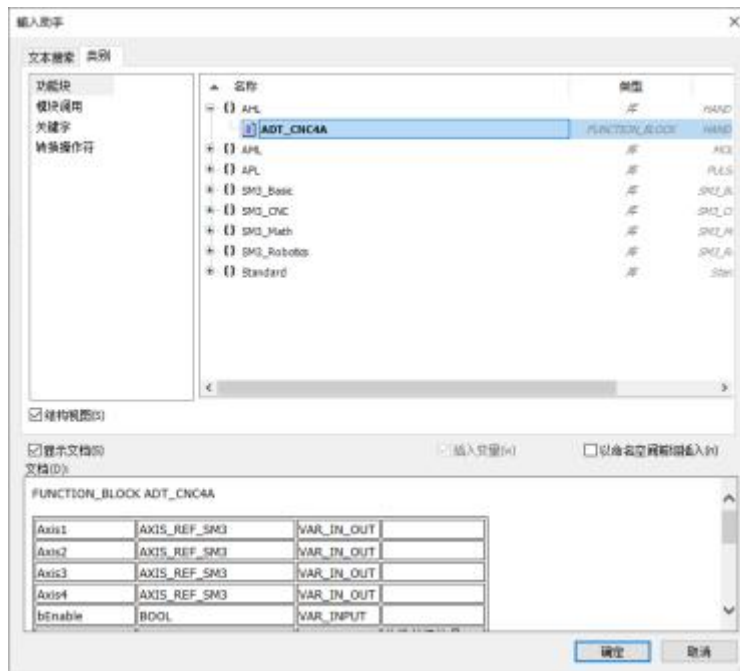


After insertion, an operation block will be inserted into the program

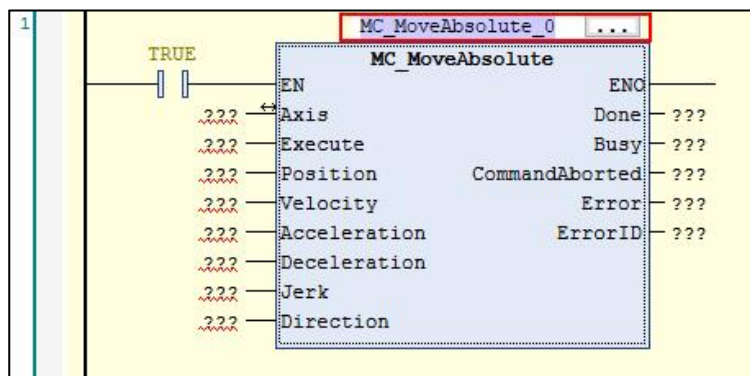


At this point, you can enter the function block name in the edit box, or click the select button to open the input assistant for selection.

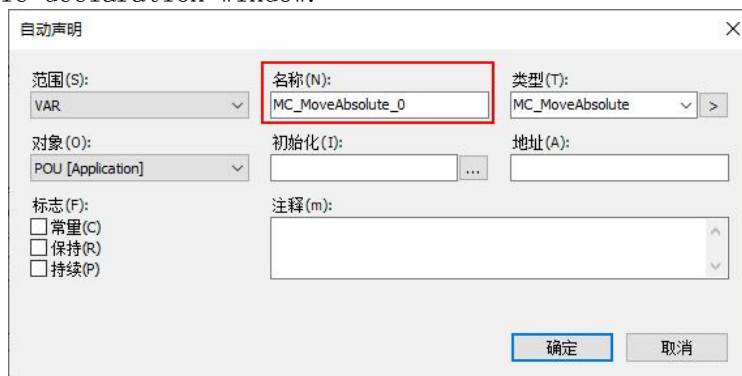
❷ If you choose to insert an operation block, it will be opened directly input assistant offer choice



After selecting the corresponding function block, you can define the function block (manually editable) as shown in the figure below.



After the function block variable is defined, click the blank space of the program to activate the automatic declaration window.



After clicking OK, the program will automatically declare the function block in the variable window

```

1 PROGRAM POU
2 VAR
3     MC_MoveAbsolute_0: MC_MoveAbsolute;
4 END_VAR
5

```

10.5.2. ST language call

Place the mouse on the blank space of the program editing window, right-click, and select Input Assistant



- *Note: You can place the mouse cursor at the position to be inserted, and use the shortcut key 'F2' to open the input assistant.*

After the addition is complete, the code block shown in the following figure will be added to the program

```

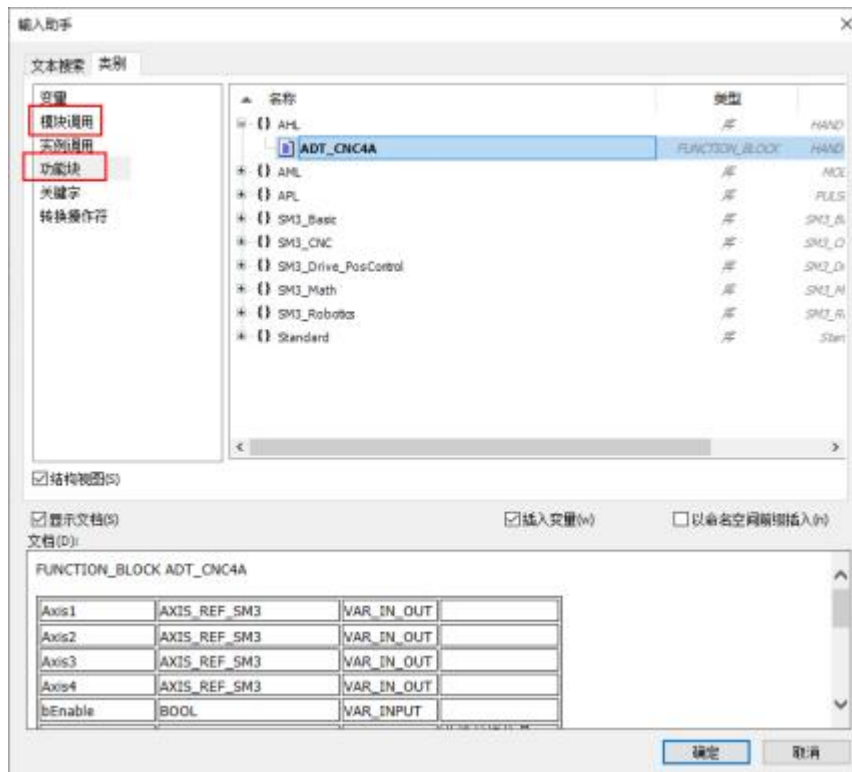
1 MC_MoveAbsolute(
2     Axis:= ,
3     Execute:= ,
4     Position:= ,
5     Velocity:= ,
6     Acceleration:= ,
7     Deceleration:= ,
8     Jerk:= ,
9     Direction:= ,
10    Done=> ,
11    Busy=> ,
12    CommandAborted=> ,
13    Error=> ,
14    ErrorID=> );

```

At this time, the name of the function block can also be modified.

At this time, you need to manually modify the name of the function block in the variable window, or use the shortcut key 'SHIFT + F2' to open the automatic declaration window to declare the function block.

10.5.3. input assistant



❶ For details, please refer to the chapter `input assistant`.

Users can filter library functions/function blocks that they want to filter by type, or find and insert by text search.

10.6. development library

The following library types can be created:

- a) *.libraryImplementation library
- b) *.compiled-library:Protection library;Resource code is not freely accessed
- c) *._Itfs.library: Interface library
- d) * Cnt.library:container library

To enable precise access to integrated objects, library namespaces can be defined. Increase the precision of access by adding namespaces before the module name.

Repositories (dongles) can be protected by using a license. If you use protected library modules, a valid licensed dongle must be inserted into the computer.

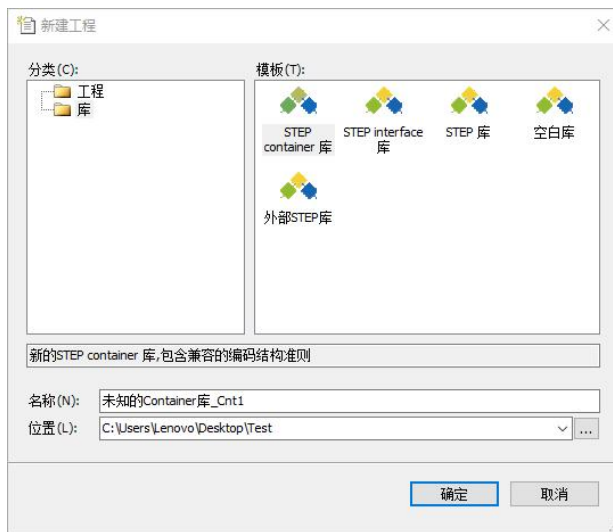
External libraries can be integrated into the application. external library isSTEP ASprogramming in a different programming language, e.g.Clanguage.

10.6.1. Development library example

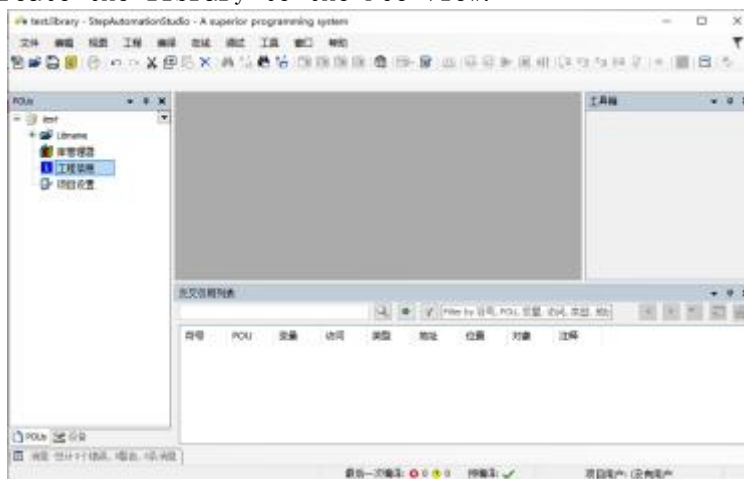
This section describes the steps to create a new library project and install it into the library repository.

1. select from the menu *document*→*New Construction*.
2. select category "*Library*", select "*Blank library*". The dialog box shown below is displayed.

Enter the name and click the [OK] button.



The project for which the library has been created. A library's project file has the extension library. For libraries, the POU view is displayed in the Navigator. Add the objects needed to create the library to the POU view.



3. Double-click the "Project Information" object of the POU view. Displays the Project Information dialog box.

Please change "Company", "Title", "Edition" as needed. This information is displayed on the selection screen when adding the library to be created to the project.

If "Publish" is checked, a confirmation message will be displayed when attempting to change the library.

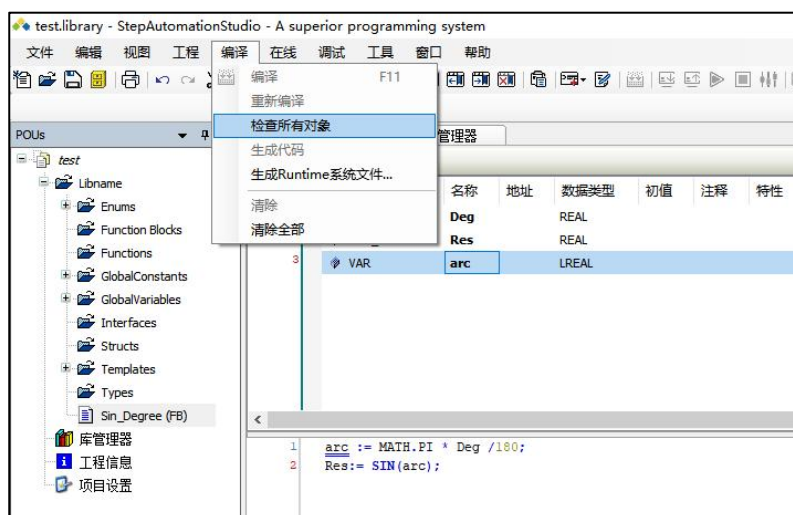
Click the [OK] button. The project information confirms the settings.

4. Right-click the object with the filename at the top of the navigator and select Add Object from the menu that appears→POU. Select "Function Block", enter a name and select a description language.



5. Enter the program for the function block.

After entering the program, execute the compile in the menu→**Check all objects**, for syntax checking. If an error is displayed after executing this command, the program must be changed, and then executed again to check all objects.

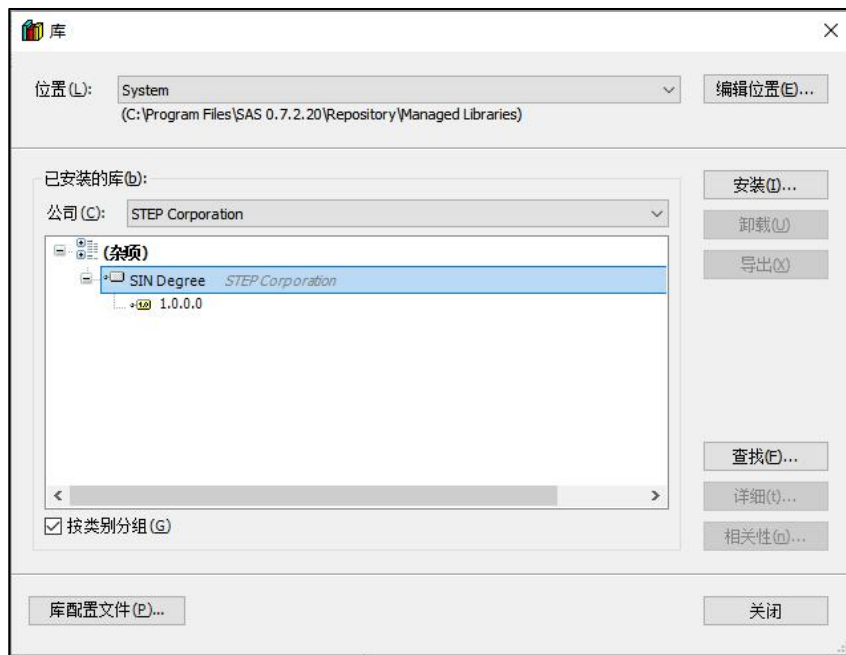


6. select from the menu **document**→**Save the project**, and load the library. The created library is saved to the local repository.

7. select from the menu **tool**→**library**.

show "library" Dialog. In "installed library" column to confirm that the created library is

installed.



Click the [Details] button to check information such as function blocks included in the library. At this point, the steps to install the library into the library repository are complete.

① Note: The source code of the installed library can also be viewed from the project that uses the library. If you do not want to view the source code, in Step10, please select the file in the menu → **Save the project as a compiled library**. Libraries will be saved as compiled libraries (.compiled-library files).

Compiled libraries do not need to be installed into the repository. Click the [Install] button in the "Library" dialog box, and then select the saved compiled library file.

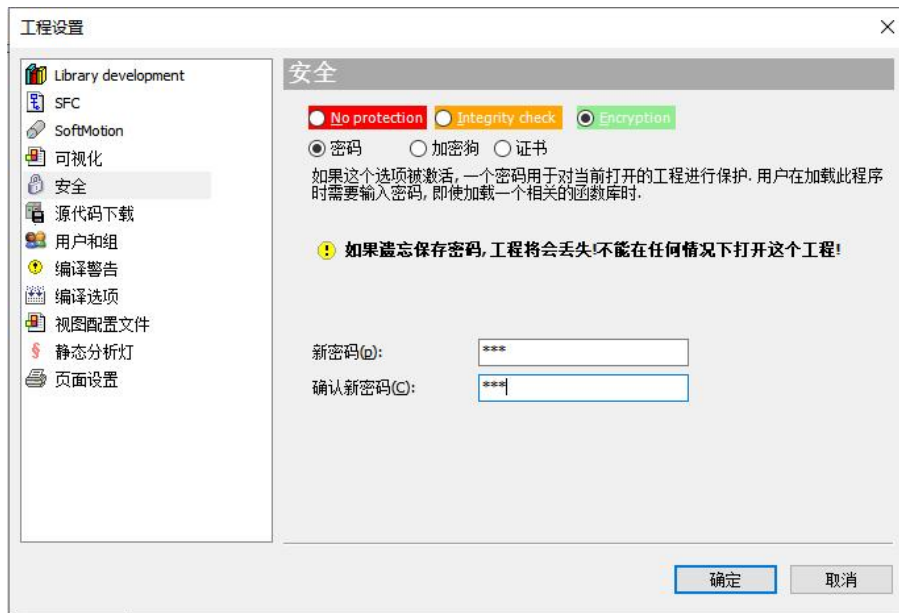
10.6.2. library version

You can install multiple versions of the library to the system at the same time. And can use a specific version.

You can integrate multiple libraries in a project at the same time (no priority).

10.6.3. library encryption

In the project settings of the development library, encryption can be set.



第十一章 security function

This section describes security-related functions and operation procedures such as user management and project encryption.

project	illustrate
User Management	The execution authority can be set for each group registered by the user for operations such as the execution of menu commands, addition, editing, and deletion of objects. Permissions can be set for each user for logging in to the device, allowing login by entering a password.
Encrypt/Sign	Can be encrypted with the password of the project file
write protection	The project file can be write-protected to prevent accidental changes to the project file due to misuse, etc.

11.1. User Management

It is divided into project user management and device user management.

11.1.1. Project user management

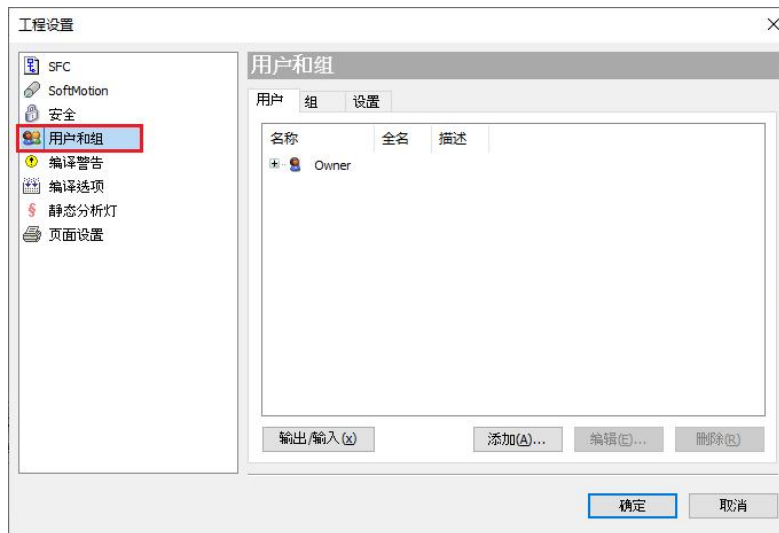
When creating a new project, the Owner group and Everyone group will be registered, and the Owner user will be generated synchronously.

Group	user	Remark
Owner	Owner	All operations can be performed. User: Owner's password is blank.
Everyone	Owner	All users will be automatically registered.

11.1.2. Create new users and groups

Operation example:Create a new group (group name: GroupA) and a user belonging to the group (user name: Fred) and set permissions so that users belonging to GroupA can access the POU object: "POU_1".

1. Select the menu bar **project** → **Project settings**. exist "Project settings" Select in dialog "users and groups" category.



2. Click [Add to] button.

show "Add user" dialog. Enter the information of the newly added user, such as Fred. Click the [OK] button.



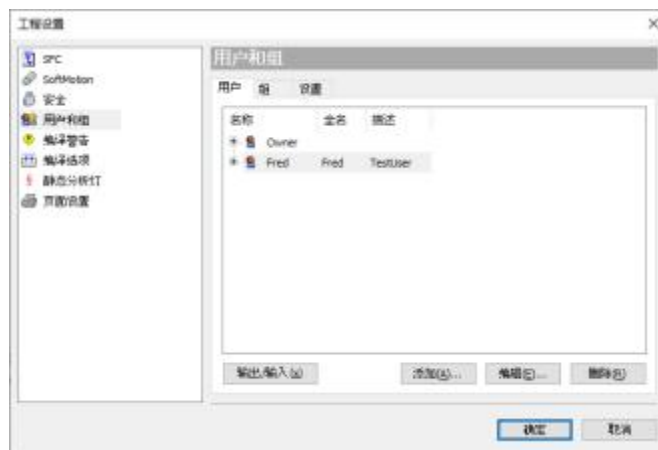
The "Login" dialog box is displayed. To add a new user, you need to log in as the Owner user.

3. Enter "Owner" in the "Username" field.



4. The initial password for "Owner" is set to blank. Please leave the "Password" field blank. Click the [OK] button.

Logging in as the Owner user is complete, and a new user (Fred) is added to the "Users" tab screen.

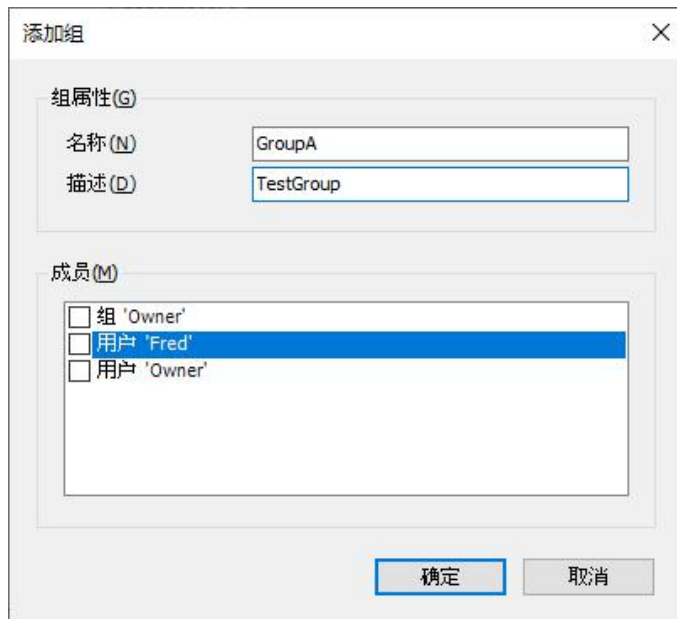


The logged in user name (Owner) is displayed in the Status field.



5. Select the "Group" tab and click the [Add] button. Displays the Add Group dialog. Enter the information for the newly added group.

Please enter information about the newly added group GroupA. Specify the members who belong to this group in the "Members" column. Select the newly added user in "Step 4".



6. Click the [OK] button.

Added a newly added group to the "Groups" tab screen. Register user: Fred as a member in GroupA.

7. Click the [OK] button.

The Project Settings dialog closes.

At this point, the user Fred's registration and registration to the GroupA group has been completed. The groups and users after the steps are completed are shown below.

Group	user	Remark
Owner	Owner	A group that can perform all operations.
Everyone	Owner, Fred	A group to which all users are automatically registered.
GroupA	Fred	Newly added group.

§ User and group information can be exported in XML format. Click the [Export/Import] button in the user and group settings screen, and then select the "Export Users and Groups" menu. The "users" file can be exported.

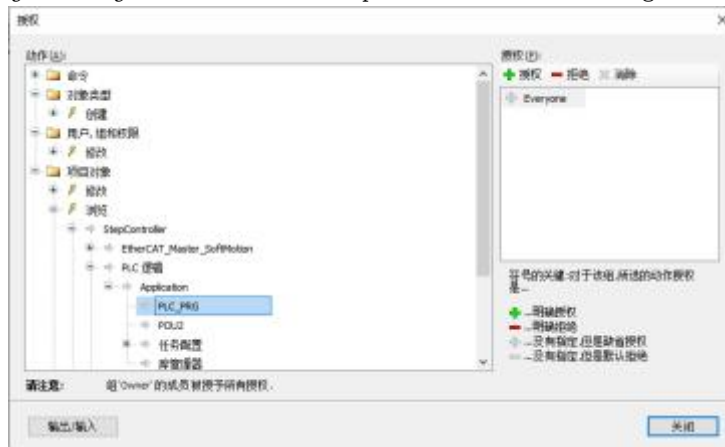
§ Select the "Import Users and Groups" menu to import the ".users" file.

11.1.3. Set operation permissions

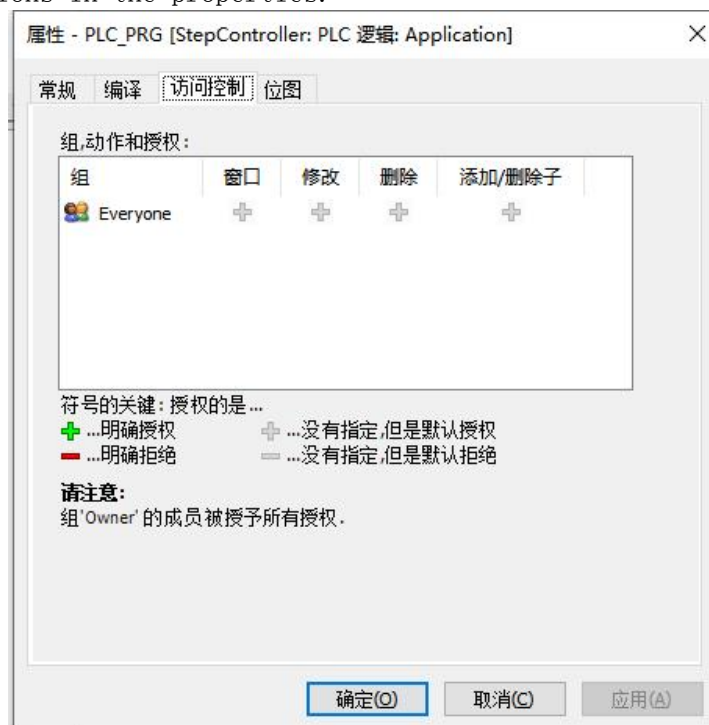
Set permissions on a single program PLC_PRG object to make it visible to users belonging to GroupA. Before performing the following steps, add the PLC_PRG object to the project.

1. Select Project → User Management → Authorization in the menu bar. The "Authorization" dialog is displayed.

2. Select the action that grants the permission from the Action column.
Select Project Object→Browse→StepController→PLC Logic→Application→PLC_PRG.



3. Set permissions in the "Authorization" column.
If the "Login" dialog box appears, enter "Owner" in the "Username" field, leave the "Password" field blank, and log in.
4. Click the [Close] button.
In addition, you can directly select the object to be set from the left view, and set the permissions in the properties.



① The settings of the operation authority can be exported in XML format. Click the [Export/Import] button in the "Authorization" dialog box, and then select the "Export all permissions" menu, or the "exportselected permissions" menu. "Can export .perms" files.

Select the "Import Permissions" menu to import ".perms" files.

① 10.3.4 Execute the action for which the permission is set

11.1.4. User login and logout

If there is a logged in user, select the [Project→User Management→User Logout](#), to perform logout.



Double-click an object with permissions set on the navigation bar window and you will be prompted to log in. or directly [Project→User Management→User Login](#)

Object display operations are restricted and the "Login" dialog box is displayed.

▶▶ Enter the "User Name" field and "Password" field, and then click the [OK] button.

▶▶ "[10.1.2 Create new users and groups](#)" Enter the username and password of the added user in. The login is complete and the POU_1 object is displayed.



The logged in username is displayed in the Status field.

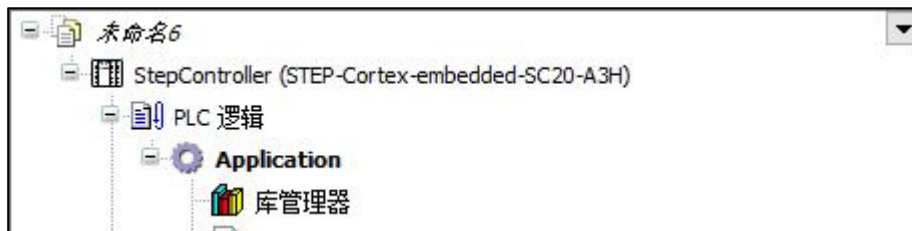
11.1.5. Device user management

Device users are users for device operations, and only device users with permissions can log in.

User: Administrator, Password: Administrator The user is pre-registered as a device user. (When logging in as the Administrator user for the first time, you need to set an arbitrary password.)

① Connects PCs and SC series controllers with STEP AS installed.

Double-click the [Device] object StepController on the navigation bar window.



The Device setting screen is displayed.

② Click the "Users and Groups" tab. The "Users and Groups" screen is displayed.



③ Click [🔄] icon (Synchronization). A confirmation dialog is displayed.

④ Click the [Yes] button. Displays the Device User Login dialog box.

⑤ Enter "Username" and "Password".

Enter Administrator in the "Username" field and Administrator in the "Password" field.



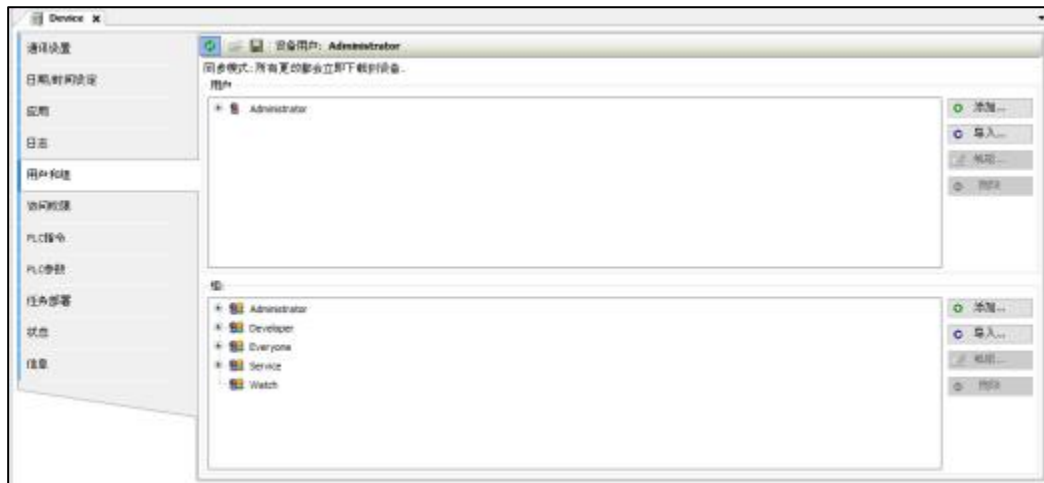
⑥ Click the [OK] button.

⑦ Enter an arbitrary password.

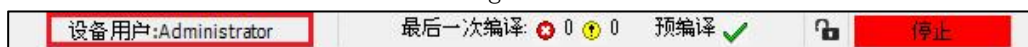
Please enter an arbitrary password to set the password of the Administrator user. If you forget your password, you will not be able to log in to the device.

⑧ Click the [OK] button.

A password for the Administrator user has been set and logged in as the Administrator user.



⑨ Select Online from the menu → Login to



● To log out the user while the user is logged in, select the **Online → Security → Logout current online user**.

● Device users can be added, deleted, and passwords changed in the Users and Groups screen.

Add device users: 添加...

To delete a device user: 删除

To change the device user's password: 编辑...



● Users registered in the project user management can be imported as device users. 导入... Clicking the button will display the "Import Users" dialog. Select the users to import, and then click the [OK] button. Passwords managed by project users are not imported at this time. Click the [Edit] button on the "Users and Groups" screen to set the password of the imported user.

● Device user management information can be exported.

On the "Users and Groups" screen, click icon (Export to Disk). Files can be saved in XML format (".dum" files).

To import the exported ".dum" file, click icon (Import from disk).

- ◆ Device user management information can be initialized by device reset.
- ◆ If you forget your password, you will not be able to log in to the SC series controller. In this case, reset the SC series controller. For the reset method of the SC series controller, refer to the SC series series user's manual (hardware).

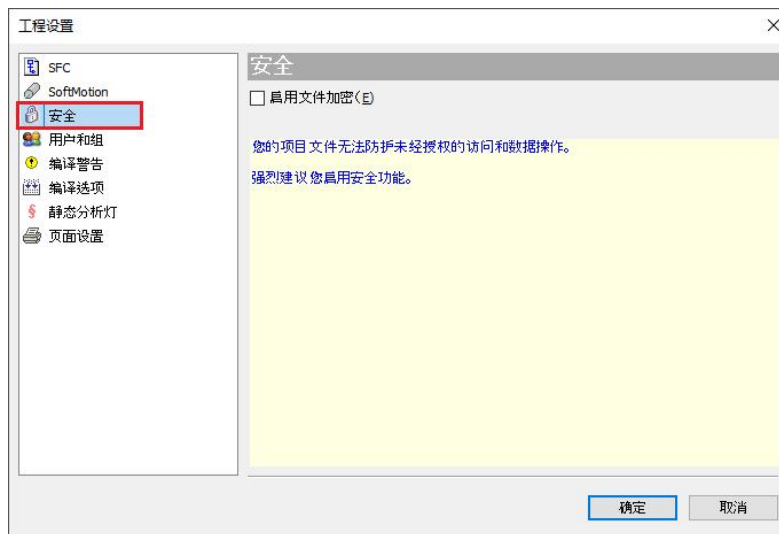
11.2. encryption

This section describes how to encrypt project files.

11.2.1. Encrypt project files

Project files can be encrypted with a password. If a password is set. A password will be required to open the project file.

1. Select Project → Project Settings in the menu bar.
2. Select the "Security" category in the "Project Settings" dialog.



The "Security" screen is displayed.

3. Tick "Enable file encryption" and enter the password.
4. Click the [OK] button.

At this point, the password setting has been completed.

When opening a project file, a screen asking for a password will be displayed. In this case, please enter the set password.

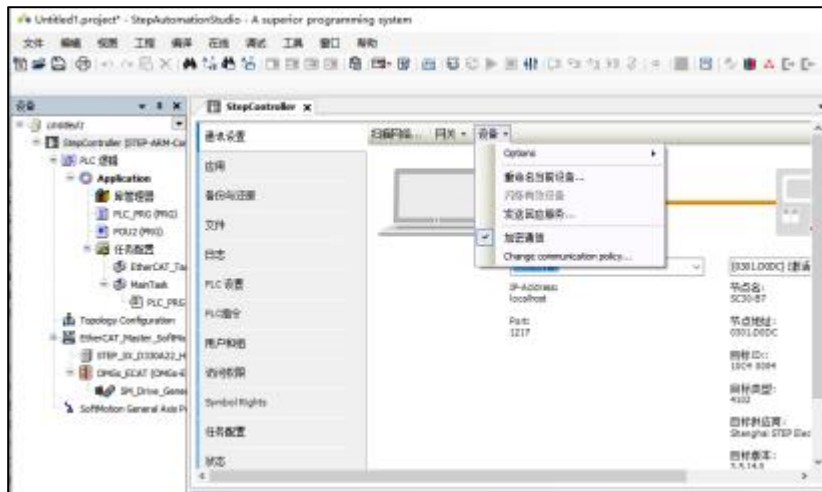
11.2.2. Encryption of communication lines

The communication between STEP AS and STEP controller can be encrypted using certificates. This section describes how to encrypt communications with the certificate held as a trusted certificate.

1. Double-click the [Device] object on the navigation bar window.

The Device setting screen is displayed. Open the "Communication Settings" tab.

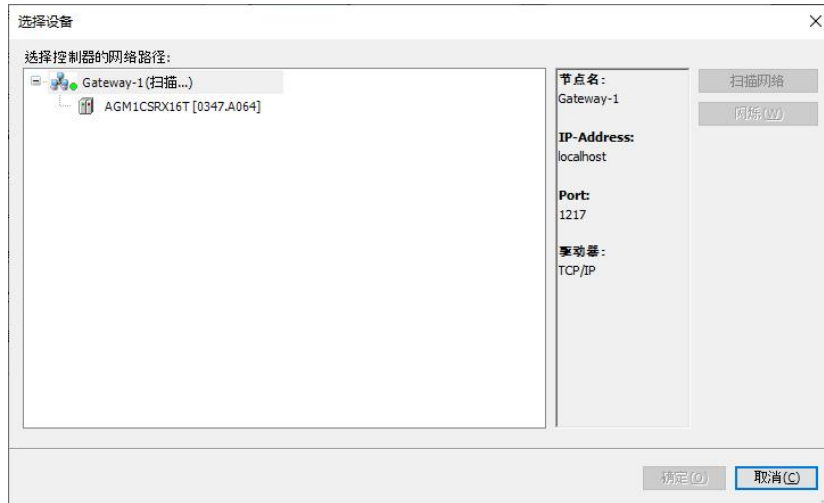
2. Enable "Encrypted Communication" in the device menu.



When "Encrypted Communication" is enabled, the connection wires between the IDE, gateway and controller are shown in yellow.



3. Click the Scan Network menu. Displays the Select Device dialog.



4. Select the connected controller and click the [OK] button.

A message appears stating that the controller's certificate has not provided a trusted signature for encrypted communications

5. When the [OK] button is clicked, the communication can be encrypted by installing the certificate displayed in the message as a trusted certificate in the local "Controller Certificate" storage area on the PC.

The registered controller certificate can be confirmed with certmgr.msc in the C:\Windows\System32 folder.

If the controller's certificate is used as a trusted certificate, the certificate is valid for 30 days.

① If the certificate has expired, the expired message will be displayed in step 4 above.

The certificate period can be extended by clicking the [OK] button.

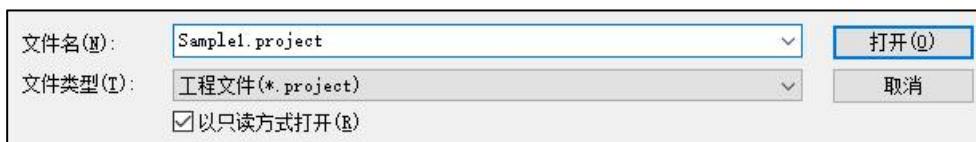
11.3. Security feature: write protection

This section describes how to write-protect the project file to prevent accidental changes to the project file due to misoperation, etc.

11.3.1. Open as read-only

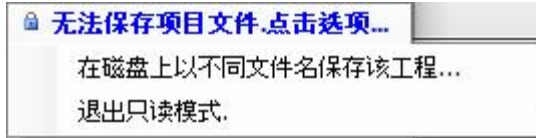
Open the project file as read-only.

When selecting an open project file, please check "Open as read-only file".



Unable to save when opened as read-only file.

To save the project file, select "Unable to save project file. Click Options" displayed in the menu bar and select the displayed menu.



project	illustrate
Save the project on disk with a different filename	Project files can be saved as writable files.
Exit read-only mode	Keep the project file open read-only.

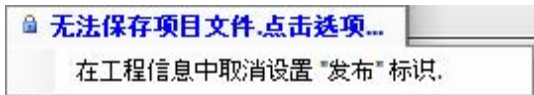
11.3.2. Set the published flag

Set the published flag in the project information of the project file.

Project files with the published flag set cannot save changes.

Select the menu bar **Project**→**Project Information**, open the screen of the "Summary" tab, and check "Publish".

To save a project file with the published flag set, select "Unable to save project file. Click Options" displayed in the menu bar, and select the displayed menu.



After-sales service

Repair and maintenance

1. For repair and maintenance, please contact the product agent first;
2. If the product has been installed in the equipment, please contact the equipment manufacturer first.

Technical Services

Customer technical consultation

Tel: (86) 13917890469 (Zhong Gong)

Consultation time: Monday to Sunday 9:00--17:30 (except specific holidays)

After-sales technical and maintenance consultation (repair of faulty parts, purchase of repair parts and optional accessories)

After-sales support: 400-168-2718

Purchase Inquiry: 13925286547Manager Zhou

Consultation time: Monday to Sunday 9:00--17:30 (except specific

holidays)

Internet technical information

