INOVANCE

FORWARD, ALWAYS PROGRESSING







Medium-Sized PLC Programming Software

User Guide

Industrial Automation | Intelligent Elevator | New Energy Vehicle | Industrial Robot | Rail Transit

Data code 19010980 B06

Preface

Introduction

This guide covers medium-sized programmable logic controllers (PLCs) of Inovance, including the AM, AC, and AP series. This guide introduces network configuration, programming environment, programming languages, program diagnosis, and other information for programming the medium-sized PLCs in InoProShop.

More Documents

Data Code	Doc Name	Description
19012378	Medium-Sized PLC Programming Guide (Motion Control)	Describes the composition of the PLC motion control system, mechanism of the motion control program, MC instructions, and simulation and debugging related operations.
19012377	Medium-Sized PLC Instruction Guide	Describes basic instructions of medium-sized PLCs.

Revision History

Date	Version	Description
December 2023	B06	**Addition**: • Added *" Project Safety Management" on page 60*. • Added *" Adding an Object Through Application" on page 63*. **Modification**: Updated *"9.8.8 Axis Diagnosis Code" on page 518*.
December 2023	B05	**Addition**: • Added *"3.8 Project Version Upgrade" on page 58*. • Added *"4.2.6 I/O Mapping Parameters" on page 127*. • Added *" Expansion Card Configuration" on page 131*. • Added the "Disabling a GL20-series I/O module" section in *"4.4.10 I/O Module" on page 186*. • Added *" CAN Free Protocol" on page 288*. • Added the "EIP configuration of global variable list" section in *"4.11.5 Configuration of PLC as the EtherNet/IP Slave" on page 324*. • Added *"5.5.2 Variable Definition" on page 370*. • Added *"5.5.4 Persistent Rules" on page 373*. • Added the "Batch update block calls" and "Update block calls" sections in *"6.3.6 LD Menu Commands" on page 416*. • Added *" SVN Function" on page 524*. **Modification**: • Added configuration of modules of the GL20 series in *"4.2.4 I/O Module Configuration" on page 81*. • Updated *"4.2.5 High-Speed I/O Configuration" on page 109*. • Optimized the sections "Display the project differences" and "Check whether the device alias takes effect after a scanned device is copied" sections in *"4.4.2 Common Functions" on page 131*. • Updated *"7.3 Fault Diagnosis" on page 440*. • Updated *"9.8.7 EtherCAT Diagnosis Code" on page 515*.
August 2023	B04	• Added guidance on upgrading the user program. • Optimized description of some functions.

Date	Version	Description
September 2022	B03	• Added the functions of online diagnosis and axis diagnosis. • Added axis diagnosis codes. • Updated EtherCAT diagnosis codes.
January 2022	B02	• Added functions related to ST programming and compiling. • Optimized EIP and variable definition functions.
November 2020	B01	• Added the HMC_Reset and HMC_TouchProbe instructions in section 6.6. • Corrected details of the earlier version.
October 2020	B00	• Added the section "3.9 EtherNET Communication". • Added the "Appendix I Synchronizing the Project Information". • Added the "Appendix J Solutions to PLC Running Errors". • Corrected details of the earlier version.
April 2018	A00	First release

Access to the Guide

This guide is not delivered with the product. You can obtain the PDF version in the following way:

- Visit *www.inovance.com*, go to "Support" > "Download", search by keyword, and then download the PDF file.
- Scan the QR code on the product with your smart phone.

Warranty

For faults and damage incurred during normal use in the warranty period, Inovance provides free repair service. (For details of the warranty period, see the purchase order.) A maintenance fee will be charged out of the warranty period.

Even in the warranty period, a maintenance fee will be charged for repair of the following damage:

- Damage caused by operations not following the instructions in the guide
- Damage caused by fire, flood, or abnormal voltage
- Damage caused by unintended use of the product
- Damage caused by use beyond the specified scope of application of the product
- Damage or secondary damage caused by force majeure (natural disaster, earthquake, and lightning strike)

The maintenance fee will be charged according to the latest Price List of Inovance if not otherwise agreed upon.

For details, see the Product Warranty Card.

Table of Contents

Preface....1

1 Product Information....11

1.1 Overview....11

1.1.1 Product Information....11

1.1.2 Product Configuration and Module Description....16

1.1.3 System Application Process....21

1.2 Overview of InoProShop....22

1.2.1 Brief Introduction....22

1.2.2 Connection Between InoProShop and Hardware....22

1.2.3 Acquisition and Installation of the Software....23

1.2.4 Installation Procedure....24

1.2.5 Uninstallation of InoProShop....28

2 Quick Start....29

2.1 Programming Environment Launching....29

2.2 Typical Procedure for Writing a User Program....31

2.2.1 Overview....31

2.2.2 User System Configuration Operations....32

2.2.3 User Program Writing Operations....32

2.2.4 Linkage Configuration Between User Program Variables and Ports....34

2.2.5 Configuration of Execution Mode and Operation Cycle of User Program....34

2.2.6 Compiling, Login, and Download of User Program....34

2.3 Writing a Marquee Sample Project with InoProShop....37

2.4 How to Log in to the Main Module....41

2.4.1 Prerequisites and Operations of Main Module Login....41

2.4.2 Scanning Medium-Sized PLC in InoProShop....42

2.4.3 Solution to Device Scanning Failure....44

3 Basic Functions....46

3.1 Page Navigation....46

3.2 Compiling a Command....46

3.3 Resources List....47

3.3.1 Overview....47

3.3.2 Features....47

3.4 Symbol Configuration....51

3.5 Cross References....56

3.6 Watch List 57
3.7 Going to a Lower Level 57
3.8 Project Version Upgrade 58
3.9 Project Safety Management 60
3.9.1 Project File Encryption 60
3.9.2 Project User Authorization Management 60
3.10 Adding an Object Through Application 63
4 Network Configuration 66
4.1 Device Configuration 66
4.1.1 Device Configuration 66
4.1.2 Network Configuration 66
4.1.3 Hardware Configuration 73
4.1.4 Device Tree Operations 75
4.1.5 Configuration Compiling Error Locating 76
4.2 CPU Configuration 76
4.2.1 Overview 76
4.2.2 General CPU Configuration Procedure 76
4.2.3 CPU Parameter Configuration 77
4.2.4 I/O Module Configuration 81
4.2.5 High-Speed I/O Configuration 109
4.2.6 I/O Mapping Parameters 127
4.3 Expansion Card Configuration 131
4.4 EtherCAT Configuration 131
4.4.1 Overview 131
4.4.2 Common Functions 131
4.4.3 EtherCAT Master 142
4.4.4 EtherCAT Slave 151
4.4.5 CiA402 Axis 168
4.4.6 Virtual Axis 178
4.4.7 GR10-4PME Positioning Module 179
4.4.8 GR10-2HCE counter module 181
4.4.9 Splitter 184
4.4.10 I/O Module 186
4.4.11 Library (Implicit Variables) 189
4.5 Modbus Device Editor 202
4.5.1 Serial Hardware Port 202

4.5.2 Network Configuration 204
4.5.3 Modbus Master Configuration 209
4.5.4 Modbus Master Communication Configuration 210
4.5.5 Modbus Master Broadcast Configuration 214
4.5.6 Modbus Slave Configuration 215
4.5.7 Modbus Device Diagnosis 217
4.5.8 Common Errors of Modbus 217
4.5.9 Modbus Variable Addressing 217
4.5.10 Modbus Communication Frame Format 219
4.6 Application of Free Protocols on Serial Ports 223
4.6.1 Overview 223
4.6.2 Serial Port Configuration 223
4.6.3 Communication Configuration 226
4.6.4 Registers for Data Sending and Receiving 226
4.6.5 Communication Tests Through the Serial Port Commissioning Assistant 228
4.7 Modbus TCP Device Editor 229
4.7.1 Overview 229
4.7.2 Modbus TCP Master Configuration 229
4.7.3 Modbus TCP Master Communication Configuration 230
4.7.4 Modbus TCP Slave Configuration 233
4.7.5 Modbus TCP Device Diagnosis 234
4.7.6 Common Errors of Modbus TCP 235
4.7.7 Modbus TCP Communication Frame Format 236
4.8 CANopen Network 240
4.8.1 Overview of CANopen Communication 240
4.8.2 CANopen Master Configuration 245
4.8.3 CANopen Slave Configuration 250
4.8.4 CANopen Module 264
4.8.5 CANopen Parameter Configuration 265
4.8.6 Programming Interface 268
4.9 CANlink 3.0 Configuration Editor 268
4.9.1 Overview 268
4.9.2 CANlink3_en.0 网络组成 269
4.9.3 General Process of Using CANlink 270
4.9.4 CANlink Network Configuration 271
4.9.5 Network Management 274
4.9.6 Send Configuration 276
4.9.7 Receive Configuration 281

4.9.8 Synchronous Write by the Master 281
4.9.9 Local Slave Configuration 283
4.9.10 设备接入CANlink3_en.0 网络 283
4.10 CAN Free Protocol 288
4.10.1 Overview 288
4.10.2 Network Configuration 289
4.10.3 CAN Free Protocol Configuration 291
4.10.4 CANBus Library 294
4.10.4.1 Enumeration Types of CANBus Library 294
4.10.4.2 Structure Types of CANBus Library 297
4.10.4.3 CANBus Function Blocks 298
4.10.4.4 Error Codes of CANBus Function Blocks 305
4.10.4.5 Example of Using CANBus Function Blocks 309
4.11 EtherNet/IP Communication 311
4.11.1 Overview of the Protocol 311
4.11.2 EtherNet/IP Communication Specifications 312
4.11.3 Configuration of PLC as the EtherNet/IP Master 313
4.11.4 Programming Example for Configuration of PLC as the EtherNet/IP Master 318
4.11.5 Configuration of PLC as the EtherNet/IP Slave 324
4.11.6 Programming Example for Configuration of PLC as the EtherNet/IP Slave 331
4.11.7 Diagnosis of EtherNet/IP Communication State 333
4.12 PROFIBUS-DP Bus 336
4.12.1 Overview 336
4.12.2 General Process of Using PROFIBUS-DP 337
4.12.3 PROFIBUS-DP Master Configuration 338
4.12.4 PROFIBUS-DP Slave Configuration 339
4.12.5 PROFIBUS-DP Module 341
4.13 HMI Communication Configuration 341
4.13.1 Communication Configuration 341
4.13.2 Communication Example 343
4.13.3 Common Faults 344
5 Programming Basics 346
5.1 Overview 346
5.2 Direct Address 346
5.2.1 Syntax 346
5.2.2 PLC Direct Address Storage Area 347
5.3 Variable 348

5.3.1 Overview 348
5.3.2 Variable Definition 348
5.3.3 Variable Type 361
5.3.4 Variable Import and Export 367
5.4 Constants 368
5.5 Persistent Variable 369
5.5.1 Overview 369
5.5.2 Variable Definition 370
5.5.3 Persistent Variable Table 372
5.5.4 Persistent Rules 373
5.5.5 Persistent Mode 374
5.5.6 Address Assignment 376
5.5.7 Recipe Operations 379
5.5.8 Description 384
6 Programming Languages 386
6.1 Programming Languages Supported by InoProShop 386
6.2 Structured Text (ST) 386
6.2.1 Overview 386
6.2.2 Expressions 386
6.2.3 ST Instruction 387
6.2.4 ST Editor 394
6.2.4.1 ST Tool Kit 394
6.2.4.2 Intelligent Input 395
6.2.4.3 Folding and Indenting Functions 395
6.2.4.4 Page Colors of IEC Text Editor 396
6.3 Ladder Diagram (LD) 399
6.3.1 Overview 399
6.3.2 LD Elements 400
6.3.3 LD Editor Options 404
6.3.4 Element Selection 408
6.3.5 Standard Edit Commands 411
6.3.6 LD Menu Commands 416
6.3.7 Single-Key Command 427
6.3.8 Line Drawing Function 427
6.3.9 Drag and Drop 430
6.3.10 Graphic Display Tool 431
6.3.11 LD Debugging 433

6.3.12 LD Data Update 436
7 Diagnosis 438
7.1 Overview 438
7.2 Configuration Diagnosis 438
7.2.1 Overview 438
7.2.2 Network Configuration Diagnosis 438
7.2.3 Hardware Configuration Diagnosis 439
7.3 Fault Diagnosis 440
7.4 Online Diagnosis 444
7.4.1 Overview 444
7.4.2 Diagnosis Procedure 444
7.4.3 Scanning Devices 445
7.4.4 Logging in to PLC 446
7.5 List of Device Self-Diagnosis Information 447
7.5.1 CPU Diagnosis 447
7.5.2 EtherCAT Diagnosis 447
7.5.3 I/O Diagnosis 448
7.5.4 PROFIBUS-DP Diagnosis 448
7.5.5 Modbus RTU Diagnosis 452
7.5.6 Modbus TCP Diagnosis 453
7.5.7 CANlink Diagnosis 453
7.6 Diagnosis Programming Interface 453
7.6.1 Overview 453
7.6.2 Overview 454
7.6.3 CPU Diagnosis Programming Interface 455
7.6.4 CANopen Diagnosis Programming Interface 457
7.6.5 PROFIBUS-DP Diagnosis Programming Interface 457
7.6.6 CANlink Diagnosis Programming Interface 459
7.6.7 Modbus Diagnosis Programming Interface 460
7.6.8 Modbus TCP Diagnosis Programming Interface 462
7.6.9 EtherCAT Diagnosis Programming Interface 464
7.6.10 CPU Stop Control 465
7.6.11 Axis Diagnosis 465
8 FAQ 468
8.1 CPU Utilization Too High 468
8.1.1 CPU Usage Definition 468
8.1.2 Analysis Procedure 468

8.1.3 Common Optimization Methods 469
8.2 Abnormal PLC Running 469
8.2.1 Overview 469
8.2.2 Symptoms 470
8.2.3 Cause Analysis and Solutions 470
8.3 Failure to Obtain Folders 475
9 Appendix 477
9.1 Communication Protocols for Communication Ports 477
9.1.1 Overview 477
9.1.2 Mini-USB Port and Built-in Communication Protocol 477
9.1.3 COM Communication Ports and Built-in Protocols 477
9.1.4 CANopen Communication Protocol 478
9.1.5 CANlink Communication Protocol 478
9.1.6 Ethernet Ports and Communication Protocols 479
9.1.7 EtherCAT Port and Communication Protocol 479
9.1.8 High-Speed I/O Interface 479
9.1.9 Mini-SD Card Slot 479
9.1.10 Local Bus Expansion Interface 479
9.1.11 PROFIBUS-DP Port 480
9.2 Soft Elements 480
9.3 Cheat Sheet of Basic Instructions 481
9.4 PLC Programming Software Upgrade 485
9.4.1 Version 485
9.4.2 Upgrade Method 486
9.4.3 FAQs 489
9.5 PLC User Program Upgrade 496
9.5.1 Upgrade Using the InoProShop 496
9.5.2 Upgrade Using an SD Card 496
9.6 AM400 or AM600 High-Speed I/O Wiring 497
9.7 High-Speed I/O Compatibility 502
9.7.1 Introduction to Earlier and Latest UIs 502
9.7.2 High-Speed I/O Diagnosis 504
9.8 Diagnosis Code and Diagnosis Information 509
9.8.1 Overview 509
9.8.2 CPU Diagnosis Code 509
9.8.3 I/O Module Diagnosis Code 512

9.8.4 PROFIBUS-DP Diagnosis Code 513
9.8.5 CANlink Diagnosis Code 513
9.8.6 Modbus Diagnosis Code 514
9.8.7 EtherCAT Diagnosis Code 515
9.8.8 Axis Diagnosis Code 518
9.9 Synchronizing the Project Information 522
9.9.1 Overview 522
9.9.2 Synchronizing the Downloaded Project Information Automatically 523
9.9.3 Synchronizing the Downloaded Project Information Manually 524
9.9.4 Special Notes on Synchronizing the Project Information 524
9.10 SVN Function 524
9.10.1 Overview 524
9.10.2 Creation of an SVN Server and SVN Library 525
9.10.3 Installation of the CodeMeter Runtime Environment and SVN Plug-in 525
9.10.4 Acquisition of the Authorization Code and Offline Authorization 527
9.10.5 SVN Operator Guidance 532
9.10.6 Uninstallation of the SVN Plug-in 535

1 Product Information

1.1 Overview

1.1.1 Product Information

Inovance medium-sized programmable logic controllers ("medium-sized PLCs"), including the AM, AC, and AP series, provide intelligent automation solutions for users. These PLCs use the IEC61131-3 programming language and support programming languages of the PLCopen standard.

- The AM400 and AM600 series are installed on racks, with each rack supporting 16 local expansion modules and also remote expansion modules through multiple industrial field buses such as PROFIBUS-DP, EtherCAT, and CANopen. The AM600 local expansion modules, that is, I/O modules, are connected through the internal bus protocol, including digital input (DI), digital output (DO), analog input (AI), analog output (AO), and temperature modules. The AM600 provides the following functions: high-performance motion control function through the EtherCAT bus; single-axis acceleration/deceleration control, electronic gear, electronic cam, and basic single-axis positioning with the maximum frequency of 200 kHz through the high-speed I/O; and communication functions through the RS485, Ethernet, and USB ports.
- The AM300 and AM500 series are Inovance standard medium-sized PLCs of the new generation. They support multiple communication protocols, including EtherCAT (supported by AM500 only), EtherNet/IP, OPC UA, Modbus TCP (supporting two IP addresses in the same network segment), TCP/IP, RS485 (extensible to 3 ports at most), and CAN bus.
- The AC series adopt the booksize structure and support multiple communication protocols such as EtherCAT, EtherNet/IP, OPC UA, Modbus TCP/RTU, UDP, and Socket, meeting various application requirements of users. Among the AC series, the AC700 supports motion control of up to 32 axes in 1 ms, while the AC800 supports motion control of up to 256 axes in 4 ms. The AP700 series intelligent mechanical controller, developed on the Intel Atom processor hardware platform, is a high-performance multi-axis motion controller in compliance with the PLCopen specifications. With the high-speed EtherCAT bus, it can implement multi-axis servo control. Its 15-inch industrial-class TFT touchscreen facilitates input of control instructions and operation control programs and display of internal data. These features make the AP700 suitable for control of high-speed production equipment and large equipment in advanced manufacturing, especially intelligent equipment like machine tools.
- The AM780-N is a medium-sized PLC with wide-temperature characteristics. It has a small size and compact structure. It is convenient to disassemble, wire, and extend, highly resistant to vibration and interference, quick in I/O control and refreshing, and highly reliable. It is widely used in wind power, metallurgy, outdoor power station, and mine industries.

The medium-sized PLCs have the following features:

- Multiple motion control functions: bus motion control and pulse motion control
- Abundant I/O channels, up to ten thousand channels
- Large program capacity and data storage
- Fast instruction execution speed
- Various high-end field buses including EtherCAT, CANopen, EtherNet/IP, and PROFIBUS-DP
- Easy-to-use software, meeting various application requirements of users

- Online debugging mode
- Online editing mode

The following table lists the CPU module types and their function differences of medium-sized PLCs.

Table 1–1 Functional characteristics of different CPU module types

Product Model[Note 1]	Number of Local Expansion Modules	Program Storage Space	Data Storage Space	Data Stored at Power Failure	Motion Control Axes	High-Speed I/O	Soft Element	Output Type
AM401-CPU1608TP	8	10 MB	20 MB	480 KB	4 servo axes, 4 positioning axes	16 inputs and 8 outputs	✓	Source
AM402-CPU1608TP	8	10 MB	20 MB	480 KB	8 servo axes, 4 positioning axes	16 inputs and 8 outputs	✓	Source
AM401-CPU1608TN	8	10 MB	20 MB	480 KB	4 servo axes, 4 positioning axes	16 inputs and 8 outputs	✓	SINK output
AM402-CPU1608TN	8	10 MB	20 MB	480 KB	8 servo axes, 4 positioning axes	16 inputs and 8 outputs	✓	SINK output
AM403-CPU1608TN	16	10 MB	20 MB	480 KB	16 servo axes, 4 positioning axes	16 inputs and 8 outputs	✓	SINK output
AM600-CPU1608TP	16	10 MB	20 MB	480 KB	Max. 32 (recommended: no more than 20)	16 inputs and 8 outputs	✓	Source
AM600-CPU1608TN	16	10 MB	20 MB	480 KB	Max. 32 (recommended: no more than 20)	16 inputs and 8 outputs	✓	SINK output
AM610-CPU1608TP	16	10 MB	20 MB	480 KB	x	16 inputs and 8 outputs	✓	Source
AC801-0221-U0R0	x	128 MB	128 MB	5 MB (external UPS required)	48	x	x	x
AC802-0222-U0R0	x	128 MB	128 MB	5 MB (external UPS required)	128	x	x	x
AC810-0122-U0R0	x	128 MB	128 MB	5 MB (external UPS required)	256	x	x	x
AC812-0322-U0R0	x	128 MB	128 MB	5 MB (external UPS required)	256	x	x	x
AP702-0221-U0R0	x	128 MB	128 MB	5 MB (external UPS required)	48	x	x	x
AP703-0221-U0R0	x	128 MB	128 MB	5 MB (external UPS required)	48	x	x	x

Product Model[Note 1]	Number of Local Expansion Modules	Program Storage Space	Data Storage Space	Data Stored at Power Failure	Motion Control Axes	High-Speed I/O	Soft Element	Output Type
AP705-LM[Note 2]	x	128 MB	128 MB	5 MB (external UPS required)	48	x	x	x
AC703	x	128 MB	128 MB	5 MB, retentive at power failure, no external UPS required	32	8 inputs and 4 outputs	x	SINK
AC702	x	128 MB	128 MB	5 MB, retentive at power failure, no external UPS required	16	8 inputs and 4 outputs	x	SINK
AM320-0808TN	16	10 MB	20 MB	512 KB	x	8 inputs and 8 outputs	x	SINK
AM521-0808TN	16	10 MB	20 MB	512 KB	8	8 inputs and 8 outputs	x	SINK
AM522-0808TN	16	10 MB	20 MB	512 KB	16	8 inputs and 8 outputs	x	SINK
AM523-0808TN	16	10 MB	20 MB	512 KB	32	8 inputs and 8 outputs	x	SINK
AM780-N	32	128 MB	128 MB	5 MB, retentive at power failure, no external UPS required	32	x	✓	x

Table 1–2 Communication characteristics of the CPU module

Product Model[Note 1]	Communication					
	EtherCAT	PROFIBUS-DP	CANopen/ CANlink	Modbus TCP	Modbus (Serial Port)	Ethernet/IP
AM401-CPU1608TP	1 (each supports up to 128 slaves)	x	1 (each supports up to 63 slaves)	1 (each supports up to 63 slaves)	1 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AM402-CPU1608TP	1 (each supports up to 128 slaves)	x	1 (each supports up to 63 slaves)	1 (each supports up to 63 slaves)	1 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AM401-CPU1608TN	1 (each supports up to 128 slaves)	x	1 (each supports up to 63 slaves)	1 (each supports up to 63 slaves)	1 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)

Product Model[Note 1]	Communication					
	EtherCAT	PROFIBUS-DP	CANopen/ CANlink	Modbus TCP	Modbus (Serial Port)	Ethernet/IP
AM402-CPU1608TN	1 (each supports up to 128 slaves)	x	1 (each supports up to 63 slaves)	1 (each supports up to 63 slaves)	1 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AM403-CPU1608TN	1 (each supports up to 128 slaves)	x	1 (each supports up to 63 slaves)	1 (each supports up to 63 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AM600-CPU1608TP	1 (each supports up to 128 slaves)	x	1 (each supports up to 63 slaves)	1 (each supports up to 63 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AM600-CPU1608TN	1 (each supports up to 128 slaves)	x	1 (each supports up to 63 slaves)	1 (each supports up to 63 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AM610-CPU1608TP	x	1 (each supports up to 124 slaves)	x	1 (each supports up to 63 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AC801-0221-U0R0	1 (each supports up to 128 slaves)	x	x	2 (each supports up to 128 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AC802-0222-U0R0	2 (each supports up to 128 slaves)	x	x	2 (each supports up to 128 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AC810-0122-U0R0	2 (each supports up to 128 slaves)	x	x	2 (each supports up to 128 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AC812-0322-U0R0	2 (each supports up to 128 slaves)	x	x	2 (each supports up to 128 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AP702-0221-U0R0	1 (each supports up to 128 slaves)	x	x	2 (each supports up to 128 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AP703-0221-U0R0	1 (each supports up to 128 slaves)	x	x	2 (each supports up to 128 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AP705-LM[Note 2]	1 (each supports up to 128 slaves)	x	x	2 (each supports up to 128 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AC703	1 (each supports up to 128 slaves)	x	x	2 (each supports up to 63 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)

Product Model[Note 1]	Communication					
	EtherCAT	PROFIBUS-DP	CANopen/ CANlink	Modbus TCP	Modbus (Serial Port)	Ethernet/IP
AC702	1 (each supports up to 128 slaves)	x	x	2 (each supports up to 63 slaves)	2 (each supports up to 31 slaves)	1 (each supports up to 64 slaves)
AM320-0808TN	x	x	1 (each supports up to 30 slaves)	2 (when serving as the master, each supports up to 63 slaves; when serving as the slave, each supports up to 16 masters)	Up to 3 channels (1 onboard and 2 expansion cards) (each supports up to 31 slaves)	2 (when serving as the master, each supports up to 16 slaves; when serving as the slave, each supports up to 16 masters)
AM521-0808TN	1 (each supports up to 127 slaves)	x	1 (each supports up to 30 slaves)	2 (when serving as the master, each supports up to 63 slaves; when serving as the slave, each supports up to 16 masters)	Up to 3 channels (1 onboard and 2 expansion cards) (each supports up to 31 slaves)	2 (when serving as the master, each supports up to 16 slaves; when serving as the slave, each supports up to 16 masters)
AM522-0808TN	1 (each supports up to 127 slaves)	x	1 (each supports up to 30 slaves)	2 (when serving as the master, each supports up to 63 slaves; when serving as the slave, each supports up to 16 masters)	Up to 3 channels (1 onboard and 2 expansion cards) (each supports up to 31 slaves)	2 (when serving as the master, each supports up to 16 slaves; when serving as the slave, each supports up to 16 masters)
AM523-0808TN	1 (each supports up to 127 slaves)	x	1 (each supports up to 30 slaves)	2 (when serving as the master, each supports up to 63 slaves; when serving as the slave, each supports up to 16 masters)	Up to 3 channels (1 onboard and 2 expansion cards) (each supports up to 31 slaves)	2 (when serving as the master, each supports up to 16 slaves; when serving as the slave, each supports up to 16 masters)
AM780-N	1 (each supports up to 128 slaves)	x	1 (each supports up to 8 slaves)	2 (when serving as the master, each supports up to 63 slaves; when serving as the slave, each supports up to 16 masters)	Up to 9 channels (1 onboard and 8 expansion cards (4 GL20-2S485-N modules)) (each supports up to 31 slaves)	2 (when serving as the master, each supports up to 32 slaves; when serving as the slave, each supports up to 64 masters)

Note

- [Note 1]: 1) The number of the power module and stopper plate are not included. 2) The AM610-CPU1608TP is no longer maintained.
- [Note 2]: The AP705-LM is used for generic machine tools.

1.1.2 Product Configuration and Module Description

Inovance provides a rich range of medium-sized PLCs for users to select the required product configuration according to the applications. Take the AM600 series PLC as an example. The AM600 includes two structures: AM600-CPU1608TP (EtherCAT+CANopen) and AM610-CPU1608TP (PROFIBUS-DP). The following are the typical system integration diagrams of the two structures.

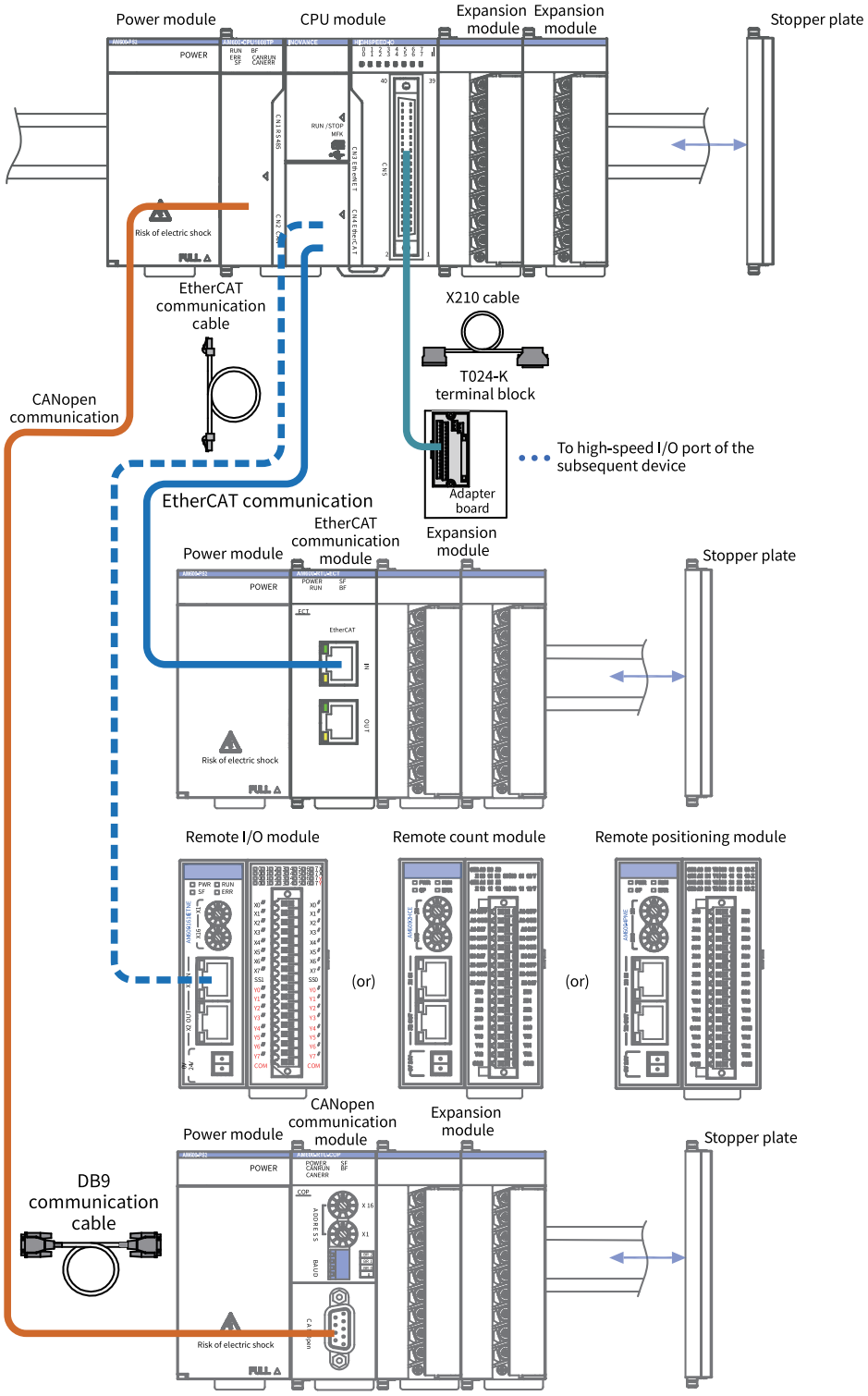


Figure 1-1 AM600-CPU1608TP typical system integration diagram

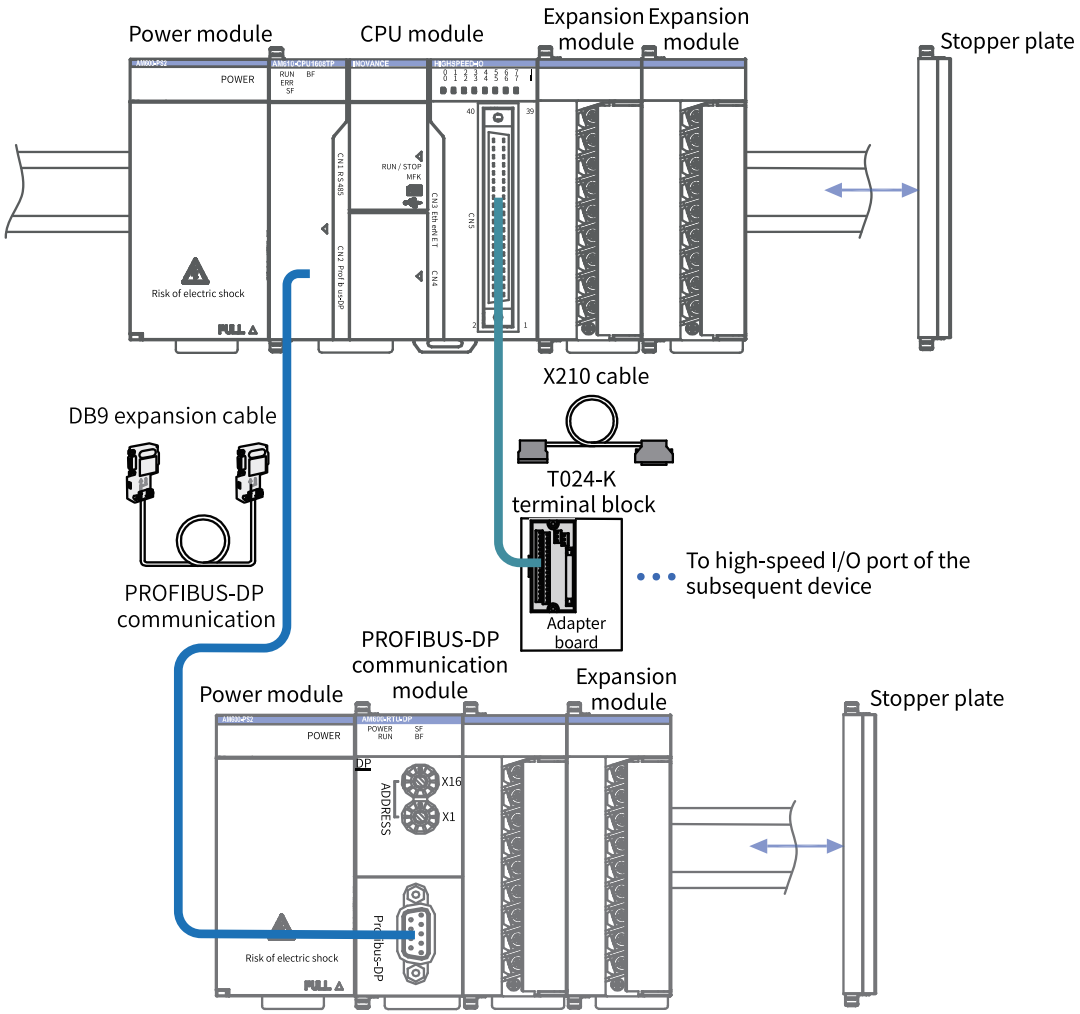


Figure 1-2 AM610-CPU1608TP typical system integration diagram

Note

The AM610-CPU1608TP and AM600-RTU-DP modules are no longer maintained.

The AM600 modules are classified into the power module, CPU module, remote communication modules, and local expansion modules based on functions, as described in the following table.

Ordering Code	Model [Note]	Category	Description
01440010	AM600-PS2	Power module	220 V voltage input, 24 V/2 A output
01440028	AM401-CPU1608TP	CPU module	1 x RS485, 1 x CANopen/CANlink, 1 x LAN 4-axis motion control, EtherCAT Built-in 16 high-speed inputs and 8 high-speed outputs Source output
01440029	AM402-CPU1608TP	CPU module	1 x RS485, 1 x CANopen/CANlink, 1 x LAN 8-axis motion control, EtherCAT Built-in 16 high-speed inputs and 8 high-speed outputs Source output

Ordering Code	Model [Note]	Category	Description
01440032	AM401-CPU1608TN	CPU module	1 x RS485, 1 x CANopen/CANlink, 1 x LAN 4-axis motion control, EtherCAT Built-in 16 high-speed inputs and 8 high-speed outputs SINK output
01440031	AM402-CPU1608TN	CPU module	1 x RS485, 1 x CANopen/CANlink, 1 x LAN 8-axis motion control, EtherCAT Built-in 16 high-speed inputs and 8 high-speed outputs SINK output
01440126	AM403-CPU1608TN		2 x RS485, 1 x CANopen/CANlink, 1 x LAN 16-axis motion control, EtherCAT Built-in 16 high-speed inputs and 8 high-speed outputs SINK output
01440014	AM600-CPU1608TP		2 x RS485, 1 x CANopen/CANlink, 1 x LAN Basic motion control, EtherCAT Built-in 16 high-speed inputs and 8 high-speed outputs Source output
01440016	AM610-CPU1608TP		2 x RS485, 1 x LAN Basic motion control, PROFIBUS-DP Built-in 16 high-speed inputs and 8 high-speed outputs Source output
01440143	AC812-0322-U0R0	Booksize controller	2 x USB2.0, 2 x USB3.0 1 x RS485/RS232, 4 x LAN Motion control of up to 256 axes, EtherCAT Multifunctional expansion slot, built-in Mini-PCIE expansion slot
01440038	AC810-0122-U0R0		
01440101	AC802-0222-U0R0		2 x USB2.0, 2 x USB3.0 1 x RS485/RS232, 4 x LAN Motion control of up to 128 axes, EtherCAT Multifunctional expansion slot, built-in Mini-PCIE expansion slot
01440103	AC801-0221-U0R0		2 x USB2.0, 2 x USB3.0 1 x RS485/RS232, Ethernet Motion control of up to 48 axes, EtherCAT Built-in Mini-PCIE expansion slot
01440066	GL10-1600END	DI module	16-point DI module, 24 VDC input (source/SINK)
01440081	GL10-0016ER	DO module	16-point DO module, relay output
01440069	GL10-0016ETP		16-point DO module, transistor output (source)
01440067	GL10-0016ETN		16-point DO module, transistor output (SINK)
01440080	GL10-4AD	AI module	4-channel AD module, voltage/current analog input

Ordering Code	Model [Note]	Category	Description
01440071	GL10-4DA	AO module	4-channel DA module, voltage/current analog output
01440082	GL10-0032ETN	DO module	32-point DO module, transistor output (SINK)
01440066	GL10-3200END	DI module	32-point DI module, 24 VDC input (source/SINK)
01440121	GL10-2PH	Differential output pulse module	2-channel high-speed differential output pulse positioning module, output frequency 1 MHz, 8-channel general inputs
01440129	GL10-4PM	Local pulse positioning module	4-channel pulse positioning output
01440075	GL10-4PT	Temperature module	4-channel thermal resistor temperature collection, supporting a variety of thermal resistors
01440078	GL10-4TC	Temperature module	4-channel thermocouple temperature collection, supporting a variety of thermocouples
01440070	GL10-8TC	Temperature module	8-channel thermocouple temperature collection, supporting a variety of thermocouples
01440074	GL10-PS2	Power module	220 V voltage input, 24 V/2 A output
01440077	GR10-0808ETNE	EtherCAT slave I/O module	8-point DOs, transistor output (SINK); 8 DIs (source)
01440255	GR10-1616ETNE	EtherCAT slave I/O module	16-point DOs, transistor output (SINK); 16 DIs (source)
01440252	GR10-4PME	EtherCAT slave positioning module	4-channel positioning module of EtherCAT slave
01440279	GR10-2HCE	EtherCAT slave counter module	2-channel counter module of EtherCAT slave
01440058	GR10-4ADE	AI module	4-channel AI module of EtherCAT slave
01440060	GR10-4DAE	AO module	4-channel AO module of EtherCAT slave
01440123	GR10-4TCS-PID (out of production)	Temperature control module	4-channel thermocouple (TC) temperature detection module
01440059	GR10-8TCE	Temperature detection module	8-channel thermocouple temperature collection, supporting a variety of thermocouples
01440127	GR10-8PBE	EtherCAT slave probe module	8-channel EtherCAT slave probe module of the GR10 series
01440256	GR10-2PHE	Differential output pulse module	2-channel high-speed differential output pulses, supporting 8 inputs
01440135	GR10-EC-6SW	6-channel EtherCAT branch module	1 x EtherCAT input, 5 x EtherCAT outputs
01440177	GR10-1616ERE-BD	Expansion board	GR10 series programmable controller 16-channel input and 16-channel output EtherCAT slave board
01440012	AM600-RTU-DP	DP communication module	PROFIBUS-DP protocol communication interface module
01440073	GL10-RTU-ECTA	EtherCAT communication module	EtherCAT protocol communication interface module
01440013	AM600-RTU-ECTA	EtherCAT communication module	EtherCAT protocol communication interface module
01440083	GL10-RTU-COP	CANopen communication module	CANopen protocol communication interface module

Note

The expansion module of the AM600 series has been upgraded to the expansion module of the GL10/GR10 series, and the modules are compatible with each other. The AM610-CPU1608TP and AM600-RTU-DP modules are no longer maintained.

1.1.3 System Application Process

The system application process of medium-sized PLCs is shown in the following figure. For module installation and wiring, see the AM600 Series PLC Hardware Guide and AC810 Series PLC Hardware Guide.

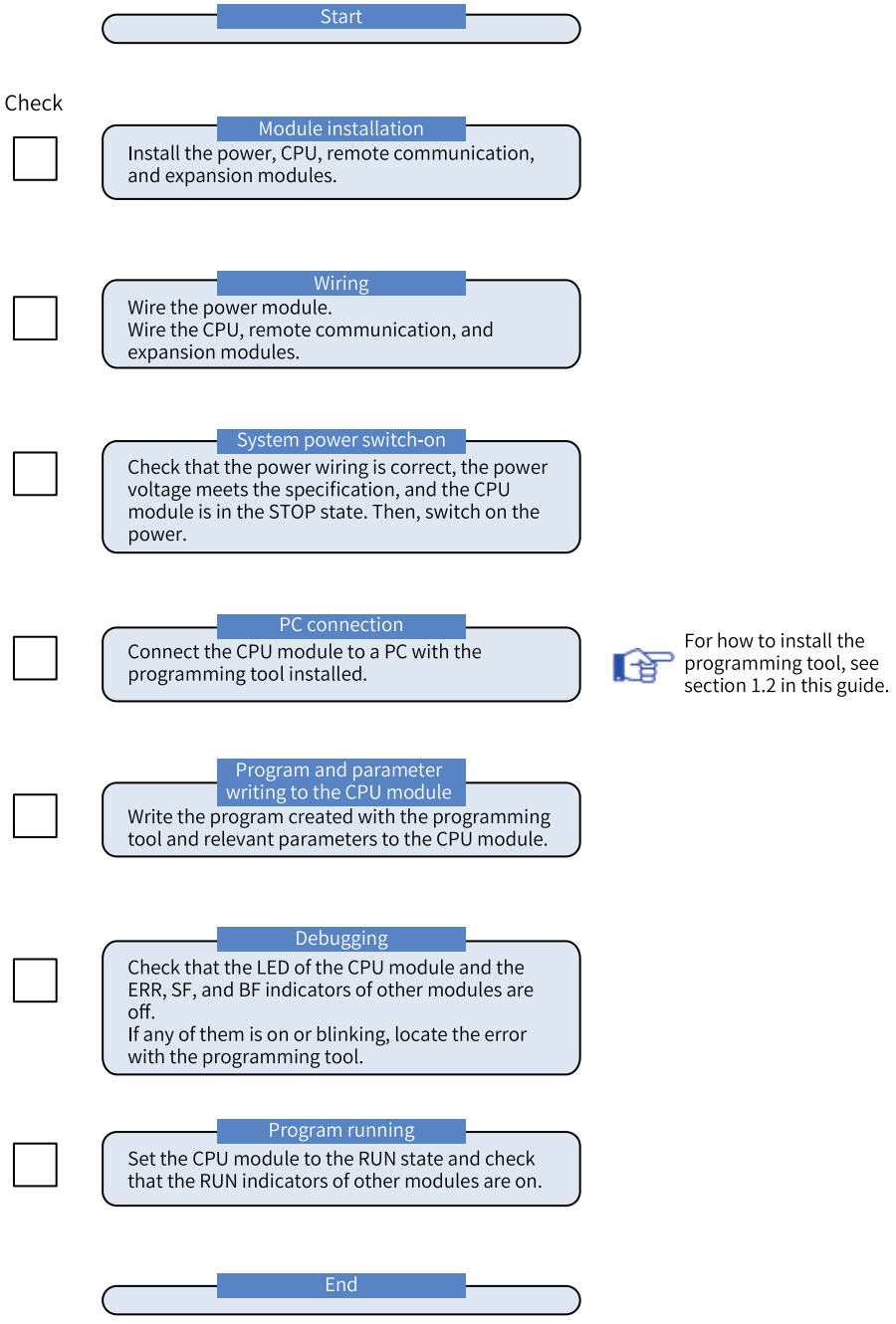


Figure 1-3 Basic application flowchart of medium-sized PLCs

1.2 Overview of InoProShop

1.2.1 Brief Introduction

InoProShop is a programming configuration software for medium-sized PLCs. Developed based on CODESYS V3 (shorted as "CODESYS"), InoProShop provides a complete configuration, programming, debugging, and monitoring environment for medium-sized PLCs with flexible processing on IEC languages.

InoProShop is used to manage projects and devices, and supports the following configurations on medium-sized PLCs:

- CPU configuration
- I/O module configuration
- EtherCAT bus
- PROFIBUS-DP bus
- CANopen/CANlink bus
- Modbus/Modbus TCP bus
- EtherNet/IP bus
- High-speed I/O

Besides programming, downloading, and debugging, this software also provides the following functions:

- Standard programming (IEC 61131-3 compliant)
  The software supports multiple programming languages, including structured text (ST), ladder diagram (LD), sequential function chart (SFC), and continuous function chart (CFC) of IEC61131-3 extended programming language.
- Flexible function block (FB) library
  The software supports full FB library and user-defined library.
- Offline simulation
  Users can complete program debugging simulation without connecting the PLC hardware.
- Intelligent error locating
  The software quickly locates errors during pre-programming and programming and provides diagnostics and logs.
- Sampling tracking
  The software can establish the timing diagram of process variables.

1.2.2 Connection Between InoProShop and Hardware

Connect the programming device to the PLC through Ethernet (such as a hub or switch) or USB, write a user program in the InoProShop, and download the program to the PLC to monitor the program and control the PLC.

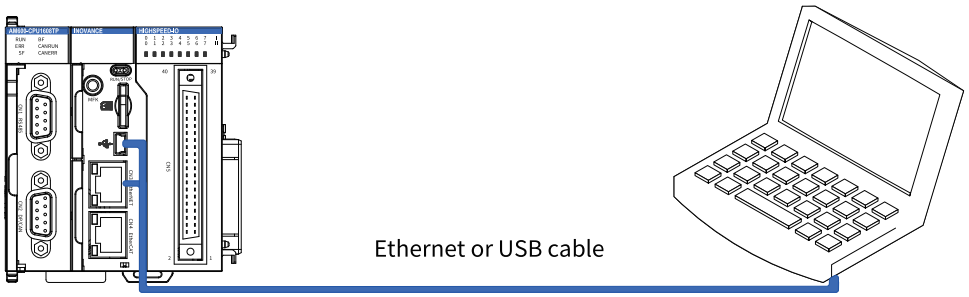


Figure 1-4 Connection between InoProShop and hardware

1.2.3 Acquisition and Installation of the Software

Software Acquisition

Inovance InoProShop is available for free. You can obtain the installation file and related reference using the following methods:

- Obtain a CD copy of the installation files from any Inovance distributor.
- Download the software installation package for free on the "Service" and "Support > Downloads" page at www.inovance.com.

As Inovance constantly improves its products and documentation, it is recommended that you update your software versions and consult the latest reference materials when needed to facilitate your application design.

Installation Environment Requirements

Prepare a desktop or portable PC meeting the following requirements:

- OS: Windows 7 or 10, 64-bit recommended
- Memory: 4 GB or above
- Space: an idle hard drive space of 5 GB or above

Connect the PC and the medium-sized PLC in the following way:

Connection Mode	Cables Required	Description
LAN network cable (recommended)	An idle LAN network port in the local network and a network cable	Long distance connection between the PC and the medium-sized PLC is supported. For example, you can program a medium-sized PLC which is operating in the workshop in your office, and the communication rate is faster.
USB cable	One USB cable, of which the end connecting the medium-sized PLC must be a Mini-USB connector.	Currently, the AM400 and AM600 series support this connection mode.

1.2.4 Installation Procedure

Preparation Before Installation

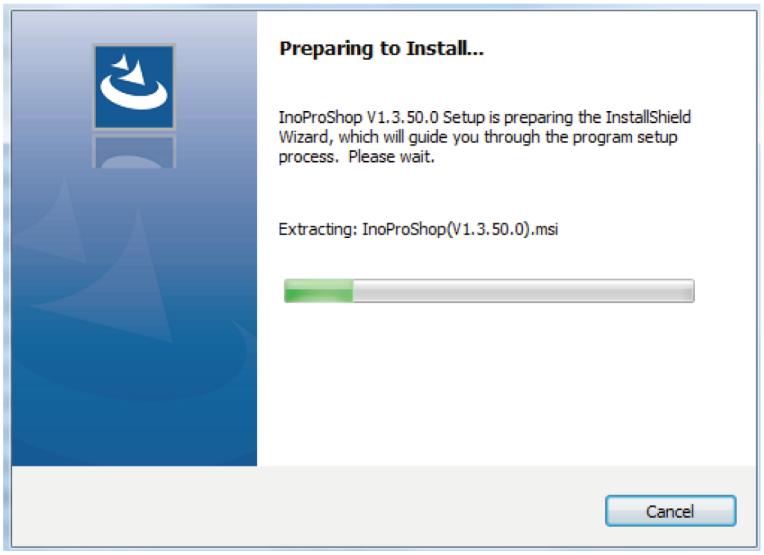
If you install InoProShop for the first time, ensure that there is at least 5 GB free space on the target hard disk. In this case, you can directly install the software.

If you are upgrading InoProShop, back up your files, uninstall the old version of InoProShop, restart the computer, and then install the new version.

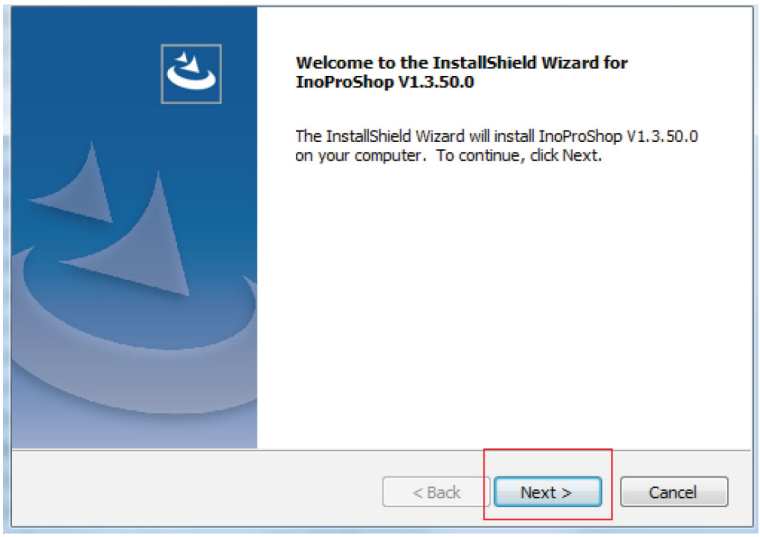
Installing InoProShop

Go to the Windows System Resource Manager, open the directory where the installation files are located, and double-click the InoProShop (V*.*.*.*).exe file (V*.*.*.* is the version of InoProShop. Make sure you have the latest version).

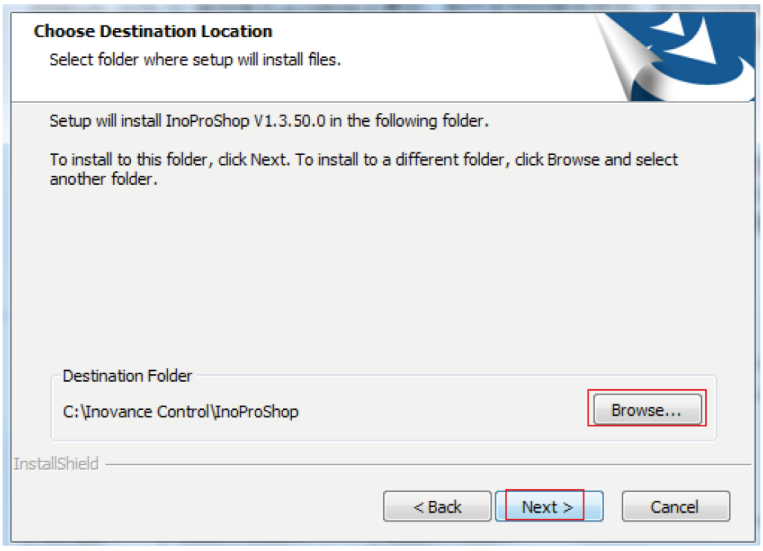
The installation wizard appears and the system prepares for the installation.



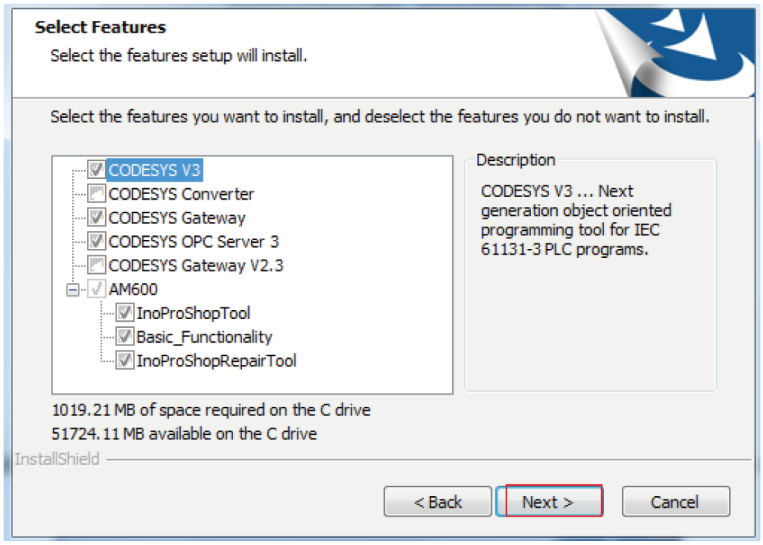
Click "Next" to start the installation.



Select the installation path and click "Next".

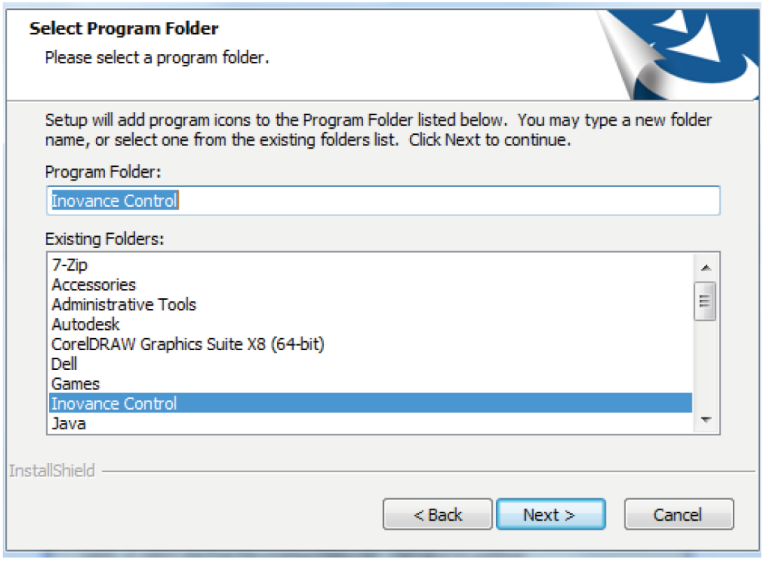


Select components that you want to install and click "Next".

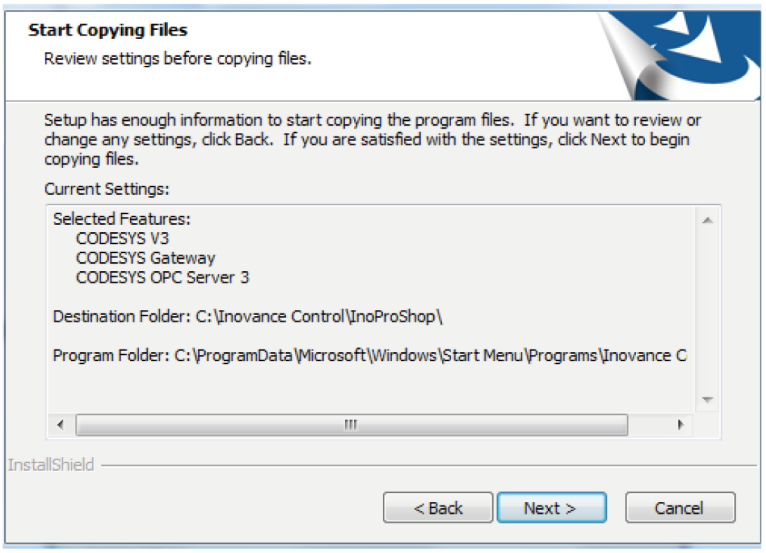


.

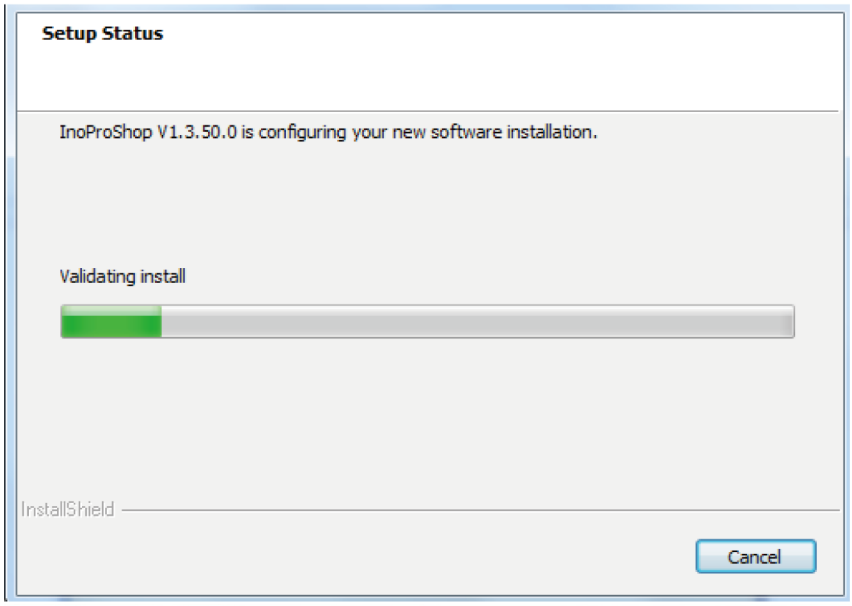
Click "Next".

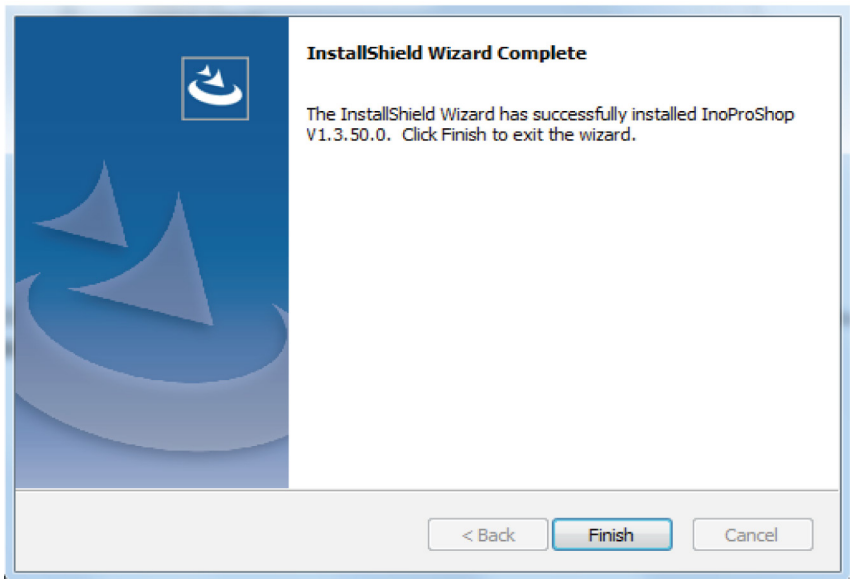


Click "Next".



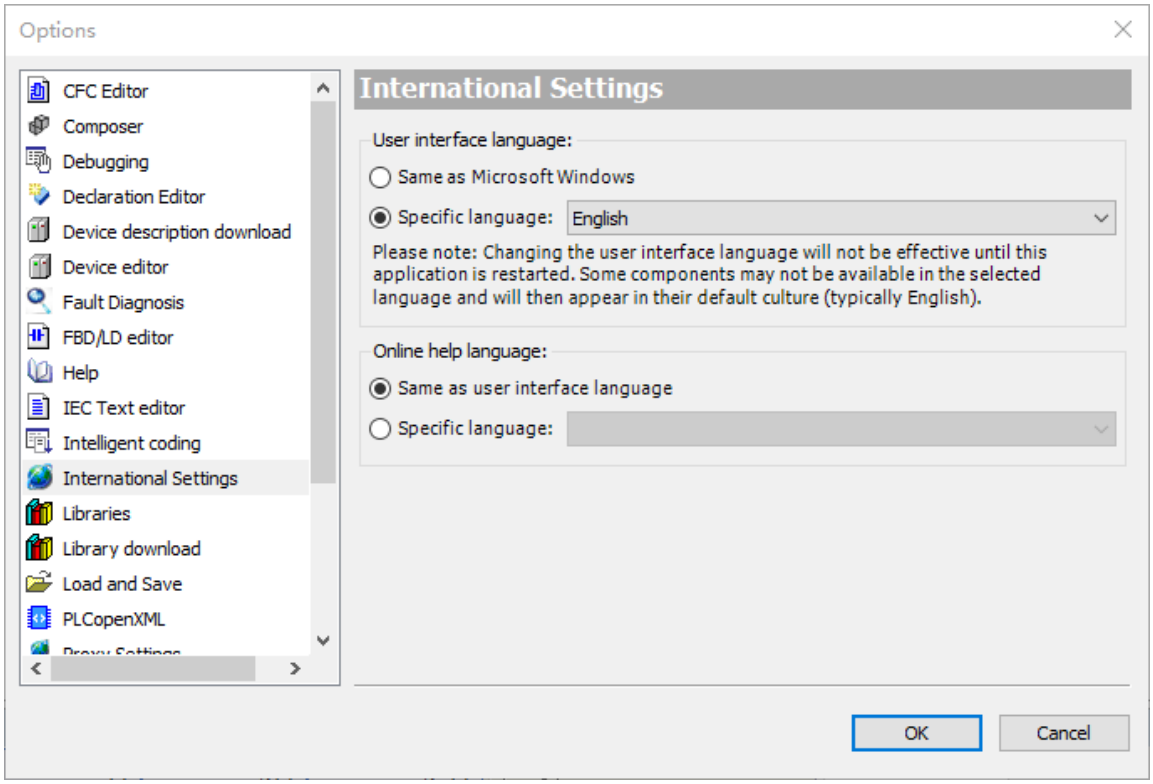
Wait until the "InstallShield Wizard Complete" page appears, and click "Finish" to complete the installation.





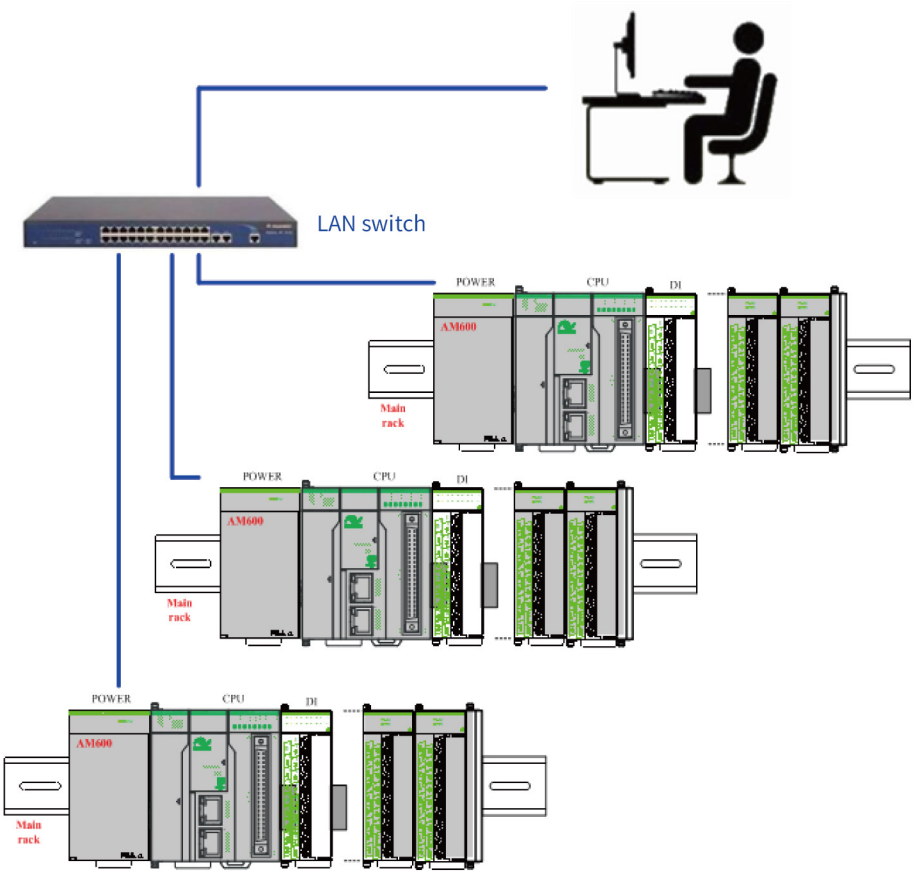
Language Setting

The default page language of InoProShop is Simplified Chinese. If you need to change the language, choose "Options" > "International Settings" on the main page of the software to select the desired language.



Confirming Whether Selected Controller Is Correct

If multiple AM600s are connected to one LAN, after you log in to a controller, you need to confirm whether the selected controller is your desired one.



For this purpose, on the "Device" tab page of InoProShop, click "System Setting", and then click "Identify Device".

① Double-click "Device".

② The "System Setting" page is displayed.

③ Click "Identify Device". PLCs selected on the "Communication Settings" page are displayed on the LEDs alternately.

1.2.5 Uninstallation of InoProShop

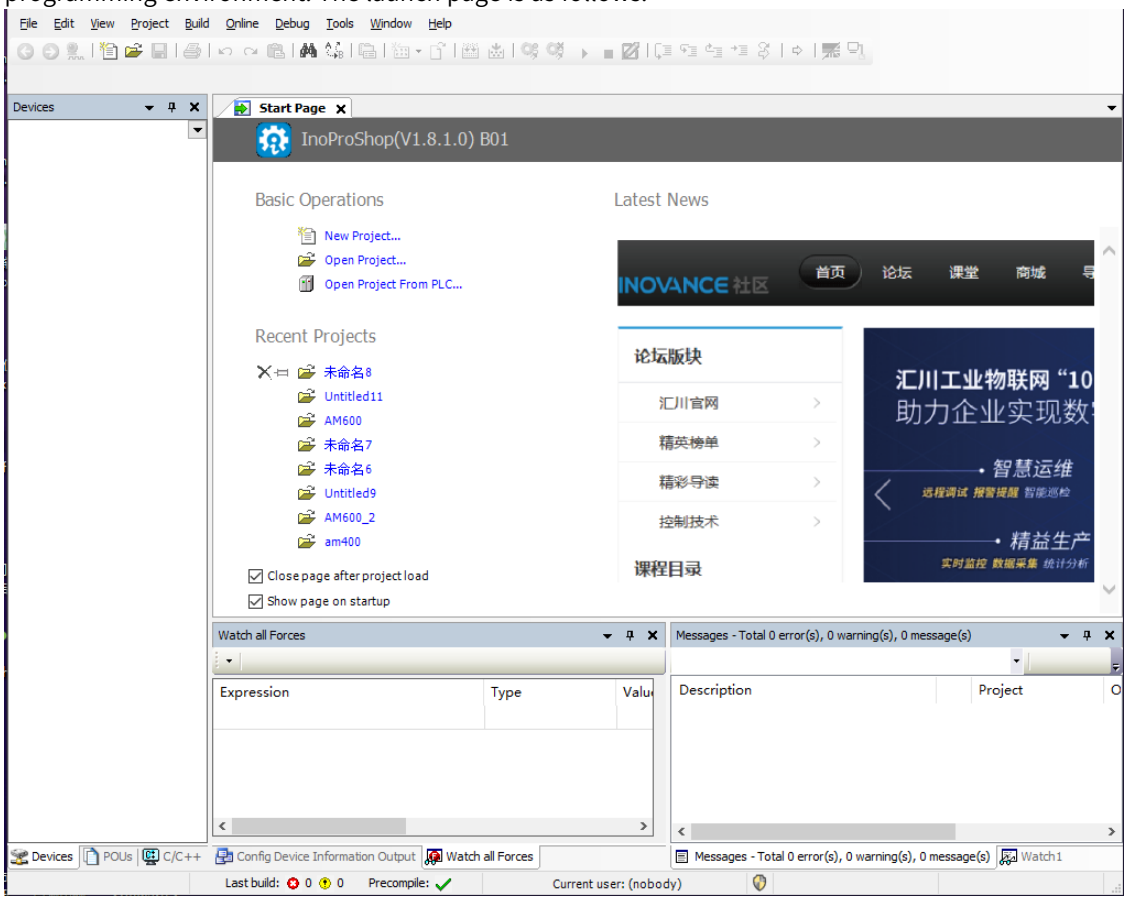
In the Windows system, you can uninstall InoProShop directly from the Control Panel as follows:

1. Quit InoProShop and ensure that Gateway is closed.
2. If the CODESYS icon exists on the task bar, right-click the icon and choose "Exit" to close Gateway.
3. Choose "Start" > "Settings" > "Control Panel".
4. Double-click "Add or Remove Programs".
5. Select InoProShop in the list.
6. Right-click it, and then click "Remove". In the confirmation dialog box, click "Confirm".

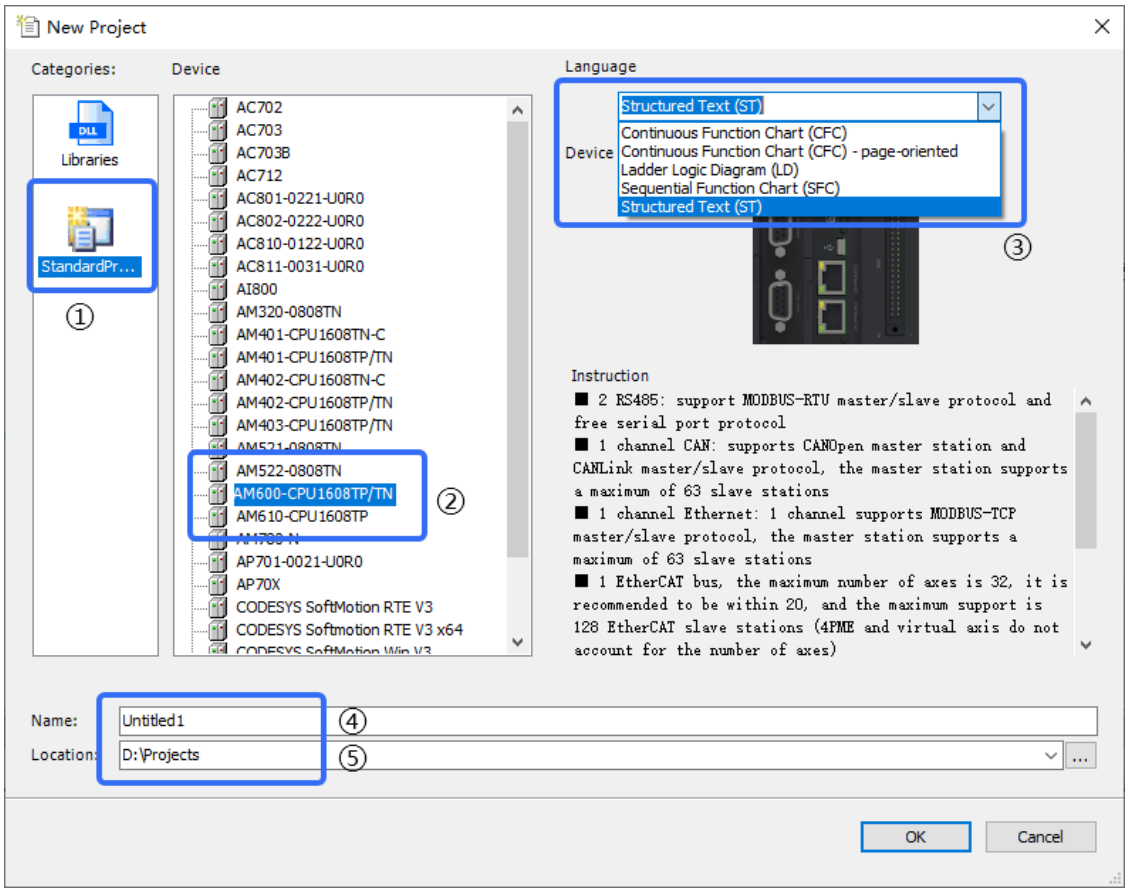
2 Quick Start

2.1 Programming Environment Launching

1. Double-click the programming software icon on the desktop to launch the InoProShop programming environment. The launch page is as follows:

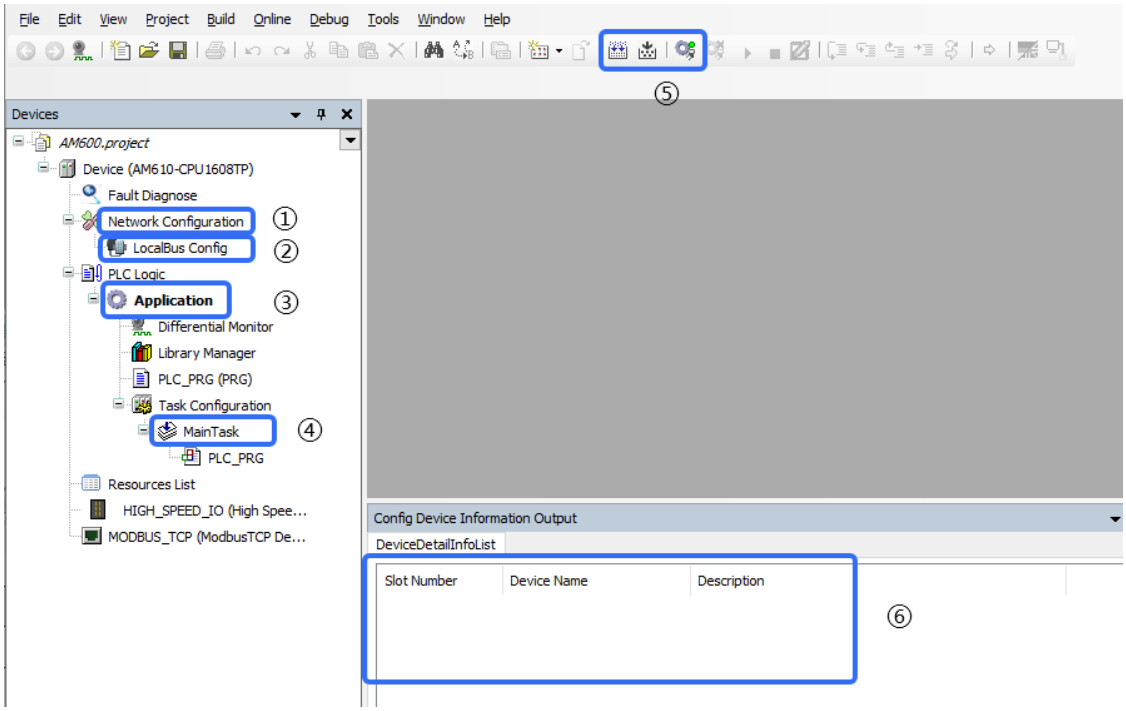


2. Click on the top left corner of the menu bar or choose "File" > "New Project" to create a project. Select "Standard project" and the programming language, and specify the project file name as well as storage path, as shown in the following figure:



No.	Description
①	Select "Standard project".
②	Select the main module.
③	Select the programming language.
④	Enter a name for the project.
⑤	Select the storage path.

3. Click "OK". The system configuration and programming page is displayed. The following figure shows common buttons and window distribution of this page.



No.	Description
①	Network configuration
②	Local bus configuration
③	User program management unit
④	Task execution method and period configuration
⑤	Compiling, login, and debugging
⑥	Device information window

2.2 Typical Procedure for Writing a User Program

2.2.1 Overview

If this is the first time you use Inovance medium-sized PLCs, note that five steps are required to write and debug a complete user program.

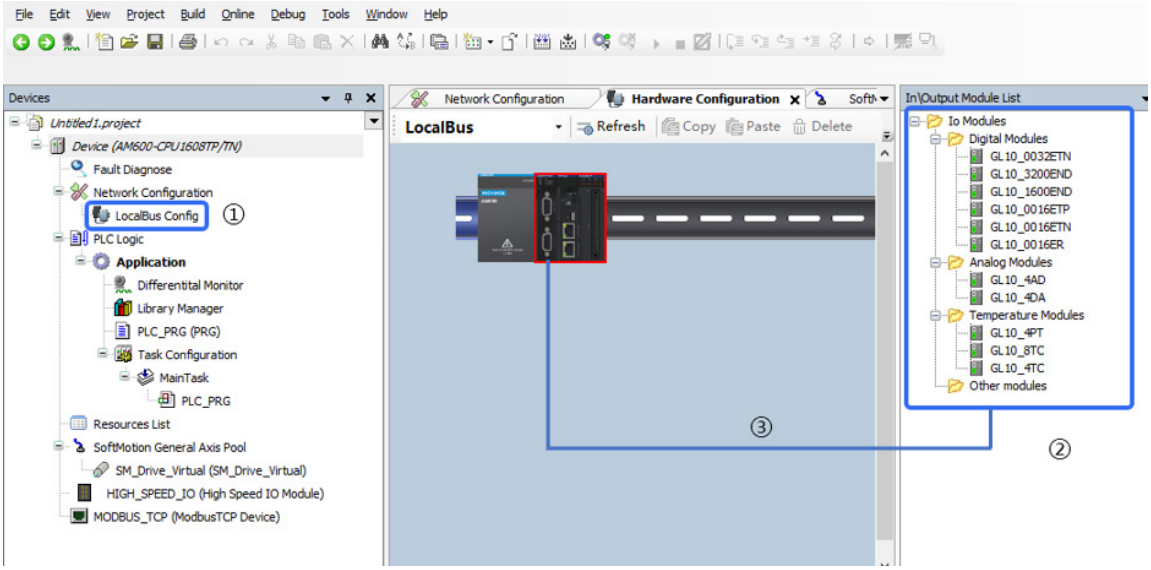
1. Configure the hardware system based on the hardware connection structure of the medium-sized PLC application system.

   - If only the CPU main module and I/O expansion module are used, you only need to configure the hardware as follows: Place the "elements" of selected module type and model and installation order into the "rack" on the InoProShop hardware configuration page.
   - If the expansion racks are used, configure the bus first. Then, add a certain number of network expansion modules according to the number of expansion racks, and add expansion modules into each rack.

2. Write the user program according to the control procedure of the application system. During programming, the variables are customized based on the data storage width and use scope, which may be independent of hardware configuration.
3. Link the input port variable (I), output status (Q), or value (M) of each hardware port in the system structure with the variables in the user program.
4. Configure the synchronization period of network communication (for example, the EtherCAT bus). Configure the execution periods of user program units according to the instantaneity requirements of tasks.
5. Log in to the medium-sized PLC in the InoProShop programming environment. Download the user program, perform simulated debugging, and rectify faults until the program runs normally.

2.2.2 User System Configuration Operations

On the InoProShop main page, double-click "LocalBus Config" in the left device tree. The "Hardware Configuration" page of the PLC main rack is displayed:



① Double-click here to open the local expansion module configuration page.

② Element library of the expansion module.

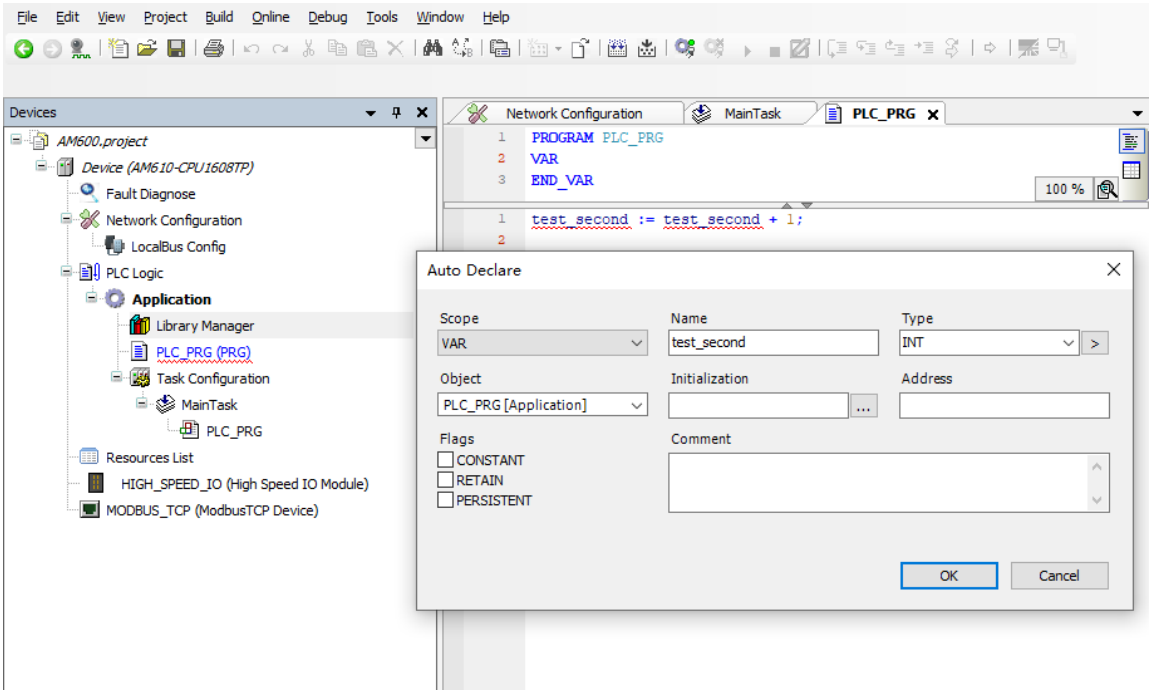
③ Click on the right side of the CPU unit in the installation slot. Double-click the required I/O module in the expansion module element library. Place the modules in turn. Double-click the required modules in turn in the right expansion module library according to the required module models and installation order, and drag the modules to the rack. To delete a module, select it and press Del.

Take AM600 for example. A maximum of 16 expansion modules can be mounted to the main rack.

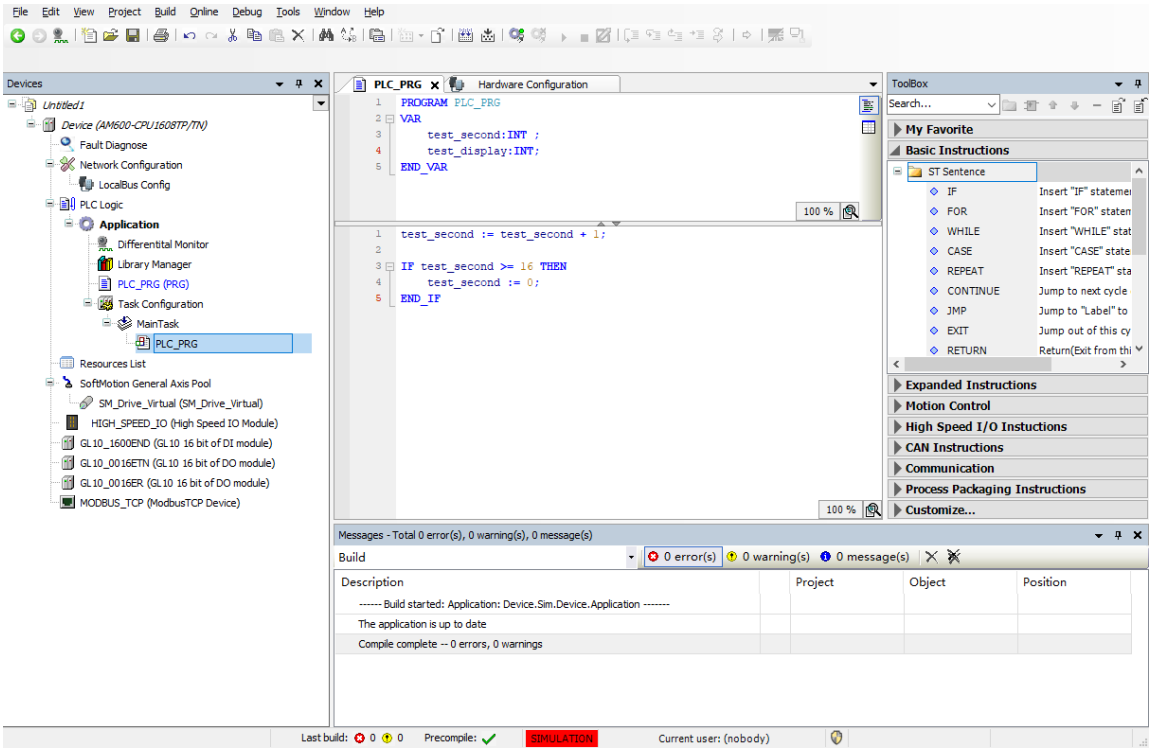
2.2.3 User Program Writing Operations

Double-click "PLC_PRG(PRG)" in the left device tree. The user programming page is displayed, on which the programming language is ST (selected in projection creation), as shown in the following figure. Similar to the C language, every variable can be used only after declaration. After a program statement is written, when you press Enter, the programming environment automatically displays a

declaration prompt. When you click "OK", the variable declaration window automatically adds this statement. This simplifies the program procedure.

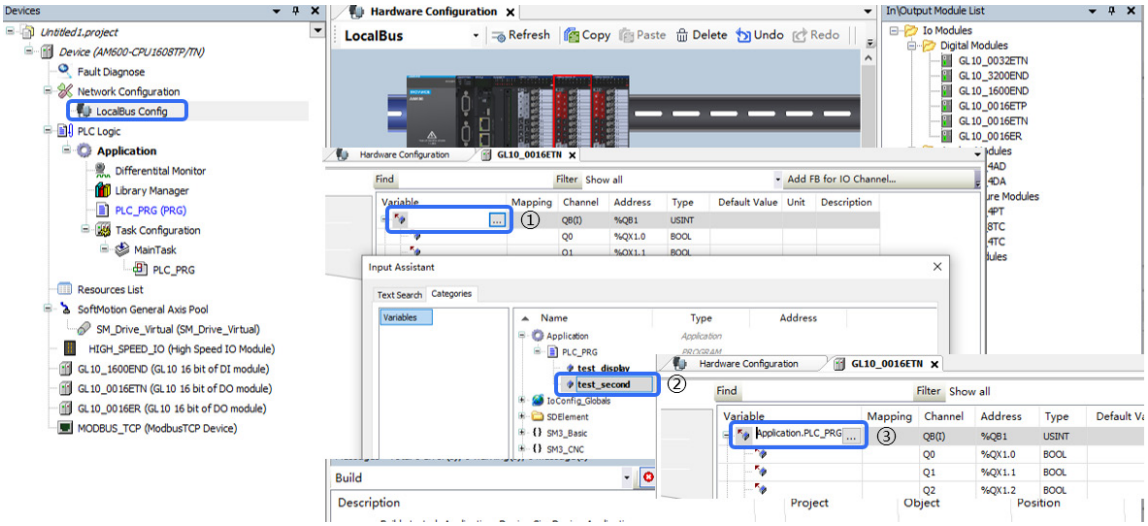


A programming example: Assign the value of the second variable to the first variable and progressively increase the value.



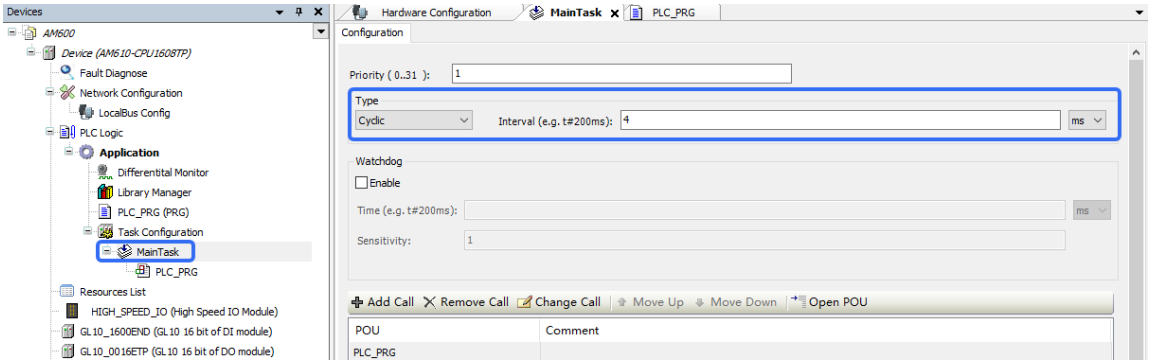
2.2.4 Linkage Configuration Between User Program Variables and Ports

On the "LocalBus Config" page, link the hardware ports with variables in the user program. As shown in the following figure, link the "test_display" variable with the output port of the first DO module. The configuration procedure is as follows:



2.2.5 Configuration of Execution Mode and Operation Cycle of User Program

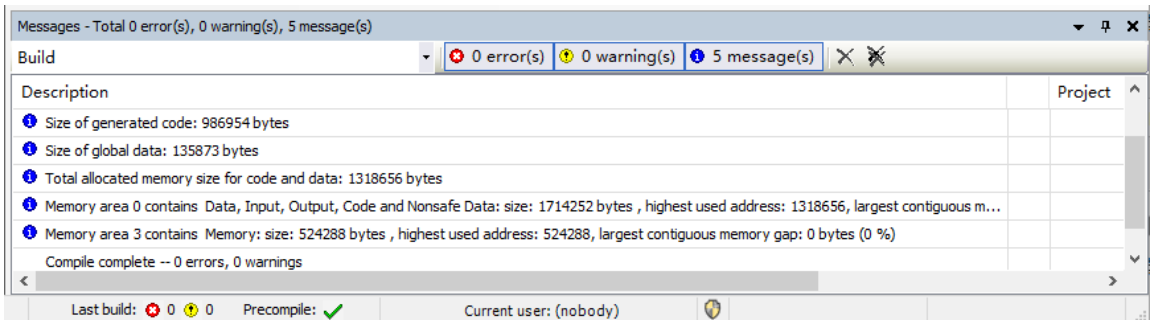
In the previous example, the sub-program is executed once every 20 ms by default. If you need to change the execution mode, for example, change it to repeated execution, scheduled execution, or execution at a specified interval, perform the following operations:



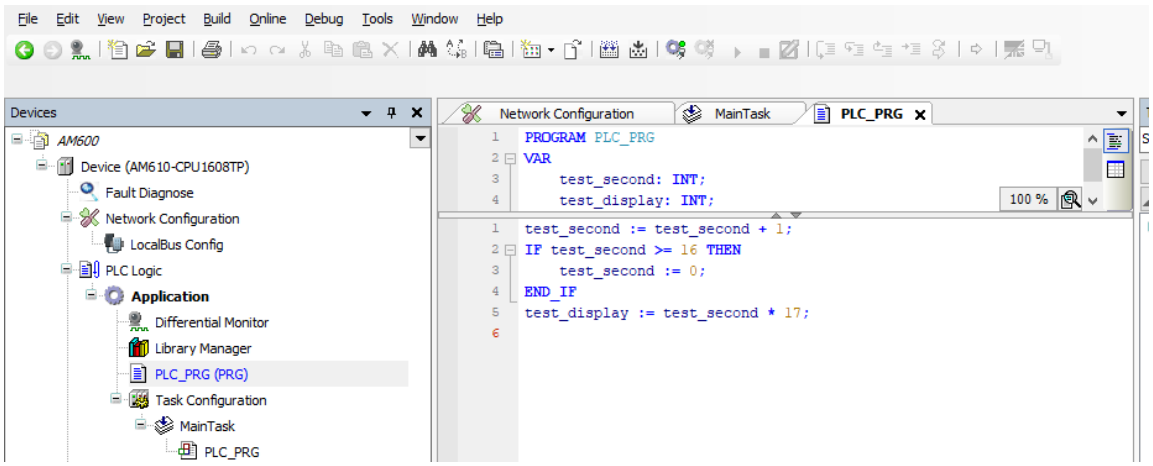
2.2.6 Compiling, Login, and Download of User Program

After programming, the program needs to be complied. Verify the compiling operation, locate errors based on the compiling information, and then repeat the operation until no error is reported.

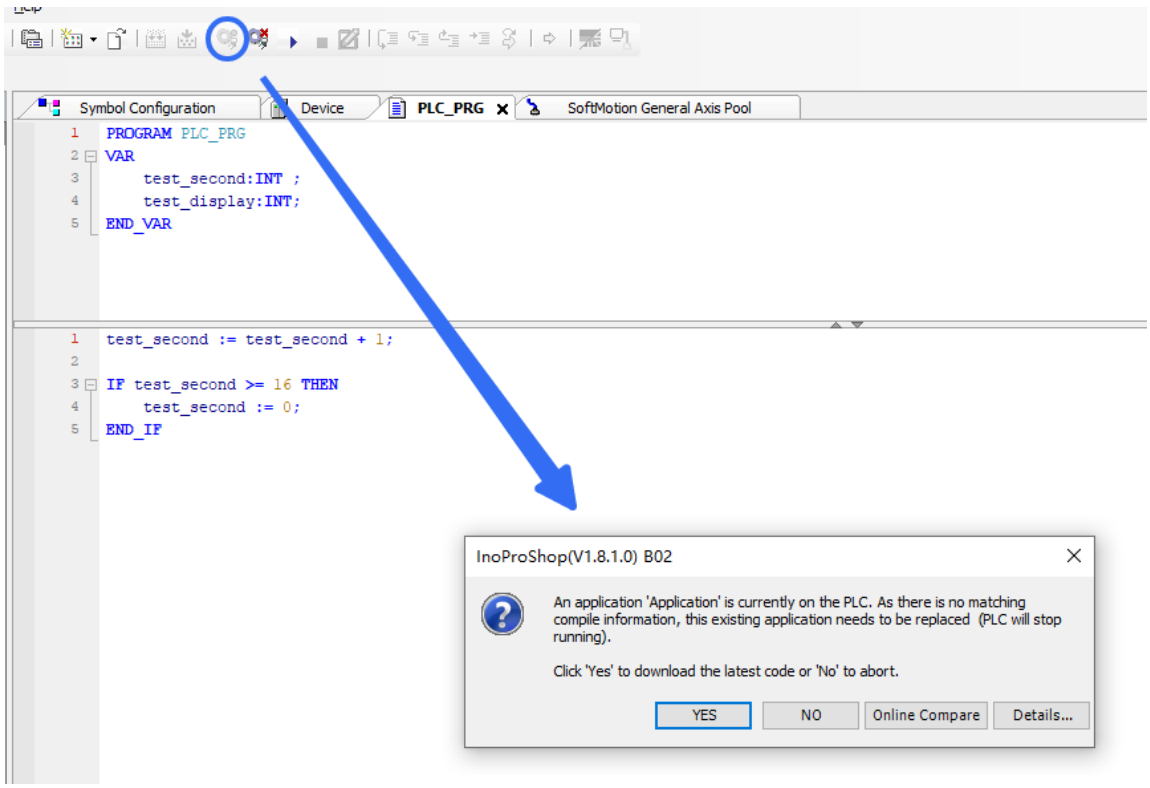
The compiling information is displayed in the following compiling information box:



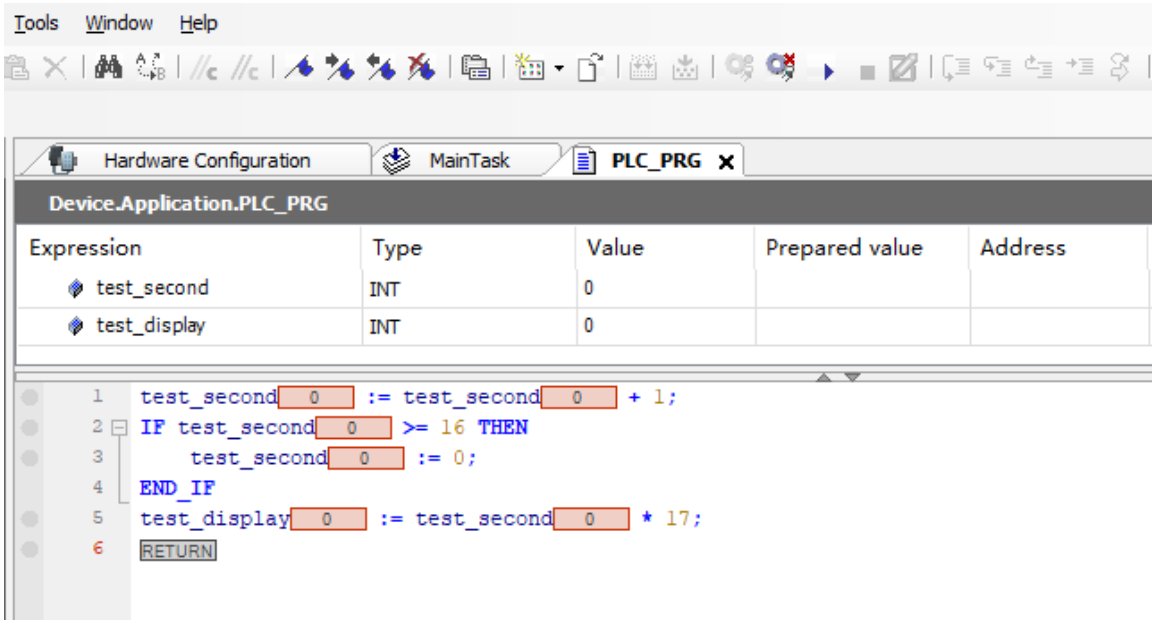
After confirming that no compiling error exists, click the "Online" and "Login" buttons, as shown in the following figure.



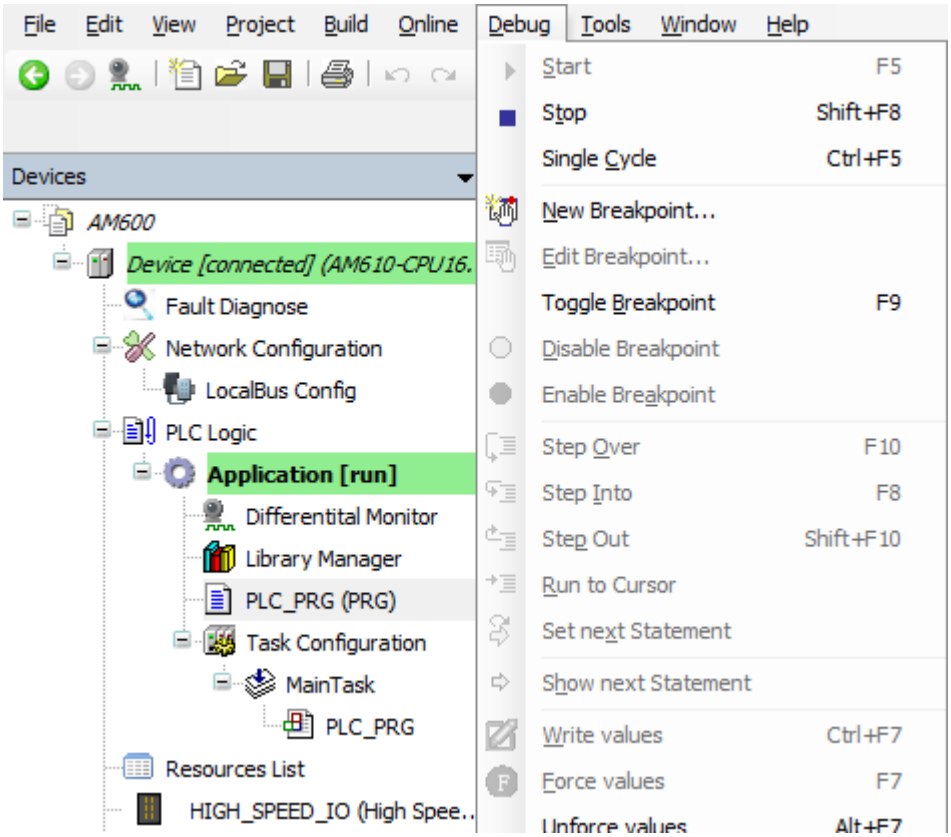
The following dialog box is displayed. Choose whether to create a project and continue to download the program.



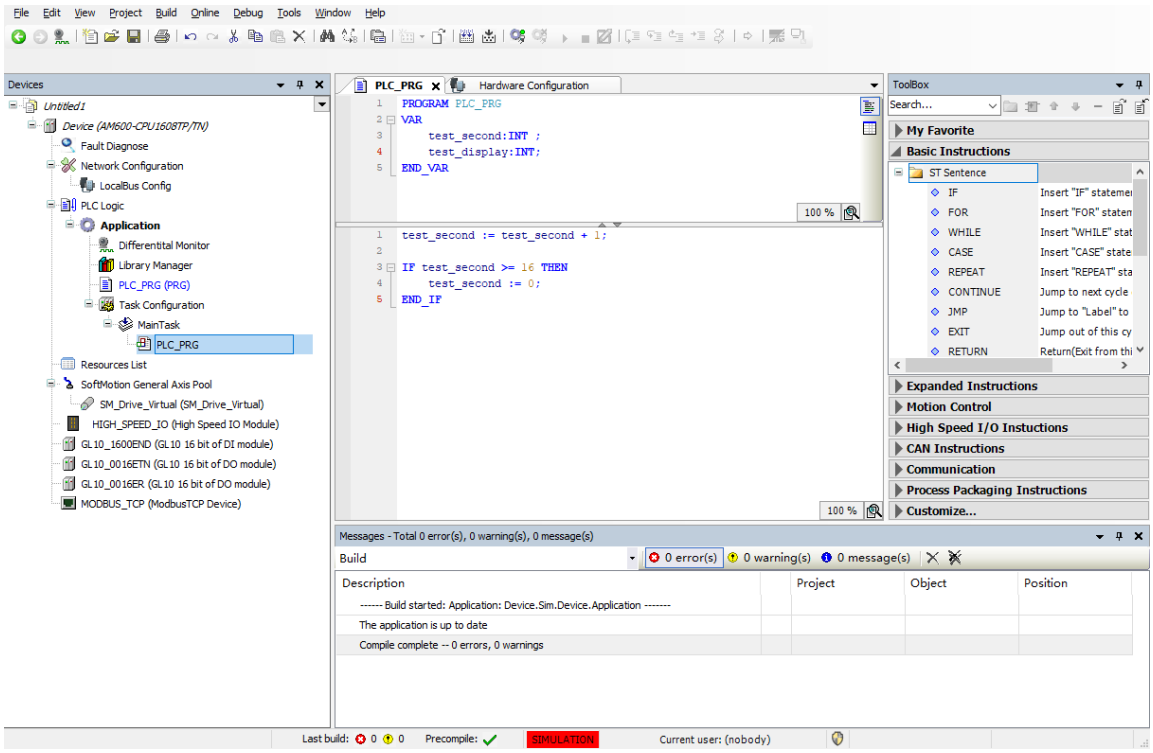
Click "Yes" to connect the host controller to the device and retain the connection. The initial status is "Stop", as shown in the following figure.



Choose "Debug" > "Start". The device enters the running state and starts to run the user program.



The following figure shows the monitoring interface of a running user program.



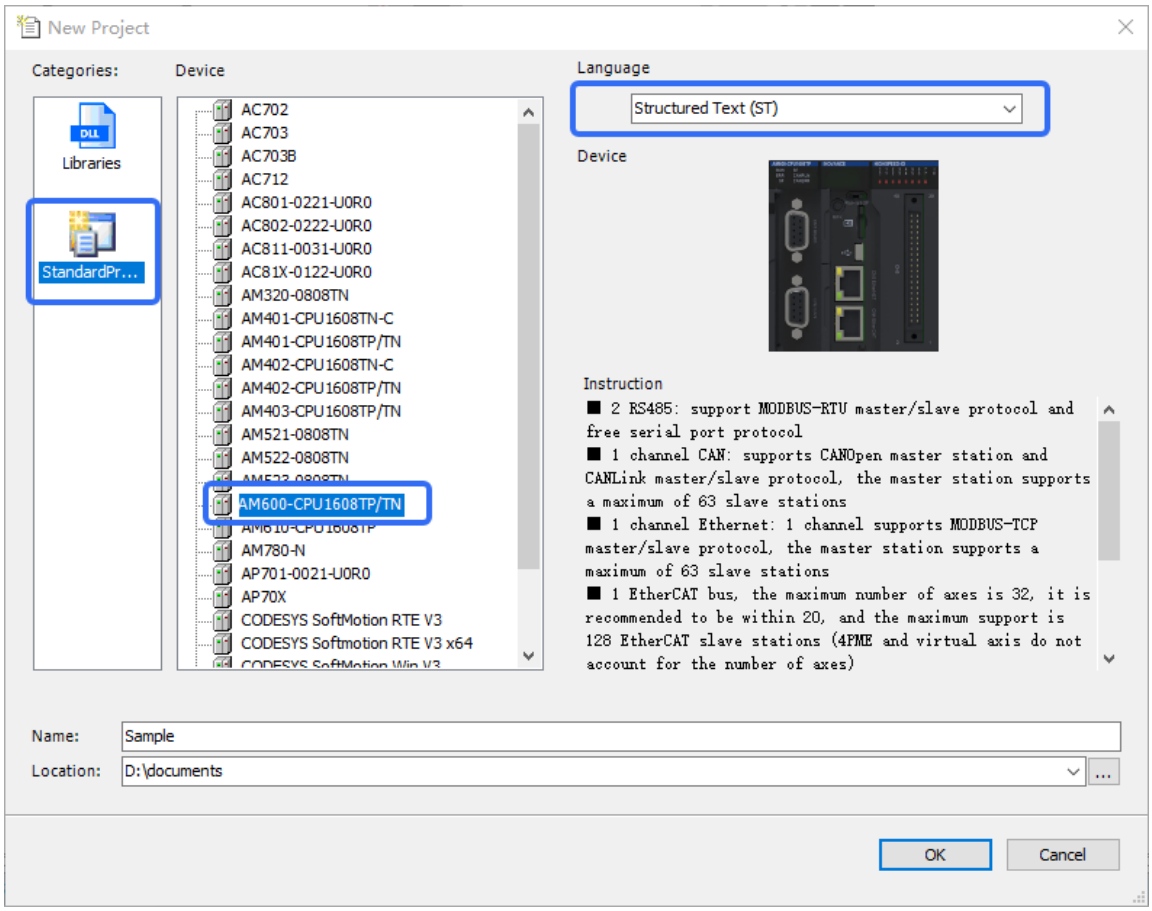
Check the first DO module behind AM600. The output status indicator cyclically counts in a binary mode.

2.3 Writing a Marquee Sample Project with InoProShop

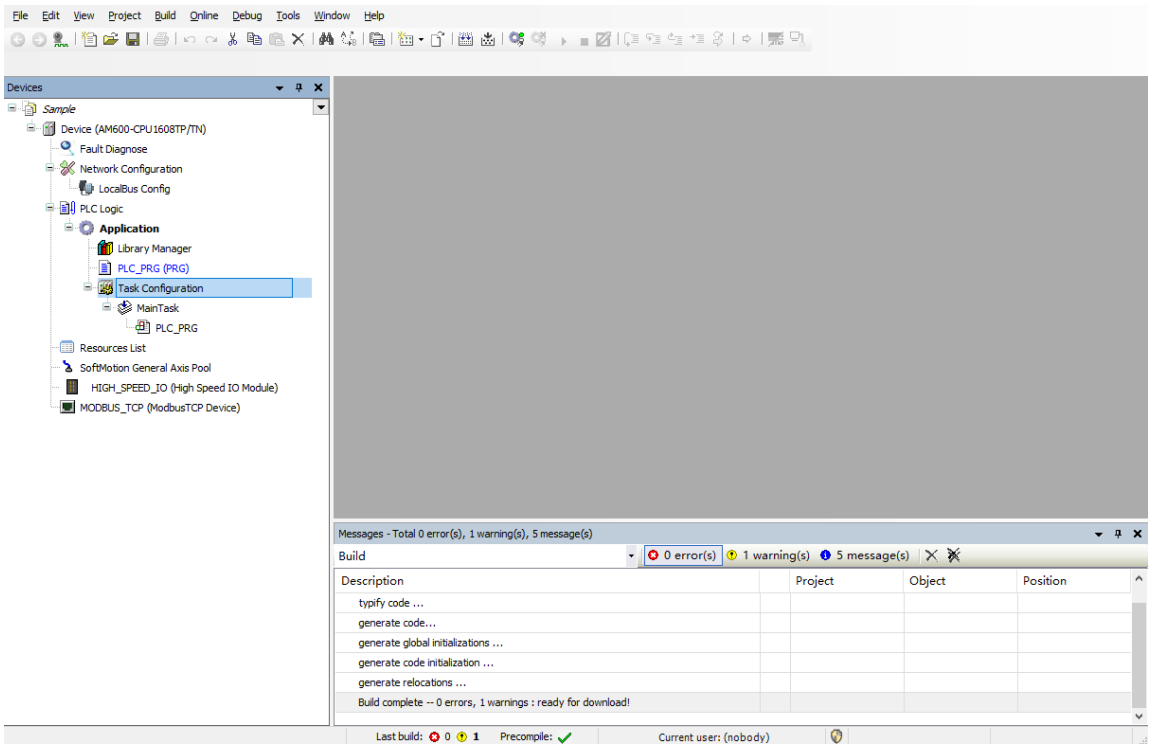
Launching the InoPro Programming Environment

Create a project:

Click on the top left corner of the menu bar or choose "File" > "New Project" to create a project. Select "Standard project", device type (model of the main module), and the programming language, and specify the project file name as well as storage path, as shown in the following figure:

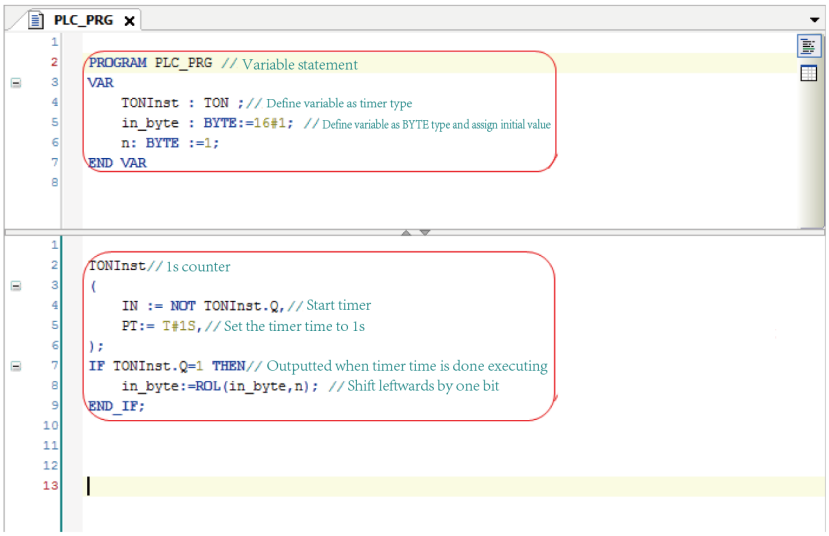


System Configuration and Programming Page



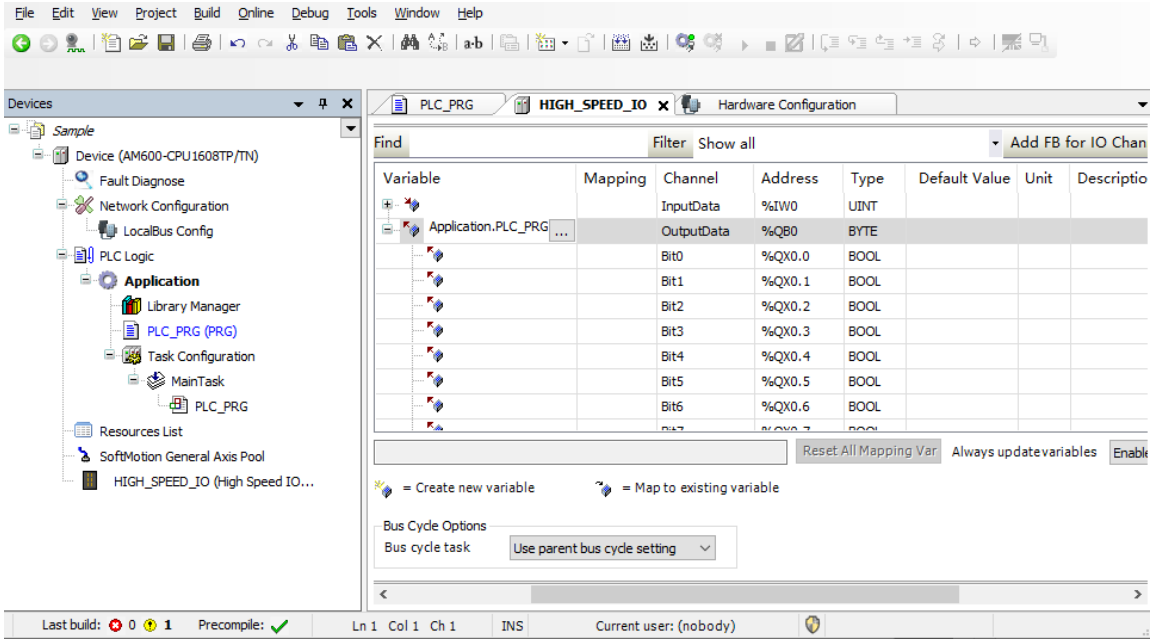
Compiling a Marquee Sample Project with the ST Language

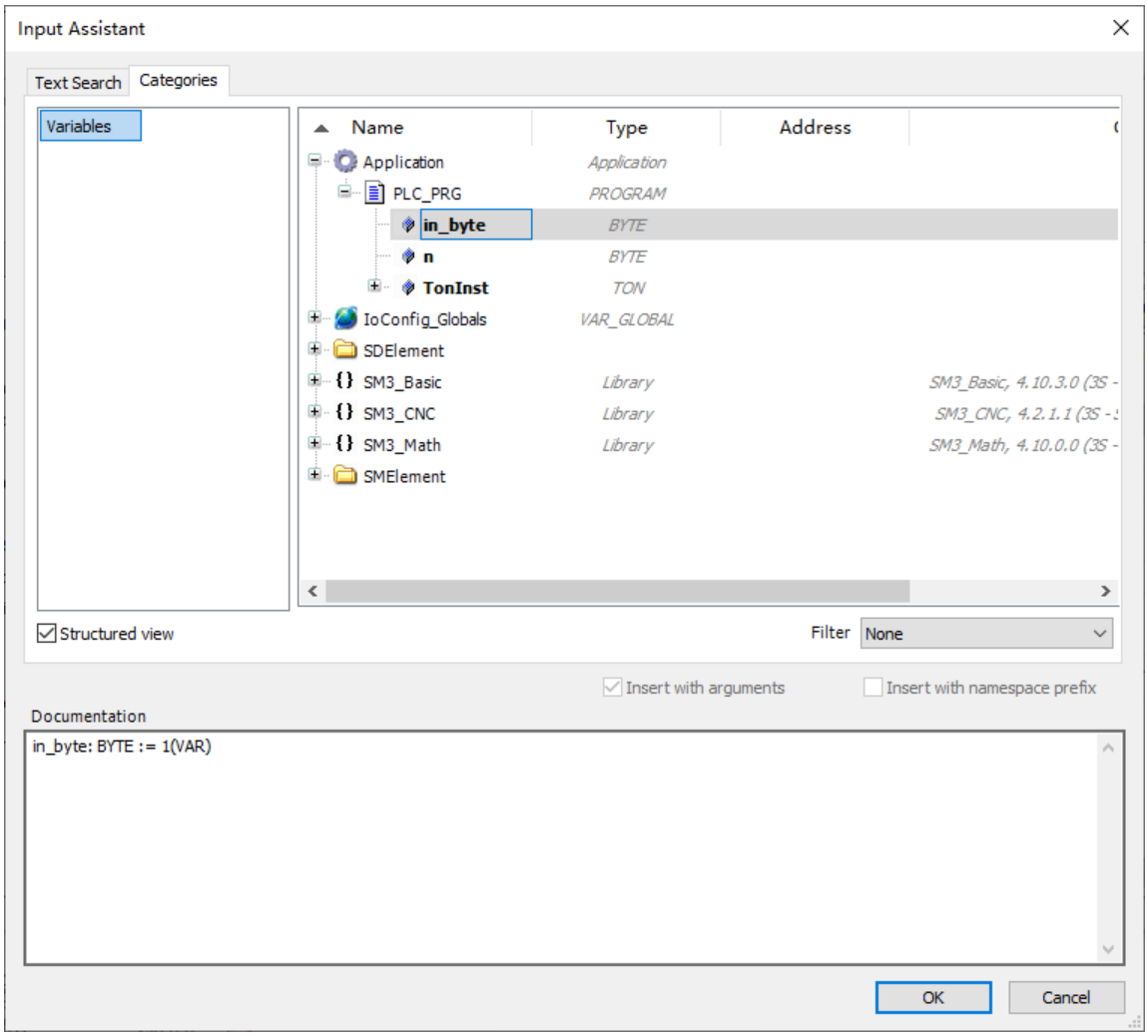
Double-click to open the "PLC_PRG" program organization unit.



Linked PLC Output I/O

Left shift the "in_byte" variable and eight output port links (bit 0 to bit 7) of the PLC. Observe the output indicator status change.





Simulation Debugging

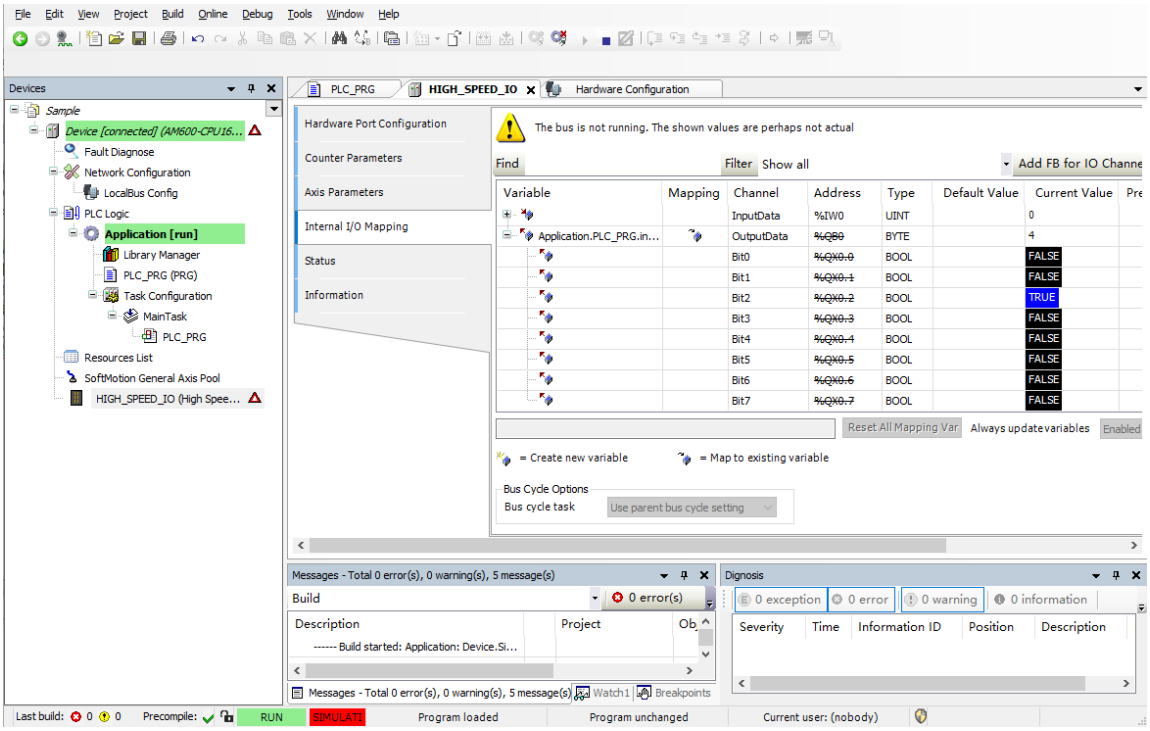
Click "Simulation" to enable the simulation function. You can view the I/O shifting status without linking to the PLC.

Downloading Program in Simulation Mode

Click "Login" to download the program in the simulation mode.

Run the PLC After Downloading

Monitoring I/O Change



2.4 How to Log in to the Main Module

2.4.1 Prerequisites and Operations of Main Module Login

Main Module Login means that InoProShop running on the PC communicates with the medium-sized PLC main module, so that the user program can be run, downloaded, started/stopped, and monitored. In addition, you can check and modify program parameters.

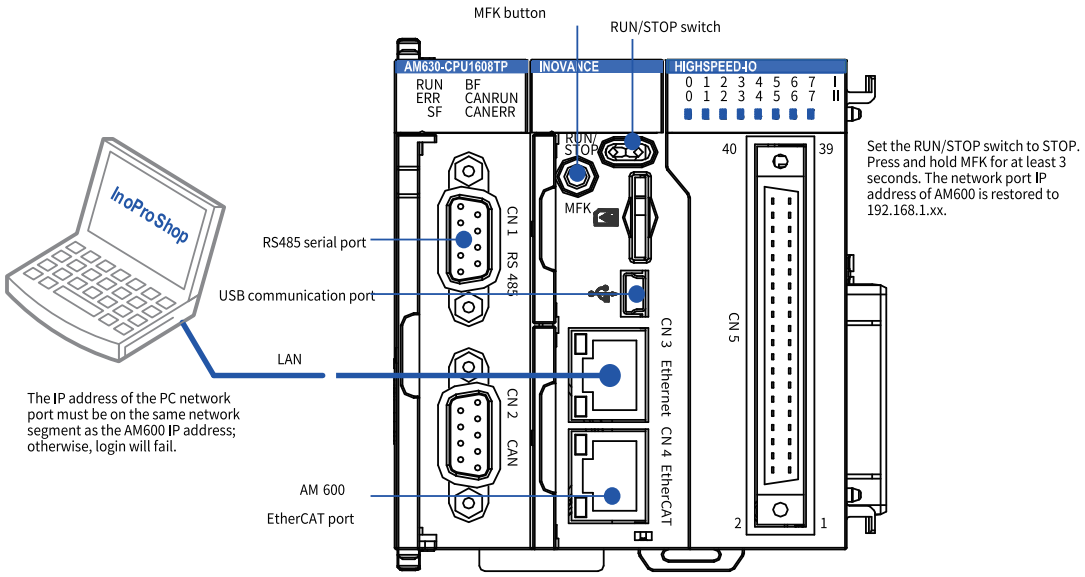
- You can log in to a medium-sized PLC through a LAN or a USB.
- The PC can be connected to the medium-sized PLC through a network cable in peer-to-peer mode, or connected to multiple medium-sized PLCs through a router or hub. Multiple PCs can also access the same medium-sized PLC.
- A PC can log in to the medium-sized PLC only when their IP addresses are in the same network segment; otherwise, InoProShop cannot detect the medium-sized PLC. For example, the default IP address of AM600 is 192.168.1.88. If a PC's IP address is 192.168.1.xxx (xxx ranges from 1 to 254, but is different from that in the IP address of AM600), InoProShop can detect AM600 and exchange data with it. Then, you can download and monitor the user program. If the IP address of AM600 has been changed to another network segment, the PC and AM600 cannot communicate with each other. In this situation, restore the default IP address 192.168.1.88 of AM600, and change the PC's IP address to 192.168.1.xxx. When a peer-to-peer connection is created, change the IP address of AM600 to the desired one.
- To log in to the PLC through a USB, connect the Mini-USB port. Wait for 2s to 60s until the device can be detected.

Precautions of USB connection:

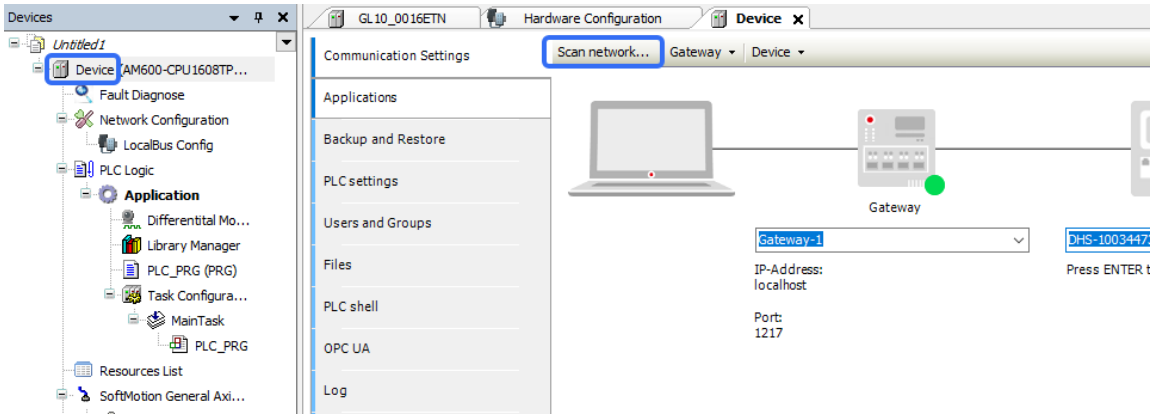
1. The USB drive is automatically installed during software installation. If not, you can find the file in the Common folder under the installation directory.
   Then update the drive in the Windows Device Manager. The drive is installed from the installation directory. After the USB connection is successfully set up, the Windows Device Manager displays the drive program installed.
2. If both USB connection and network connection are available, the network connection is used by default because its network scanning speed is faster.

2.4.2 Scanning Medium-Sized PLC in InoProShop

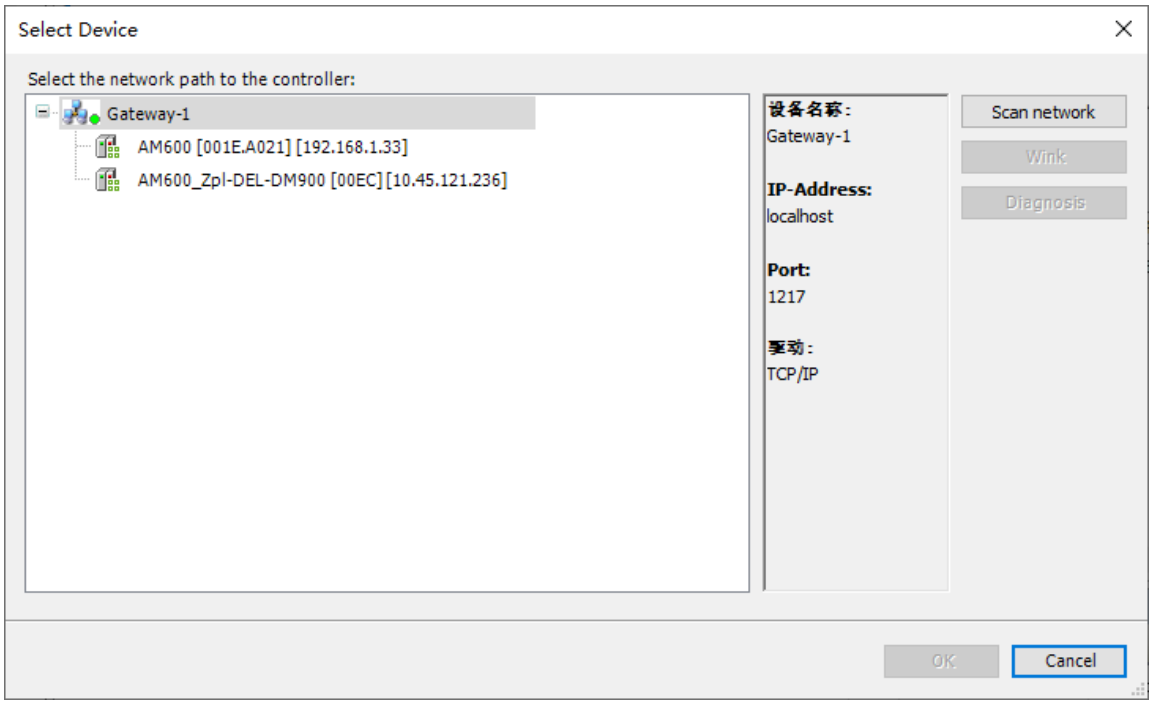
The PC can log in to the medium-sized PLC through LAN. Taking AM600 for example, the connection is as follows:



In InoProShop, double-click "Device (AM600-CPU-1608TP/TN)". The following page is displayed.



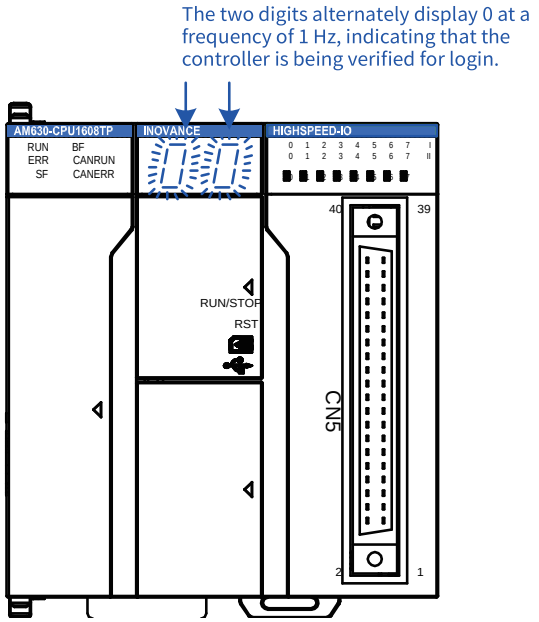
Click "Scan network...". The following page is displayed. In the left part of the window, click AM600-CPU. Its brief information is displayed in the right part of the window.



The previous figure shows two controllers, which are displayed in two rows:

- 9527[0003.6CAC.A0DF]: It is in the network segment and named 9527. The last two digits "DF" in brackets is the fourth bits in the IP address of AM600. "DF" is in hexadecimal format, and its decimal notation is 223.
- AM600CPU-V01B02D01WJZ01[0003.6CAC.A057]: It is another device in the network segment and named AM600CPU-V01B02D01WJZ01. After login, you can modify the device name so that you can easily identify the devices when there are multiple controllers.

Then, the two-digit LEDs on the AM600 or AM610 to which you logged in will display the character "0" alternately, as shown in the following figure.



The LEDs stop flicking after you click "OK" in the window displayed in InoProShop, and the original information is displayed.

Double-click the selected device, or select a device and click "OK". The host computer is connected to the device.

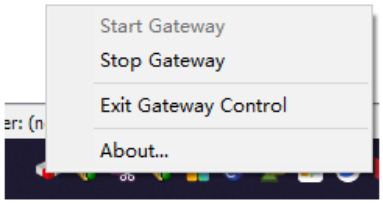
If the controller ID recorded in the project is different from the selected controller ID, the following information may be displayed. To connect to the network, click "Yes".

2.4.3 Solution to Device Scanning Failure

If InoProShop cannot detect AM600, the possible causes and solutions are as follows:

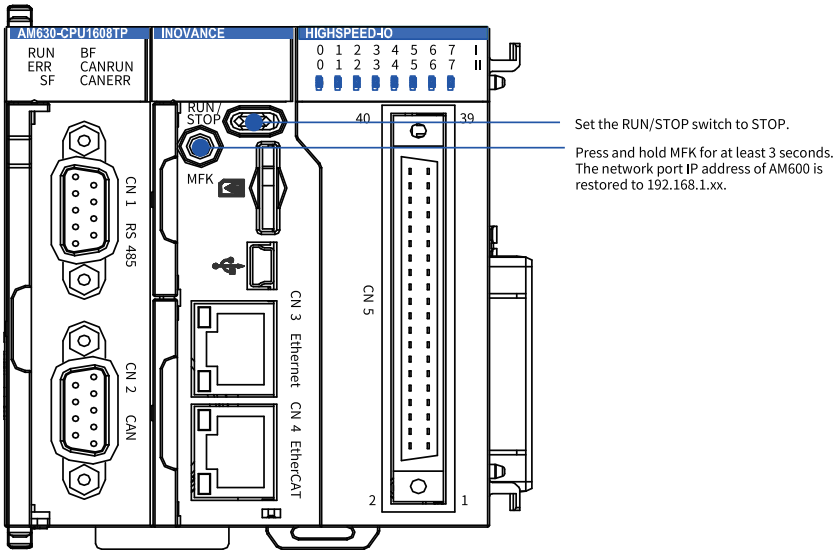
1. The CODESYS gateway is not started.
   Solution: Restart the gateway and run scanning again.





2. The IP addresses of the PC and AM600 are in different network segments.
   Solution: Set the IP address of the PC in the same network segment of the IP address of the AM600. If you forget the IP address of the AM600, restore it to the default IP address, and then set the IP address of the PC to the network segment 192.168.1.xxx. Then, the device can be scanned.

   a. Access the "Resource Manager" on the PC. Click "Local Connection" to check and modify the IP address settings.
   b. Restore the default IP address 192.168.1.88 of the AM600. After AM600 is powered on, slide the RUN/STOP switch to STOP. Hold down the MFK button and then release it until the IP address is displayed on the LEDs. To restore the IP address, a countdown starting from 10 is displayed on the LEDs. Before the restoration, there are countdown reminder of numbers "10" to "0". During the countdown, you can press MKF again to cancel the restoration.



The modified IP address takes effect immediately no matter whether you restore the default IP address of AM600 or change the IP address using InoProShop.

Once the scanning and network connection are successful, the following network status information is displayed on the "DeviceScan" page:

3 Basic Functions

3.1 Page Navigation

The left and right arrows are used to the previous editing position and the next editing position. You can click the relevant icon to quickly locate the part of the user program you want to modify.



3.2 Compiling a Command

The "Build" menu of the software provides functions such as "Build" and "Clear". The following figure shows the difference between functions of different versions.

Version 1.5.2	Version 1.6.0
	

Version 1.5.2	Version 1.6.0
Build	Check Application
Generate Code	Build
Rebuild	Rebuild
Generate runtime system files...	Generate runtime system files...
Clean	Clean
Clean all	Clean all
Pack user program	Pack user program

- "Check Application": Check whether the user program is compiled correctly.
- "Build": Compile and link all code into code that the PLC can execute.
- "Rebuild": Clear the information from the last compilation and re-compile and name the program.
- "Generate runtime system files": Used for R&D and test purposes. Not described in details here.
- "Clean": Clean the information of the last compilation and download operation.
- "Clean all": Clean the compilation information, download information, and reference information. The data compilation information of all libraries and projects will be refreshed.
- "Pack user program": This function will be described in another section.

After "Generate Code" is updated to "Build", to use the file generation function required for symbol configuration in label communication, click "Build" or the shortcut .

3.3 Resources List

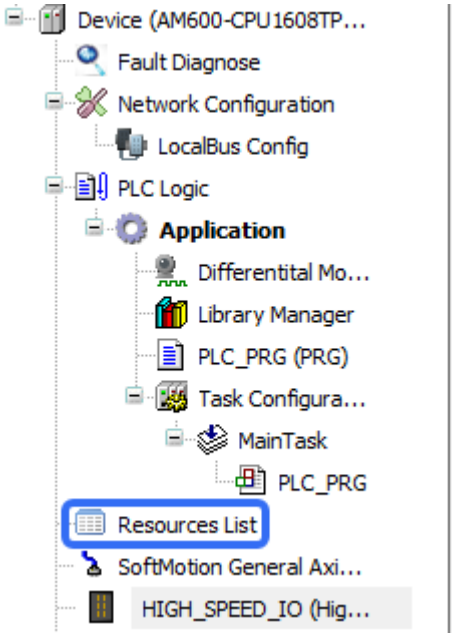
3.3.1 Overview

The resources list provides the following functions:

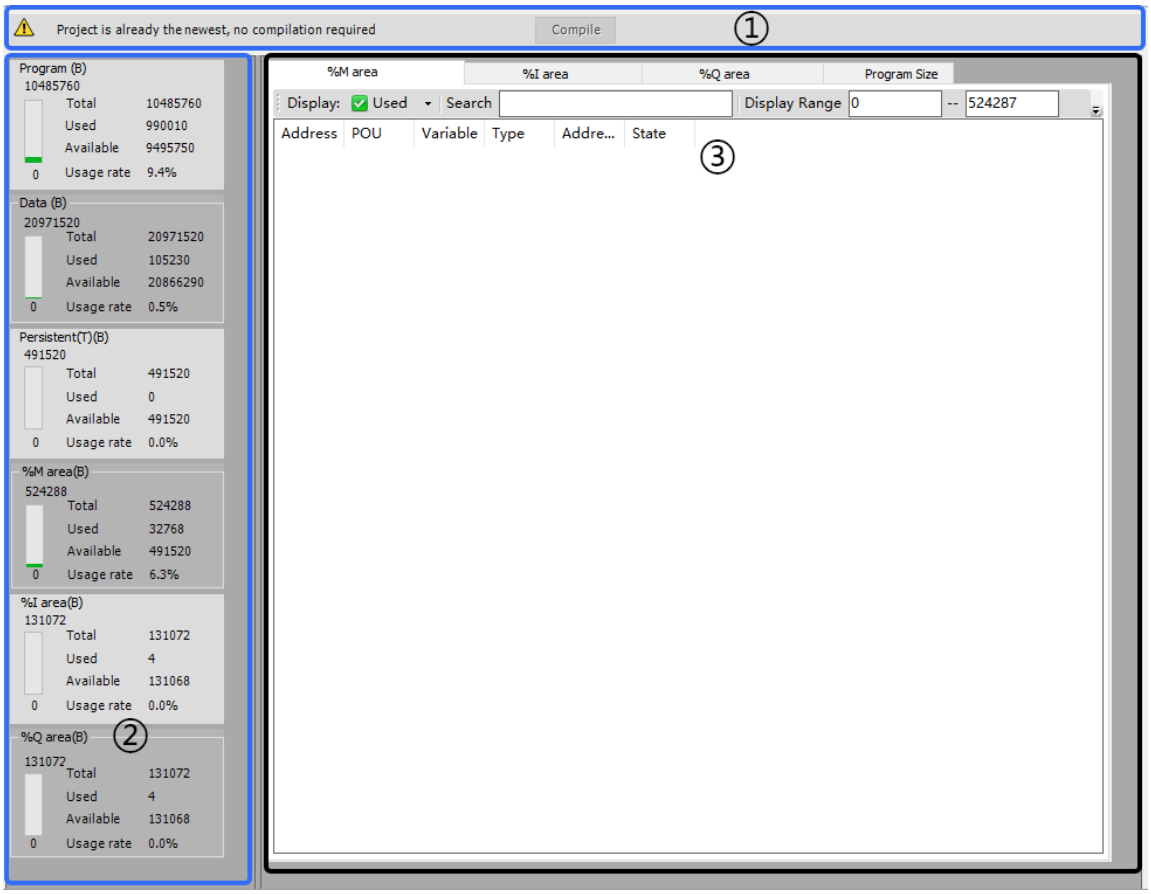
- Display the address occupation in the I/Q/M area, including occupied addresses, conflicting addresses, and unoccupied addresses.
- Display the occupied size, available size, and utilization of the program area, data area, retention at power supply area.

3.3.2 Features

After a project is created, you can double-click "Resources List" on the "Device" tree.

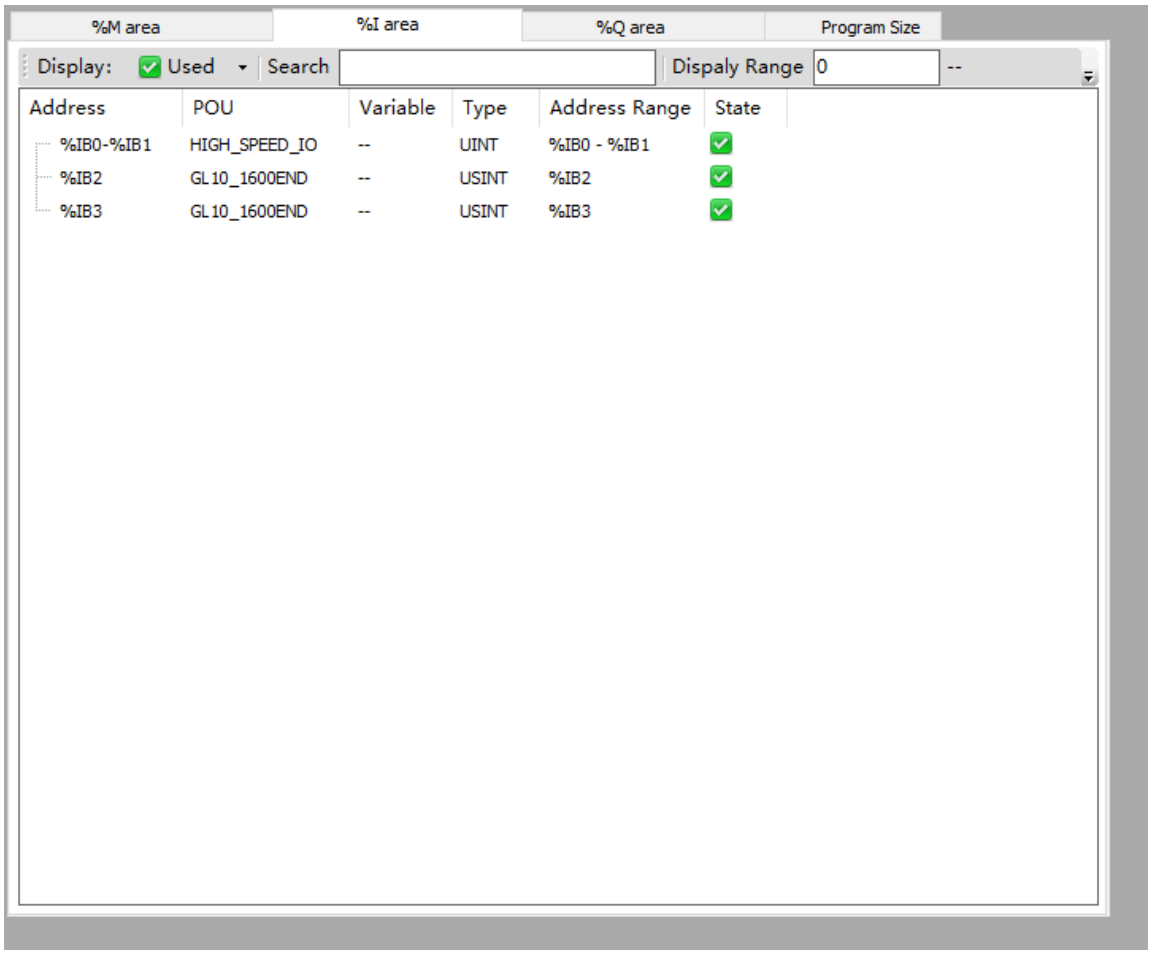


The following page is displayed.



No.	Item	Description
①	Data refresh section	Click "Generate Code". Based on the generated data, the system extracts the information required by this page and then refreshes the page. Or, you can choose "Build" > "Generate Code" to refresh the page. [Note] This page can be refreshed only after the code is generated.
②	Area utilization information section	This section displays basic information of the program area, data area, persistent area, and I/Q/M area, including the total capacity, occupied size, available size, and utilization of each area. [Note] The persistent area is divided into the original persistent area allocated by the system in the traditional mode, and the overlapped M area (covering the whole M area at most. The range of the M area can be adjusted) in the standard mode.
③	Display area of the utilization details of the I/M/Q area and the program block information	It displays the address utilization of the I/M/Q area, including variables associated with an address, address conflicting, and information of unoccupied addresses. Information of program blocks includes the type and size of function blocks called by the program. Other functions are provided here, including finding an address, address range setting, and page locating.

The following figure shows a page of information display area.



This page contains four display options: %M area, %I area, %Q area, and program block size, which display the utilization information of each area respectively.

Menu Options

The following menu options are provided on this page: the "Display" option, the "Find" option, display range, the "Application" button, and the paging option.

No.	Item	Description
1	"Display" option	In the %M, %I, and %Q areas, options are "All", "Used", "Conflict", and "Idle". On the program block size display page, options are "All", "Struct", and "Function Block".
2	"Find" option	This function allows you to make fuzzy search by address, POU, variable, or address range. When search conditions change, the displayed table will be refreshed accordingly.
3	Range setting	You can set an address range within the allowed range. The set address range takes effect only after the "Application" button is clicked. This function is not available on the program block size page.
4	"Applica-tion" option	After this button is clicked, the system filters the data according to the display conditions, search criteria, and display range, and those that meet the requirements are displayed in the table.
5	Paging option	To ensure the refresh efficiency, up to 1,000 address segment units can be added to the table, and extra addresses will be displayed in other pages. You can click the paging up or down icon or specify a page, and then the system jumps to the specific page.

Table

The table displays the utilization information of address segments meeting the selection conditions.

No.	Item	Description
1	Address	The system displays the address ranges in ascending order, with the smallest unit being byte. When the addresses occupied by a variable are not conflicting, the addresses will be displayed in the form of address range (such as "%MB0-%MB3"). One address can be associated with one or more variables. When multiple variables are associated with one address, sub-nodes will be displayed under this address in the "Variable" column.
2	POU	The POU name of the variable is displayed.
3	Variable	The variable associated with this address range is displayed.
4	Type	This column displays the variable type.
5	Address Range	This column displays the address range of the variable.
6	State	This column displays the state of the address range. When an address is occupied, the icon is . When an address conflicts with another address, the icon is . When an address is not occupied, no icon is displayed. When an address is occupied by the system, is displayed. Address ranges (%MB491520 to %MB524287) occupied by the system are displayed in the M area of the AM600 and AM400 series. When a variable is associated with such an address range, no error is reported and the icon is displayed on this page. Addresses occupied by the system are not displayed in the %M area of the AM800 and AM700 series. Note: • Address conflicting is detected by bytes. When different variables use different bits of the same address, address conflicting is marked. • When a user-defined variable conflicts with an I/O address allocated by the system, it means that this address is also occupied by other variables. In this case, you can judge the problem based on whether a function actually occupies this address.

PLC Direct Address Storage Area

The direct address storage area varies with PLCs. PLC data is not retained upon power failure for the %I and %Q areas, but is retained for the %M area. The AM600, AM610, AM401, and AM402 programming systems provide the 128-KB (byte) input area (I area), 128-KB (byte) output area (Q area), and 512-KB storage area (M area). The first 480 KB of the storage area can be used directly, whereas the last 32 KB are used by the system, mainly as soft elements, and cannot be used directly by users. During programming, users can directly access addresses or define a variable, map the variable to an address, and then access the address. The following table lists storage areas and the address ranges they use.

Area	Use	Size	Address Range
I area (%I) 128 KB	For users	64kWords	%IW0 to %IW65535
Q area (%Q) 128 KB	For users	64kWords	%QW0 to %QW65535
M area (%M) 512 KB	For users	240kWords	%MW0 to %MW245759
	For SD elements	10000Words	%MW245760 to %MW255759
	For SM elements	10000BytesWords	%MB511520 to %MB521519
	Reserved	2768Bytes	%MB521520 to %MB524287

The AC800-series programming system provides a 128-KB input area (I area), a 128-KB output area (Q area), and a 5-MB storage area (M area). The AC800 series does not support SD and SM soft elements

and addresses in the %M can be used without restriction. The following table lists storage areas and the address ranges they use.

Area	Use	Size	Address Range
I area (%I) 128 KB	For users	64kWords	%IW0 to %IW65535
Q area (%Q) 128 KB	For users	64kWords	%QW0 to %QW65535
M area (%M) 5 MB	For users	2.5MWords	%MW0 to %MW2321439

3.4 Symbol Configuration

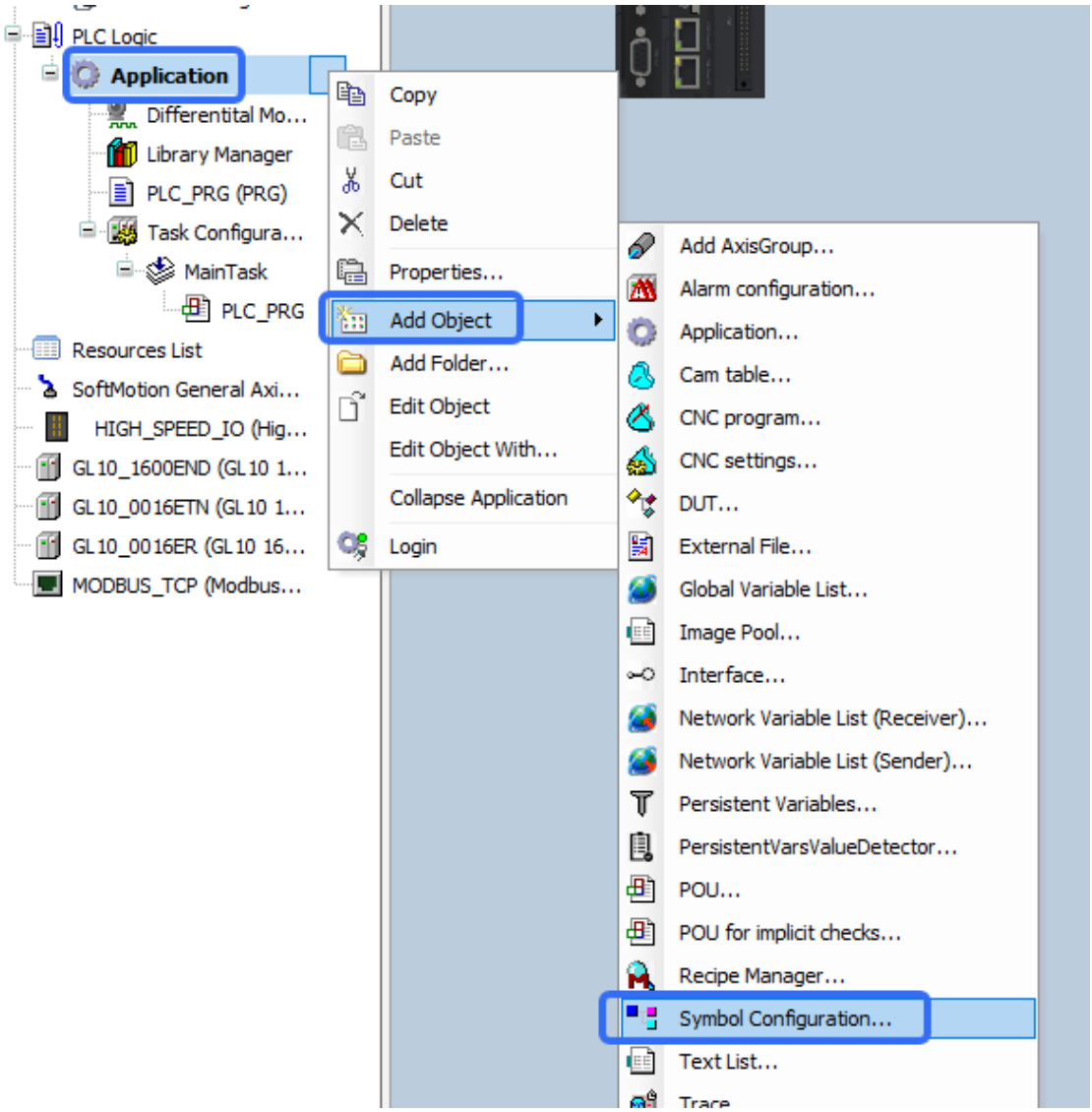
The symbol configuration function allows you to define symbols of specific access permissions for project variables. Then you can use these symbols to access variables from an external device such as an OPC server. When the code of a project is generated, a symbol configuration file named in the format of suffixed by <project name><device name<application name.xml is also generated under the project directory, and the file contains symbol description. For example, you can import the file to an HMI for label communication or variable access.

Adding Symbol Configuration

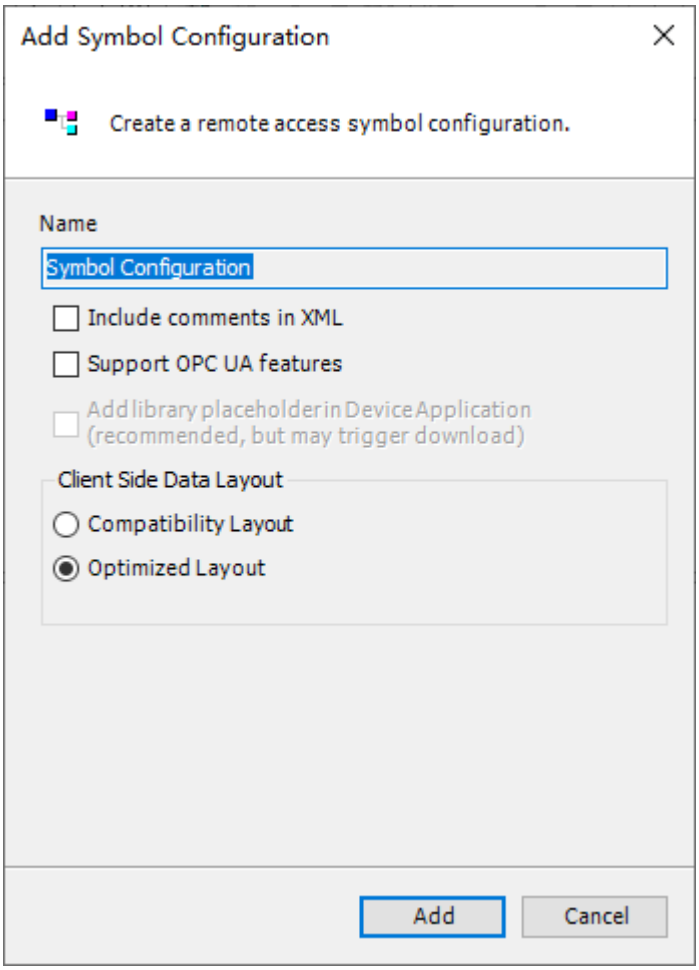
Note

The symbol configuration is generated only when no compile error exists.

On the project tree, right-click "Application", and then choose "Add Object" > "Symbol Configuration".



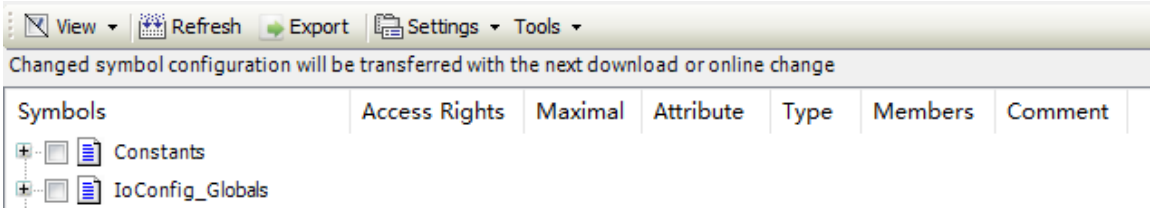
The following window is displayed.



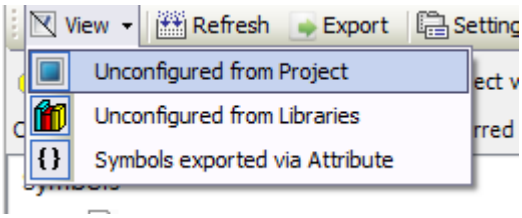
Option	Description
Include comments in XML	When this option is checked, the exported XML file contains variable comments.
Support OPC UA features	When this option is checked, OPC UA can access symbol variables.
Compatibility Layout	It is consistent with the type member definition offset size. If the type member does not fully support symbol access, the offset size is based on the actual compiled offset, with a gap between members.
Optimized Layout	The offset is calculated based on the selected type member. If this option is not checked, the offset is not calculated.

Symbol Configuration Page

The following figure shows a symbol configuration page.

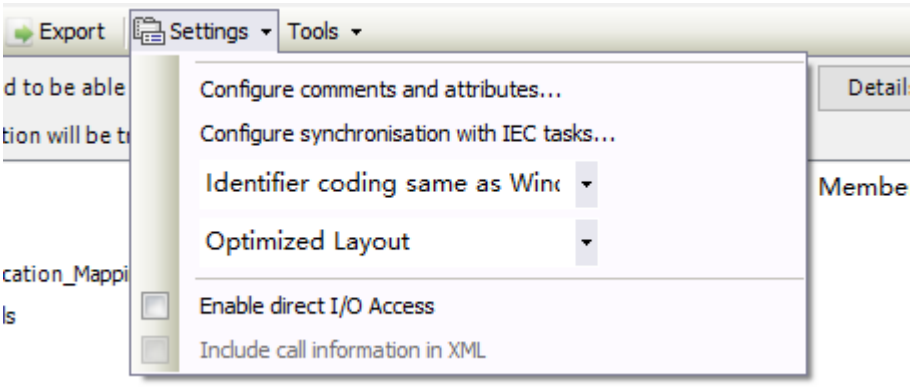


The following figure shows the options of the "Show" menu.

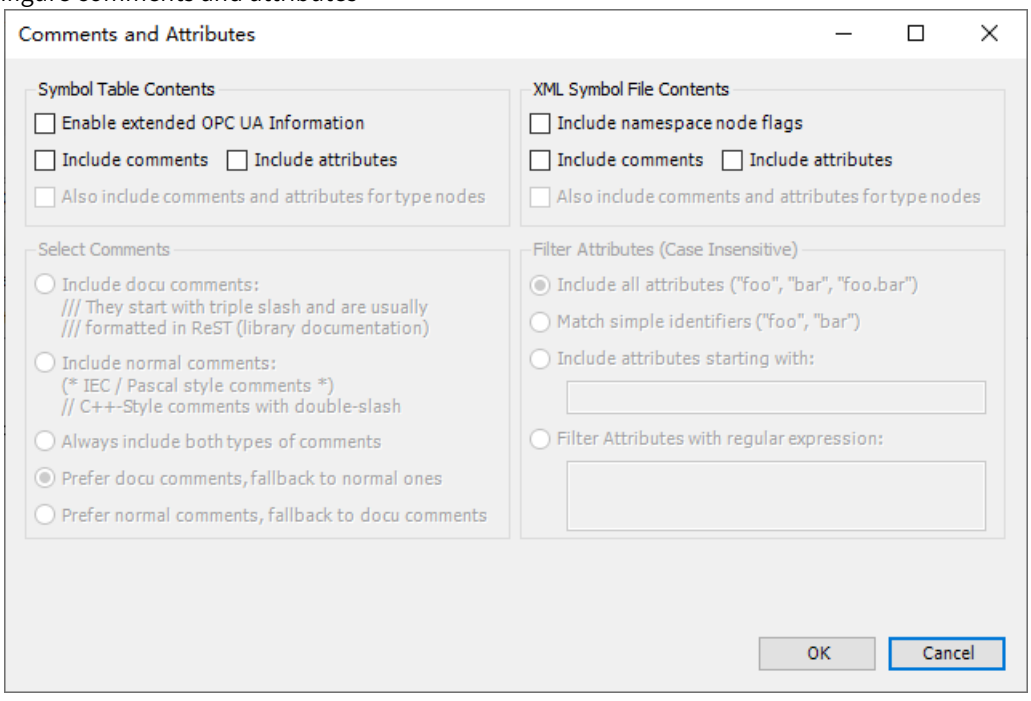


Name	Description
Unconfigured from Project	All variables of the project are displayed.
Unconfigured from Libraries	All variables of the referenced library are displayed.
Symbols exported via Attribute	Variables exported based on attribute settings ({attribute 'symbol' := read}) are displayed.

The following figure shows the "Settings" menu.



- Configure comments and attributes



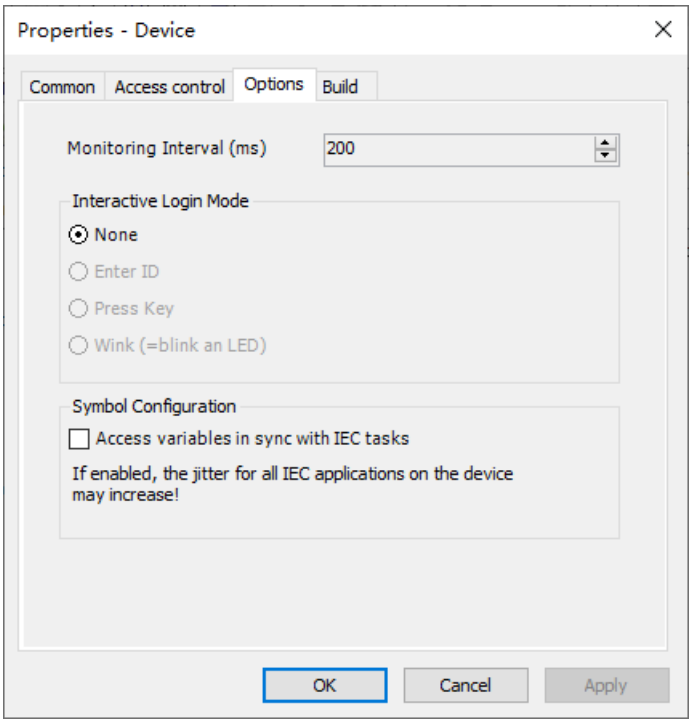
Data downloaded to the PLC is displayed in the upper left part, while the data format of the exported XML file is displayed in the upper right part.

Symbol: It generally indicates a variable. The symbol attribute indicates the variable feature (Attribute information.)

Comment format: It indicates the format of comments for downloading or display.

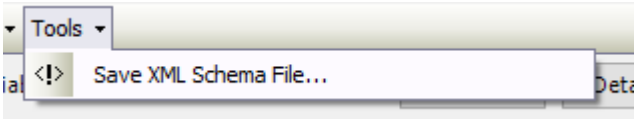
Attribute filtering: It indicates which attributes are contained when the XML file is exported or downloaded to the PLC. Filter rules are "Include all attributes", "Match simple identifiers", "Include attributes starting with", and "Filter Attributes with regular expression".

- Configure synchronisation with IEC tasks



This function indicates whether a symbol variable accessed by other interface is synchronized with IEC tasks. When an IEC task is being executed, do not access the symbol variable; otherwise, the variable cannot be synchronized with the IEC task.

- Options of "Tools"



You can export the XML data model which can be referenced when a third party parses offline symbols.

Symbol Configuration Example

Create global variables A_0, A_1, and A_2, and apply at least one variable to the user program, as shown in the following figure.

Note

If no variable in a global variable list is applied to the user program, this variable table is not available on the "Symbol Configuration" page.

```
VAR_GLOBAL
    A_0:ARRAY[0..9] OF INT;
    A_1:ARRAY[0..19] OF DINT;
    A_2:BOOL;                    A_2:=1;
END_VAR
```

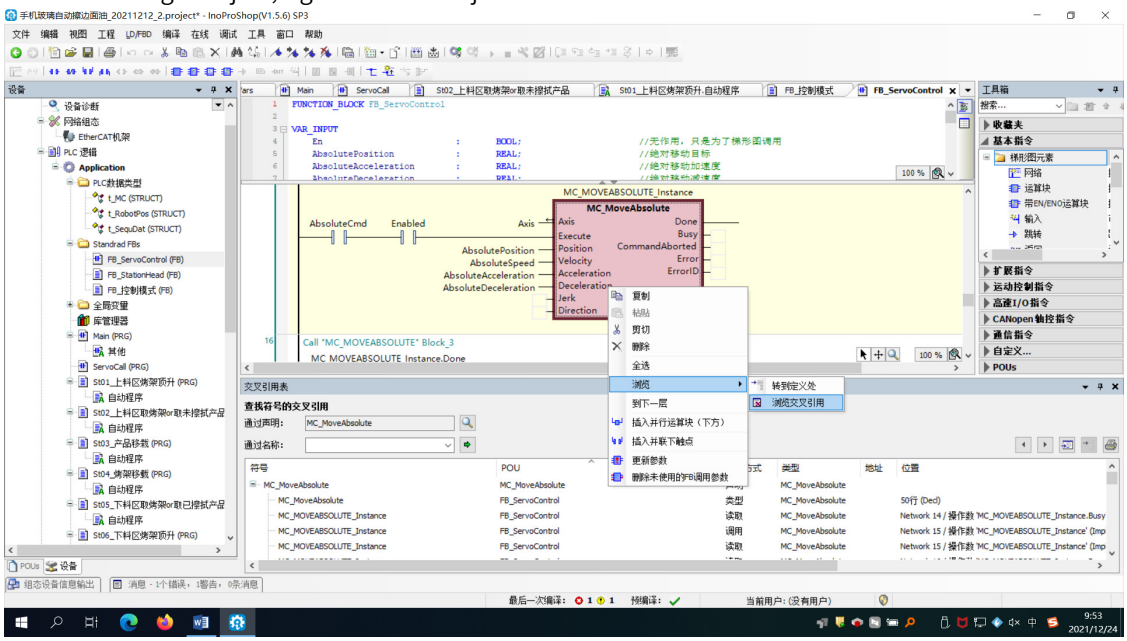
Choose "Application" > "Add Object" > "Symbol Configuration". On the page displayed, check "Include comments in XML". In the upper toolbar, select "Check Application". The variable table and variables are displayed on the symbol configuration page. Check variable tables you want to configure, and allocate access permissions (options are read-only , write-only , and read-write). In the upper toolbar, select "Compile (Generate Code)".

You can locate the .xml file generated under the directory of the project and import the file to IT7000 for label communication.

3.5 Cross References

The "Cross References" function allows you to quickly locate the call position of the target object.

1. Locate the target object, right-click the object to call out "Cross References".



2. In the "Cross Reference List" under the project, view the call information of the target object in the project. Double-click an item in the cross reference list. The system displays the specific call position of the object in the project.



3.6 Watch List

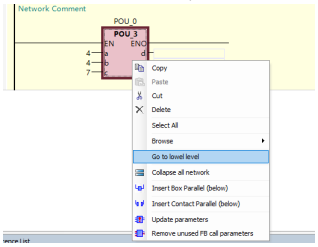
The watch list function allows you to monitor variables and addresses. When a program is running, you can view the data type and current value of monitored variables through the watch list or input a value for variables.

1. In the toolbar, choose "View" > "Monitoring", and then add a monitoring view.
2. Under a project, add a variable or address you want to monitor.

3.7 Going to a Lower Level

User-Defined Function Block

1. Locate the target function block, right-click the block, and then click "Go to lower level".



2. The information of the function block is displayed.

Creating an "Action" Under a Program Organization Unit

1. Locate the target "action", right-click the action, and then click "Go to lower level".
2. The "action" information is displayed.

Creating a "Method" Under a Program Organization Unit

1. Locate the target "METH", right-click "METH", and then click "Go to lower level".
2. The "METH" information is displayed.

3.8 Project Version Upgrade

Caution

This function is supported by InoProShop V1.8.0.0 and later versions.

Procedure

1. Open the dialog box for project version upgrade.
   - In InoProShop V1.8.0.0 and later versions, open a project of an earlier version. If an upgradeable item is available, the "Project Environment" dialog box is automatically displayed.
   - In the toolbar, choose "Project" > "Project Environment". The "Project Environment" dialog box is displayed.
   - In the status bar, double-click . The "Project Environment" dialog box is displayed.
2. (Optional) To specify the target version of the Softmotion library or EtherCAT library, select the target version from the drop-down list of "Newest". The system upgrades the project to the latest version by default.
3. Click "Upgrade". All upgradeable items are upgraded to their latest version (if a specific version is specified, the Softmotion library or EtherCAT library is upgraded to the specified version), and the old project information is backed up to the directory of the original project.
4. (Optional) To view the upgrade details, click "Details".

Upgrade description

- Compiler
  You may retain the compiler at its original version or upgrade it. After upgrade, the project compiler version automatically matches the Omron library version according to the following mapping relationships. The match is a prerequisite for successful project compilation.

Version of HC_OmronUtils Library	Compiler Version
1.1.0.0 (initial version)	• 3.5.11.10 • 3.5.11.11
1.2.0.0 or later	3.5.11.70 or later

Note

On the "Compile Options" page, the upgrade option for the compiler version is disabled. To access the "Compile Options" page, choose "Project" > "Project Settings", and then click "Compile Options".

- LD/FBD programming language
  You may retain the LD/FBD programming language at its original version or upgrade it to the latest version. Projects created in InoProShop 1.5.2 and later versions are of the latest version by default.
- SoftMotion library
  Open a project and view the version of the SoftMotion library.

Original SoftMotion Version	SoftMotion Version After a Project Opened
1.0.0.0 to 1.3.0.3	Upgraded to the latest version by default
Initial version	Initial version retained
1.4.0.0 or later	Original version retained

The library will be upgraded to the latest version.

- EtherNet/IP library
  Open a project and view the version of the EtherNet/IP library.

Original EtherNet/IP Version	EtherNet/IP Version After a Project Opened
Initial version	• Open a project that is created in a version earlier than InoProShop V1.7.3 SP5 and the project is not saved in InoProShop V1.7.3 SP5/SP6: the project is automatically upgraded to the latest version by default. • Open a project that is saved in InoProShop V1.7.3 SP5/SP6: the original version is retained.
1.0.0.0 or later	Original version retained

The library will be upgraded to the latest version.

- EtherCAT library
  You may retain the EtherCAT library at its original version or upgrade it to the latest version.
- Omron library (HC_OmronUtils)
  Open a project and view the version of the Omron library.

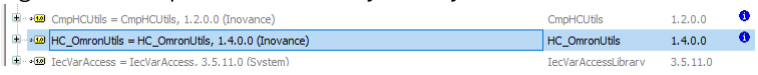
Original Omron Library	Omron Version After a Project Opened
Initial version	Original version retained
1.0.2.0 or later	If the project directly references the Omron library and the Omron library in the third-party library, the original version is retained.

If the project is upgraded or a new project is created, the Omron library directly referenced by the project is upgraded to the latest version, and the Omron library in the third-party library retains the original version. To upgrade the Omron library in the third-party library, upgrade the project by the source code of the corresponding library.

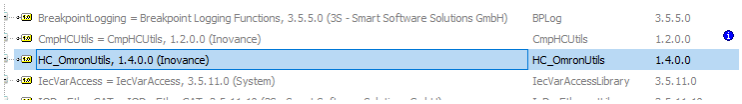
If you create a project or upgrade a project in InoProShop V1.8.0.0, the version of the directly referenced Omron library must be consistent with the version of the Omron library indicated in the project version information. If the version of the Omron library modified in the library manager is inconsistent with that indicated in the project version information, a compilation error will be reported. You are not recommended to manage the Omron library version through the library manager.

You are not recommended to use the Omron placeholder library in a third-party library, because the version of the placeholder library changes with the position of the placeholder in the project. The embedded library is recommended.

- The following figure shows a placeholder library with symbols.



- The following figure shows an embedded library with a comma.



- Local high-speed I/O library
  You may retain the local high-speed I/O library at its original version or upgrade it to the latest version.
- Configuration
  For projects created in InoProShop 1.5.2 and later versions, the network configuration, hardware configuration, and device diagnosis functions are upgraded to the latest version by default.
- Device version
  You may retain the device at its original version or upgrade it to the latest version.

3.9 Project Safety Management

3.9.1 Project File Encryption

Project files can be protected by password to prevent unauthorized use. After a password is set for a project file, you need to input the correct password before opening the project again and using the project files normally.

Caution

Do memorize the password for the project file. If lost, this password cannot be retrieved and the project file will be permanently lost.

Procedure

1. In the toolbar, choose "Project" > "Project Settings". The "Project Settings" dialog box is displayed.
2. Click "Security". Check "Enable project file encryption", input the current password and a new password, input the new password again, and then click "OK". The project file is encrypted.

Note

When you set a password for a project file for the first time, you do not need to input the current password.

Subsequent Operation

1. When the project is opened again, the "Password" dialog box is displayed.
2. Input the project file password and click "OK".

3.9.2 Project User Authorization Management

This function allows you to configure user authorization on the current project, such as modifying, browsing, adding, deleting, and removing sub-items of the project.

The project user authorization can be managed by users and groups. The system provides two groups (Everyone and Owner) and one user (Owner) by default. A user can be a member of a group, and a group can be a member of another group. The Everyone and Owner groups can only be renamed but cannot be deleted. All new users will be automatically added to the Everyone group. The Owner user cannot be deleted, but its name and password can be changed. Its initial password is empty. It is recommended to change the initial password of Owner; otherwise, any user can log in to the system through the Owner account, making the authorization allocation exist in name only.

After adding a user and a group, you need to allocate authorization by group (to the group Everyone) for node functions on the device tree. The default authorization is "Authorize". You can manage the project based on the allocated authorization after login.

1. In the toolbar, choose "Project" > "Project Settings". The "Project Settings" dialog box is displayed.
2. Click "Users and Groups". The "Users and Groups" page is displayed.

Adding a user

1. On the "Users and Groups" page, click "User". On the tab page displayed, click "Add", fill in the user information, and then click "OK".

Note

- New users are added to the group "Everyone" by default. To add a new user to another group, check the specific group.
- To edit a user, on the "Users and Groups" page, click "User". On the tab page displayed, select the user, and click "Edit". In the dialog box displayed, modify the user information, and then click "OK".
- To delete a user, on the "Users and Groups" page, click "User". On the tab page displayed, select the user, and click "Delete".

---

2. (Optional) Input "Owner" in the "User Name" field (the initial password is empty), and then click "login".

Note

If this is the first time you add or delete a user to or from a project or edit a user of a project, you need the user authorization of the group "Owner" (the user Owner is created by default).

Exporting/Importing user information

- To export information of a user, on the "Users and Groups" page, click "User". On the tab page displayed, click "Output/Input", and then select "Output Users and Groups". On the page displayed, select a local path to save the user information, and then click "Save".
- To import information of a user, on the "Users and Groups" page, click "User". On the tab page displayed, click "Output/Input", and then select "Output Users and Groups". On the page displayed, select the user file saved in a local path, and then click "Open".

Adding a group

1. On the "Users and Groups" page, click "Group". The group management page is displayed.
2. Click "Add", fill in the group information, and then click "OK".

Note

- The system provides two groups "Everyone" and "Owner" by default. One group can be a member of another group.
- To edit a group, on the "Users and Groups" page, click "Group". On the tab page displayed, select the group, and click "Edit". In the dialog box displayed, modify the group information, and then click "OK".
- To delete a group, on the "Users and Groups" page, click "Group". On the tab page displayed, select the group, and then click "Delete".

3. (Optional) Input "Owner" in the "User Name" field (the initial password is empty), and then click "login".

Note

If this is the first time you add or delete a group to or from a project or edit a group of a project, you need the user authorization of the group "Owner" (the user Owner is created by default).

Exporting/Importing a group

- To export information of a group, on the "Users and Groups" page, click "Group". On the tab page displayed, click "Output/Input", and then select "Output Users and Groups". On the page displayed, select a local path to save the group information, and then click "Save".
- To import information of a group, on the "Users and Groups" page, click "Group". On the tab page displayed, click "Output/Input", and then select "Output Users and Groups". On the page displayed, select the group file saved in a local path, and then click "Open".

Settings

1. On the "Users and Groups" page, click "Settings". The "Settings" page is displayed.
2. Set relevant items and click "OK".

Parameter	Description	Value
Maximum number of licenses	It specifies the maximum number of attempts that you can log in to a user account by password. If this number is exceeded, this user account will be deactivated.	Set as needed Default: 3
Auto logout after no action	If no operation is made by the mouse or keypad within the time (in minutes) specified here, the user account is automatically logged out.	Set as needed Default: 10

3. (Optional) Input "Owner" in the "User Name" field (the initial password is empty), and then click "login".

Note

If this is the first time you open the project management and settings page, you need the user authorization of the group "Owner" (the user Owner is created by default).

Allocating authorization

1. In the left device tree, right-click the target node (the "Application" node is used in this example), and select "Attribute". The "Attribute" dialog box is displayed.
2. Click "Access Control". On the tab page displayed, select the group to which you want to allocate authorization, click the "+" in the corresponding action column, select the option "Authorize", "Reject", or "Clear", and then click "OK".

Project login/logout

After you add a user and a group and allocate authorization for them, you can log in to the project and use functions of relevant nodes.

You can log in to or out of a project in the following ways:

- Using the menu bar
  - Logging in to the project through a user account

    1. In the menu bar, choose "Project" > "User Management" > "User Logout". The "Login" dialog box is displayed.
    2. Enter the user name and password, and click "OK". The user logs in to the project.

  - Logging out of a project
    In the menu bar, choose "Project" > "User Management" > "User Logout". The user logs out of the project.
- Through the status bar
  - Logging in to the project through a user account

    1. In the status bar, double-click "Current user: xx". The "login" dialog box is displayed.
    2. Enter the user name and password, and click "OK". The user logs in to the project.

  - Logging out of a project
    In the status bar, double-click "Current user: xx" and then click "Logout". The user logs out of the project.

3.10 Adding an Object Through Application

The function "Application" allows you to add an object function. Only the functions listed below are supported.

- CAM table
- DUT
- Program organization unit
- Persistent variable
- Symbol configuration
- Trace
- Interface
- Global variable list
- Application

- POU for implicit checking

In the left device tree, right-click "Application", and then choose "Add Object" > "Cam table" (for example).

CAM Table

For details of the CAM table, see the "CODESYS Programming System > SoftMotion > Object Editor > CAM Editor" section of *Online Help*.

DUT

For details of DUT, see the "CODESYS Programming System > CODESYS Development System > References > User Interfaces > Objects > DUT" section of *Online Help*.

Program Organization Unit

For details of the program organization unit, see the "CODESYS Programming System > CODESYS Development System > References > User Interfaces > Objects > POU" section of *Online Help*.

Persistent Variable

For details about the persistent variable, see *" Persistent Variable" on page 369*.

Symbol Configuration

For details of symbol configuration, see "CODESYS Programming System > CODESYS Development System > References > User Interfaces > Objects > Symbol Configuration" section of *Online Help*.

Trace

For details of the trace, see the "CODESYS Programming System > CODESYS Development System > References > User Interfaces > Objects > Trace" section of *Online Help*.

Interface

For details of the interface, see the "CODESYS Programming System > CODESYS Development System > References > User Interfaces > Objects > POU > Interface" section of *Online Help*.

Global Variable List

For details of the global variable list, see the "CODESYS Programming System > CODESYS Development System > References > User Interfaces > Objects > GVL - Global Variable List" section of *Online Help*.

Application

For details of the application, see the "CODESYS Programming System > CODESYS Development System > References > User Interfaces > Objects > Application" section of *Online Help*.

POU for Implicit Checking

For details of the POU for implicit checking, see the "CODESYS Programming System > CODESYS Development System > References > User Interfaces > Objects > POU for Implicit Checking" section of *Online Help*.

4 Network Configuration

4.1 Device Configuration

4.1.1 Device Configuration

Device configuration is the first step of PLC programming. It involves two functions: network configuration and hardware configuration. You can use the two functions to deploy the device.

- Network configuration
  It is designed from the perspective of bus-type network topology, and is the entrance of device configuration.
- Hardware configuration
  It is used to add the expansion I/O modules of medium-sized PLC.

4.1.2 Network Configuration

After creating an InoProShop project, double-click the "Network Configuration" node in the left device tree, as shown in the following figure.

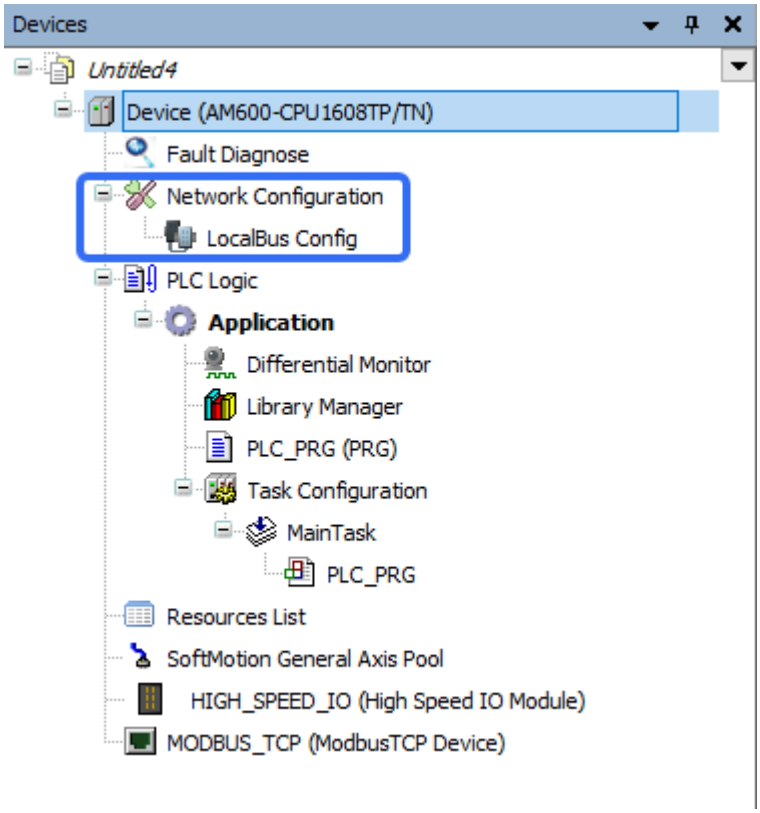


Figure 4-1 Network Configuration node

Double-click this node to open the "Network Configuration" page and "Network Devices List" (as shown in *" Configuring a PLC as a Master or a Slave" on page 67*). The "Network Configuration" page

displays the PLC currently used by the user program, and the "Network Devices List" displays all the devices supported by the PLC.

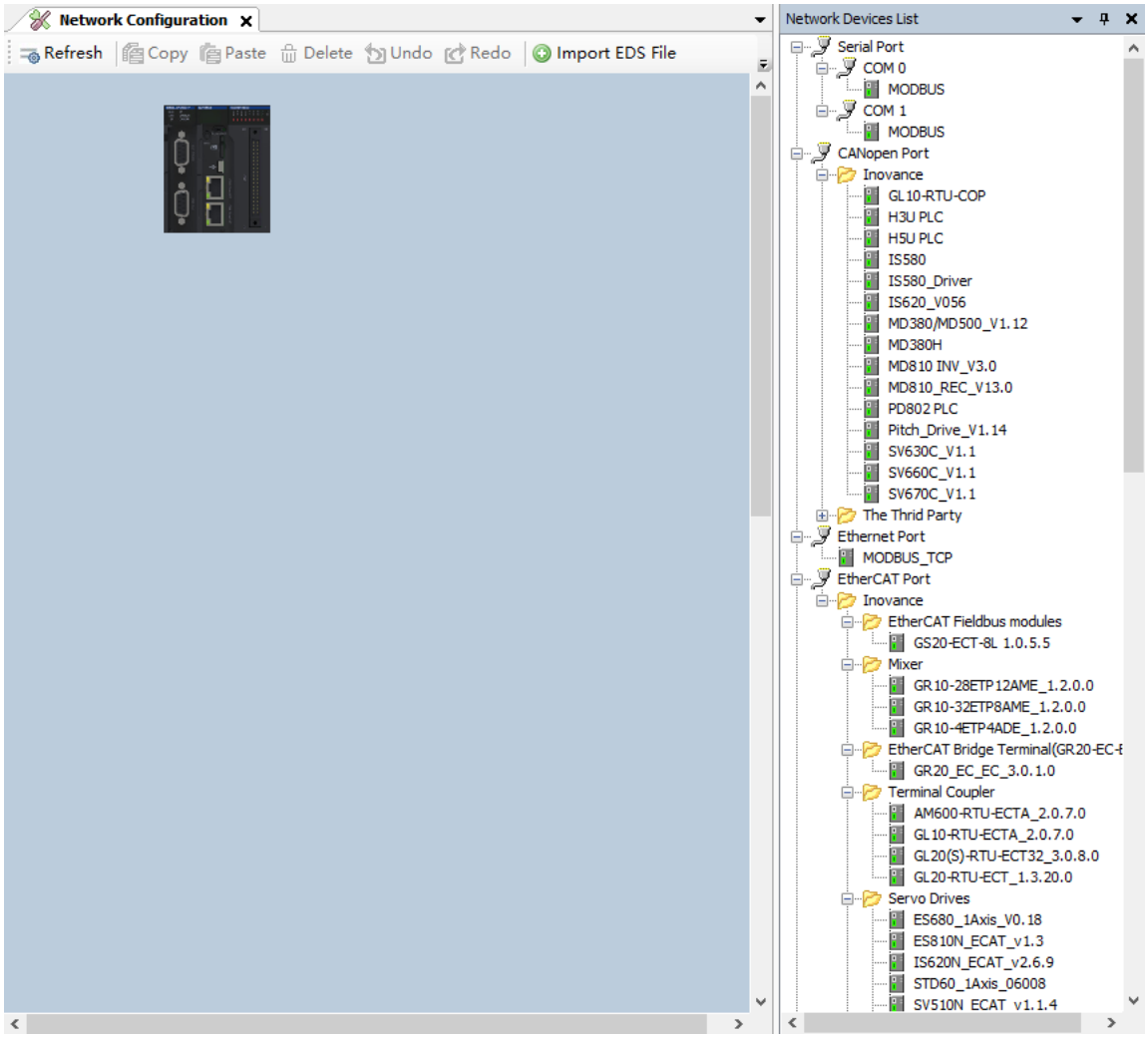
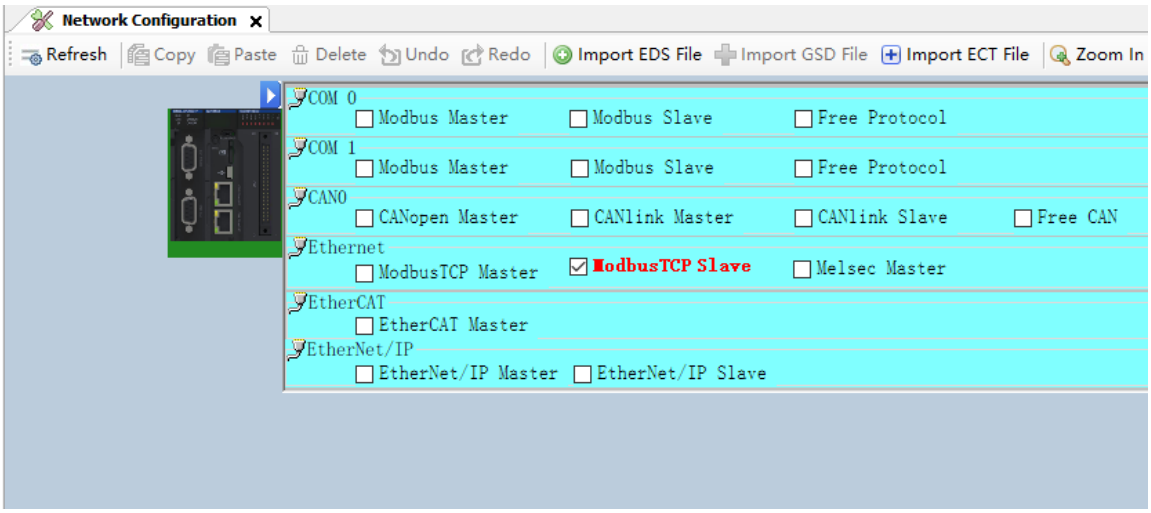


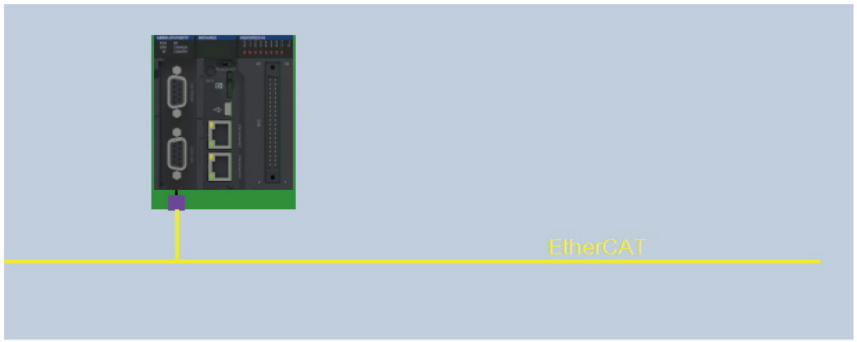
Figure 4-2 "Network Configuration" page

Configuring a PLC as a Master or a Slave

When you click a PLC on the "Network Configuration" page, the masters/slaves supported by the PLC are displayed, as shown in the following figure. Click the target checkbox to enable the master/slave supported by the CPU.

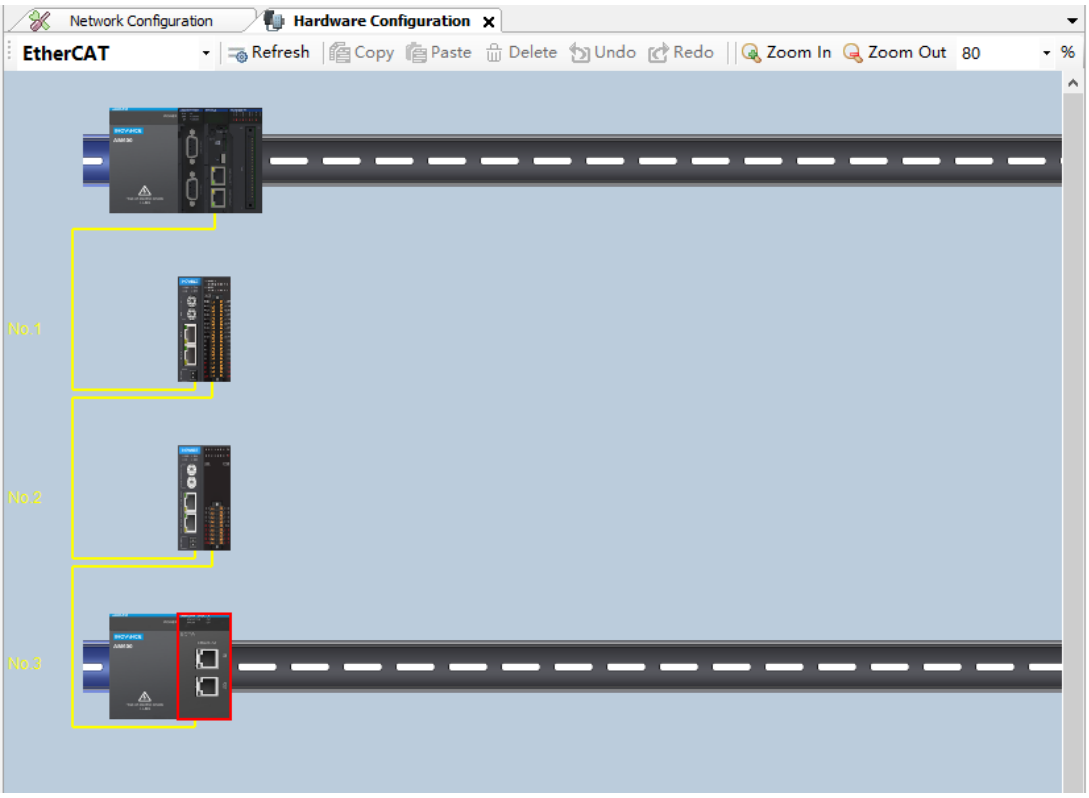


When a master (except the CANlink master) function is enabled for the CPU, the bus-type topology is displayed. For example, the following figure shows the enabled EtherCAT master.



- Add a slave
  After a master in the CPU is enabled, you can add the slave under the corresponding bus. The slave can be added by three methods (taking the EtherCAT bus for example).

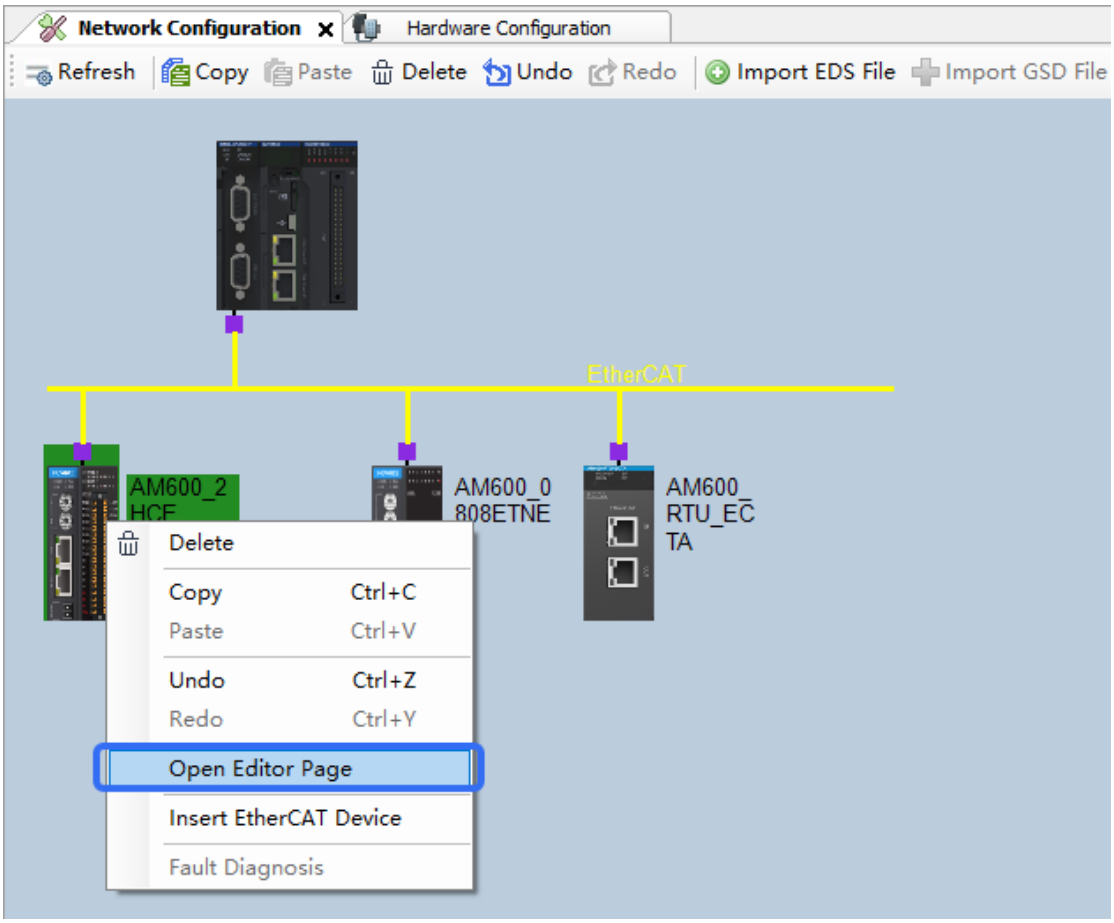
  1. Enable the EtherCAT master function, and then select a slave node under the EtherCAT port node in the network device list. Drag the node to the network configuration page.
  2. Enable the EtherCAT master function, and double-click a slave node under the EtherCAT port node in the network device list as shown in *" Configuring a PLC as a Master or a Slave" on page 67*.
  3. Double-click a slave node under the EtherCAT port node in the network device list. If you use this method, the internal master function of the CPU will be enabled first.
  4. For details about how to add an EtherCAT splitter, see *" Adding a Splitter and Adding Slaves to the Splitter" on page 185*.To add I/O modules for an added slave (slave for the AM600), double-click the device to open the "Hardware Configuration" page.



- View basic device information.
  Click a device on the "Network Configuration" page, and you can see the basic device information in "Config Device Information Output" > "DeviceDefaultInfoList".

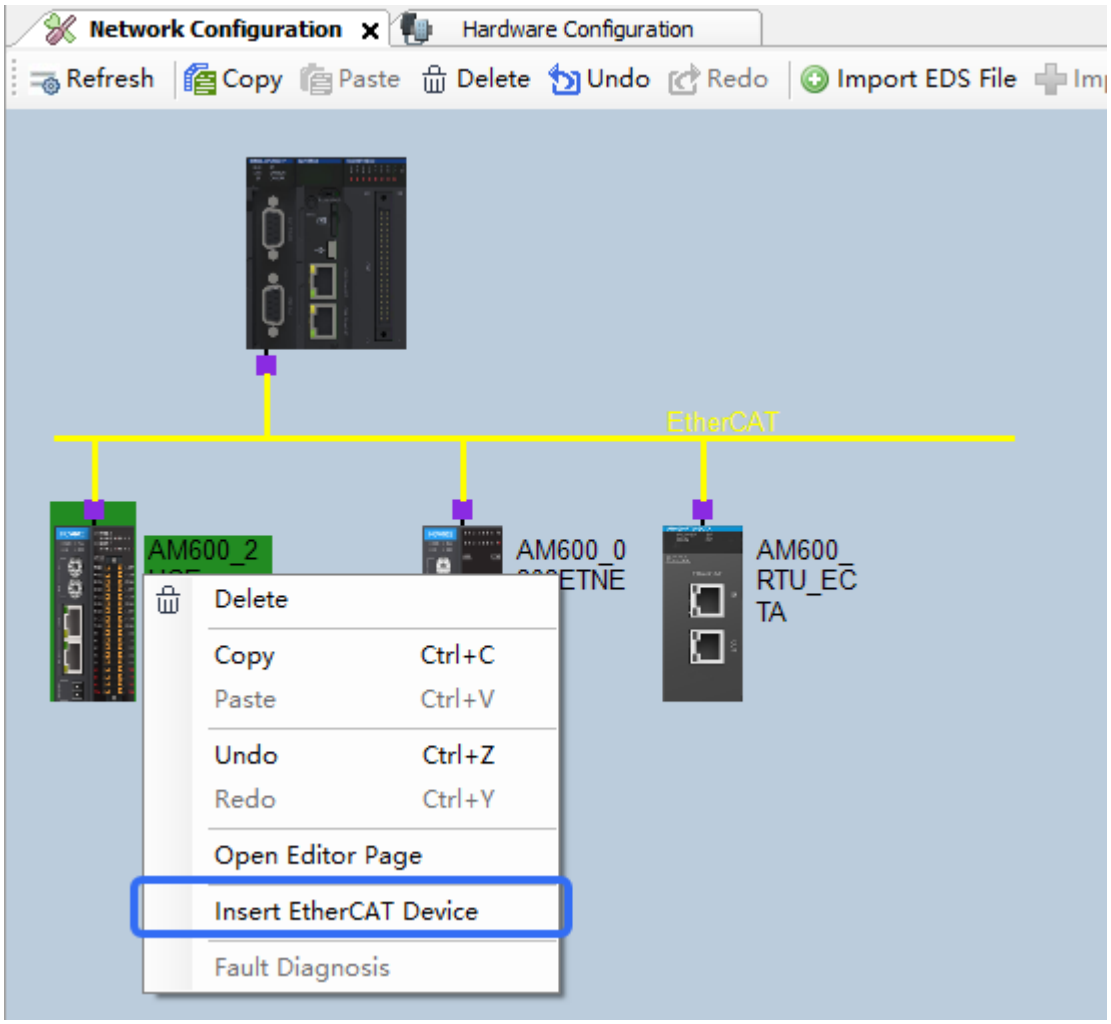


- Open the device configuration page
  Right-click the EtherCAT slave in network configuration, and select "Open Editor Page" from the shortcut menu, as shown in the following figure.



- Insert the EtherCAT slave

  Right-click the EtherCAT slave in network configuration, and select "Insert EtherCAT Device" from the shortcut menu, as shown in the following figure.



- Configuration devices can be copied, deleted, and added. For details, see the section "Configuration Device Common Operations".
- Edit configuration
  If the network configuration includes repeated Modbus slave addresses or ModubsTCP slave IP addresses, the repeat information is displayed in the output box during project compiling. For details, see the section "Configuration Compiling Error Locating".

Device Information List

To open the device information list, choose "View" > "Configuration Device Info View". The configuration device basic information is displayed, including the slot number, device name, and description. The information is minimized at the bottom of the page by default. You need to click the list to open it.

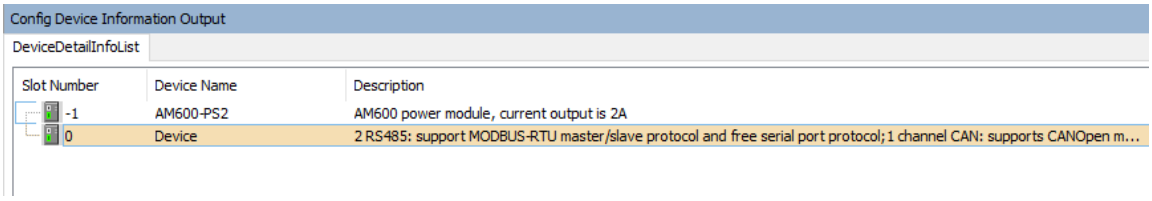


Figure 4-3 Device Information List

- Slot number
  The slot numbers match the slot numbers in "Hardware Configuration". The numbers of the slots on the main rack and on the communication slave both start from 1. Slot –1 matches the AM600 power module, and slot 0 matches the CPU.
- Device name
  The device names are the same as the device names in the left device view.
- Description
  The basic description of devices, including operating indicators and functions.

To locate a device on the configuration page, click a row in the device list. To open the configuration page of a device, double-click a row.

Configuration Device Common Operations

The common operations of configuration devices include copying, pasting, canceling, restoring, deleting, importing EDS, GSD, and ECT files, zooming in, and zooming out.

Note

- The copying, pasting, deleting, canceling, and restoring operations only apply to I/O modules on the "Hardware Configuration" page and to slaves on the "Network Configuration" page.
- If you perform copying, pasting, or deleting operation on the slave on the "Network Configuration" page, the same operation is also performed on its modules.

- Import EDS files: The network device list contains some CANopen devices by default. If you need to add other CANopen devices or EtherNet/IP devices, import the corresponding standard EDS file. After the file is imported, the device is added to the network device list. If the imported device is from Inovance, it will be displayed under the Inovance node; otherwise, it is displayed under the third-party vendor node.
- Import GSD files: The network device list contains some DP devices by default. If you need to add other DP devices, import the corresponding standard GSD file. After the file is imported, the device is added to the DP port node in the network device list. If the imported device is from Inovance, it will be displayed under the Inovance node; otherwise, it is displayed under the third-party vendor node.
- Import ECT files: The network device list contains some EtherCAT devices by default. If you need to add other EtherCAT devices, import the corresponding standard EtherCAT xml (*.xml) file. After the file is imported, the device is added to the EtherCAT port node in the network device list. If the imported device is from Inovance, it will be displayed under the Inovance node; otherwise, it is displayed under the third-party vendor node.

4.1.3 Hardware Configuration

The hardware configuration uses the rack and slot used in device configuration to simulate the field device modular configuration. Hardware configuration applies to the I/O modules of medium-sized PLCs.

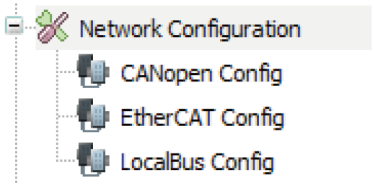
From the aspect of configuration procedure, to add a remote I/O module, you need to configure the communication module in network configuration first, and then configure the I/O module in hardware configuration. To add a local I/O module, directly open the "Hardware Configuration" page to perform operations. Hardware configuration supports multi-bus I/O configuration, depending on the used CPU model.

Accessing the Hardware Configuration Page

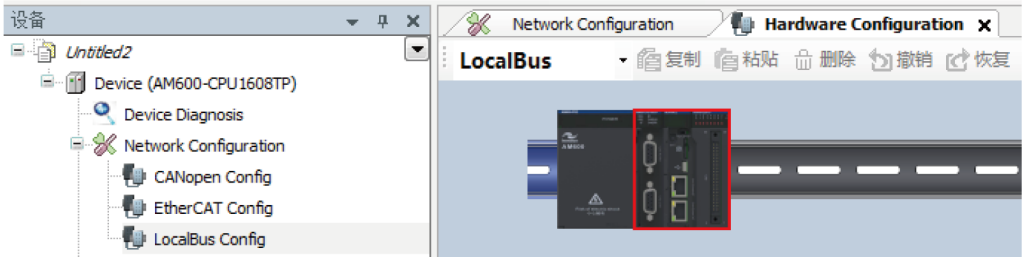
Except Modbus and Modbus TCP devices, other bus-type devices should have matching "Hardware Configuration" pages.

You can access the "Hardware Configuration" page in two ways:

1. Double-click a device on the "Network Configuration" page.
2. Double-click a bus node under the "Network Configuration" node in the left device tree, as shown in the following figure.



By default, the "LocalBus Config", namely, local bus configuration node, is available. Double-click it to perform the local module configuration.

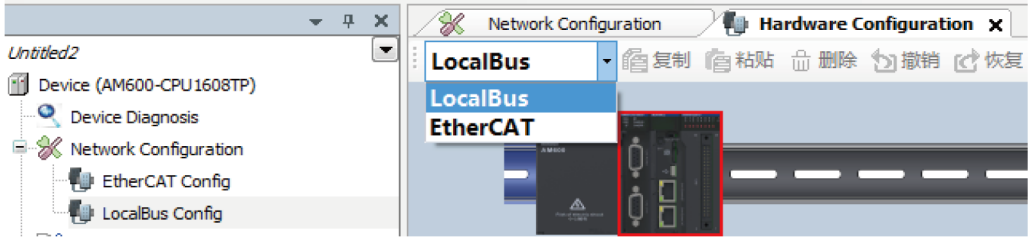


In addition, the "Input/Output Module List" is displayed on the right.

Switching Bus

You can switch the bus of hardware configuration in two ways:

- Double-click a bus node under the "Network Configuration" node in the left device tree.
- Select another bus type on the current "Hardware Configuration" page, as shown in the following figure.



Adding a Module

You can add I/O modules in three ways:

1. Double-click an idle slot on the rack. In the module list displayed, double-click a module to add it.
2. Select a node from the right module list, and drag it to an idle slot.
3. Select a rack (blue part in the figure) or device, and double-click a device in the right module list. Then you can add the devices to the idle slots in order. If you click an idle slot, the selected device is added to this slot.

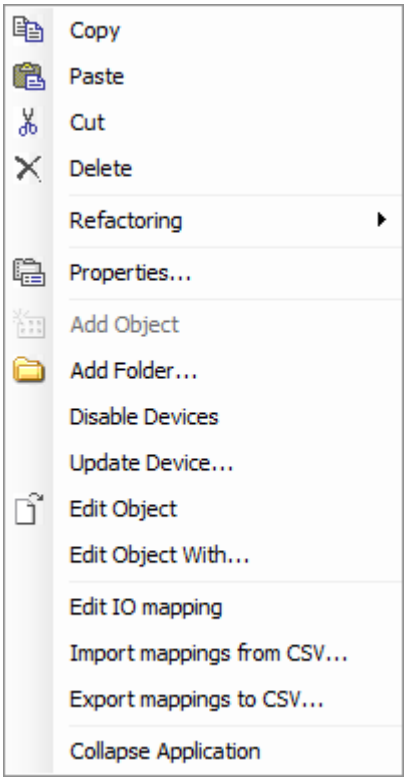


Dragging a Module

Select a module and drag it to the target slot. By using the dragging function, you can exchange the locations of two modules or move a module to an idle slot. However, the modules in the main rack and expansion rack cannot be interchanged.

4.1.4 Device Tree Operations

After adding a bus device, you can select a device in the device tree and use the shortcut menu or shortcut keys to copy and paste, delete, cut, or drag the device. The following figures show the menu options.



These function operations match the basic standard operations.

Note

• The copy-paste and cut-paste functions only apply to local masters, local slaves, and individual axis devices.

4.1.5 Configuration Compiling Error Locating

The configuration device defines configuration rules and error detection mechanism, for example, repeated station addresses of Modbus devices or IP addresses of TCP devices in network configuration. If the slave in the expansion rack of hardware configuration is not connected to the I/O module, a configuration compiling error will be reported.

If a configuration error occurs during project compiling, the InoProShop message box will display the error. Double-click the error list to go to the corresponding configuration page and the red rectangular box blinks three times.

4.2 CPU Configuration

4.2.1 Overview

The CPU module is the main module of a medium-sized PLC. The CPU is configured based on the control system requirements of the PLC hardware, to complete the configurations of the PLC and its control system. Medium-sized PLCs support EtherCAT bus, PROFIBUS DP, Modbus RTU, CAN bus, and Modbus TCP, and also support high-speed I/O modules. Therefore, to complete all CPU configurations, you need to set the bus parameters according to the PLC hardware network configuration.

For example, the CPU module of the AM600 has built-in high-speed I/O modules, and local I/O modules can be configured. In addition, on the CPU configuration page, you can set the CPU system parameters and CPU firmware upgrade.

4.2.2 General CPU Configuration Procedure

1. Design the entire CPU hardware network structure.
2. Activate the corresponding bus in network configuration and add the slaves corresponding to the bus. Currently, the CPU supports EtherCAT bus, DP bus, CANopen, CANlink, Modbus RTU, and Modbus TCP.
3. For an EtherCAT AM600 slave, CANopen AM600 slave, or DP AM600 slave, add I/ O modules to hardware configuration.
4. Configure the master, slave, and module configuration parameters corresponding to the bus.

By default, the AM600 CPU module has the high-speed I/O function. Each CPU can be configured with up to 16 local I/O modules. In addition, you need to configure the CPU system parameters, PLC I/O update, PLC bus task, PLC user management, log, upgrade, and tasks according to the actual requirements.

To know the parameter settings of buses and their slaves, see the chapters of the buses. For the built-in functions of CodeSys, such as PLC I/O update, PLC bus task, PLC user management, log, and tasks, see the CodeSys software help. In this guide, the CPU configuration mainly involves the functions of medium-sized PLCs: CPU parameter configuration, I/O module configuration, and high-speed I/O configuration.

4.2.3 CPU Parameter Configuration

System Settings

System settings include the configurations of downtime caused by CPU fault, location retaining at power failure, network address, and system time, as shown in the following figure.

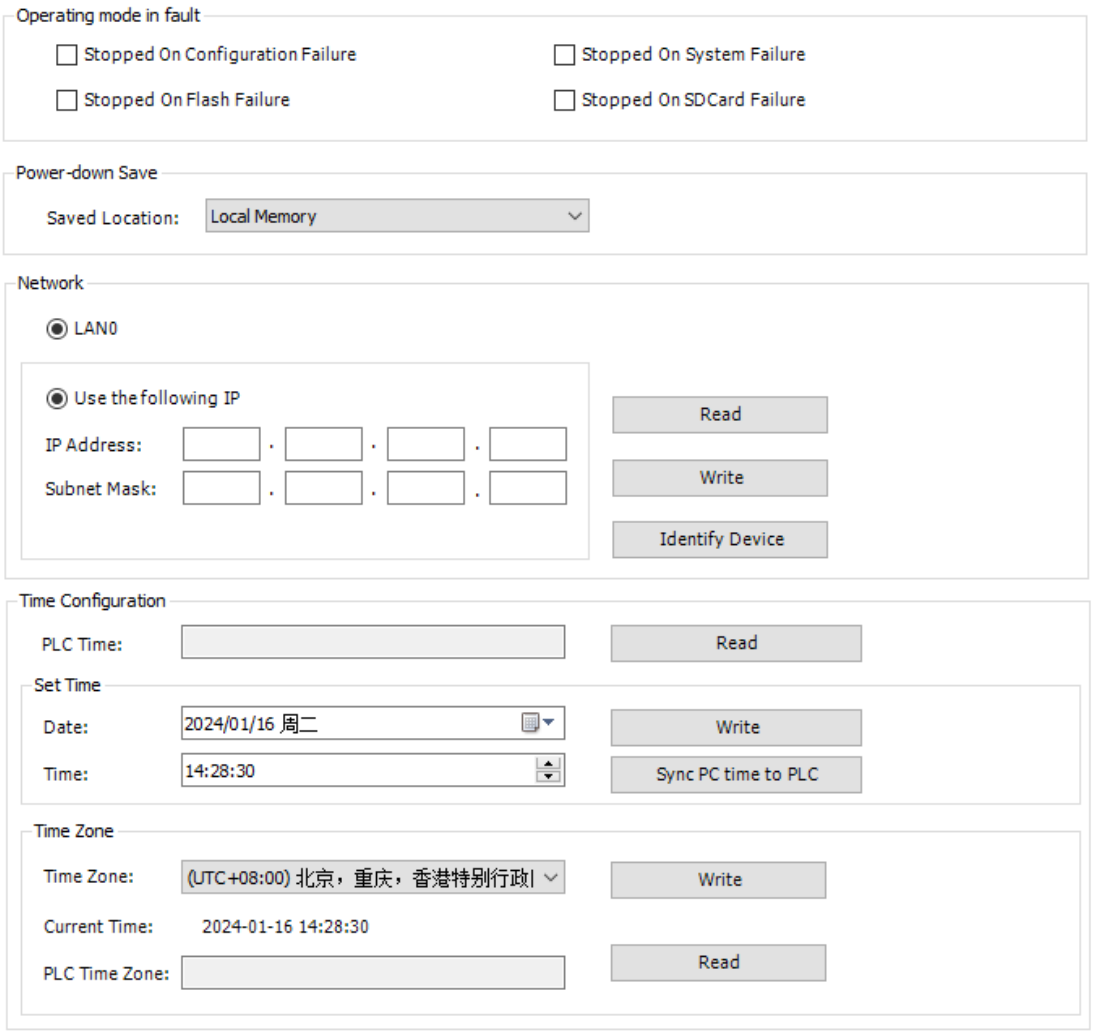


Figure 4-4 System settings dialog box

Operating mode in fault

- Stopped On Configuration Failure: Whether the CPU stops running when configurations are inconsistent, for example, when the configured I/O module mounted to the CPU does not match the physically connected I/O module.

- Stopped On System Failure: Whether the CPU stops running when a system error occurs, for example, when an interrupt error or stack overflow occurs.
- Stopped on Flash Failure: Whether the CPU stops running when a Flash error occurs. This function is unavailable currently.
- Stopped on SD Card Failure: Whether the CPU stops running when an SD card error occurs, for example, when the SD card memory is used up or the SD card is lost. This function is unavailable currently.

Power-down Save

- Saved Location: Sets the data saving location upon power failure, including the local memory and SD card.

Note

When you set the saved location as SD card, ensure that an SD card is available; otherwise, data will be lost upon power failure.

Network

- LAN0: Indicates the network interface name used by the PLC for Ethernet communication. The AM600 and AM400 series have only one EtherNET interface, and the AC800 series has two EtherNET interfaces. The network names vary with PLCs. You can configure differentiated network information according to network interface configurations.
- Use the following IP: The IP address of the PLC can be manually modified or automatically obtained. This configuration is used to manually modify the PLC network information.
- Obtain IP address automatically: The IP address of the PLC is assigned by a router or switch. Note: This function is available only for the AC800.
- IP Address: Indicates the IP address of the PLC.
- Subnet Mask: Indicates the subnet mask of the PLC.
- Gateway: Sets the gateway for the PLC. Note: This function is available only for the AC800.
- Read: Reads the IP address and subnet mask of the PLC, which are displayed in the IP address and subnet mask edit boxes.
- Write: Writes the IP address and subnet mask in the edit boxes into the PLC. If you have logged in and connected to the network, you will be logged out. In this case, you need to re-log in to the network. If the USB connection is used, you do not need to re-connect the device.

Note

- Do not set IP addresses in the same network segment for the two network interfaces of the AI800-series and AC800-series PLCs; otherwise, connection will be affected.
- When the IP address needs to be read and written, select the PLC to be read and written on the "Communication Settings" tab. In addition, the IP address and subnet mask to be written must comply with the related standards.

- Identify Device: Identifies the PLC to be connected. After you configure PLC scanning on the "Communication Settings" tab, multiple PLCs may be detected. After selecting one PLC, click this button. Then the two digits of the LED on the PLC panel will alternately display 0, as shown in the following figure.

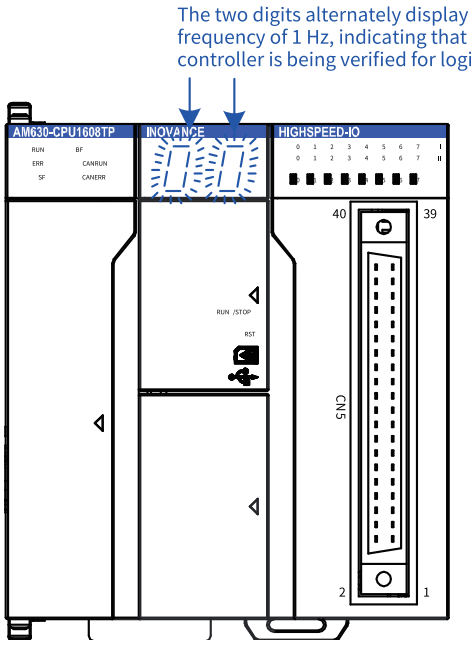


Figure 4-5 PLC LED in identifying state

Moreover, the software tool InoProShop displays the dialog box as shown in the following figure. Close the dialog box to complete the identification, and then the LED restores the default state.

RTC Configuration

- PLC Time: Displays the current PLC time.
- Read: Reads the PLC time.
- Write: Writes the current date and time set on the PLC. The current date and time are displayed in the left edit boxes.
- Sync To Local Date/Time: Writes the current date and time from the PC to the PLC.

Note

When the system writes the PLC time or synchronizes the local date/time to the PLC, the PLC may be affected, for example, the bus synchronization may be affected. Therefore, before writing the PLC time, ensure that the PLC is in the "Stop" state. It is recommended to hot reset the PLC after the time is written.

Time Zone

- Time Zone: Reads the PLC time zone. For example, the time zone of Beijing is UTC+8.
- Read: Reads the time zone of the PLC.
- Write: Writes the time zone selected for the PLC.

Upgrade

The upgrade page is used for the upgrade of PLC firmware, as shown in *"Figure 4–6 Upgrade dialog box" on page 80*. The PLC firmware upgrade package provides the software data for upgrade, which may include UBOOT, Device Tree, kernel, and system program. Generally, the upgrade package includes only the system program.

Note

The upgrade function is unavailable. Use the InoProShop tool to perform an upgrade.

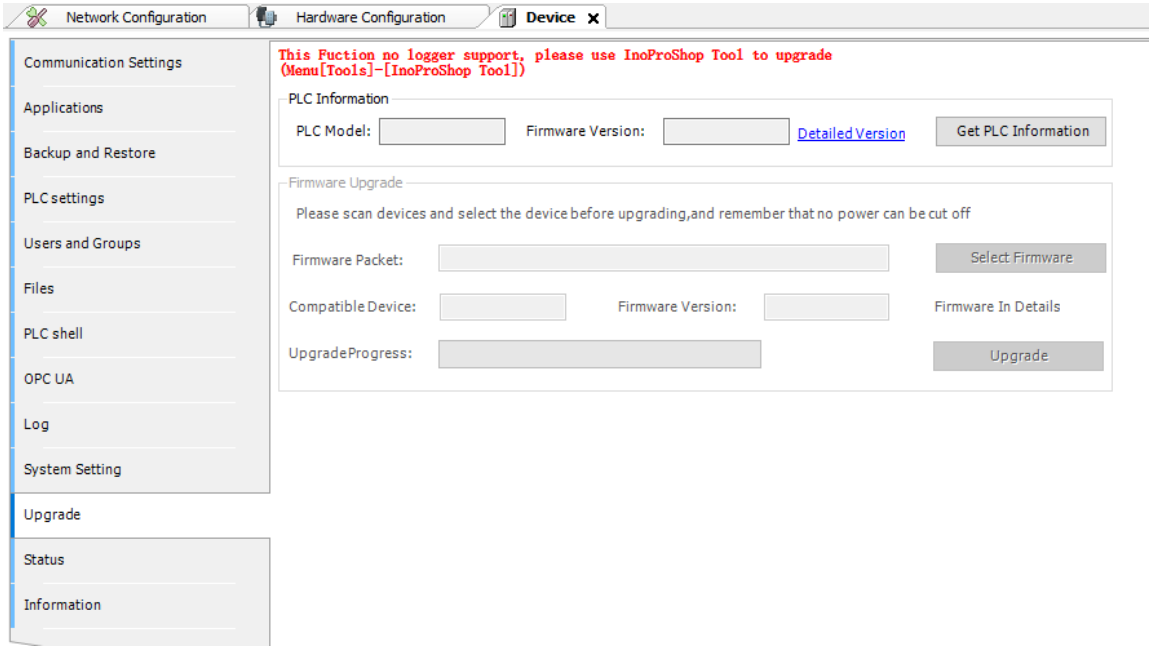


Figure 4-6 Upgrade dialog box

PLC Information

- PLC Model: The model of the current PLC, such as AM600 or AM610.
- Firmware Version: The firmware version of the PLC, for example, 1.2.3.0.
- Detailed Version: The detailed version information about the PLC, which may include UBOOT, Device Tree, kernel, and system program, as shown in Figure 4-8. If the PLC firmware has not been upgraded, the version information may not be included.
- Get PLC Information: Gets the PLC model and firmware information. If the PLC firmware has not been upgraded, the PLC information may not be obtained.

Firmware Upgrade

- Firmware packet: Sets the firmware upgrade package, with the file name expansion ".upgrade".
- Compatible Device: Displays the devices compatible with the firmware upgrade package. The upgrade can be performed only when the device is compatible with the PLC model.
- Firmware Version: Displays the firmware version of the upgrade package.
- Firmware In Details: Gets detailed information about the firmware upgrade package.
- Upgrade: After this button is clicked, the firmware upgrade starts. Before performing the upgrade, the system checks the device type and upgrade firmware file version. If the upgrade firmware version is later than the PLC firmware version, the upgrade is performed. If the versions are the same, the upgrade is not performed. If the upgrade firmware version is earlier than the PLC firmware version, the upgrade is performed only after confirmation.

Note

- Before the upgrade, scan the device to be upgraded on the "Communication Settings" page and select the PLC to be upgraded.
- Do not power off the device during the upgrade; otherwise, unrecoverable system faults may occur.
- The upgrade will last about two minutes. After the upgrade, the device automatically restarts.
- After restart (upgrade completed), the LED displays 00 or dynamically changing digits.
- After the upgrade, the PLC device name may be changed. Scan the device again.

After the upgrade, the system displays and verifies that the PLC information and detailed version information are consistent with the firmware version and detailed information.

Information

The PLC basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Order Number, Description, and Image, as shown in the following figure. After login, the "Information" page displays the PCB software version of the CPU and logic software version. The PCB software version is the CPU system program version, and the logic software version is the FPGA software version within the CPU.

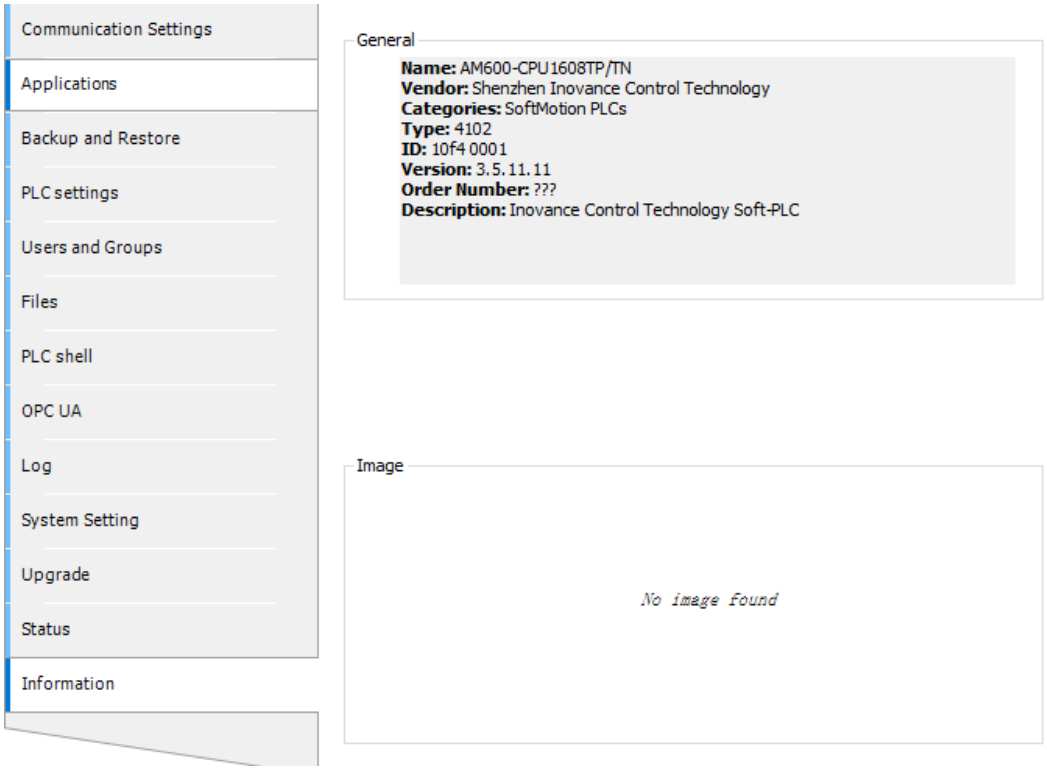


Figure 4-7 CPU information page

4.2.4 I/O Module Configuration

Overview

The following table lists the I/O modules of the GL10 series and GL20 series.

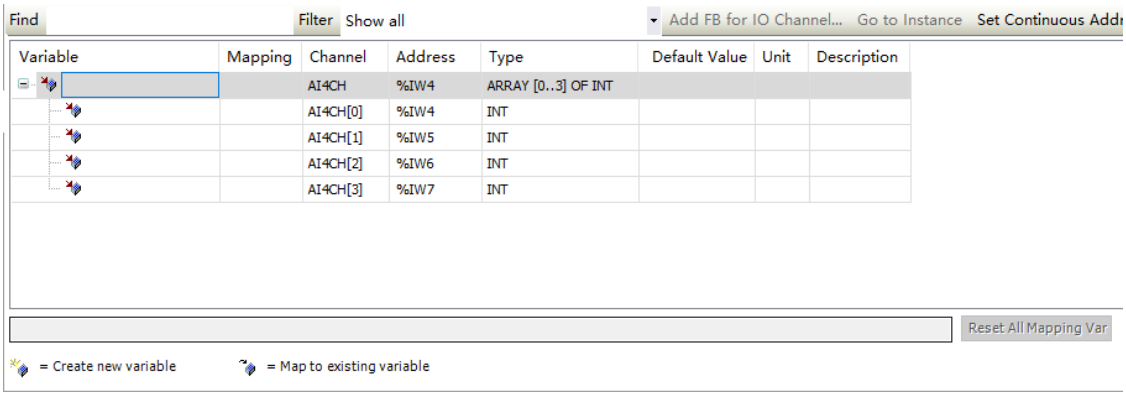
Series	Module Type	
GL10 series	Digital input (DI)	
	Digital output (DO)	Relay output (ER type)
		NPN output (ETN type)
		PNP output (ETP type)
	Analog input (AD)	
	Analog output (DA)	
	Temperature detection module	4TC (4-channel temperature detection module, supporting thermocouple)
		8TC (8-channel temperature detection module, supporting thermocouple)
		4PT (4-channel temperature detection module, supporting resistance temperature detector)
GL20 series	Digital Input (DI)	
	Digital Output (DO)	Relay output (ER type)
		NPN output (ETN type)
		PNP output (ETP type)
	Digital input/output (DI/DO)	
	Analog input (AD)	
	Analog output (DA)	
	Temperature detection module	4PT (4-channel temperature detection module, supporting resistance temperature detector)
		4TC (4-channel temperature detection module, supporting thermocouple)
	Communication module	2CAN (2-channel CAN communication module)
		2S485 (2-channel RS485 expansion module)
		2SCOM (2-channel serial port module)
		1DNM (1-channel DeviceNet main module)
	Process module	2SSI (2-channel SSI communication)

DI Modules of the GL10 Series

There is no module parameter configuration for DI modules. Only the "I/O Mapping", "Status", and "Information" pages are available. You only need to map the I/O variables on the "I/O Mapping" page to obtain the DI values. The 16-channel DI module is used as an example here.

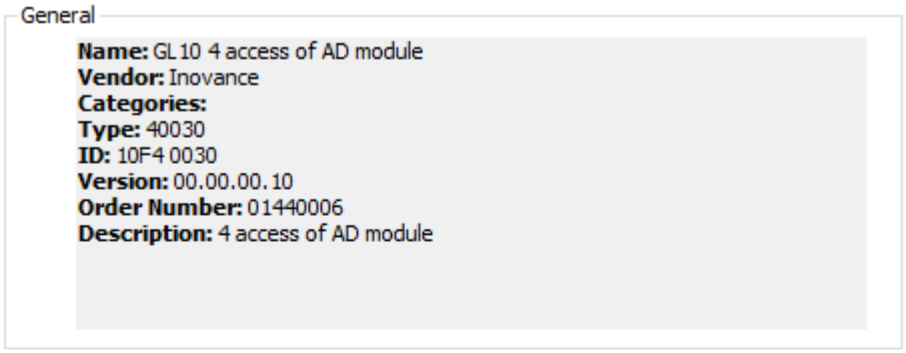
1. DI16 I/O mapping

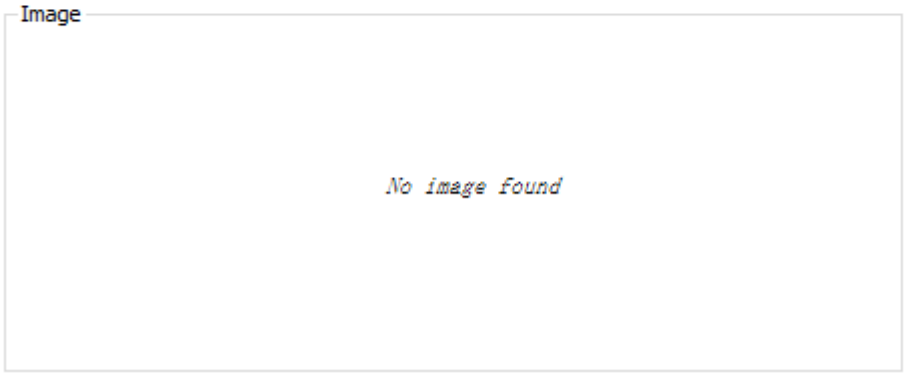
   DI16 is a 16-bit digital input module. As shown in the following figure, on the "I/O Mapping" page, each bit or every eight bits can be mapped to one variable to obtain the input value. For details, click the "I/O Mapping" link.



2. Information

The DI16 module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Order Number, Description, and Image, as shown in the following figure. After login, the "Information" page displays the DI module logic software version, which is the FPGA software version within the DI module.





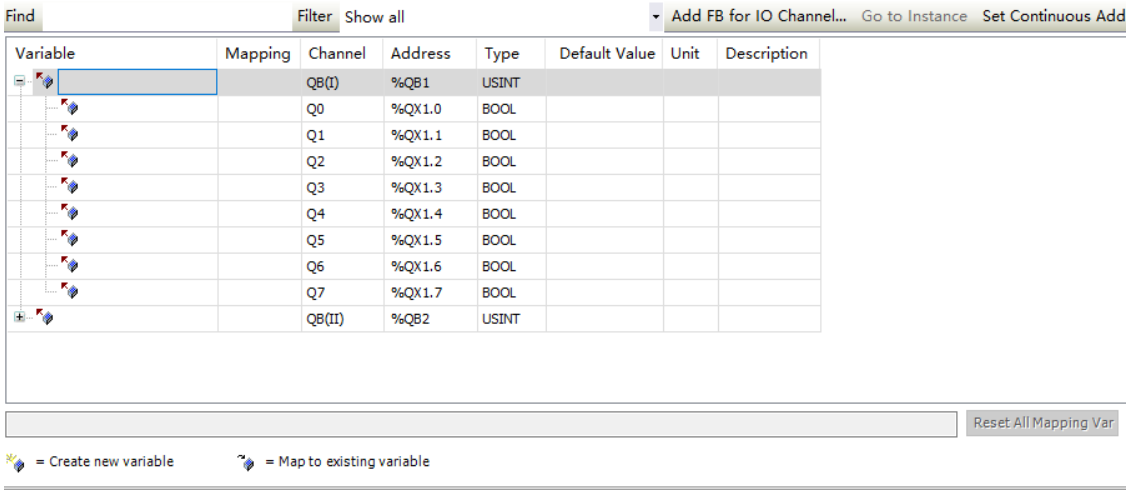
DO Modules of the GL10 Series

There is no module parameter configuration for DO modules. Only the "I/O Mapping", "Status", and "Information" pages are available. You only need to map the I/O variables on the "I/O Mapping" page and output the mapped variable values to the DO module. DO modules include 16-channel modules

and 32-channel modules. They have similar "I/O Mapping" pages. The 16-channel DO module is used as an example here.

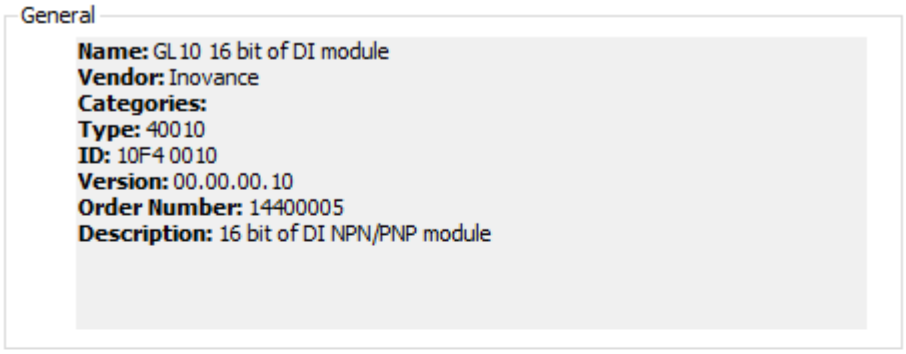
1. DO16 I/O mapping

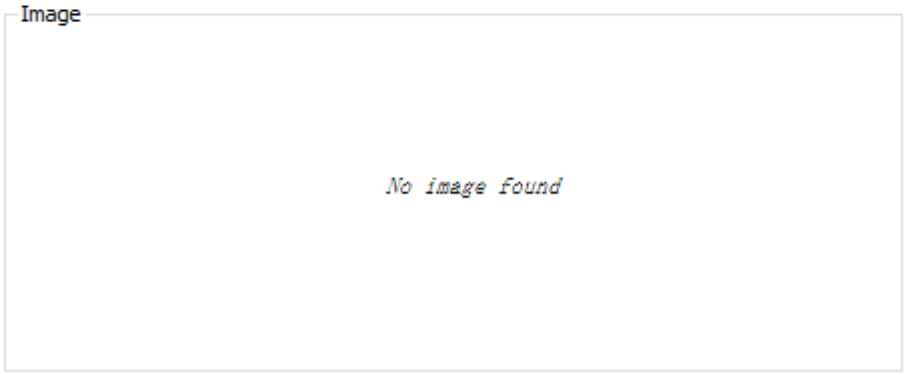
   DO16 is a 16-bit digital output module. As shown in the following figure, on the "I/O Mapping" page, each bit or every eight bits can be mapped to one variable to obtain the output value. For details, click the "I/O Mapping" link.



2. Information

   The DO16 module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Model Number, Description, Order Number, and Image, as shown in the following figure. After login, the "Information" page displays the DO module logic software version, which is the FPGA software version within the DO module.

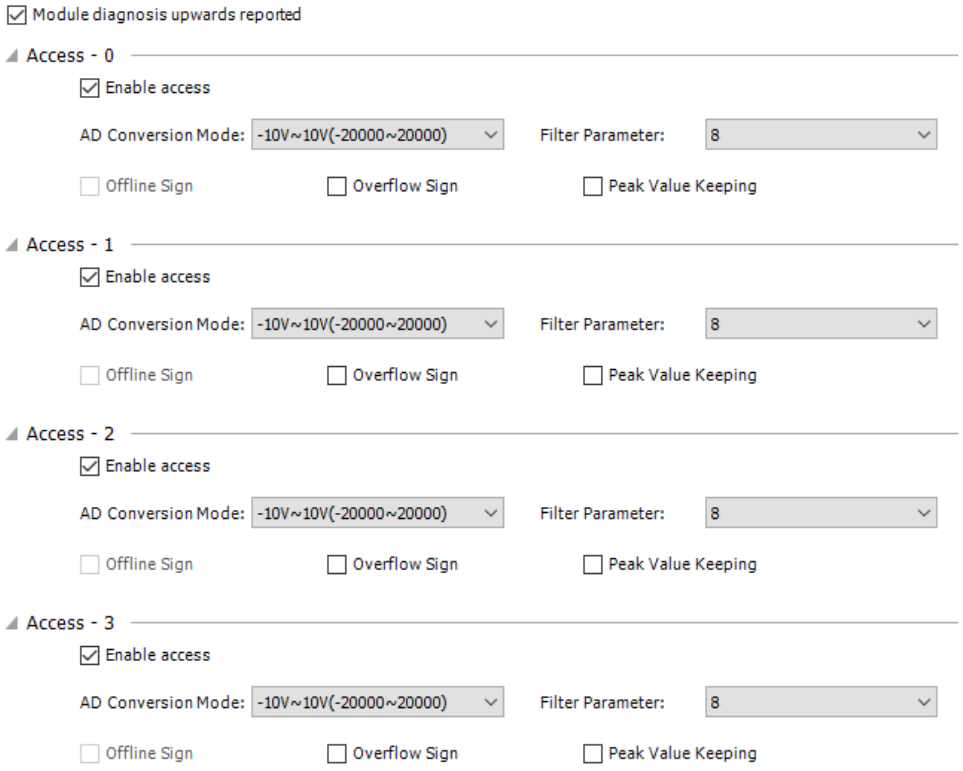




AI Modules of the GL10 Series

1. General settings

   An AI module has four channels, each of which has independent parameter settings and I/O mapping register (16-bit) settings. The following is the description of one channel in each module.



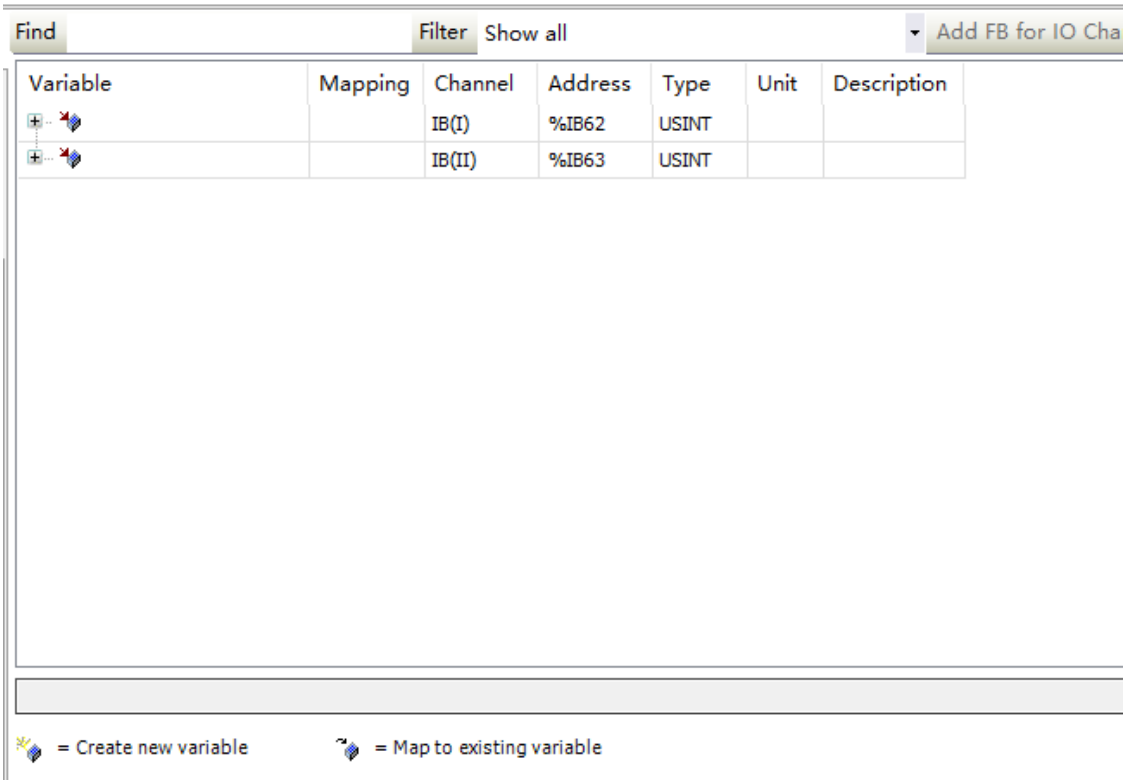
- Module diagnosis upwards reported: Specifies whether to report the module faults to the parent device (such as the CPU and remote module slave). If this function is checked and the parent device is configured with "Stopped On Failure", the parent device will stop the running of this device.
- Enable access: Specifies whether to activate the channel. The channel is available only after it is activated.
- Channel diagnosis upwards reported: Specifies whether to report the channel faults to the parent device (such as the CPU and remote module slave). If this function is checked and the parent device is configured with "Stopped On Failure", the parent device will stop the running of this device.
- AD Conversion Mode: Specifies the conversion mode of the analog input. This setting specifies the channel input conversion type, conversion value range, and the mappings between conversion types and digital values. The following table lists mappings between analog values of analog input and digital values.

-	Rated Input Range	Rated Digital Value	Input Limit Range	Digital Value Limit
Analog voltage input	–10 V to +10 V	–20000 to +20000	–11 V to +11 V	–22000 to +22000
	0 V to 10 V	0 to 20000	–0.5 V to +10.5 V	–1000 to +21000
	–5 V to +5 V	–20000 to +20000	–5.5 V to +5.5 V	–22000 to +22000
	0 V to 5 V	0 to 20000	–0.25 V to +5.25 V	–1000 to +21000
	1 V to 5 V	0 to 20000	0.8 V to 5.2 V	–1000 to +21000
Analog current input	–20 mA to +20 mA	–20000 to +20000	–22 mA to +22 mA	–22000 to +22000
	0 mA to 20 mA	0 to 20000	–1 mA to +21 mA	–1000 to +21000
	4 mA to 20 mA	0 to 20000	3.2 mA to 20.8 mA	–1000 to +21000

- Filter Parameter: Indicates the filter time of the analog input channel, ranging from 1 ms to 255 ms.
- Offline Sign: Specifies whether to detect the offline state of the AI channel. The system cannot distinguish the input value 0 and offline state of the AI module, so all values in the conversion mode range, including 0, cannot activate the offline sign.
- Overflow Sign: Specifies whether to detect overflow of the AI channel.
- Peak Value Keeping: Specified whether to keep the peak value input of the AI channel.

2. AI4 I/O mapping

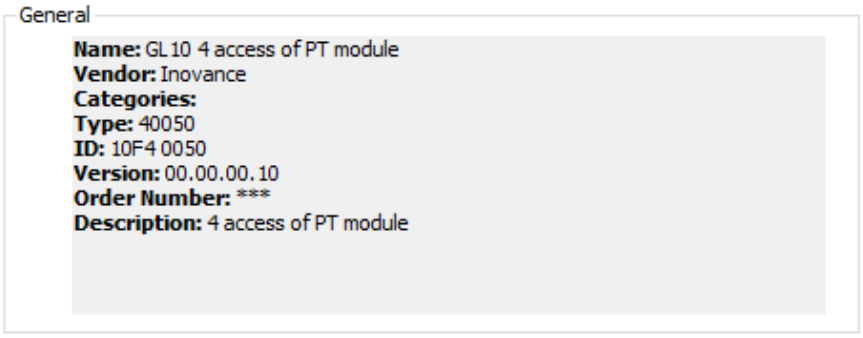
AI4 is 4-channel analog input. Each channel matches a 16-digit integer. For the mappings between analog values and digital values, see general settings of analog input. On this page, each 16-digit integer can be mapped to a variable to obtain the digital value matching an analog value of the input channel. For details, click "I/O Mapping".

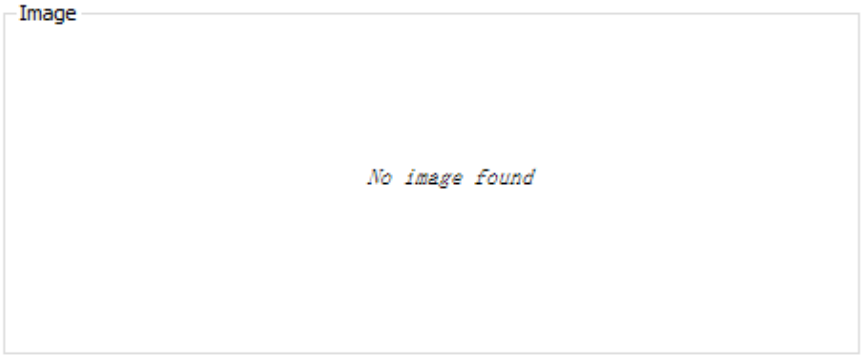


3. Information

The AI4 module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Model Number, Description, Order Number, and Image.

After login, the "Information" page displays the PCB software version and logic software version of the AI4 module. The PCB software version is the embedded software version of AI4 and the logic software version is the FPGA software version within the AI4 module.

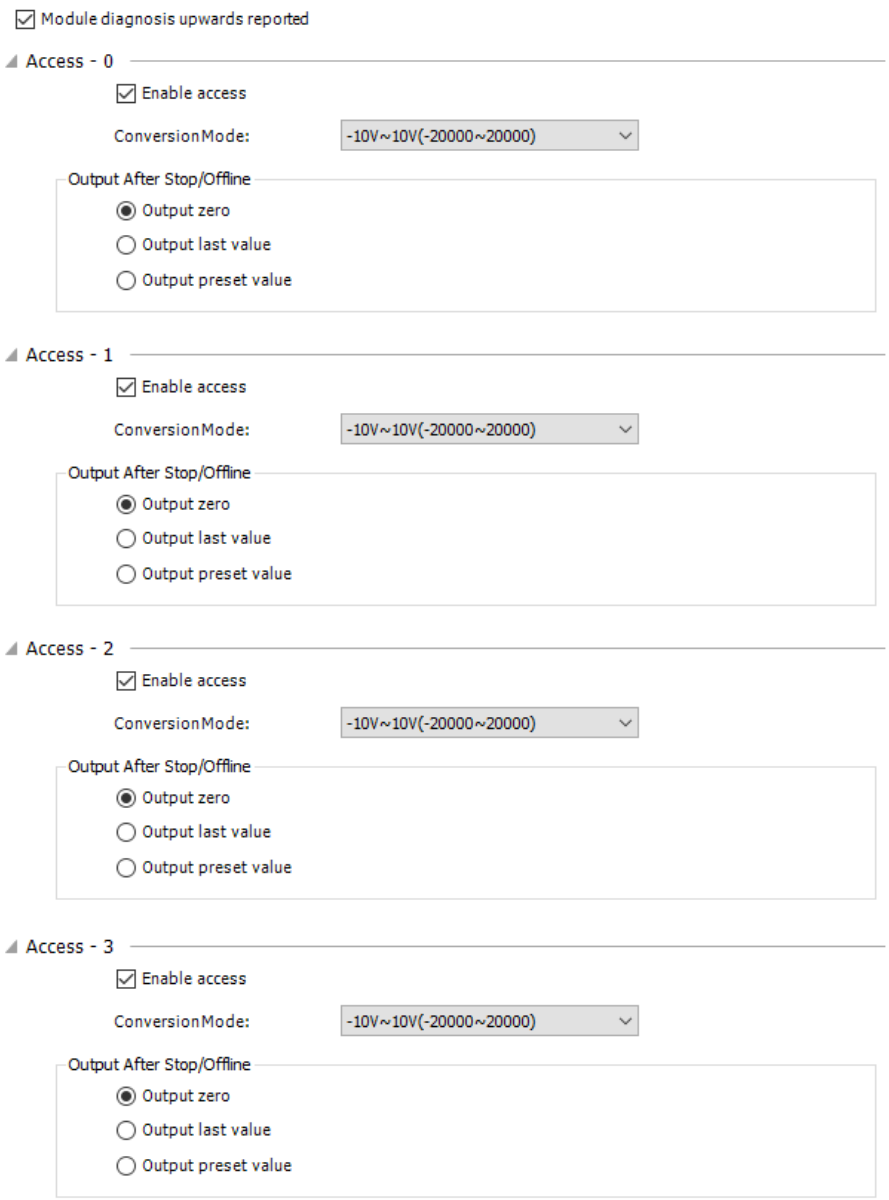




AO Modules of the GL10 Series

1. General settings

   An AO module has four channels, each of which has independent parameter settings and I/O mapping register (16-bit) settings. The following is the description of one channel in each module.



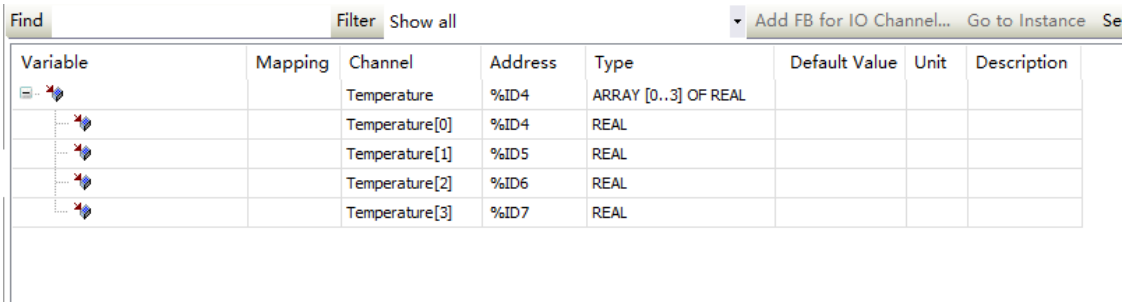
- Module diagnosis upwards reported: Specifies whether to report the module faults to the parent device (such as the CPU and remote module slave). If this function is checked and the parent device is configured with "Stopped On Failure", the parent device will stop the running of this device.
- Enable access: Specifies whether to activate the channel. The channel is available only after it is activated.
- Channel diagnosis upwards reported: Specifies whether to report the channel faults to the parent device (such as the CPU and remote module slave). If this function is checked and the parent device is configured with "Stopped On Failure", the parent device will stop the running of this device.
- AD Conversion Mode: Specifies the conversion mode of the analog output. This setting specifies the channel output conversion type, conversion value range, and the mappings between conversion types and digital values. The following table lists mappings between analog values of analog output and digital values.

-	Rated Output Range	Rated Digital Value	Output Limit Range	Digital Value Limit
Analog voltage output	–10 V to +10 V	–20000 to +20000	–11 V to +11 V	–22000 to +22000
	0 V to 10 V	0 to 20000	–0.5 V to +10.5 V	–1000 to +21000
	–5 V to +5 V	–20000 to +20000	–5.5 V to +5.5 V	–22000 to +22000
	0 V to 5 V	0 to 20000	–0.25 V to +5.25 V	–1000 to +21000
	1 V to 5 V	0 to 20000	0.8 V to 5.2 V	–1000 to +21000
Analog current output	0 mA to 20 mA	0 to 20000	0 mA to 21 mA	0 to 21000
	4 mA to 20 mA	0 to 20000	3.2 mA to 20.8 mA	–1000 to +21000

- Output After Stop/Offline: Sets the output retaining value after the module stops running.
- Output zero: Always outputs 0 after the module stops running.
- Output last value: Always outputs the last value after the module stops running.
- Output preset value: Always outputs the preset value after the module stops running. The preset value can be an analog value or a digital value. The analog values and digital values have the mapping relationship. If the analog or digital value is changed, its corresponding digital or analog value is also changed. The preset value range depends on the conversion mode. For details, see the description of conversion mode.

2. AO4 I/O mapping

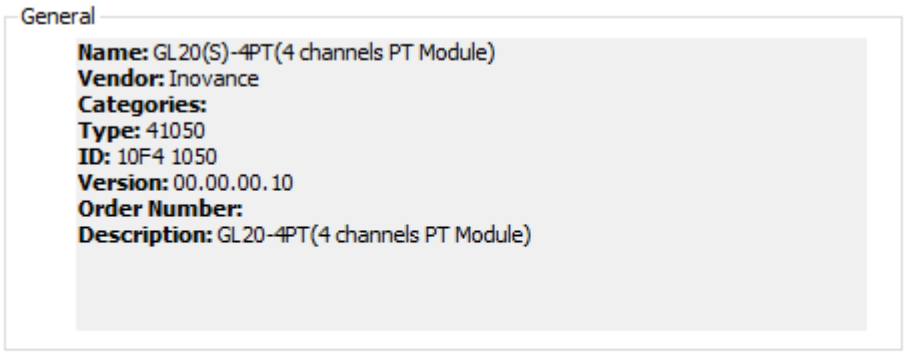
   AO4 is 4-channel analog output. Each channel matches a 16-digit integer. For the mappings between analog values and digital values, see general settings of analog output. The following figure shows the mapping page. On this page, each 16-bit integer can be mapped to a variable, and this variable is output to the current channel. Then the AO module converts the variable into the analog value for output. For details, click "I/O Mapping".

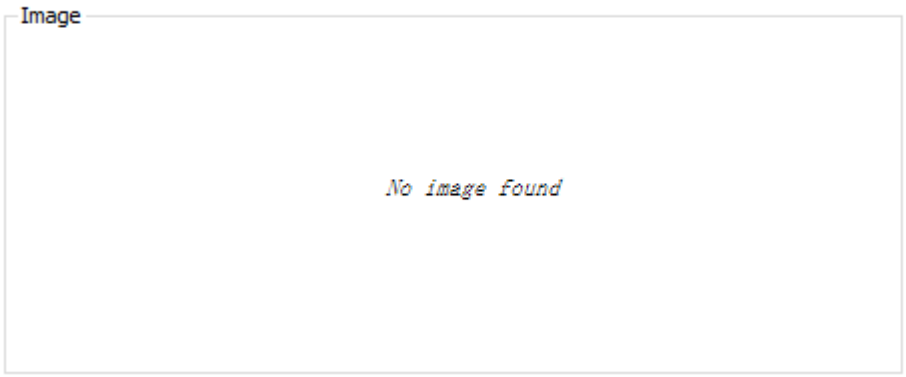


3. Information

   The AO4 module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Model Number, Description, Order Number, and Image.

   After login, the "Information" page reads and displays the PCB software version and logic software version of the AO4 module. The PCB software version is the embedded software version of AO4 and the logic software version is the FPGA software version within the AO4 module.





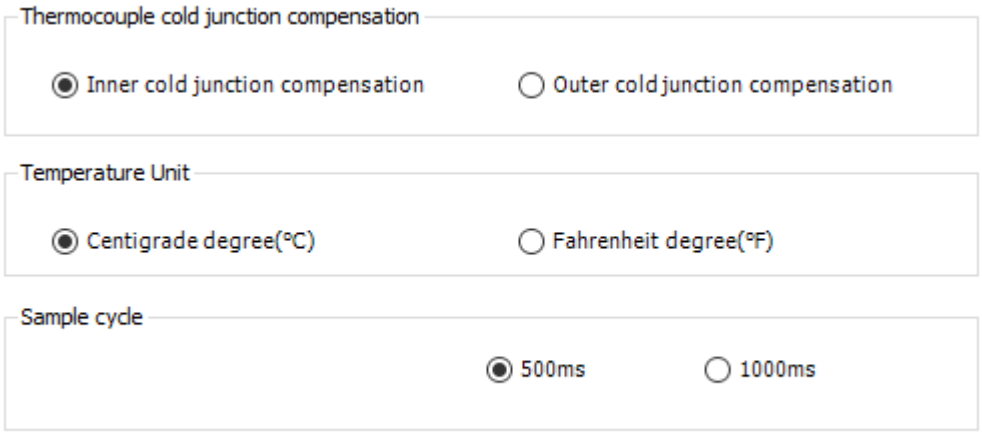
Temperature Detection Module of the GL10 Series

Temperature detection modules include 4TC modules (4-channel, supporting thermocouple), 8TC modules (8-channel, supporting thermocouple), and 4PT modules (4-channel, supporting resistance temperature detector). All the modules have their respective general settings and channel settings.

General settings include the unit type and sample cycle of the temperature detection module. Channel settings include the sensor type, filter time, overflow, and temperature offset of each channel.

1. General settings

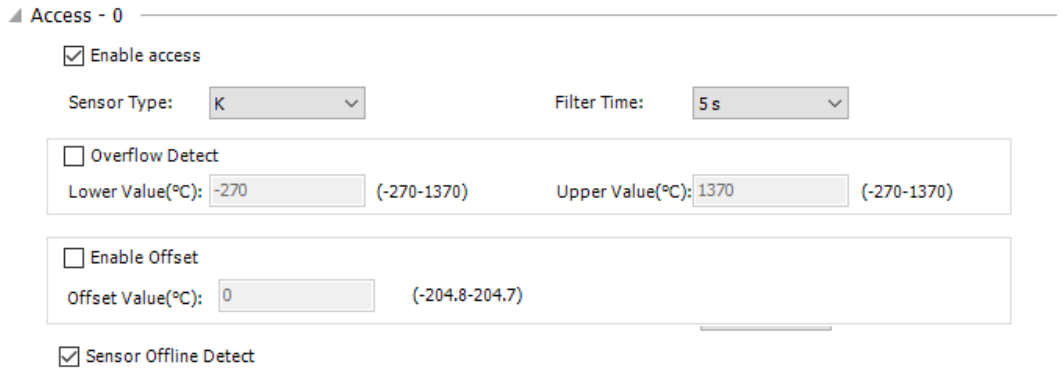
   The temperature detection module settings vary with the module type. The 4TC and 8TC modules support the cold junction compensation function, but the 4PT module does not. In addition, the 8TC module supports external cold junction compensation, but the 4TC module does not. The following figure shows the configuration page of the 8TC module.



- Module diagnosis upwards reported: Specifies whether to report the module faults to the parent device (such as the CPU and remote module slave). If this function is checked and the parent device is configured with "Stopped On Failure", the parent device will stop the running of this device.
- Cold junction compensation: Selects the cold junction compensation mode. Only the 8TC module supports external cold junction compensation, and the 8TC module uses channel 7 (the last channel) for the input of external cold junction compensation.
- Temperature Unit: Sets the input unit used by the temperature detection module, including Centigrade degree and Fahrenheit degree.
- Sample cycle: Sets the sample cycle used by the temperature detection module, including 250 ms, 500 ms, and 1000 ms.

2. Channel settings

Different types of modules support different numbers of channels. The 4TC and 4PT modules support 4 channels, and the 8TC module supports 8 channels. The channels have similar parameter settings. The following is the description for one channel. The following figure shows the channel setting interface of the 8TC module.



- Enable access: Specifies whether to activate the channel. The channel is available only after it is activated.
- Channel diagnosis upwards reported: Specifies whether to report the channel faults to the parent device (such as the CPU and remote module slave). If this function is checked and the parent

device is configured with "Stopped On Failure", the parent device will stop the running of this device.

- Default: Restores the default settings of the channel.
- Sensor Type: The sensor type and specifications of the 8TC and 4TC modules are listed in the following table. By default, the K sensor is used.

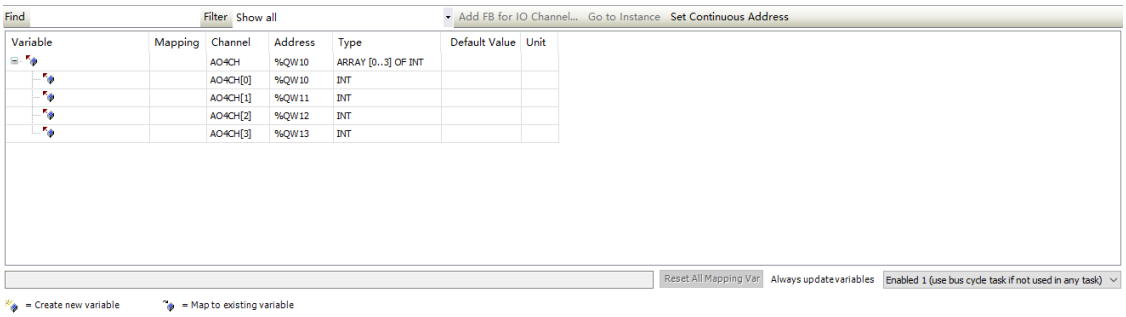
Item	Sensor Name	Temperature Range (°C)	Temperature Range (°F)
Thermocouple	B	250°C to 1800°C	482°F to 3272°F
	E	–270°C to +1000°C	–454°F to +1832°F
	N	–200°C to +1300°C	–328°F to +2372°F
	J	–210°C to +1200°C	–346°F to +2192°F
	K	–270°C to +1372°C	–454°F to +2502°F
	R	–50°C to +1768°C	–58°F to +3214°F
	S	–50°C to +1768°C	–58°F to +3214°F
	T	–270°C to +400°C	–454°F to +752°F

Item	Sensor Name	Temperature Range (°C)	Temperature Range (°F)
Resistance temperature detector	Pt100	–200°C to +850°C	–328°F to +1562°F
	Pt500	–200°C to +850°C	–328°F to +1562°F
	Pt1000	–200°C to +850°C	–328°F to +1562°F
	Cu100	–50°C to +150°C	–58°F to +302°F

- Filter Time: Specified the filter time of the temperature detection module when this channel is used, ranging from 0s to 100s. The default value is 5s.
- Overflow Detect: Enables overflow detection for the channel. If the temperature is beyond the specified range, an overflow fault is reported. For the temperature range, see the preceding table.
- Enable Offset: Sets the offset compensation for the temperature detection module, ranging from –204.8 to +204.7.
- Sensor Offline Detect: Enables offset alarming of the sensor.

3. I/O mapping

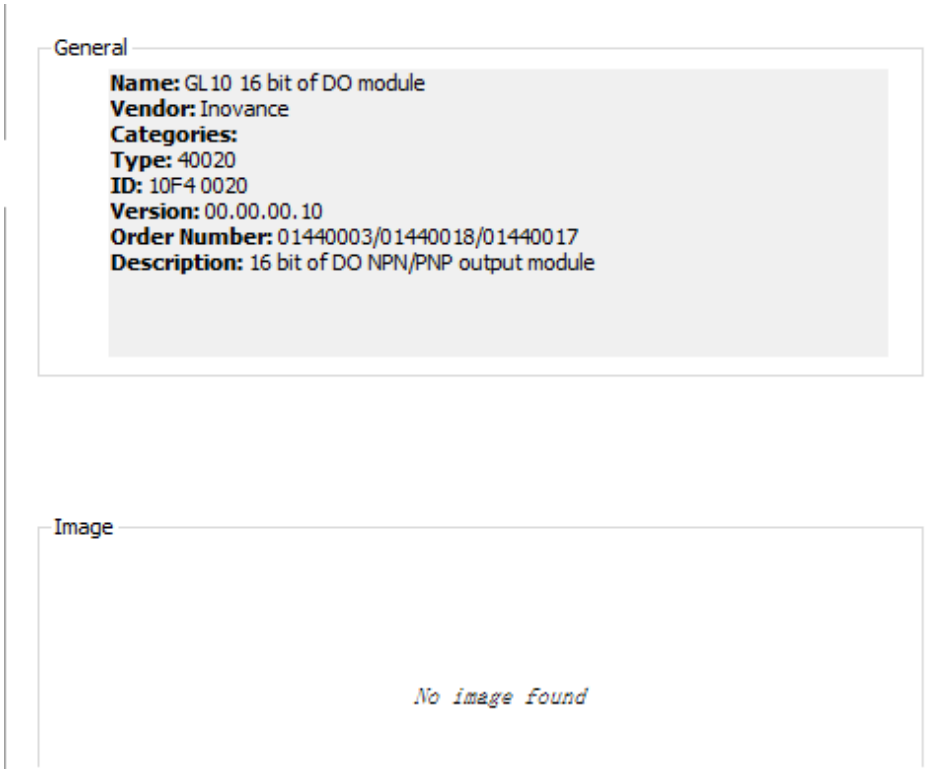
The numbers of supported channels and I/O mappings vary with the temperature detection module type. The following figure shows the I/O mapping page of the 4PT module. The parameter value of each channel is the temperature value. For details, click "I/O Mapping".



4. Information

The basic information of the temperature detection module is displayed, including Name, Vendor, Categories, Type, ID, Version, Model Number, Description, Order Number, and Image.

After login, the "Information" page displays the PCB software version and logic software version of the temperature detection module. The PCB software version is the embedded software version of the temperature detection module and the logic software version is the FPGA software version within the temperature detection module.

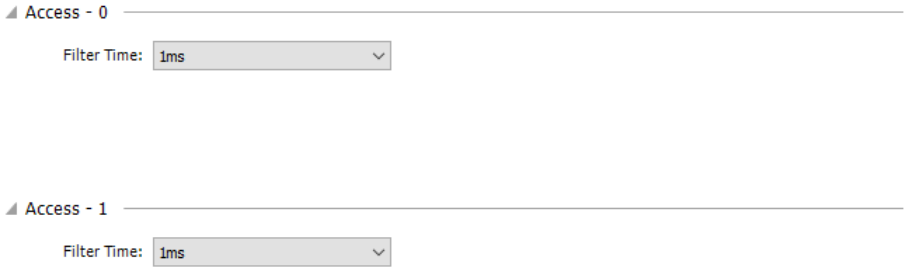


DI Modules of the GL20 Series

DI modules have the "Channel Settings", "Device Diagnosis", "I/O Mapping", "Status", and "Information" pages. You only need to map the I/O variables on the "I/O Mapping" page to obtain the DI values. DI modules include 8-channel, 16-channel, and 32-channel modules, with similar pages. The following takes the 16-channel DI module as an example.

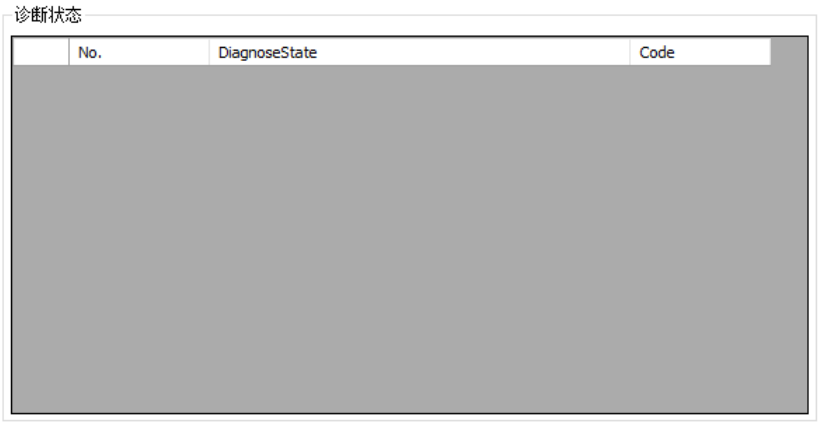
1. Channel Settings

   The following figure shows the "Channel Settings" page, on which you can set a filter time for each channel.



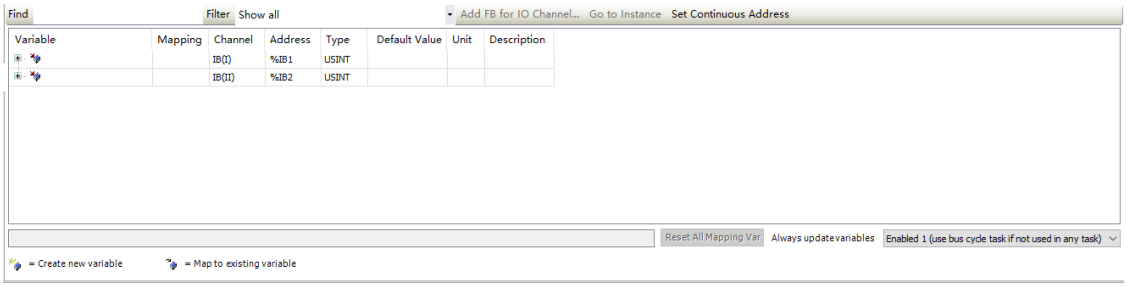
2. Device Diagnosis

   The following figure shows the "Device Diagnosis" page, on which you can view the error information of a module.



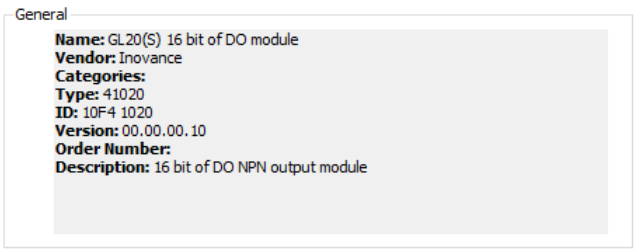
3. I/O Mapping

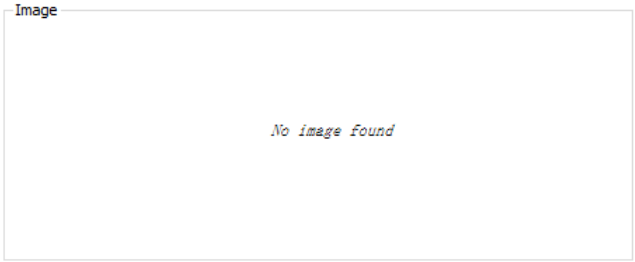
   The following figure shows the "I/O Mapping" page. Each bit or every eight bits can be mapped to one variable to obtain the input value. For details, see *“4.2.6 I/O Mapping Parameters” on page 127*.



4. Information

   The module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Order Number, Description, and Image, as shown in the following figure. After login, the "Information" page displays the module logic software version, which is the FPGA software version within the module.



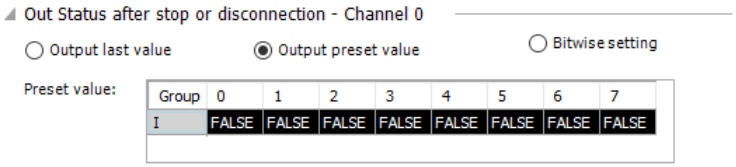


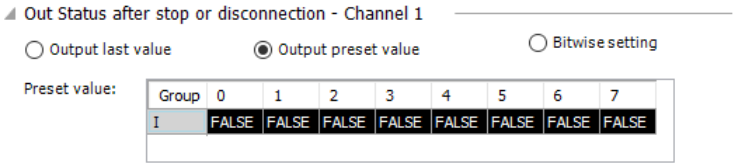
DO Modules of the GL20 Series

DO modules have the "Channel Settings", "Device Diagnosis", "I/O Mapping", "Status", and "Information" pages. You only need to map the I/O variables on the "I/O Mapping" page to obtain the DO values. DO modules include 8-channel, 16-channel, and 32-channel modules, with similar pages. The following takes the 16-channel DO module as an example.

1. Channel Settings

   The following figure shows the "Channel Settings" page, on which you can set the output status of each channel after the channel stops or is disconnected. Options are "Output last value", "Output preset value", and "Bitwise setting".





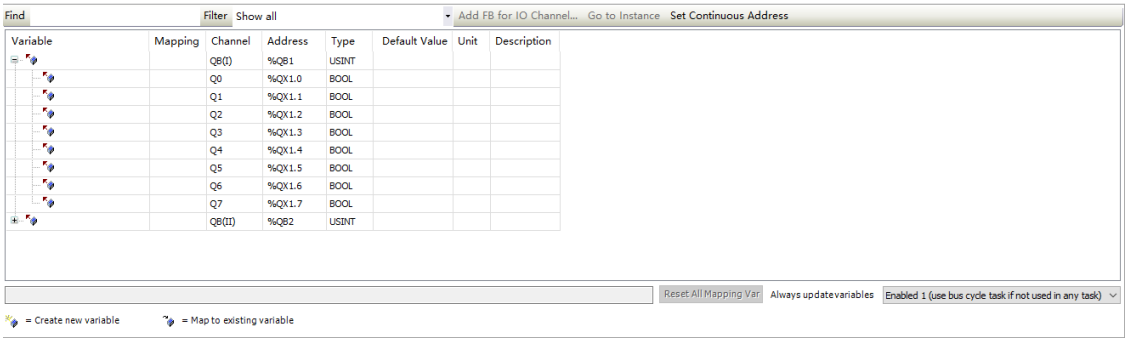
2. Device Diagnosis

   The following figure shows the "Device Diagnosis" page, on which you can view the error information of a module.



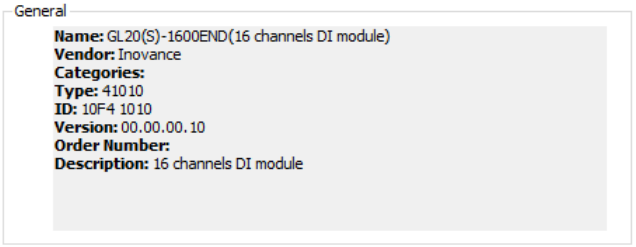
3. I/O Mapping

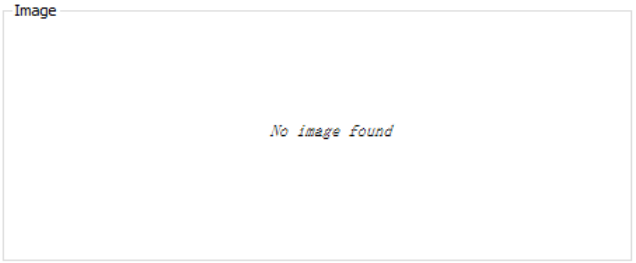
   The following figure shows the "I/O Mapping" page. Each bit or every eight bits can be mapped to one variable to obtain the output value. For details, see *“4.2.6 I/O Mapping Parameters” on page 127*.



4. Information

   The DO16 module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Model Number, Description, Order Number, and Image, as shown in the following figure. After login, the "Information" page displays the DO module logic software version, which is the FPGA software version within the DO module.



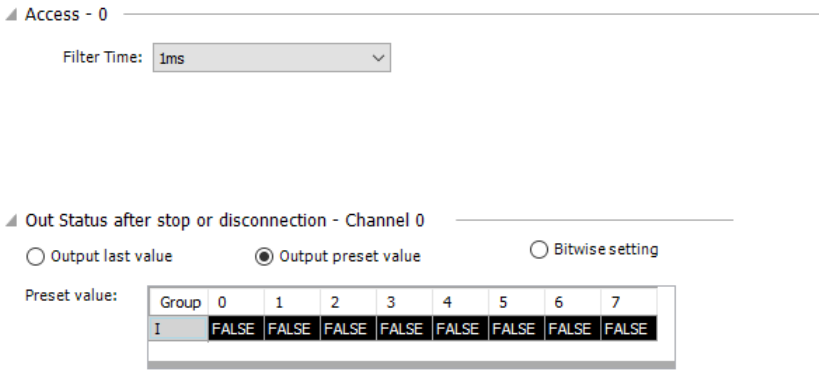


DI/DO Modules of the GL20 Series

DI/DO modules have the "Channel Settings", "Device Diagnosis", "I/O Mapping", "Status", and "Information" pages. The following takes the 8-channel DI module as an example.

1. Channel Settings

   The following figure shows the "Channel Settings" page, on which you can set the filter time for each channel and the output status of each channel after the channel stops or is disconnected. Options are "Output last value", "Output preset value", and "Bitwise setting".



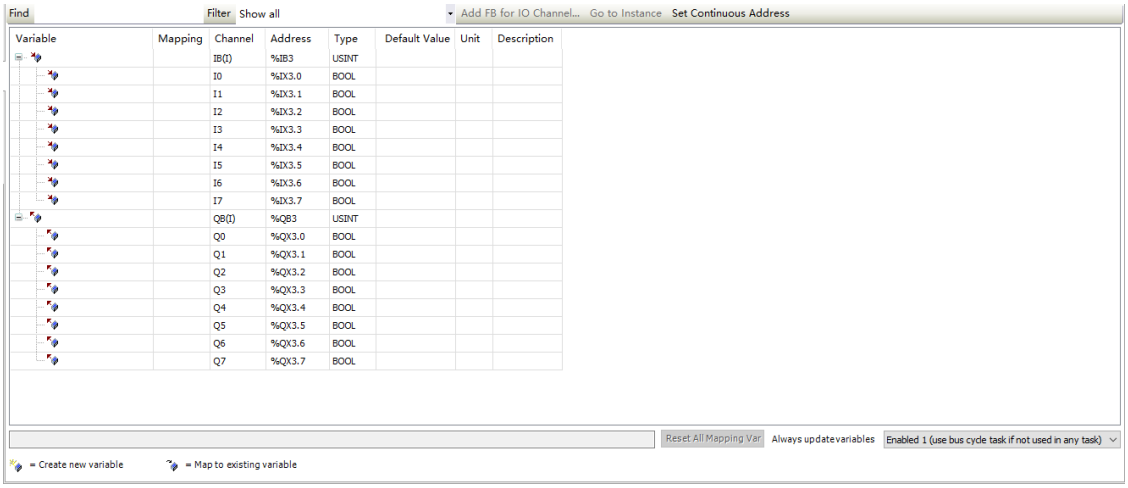
2. Device Diagnosis

The following figure shows the "Device Diagnosis" page, on which you can view the error information of a module.



3. I/O Mapping

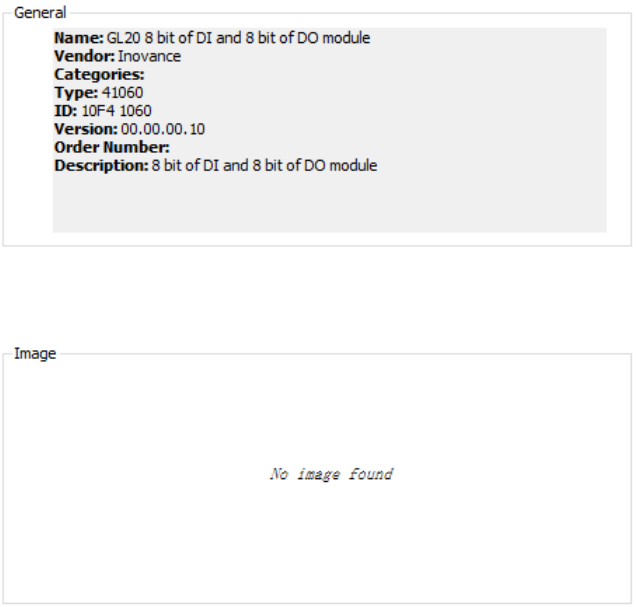
The following figure shows the "I/O Mapping" page. Each bit or every eight bits can be mapped to one variable to obtain the input value. For details, see *"4.2.6 I/O Mapping Parameters" on page 127*.



4. Information

The module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Order Number, Description, and Image, as shown in the following figure. After login, the

"Information" page displays the DI/DO module logic software version, which is the FPGA software version within the DI/DO module.

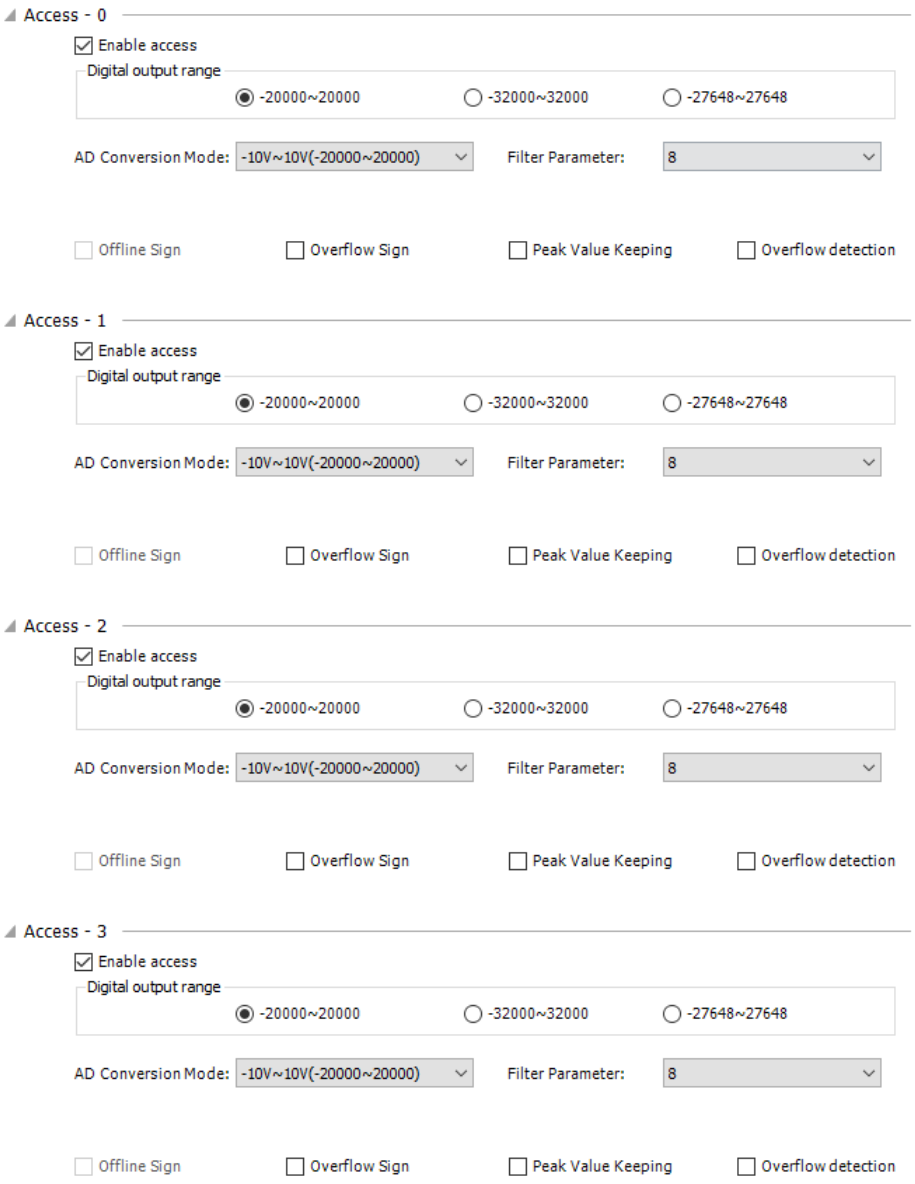


AI Modules of the GL20 Series

AI modules include 4AD modules (4-channel AI), 8ADI modules (8-channel AI, supporting current input), and 8ADV modules (8-channel AI, supporting voltage input). The "Channel Settings", "Device Diagnosis", "I/O Mapping", "Status", and "Information" pages are available for these modules. The following takes the 4AD AI module as an example.

1. Channel Settings

   An AI module has four channels, each of which has independent parameter settings and I/O mapping register (16-bit) settings. The following is the description of one channel in each module.



- Enable access: Specifies whether to activate the channel. The channel is available only after it is activated.
- Digital output range: Sets the analog output range, which determines the range of the values output by this channel for conversion.
- AD Conversion Mode: Specifies the conversion mode of the analog input. This setting specifies the channel input conversion type, conversion value range, and the mappings between conversion types and digital values. The following table lists mappings between analog values of analog input and digital values.

-	Rated Input Range	Rated Digital Value	Input Limit Range	Digital Value Limit
Analog voltage input	–10 V to +10 V	–20000 to +20000 –32000 to +32000 –27648 to +27648	–10.24 V to +10.24 V	–20400 to +20400 –32640 to +32640 –28200 to +28200
	0 V to 10 V	0 to 20000 0 to 32000 0 to 27648	–0.5 V to +10.24 V	–1000 to +20400 –1600 to +32640 –1382 to +28200
	–5 V to +5 V	–20000 to +20000 –32000 to +32000 –27648 to +27648	–5.12 V to +5.12 V	–20400 to +20400 –32640 to +32640 –28200 to +28200
	0 V to 5 V	0 to 20000 0 to 32000 0 to 27648	–0.25 V to +5.12 V	–1000 to +20400 –1600 to +32640 –1382 to +28200
	1 V to 5 V	0 to 20000 0 to 32000 0 to 27648	0.8 V to 5.12 V	–1000 to +20400 –1600 to +32640 –1382 to +28200
Analog current input	–20 mA to +20 mA	–20000 to +20000 –32000 to +32000 –27648 to +27648	–20.56 mA to +20.56 mA	–20400 to +20400 –32640 to +32640 –28200 to +28200
	0 mA to 20 mA	0 to 20000 0 to 32000 0 to 27648	–1 mA to 20.56 mA	–1000 to +20400 –1600 to +32640 –1382 to +28200
	4 mA to 20 mA	0 to 20000 0 to 32000 0 to 27648	3.2 mA to 20.56 mA	–1000 to +20400 –1600 to +32640 –1382 to +28200

- Filter Parameter: Indicates the filter time of the analog input channel, ranging from 1 ms to 255 ms.
- Offline Sign: Specifies whether to detect the offline state of the AI channel. The system cannot distinguish the input value 0 and offline state of the AI module, so all values in the conversion mode range, including 0, cannot activate the offline sign.
- Overflow Sign: Specifies whether to detect overflow of the AI channel.
- Peak Value Keeping: Specified whether to keep the peak value input of the AI channel.

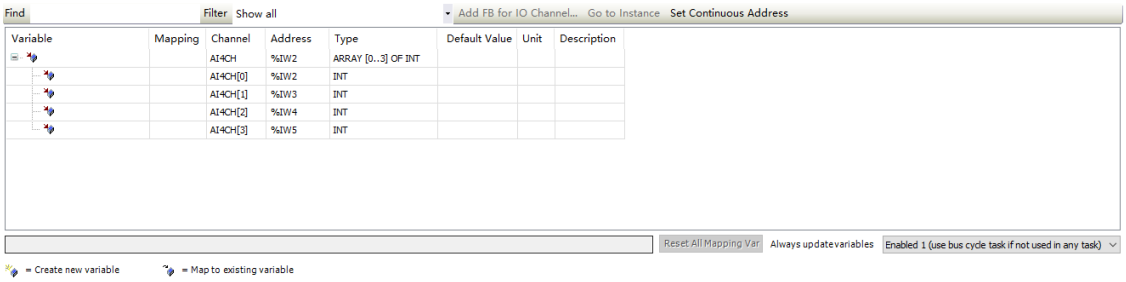
2. Device Diagnosis

The following figure shows the "Device Diagnosis" page, on which you can view the error information of a module.



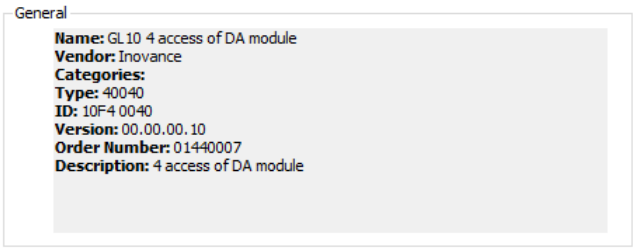
3. I/O Mapping

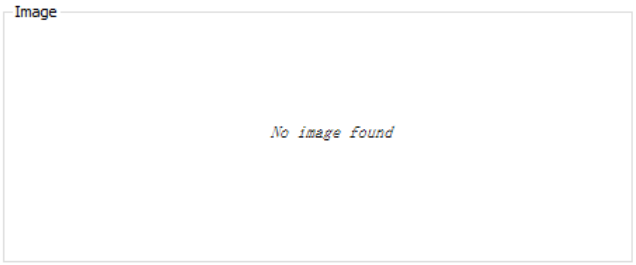
   AI4 is 4-channel analog input. Each channel matches a 16-digit integer. For the mappings between analog values and digital values, see the section "Channel Settings". On this page, each 16-digit integer can be mapped to a variable to obtain the digital value matching an analog value of the input channel. For details, see *"4.2.6 I/O Mapping Parameters" on page 127*.



4. Information

   The module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Model Number, Description, Order Number, and Image. After login, the "Information" page displays the PCB software version and logic software version of the AI4 module. The PCB software version is the embedded software version of AI4 and the logic software version is the FPGA software version within the AI4 module.



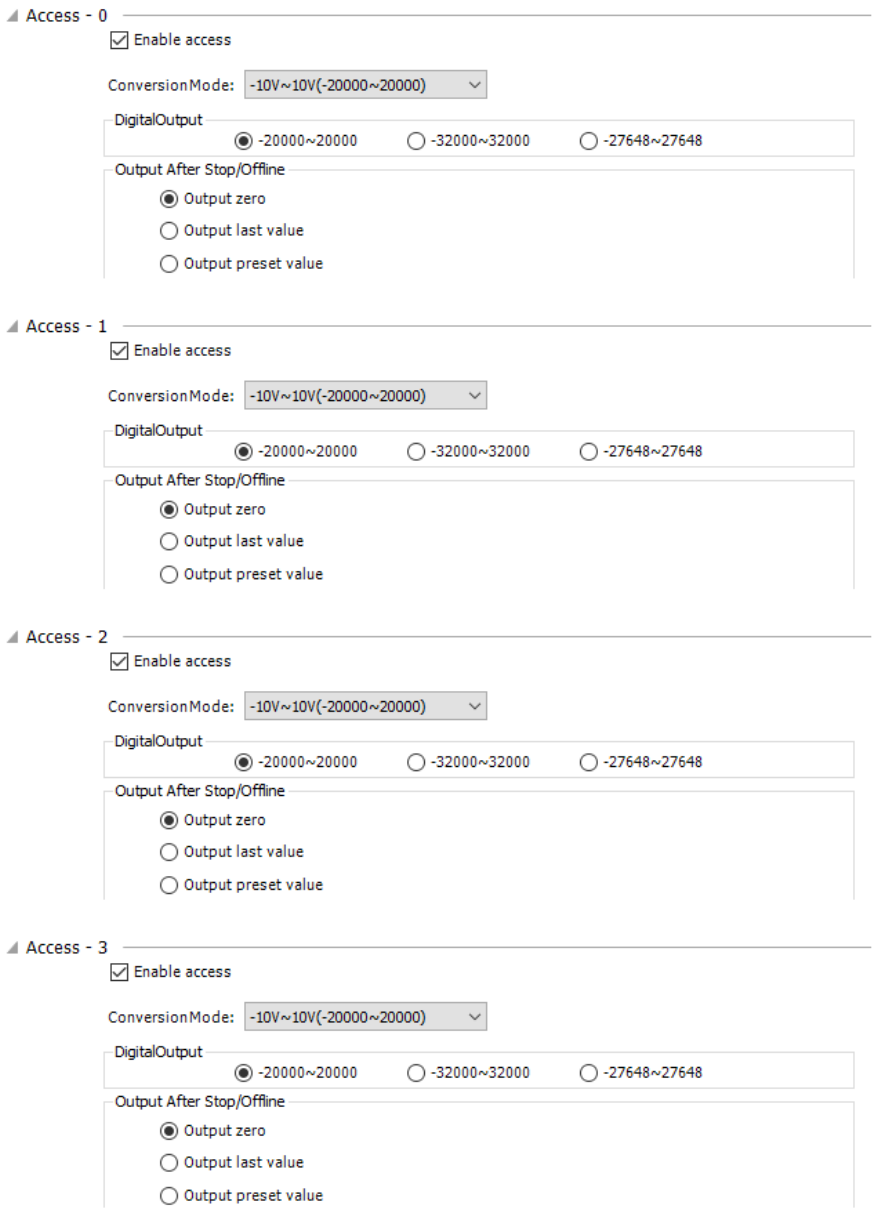


AO Modules of the GL20 Series

AO modules have the "Channel Settings", "Device Diagnosis", "I/O Mapping", "Status", and "Information" pages.

1. Channel Settings

   An AO module has four channels, each of which has independent parameter settings and I/O mapping register (16-bit) settings. The following is the description of one channel in each module.



- Enable access: Specifies whether to activate the channel. The channel is available only after it is activated.
- Digital output range: Sets the analog output range, which determines the range of the values output by this channel for conversion.
- AD Conversion Mode: Specifies the conversion mode of the analog output. This setting specifies the channel output conversion type, conversion value range, and the mappings between conversion types and digital values. The following table lists mappings between analog values of analog output and digital values.

-	Rated Output Range	Rated Digital Value	Output Limit Range	Digital Value Limit
Analog voltage output	–10 V to +10 V	–20000 to +20000 –32000 to +32000 –27648 to +27648	–10.24 V to +10.24 V	–20400 to +20400 –32640 to +32640 –28200 to +28200
	0 V to 10 V	0 to 20000 0 to 32000 0 to 27648	–0.5 V to +10.24 V	–1000 to +20400 –1600 to +32640 –1382 to +28200
	–5 V to +5 V	–20000 to +20000 –32000 to +32000 –27648 to +27648	–5.12 V to +5.12 V	–20400 to +20400 –32640 to +32640 –28200 to +28200
	0 V to 5 V	0 to 20000 0 to 32000 0 to 27648	–0.25 V to +5.12 V	–1000 to +20400 –1600 to +32640 –1382 to +28200
	1 V to 5 V	0 to 20000 0 to 32000 0 to 27648	0.8 V to 5.12 V	–1000 to +20400 –1600 to +32640 –1382 to +28200
Analog current output	0 mA to 20 mA	0 to 20000 0 to 32000 0 to 27648	–1 mA to 20.56 mA	–1000 to +20400 –1600 to +32640 –1382 to +28200
	4 mA to 20 mA	0 to 20000 0 to 32000 0 to 27648	3.2 mA to 20.56 mA	–1000 to +20400 –1600 to +32640 –1382 to +28200

- Output After Stop/Offline: Sets the output retaining value after the module stops running.
- Output zero: Always outputs 0 after the module stops running.
- Output last value: Always outputs the last value after the module stops running.
- Output preset value: Always outputs the preset value after the module stops running. The preset value can be an analog value or a digital value. The analog values and digital values have the mapping relationship. If the analog or digital value is changed, its corresponding digital or analog value is also changed. The preset value range depends on the conversion mode. For details, see the description of conversion mode.

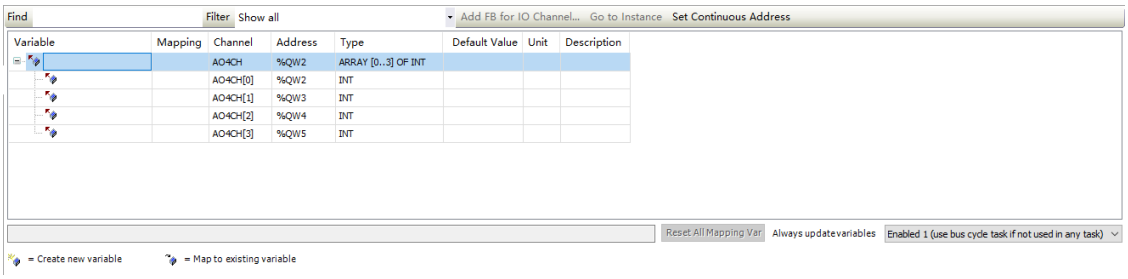
2. Device Diagnosis

The following figure shows the "Device Diagnosis" page, on which you can view the error information of a module.



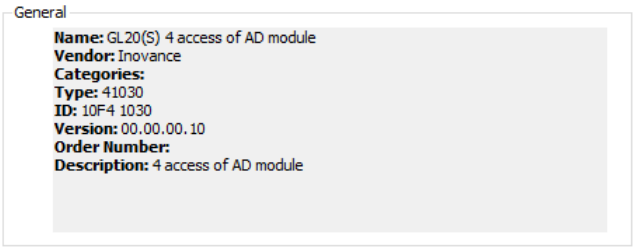
3. I/O Mapping

AO4 is 4-channel analog output. Each channel matches a 16-digit integer. For the mappings between analog values and digital values, see the section "Channel Settings". The following figure shows the "Channel Settings" page. On this page, each 16-bit integer can be mapped to a variable, and this variable is output to the current channel. Then the AO module converts the variable into the analog value for output. For details, see *"4.2.6 I/O Mapping Parameters" on page 127*.



4. Information

The module basic information is displayed, including Name, Vendor, Categories, Type, ID, Version, Model Number, Description, Order Number, and Image. After login, the "Information" page reads and displays the PCB software version and logic software version of the AO4 module. The PCB software version is the embedded software version of AO4 and the logic software version is the FPGA software version within the AO4 module.

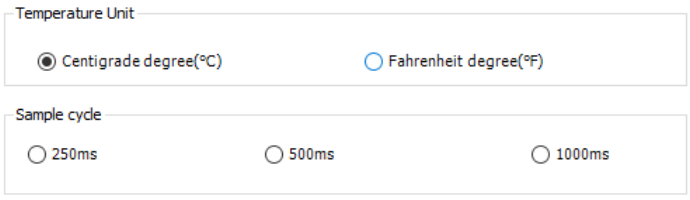




Temperature Detection Modules of the GL20 Series

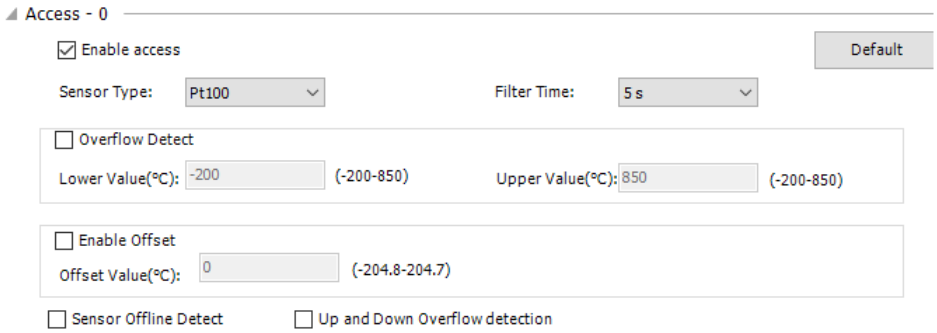
Temperature detection modules include 4PT modules (4-channel, supporting resistance temperature detector) and 4TC modules (4-channel, supporting thermocouple). All the modules have their respective general settings and channel settings. General settings include the unit type and sample cycle of the temperature detection module. Channel settings include the sensor type, filter time, overflow, and temperature offset of each channel. The following takes the 4PT temperature detection module as an example.

1. General settings



- Temperature Unit: Sets the input unit used by the temperature detection module, including Centigrade degree and Fahrenheit degree.
- Sample cycle: Sets the sample cycle used by the temperature detection module, including 250 ms, 500 ms, and 1000 ms.

2. Channel Settings



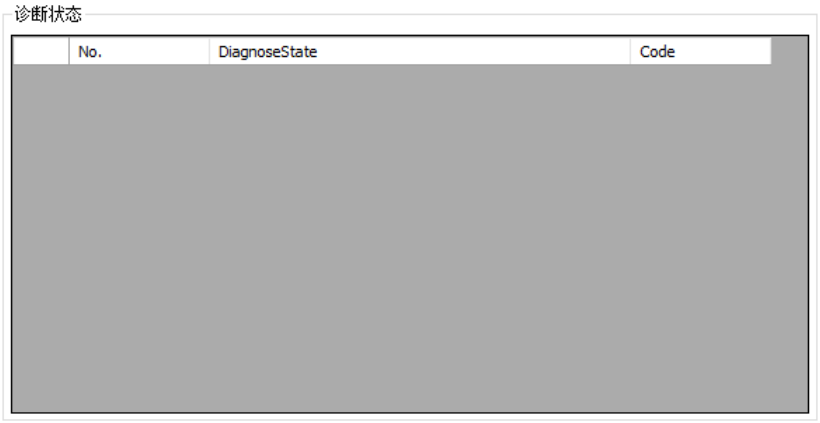
- Enable access: Specifies whether to activate the channel. The channel is available only after it is activated.
- Default: Restores the default settings of the channel.
- Sensor Type: The sensor type and specifications of the 4PT module are listed in the following table. By default, the K sensor is used.

Type	Sensor Name	Temperature Range (°C)	Temperature Range (°F)
Resistance temperature detector	Pt100	–200°C to +850°C	–328°F to +1562°F
	Pt500	–200°C to +850°C	–328°F to +1562°F
	Pt1000	–200°C to +850°C	–328°F to +1562°F
	Cu100	–50°C to +150°C	–58°F to +302°F
	KTY84	–50°C to +150°C	32°F to 392°F
	NTC5K (B value: 2000)	–30°C to +200°C	–22°F to +392°F
	NTC5K (B value: 3950)	–15°C to +100°C	5°F to 212°F
	NTC5K (B value: 6000)	0°C to 100°C	32°F to 212°F
	NTC10K (B value: 2000)	–25.0°C to +200°C	–13°F to +392°F
	NTC10K (B value: 3950)	0°C to 150°C	32°F to 302°F
	NTC10K (B value: 6000)	–6°C to +100°C	42.8°F to 212°F

- Filter Time: Specified the filter time of the temperature detection module when this channel is used, ranging from 0s to 100s. The default value is 5s.
- Overflow Detect: Enables overflow detection for the channel. If the temperature is beyond the specified range, an overflow fault is reported.
- Enable Offset: Sets the offset compensation for the temperature detection module, ranging from –204.8 to +204.7.
- Sensor Offline Detect: Enables offset alarming of the sensor.
- Overflow detection: Enables the overflow alarming function.

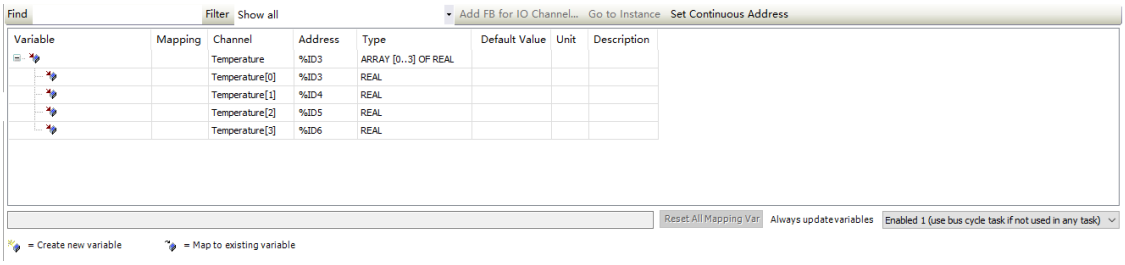
3. Device Diagnosis

The following figure shows the "Device Diagnosis" page, on which you can view the error information of a module.



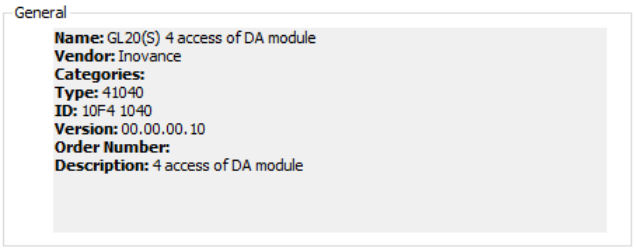
4. I/O Mapping

   The numbers of supported channels and I/O mappings vary with the temperature detection module type. The following figure shows the "I/O Mapping" page of the 4PT module. The parameter value of each channel is the temperature value. For details, see *“4.2.6 I/O Mapping Parameters” on page 127*.



5. Information

   The basic information of the temperature detection module is displayed, including Name, Vendor, Categories, Type, ID, Version, Model Number, Description, Order Number, and Image. After login, the "Information" page displays the PCB software version and logic software version of the temperature detection module. The PCB software version is the embedded software version of the temperature detection module and the logic software version is the FPGA software version within the temperature detection module.





Communication Modules of the GL20 Series

The GL20 series has the communication modules 2CAN, 2S485, 2SCOM, and 1DNM. For configuration of each module, see the *user guide* of the specific module.

Process Module of the GL20 Series

The GL20 series has a GL20-2SSI 2-channel encoder input module. For the module configuration, see the *GL20-2SSI 2-Channel Encoder Input Module User Guide*.

4.2.5 High-Speed I/O Configuration

Overview

Caution

- The AM300, AM400, AM500, AM600, and AC700 series support the high-speed I/O function.
- Ports enabled with high-speed I/O can no longer serve as ordinary ports and their I/O status cannot be changed through I/O mapping.

High-speed I/O provides the following functions:

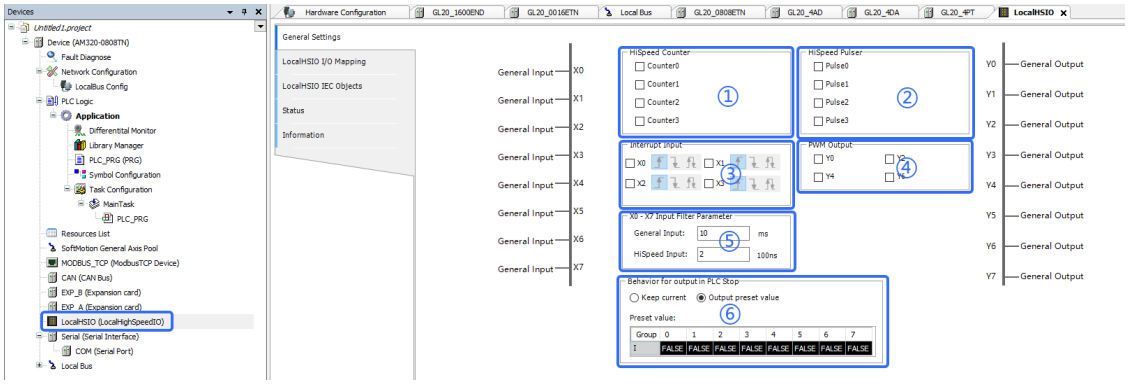
- High-speed counter
- High-speed output
- PWM output
- High-speed input edge interrupt

The following table lists the number of channels supported by high-speed I/O functions of different PLC series.

Function	AM400/600	AM300/500	AC700
High-speed counter	8	4	4
High-speed pulse	4	4	-
PWM output	8	4	2
High-speed input edge interrupt	8	4	-

High-Speed I/O Configuration of AM300/AM500/AC700

1. For high-speed I/O wiring guide of the AM300/AM500/AC700 series, see the *hardware guide* or *user guide* of the corresponding series.
2. In the left device tree, double-click "LocalHSIO". The "LocalHSIO" configuration page is displayed.



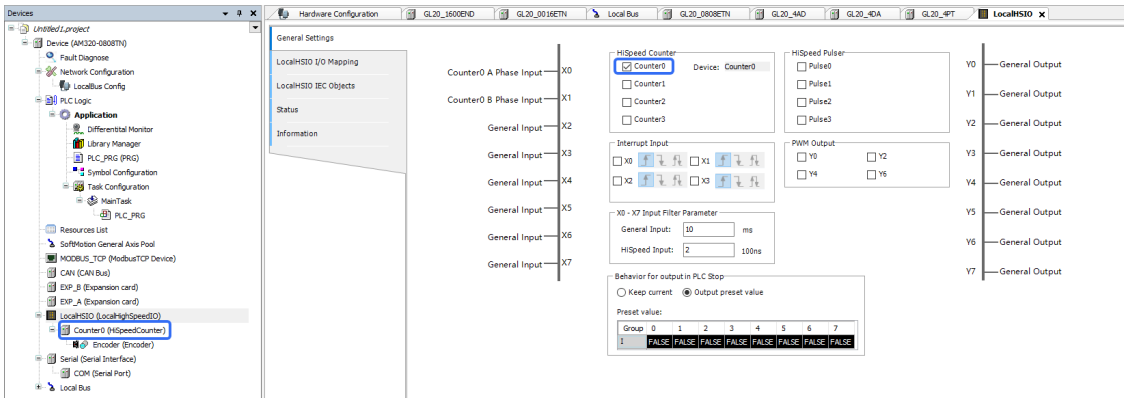
No.	Function Configuration
①	High-speed counter configuration
②	High-speed pulse output configuration
③	High-speed input edge interrupt configuration
④	PWM output configuration
⑤	Terminal filter parameter configuration
⑥	Configuration of output status after PLC stop

High-speed counter configuration

High-speed counter function parameters include the counter mode (such as single-phase, A/B phase single/double/quadruple frequency, pulse+direction, and CW/CCW), hardware reset, probe, preset, and comparison output.

Taking "Counter 0" for example (counter numbers range from 0 to 3), the configuration procedure is as follows.

1. On the "Basic Configuration" page, check "Counter 0". The item "Counter0 (High-speed Counter)" is automatically inserted to the left device tree and counter 0 is enabled.

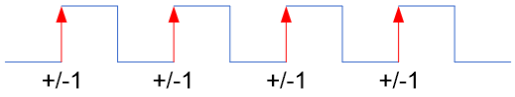


2. Configure the counter mode and signal source.

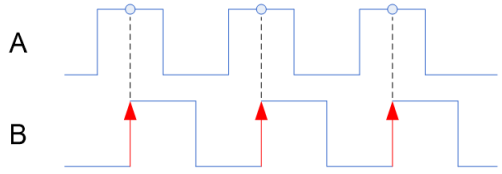
   a. Double-click "Counter0 (High-speed Counter)". The "Counter0" page is displayed.

   b. On the "Basic Configuration" page, select a counter mode from the drop-down list of "Mode" and select the hardware input terminal used from the drop-down list of "Single Source".

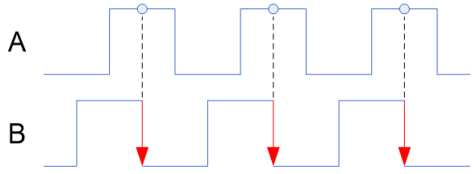
   - Single-phase Count: Select "Single-phase Count" from the drop-down list of "Mode". Then, the system receives the pulse signal of the external single-phase encoder and only one hardware input pot is occupied. Or, you can select the pulse signal regularly generated in the software at an interval of 1 μs/1 ms, and no hardware input port is occupied.



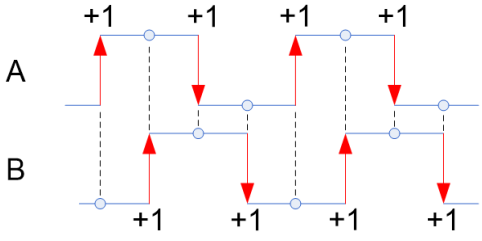
   - A/B-phase counter: Select "Phase A/B single frequency" from the drop-down list of "Mode". The system receives the pulse signal of the external A/B-phase encoder and double/quadruple frequency can be configured for the A/B phase. In this case, two hardware input ports are occupied.



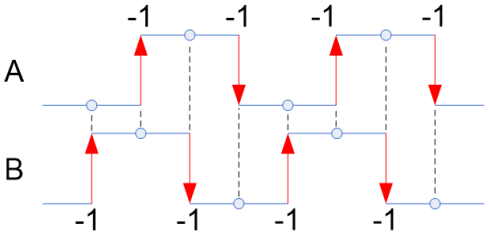
Phase A leading phase B Incremental count



Phase B leading phase A Decremental count

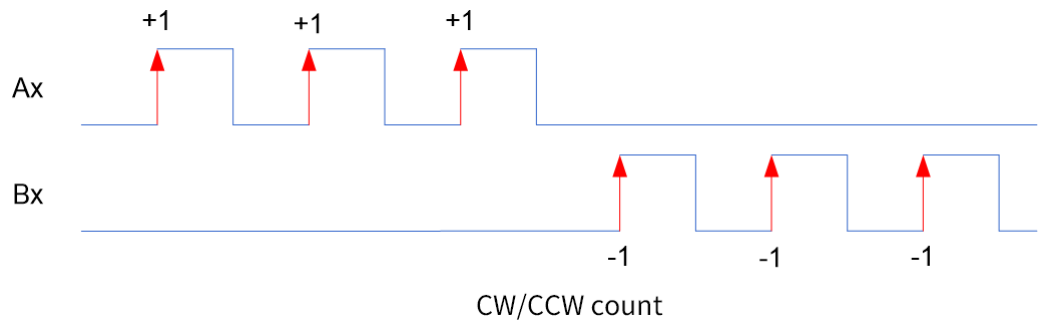


Quadruple incremental count



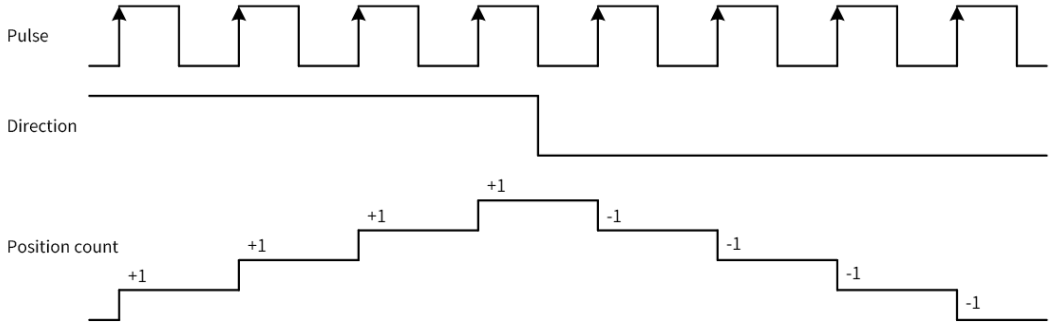
Quadruple decremental count

   - CW/CCW Count: Select "CW/CCW" from the drop-down list of "Mode". The system receives the pulse signal of the external CW/CCW encoder and two hardware input ports are occupied.

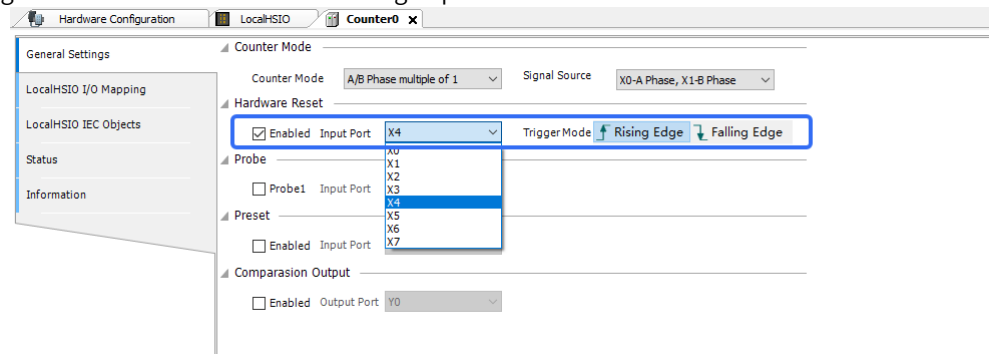


CW/CCW count

- Pulse+direction counter: Select "Pulse+Direction" from the drop-down list of "Mode". In this mode, when the direction signal is ON, the high-speed counter counts up the pulse signals. When the direction signal is OFF, the high-speed counter counts down the pulse signals.



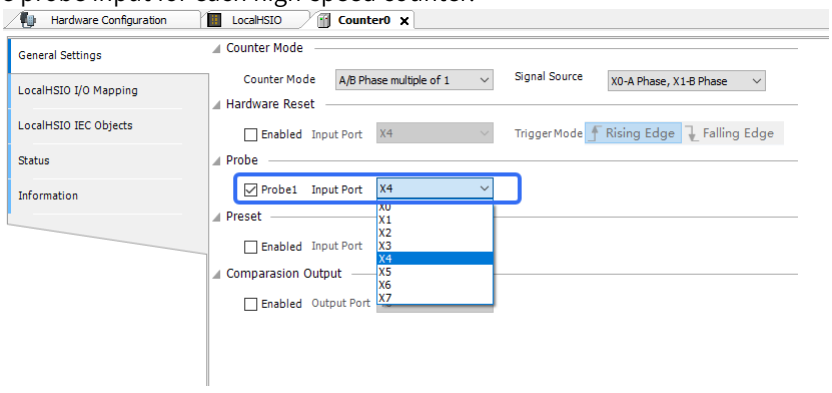
3. Configure one hardware reset for each high-speed counter.



a. Check "Enable".

b. Set the input terminal and trigger mode as needed.

4. Configure one probe input for each high-speed counter.

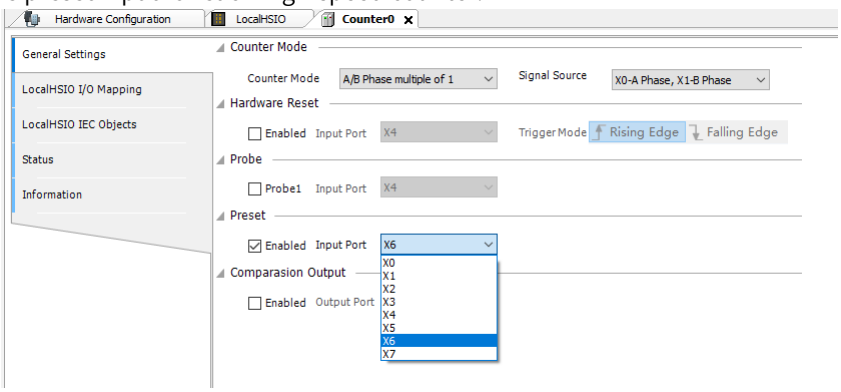


a. Check "Probe 1".

b. Set the input terminal as needed.

After configuration, the position latch of the counter can be realized through the HC_TouchProbe function block, or the position latch of the specified axis can be realized through the HC_ VirtualTouchProbe function block.

5. Configure one preset input for each high-speed counter.

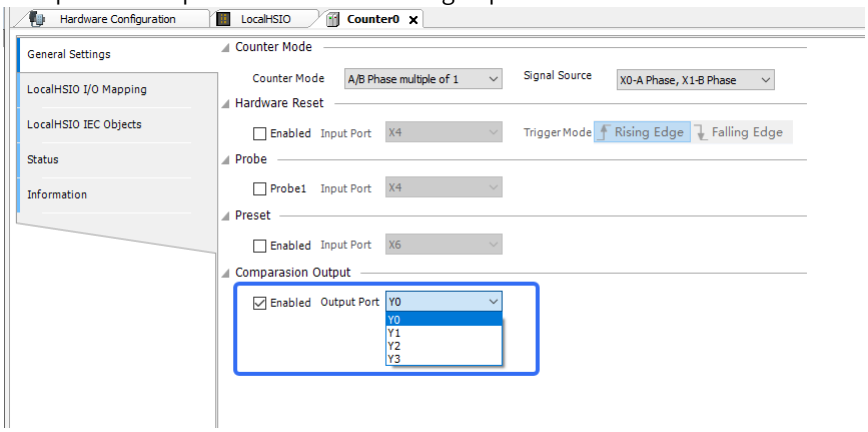


a. Check "Enable".

b. Set the input terminal as needed.

After configuration, the position preset of the counter can be realized through the HC_Preset function block.

6. Enable one comparison output function for each high-speed counter.



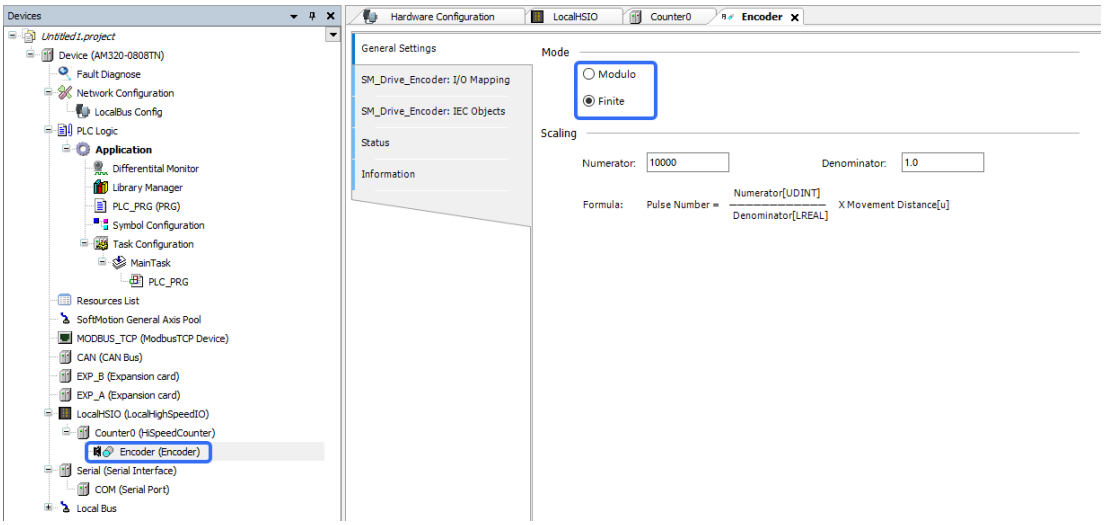
a. Check "Enable".

b. Set the output terminal as needed.

After configuration, the position comparison output of the counter is realized through the function blocks HC_Compare, HC_ArrayCompare, and HC_StepCompare.

7. Configure the encoder axis mode.

a. Double-click "Encoder". The "Encoder" page is displayed.



b. Select the Modulo mode or linear mode as needed.

- Modulo mode: The high-speed counter operates cyclically in the interval of [0, rotation cycle). Since the high-speed counter is a 32-bit counter, the rotation cycle must be within the 32-bit integer range [–2147483648, +2147483647] after being converted to pulse units.
- Linear mode: The high-speed counter operates in the interval of [negative limit, positive limit]. When the direction is negative, the count value decreases in the negative direction. After the negative limit is reached, the count value no longer decreases. When the direction is positive, the count value increases in the positive direction. After the positive limit is reached, the count value no longer increases. Since the high-speed counter is a 32-bit counter, the negative and positive limits must be within the 32-bit integer range [–2147483648, +2147483647] after being converted to pulse units.

c. Set the numerator and denominator of the scaling ratio.

High-speed counters use pulse units during counting, and use common measurement units for motion control instructions such as millimeters, degrees, and inches, which are called user units (Unit).

The calculation equation from user unit to pulse unit is as follows:

$$\text{Number of pulses (unit: pulse)} = \frac{\text{Scaling numerator [UDINT]}}{\text{Scaling denominator [LREAL]}} \times \text{Travel distance [unit in application]}$$

Example:

Scaling ratio numerator = 10000

Scaling ratio denominator = 1.0

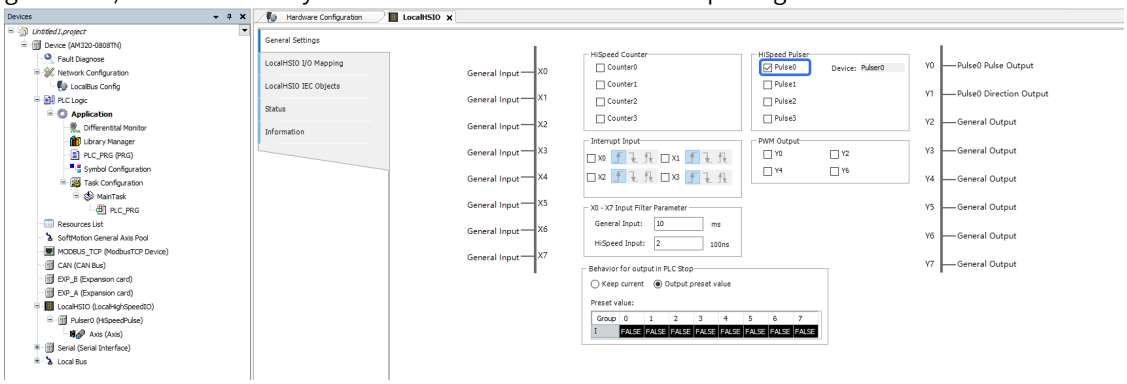
When the number of pulses received by the high-speed counter is 10,000, the count value of the high-speed counter increases by 1.

Configuring the high-speed pulse output function

High-speed pulse axis functions include output mode (such as pulse only, A/B phase, pulse+direction, and CW/CCW), probe, home, positive limit, negative limit, and emergency stop.

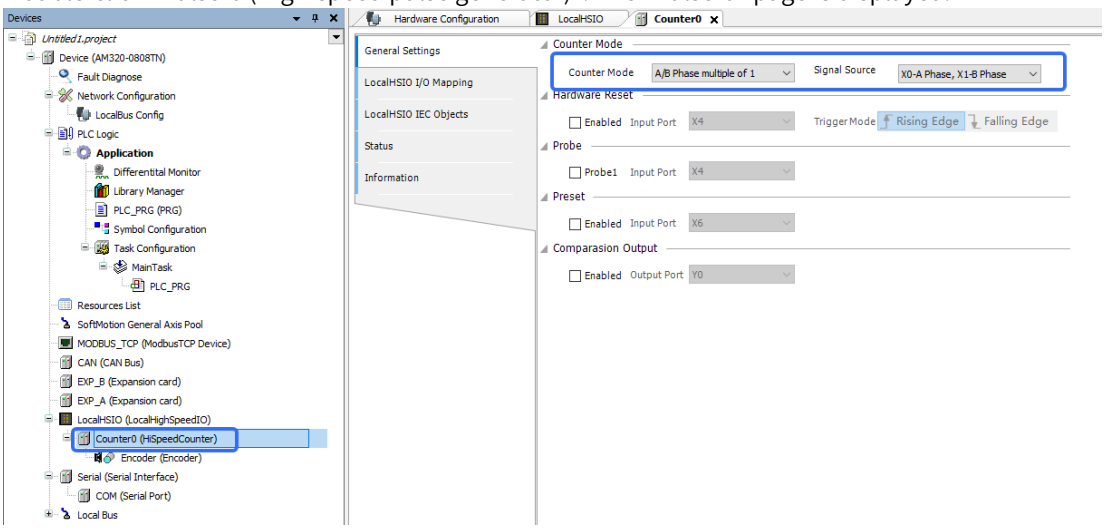
The following takes "Pulse generator 0" (pulse generators 0 to 3 supported) as an example.

1. On the "Basic Configuration" page, check "Pulse generator 0". The item "Pulser0 (High-speed pulse generator)" is automatically inserted to the left device tree and pulse generator 0 is enabled.



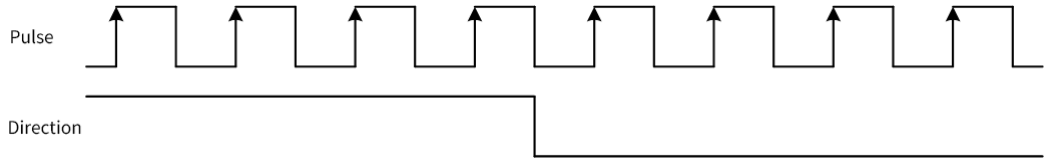
2. Set the output mode and output terminal.

a. Double-click "Pulser0 (High-speed pulse generator)". The "Pulser0" page is displayed.

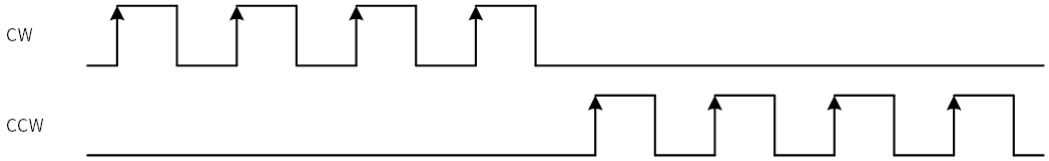


b. On the "Basic Configuration" page, select an output mode from the drop-down list of "Output Mode" and select the hardware output terminal used from the drop-down list of "Output Terminal".

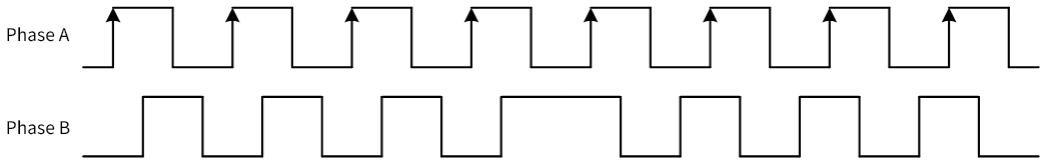
- Pulse+direction: Select "Pulse+Direction" from the drop-down list of "Output Mode". The system outputs the pulse+direction signal and two hardware output ports are occupied.



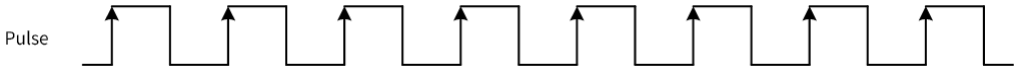
- CW/CCW: Select "CW/CCW" from the drop-down list of "Output Mode". The system outputs the CW/CCW signal and two hardware output ports are occupied.



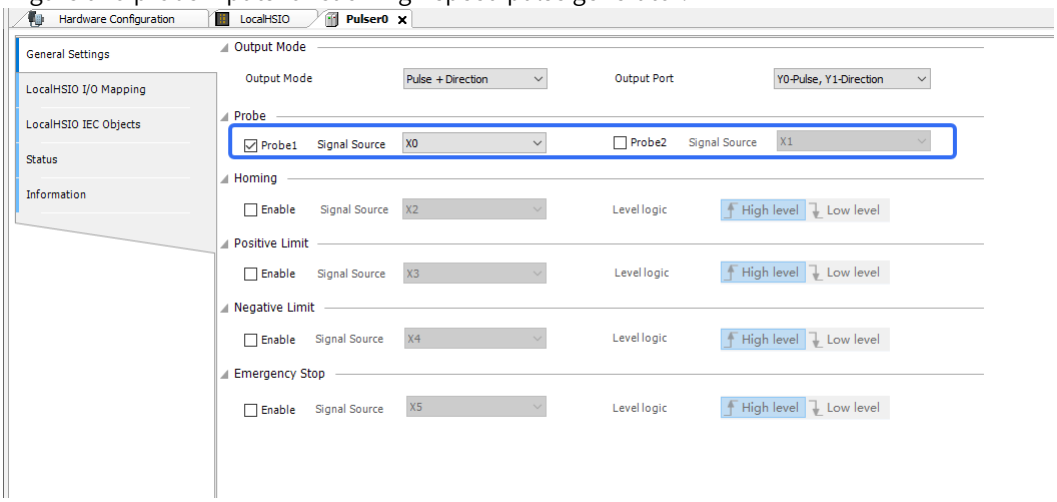
- A/B phase: Select "A/B Phase" from the drop-down list of "Output Mode". The system outputs the A/B phase signal and two hardware output ports are occupied.



- Pulse only: Select "Single Pulse" from the drop-down list of "Output Mode". The system outputs the single phase pulse signal and only one hardware output port is occupied. Only Y0/Y1 is supported.



3. Configure two probe inputs for each high-speed pulse generator.

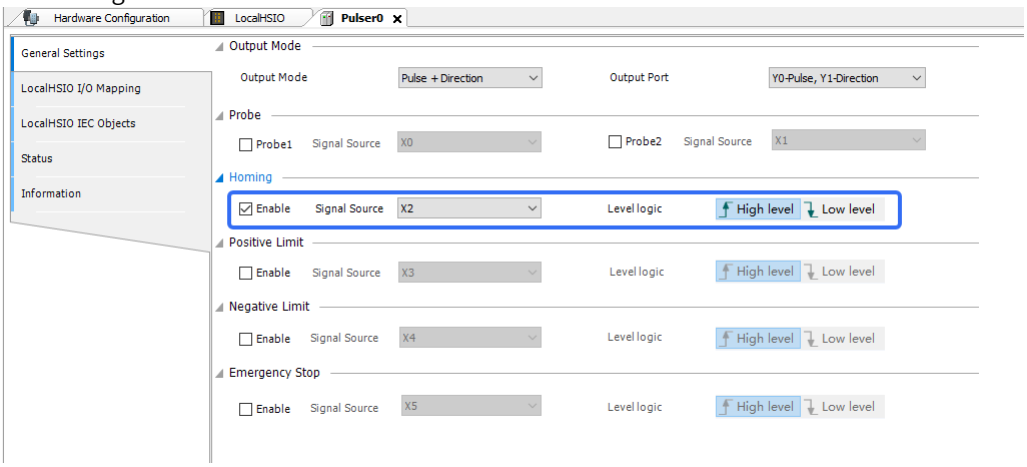


a. Check "Probe 1" or "Probe 2".

b. Set the signal source as needed.

After configuration, the position latch of the pulse axis can be realized through the MC_TouchProbe function block.

4. Set the homing function.

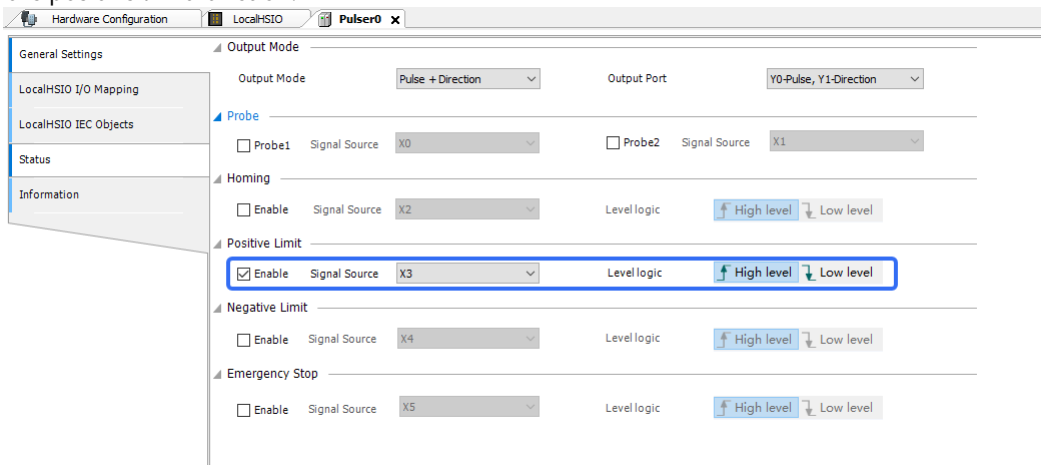


a. Check "Enable".

b. Set the signal source and level logic as needed.

After configuration, the homing function of the pulse axis can be realized through the MC_Home function block. The 17 to 30 and 35 homing modes of the CiA 402 protocol are supported.

5. Set the positive limit function.

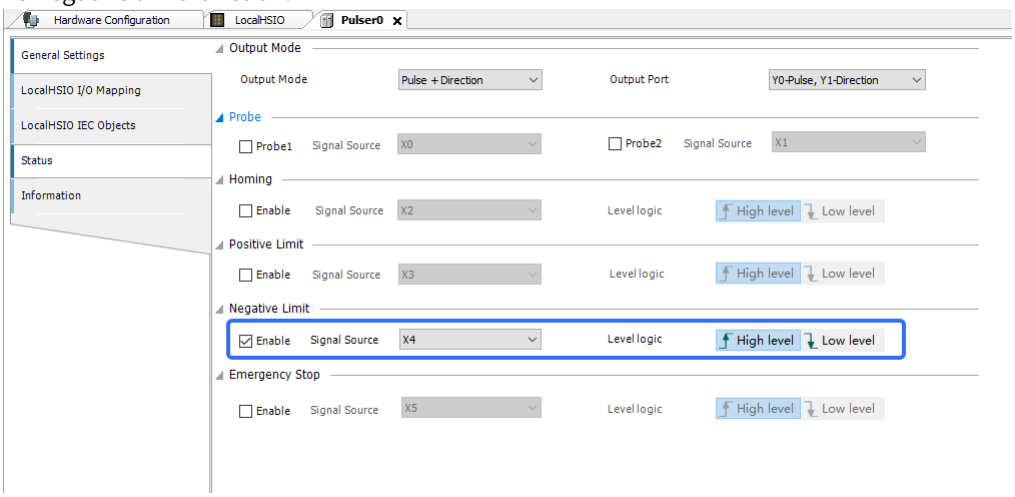


a. Check "Enable".

b. Set the signal source and level logic as needed.

After configuration, the positive limit and homing functions of the pulse axis can be realized through the MC_Home function block. The 17 to 30 and 35 homing modes of the CiA 402 protocol are supported.

6. Set the negative limit function.

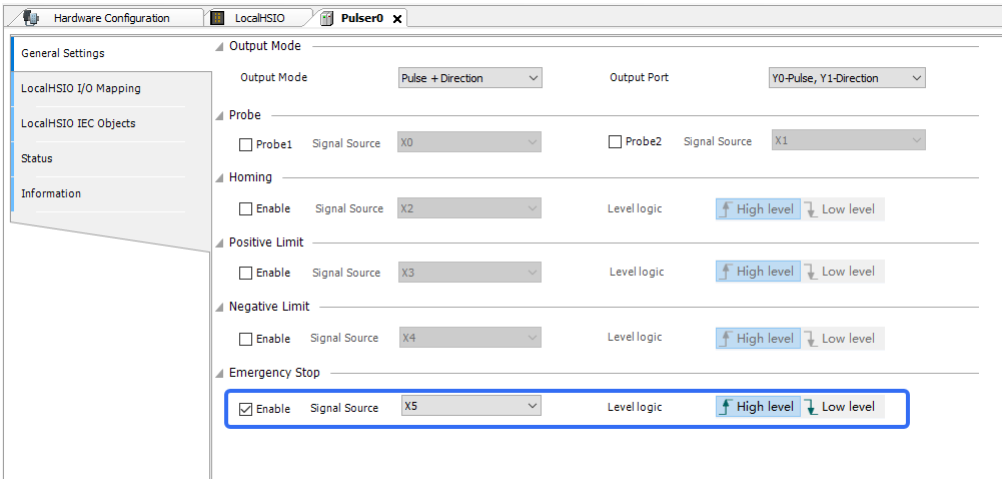


a. Check "Enable".

b. Set the signal source and level logic as needed.

After configuration, the negative limit and homing functions of the pulse axis can be realized through the MC_Home function block. The 17 to 30 and 35 homing modes of the CiA 402 protocol are supported.

7. Set the emergency stop function.



a. Check "Enable".

b. Set the signal source and level logic as needed.

After configuration, the emergency stop function of the pulse axis is triggered by the level signal of the external terminal.

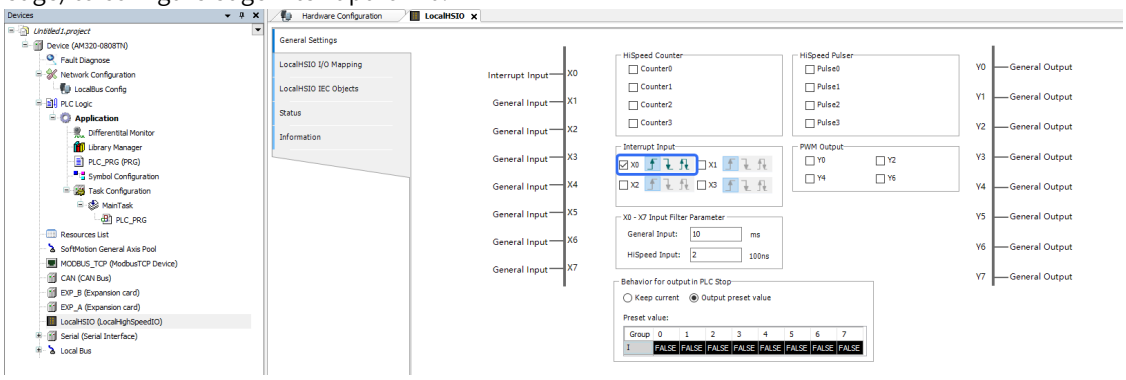
8. Set the basic settings and homing parameters of the pulse axis. For details, see *"4.4.5 CiA402 Axis" on page 168*.

Configuring High-Speed Input Interrupt

The external interrupt frequency cannot exceed 1 kHz; otherwise, interrupt loss will occur.

The following takes X0 interrupt as an example.

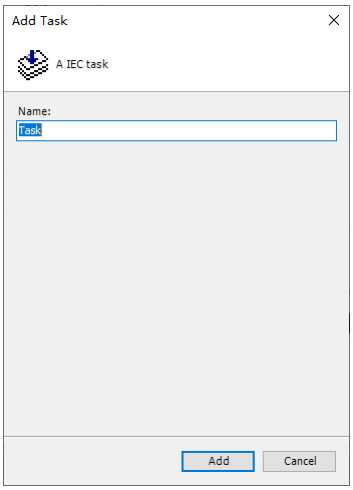
1. On the "Basic Configuration" page, check "X0" to enable X0 interrupt input and click the icon (triggered upon rising edge, triggered upon falling edge, and triggered upon rising edge+ falling edge) to configure edge interrupt for X0.



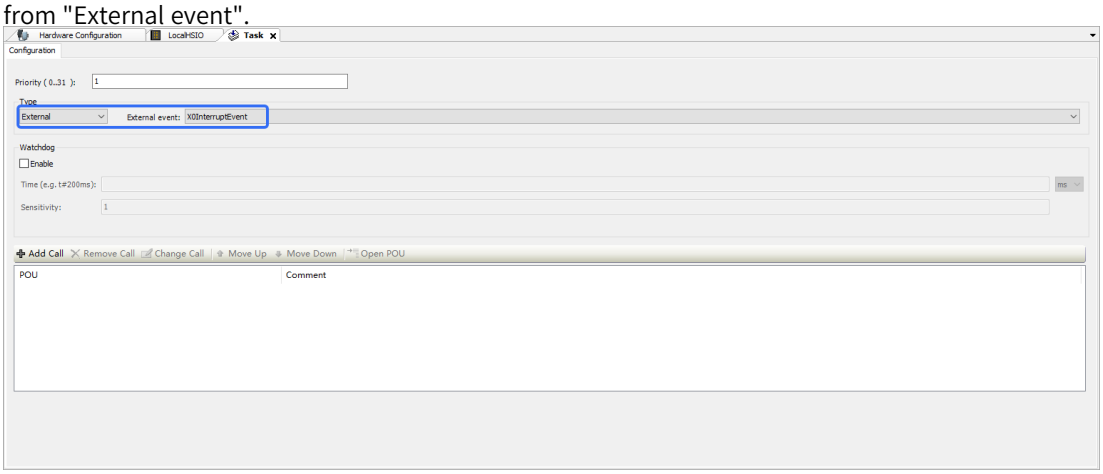
2. Configure the interrupt task.

a. In the left device tree, right-click "Task Configuration". In the shortcut menu displayed, choose "Add Object" > "Task".

b. In the dialog box displayed, click "Open".



c. On the "Task" page, select "External" from the drop-down list of "Type" and "X0InterruptEvent" from "External event".

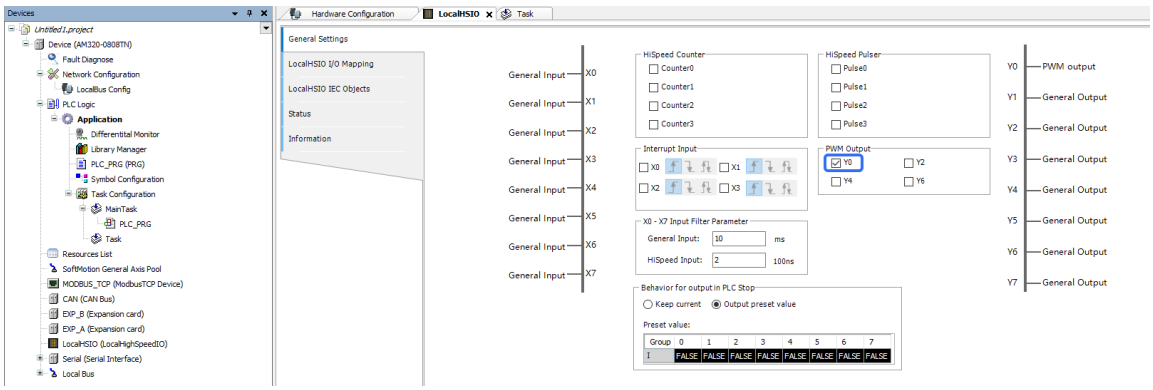


d. Call the external interrupt function block "HC_EnableInterrupt" in the main program to trigger the program execution of the interrupt task. For details about HC_EnableInterrupt, see *Medium-Sized PLC Instruction Guide*.

Configuring the PWM Output Function

The following takes the Y0 output terminal as an example.

On the "Basic Configuration" page, check "Y0" to enable the Y0 output terminal. Call the HC_PWM function block in the program to realize the output of the PWM wave.



The following table lists the PWM output specifications of the AM300-/AM500-series and AC700-series PLCs.

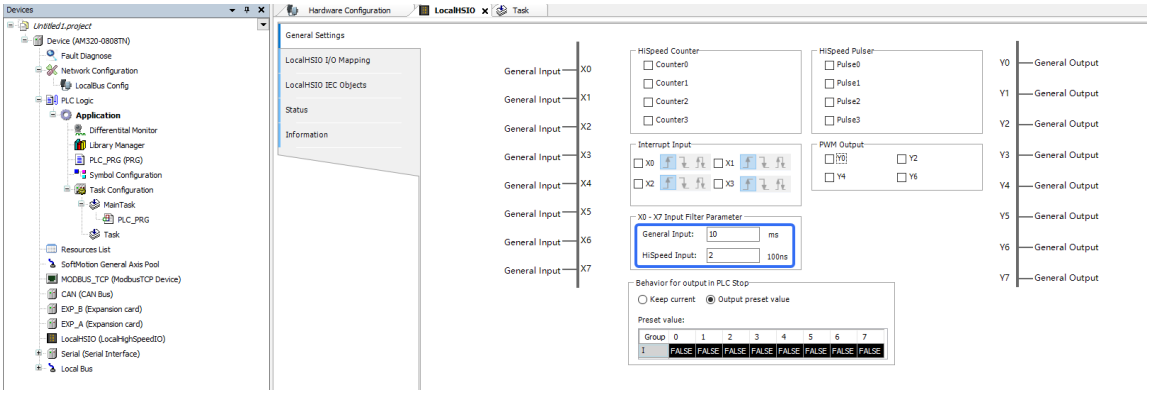
Type	AM300/500	AC700
Max. pulse period	10000000 μs	3355443 μs
Min. pulse period	0.5 μs	0.5 μs
Max. pulse width	10000000 μs	3355443 μs
Min. pulse width	0.2 μs	0.2 μs

Note

The AM300-/500-series PLCs support four PWM outputs (Y0/Y2/Y4/Y6), and the AC700-series PLC supports two PWM outputs (Y0/Y2).

Configuring Terminal Filter Parameters

On the "Basic Configuration" page, set the general input and high-speed input filter parameters as needed.



For terminals not configured in the high-speed counter signal source, general filter parameter settings are active. For terminals configured in the high-speed counter signal source, high-speed filter parameter settings are active.

The following table lists the filter parameter configuration settings of the AM300-/AM500-series and AC700-series PLCs.

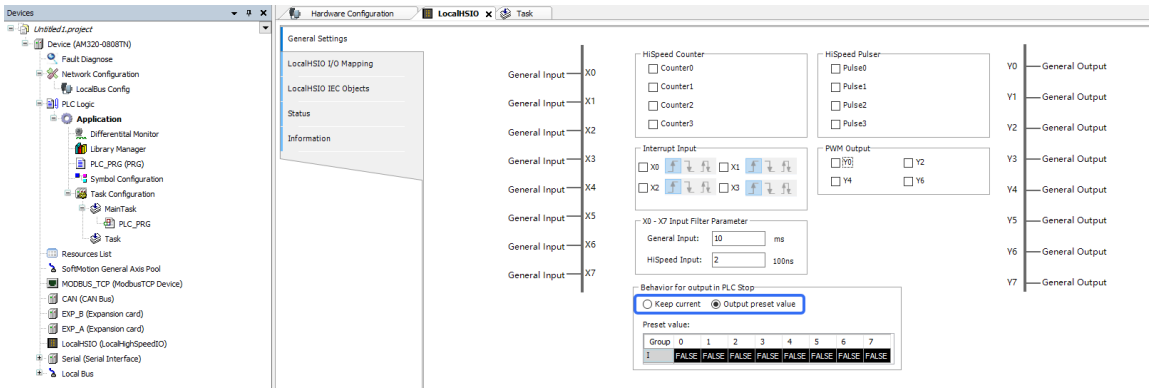
Difference	AM300/500	AC700
Limit of general input value	2 ms to 1000 ms	1 ms to 1000 ms
Unit of high-speed input values	100 ns	200 ns
Limit of high-speed input value	2 to 1000	1 to 1000

Configuring Output Status After PLC Stop

After PLC stop is enabled, the level status of the Y output terminal can be set to "Output last value" and "Output preset value".

- Output last value: The Y output terminal keeps the status upon stop after the PLC stops running.
- Output preset value: The Y output terminal keeps the preset status after the PLC stops running.

On the "Basic Configuration" page, select "Output last value" or "Output preset value" in the "PLC Output After Stop/Offline" section.

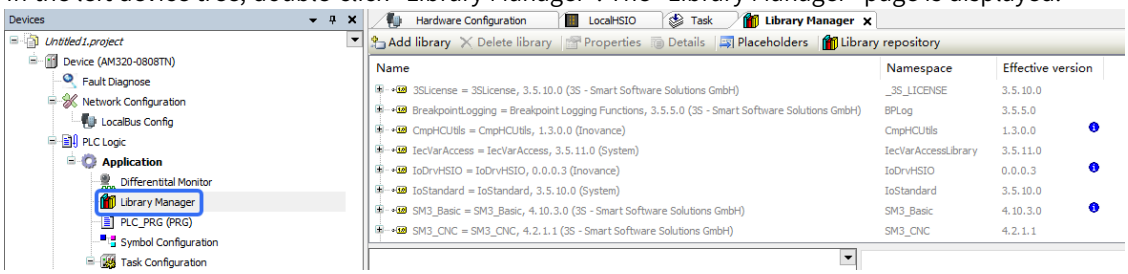


Caution

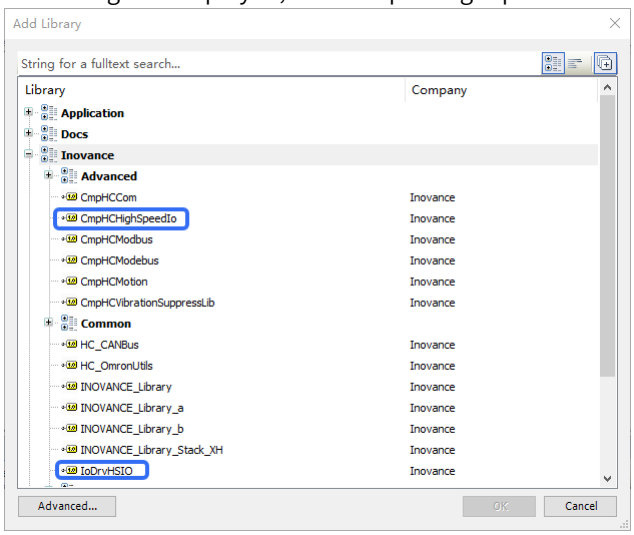
When "PLC Output After Stop/Offline" is set to "Output last value", you need to set "Behaviour for outputs in Stop" to "Keep current values" on the "PLC settings" page.

Adding a High-Speed I/O Library

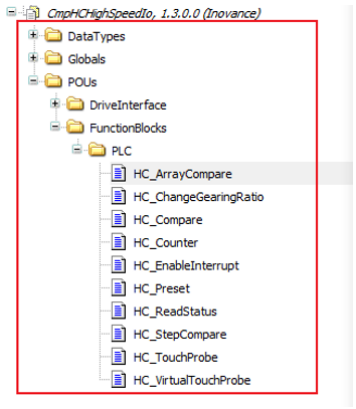
1. In the left device tree, double-click "Library Manager". The "Library Manager" page is displayed.



2. Click "Add Library". In the dialog box displayed, add "CmpHCHighSpeedIo" and "IoDrvHSIO".



Then the function blocks of the high-speed I/O library are displayed on the page.

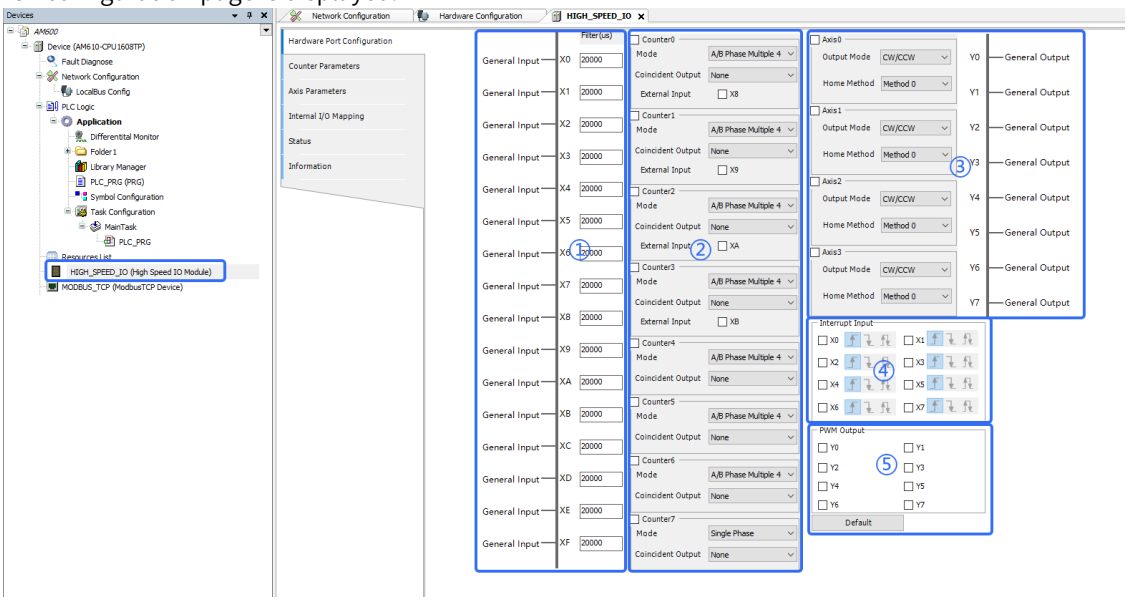


Caution

The AM300-/AM500-series and AC700-series PLCs share the same high-speed I/O library, but the AC700-series PLCs do not support pulse output and interrupt (HC_EnableInterrupt) function blocks.

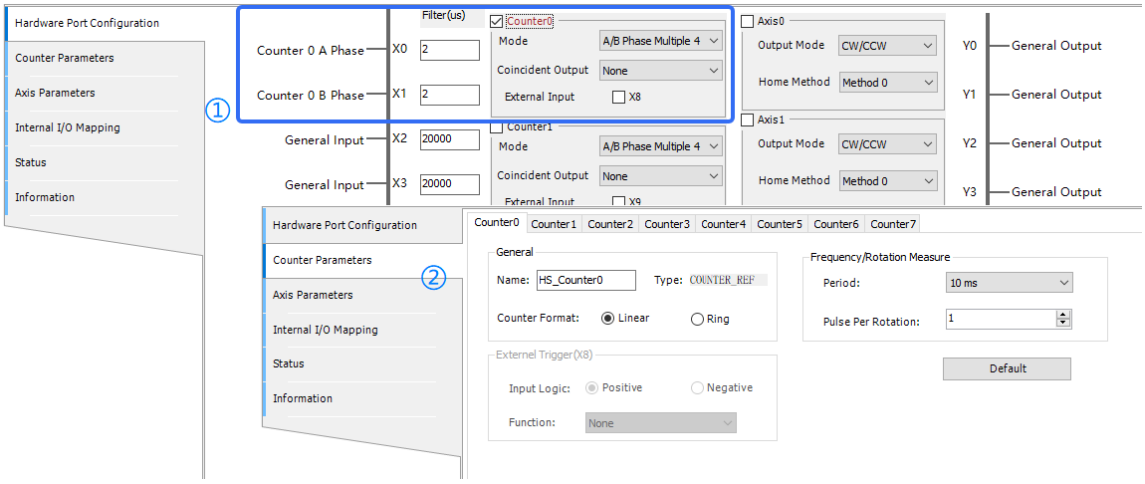
High-Speed I/O Configuration of AM400/AM600

1. For high-speed I/O wiring guide of AM400/AM600, see *"9.6 AM400 or AM600 High-Speed I/O Wiring" on page 497*.
2. In the left device tree, double-click "HIGH_SPEED_IO (high-speed I/O module)". The "HIGH_SPEED_IO" configuration page is displayed.



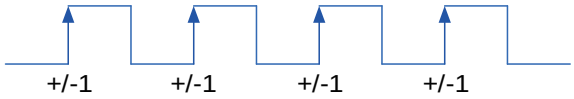
No.	Function Configuration
①	Terminal filter parameter configuration
②	High-speed counter configuration
③	High-speed pulse output configuration
④	High-speed input edge interrupt configuration
⑤	PWM output configuration

Configuring High-Speed Counter Function

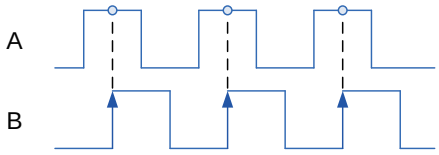


1. High-speed counter parameters include the count mode, comparison consistence output, and external triggering input. Count modes include single-phase counter, A/B phase, CW/CCW, and internal clock. Taking the "Counter 0" (counter numbers range from 0 to 7) as an example, the configuration procedure is as follows:

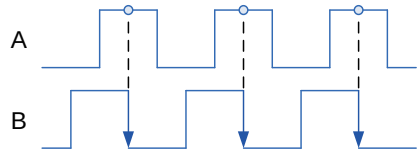
   - Check "Counter 0".
   - Set the high-speed counter mode:
     Single-phase counter: The system receives pulse signals from external single-phase encoders. Only the hardware port X0 is occupied.



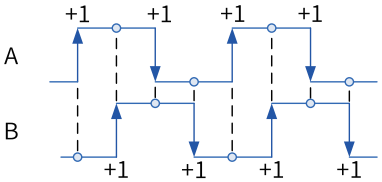
     A/B phase counter: The system receives pulse signals from external A/B phase encoders. The quadruple frequency can be configured for the A/B phase. In this case, two hardware ports X0 and X1 are occupied.



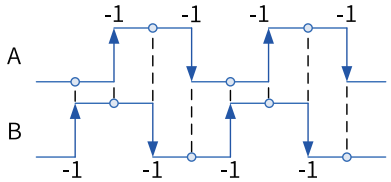
Phase A leading phase B, incremental count



Phase B leading phase A, decremental count

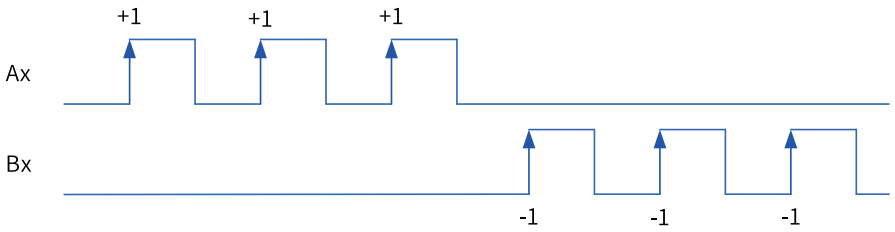


Quadruple incremental count



Quadruple decremental count

     CW/CCW counter: The system receives pulse signals from external CW/CCW encoders. Hardware ports X0 and X1 are occupied.



CW/CCW count

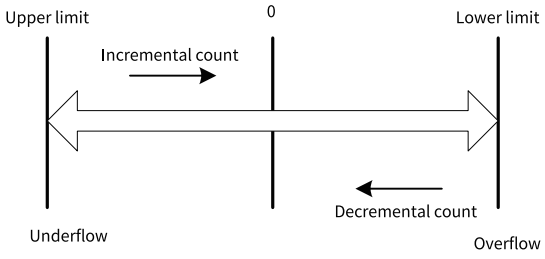
Internal clock: The system uses the pulse signals of the high-speed counter 0 regularly generated in the software at an interval of 1 μs, 10 μs, 100 μs, or 1 ms.

- Coincident Output: When the counter of high-speed I/O reaches the specified value, the corresponding hardware output port will output the matching level signals. When this function is enabled, the HC_EnableInterrupt and HC_ SetCompare/HC_SetCompareM functions need to be called.
- External Trigger: When the external trigger input pin is enabled, the high-speed counter value latch and pulse width measurement functions are available. The counter latch function needs to call HC_TouchProbe and the pulse width measurement function needs to call HC_ MeasurePulseWidth.
- Filter Time: Sets the filter time of the high-speed counter port. The default value is 2 μs.

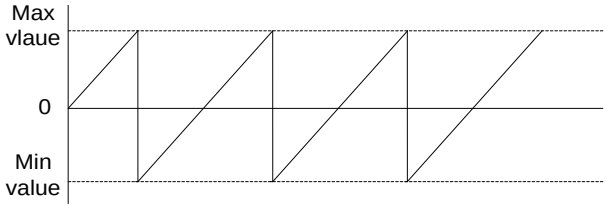
2. Create an instance for the high-speed counter, with the data type COUNTER_REF. Take Counter 0 as an example:
   The default name of Counter 0 instance is HS_Counter0.

   Counter mode:

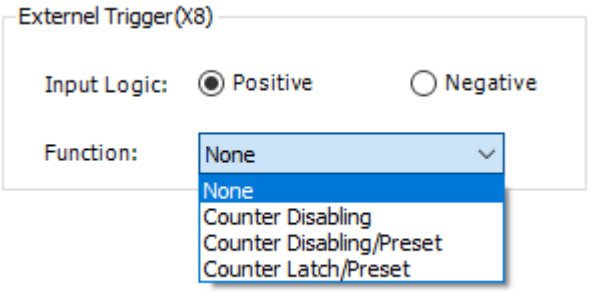
   - Linear counter: Count between the maximum and minimum values. The counter stops when the count-up reaches the maximum or the count-down reaches the minimum, and the overflow sign takes effect.



   - Ring count: Used together with "HC_SetRing".
     Count between the maximum and minimum values. When the count-up exceeds the maximum, the value skips to the minimum. When the count-down exceeds the minimum, the value skips to the maximum.

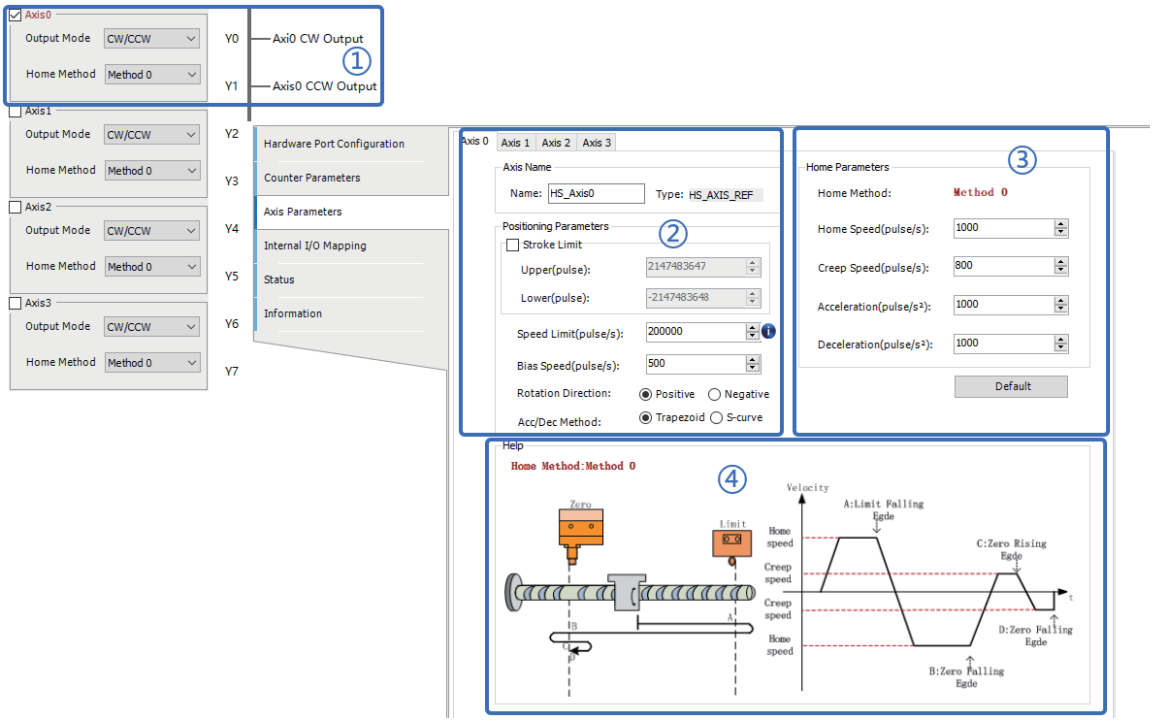


3. Select the external trigger.



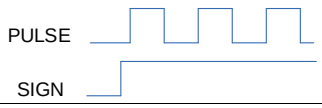
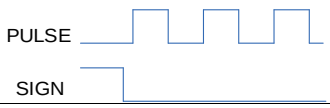
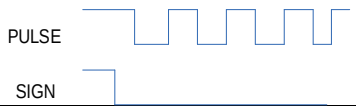
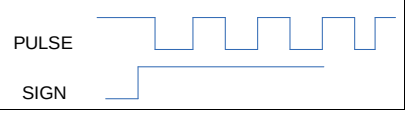
When the input level of X8 is valid, you can set the counter function. For example, configure X8 as the signal of disabling Counter 0. You can also disable the preset and counter latch functions of Counter 0.

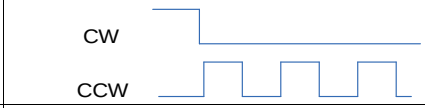
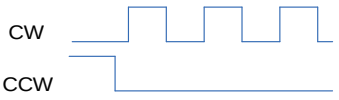
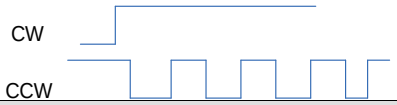
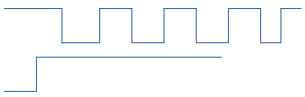
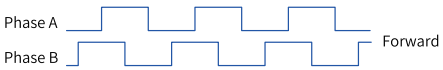
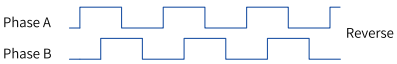
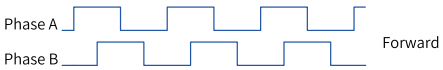
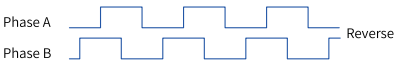
Configuring the High-Speed Pulse Output Function



1. Configure the high-speed pulse output function, including the output pulse mode (pulse+direction, CW/CCW, and A/B phase) and homing mode.
   Take "Axis 0" as an example (axis numbers range from 0 to 3):

   - Check "Axis 0".
   - Configure the high-speed pulse output mode:

Pulse Reference Form	Pulse+Direction	
	Forward rotation	Reverse rotation
Positive logic		
Negative logic		

Pulse Reference Form	CW/CCW	
	Forward rotation	Reverse rotation
Positive logic		
Negative logic		
Pulse Reference Form	Phase A/B	
	Forward rotation	Reverse rotation
Positive logic		
Negative logic		

- Homing mode
  Methods 0 to 3 are supported. For details, see the homing diagram.

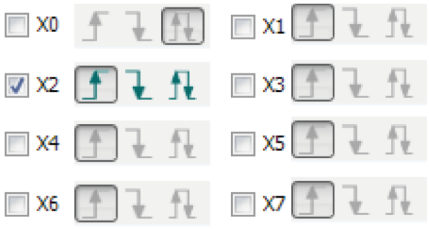
2. Create an instance of the axis, with data type HS_AXIS_REF. Take "Axis 0" as an example:

- The default name of the Axis 0 instance is HS_Axis0.
- Stroke Limit: soft limit
- Speed Limit: limits the maximum speed.
- Bias Speed: baseline speed when pulse is started
- Acc Method: trapezoid and S-curve

3. Configure the homing parameters, including the homing speed and creep speed.
   This step must be used together with the function block "MC_Home_P".

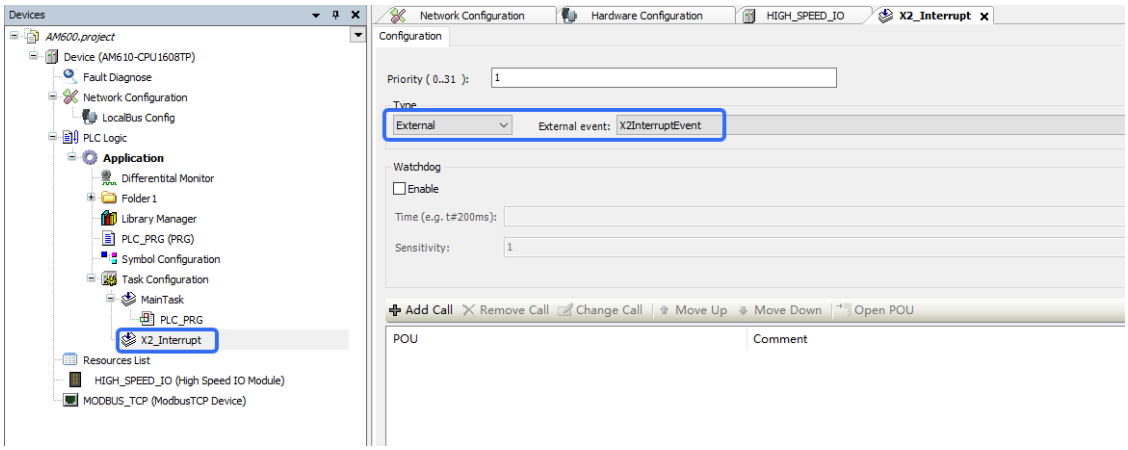
4. Homing method diagram

Configuring High-Speed Input Interrupt



1. Check high-speed input interrupt X2.
   Select X2 edge interrupt: raising edge, falling edge, and both raising and falling edges.

2. Configure the interrupt task.



Launch the "HC_EnableInterrupt" function to add the X2 interrupt task. When the raising edge signal of X2 takes effect, the program in the interrupt task starts to run.

Adding a High-Speed I/O Library

1. In the left device tree, double-click "Library Manager". The "Library Manager" page is displayed.
2. Click "Add Library". In the dialog box displayed, unfold "Misc", select "CmpHSIO", and click "OK". The new high-speed I/O library is added.
3. In the library list, click the new high-speed I/O library, and unfold "HSIO". The function blocks of the new high-speed I/O library are displayed.

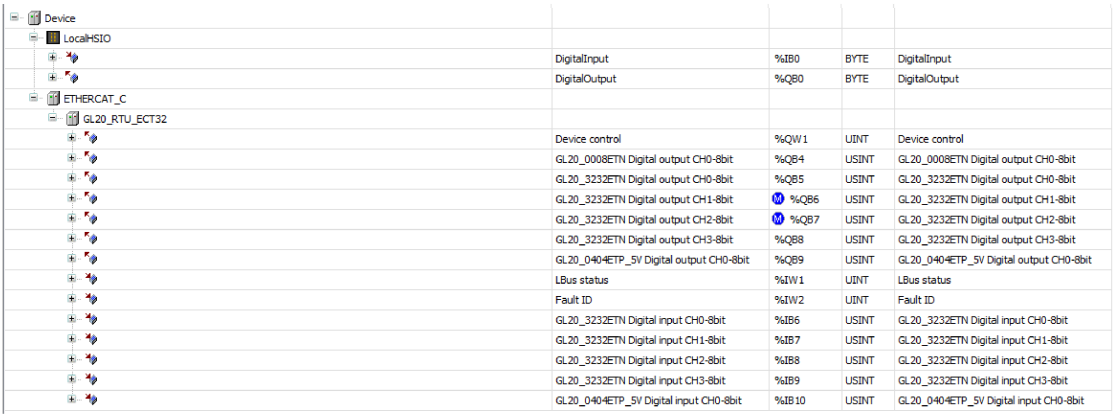
High-Speed I/O Diagnosis

For details, see *"9.7.2 High-Speed I/O Diagnosis" on page 504*.

4.2.6 I/O Mapping Parameters

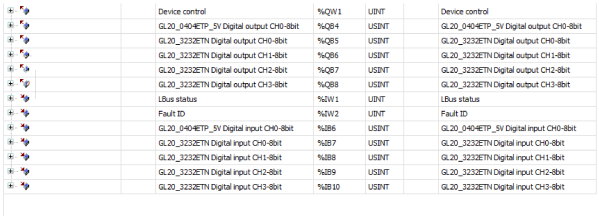
You can access the I/O mapping configuration page in two ways:

- In the left device tree, right-click a device. In the shortcut menu displayed, select "Edit IO mapping". On the page displayed, the I/O mapping parameters of the current device and all its sub-devices are displayed.



Device				
LocalHSIO				
	DigitalInput	%IB0	BYTE	DigitalInput
	DigitalOutput	%QB0	BYTE	DigitalOutput
ETHERCAT_C				
GL20_RTU_ECT32				
	Device control	%QW1	UINT	Device control
	GL20_0008ETN Digital output CH0-8bit	%QB4	USINT	GL20_0008ETN Digital output CH0-8bit
	GL20_3232ETN Digital output CH0-8bit	%QB5	USINT	GL20_3232ETN Digital output CH0-8bit
	GL20_3232ETN Digital output CH1-8bit	%QB6	USINT	GL20_3232ETN Digital output CH1-8bit
	GL20_3232ETN Digital output CH2-8bit	%QB7	USINT	GL20_3232ETN Digital output CH2-8bit
	GL20_3232ETN Digital output CH3-8bit	%QB8	USINT	GL20_3232ETN Digital output CH3-8bit
	GL20_0404ETP_5V Digital output CH0-8bit	%QB9	USINT	GL20_0404ETP_5V Digital output CH0-8bit
	LBus status	%IW1	UINT	LBus status
	Fault ID	%IW2	UINT	Fault ID
	GL20_3232ETN Digital input CH0-8bit	%IB6	USINT	GL20_3232ETN Digital input CH0-8bit
	GL20_3232ETN Digital input CH1-8bit	%IB7	USINT	GL20_3232ETN Digital input CH1-8bit
	GL20_3232ETN Digital input CH2-8bit	%IB8	USINT	GL20_3232ETN Digital input CH2-8bit
	GL20_3232ETN Digital input CH3-8bit	%IB9	USINT	GL20_3232ETN Digital input CH3-8bit
	GL20_0404ETP_5V Digital input CH0-8bit	%IB10	USINT	GL20_0404ETP_5V Digital input CH0-8bit

- In the left device tree, double-click a communication interface module. On the page displayed, click the "xx I/O mapping" tab. On the tab page displayed, the I/O mapping parameters of the current device and all its sub-devices are displayed.



	Device control	%QW1	UINT	Device control
	GL20_0404ETP_5V Digital output CH0-8bit	%QB4	USINT	GL20_0404ETP_5V Digital output CH0-8bit
	GL20_3232ETN Digital output CH0-8bit	%QB5	USINT	GL20_3232ETN Digital output CH0-8bit
	GL20_3232ETN Digital output CH1-8bit	%QB6	USINT	GL20_3232ETN Digital output CH1-8bit
	GL20_3232ETN Digital output CH2-8bit	%QB7	USINT	GL20_3232ETN Digital output CH2-8bit
	GL20_3232ETN Digital output CH3-8bit	%QB8	USINT	GL20_3232ETN Digital output CH3-8bit
	LBus status	%IW1	UINT	LBus status
	Fault ID	%IW2	UINT	Fault ID
	GL20_0404ETP_5V Digital input CH0-8bit	%IB6	USINT	GL20_0404ETP_5V Digital input CH0-8bit
	GL20_3232ETN Digital input CH0-8bit	%IB7	USINT	GL20_3232ETN Digital input CH0-8bit
	GL20_3232ETN Digital input CH1-8bit	%IB8	USINT	GL20_3232ETN Digital input CH1-8bit
	GL20_3232ETN Digital input CH2-8bit	%IB9	USINT	GL20_3232ETN Digital input CH2-8bit
	GL20_3232ETN Digital input CH3-8bit	%IB10	USINT	GL20_3232ETN Digital input CH3-8bit

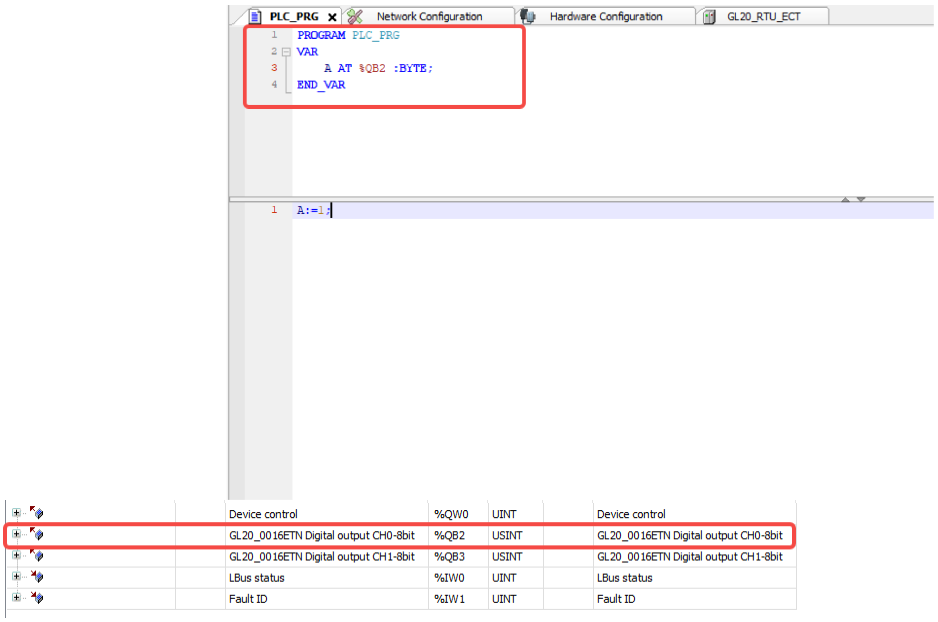
The following table lists parameters displayed on the page.

Parameter	Description
Find	Searches for the I/O mapping channels matching the keywords.
Filter	Displays variables matching the filter rules.
Set Continuous Address	Re-allocates the I/O mapping addresses of the device as continuous addresses (available only in the timing address allocation mode).
Variable	• Map the I/O mapping channel to the current variable: Double-click an item in the "Variable" column, and then click "...". On the "Input Assistant" page displayed, select an existing variable to be mapped and then click "OK". If the variable does not exist, a compiling error will be reported. • Map the I/O mapping channel to a new variable: Double-click an item in the "Variable" column, input the name of the new variable in the text box, and then press "Enter". The name of the new variable must meet the IEC programming specification; otherwise, a compiling error will be report.
Mapping	Click the icon to switch the mapping mode of the I/O mapping channel between "Create new variable" and "Map to existing variable". • : Indicates creating a new variable. • : Indicates mapping to the existing variable.
Channel	The name of the current I/O mapping channel.

Parameter	Description
Address	The address allocated to the current I/O mapping variable. To specify a user-defined I/O address, modify the address in the corresponding cell of "Address". Address allocation modes include timing address allocation (default) and sequential address allocation. • Timing address allocation: Addresses for I/O mapping variables are allocated based on the addition order of devices. • Sequential address allocation: Addresses for I/O mapping variables are allocated based on the position of devices in the device tree. Set the address allocation mode: In the left device tree, double-click a PLC device such as "Device(AC702)". On the page displayed, click "PLC settings". On the page displayed, set "sequentially allocate I/O addresses" in the "Additional settings" section. • When this item is selected, the sequential address allocation mode is used. • When this item is de-selected, the timing address allocation mode is used.
Type	The data type of the current I/O mapping variable.
Unit	The unit of the current I/O mapping variable.
Description	The description of the current I/O mapping variable.
Default	The default value of the current I/O mapping variable. **Note**: The default value is displayed only when "Behaviour for outputs in Stop" on the "PLC settings" is set to "Set all outputs to default".
Current Value	The actual value of the current I/O mapping variable during program running. **Note**: This parameter is displayed only when the project is in online state.
Prepared Value	The value to be written to the current I/O mapping variable. Double-click a cell in the "Prepared Value" column to edit the prepared value. Then, you can press "Ctrl+F7" or use the menu command to write the prepared value to the I/O mapping variable. **Note**: This parameter is displayed only when the project is in online state.
Reset All Mapping Var	Set the value of all mapping variables in the current I/O mapping table to a null value. **Note**: This operation only resets the values of mapping variables, but does not reset default values and addresses of mapping variables.
Always update variables	Set the task execution mode of the I/O mapping variable. [1] • Use parent device setting: I/O mapping cycle is not performed during system running. • Enabled 1 (use bus cycle task if not used in any task): Select tasks having used the address of the current device first. If no such task exists, use the bus task of the current device. • Enabled 2 (always in bus cycle task): Use the bus task of the current device.

[1]: Task execution mode example of the I/O mapping variable

The address "%QB2" is used in the POU "PLC_PRG" and allocated to a module under the GL20 communication interface module that is an EtherCat slave, as shown in the following figure.



Two tasks are available under "Task Configuration": ETHERCAT_C and MainTask. The ETHERCAT_C task executes the ETHERCAT bus cycle, while the MainTask task executes PLC_PRG.

- When "Always update variables" of the communication interface module is set to "Use parent device setting", no POU with data refreshing will be generated and the I/O data remains unchanged during project running.
- When "Always update variables" of the communication interface module is set to "Enabled 1 (use bus cycle task if not used in any task)", a POU with data refreshing is generated and this POU is executed in the MainTask task.
- When "Always update variables" of the communication interface module is set to "Enabled 2 (always in bus cycle task)", a POU with data refreshing is generated and this POU is executed in the EtherCat_C task.
- If the address is not used in PLC_PRG, the POU with data refreshing generated based on the I/O configuration (no matter it is set to "Enabled 1" or "Enabled 2") will be executed in the EtherCat_C task.
- The task in which the I/O configuration is executed is not strongly related to the task name. Instead, it is related to the task depended by the bus cycle of the current device.

Description

- Cross references: After the I/O mapping channel is associated with the variable, you can double-click the variable cell, and then choose "Browse" > "Browse Cross References" to view the cross references of the current variable in the program.
- Address allocation: After an address is allocated to a device, this address is a resource of the device, no matter whether it is mapped to a variable. Therefore, do not use the address allocated to the device in the POU. If you do need to use this address of the device, map the address to a variable to implement relevant functions.
- Mode switchover: Switchover from sequential mode to timing mode does not change the device address allocation. However, switchover from timing mode to sequential mode causes re-allocation of addresses based on the position of devices in the device tree.
- Sequential mode: In this mode, manual addresses are prioritized over automatic addresses. When a manual address conflicts with an automatic address, the automatic address is re-allocated first.

- Timing mode: In this mode, manual addresses have the same priority as automatic addresses. When a manual address conflicts with an automatic address, the system prompts the conflict and does not support address allocation.
- When a single I/O mapping channel (Device Control) is allocated to a specified address, the address of this I/O mapping channel can be manually allocated. Bit0 to Bit15 are sub-elements of the I/O mapping channel. Addresses of sub-elements cannot be manually allocated and are determined by the address of the I/O mapping channel.

4.3 Expansion Card Configuration

Caution

Only the AM300-/AM500-series PLCs support expansion card configuration. Skip this section for PLCs of other series.

For how to configure expansion cards for the AM300/AM500 series PLCs, see the *user guide* of the corresponding expansion card.

4.4 EtherCAT Configuration

4.4.1 Overview

EtherCAT is an open industrial field technology over the Ethernet. It features short communication update interval, low synchronization jitter, and low hardware cost. EtherCAT supports the linear, tree, star, and hybrid topologies. EtherCAT slaves must use dedicated communication chipset (ESC), and EtherCAT masters can use a standard Ethernet controller.

For details about EtherCAT principles and related technologies, see the book "Industrial Ethernet Fieldbus EtherCAT Driver Design and Applications" or visit the official website of the EtherCAT Technical Group at https://www.EtherCAT.org.cn.

4.4.2 Common Functions

Installing a Device

EtherCAT device installation is to import the device description XML file in compliance with EtherCAT Technology Group (ETG) standards into the programming software InoProShop. After the software parses and processes the file, it generates the EtherCAT configuration devices that can be added and deleted by users. InoProShop integrates all EtherCAT slaves of Inovance, and you do not need to install them. If you need to use third-party EtherCAT devices, install the device description files provided by the third-party vendors.

Two installation methods are available: installation on the "Network Configuration" page and installation by using the menu bar.

- Installation on the "Network Configuration" page

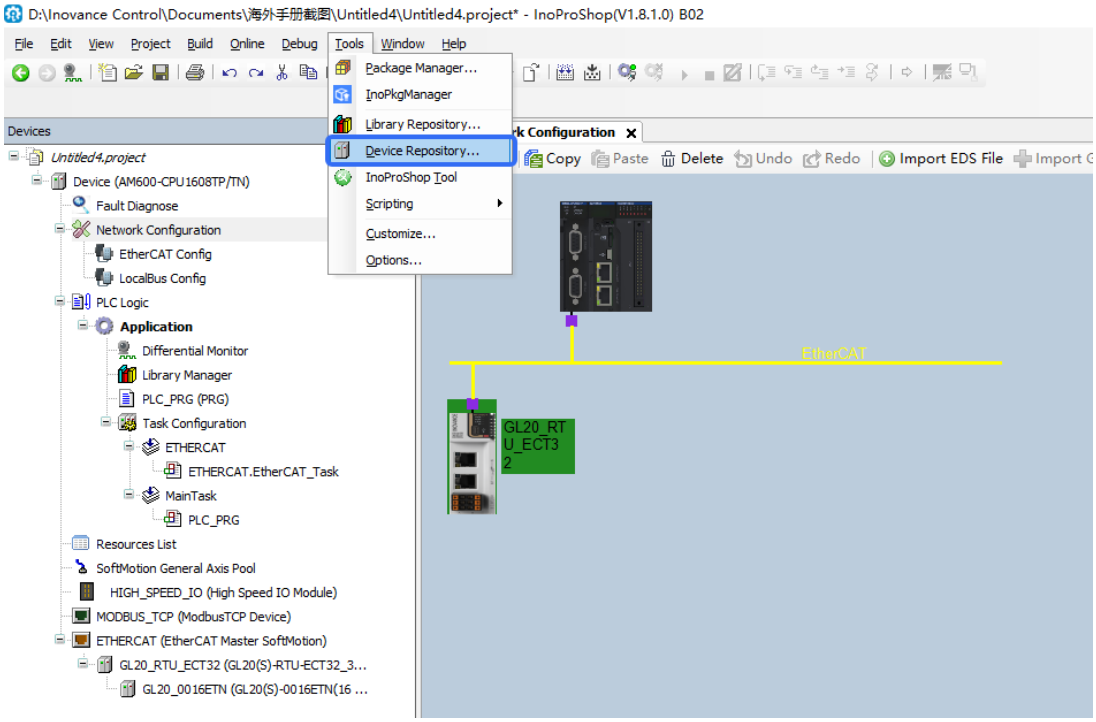
  1. On the "Network Configuration" page, click "Import ECT File".



  2. In the dialog box displayed, select the device XML file, and then click "Open".

- Installation by using the menu bar

  1. In the menu bar, choose "Tools" > "Device Repository".

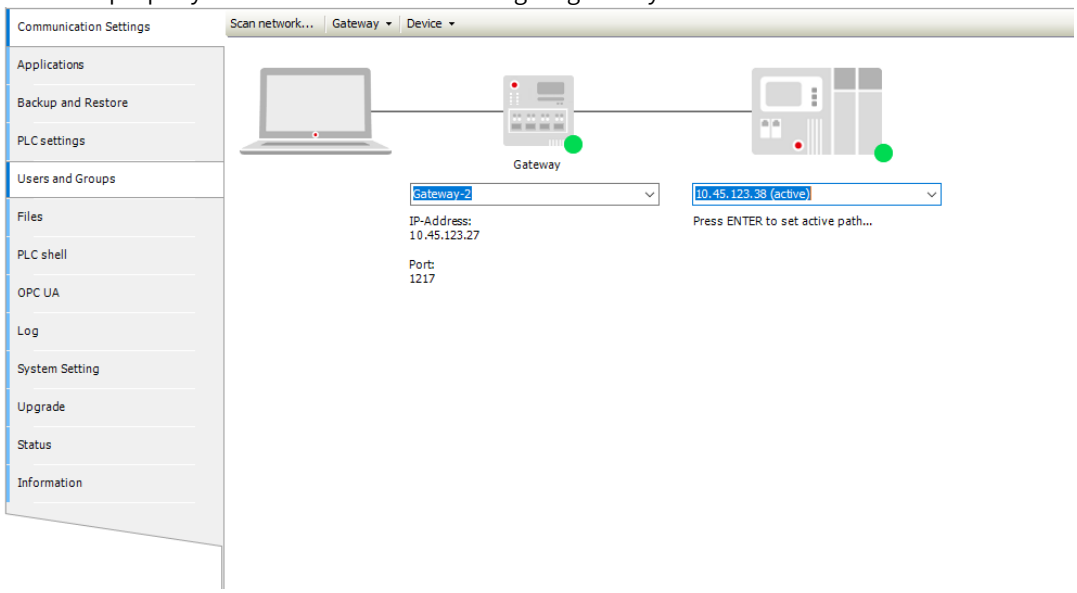


  2. In the dialog box displayed, click "Install (I)".
  3. In the dialog box displayed, select "EtherCAT XML Device Description Configuration Files", locate the description file of the slave saved in the local PC, and click "Open".

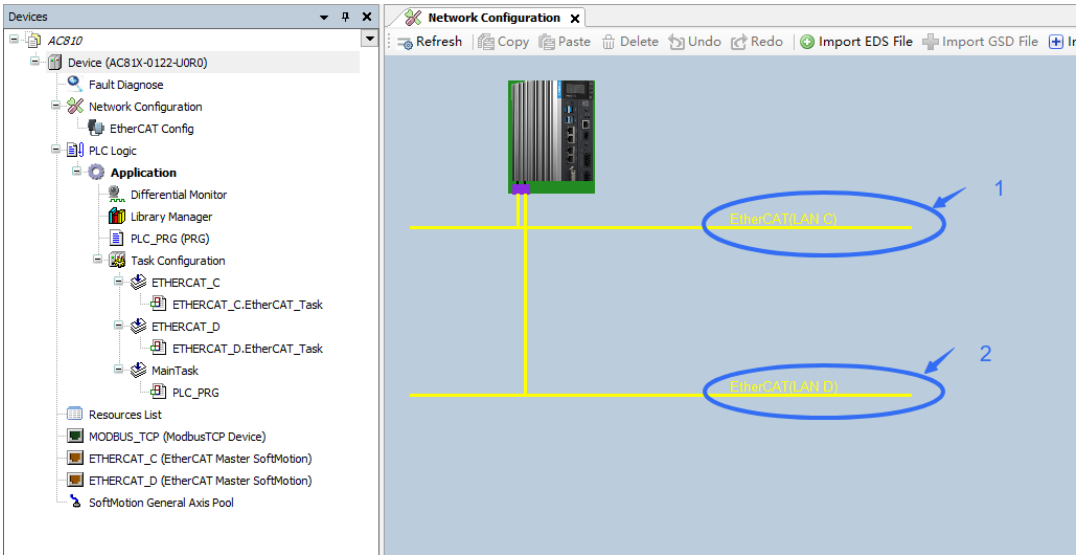
Scanning a Device

The scanning function is recommended. The procedure is hot reset > logout > device scanning.

- **Prerequisites**
  - The PC is properly connected to the PLC through a gateway.



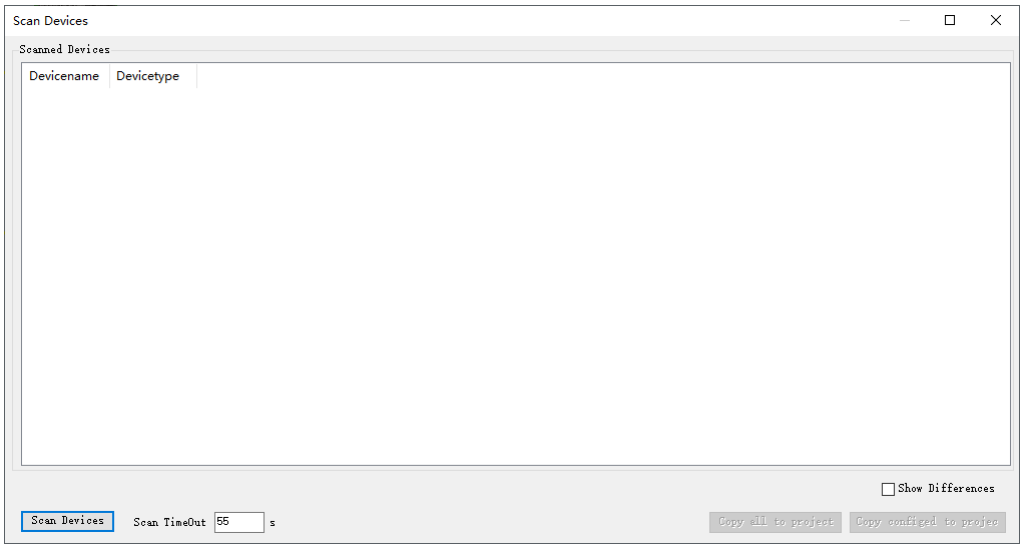
  - The PLC is networked with the EtherCAT slave.
  - The port configuration of the programming software configuration is consistent with that of the physically connected port of the PLC.
    To keep configuration consistency, download the port configuration information first before using the scan function.



  - The PLC stops.

- **Procedure**

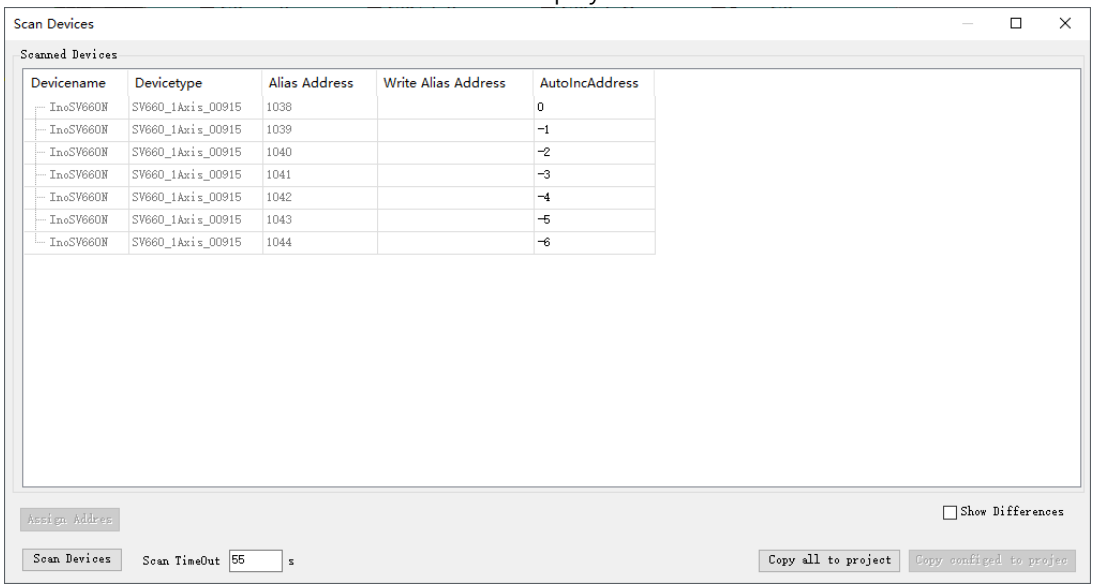
1. In the left device tree, right-click "EtherCAT_***X*** (EtherCAT Master SoftMotion)". In the shortcut menu displayed, select "Scan Devices". The "Scan Devices" dialog box is displayed.



Note

Scan TimeOut: Indicates the maximum timeout time of one scan operation. When no device is detected, increase the timeout time. The minimum value is 20s.

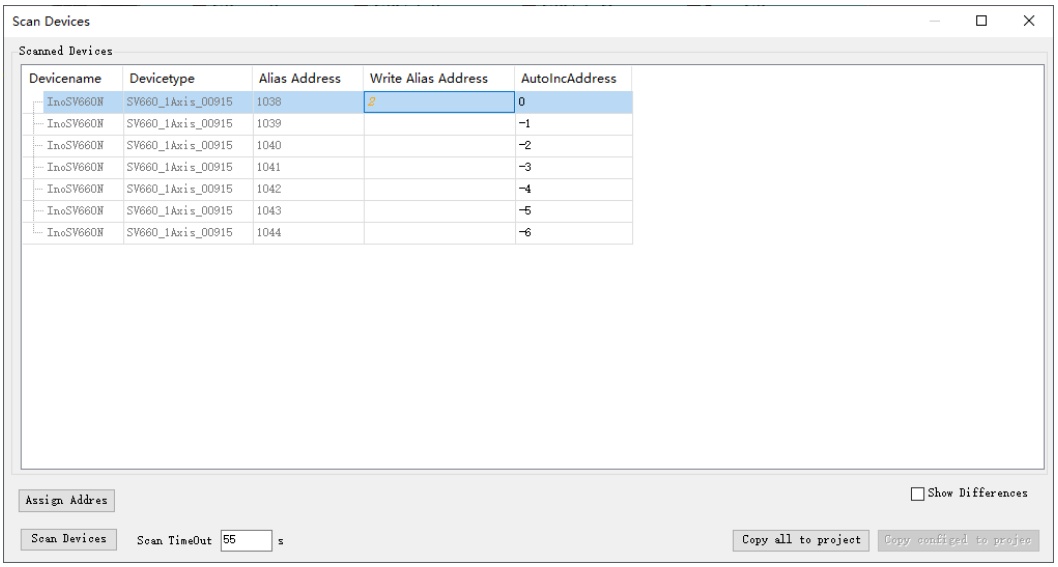
2. Click "Scan Devices". The scanned devices are displayed in the device list.



3. Click "Copy all to project" to add these devices to the device tree and configuration.

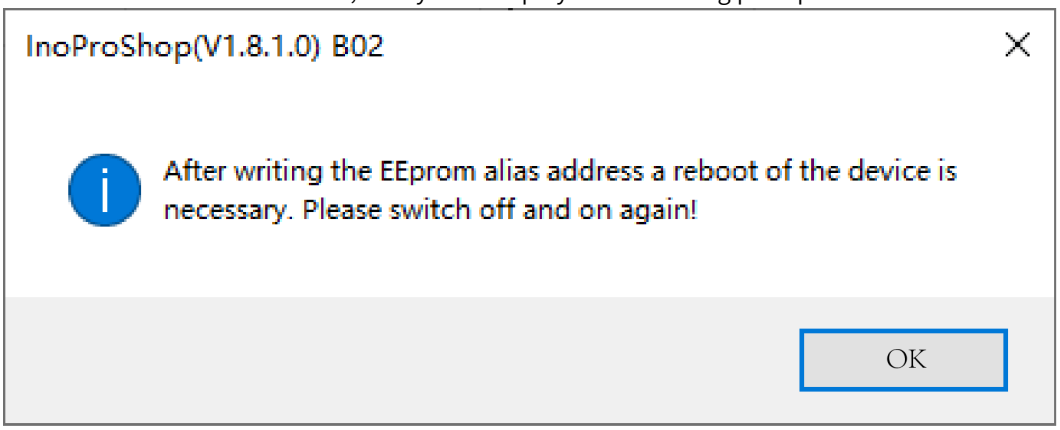
- **Scanning operations**
  - Allocate an alias address
    1. In the "Write Alias Address" column, double-click the alias address you want to modify and input a new alias address.

       The new alias address is displayed in a color font.



2. Click "Assign Address" to make the new alias address take effect.

   If the modification is successful, the system displays the following prompt:



- Display the project differences

  Select "Show Differences". The differences between the configuration device and scanned device are displayed, as shown in the following figure.

  - When the current configuration device has no alias address, the system compares the device types only, but does not compare the alias addresses. If the device types are matching, the device names are displayed in green.
  - When the current configuration device has an alias address, the system compares the device types and alias addresses. When both the device types and alias addresses are consistent, the system determines that the two devices are one device; otherwise, the device names are displayed in red.

  The following table lists each function in the preceding table and its operation.

No.	Function	Operation
①	CopyBefore	Select the scanned device on the left and the current device on the right, and click . The scanned device is inserted in front of the selected device on the right. This command can also be used between modules.
②	CopyAfter	Select the scanned device on the left and the current device on the right, and click . The scanned device is inserted behind the selected device on the right. This command can also be used between modules.
③	Replace	Select the scanned device on the left and the current device on the right (the left device and the right device must be of the same type), and select . The right device is replaced with the scanned device.
④	CopyAll	Click . Then all the devices on the right are cleared, and all the scanned devices on the left are copied to the right side.
⑤	Delete	Select a device on the right, and then click to delete it. This command can also be used on modules.

- Check whether the device alias takes effect after a scanned device is copied
  After "Copy all to project" is clicked, the system sets the alias device of the scanned device based on the slave communication mode set on the EtherCAT overview page.
  - If the slave communication mode set on the EtherCAT overview page is "Classic mode" or "Sequential model", the "sequential model" is applied to the scanned device and the device generated based on the project configuration by default, and the alias address of the scanned device is saved to the device data but does not take effect.
    In the following figure, the slave communication mode on the EtherCAT is set to "Classic mode", the type of the scanned device is the same as that of the configured device, but their alias addresses are different.

    After "Copy all to project" is clicked, the system creates a device object based on the type of the scanned device and saves its alias address. However, the communication mode of this device is still "Sequential model" and its alias address does not take effect.

    After the communication mode is switched to "Alias mode", the device alias address is the scanned one.
  - If the slave communication mode set on the EtherCAT overview page is "Alias mode", the "Alias mode" is applied to the scanned device and the device generated based on the project configuration by default, and the alias address of the scanned device is saved to the device data and takes effect immediately.
    In the following figure, the slave communication mode on the EtherCAT is set to "Alias mode", the type of the scanned device is the same as that of the configured device, but their alias addresses are different.

    After "Copy all to project" is clicked, the system creates a device object based on the type of the scanned device. The device communication mode is "Alias mode" and the alias address of the generated new device object is the scanned alias address.

- **Errors**
  When the port information in the programming software configuration is different from the information of the port connected to the PLC, the following prompt is displayed.

In this case, download the port information in the programming software again.

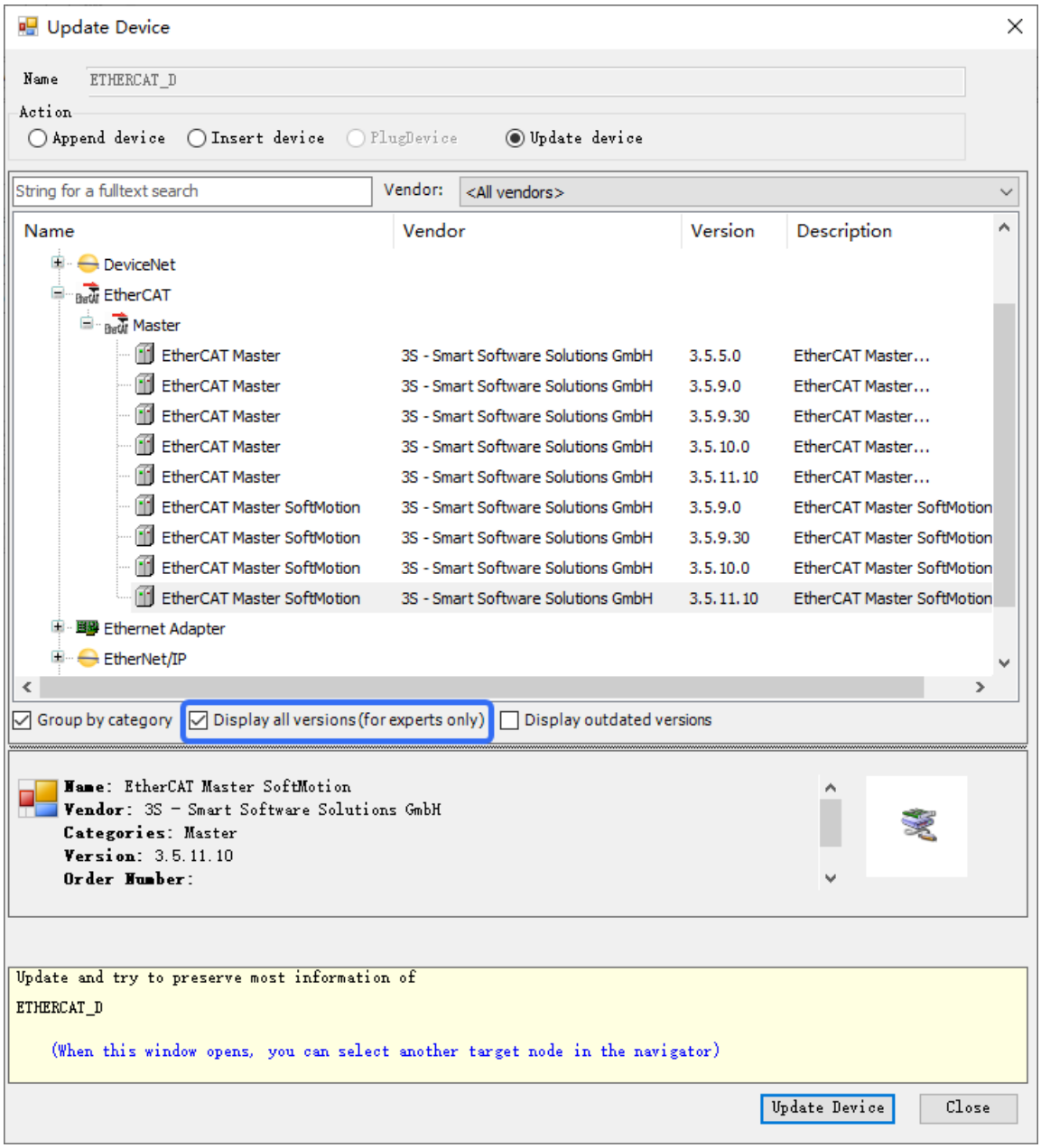
For example, the port configured in the programming software configuration is "EtherCAT_C", but the actual MAC address is "EtherCAT_D", the device of port D is displayed while the device of port C is scanned.

If you log out after the EtherCAT bus starts and perform the device scanning operation, the scanning operation may fail or return an inaccurate result. The reason is that reading the slave information (for example, object directory 0xF050) through SDO communication is timed out, so the scanning result is affected. (SDO communication is asynchronous, so the CPU load, bus cycle period, and user program execution time will affect the SDO communication.)

Updating a Device

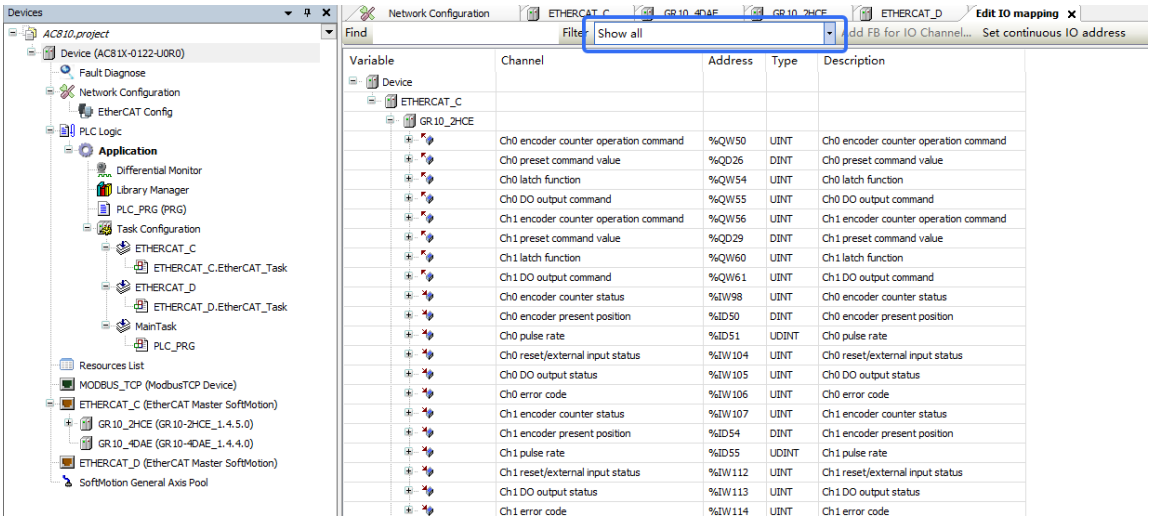
If the master version does not match or needs to be upgraded, you can run the device update command.

Procedure: Right-click the EtherCAT master. In the shortcut menu displayed, select "Update Device". The following dialog box is displayed. Click "Display all versions (for experts only)", select the desired version, and click "Update Device".



Editing I/O Mappings

After you click "Edit IO mapping", the following page is displayed.



In the "Filter" field, select a filter to filter out undesired items.

Bus Tasks

All IEC bus tasks of the PLC are scanned and executed cyclically strictly based on the same logical sequence. The logical sequence includes four steps: input update (1), IEC task execution (2), output update (3), bus cycle execution (4), as shown in the following figure.

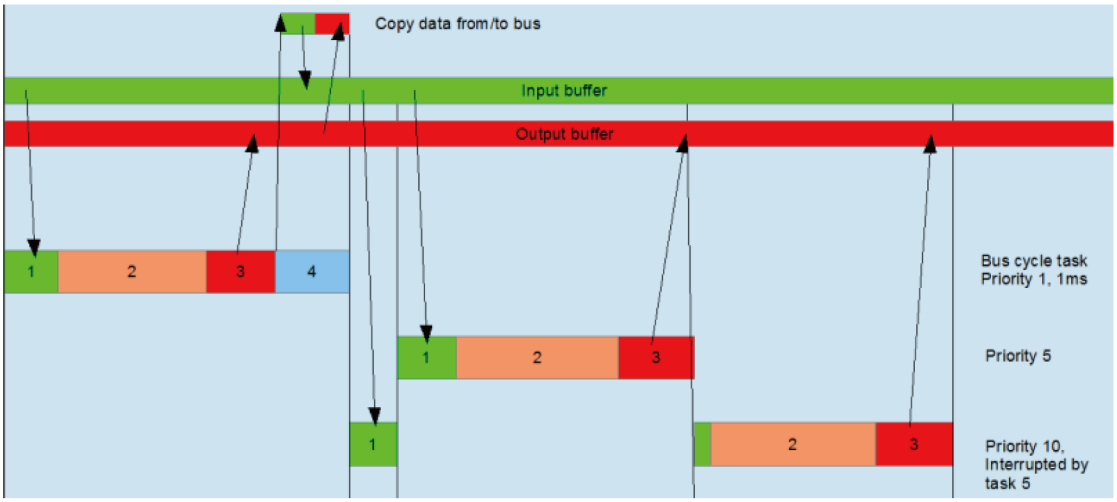


Figure 4-8 Bus cycle tasks

Each step is described in the following table.

No.	Step	Color	Description
1	Input update	Green	Before the IEC task starts, data is read from the bus input buffer, and copied to the task-related input variable.
2	IEC task execution	Orange	Scan and execute the POU under the bus task.

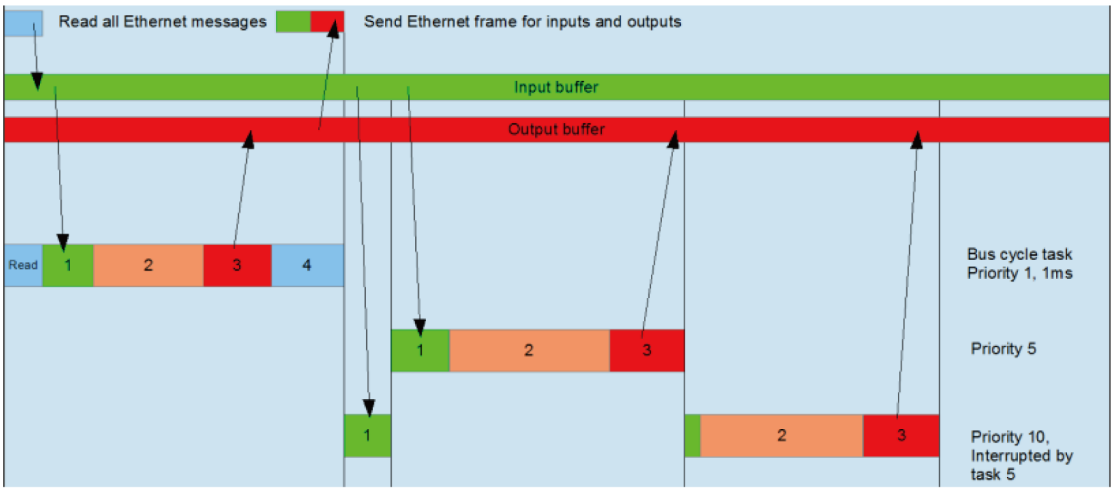
No.	Step	Color	Description
3	Output update	Red	Before the IEC task is completed, the bus-related output variables in the task are copied to the bus output buffer.
4	Bus cycle	Blue	It is the bus communication execution program implemented by bottom-layer I/O drive. It includes two functions: Transfer the data from the bus output buffer to the data reception buffer of the remote slave; and transfer the data in the sending buffer of the remote slave to the bus input buffer.

Warning

- If an output variable is used by multiple tasks, the variable value is uncertain (the output variable value of this task may be changed or overwritten by other tasks).
- If a task is interrupted by another task with a higher priority, the high-priority task reads data from the input buffer, and synchronizes the data to the input variable of the current task. Therefore, the input variables within a scan cycle may be different. To avoid this problem, copy the input variable value before the task starts so that the task will call the copied input variable.

Special Bus Cycle Action of EtherCAT

The bus data in the last cycle is copied before IEC input.



The "Enable each task message" option in EtherCAT master configuration is available. After this option is activated, the additional information of each task will be sent to the device. In this situation, bus communication can be executed under multiple tasks, to reduce the bus load.

The following figure shows an EtherCAT bus cycle table when the "Enable each task message" option is activated.

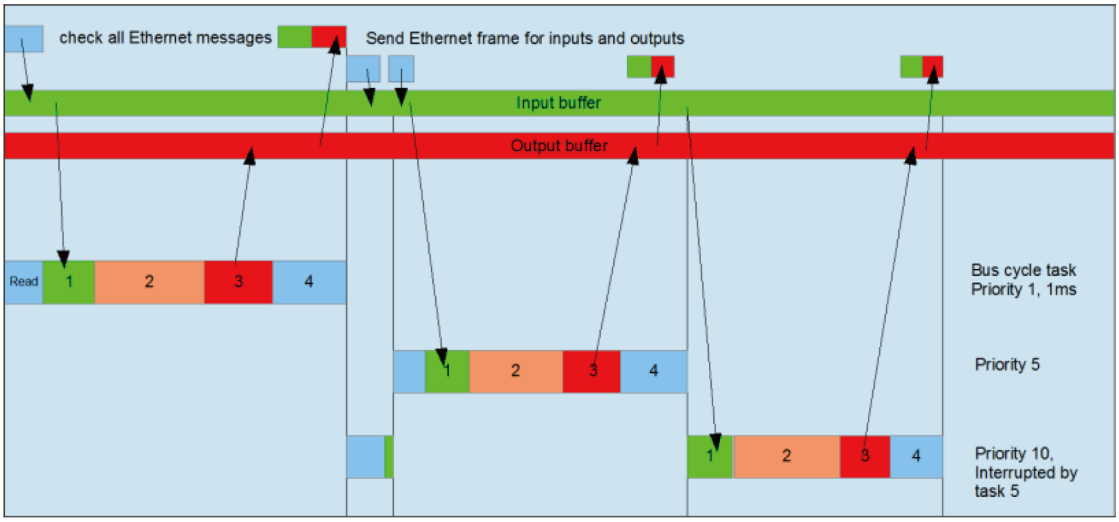
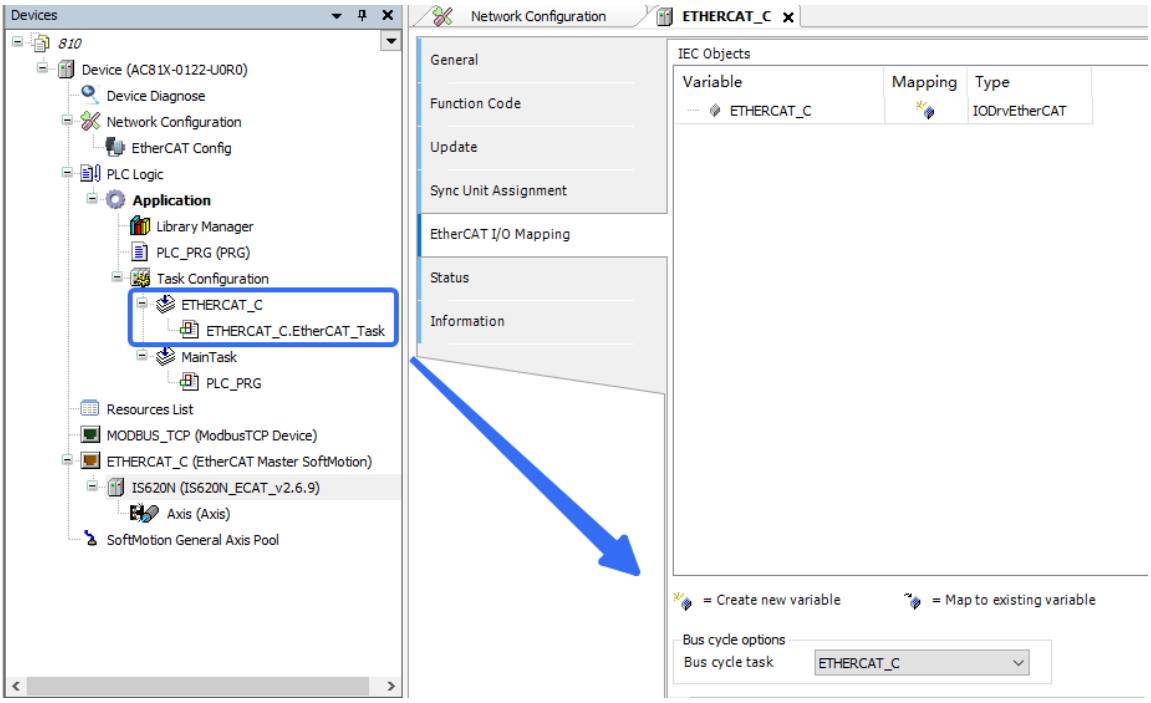


Figure 4-9 EtherCAT bus cycle table

Note

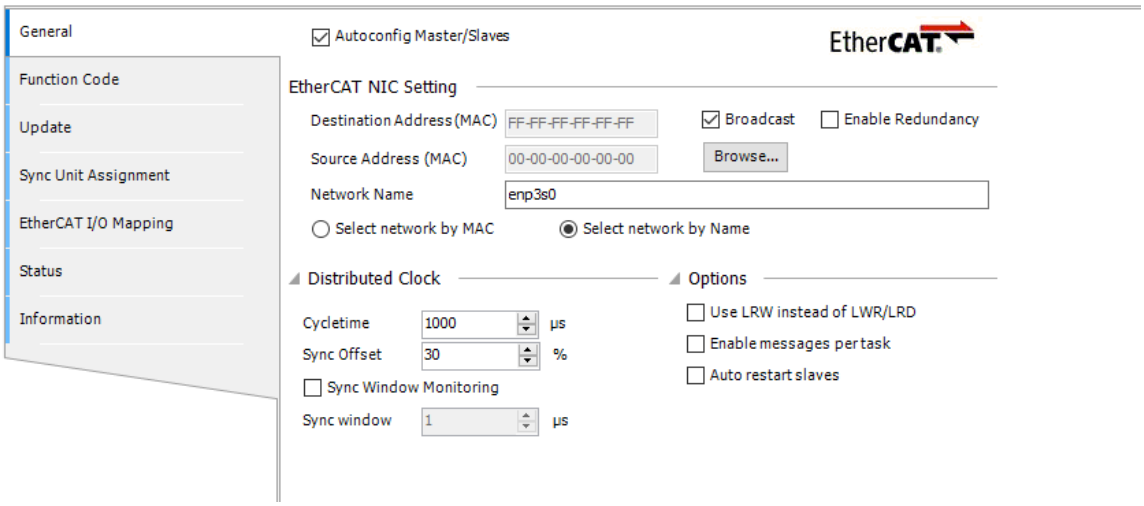
- After the EtherCAT master is automatically inserted, the "EtherCAT_***" task is also inserted to the current task configuration.
- The bus cycle task of the EtherCAT master must be executed in the same task as the EtherCAT_***.EtherCAT_Task.
- The input and output of the EtherCAT master are executed in the same task as EtherCAT_***.EtherCAT_Task. The corresponding EtherCAT task must be configured for the bus cycle task in I/O mappings of the EtherCAT master. Therefore, it is recommended to execute the device control program (such as the PLCOpen axis control command) under this task.



4.4.3 EtherCAT Master

General settings

The EtherCAT master configuration dialog box provides main settings of the master.



Autoconfig Master/Slaves

If you select this option, the main settings of the master and slaves will be automatically completed. In this case, all the editors of slaves will not display the FMMU/Sync tab.

Note

- The automatic settings are default settings, which are strongly recommended to standard applications.
- If this option is not selected, you must manually complete all settings of the master and slaves. Therefore, you must have professional skills.

EtherCAT NIC Setting

- Destination Address (MAC)
  The MAC address of the EtherCAT network member that receives packages. If "Broadcast" is selected, use a broadcast address (FF FF FF FF FF FF).
- Enable Redundancy
  If a ring topology is selected, the redundancy option needs to be enabled. When this option is enabled, if a single point failure occurs in network connection, the EtherCAT network can still work normally. After this option is enabled, you also need to define the second EtherCAT NIC.

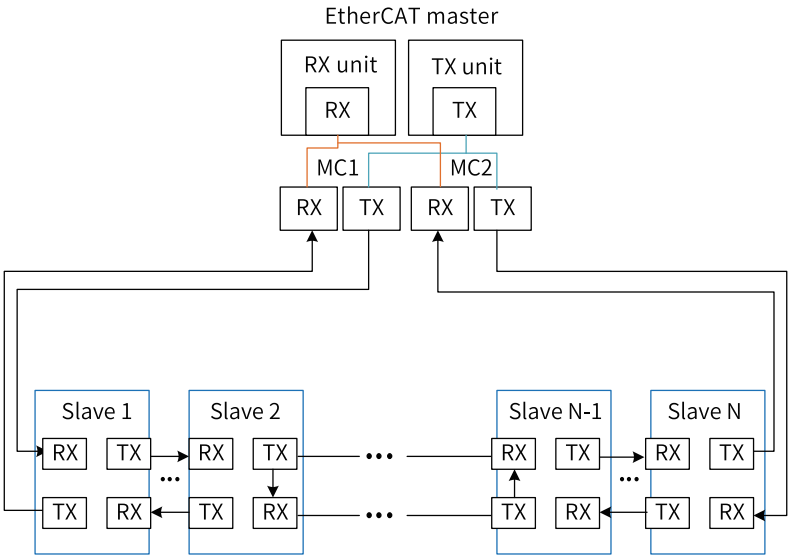
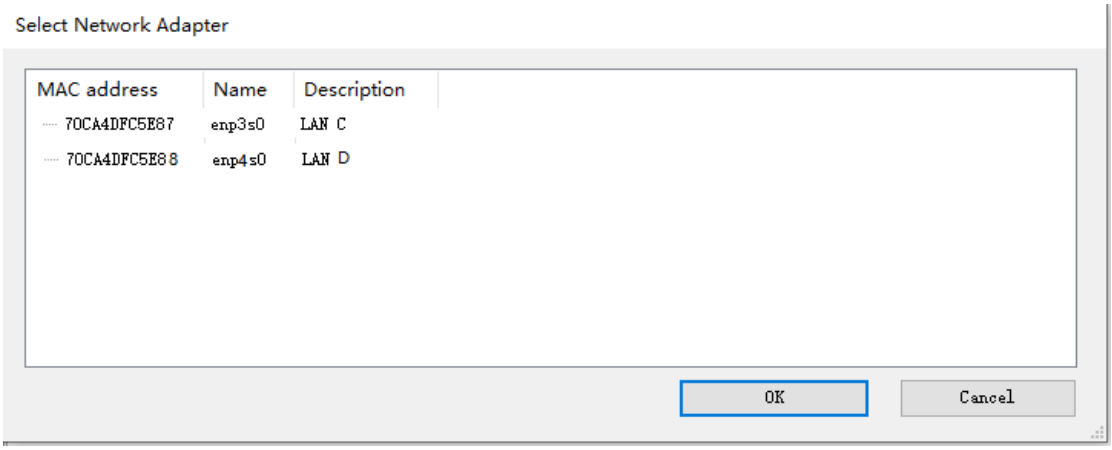


Figure 4-10 EtherCAT ring topology (redundancy)

- Source Address (MAC)
  The MAC address of the PLC.
- Network Name
  The NIC name, including the following options:
- Select network by MAC/Select network by Name
  Each EtherCAT NIC has a unique MAC address. Therefore, if "Select network by MAC" is selected, this project cannot be used on other devices.

  If you need the project to be independent of devices, click "Select network by Name". In each option, you can click "Browse..." to display the MAC addresses of available target devices and their names, as shown in the following figure.

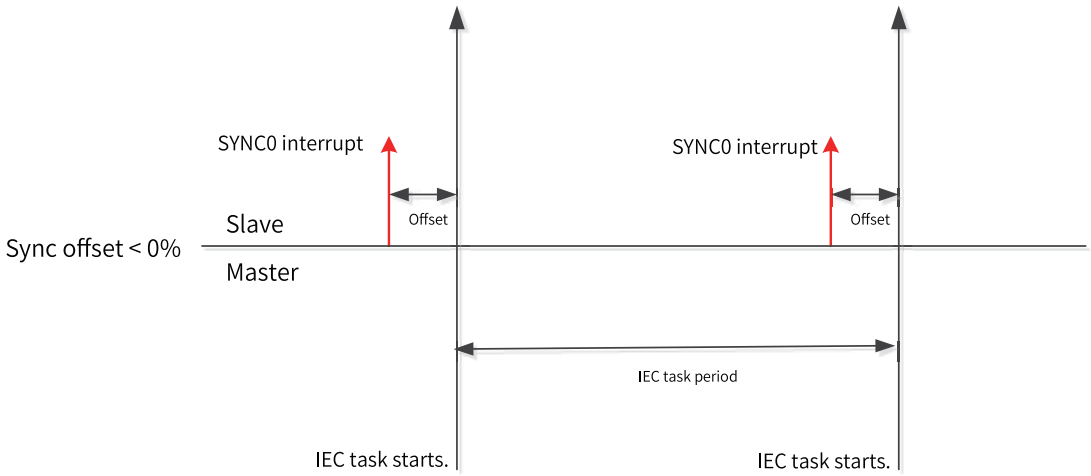


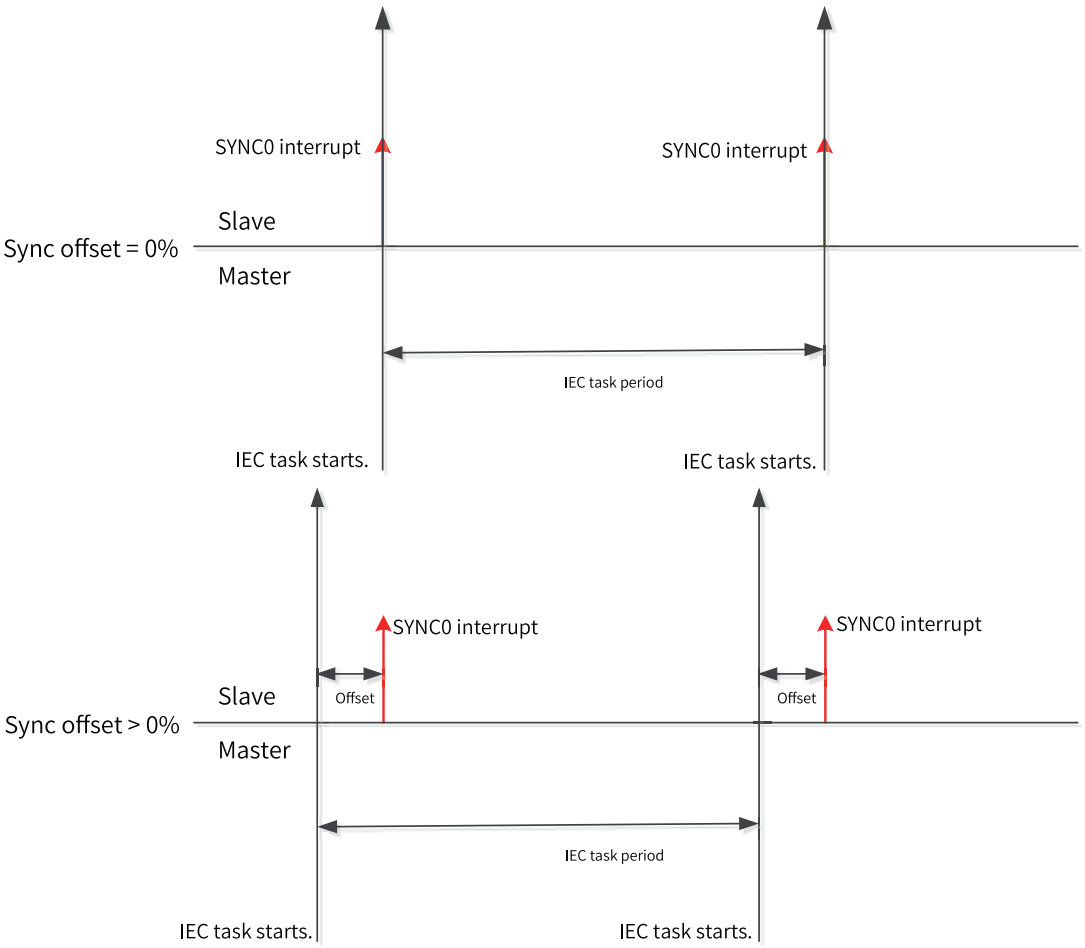
- Redundant EtherCAT NIC Setting
  If "Enable Redundancy" is selected, this setting is available. You can set the options of redundant EtherCAT NIC.

Distributed Clock

- Cycle time [μs]
  The execution cycle time of the EtherCAT master function. It must be the same as the cycle time of the IEC task bound to the EtherCAT master. If the distributed clock function of the slave is enabled, the cycle time is synchronized with the "Distributed Clock" settings in the slave editor.
- Sync Offset [%]
  The ratio of the cycle time of the EtherCAT master's IEC task (PLC task) to the reference distributed clock (generally, SYNC0 interrupt), ranging from –50% to +50%, with the default value 30%.

  Sync offset [%] = (SYNC0 interrupt time – PLC task cycle start time)/PLC task cycle time





Note

- By default, the PLC task cycle time is the same as the distributed clock cycle time of the slave.
- In actual settings, consider the clock jitter of the controller master (system instantaneity), PLC task execution time, PLC task cycle time, and number of slaves.
- In InoProShop V1.5.0 and later versions, the EtherCAT master type of new projects is "EtherCAT Master SoftMotion", which is different from the "EtherCAT Master" type in earlier versions. The default sync offset of the master is 30s. Do not modify this value unless specially required; otherwise, an EtherCAT data synchronization error may occur.

- Sync Window Monitoring
  After this option is enabled, synchronization of the slave is monitored.
- Sync Window
  The synchronization window monitoring time. If the synchronization information of all slaves is included in this window, the xSyncInWindow (IoDrvEtherCAT) variable is set to TRUE; otherwise, set it to FALSE.
- Diagnostic Info
  In the online mode, the diagnosis information includes the information about EtherCAT master startup and running.

Options

- Use LRW instead of LWR/LRD
  This option enables the slave-slave communication. EhterCAT master logical addressing will use the combination of read/write command (LRW) to replace the read-only (LRD) and write-only (LWR) commands.
- Enable messages per task
  After this option is selected, the read and write commands of input and output information will be completed by different tasks.
- Auto restart slaves
  After this option is selected, the master will restart the slaves upon communication error.

Master Settings

The master settings can be completed only when the auto mode is disabled (see the following description); otherwise, these settings are automatically completed and hidden in the dialog box.

- Image In Address: Inputs the first logical address of the first slave.
- Image Out Address: Outputs the first logical address of the first slave.

Function Code

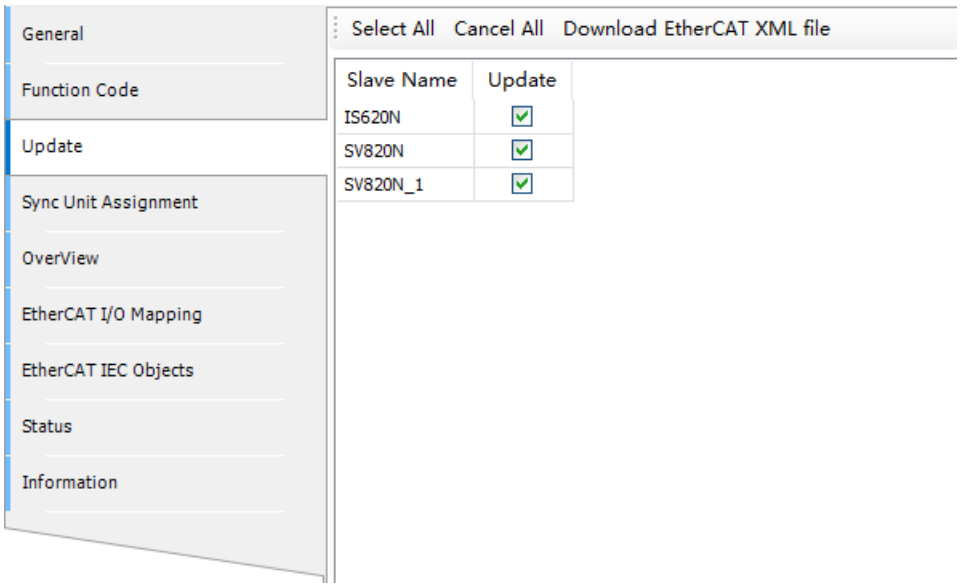
Function codes refer to the vendor-specific parameters of Inovance servo products. By using the master options, you can read/write, import, and export vendor-specific parameters of multiple products, for debugging and maintenance.

FunCode	Name	CurrentValue	WriteValue	DefaultValue	Range	Access
H02						
H02-00	Control mode			9	0-9	RO
H02-01	Absolute Encoder M...			0	0-3	RW
H02-02	Rotating direction			0	0-1	RW
H02-03	Direction of output p...			0	0-1	RW
H02-05	Stop mode at servo ...			0	0-1	RW
H02-06	Stop mode at fault 2			1	0-2	RW
H02-07	Stop mode at overtr...			1	0-2	RW
H02-08	Stop mode at fault 1			0	0-0	RW
H02-09	Brake release comm...			250	0-500	RW
H02-10	Servo drive disable ...			150	1-1000	RW
H02-11	Output speed limit o...			30	0-3000	RW
H02-12	Waiting time from se...			500	1-1000	RW
H02-15	Display of keypad w...			0	0-1	RW
H02-21	Allowed minimum br...			40	1-1000	RO
H02-22	Power of built-in br...			40	1-65535	RO
H02-23	Resistance of built-i...			50	1-1000	RO
H02-24	Resistor heat dissipa...			30	10-100	RW
H02-25	braking resistor type			0	0-3	RW
H02-26	Power of external d...			40	1-65535	RW
H02-27	Resistance of exter...			50	1-1000	RW
H02-31	Parameter initialization			0	0-2	RW
H02-32	Default keypad display			50	0-99	RW
H02-35	Display frequency of...			0	0-20	RW

- Select All: Selects all slaves, axes, and servo function codes under the axes.
- Cancel All: Cancels all selected options.
- Read the Parameter: Reads the servo function codes carrying the RO or RW attribute when the servo is running.

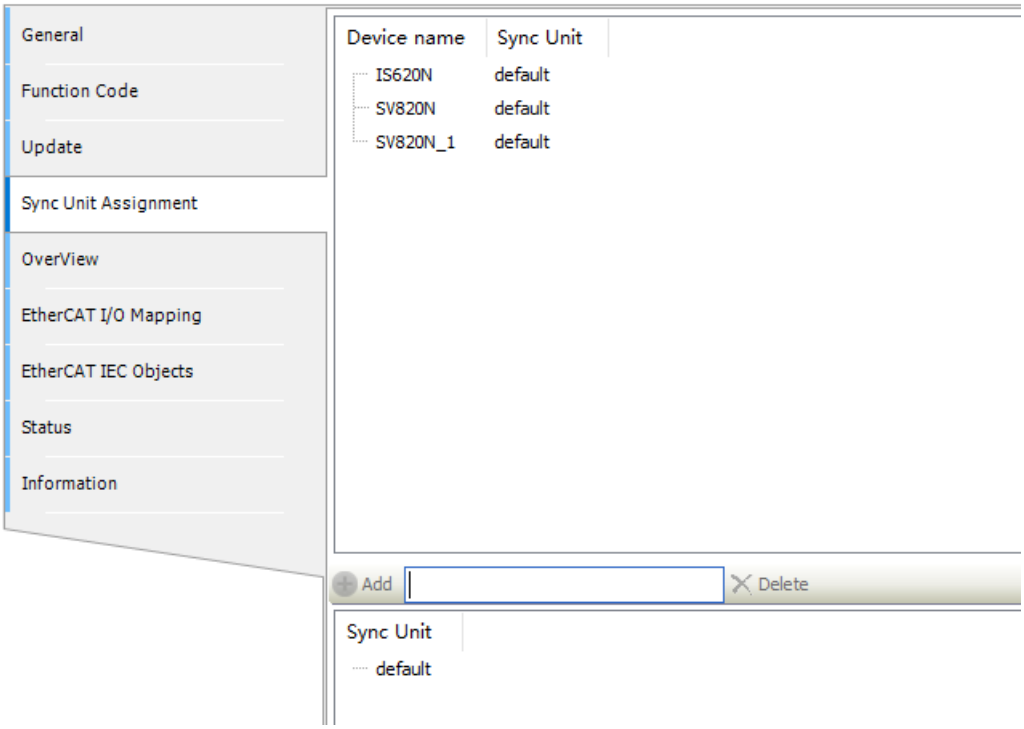
- Write the Parameter: Writes the servo function codes carrying the WO or RW attribute when the servo is running.
- Export Function Code: Exports servo function codes of all slaves to an Excel file.
- Import Function Code: Imports function codes of all slaves from an Excel file. If the configuration is inconsistent with that in the file, an error is reported.

Upgrade



- Select All: Selects all slaves.
- Cancel All: Cancels all selected slaves.
- Download EtherCAT XML file: Downloads the EtherCAT slave's XML file from InoProShop to the EEPROM of the slave. To perform batch download, select multiple slaves.

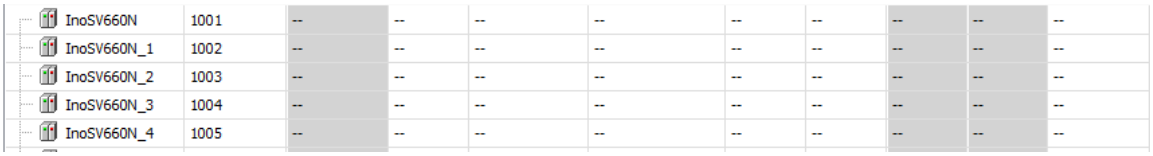
Sync Unit Assignment



- Sync Unit: Groups the slaves. If any slave in the group is lost, the entire group is lost, but other groups are not affected.
- Add: Adds a group. Then you can select the slave group.

Overview

The "Overview" page displays the slave communication mode and communication state, as shown in the following figure.



The following table lists the tabs on this page and their descriptions.

Parameter	Description
UpdateData	Refreshes the slave data once.
Auto Update	Cyclically refreshes monitoring data

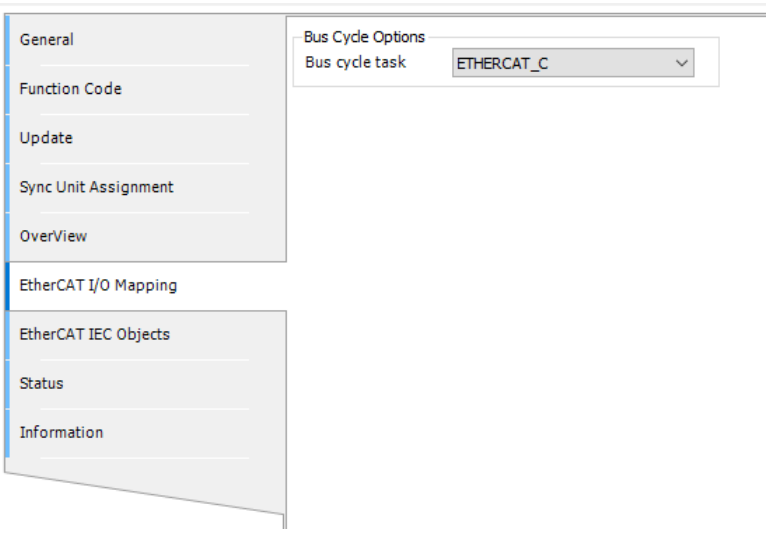
Parameter	Description
-	• Classic mode: Slave communication supports a hybrid mode of alias addresses and sequential addresses. The alias addresses are editable only in the slave window, but are not editable in the overview window. • Sequential model: Device communication is performed based on sequential addresses. The alias addresses are editable in both the slave window and overview window. • Alias mode: All slaves communicate with each other with their alias addresses. In this case, the alias addresses are editable only in the overview window, but are not editable in the slave window.
Export	Exports the table from the page.
ResetErrorFrameCount	Clears the error frame count and link lost count.
Reset Failed Slave	Clears the fault information on the overview page.

The following table lists the parameters on this page and their descriptions.

Parameter	Description
Name	Displays the name of all slaves under the master.
Slave Address	Displays the address of the slave.
Alias Address	Displays the alias of the slave when the slave communication mode is set to "Alias mode". • After the slave communication mode is set to "Alias mode", an alias address is assigned to every new slave and "Alias mode" is selected. • After the slave communication mode is set to "Classic mode" or "Sequential model", no alias address is assigned to the new slave and "Sequential model" is selected.
State	• Initialize: The slave is being initialized. • Pre-Op: The slave is in the pre-operation state. • Safe-Op: The slave is in the safe operation state. • Op: The slave is in the operation state. • BootStrap: The slave is in the boot state. • No Communication: The communication of the slave fails. • Error: The slave state is incorrect.
ErrorFrameCount	Counts the error of each activated port.
LinkLostCount	Counts the disconnected links of each activated port.
Port	Displays the loading state of each port.
Fault Information	Describes the slave fault.

EtherCAT I/O Mapping

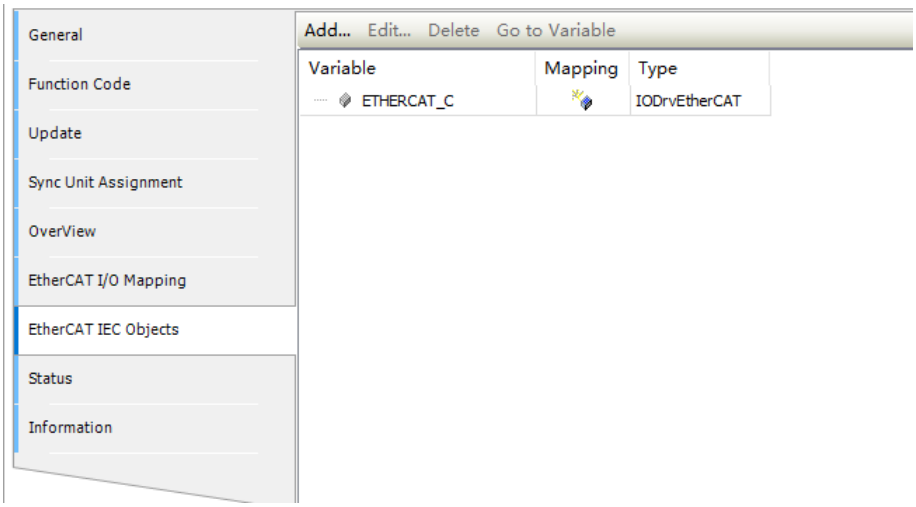
On the "EtherCAT I/O Mapping" tab page, you can set the bus cycle task for the master.



EtherCAT IEC Object

This is a tab on the EtherCAT master configuration editor, in which the instance (variable) of IODrvEtherCAT type is specified for EtherCAT I/O, so that the PLC connected to EtherCAT can be controlled by the user program. For the description of mapping, see the section "I/O Mapping".

The automatically created master instance is displayed at the bottom of the "IEC Objects" dialog box.

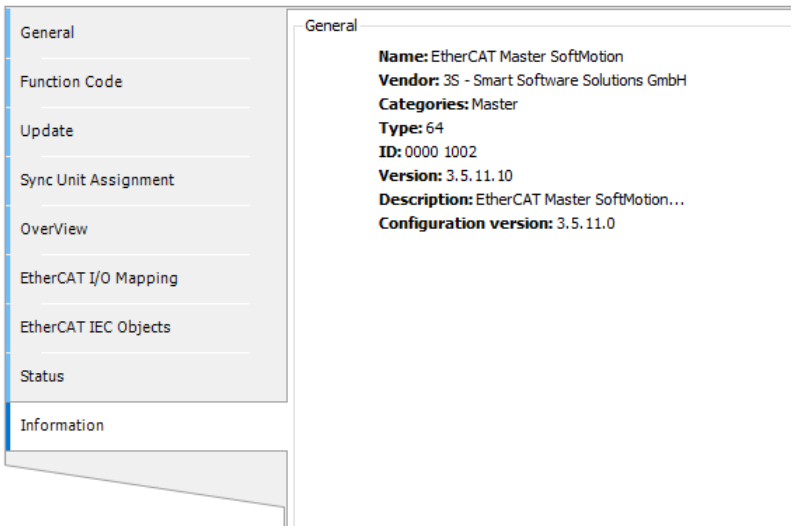


Note

The variables and types of mappings must be consistent.

Information

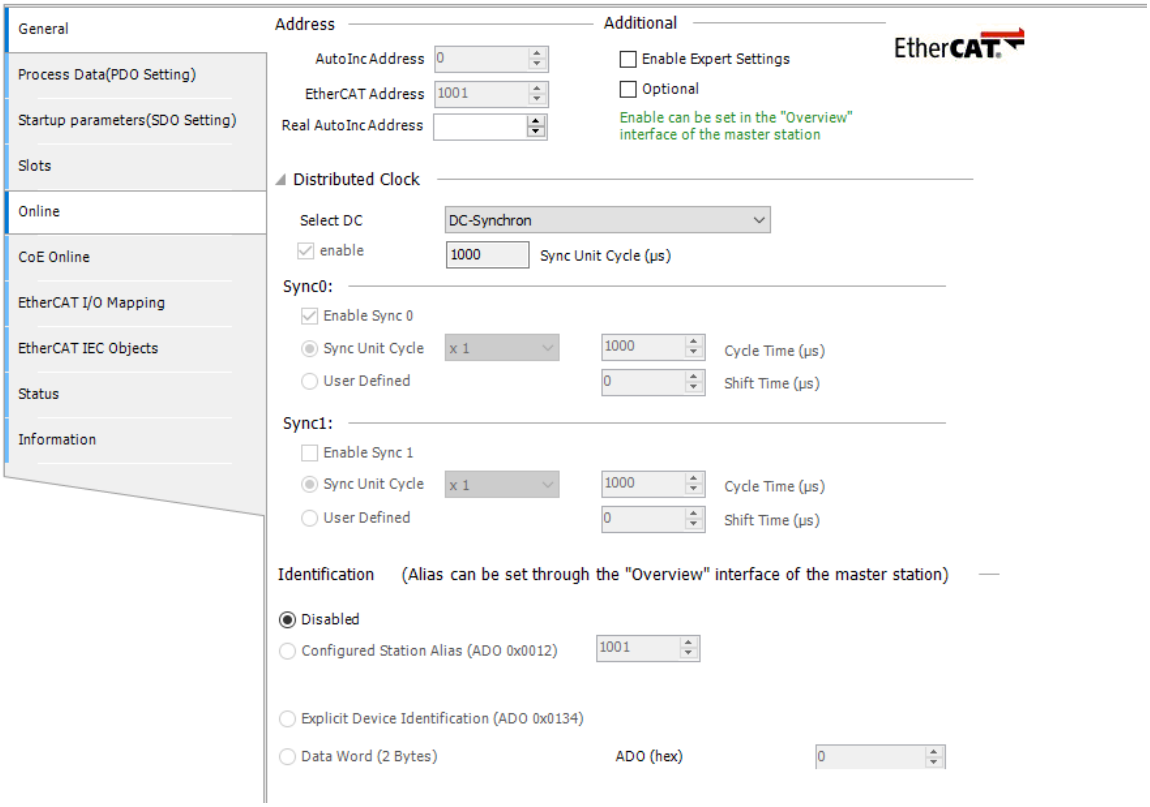
This dialog box displays the following information about the current module: name, vendor, group, category, ID, and version.



4.4.4 EtherCAT Slave

General Settings

The following figure shows the "General" page of the EtherCAT slave, which provides basic settings of the slave.



Address

If "Autoconfig Master/Slaves" in the master editor is not enabled, the following options are available:

- AutoInc Address: Indicates the automatically incremental address (16 bits), which is determined by the physical topology location of the slave in the network. This address is used only when the

EtherCAT master is started. The EtherCAT slave address is allocated to the slave at the corresponding physical topology location by using the sequential addressing method.

During sequential addressing, according to the EtherCAT protocol, the automatically incremental address of the slave is determined by its connection location in the physical topology network, and is represented by a negative number. Sequential addressing sends the subframe, in which the automatically incremental address is increased by 1 every time the subframe passes a slave. When the physical slave receives the frame, it determines whether the frame belongs to itself by checking whether the automatically incremental address in the frame is 0. This mechanism is called sequential addressing or automatically incremental addressing.

- Slave Address: Indicates the final address (formal address) of the slave. It is assigned by the master when the master starts. This address is independent of the actual location on the network. The slave address is irrelevant to the connection order on the network segment.
- Enable Expert Settings: If this option is enabled, the settings of the distributed clock, auto check upon startup, timeout, cycle unit control, and watchdog can be performed.

Distributed Clock

- If "Enable" is selected, the distributed clock function is enabled.
- Sync Unit Cycle (μs): If the distributed clock function is enabled, the syn unit cycle value is the same as the cycle time of the EtherCAT master.
- Select DC: This option provides all distributed clock settings in the device description file, including AutoRun, SM Event Synchron, and DC Synchron. The following table describes options.

No.	Option	Function
1	AutoRun	In this mode, the local control cycle is generated by a local timer interrupt. The cycle time is determined by the master. This is an optional function of the slave.
2	SM Event Synchron	Synchronization is triggered by the data input or output event within the local cycle. The master can write the sending period of the process data frames to the slave, and the slave checks whether this cycle time is supported or optimizes the cycle time. This is an optional function of the slave. Synchronization is usually triggered by data output events. If the slave has only the data input event, synchronization is triggered by the data input event.
3	DC Synchron	Synchronization is triggered by the SYNC event within the local cycle. The master must complete data frame sending before the SYNC event occurs. Therefore, the master clock must be synchronized with the reference clock.
4	SYNC0: SYNC1:	It indicates the slave synchronization signal 0/1. The distributed clock control unit (internal function of the EtherCAT dedicated communication chip) of the slave can generate two synchronization signals: SYNC 0 and SYNC 1, which provide the interrupt sign to the application-layer programs of the slave, or directly trigger the output data update.
5	Enable SYNC 0: Enable SYNC 1:	After this option is selected, SYNC0/SYNC1 synchronization signals are started. Sync Unit Cycle: If this option is selected, the synchronization cycle time of the slave is (master cycle time x selected coefficient). The "Cycle time (μs)" field displays the current cycle time.

- User Defined: If this option is selected, you can enter the desired cycle time in μs in the "Cycle time (μs)" field.

Diagnosis

This part is available only in the online mode.

- Current State: Displays the current communication state machine of the slave. The possible values include initialization, pre-operation, safe operation, operation, and BootStrap (not supported by Inovance servo currently). If the state is "Op", the configuration of the slave is completed.

Options

Options: If a slave is set as optional, no error message is generated, so that the device will not be included in the bus system. To enable this option, you must save a station address in the slave. Therefore, define and write the station alias address into EEPROM. In addition, this option is valid only when "Autoconfig Master/Slaves" in EtherCAT master settings is selected and this option is supported by the EtherCAT slave.

Startup Checking

By default, the system automatically checks the vendor ID or product ID after startup. If the IDs do not match, the bus stops running and does not perform the subsequent operations. This avoids downloading incorrect configuration.

Timeout

By default, the following operations are not defined as timeout. If you need to know whether the operations exceed the specified time, set the time here.

- SDO Access: Specifies the timeout interval in which the service data object (SDO) of the EtherCAT master accesses the slave.
- I -> P: The communication state machine of the slave changes from initialization to pre-operation.
- P -> S / S-> O: The communication state machine of the slave changes from pre-operation to safe operation, or from safe operation to operation.
- DC cyclic unit control: Allocated to the local microprocessor to set the distributed clock options. This function is completed in register 0x980 of the EtherCAT slave. The possible values include cyclic unit, latch unit 0, and latch unit 1.

Watchdog

- Set multiplier: Sets the frequency multiplication ratio of the watchdog timer to determine the minimum increment unit. The default value is 2498. The minimum increment unit is 100 μs.
- Set PDI watchdog: If the process data interface (PDI) watchdog is enabled, when the PDI communication time of the EtherCAT slave exceeds the specified value, the watchdog is triggered.
- Set SM watchdog: If the synchronization management (SM) watchdog is enabled, when the process data (PD) communication time of the EtherCAT slave exceeds the specified value, the watchdog is triggered

Slave Alias

The settings are valid only when "Options" is selected and the slave supports alias address (defined in the device description file). If the alias address of the slave has been configured, the slave can normally run without the need of modifying the user program configuration when you adjust its location on the physical topology network.

Note that, after the slave alias address is changed, you must download the user program again to make the modification effective. In addition, the alias addresses of some slaves can take effect only after power cycle. For details, see the slave user guides.

- Disable: If this option is selected, the slave will not detect the alias address.
- Configured Station Alias (ADO 0x0012): When this option is supported by the slave and "Options" is selected, the alias address can be written in the online running state.
- Write into EEPROM: This option is available only in the online mode. It writes the defined address into the slave EEPROM. If the slave does not support this option, this option is invalid and the slave cannot work with the alias.
- Actual address: This column is available only in the online mode. It displays the actual address of the slave. It is used to check whether the EEPROM writing command is successful.
- Explicit Device Identification (ADO 0x0134): Reserved.
- Data Word (2 Bytes): Reserved.

FMMU/Sync

If the autoconfig mode of the master is not enabled, this dialog box is only provided by the EtherCAT slave configuration editor. It displays the slave's fieldbus memory management units (FMMUs) defined in the device description file and the synchronization manager (Sync). These settings can be modified, for example, the communication between slaves.

Note

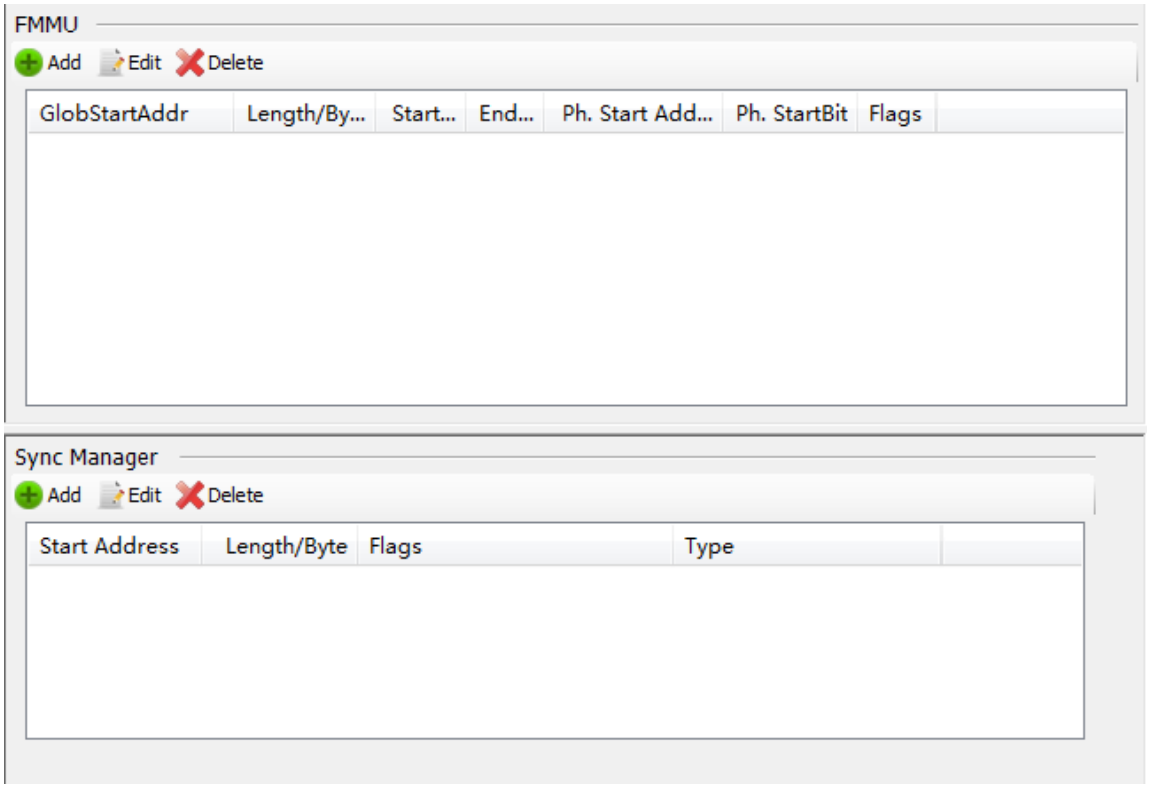
These are advanced settings, and are unnecessary for standard applications.

FMMU

This tab allows you to configure the FMMUs of the slave used to process the process data, including the logical address (Ph. Start Address) mapping to each physical address (Ph. GlobStartAddr). You can add a new unit by clicking "Add..." or "Edit..." in the FMMU dialog box.

Sync Manager

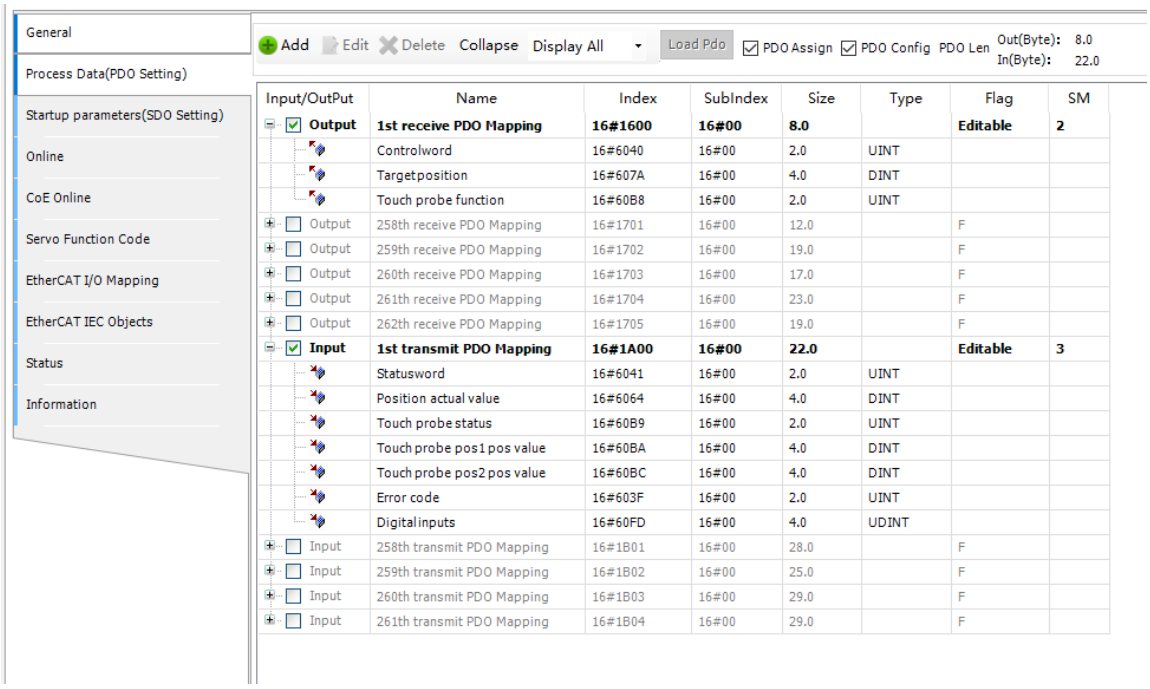
This tab displays the synchronization manager of the slave. On this tab page, you can set the following information: the synchronization manager type (Mailbox In, Mailbox Out, Inputs, or Outputs), physical start address, access type, buffer, and physical address to be accessed by the interrupt. In the synchronization manager editor, you can click "Add" or "Edit" to add or modify synchronization management.



Process Data

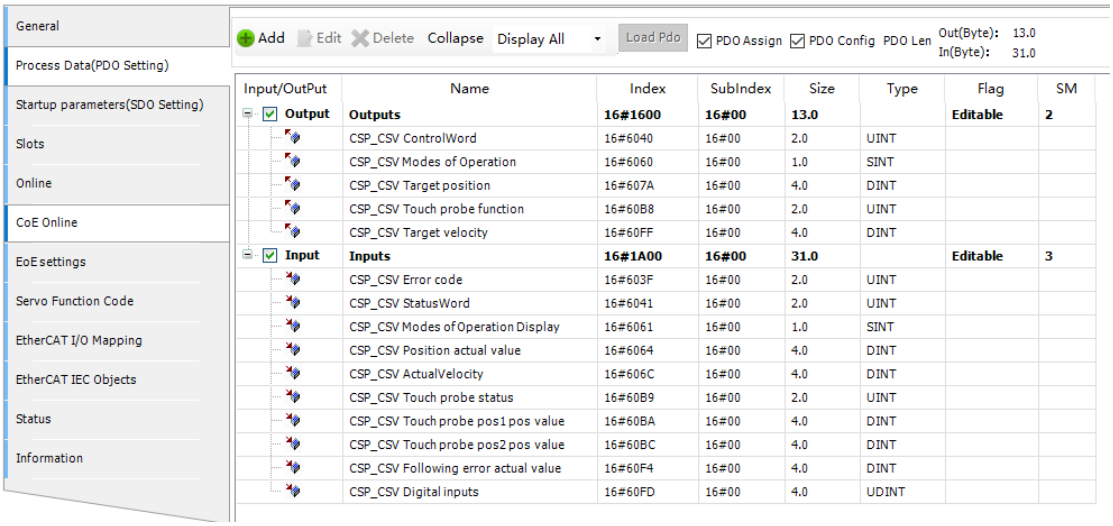
In the automated control system, application programs exchange data in two modes: time-critical and non-time-critical. Time-critical means that the specified action must be completed within the specified time window. If the action cannot be completed in the specified time window, the control is ineffective. The process for periodically sending time-critical data is called PDO. Non-time-critical data does not need to be periodically sent, and uses MailBox data communication SDO in EtherCAT.

The first row of the "Process Data" tab page is the PDO edit function key and displays PDO information, as shown in the following figure.

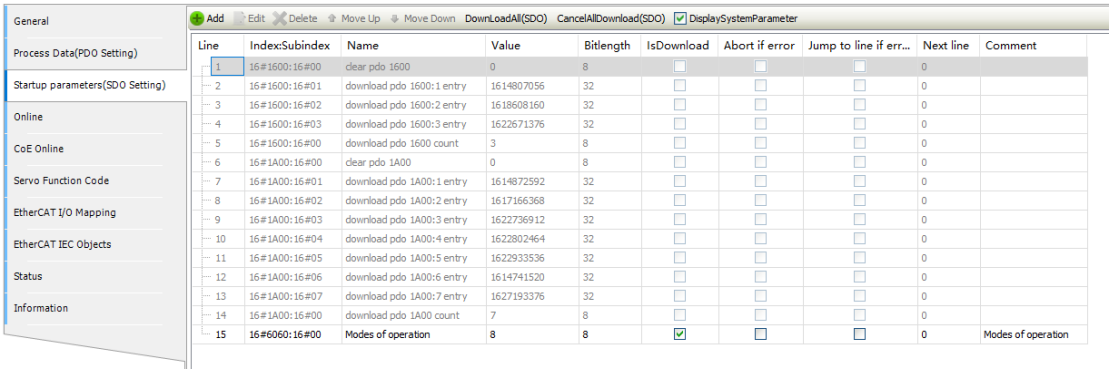


Input/OutPut	Name	Index	SubIndex	Size	Type	Flag	SM
Output	**1st receive PDO Mapping**	**16#1600**	**16#00**	**8.0**		**Editable**	**2**
	Controlword	16#6040	16#00	2.0	UINT		
	Target position	16#607A	16#00	4.0	DINT		
	Touch probe function	16#60B8	16#00	2.0	UINT		
Output	258th receive PDO Mapping	16#1701	16#00	12.0		F	
Output	259th receive PDO Mapping	16#1702	16#00	19.0		F	
Output	260th receive PDO Mapping	16#1703	16#00	17.0		F	
Output	261th receive PDO Mapping	16#1704	16#00	23.0		F	
Output	262th receive PDO Mapping	16#1705	16#00	19.0		F	
Input	**1st transmit PDO Mapping**	**16#1A00**	**16#00**	**22.0**		**Editable**	**3**
	Statusword	16#6041	16#00	2.0	UINT		
	Position actual value	16#6064	16#00	4.0	DINT		
	Touch probe status	16#60B9	16#00	2.0	UINT		
	Touch probe pos1 pos value	16#60BA	16#00	4.0	DINT		
	Touch probe pos2 pos value	16#60BC	16#00	4.0	DINT		
	Error code	16#603F	16#00	2.0	UINT		
	Digital inputs	16#60FD	16#00	4.0	UDINT		
Input	258th transmit PDO Mapping	16#1B01	16#00	28.0		F	
Input	259th transmit PDO Mapping	16#1B02	16#00	25.0		F	
Input	260th transmit PDO Mapping	16#1B03	16#00	29.0		F	
Input	261th transmit PDO Mapping	16#1B04	16#00	29.0		F	

- Add: Adds a PDO option based on the PDO group attributes (only editable attributes). You can add one group or multiple groups of data. To add multiple groups of data, press "Ctrl+" and "Shift+", and click the desired groups. Ensure that the object dictionary indexes of the options to be added are correct. Note that the number of PDOs to be added cannot exceed the limitation described in the servo guide.
- Edit: Edits a PDO option based on the PDO group attributes (only editable attributes).
- Delete: Deletes a PDO option based on the PDO group attributes (only editable attributes). You can delete one or multiple options. To delete multiple options, press "Ctrl+" and "Shift+", click the desired groups, and then click "Delete". Or right-click the selected options and select "Delete" from the shortcut menu.
- Collapse all: Collapses all PDO groups.
- Filter: Options are "Display All", "Display Output PDO", "Display Input PDO", "Display Input and Output PDO", "Display Only Output PDO", and "Display Only Input PDO".
- Load PDO: The PDO group data of the slave can be uploaded to the programming software only when the slave is running.
- PDO Assign: After this option is selected, click "DisplaySystemParameter" on the "Startup parameters" page. Then the input and output PDO group assignment information is added to the startup parameter group, as shown in the following figure.



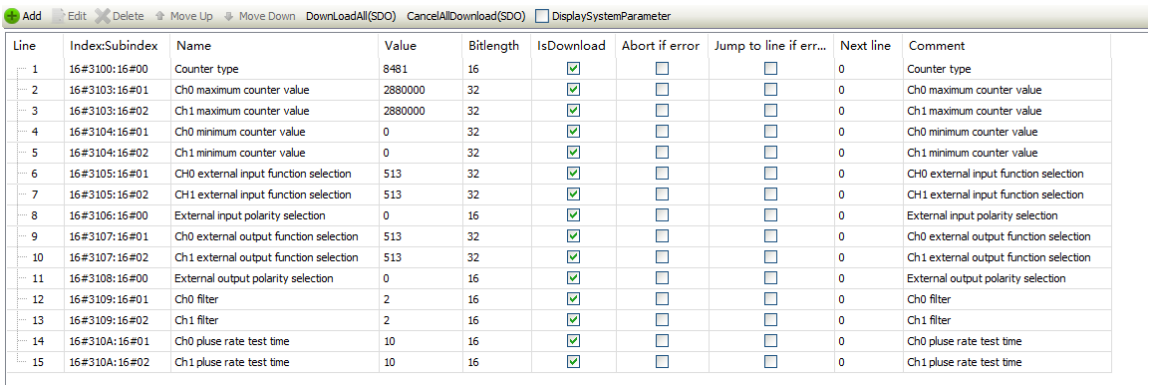
- PDO Config: Select "PDO Config" and click the "DisplaySystemParameter" on the "Startup parameters" page. Then the input and output PDO group information is added to the startup parameter group, as shown in the following figure.



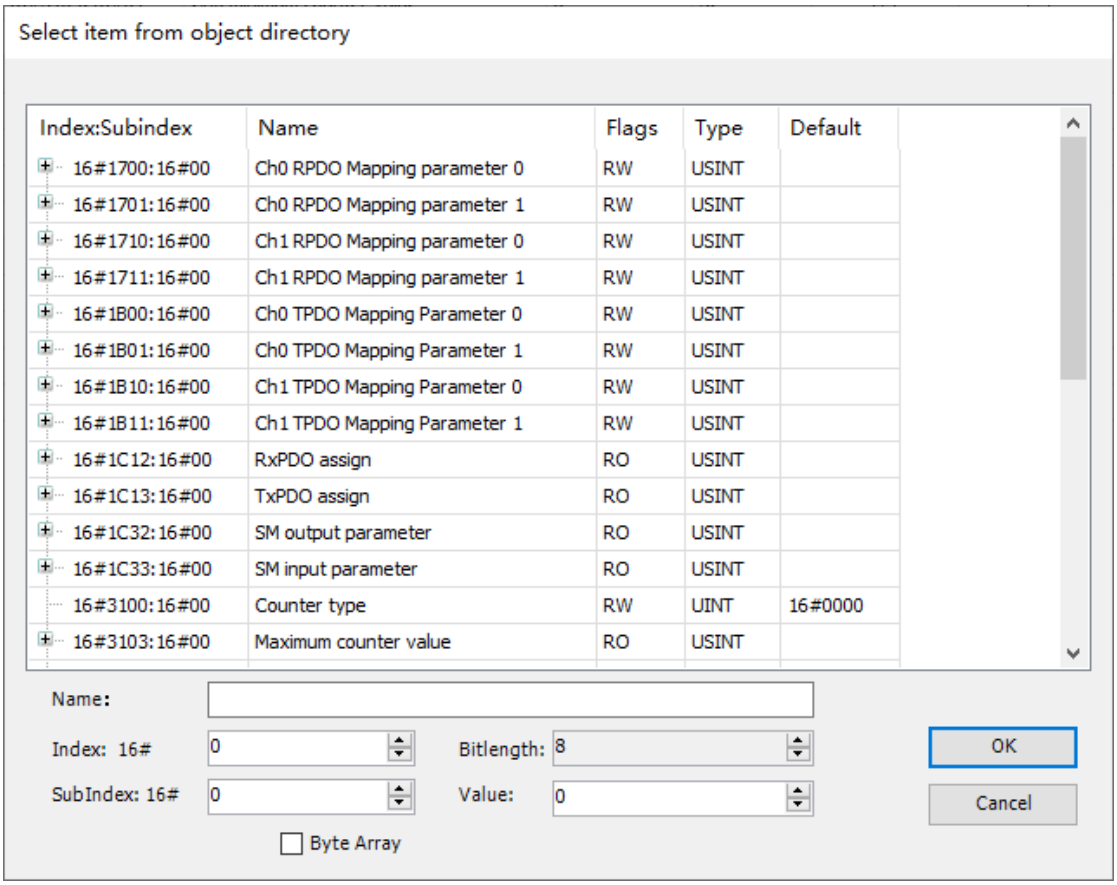
- PDO Len: The data size includes the total input and output PDO length.

Startup Parameters

The startup parameters can be transmitted to the slave through the service data object (SDO) upon system startup. The startup parameters include the basic configuration parameters used for the startup of the slave. The page is as follows.



- Add: Adds an SDO project to the startup parameter list. The object dictionary dialog box is displayed, as shown in the following figure.



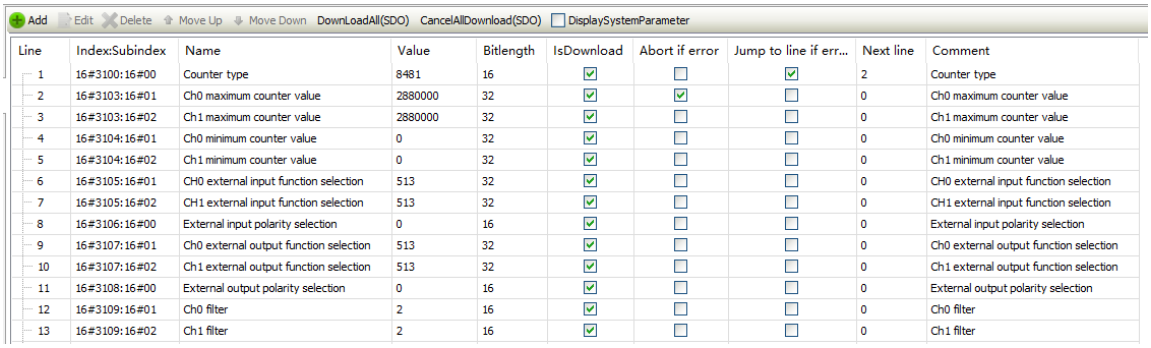
Before adding the SDO, you can modify its parameters below the edit bar, including the index, sub-index, bit length, and value, to form a new startup parameter. To add multiple groups of startup parameters, press "Ctrl+" and "Shift+", and click multiple groups of startup parameters.

- Edit: Edits the options. Read-only options cannot be edited, such as system parameters.
- Delete: Deletes the options. To delete multiple groups of startup parameters, press "Ctrl+" and "Shift+", click multiple groups of startup parameters, and click "Del" or press the "Del" key.
- Move Up, Move Down
  The SDO list order (from top to bottom) indicates the order in which the startup parameters are transmitted to the module. By clicking the "Move Up" and "Move Down" buttons, you can change the parameter transmission order.
- DownLoadAll, CancelAllDownload
  After an SDO is downloaded, you do not need to download it again. By clicking "CancelAllDownload" (system parameters cannot be canceled), you can cancel the attribute download. You can also select some attributes to be downloaded. By clicking "DownLoadAll", you can download all attributes.
- DisplaySystemParameter
  After "PDO Assign" and "PDO Config" are selected, the parameters added to the SDO are only for comparison.
- To edit the "Value" and "Comment" columns of the SDO, press "Space" or click the blank space.

Note

If an error occurs in SDO transmission, the following operations are supported:

- Abort if error: If an error is detected, SDO transmission is stopped.
- Jump to line if error or Next Line: If an error is detected, SDO transmission skips to the line of which the line number is specified. (The line number is displayed in the line column.)

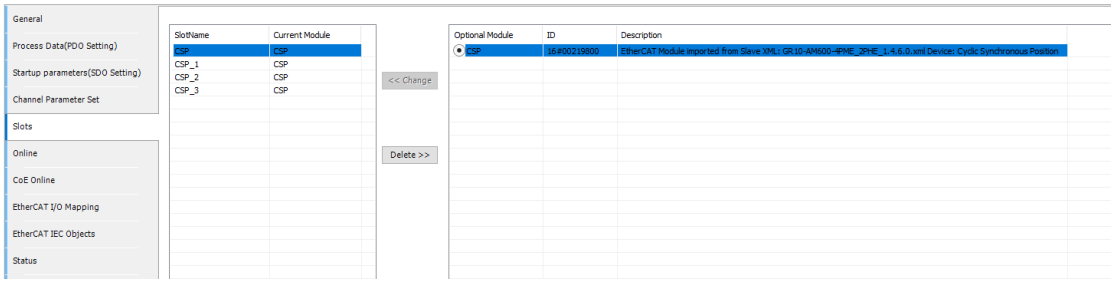


Line	Index:Subindex	Name	Value	Bitlength	IsDownload	Abort if error	Jump to line if err...	Next line	Comment
1	16#3100:16#00	Counter type	8481	16	☑	☐	☑	2	Counter type
2	16#3103:16#01	Ch0 maximum counter value	2880000	32	☑	☑	☐	0	Ch0 maximum counter value
3	16#3103:16#02	Ch1 maximum counter value	2880000	32	☑	☐	☐	0	Ch1 maximum counter value
4	16#3104:16#01	Ch0 minimum counter value	0	32	☑	☐	☐	0	Ch0 minimum counter value
5	16#3104:16#02	Ch1 minimum counter value	0	32	☑	☐	☐	0	Ch1 minimum counter value
6	16#3105:16#01	CH0 external input function selection	513	32	☑	☐	☐	0	CH0 external input function selection
7	16#3105:16#02	CH1 external input function selection	513	32	☑	☐	☐	0	CH1 external input function selection
8	16#3106:16#00	External input polarity selection	0	16	☑	☐	☐	0	External input polarity selection
9	16#3107:16#01	Ch0 external output function selection	513	32	☑	☐	☐	0	Ch0 external output function selection
10	16#3107:16#02	Ch1 external output function selection	513	32	☑	☐	☐	0	Ch1 external output function selection
11	16#3108:16#00	External output polarity selection	0	16	☑	☐	☐	0	External output polarity selection
12	16#3109:16#01	Ch0 filter	2	16	☑	☐	☐	0	Ch0 filter
13	16#3109:16#02	Ch1 filter	2	16	☑	☐	☐	0	Ch1 filter

Figure 4-11 Error handling configuration example

Slots

- On the "Slots" tab page, you can configure modules or functions for slaves in compliance with the ETG5001.1 protocol.
  As shown in the following figure, the current mode is CSP_CSV.



The following table lists available modes.

No.	Mode	Definition
1	CSP	Cyclic Synchronous Position
2	PP	Profile Position
3	CSV	Cyclic Synchronous Velocity
4	PV	Profile Velocity
5	CST	Synchronous Torque
6	PT	Profile Torque

If the position velocity mode does not meet requirements, change it to another mode, for example, CST.

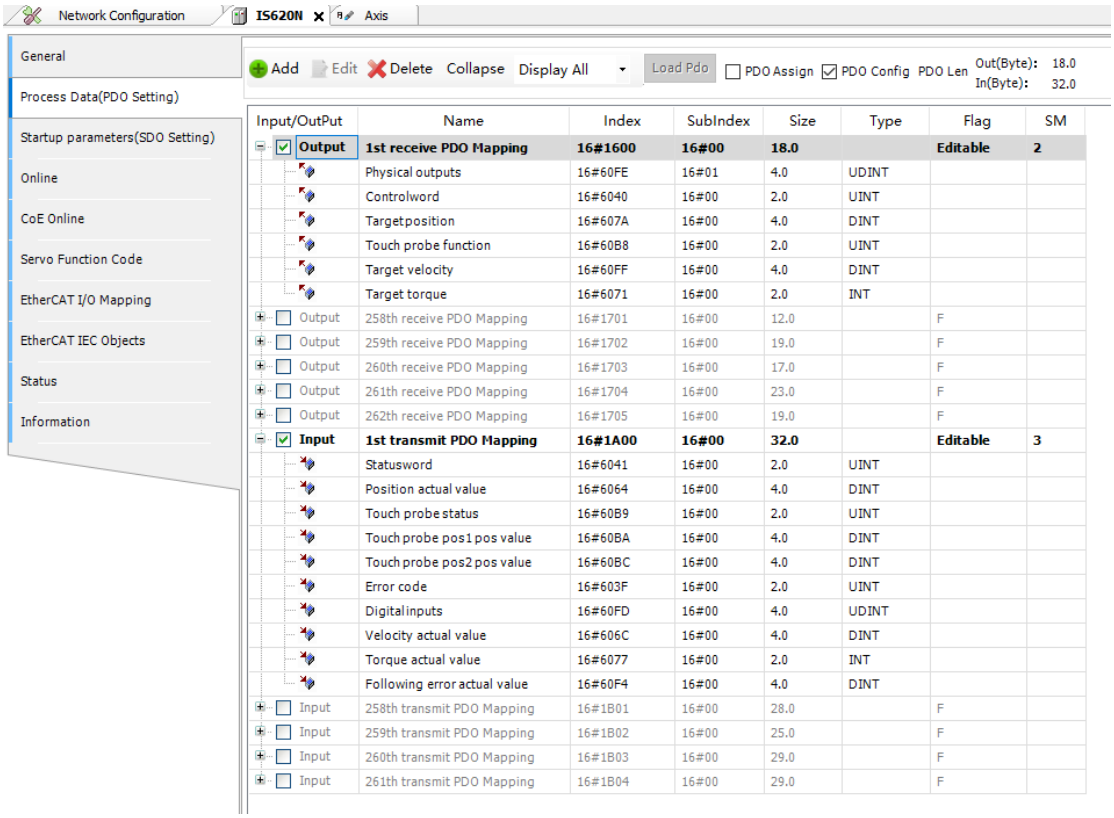
The default mode "CSP/CSV" is applicable to most application scenarios. If the field drive shaft needs to use CSP and CST, select "CSP/CST".

Note

- Change: Switches to another mode.
- Delete: Deletes the current mode.
- To change the mode, delete the current slot mode first.

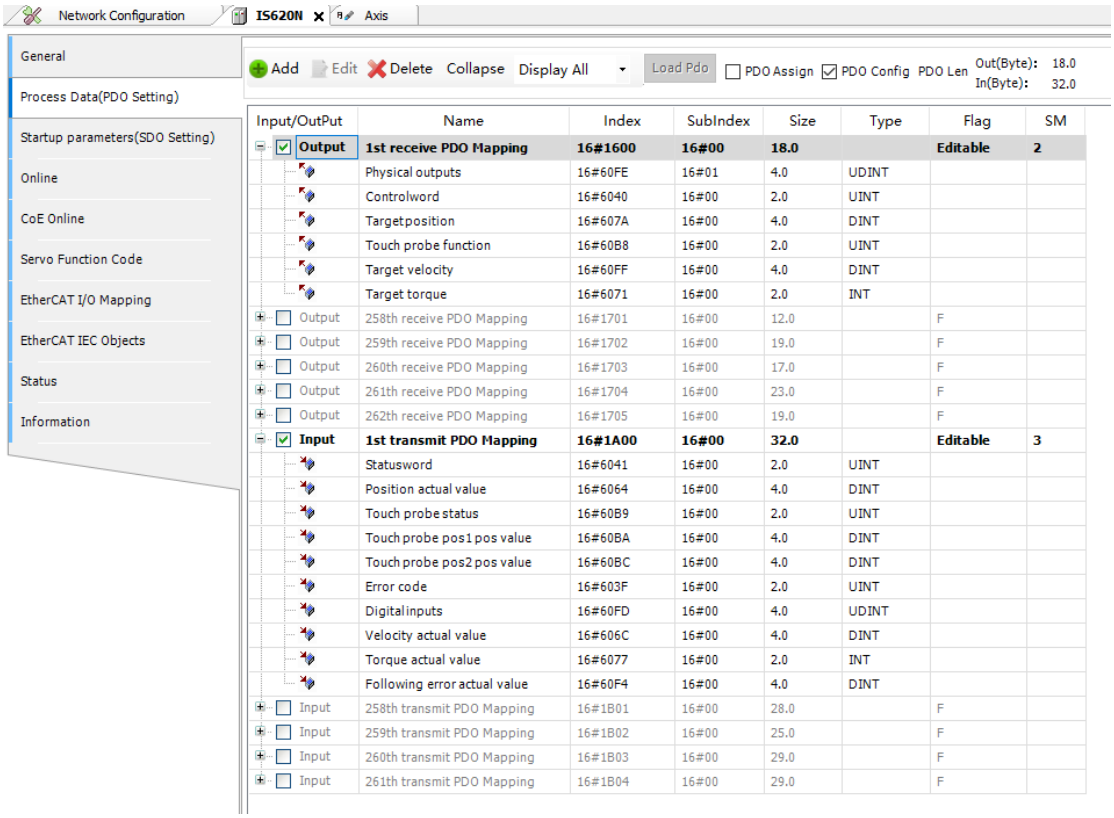
After the CSP_CSV mode is changed to the CST mode, the process data and I/O mapping are also changed:

Before the change to CST:

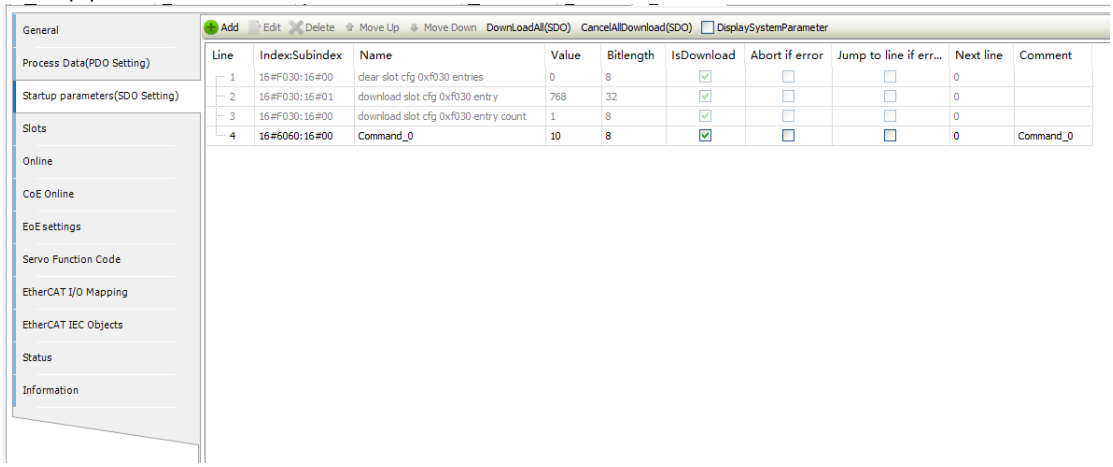


Input/OutPut	Name	Index	SubIndex	Size	Type	Flag	SM
Output	**1st receive PDO Mapping**	**16#1600**	**16#00**	**18.0**		**Editable**	**2**
	Physical outputs	16#60FE	16#01	4.0	UDINT		
	Controlword	16#6040	16#00	2.0	UINT		
	Target position	16#607A	16#00	4.0	DINT		
	Touch probe function	16#60B8	16#00	2.0	UINT		
	Target velocity	16#60FF	16#00	4.0	DINT		
	Target torque	16#6071	16#00	2.0	INT		
Output	258th receive PDO Mapping	16#1701	16#00	12.0		F	
Output	259th receive PDO Mapping	16#1702	16#00	19.0		F	
Output	260th receive PDO Mapping	16#1703	16#00	17.0		F	
Output	261th receive PDO Mapping	16#1704	16#00	23.0		F	
Output	262th receive PDO Mapping	16#1705	16#00	19.0		F	
Input	**1st transmit PDO Mapping**	**16#1A00**	**16#00**	**32.0**		**Editable**	**3**
	Statusword	16#6041	16#00	2.0	UINT		
	Position actual value	16#6064	16#00	4.0	DINT		
	Touch probe status	16#60B9	16#00	2.0	UINT		
	Touch probe pos1 pos value	16#60BA	16#00	4.0	DINT		
	Touch probe pos2 pos value	16#60BC	16#00	4.0	DINT		
	Error code	16#603F	16#00	2.0	UINT		
	Digital inputs	16#60FD	16#00	4.0	UDINT		
	Velocity actual value	16#606C	16#00	4.0	DINT		
	Torque actual value	16#6077	16#00	2.0	INT		
	Following error actual value	16#60F4	16#00	4.0	DINT		
Input	258th transmit PDO Mapping	16#1B01	16#00	28.0		F	
Input	259th transmit PDO Mapping	16#1B02	16#00	25.0		F	
Input	260th transmit PDO Mapping	16#1B03	16#00	29.0		F	
Input	261th transmit PDO Mapping	16#1B04	16#00	29.0		F	

After the change to CST:



- Download SlotsConfig: After the module is configured, the configured slot information needs to be downloaded to the device. After this option is selected, the following parameters are added to the startup parameters:

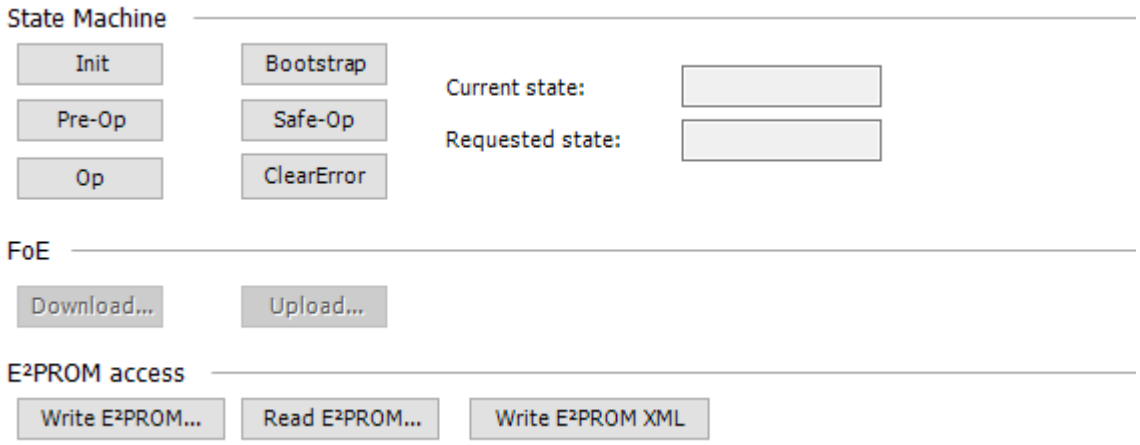


① - Indicates the mode ID of the current module.

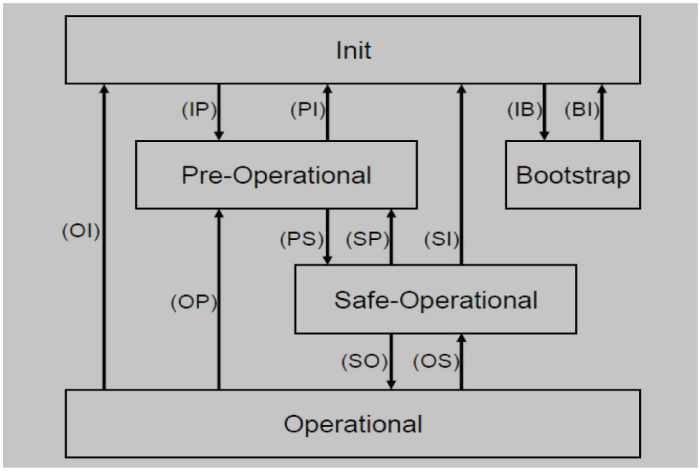
② - Indicates the number of modes of the module to be downloaded

Online

You can use the slave online configuration editor only after logging in to the device. On this page, you can switch the slave state machine manually, read/write the EEPROM of the slave, perform FoE-based upload/download, and slave firmware upgrade.



The EtherCAT state machine coordinates the state relationship between the master and slave applications during initialization and operation.



EtherCAT state conversion order: initialization > pre-operation > safe operation > operation.

The following table lists the state conversion process.

IP	Start Mailbox Communication
PI	Stop Mailbox Communication
PS	Start Input Update
SP	Stop Input Update
SO	Stop Output Update
OS	Stop Output Update
OP	Stop Output Update, Stop Input Update
SI	Stop Input Update, Stop Mailbox Communication
OI	Stop Output Update, Stop Input Update, Stop Mailbox Communication
IB	Start Bootstrap Mode
BI	Restart Device

The "Init", "Pre-Op", "Safe-Op", "Op", and "ClearError" keys can be used for debugging.

EtherCAT file access

To transmit a firmware file to the slave for firmware upgrade, click "Bootstrap" to convert the slave into the "Bootstrap" mode.

By clicking the corresponding button, you can download and upload the firmware file. Then a save or firmware file selection dialog box is displayed. The file is transmitted by using text string and password. The information is provided by the slave, and recorded in the data table of the slave. During firmware upgrade, do not power off the device or switch the state. Perform the operations after the upgrade is complete.

EEPROM access

Slave configuration can be read from EEPROM and written into EEPROM. Similar to firmware file transmission, this operation also displays a dialog box, asking you to save or open the file.

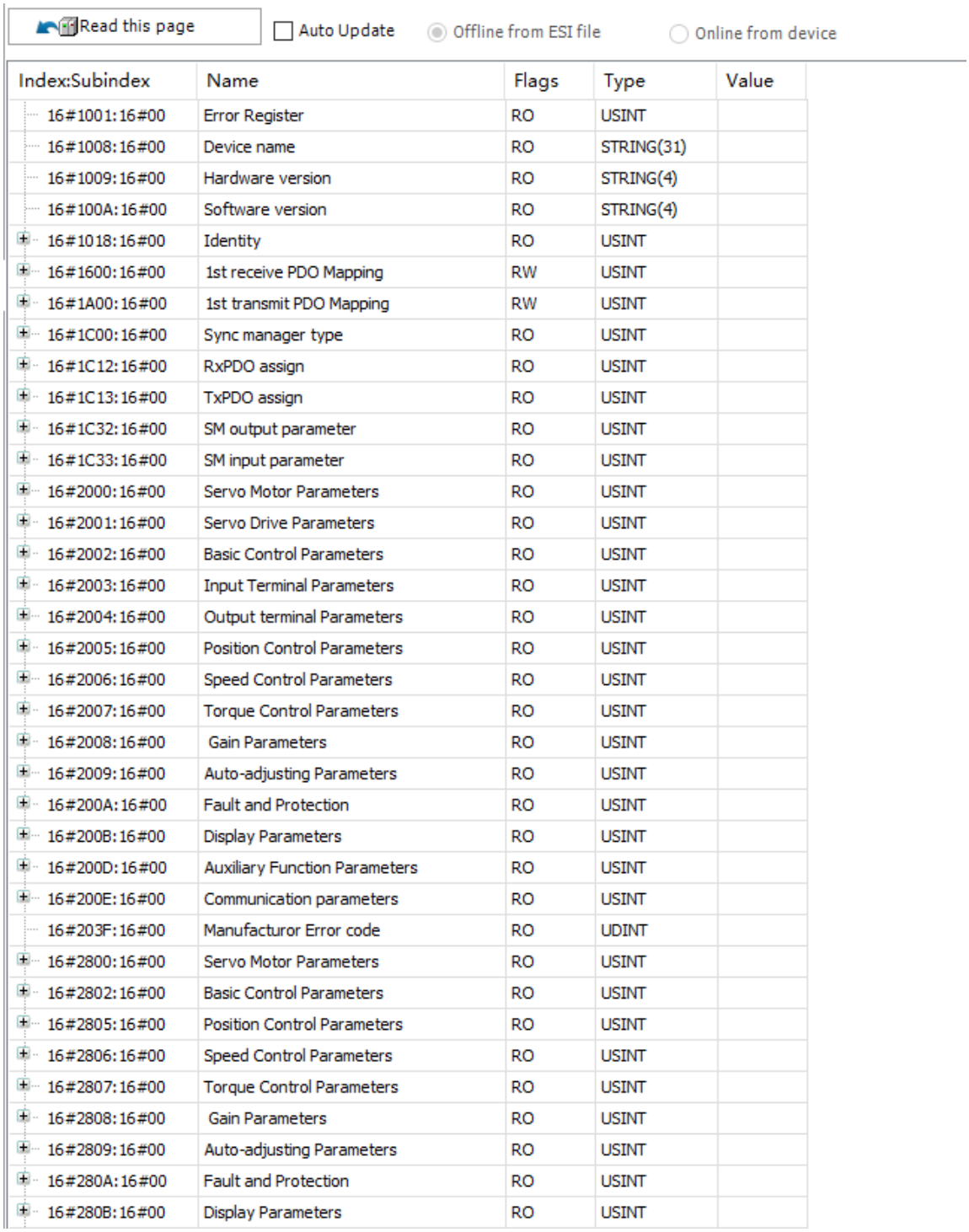
You can run the "write EEPROM XML" command to write the slave configuration to the device through an XML file. This command is valid only when the XML file contains configuration data (in the configuration data part).

ClearError

When the current state has an error, click this button to clear the error.

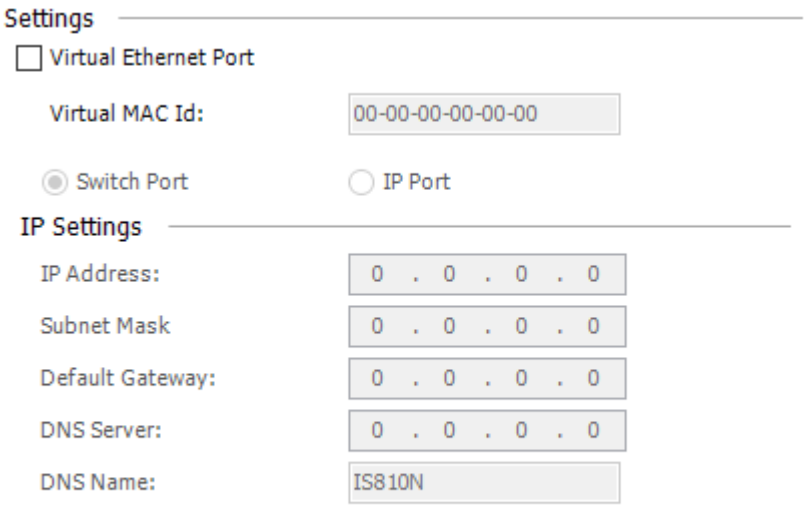
COE Online

The online COE values can be read only when the bus is normal and you have logged in to the PLC, as shown in the following figure.


Read this page
Auto Update
Offline from ESI file
Online from device

Index:Subindex	Name	Flags	Type	Value
16#1001:16#00	Error Register	RO	USINT	
16#1008:16#00	Device name	RO	STRING(31)	
16#1009:16#00	Hardware version	RO	STRING(4)	
16#100A:16#00	Software version	RO	STRING(4)	
16#1018:16#00	Identity	RO	USINT	
16#1600:16#00	1st receive PDO Mapping	RW	USINT	
16#1A00:16#00	1st transmit PDO Mapping	RW	USINT	
16#1C00:16#00	Sync manager type	RO	USINT	
16#1C12:16#00	RxPDO assign	RO	USINT	
16#1C13:16#00	TxPDO assign	RO	USINT	
16#1C32:16#00	SM output parameter	RO	USINT	
16#1C33:16#00	SM input parameter	RO	USINT	
16#2000:16#00	Servo Motor Parameters	RO	USINT	
16#2001:16#00	Servo Drive Parameters	RO	USINT	
16#2002:16#00	Basic Control Parameters	RO	USINT	
16#2003:16#00	Input Terminal Parameters	RO	USINT	
16#2004:16#00	Output terminal Parameters	RO	USINT	
16#2005:16#00	Position Control Parameters	RO	USINT	
16#2006:16#00	Speed Control Parameters	RO	USINT	
16#2007:16#00	Torque Control Parameters	RO	USINT	
16#2008:16#00	Gain Parameters	RO	USINT	
16#2009:16#00	Auto-adjusting Parameters	RO	USINT	
16#200A:16#00	Fault and Protection	RO	USINT	
16#200B:16#00	Display Parameters	RO	USINT	
16#200D:16#00	Auxiliary Function Parameters	RO	USINT	
16#200E:16#00	Communication parameters	RO	USINT	
16#203F:16#00	Manufacturor Error code	RO	UDINT	
16#2800:16#00	Servo Motor Parameters	RO	USINT	
16#2802:16#00	Basic Control Parameters	RO	USINT	
16#2805:16#00	Position Control Parameters	RO	USINT	
16#2806:16#00	Speed Control Parameters	RO	USINT	
16#2807:16#00	Torque Control Parameters	RO	USINT	
16#2808:16#00	Gain Parameters	RO	USINT	
16#2809:16#00	Auto-adjusting Parameters	RO	USINT	
16#280A:16#00	Fault and Protection	RO	USINT	
16#280B:16#00	Display Parameters	RO	USINT	

EOE settings



EtherNET over EtherCAT (EOE) allows any EtherNET device to connect to EtherCAT through a conversion terminal, without affecting the instantaneity of EtherCAT. Similar to popular Internet protocols (such as TCP/IP, VPN, and PPPoE(DSL)), Ethernet frames are transmitted using the EtherCAT protocol. This allows standard network devices to connect to terminals, such as printer and PC, through switches.

For slaves supporting EOE, you can configure the communication function. The preceding page is available only when the device supports EOE.

- Virtual Ethernet Port: Enables the EOE function for the slave. If this option is selected, a special virtual MAC address must be specified.
- Switch Port: Uses this device as an IP port. The Ethernet communication parameters must be set.
- IP Port: Uses this device as an IP port. The Ethernet communication parameters must be set.

Ethernet communication parameters must be set based on the virtual Ethernet adapter parameters. Four bytes are assigned to each of the IP address, subnet mask, and default gateway to identify the slave on the network. When you hover the cursor over the edit area, you can modify the default settings.

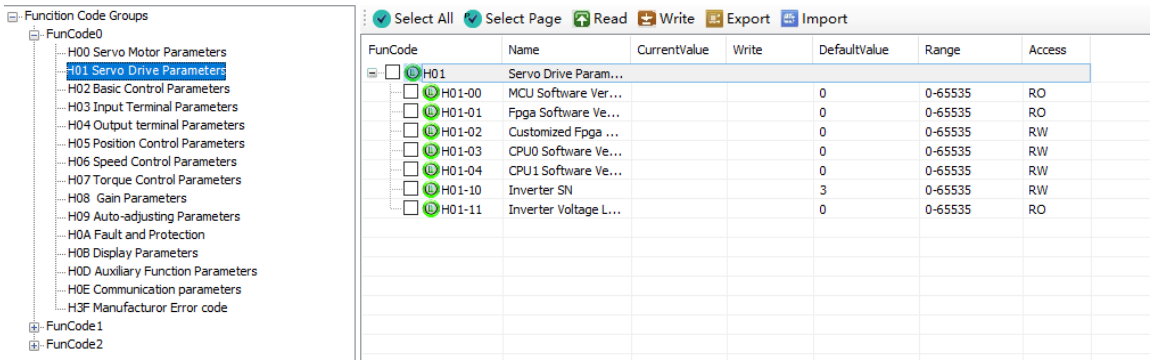
Note

The IP port must be in the same network segment as the virtual Ethernet adapter. For example, if the Ethernet adapter's IP address is 192.168.1.1 and subnet mask is 255.255.255.0, the IP port must be within the range from 192.168.1.2 to 192.168.1.254.

- DNS Server: The IP address of the DNS server.
- DNS Name: The name of the DNS server.

Servo Function Codes

Function codes refer to the vendor-specific parameters of Inovance servo products. On the "Function Code" tab page of the slave, you can read, write, import, and export vendor-specific parameters for debugging and maintenance, as shown in the following figure.



- Select All: Selects all servo function codes or cancels all selections.
- Select Page: Selects all or cancels all selections on the current page.
- Read: Reads the selected parameters only when the slave is in the operation state and the project attribute is "Read".
- Write: Writes the selected parameters only when the slave is in the operation state and the project attribute is "Write".
- Export: Exports the selected function codes.
- Import: Imports function codes.

ESC Register

"ESC Register" is available only when the "Enable Expert Settings" option is selected. It is used to read the ESC chip register address in advanced debugging, as shown in the following figure.

SelectAll CancleAll Read Write Export Import

Selected	Address	Value	Write Value	Flags	Length(Byte)	Description
☐	16#0000			R	2	Revision/Type
☐	16#0002			R	2	Build
☐	16#0004			R	2	SM/FMMU channels
☐	16#0006			R	2	Ports Config/DPRAM Size
☐	16#0008			R	2	Features
☐	16#0010			RW	2	Configured Station Address
☐	16#0012			R	2	Configured Station Alias
☐	16#0020			RW	2	Register Write Protection/Enable
☐	16#0030			RW	2	ESC Write Protection/Enable
☐	16#0040			RW	1	ESC Reset ECAT
☐	16#0041			R	1	ESC Reset PDI
☐	16#0100			RW	2	ESC DL Control_L
☐	16#0102			RW	2	ESC DL Control_H
☐	16#0108			RW	2	Physical RW offset
☐	16#0110			R	2	ESC DL Status
☐	16#0120			RW	2	AL Control
☐	16#0130			R	2	AL Status
☐	16#0134			R	2	AL Status Code
☐	16#0140			R	1	PDI Control
☐	16#0141			R	1	ESC Config
☐	16#0150			R	2	PDI Config_1
☐	16#0152			R	2	PDI Config_2
☐	16#0200			RW	2	ECAT IRQ Mask
☐	16#0204			R	2	AL IRQ Mask_L
☐	16#0206			R	2	AL IRQ Mask_H
☐	16#0210			R	2	ECAT IRQ
☐	16#0220			R	2	AL IRQ_L
☐	16#0222			R	2	AL IRQ_H
☐	16#0300			RW	2	RX Error Counter A
☐	16#0302			RW	2	RX Error Counter B
☐	16#0304			RW	2	RX Error Counter C
☐	16#0306			RW	2	RX Error Counter D
☐	16#0308			RW	2	Forwarded RX Error Counter B/A
☐	16#030A			RW	2	Forwarded RX Error Counter D/C

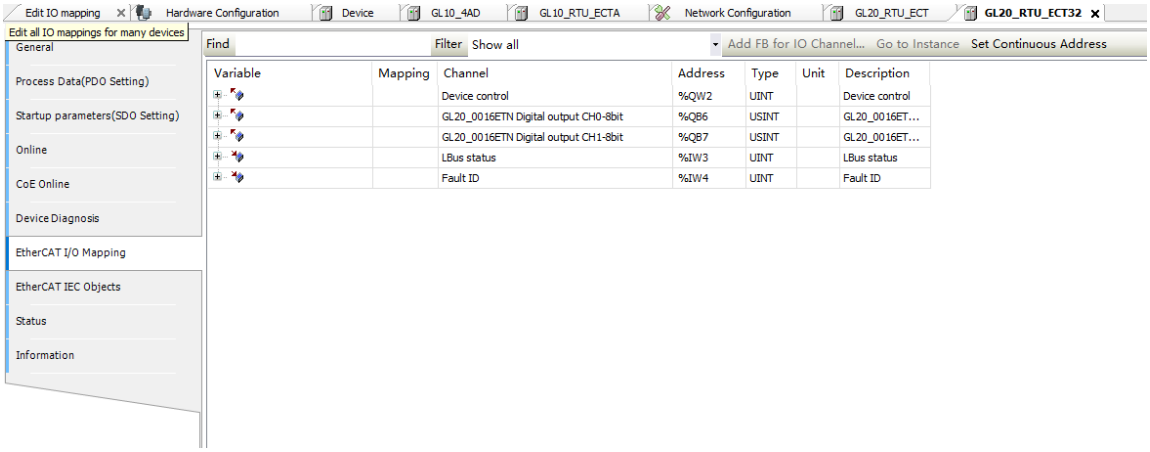
- Select All: Selects all items.
- Cancel All: Cancels all selections.
- Read: Reads option values in the operation state.
- Write: Writes the values carrying "Write" attributes in the operation state.
- Export: Exports the selected items in the XML format.
- Import: Imports a valid XML file. Only the exported XML items are displayed.
- Shortcut menu: Implements conversion among hexadecimal, decimal, and binary formats.

EtherCAT I/O Mapping

This is a tab on the EtherCAT slave configuration editor, in which the ETCSlave instance (variable) and I/O variable defined by the slave are specified for EtherCAT I/O. Therefore, the EtherCAT slave connected to the PLC can be controlled by user program.

For the description of mapping, see the section "I/O Mapping".

The automatically created slave instance is displayed in "IEC Objects" of the master instance at the bottom of the dialog box.



Note

The variables and types of mappings must be consistent.

State

This configuration editor is used to configure the EtherCAT slave, including the NIC and internal bus system status information (such as startup and stop), and diagnosis information of the specified device.

Information

This tab page is provided in EtherCAT master or slave configuration. The configuration of a module includes: Name, Vendor, Categories, Version, Type, Order ID, Description, and Image.

4.4.5 CiA402 Axis

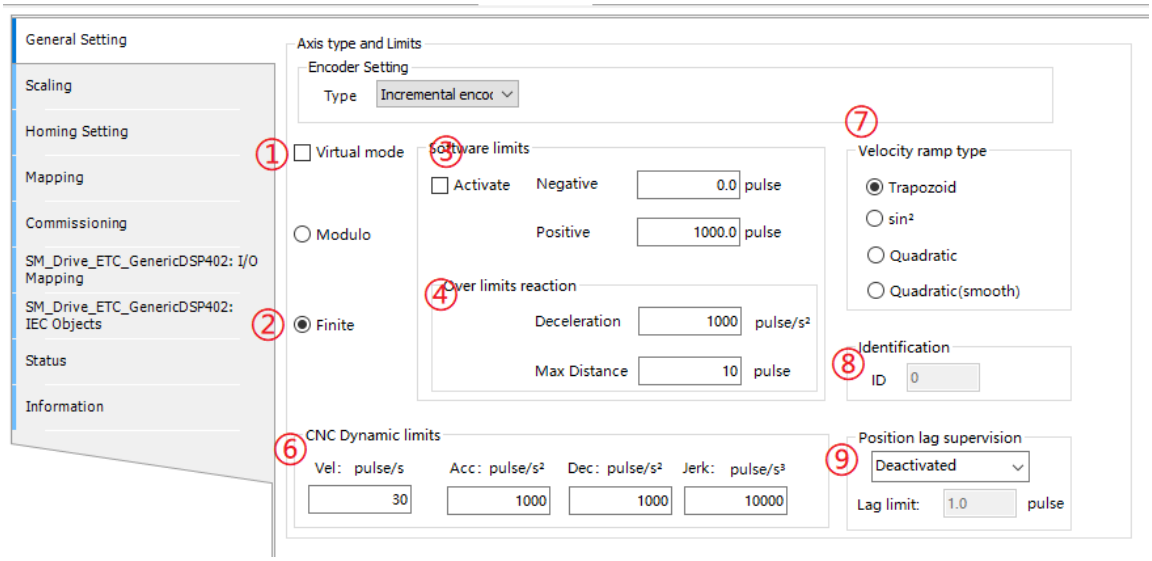
After adding a servo slave, double-click an axis. The axis configuration page is displayed. The following is the description of the options on the axis configuration page from top to bottom.

General Setting

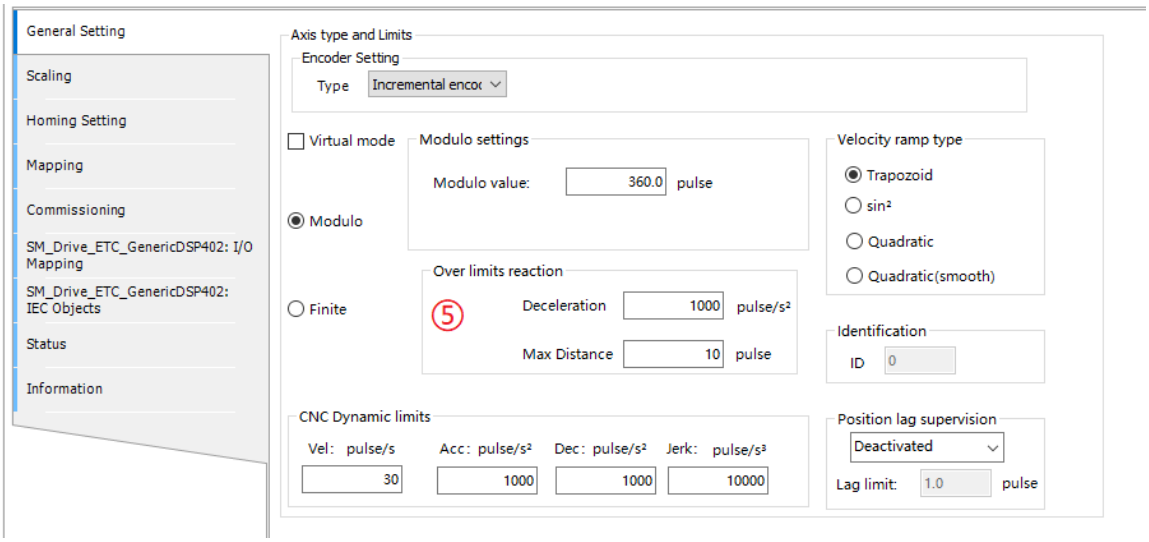
Axis general settings include virtual axis and physical axis. The definitions are as follows.

Type	Function
Virtual mode	In this mode, the device runs without a physical servo or motor. It obtains desired parameters by simulated operation. This mode is not affected by external environment.
Physical mode	In this mode, the device must run with a servo and motor. Some parameters can be obtained only in the physical mode, for example, online COE. This mode may be affected by external environment, for example, the trace operation.

Settings of virtual and physical axes



Settings of modulo and finite modes

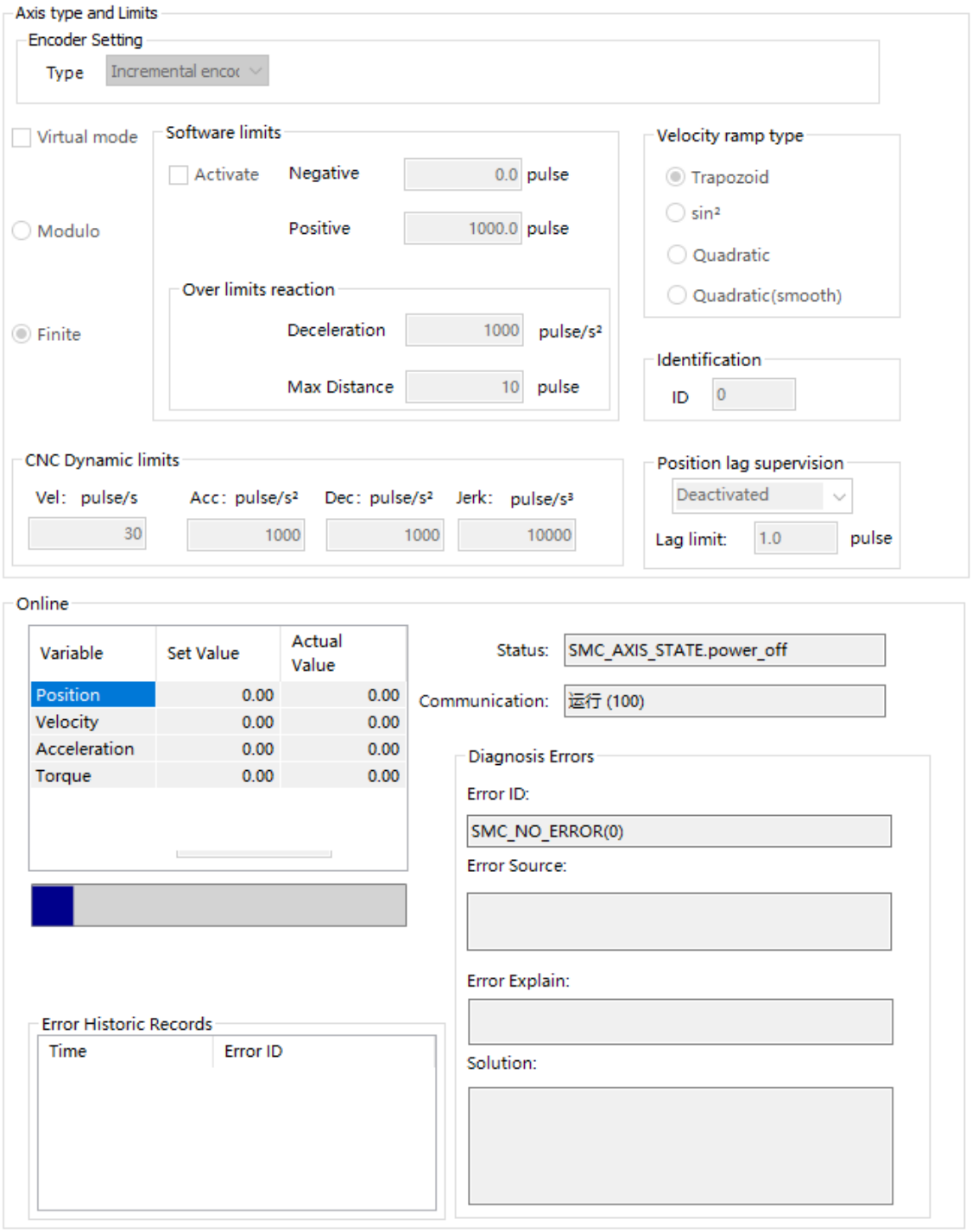


The following table lists functions on the page.

No.	Option	Function
1	Virtual mode	If this option is selected, the virtual axis mode is used; otherwise, the physical axis mode is used.
2	Axis type	Modulo mode: Axis locations are added or reduced in modulo way. Finite mode: Axis locations are added or reduced within a specified range.
3	Software limits	After this option is selected, the negative and positive location limits on axis are applied to the modulo mode.

No.	Option	Function
4	Software error reaction	It is relevant to software limits, and is effective only when the software limits function is enabled. After this option is selected, if the axis location exceeds the software limit, the software reacts in response to the error. That is, it decelerates to the maximum distance.
5	Modulo settings	It applies to the finite mode to limit the finite cycle. This parameter is associated with the "command pulse count per motor rotation" parameter on the "Scaling" page, the homing parameters on the "Homing Setting" page, the maximum velocity parameter on the "Mapping/Other Setting" page. When setting this parameter, note the associated parameter settings. If the associated parameter settings do not match this parameter, an error will be reported and the correct parameter values will be displayed.
6	CNC Dynamic limits	Applied to the settings of the CNC function axis.
7	Velocity ramp type	Applied to the axis velocity change track.
8	Identification	The external ID of the axis.
9	Position lag supervision	The axis operation mode when the position lags.

The following figure shows the axis operation in the modulo mode after the servo starts, including the real-time position, velocity, acceleration rate, torque, and communication status. If an error occurs, the error information is displayed.

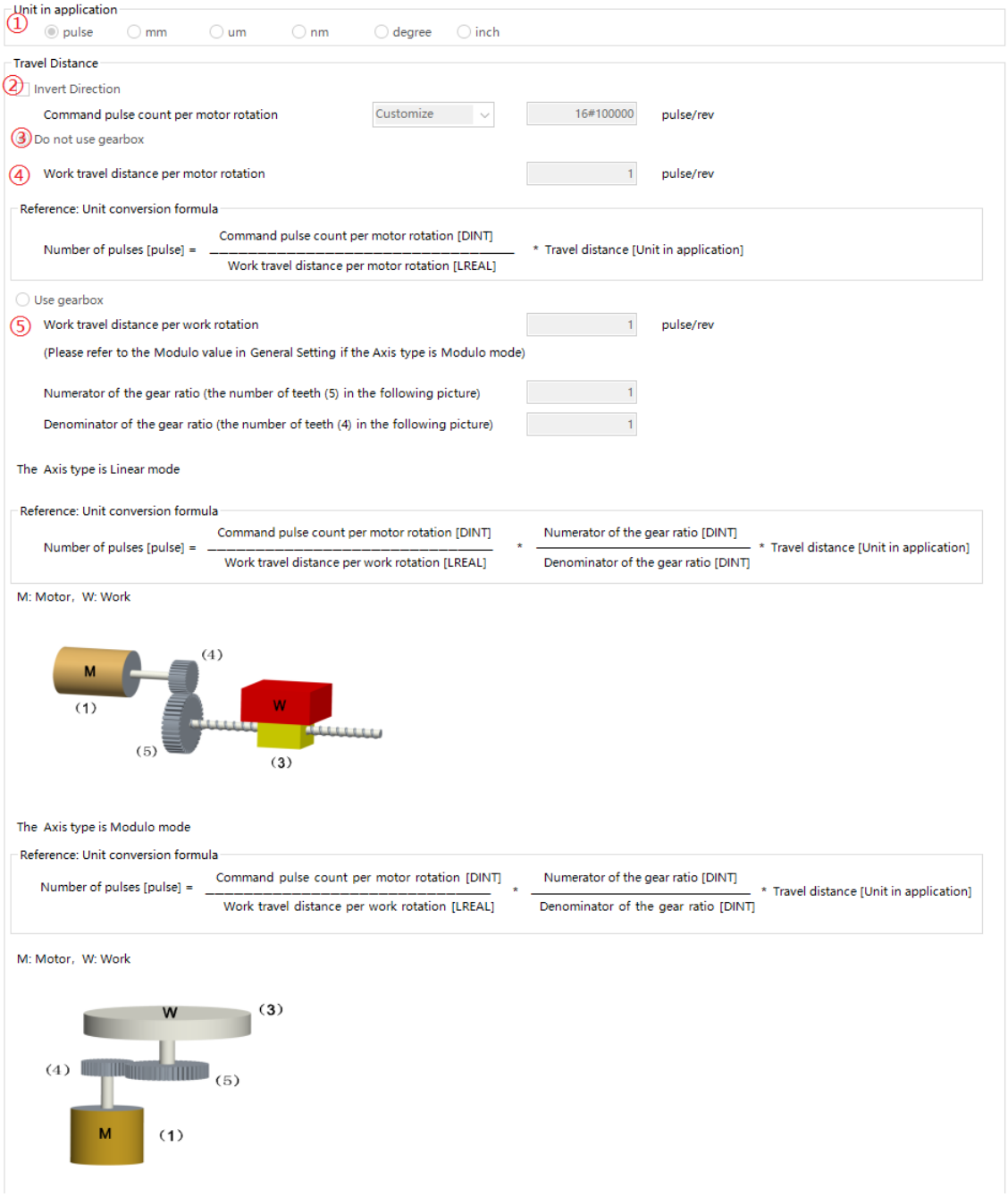


Note

If the encoder location information is lost, check whether the encoder battery is properly connected and fully charged. Store the battery in an environment within the specified temperature range.

Scaling

On this page, you can calculate the number of pulses by setting the related parameters.



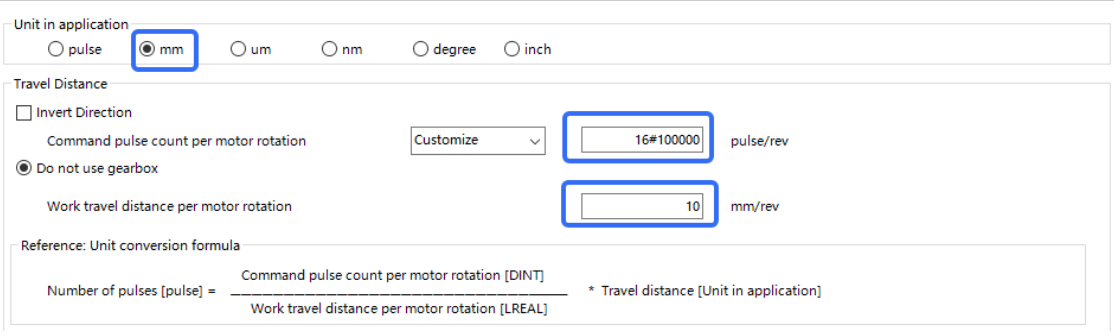
The following table lists options and functions in the previous figure.

No.	Option	Function
1	Unit in application	Sets the length unit as needed. After a unit is selected, the unit settings in axis general settings, scaling, homing settings, and mapping/other setting modules are also changed.
2	Invert Direction	Enables the axis to run in the invert direction
3	Command pulse count per motor cycle	Sets the encoder resolution of the motor, namely, the number of pulses required for a motor to rotate a cycle. The default value is 1048576 (Inovance 20-bit encoder).

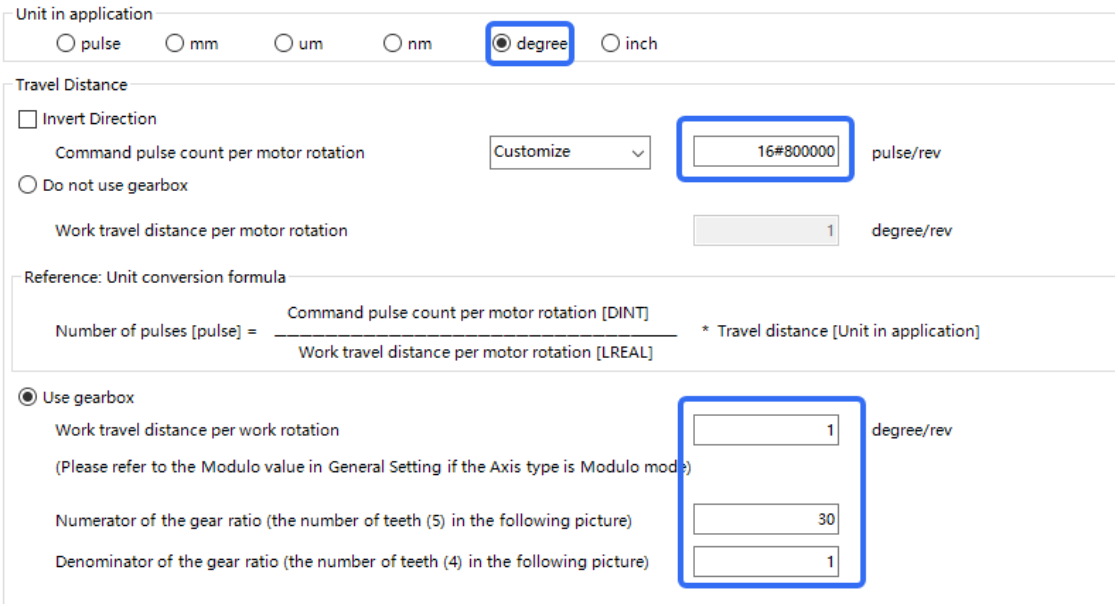
No.	Option	Function
4	Do not use gearbox	Sets the work travel distance per motor rotation according to the device situation. Work travel distance per motor rotation: travel distance (unit in application) of the worktable (such as belt pulley, gear, and reducer) from the end of machinery per rotation. The number of pulses can be calculated by referencing the unit conversion formula.
5	Use gearbox	Sets the work travel distance, numerator of gear ratio, and denominator of gear ratio per motor rotation according to the device situation. Work travel distance per motor rotation: travel distance (unit in application) of the worktable from the end of machinery per rotation; numerator of gear ratio: number of teeth of the worktable; denominator of gear ratio: number of teeth of the motor gear. The number of pulses can be calculated based on the axis type selected in axis basic settings and the corresponding reference unit. Note: The denominator and numerator of gear ratio can be scaled up and down by ratio.

Application Example

1. The motor directly drives the screw rod to move, and the motor moves 10 mm per circle around the screw rod. The motor is Inovance IS620N incremental motor (20-bit encoder resolution ratio). The configuration is as follows:



2. The motors are connected by driving turnplate. The reduction ratio between motor and turnplate is 30:1. (If the number of motor gear teeth is 1, the number of worktable teeth is 30. That is, when the worktable gear rotates 1 round, the motor gear rotates 30 rounds.) The travel distance of the turnplate is 0-360 degrees. The motor is the Inovance IS620N absolute motor (23-bit encoder resolution). The configuration is as follows:

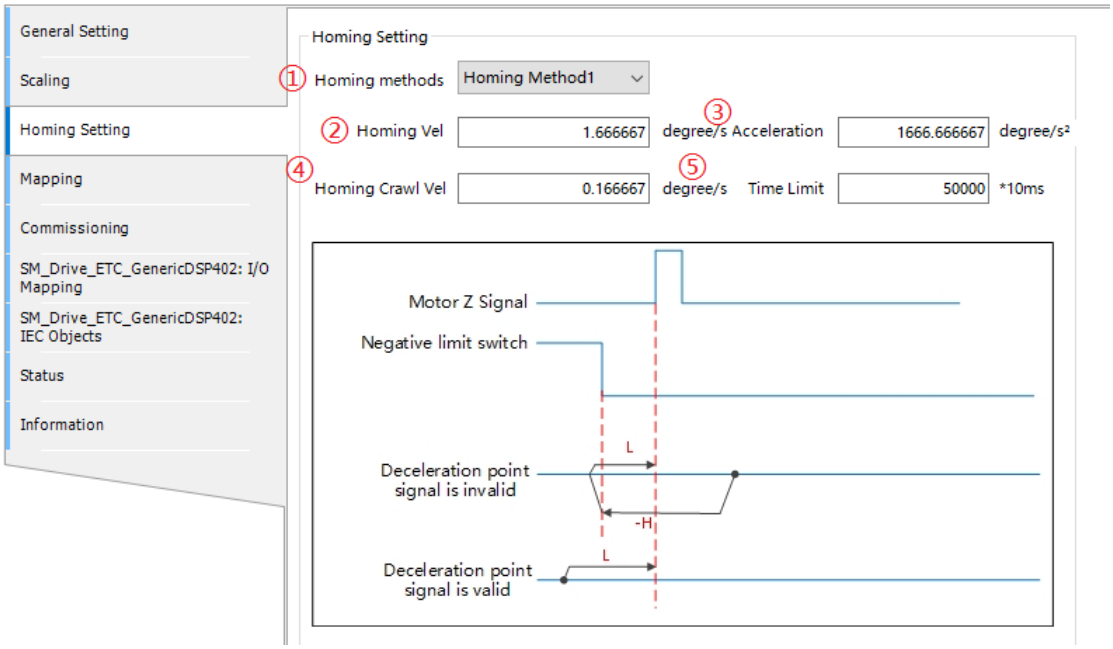


Note

After the numerator of gear ratio, denominator of gear ratio, and work travel distance per motor rotation are modified, the axis parameters on other pages are affected. Therefore, you need to modify them accordingly.

Homing Setting

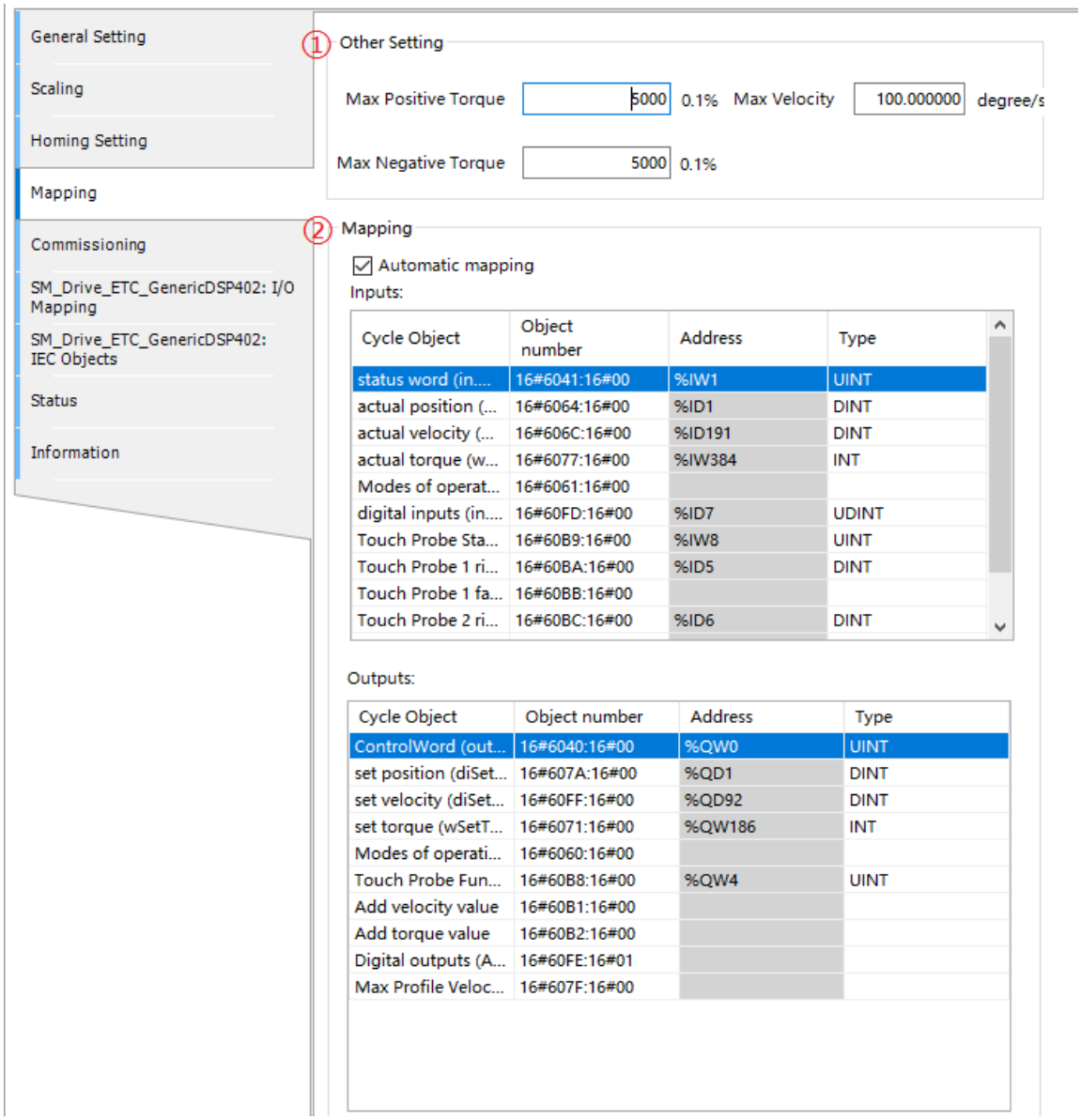
Homing settings include the graphic parameter settings for axis homing, as shown in the following figure. This page provides graphic setting instruction, so you can select the homing mode from the drop-down list without reading the servo guide. In this way, you can visibly and easily complete parameter settings, as shown in ① in the following figure.



The following table lists options and functions in the previous figure.

No.	Option	Description
1	Homing methods	The homing method of the drive. A total of 35 options are supported (the actual homing method is determined by the drive). The diagrams below vary with the homing method. Select a homing method as needed.
2	Homing Vel	The high velocity of the axis when searching for the signal of the deceleration point, for example, H in the figure. Its value is a floating point number with a minimum retention of 6 decimal places, compatible with the system of commas as decimal points.
3	Acceleration	The acceleration rate of the axis when searching for the velocity change. Its value is a floating point number with a minimum retention of 6 decimal places, compatible with the system of commas as decimal points.
4	Homing Crawl Vel	The low velocity of the axis when searching for the homing, for example, L in the figure. Its value is a floating point number with a minimum retention of 6 decimal places, compatible with the system of commas as decimal points.
5	Time Limit	The total homing time. If timeout, an alarm is reported. It is the maximum time allowed for the axis performing the homing operation. If homing times out, the axis homing operation fails.

Mapping/Other Setting



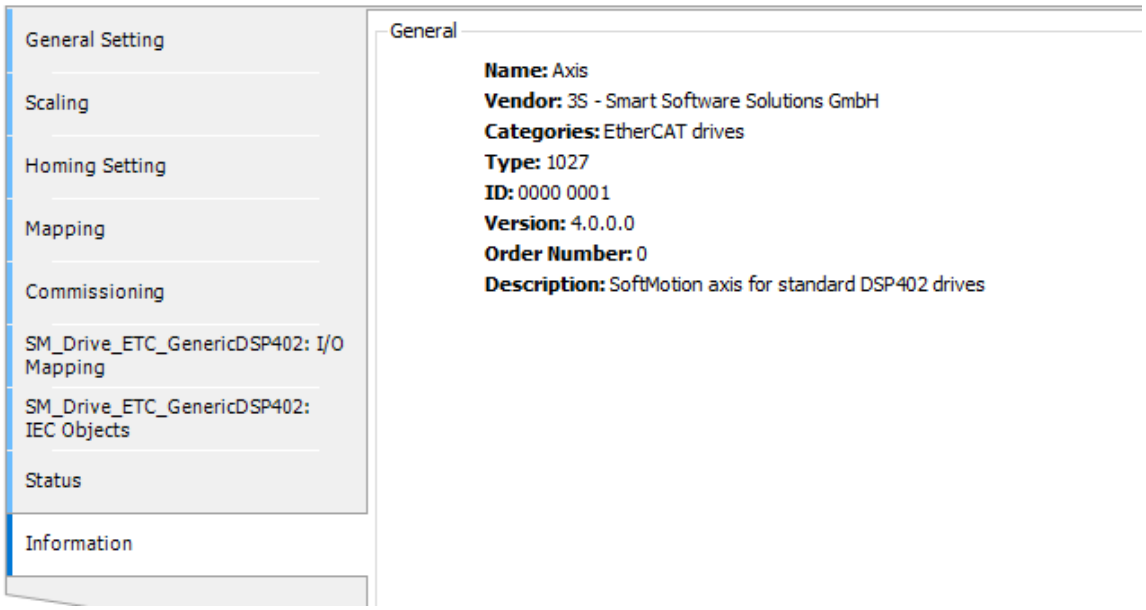
The following table lists options and functions in the previous figure.

No.	Option	Description
1	Other Setting	Sets the maximum values and velocities of positive and negative torques. Max Positive Torque/Max Negative Torque: Torque instruction limits set for protecting the drive. When the drive torque value is larger than the limit, the actual drive torque value is changed to be consistent with the limit. Max Velocity: Limits the velocity within the specified limit. If the torque value is larger than the mechanical load torque, the motor keeps speeding up, and overspeed may occur, which damages the mechanical device. After a velocity limit is set, the actual velocity will be kept within the limit. Its value is a floating point number with a minimum retention of 6 decimal places, compatible with the system of commas as decimal points. Note: The maximum velocity cannot be set to 0; otherwise, an error may occur when the axis runs.
2	Mapping	When "Automatic mapping" is selected, the slave is associated with the axis. The slave data is directly mapped to the axis. If this option is not selected, you can manually modify the address in the axis mapping data. Specifically: The input format is %I + Type letters + Arabic numbers. The output format is %Q + Type letters + Arabic numbers. Type letters (bytes occupied by the type) include SINT-B, UINT-W, DINT-D, and UDINT-D. When a compiling error is reported during manual address input, delete the input address and enter the correct address. Note: The input address must be quoted by single quotes. If a compiling error is reported by the display is normal, delete the displayed result and enter the address in correct format.

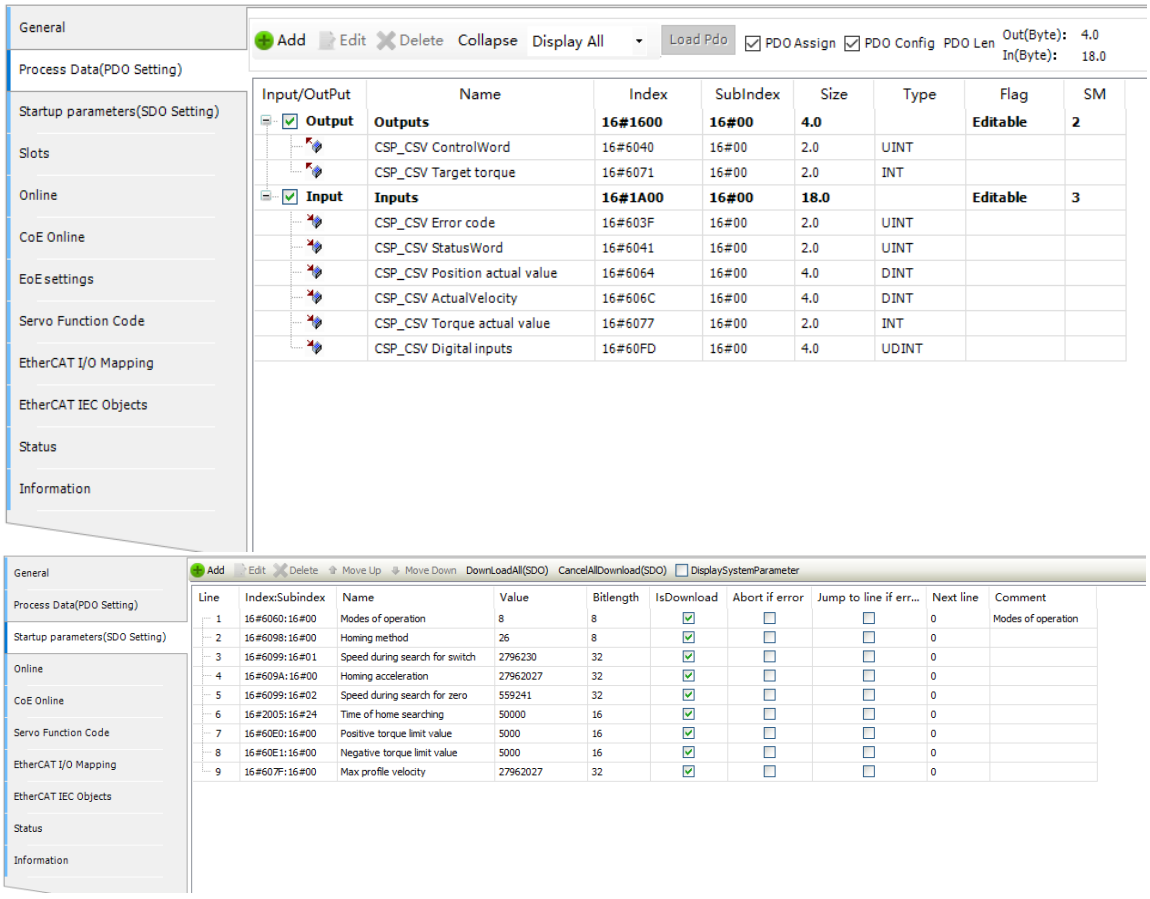
Note: To modify the mapping data manually with "Automatic mapping" selected, right-click the target PDO and click "Delete" in the shortcut menu, or select the target PDO and press the "Delete" key on the keypad. If an item is deleted by mistake, restore the default mapping table to undo the operation.

Information

This page displays the axis basic settings, including Name, Vendor, Group, Categories, ID, Version, Module Number, and Description.



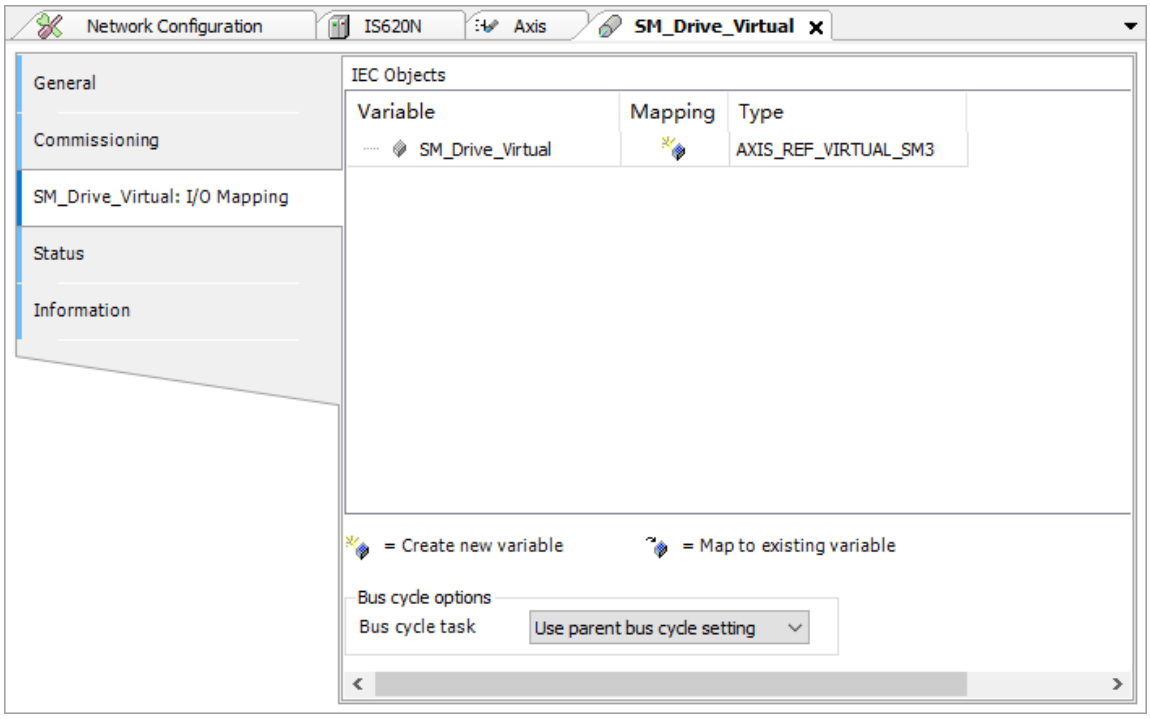
If the axis parameter settings are not modified, the default parameter settings of PDO and SDO are also retained. If the axis parameter settings have been modified, the default parameter settings of PDO and SDO are also modified.



4.4.6 Virtual Axis

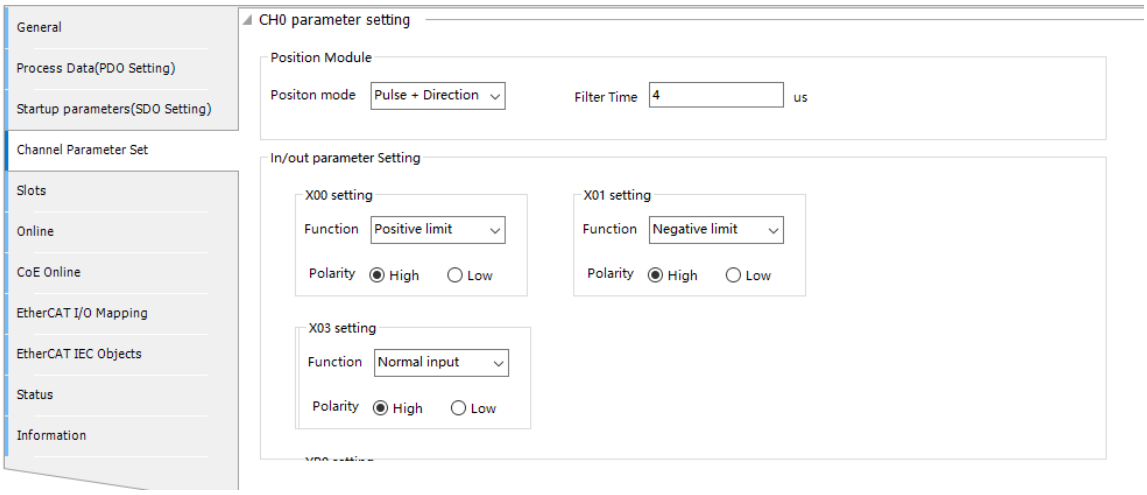
The virtual axis page is similar to the EtherCAT configuration page. Note that when multiple EtherCAT masters are enabled and the virtual axis is enabled (by choosing "Axis Pool" > "Virtual Axis"), the bus cycle task of the virtual axis must be bound to the called EtherCAT task (The default setting is "Use parent bus cycle setting". This only applies to the single-master scenarios.) One virtual axis cannot be called by multiple EtherCAT tasks; otherwise, a running error will occur.

The settings in red box in the following figure are bus cycle task settings.



4.4.7 GR10-4PME Positioning Module

The positioning module pages are the same as CiA402 axis settings pages, except that the following page is added:



The GR10-4PME module is a pulse positioning module that has four high-speed output channels. It implements speed and position control for the pulse servos and stepping drives that use pulse as signals. This page allows you to set the parameters of the GR10-4PME module. Taking the first channel of GR10-4PME as an example, the following functions can be set on this page.

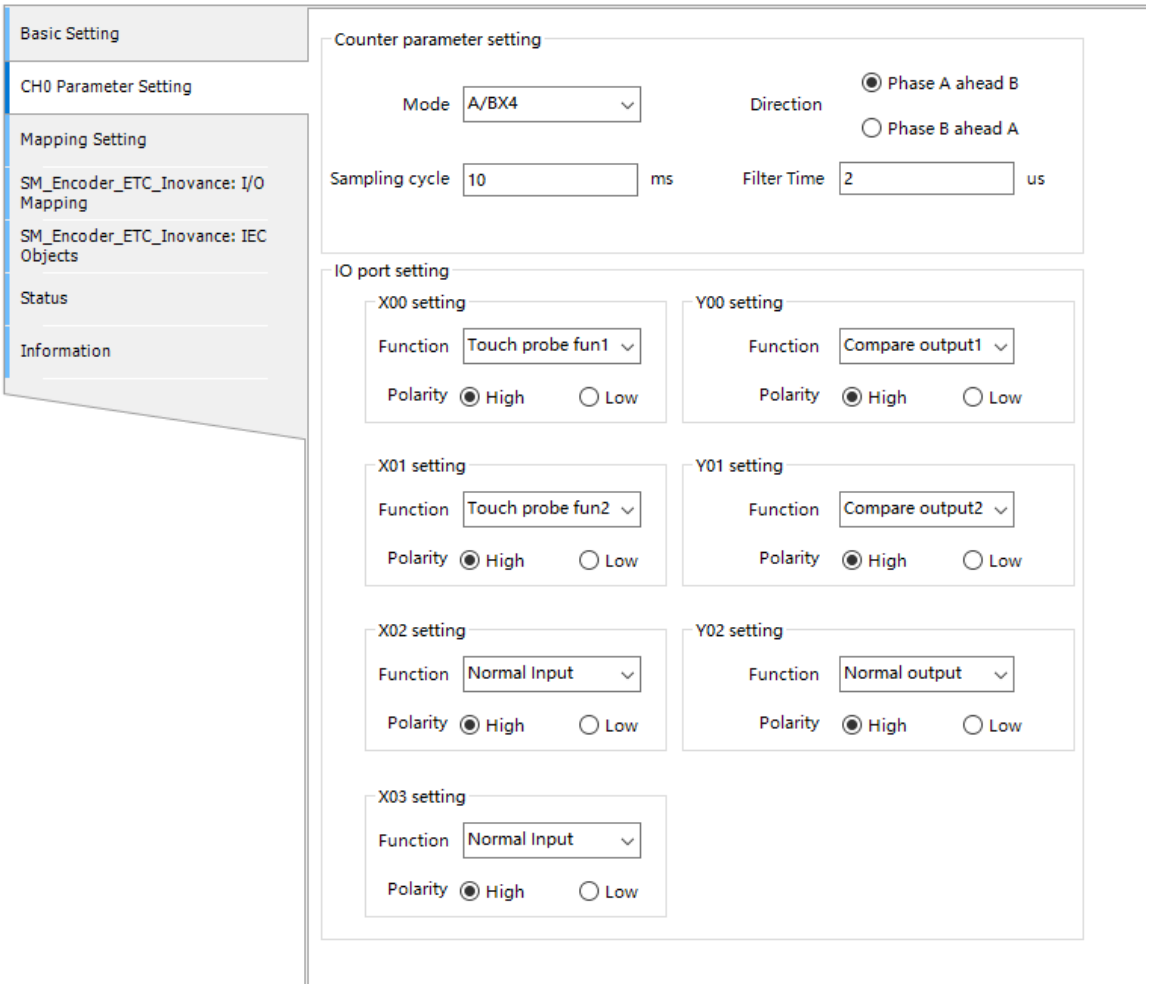
Name		Description	Default Value
Position Mode		Indicates the output pulse type on the high-speed pulse output port. Phase A/B single frequency Pulse+Direction CW/CCW	Pulse +Direction
Filter Time		Indicates the pulse input filter time of the digital input terminal.	4 μs
X00 Setting	Function	Selects the functions of the X00 digital input terminal. General input Emergency stop switch Positive limit	Positive limit
	Polarity	Selects the effective level logic of the X00 digital input. High - Input high level is effective. Low - Input low level is effective.	High
X01 Setting Polarity	Function	Selects the functions of the X01 digital input terminal. General input Emergency stop switch Negative limit	Negative limit
	Selects the effective level logic of the X01 digital input. High - Input high level is effective. Low - Input low level is effective.	High	
X02 Setting Polarity	Function	Selects the functions of the X02 digital input terminal. General input Emergency stop switch Home switch	Home switch
	Selects the effective level logic of the X02 digital input. High - Input high level is effective. Low - Input low level is effective.	High	
X03 Setting Polarity	Function	Selects the functions of the X03 digital input terminal. General input Emergency stop switch	General input
	Selects the effective level logic of the X03 digital input. High - Input high level is effective. Low - Input low level is effective.	High	

Name		Description	Default Value
YR0 Setting	Function	Sets the YR0 function of the digital output terminal. Normal digital output terminal Enable servo (output signal)	Enable servo (output signal)
Polarity	Energizes or de-energizes the digital output terminal YR0. High - Energized when control is set to 1. High - De-energized when control is set to 0.	High	

For the applications of the positioning module, see "EtherCAT Remote Communication Application Guide".

4.4.8 GR10-2HCE counter module

The counter module pages are the same as CiA402 axis settings pages, except that the following page is added:



The GR10-2HCE module is a pulse counter module that has two high-speed input channels. It implements pulse counting and frequency measurement in A/B phase pulse, pulse+direction, and CW/

CCW modes. This page allows you to set the parameters of the GR10-2HCE module. Taking the first channel of GR10-2HCE as an example, the following functions can be set on this page.

Name		Description	Default Value
Mode		Selects the input mode of channel input pulse.	Phase A/B quadruple frequency
		Phase A/B single frequency	
		Phase A/B double frequency	
		Phase A/B quadruple frequency	
		Pulse+Direction	
		CW/CCW	
Sampling cycle		Calculates sampling cycle by input filter frequency.	10 ms
Filter Time		Sets the sampling filter of the pulse input channel and digital input channel.	2 μs
Direction	Phase A ahead B	Phase A/B	Phase A ahead B
		The counter is increased when phase A is ahead of phase B.	
		Pulse+Direction	
		The counter is increased when the phase B input level is high.	
		CW/CCW	
		The counter is increased when phase A has counts.	
	Phase B ahead A	Phase A/B	
		The counter is increased when phase B is ahead of phase A.	
		Pulse+Direction	
		The counter is increased when the phase B input level is low.	
		CW/CCW	
		The counter is increased when phase B has counts.	
X00 Setting	Function	Selects the functions of the X00 digital input terminal.	Probe 1
		General input	
		Probe 1	
		Reset counter to 0	
		Preset counter	
		Door control	
	Polarity	Selects the effective level logic of the X00 digital input.	High
		High - Input high level is effective.	
		Low - Input low level is effective.	

Name		Description	Default Value
X01 Setting Polarity	Function	Selects the functions of the X01 digital input terminal.	Probe 2
		General input	
		Probe 2	
		Reset counter to 0	
		Preset counter	
		Door control	
	Selects the effective level logic of the X01 digital input.	High	
	High - Input high level is effective.		
	Low - Input low level is effective.		
X02 Setting Polarity	Function	Selects the functions of the X02 digital input terminal.	General input
		General input	
		Reset counter to 0	
		Preset counter	
		Door control	
	Selects the effective level logic of the X02 digital input.	High	
	High - Input high level is effective.		
	Low - Input low level is effective.		
X03 Setting Polarity	Function	Selects the functions of the X03 digital input terminal.	General input
		General input	
		Reset counter to 0	
		Preset counter	
		Door control	
	Selects the effective level logic of the X03 digital input.	High	
	High - Input high level is effective.		
	Low - Input low level is effective.		
Y00 Setting Polarity	Function	Sets the Y00 function of the digital output terminal.	Comparison output 1
		Normal output	
		Comparison output 1	
	Energizes or de-energizes the digital output terminal Y00.	High	
	High - Energized when control is set to 1.		
	High - De-energized when control is set to 0.		

Name		Description	Default Value
Y01 Setting Polarity	Function	Sets the Y01 function of the digital output terminal.	Comparison output 2
		Normal output	
		Comparison output 2	
	Energizes or de-energizes the digital output terminal Y01.	High	
	High - Energized when control is set to 1.		
	High - De-energized when control is set to 0.		
Y02 Setting Polarity	Function	Sets the Y02 function of the digital output terminal.	Normal output
		Normal output	
	Energizes or de-energizes the digital output terminal Y02.	High	
	High - Energized when control is set to 1.		
	High - De-energized when control is set to 0.		

For the applications of the positioning module, see "EtherCAT Remote Communication Application Guide".

4.4.9 Splitter

Overview

The splitter module is used to expand EtherCAT ports, as shown in the following figure.

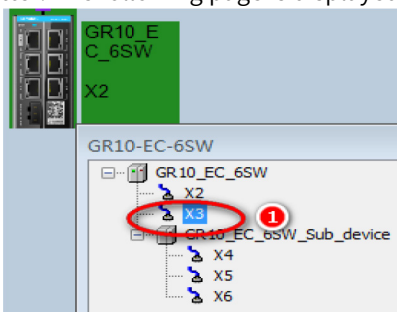


Interface	Option	Function
IN1	Splitter input port	The splitter input port is connected to the EtherCAT output port of the device.
X2 to X6	Splitter output ports	X2 to X6 are splitter output ports independent of each other. Each output port can connect to one EtherCAT slave.

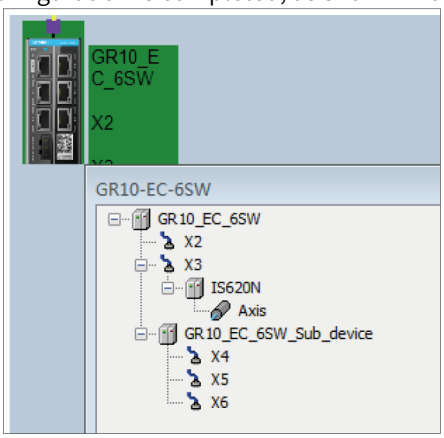
Adding a Splitter and Adding Slaves to the Splitter

You can add, delete, copy, and paste splitters in the same way as that for common slaves (see *" Configuring a PLC as a Master or a Slave" on page 67* and *" Configuration Device Common Operations" on page 73*). After a splitter is added, perform the following operations to add its slaves:

1. Double-click a configuration splitter. The following page is displayed.



2. Select a node consistent with the physical configuration, and then select a device from the network device list. The splitter slave configuration is completed, as shown in the following figure.



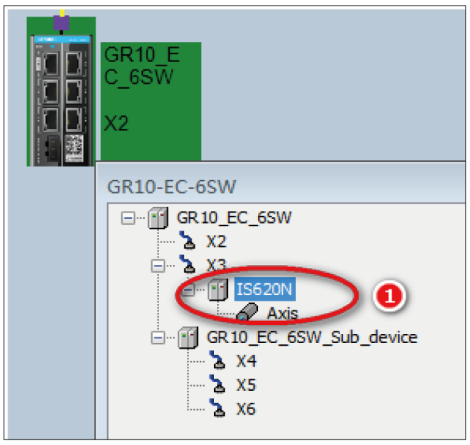
You can run the scan device command (see *" Scanning a Device" on page 132*) to scan the splitter and its slaves, and copy the detected information to the project. The system then automatically completes the device configuration.

Deleting a Splitter and Its Slaves

The splitter deletion operation is similar to the slave deletion operation (see *" Configuration Device Common Operations" on page 73*).

Perform the following operations to delete slaves of a splitter:

1. Access the splitter configuration page.
2. Select the target slave (numbered 1 in the following figure).



Press "Delete".

Note

The deletion operation cannot be canceled. To use the device again, you need to add the device configuration again.

4.4.10 I/O Module

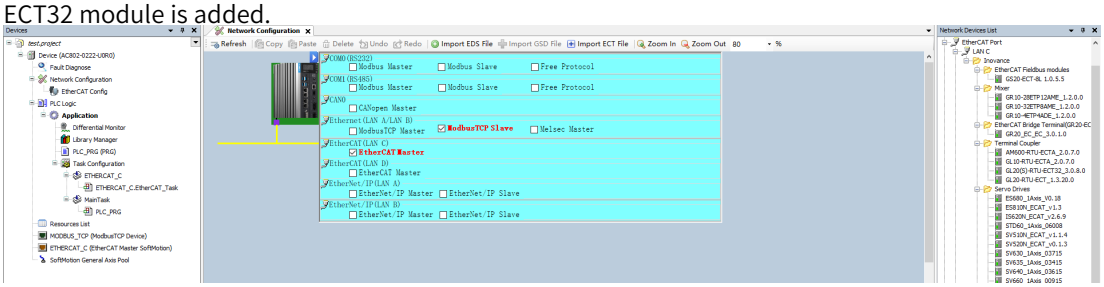
Adding an I/O module

The I/O module addition operations of communication interface modules AM600-RTU-ECTA, GL10-RTU-ECTA, GL20-RTU-ECT, and GL20-RTU-ECT32 are similar. The following takes GL20-RTU-ECT32 as an example to describe how to add an I/O module.

1. Enable the host EtherCAT master and add the GL20-RTU-ECT32 module.

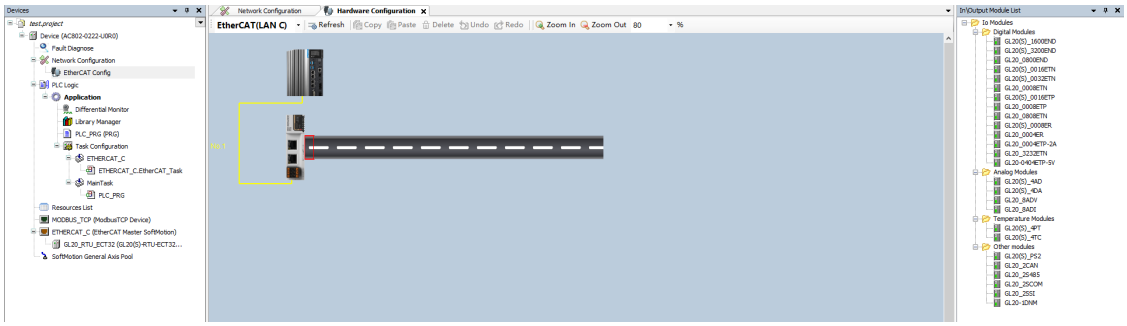
   a. In the left device tree, double-click "Network Configuration", and select the EtherCAT master corresponding to the physical network port of the host connected to the GL20-RTU-ECT32 module to enable the host EtherCAT master.

   b. In the network device list on the right, double-click "GL20-RTU-ECT32_***X.X.X.X***". The GL20-RTU-ECT32 module is added.



2. Add the module.

   In the left device tree, double-click "EtherCAT Config". In the module list on the right, double-click the *XX* module. The *XX* module is added.



Note

Another module addition method: Right-click "GL20_RTU_ECT32" under "ETHERCAT (EtherCAT Master SoftMotion)" in the left device tree. In the shortcut menu displayed, click "Add Device". On the page displayed, select "GL20-*XX*" and then click "Add Device".

3. Configure the module parameters. For details, see the *user guide* of the corresponding module.

Note

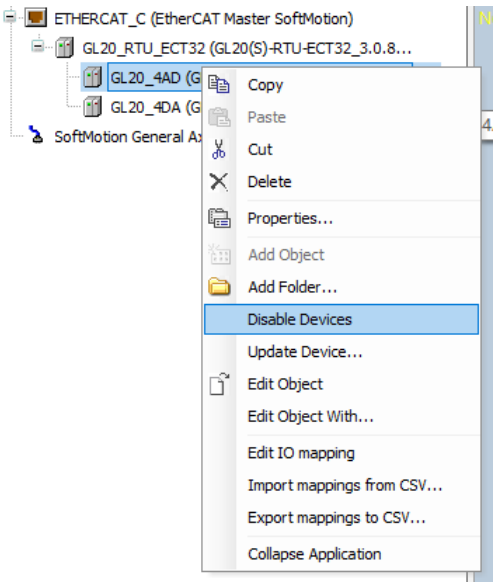
For how to configure the GL10-series modules, see "*EtherCAT Remote Communication Application Guide*".

Disabling a GL20-series I/O module

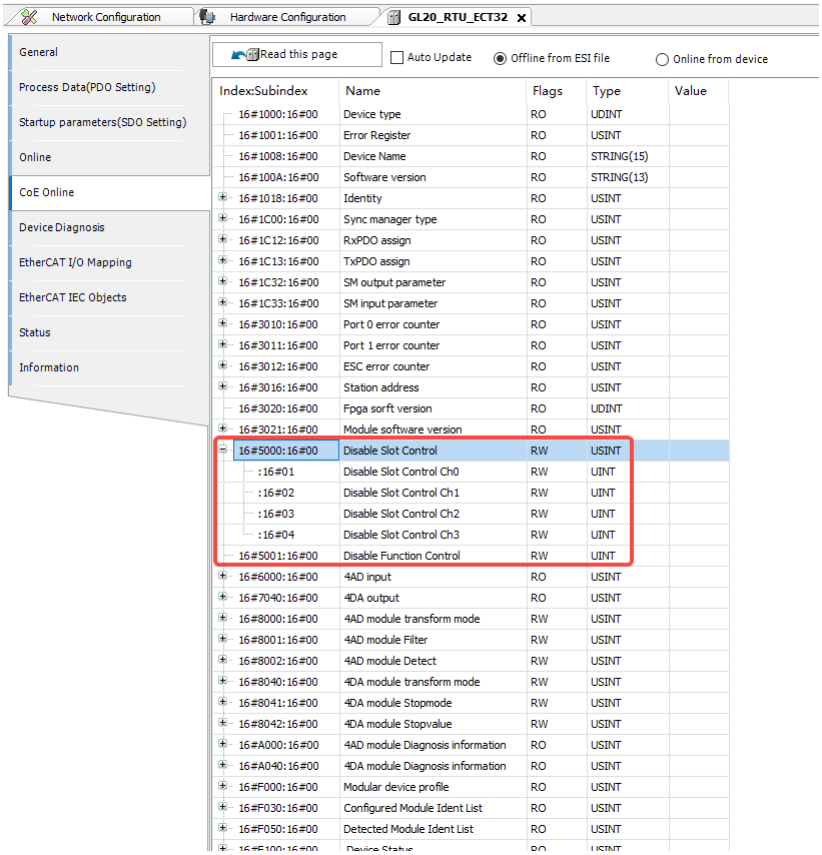
Caution

This function is supported in InoProShop V1.7.3 SP4 and later versions as well as the XML files of the versions 2.0.2.0 and later of the GL20-RTU-ECT32 module. This function is not supported by the GL20-RTU-ECT module.

Right-click a GL20-series I/O module. In the shortcut menu displayed, click "Disable Devices". The module is disabled. The following figure illustrates this process using the GL20-4AD module as an example.



You can use the object dictionary to disable GL20-series I/O modules and view the state through "Online CoE", as shown in the following figure.



The following table lists the object dictionaries and their descriptions.

Object Dictionary	Data Type	Function
16#5001:16#00	UINT	Indicates whether a module is disabled. • 0: No disabled module • 1: Disabled module existing
16#5000:16#01	UINT	Indicates the disabling state of slots 1 to 16. Each bit corresponds to a slot. Mapping between bits and slots: Slot 16 | Slot 15 | Slot 14 | ... | Slot 3 | Slot 2 | Slot 1 • Bit value being 0: The slot is not disabled. • Bit value being 1: The slot is disabled.
16#5000:16#02	UINT	Indicates the disabling state of slots 17 to 32. Each bit corresponds to a slot. Mapping between bits and slots: Slot 32 | Slot 31 | Slot 30 | ... | Slot 19 | Slot 18 | Slot 17 • Bit value being 0: The slot is not disabled. • Bit value being 1: The slot is disabled.

<table>
<tr><th>Object Dictionary</th><th>Data Type</th><th>Function</th></tr>
<tr><td>16#5000:16#03</td><td>UINT</td><td>Indicates the disabling state of slots 33 to 48. Each bit corresponds to a slot.
Mapping between bits and slots:
<table><tr><td>Slot 48</td><td>Slot 47</td><td>Slot 46</td><td>…</td><td>Slot 35</td><td>Slot 34</td><td>Slot 33</td></tr></table>• Bit value being 0: The slot is not disabled.
• Bit value being 1: The slot is disabled.</td></tr>
<tr><td>16#5000:16#04</td><td>UINT</td><td>Indicates the disabling state of slots 49 to 64. Each bit corresponds to a slot.
Mapping between bits and slots:
<table><tr><td>Slot 64</td><td>Slot 63</td><td>Slot 62</td><td>…</td><td>Slot 51</td><td>Slot 50</td><td>Slot 49</td></tr></table>• Bit value being 0: The slot is not disabled.
• Bit value being 1: The slot is disabled.</td></tr>
</table>

4.4.11 Library (Implicit Variables)

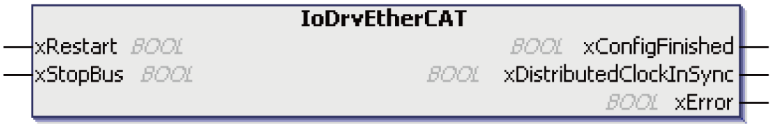
Master Implicit Instance

An IoDrvEtherCAT implicit instance is created as long as the EtherCAT master is inserted into the device list. The instance name is the same as the device name in the device list.

Note

Implicit instances are generated automatically by the system. You cannot define an implicit instance in the program; otherwise, the PLC cannot run normally.

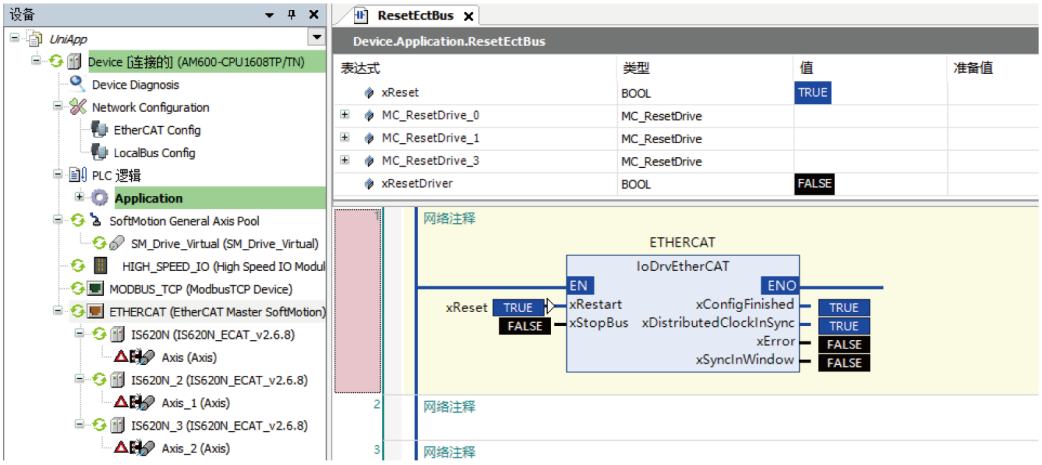
The following table describes the definition of the IoDrvEtherCAT implicit instance.

<table>
<tr><th colspan="2">IoDrvEtherCAT Implicit Instance</th></tr>
<tr><td colspan="2">
</td></tr>
<tr><th>Input parameter</th><th>Definition</th></tr>
<tr><td>xRestart</td><td>Bus restart: In rising edge, the master restarts, and all configuration parameters are reloaded.</td></tr>
<tr><td>xStopBus</td><td>Bus stop: Triggered by the level. When the input value is "TRUE", the EtherCAT bus communication stops, and a communication error occurs. To resume the communication, run "xRestart" to restart the EtherCAT communication.</td></tr>
<tr><th>Output parameter</th><th>Definition</th></tr>
<tr><td>xConfigFinished</td><td>If the parameter value is "TRUE", the transmission of all configuration parameters is complete. The communication is on-going.</td></tr>
</table>

xDistributedClockInSync	If distributed clock (DC) is configured for the EtherCAT slave, the EtherCAT slave parameters are set first when the bus starts. When parameter settings are complete (the value of "xConfigFinished" is changed to "TRUE"), the clocks of slave and master are adjusted. When the synchronization between master and slave clocks is successful, the value "TRUE" is output. If loss of synchronization occurs due to a bus fault during running, the value "FALSE" is output. In the DC mode, the motion control function block can be started only when this parameter is changed to "TRUE"; otherwise, the position of motion axis may jump.
xError	If an error is detected during the start of EtherCAT master or communication is interrupted when the slave communication state machine enters "Operational", "TRUE" is output because the EtherCAT master cannot receive any message (for example, the connection is disconnected). In this case, the error reason can be located in the diagnosis information or log of the master.

Example

The following is a program example, in which the function block name is IoDrvEtherCAT, and the master name is ETHERCAT (default for AM600) (the instance name). You do not need to claim the instance in the program. The default master name for the AC800 series is ETHERCAT_C or ETHERCAT_D.



As shown in the preceding figure, when the xRestart input variable changes from "FALSE" to "TRUE", the master restarts at the rising edge. When the value of xStopBus is "True", the bus communication stops upon the level. You can judge the bus communication information based on the output parameter status.

Master Attributes

Attribute	Definition
AutoSetOperational	If this attribute is set to "TRUE", the master always tries to restart the slave upon communication interruption. Default value: FALSE
ConfigRead	If this attribute returns "TRUE", the configuration reading is completed. You can edit the configuration. For example, you can add customized SDOs.
DCInSyncWindow	Time window condition for setting XDistributedClockInSync to "TRUE". The value of XDistributedClockInSync is "TRUE" only when the master synchronization jitter is within this window. Default value: 50 μs
DCIntegralDivider	Integral divider of DC used for circuit control. Default value: 20

Attribute	Definition
DCPropFactor	Proportion factor of DC used for circuit control. Default value: 25
DCSyncToMaster	Synchronization between DC and master. If it is set to "TRUE", all slaves are synchronized with the master, rather than the first slave synchronizing with the PLC. Default value: FALSE
DCSyncToMasterWith-SysTime	Synchronization with the master DC. If it is set to "TRUE", all slaves are synchronized with the master system time. The time read by SysTimeRtcHighResGet can also be used to synchronize the PLC with all EtherCAT slaves. Default value: FALSE
EnableTaskOutputMes-sage	Generally, EtherCAT messages are sent by a bus cycle task, while some messages may be sent by each slave output task. All outputs are written into the bus cycle task, and all inputs will be read. In other tasks, outputs are sent one more time so that they can be written into the corresponding slaves immediately. Therefore, the deadline should be shortened to ensure a fast writing. When DC is available, some slaves may encounter problems. For example, synchronization between the servo controller and synchronization interrupt is lost, but the written time is used for internal synchronizer. In this situation, multiple write access operations may exist in a cycle. If EnableTaskOutputMessage is set to "FALSE", only the bus cycle task is used, but addition tasks will not affect messages. Default value: TRUE
FirstSlave	Pointer of the first slave under the master.
FrameAtTaskStart	If FrameAtTaskStart is set to "TRUE", the frame content to the slave will be sent when the task starts, to ensure the minimum jitter. This command ensures the smooth motion of the servo drive. If this flag is set to "TRUE", the output buffer frame is written into the next cycle. Default value: FALSE
LastInstance	Pointer associated with the master list pointing to the previous master.
LastMessage	This attribute with the EtherCAT latest message together returns a text string. If the startup is successful, "all slaves completed" is returned. The function of the text string is the same as the diagnosis information displayed in the EtherCAT master editor in the online mode.
NextInstance	Pointer associated with the master list pointing to the next master.
NumberActiveSlaves	This attribute returns the number of connected slaves. If StartConfigWithLessDevice is set to "TRUE", the number of devices can be determined.
OpenTimeout	Timeout of opening the NMS. Default value: 4s
StartConfigWithLessDe-vice	This attribute can affect stack start action. If five servo controllers are configured but only three are connected, the stack stops immediately, the bus configuration fails, and the PLC reports an error. However, if StartConfigWithLessDevice in the first cycle is set to "TRUE", the stack tries to start, the bus configuration runs normally, and the PLC does not report any error. In the following scenario, if one mismatch is detected, the stack stops: Ten servo controllers are configured, the number of connected controllers is changeable, and the vendor ID and product ID of each slave will be checked.

If supported by the device, the interface provided by IIoDrvBusControl.library can be used to access the EtherCAT device from external applications.

Slave Implicit Instance

An "ETCSlave" implicit instance is created as long as the EtherCAT slave is inserted into the device list. The instance name is the same as the device name in the device list.

The input and output parameters of instance objects are used for special purposes. For example, during application running, the slave status is obtained, switched, and checked by using the slave instances. The following table describes the definition of the ETCSlave implicit instance.

The following table describes the definition of the ETCSlave implicit instance.

ETCSlave Implicit Instance	
	
Input parameter	Definition
xSetOperational	In rising edge, the slave communication state machine is attempted to be set as "ETC_SLAVE_OPERATIONAL".
Output parameter	Definition
wState	Return the current status of the slave. The possible values include: 0: ETC_SLAVE_BOOT 1: ETC_SLAVE_INIT 2: ETC_SLAVE_PREOPERATIONAL 4: ETC_SLAVE_SAVEOPERATIONAL 8: ETC_SLAVE_OPERATIONAL

Note

The ETC_SLAVE_OPERATIONAL state indicates that configuration is completed. If a configuration error occurs, the device may return to the previous state. The following is an example of IS620N slave.

ETCSlave Example

Taking 620N as an example, add the instance name 620N. Definition: nSlaveState: ETC_SLAVE_STATE;

Programming	Description
IS620N(xSetOperational:= , wState=> nSlaveState);	By calling the slave IS620N implicit instance, the slave state is output to the nSlaveState variable.

Slave Attributes

Attribute	Definition
VendorID	After the EtherCAT master starts, this attribute returns the vendor ID read from the device.
ConfigVendorID	This attribute reads the vendor ID from configuration.
ProductID	After the EtherCAT master starts, this attribute returns the product ID read from the device.
ConfigProductID	This attribute reads the product ID from configuration.
SerialID	After the EtherCAT stack starts, this attribute carries the device SN.
LastEmergency	If a message is received, the message is stored in the slave. This attribute can be used to read information from application. In addition, a log message is added.

If supported by the device, the interface provided by IIoDrvBusControl.library can be used to access the EtherCAT device from external applications. After the vendor and product IDs are activated in advanced settings, if the vendor ID does not match the configured vendor ID or the product ID does not match the configured product ID, the master is stopped.

Checking All Slave Link Tables

A function block instance is created between each pair of EtherCAT master and EtherCAT slave in implicit way. The instance monitors the state of each slave. Therefore, this instance must be called in the application program. The slave state is read by using wState. To simplify the programming, all masters and slaves can be found in link tables. Therefore, all slaves can be checked cyclically by a simple "WHILE" clause.

The masters and slaves correspond to attributes NextInstance and LastInstance, respectively, returning the pointers to next and previous stations. In addition, the FirstSlave attribute of the master is effective. It provides the pointer to the first slave.

Link Table Function Example

Check the state of all slaves. Definition: pSlave: POINTER TO ETCSlave;

Programming	Description

pSlave := EtherCAT_Master.FirstSlave;
WHILE pSlave <> 0 DO
pSlave^();
//TODO: Add the code, such as counting the Op state of the slave.
pSlave := pSlave^.NextInstance;
END_WHILE
	The first slave of the master is found by using EtherCAT_Master. FirstSlave.
	Instances are called in the "WHILE" loop, to determine wState. Then the state is checked.
	The pointer to the next slave is found by using pSlave^. NextInstance.
	The pointer at the end of table is null, and the loop is finished.

CoE IODrvEtherCAT Function Library

CoE function block: CANopen over EtherCAT

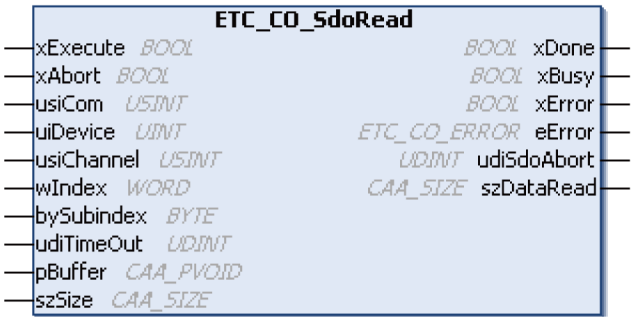
After EtherCAT configuration is enabled for IODrvEtherCAT.library of EtherCAT, the library is automatically added to project, including the read/write function block. Therefore, the special parameters can be checked and modified in the online mode. When the CANopen over EtherCAT function block is used, multiple function modules can be called. Internal requests are processed in queue.

The CANopen over EtherCAT function block includes the following function blocks:

- ETC_CO_SdoRead (retrieve parameter, of which the length may exceed four bytes)
- ETC_CO_SdoRead4 (read parameter, of which the length does not exceed four bytes)
- ETC_CO_SdoReadDword (read parameter, of which the value is stored in DWORD)
- ETC_CO_SdoRead_Access (read all records)
- ETC_CO_SdoRead_Channel (read slave parameters)
- ETC_CO_SdoWrite (write parameter, of which the length may exceed four bytes)
- ETC_CO_SdoWrite4 (write parameter, of which the length does not exceed four bytes)
- ETC_CO_SdoWriteDWord (value is written into DWORD)
- ETC_CO_SdoWriteAccess (write slave parameters)

ETC_CO_SdoRead

This function module is provided by IODrvEtherCAT.library to read EtherCAT slave parameters. Different from ETC_CO_SdoRead4, this module supports parameters longer than four bytes. The read parameters are specified by object dictionary indexes and sub-indexes.

ETC_CO_SdoRead Function Block


Input parameter	Definition
xExecute	In the input rising edge, read slave parameters. To obtain the storage unit of internal channel, this instance must be called at least once by "xExecute:= FALSE".
xAbort	If it is set to "TRUE", the read process is aborted.
usiCom	Number of EtherCAT masters: If only one EtherCAT master is used, the usiCom value is 1. If multiple masters exist, the value of the first master is 1, the value of the second one is 2, and so on.
uiDevice	Physical address of the slave. If the automatic configuration mode of the master is disabled, you can configure a special address for the slave. The address selected arbitrarily must be entered here. If the automatic configuration mode is enabled, the first slave obtains address "1001". The current slave address can be checked in the "EtherCAT address" field of the slave configuration dialog box of the device editor.
usiChannel	Reserved for expansion.
wIndex	Parameter index in the object dictionary.
bySubIndex	Parameter sub-index in the object dictionary.
udiTimeout	You can set the timeout interval in milliseconds. If the read parameter is not executed within the timeout interval, an error is reported.
pBuffer	Pointer of the data buffer. The data buffer refers to the storage area where the successfully transmitted parameters are stored.
szSize	Size of the data buffer (see pBuffer), in bytes.
Output parameter	Definition
xDone	The value is "TRUE" when parameter reading is completed.
xBusy	The value is "TRUE" when parameter reading has not been completed.
xError	The value is "TRUE" when an error occurs. The eError parameter displays the error reason.
eError	The output (ETC_CO_ERROR) displays the error reason specified by xError. For example, "ETC_CO_TIMEOUT" indicates a timeout error.
udiSdoAbort	When an error occurs in device checking, this output provides more error information.
szDataRead	Number of read bytes, namely, the maximum szSize (see input parameters).

ENUM ETC_CO_ERROR

Error	Code	Description
ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	The error reason is stored in the output of udiSdoAbort.
ETC_CO_OTHER_ERROR	5751	No master is found.
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited, of which the length is larger than 4.
ETC_CO_TIME_OUT	5753	Timeout occurs.

Error	Code	Description
ETC_CO_FIRST_MF	5770	Not in use.
ETC_CO_LAST_ERROR	5799	Not in use.

ETC_CO_SdoRead4

This function module is provided by IODrvEtherCAT.library to read EtherCAT slave parameters. Different from ETC_CO_SdoRead, this function module reads only parameters not longer than four bytes. The read parameters are specified by object dictionary indexes and sub-indexes.

ETC_CO_SdoRead4 Function Block


Input parameter	Definition
xExecute	In the input rising edge, read slave parameters. To obtain the storage unit of internal channel, this instance must be called at least once by "xExecute:= FALSE".
xAbort	If it is set to "TRUE", the read process is aborted.
usiCom	Number of EtherCAT masters: If only one EtherCAT master is used, the usiCom value is 1. If multiple masters exist, the value of the first master is 1, the value of the second one is 2, and so on.
uiDevice	Physical address of the slave. If the automatic configuration mode of the master is disabled, you can configure a special address for the slave. The address selected arbitrarily must be entered here. If the automatic configuration mode is enabled, the first slave obtains address "1001". The current slave address can be checked in the "EtherCAT address" field of the slave configuration dialog box of the device editor.
usiChannel	Reserved for expansion.
wIndex	Parameter index in the object dictionary.
bySubIndex	Parameter sub-index in the object dictionary.
udiTimeout	You can set the timeout interval in milliseconds. If the read parameter is not executed within the timeout interval, an error is reported.
Output parameter	**Definition**
wState	Return the current status of the slave. The possible values include: 0: ETC_SLAVE_BOOT 1: ETC_SLAVE_INIT 2: ETC_SLAVE_PREOPERATIONAL 4: ETC_SLAVE_SAVEOPERATIONAL 8: ETC_SLAVE_OPERATIONAL
xDone	The value is "TRUE" when parameter reading is completed.
xBusy	The value is "TRUE" when parameter reading has not been completed.
xError	The value is "TRUE" when an error occurs. The eError parameter displays the error reason.
eError	The output (ETC_CO_ERROR) displays the error reason specified by xError. For example, "ETC_ CO_TIMEOUT" indicates a timeout error.

ETC_CO_SdoRead4 Function Block	
abyData	The read parameter data is copied to this 4-byte array. If the first byte has been read, it is stored in the first index of the array. The 2 or 4-byte data is copied to this array in Intel byte order.
usiDataLength	Number of read bytes (1, 2, or 4).

ENUM ETC_CO_ERROR

Error	Code	Description
ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	The error reason is stored in the output of udiSdoAbort.
ETC_CO_OTHER_ERROR	5751	No master is found.
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited, of which the length is larger than 4.
ETC_CO_TIME_OUT	5753	Timeout occurs.
ETC_CO_FIRST_MF	5770	Not in use.
ETC_CO_LAST_ERROR	5799	Not in use.

ETC_CO_SdoReadDword

This function block is provided by IODrvEtherCAT.library. Similar to ETC_CO_SdoRead4, it reads EtherCAT slave parameters. However, the read data is copied to DWORD (dwData), rather than an array. Byte exchange is automatically performed. Therefore, the read data can be directly used by subsequent processes.

ETC_CO_SdoRead_Access

This function block is provided by IODrvEtherCAT.library. Similar to ETC_CO_SdoRead, it reads EtherCAT slave parameters. All record indexes can be read by inputting xCompleteAccess (BOOL). Therefore, xCompleteAccess must be set to "TRUE", and bySubIndex must be 0.

ETC_CO_SdoRead_Channel

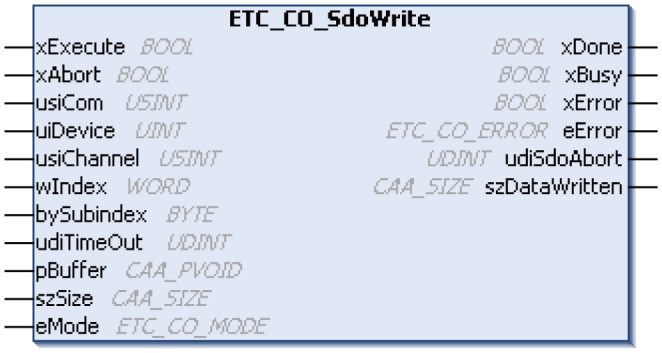
The EtherCAT programming interface in EtherCAT configuration editor is used by the ETC_CO_ SdoRead_Channel function block of IODrvEtherCAT in CAN over EtherCAT.

This function block is provided by IODrvEtherCAT.library. Similar to ETC_CO_SdoRead_Access, it reads parameters of all EtherCAT slaves.

However, it has an additional input byChannelPriority (BYTE) that specifies the channel and priority in the CoE email box message. The first 6 bits specify the channel, and the last 2 bits specify the priority.

ETC_CO_SdoWrite

This function module is provided by IODrvEtherCAT.library to write EtherCAT slave parameters. Different from ETC_CO_SdoWrite4, this function module can read parameters larger than 4 bytes. The written parameters are specified by object dictionary indexes and sub-indexes.

ETC_CO_SdoWrite Function Block	
	
Input parameter	Definition
xExecute	In the input rising edge, write slave parameters. To obtain the storage unit of internal channel, this instance must be called at least once by "xExecute:= FALSE".
xAbort	If it is set to "TRUE", the write process is aborted.
usiCom	Number of EtherCAT masters: If only one EtherCAT master is used, the usiCom value is 1. If multiple masters exist, the value of the first master is 1, the value of the second one is 2, and so on.
uiDevice	Physical address of the slave. If the automatic configuration mode of the master is disabled, you can configure a special address for the slave. The address selected arbitrarily must be entered here. If the automatic configuration mode is enabled, the first slave obtains address "1001". The current slave address can be checked in the "EtherCAT address" field of the slave configuration dialog box.
usiChannel	Reserved for expansion.
wIndex	Parameter index in the object dictionary.
bySubIndex	Parameter sub-index in the object dictionary.
udiTimeout	You can set the timeout interval in milliseconds. If the write parameter is not executed within the timeout interval, an error is reported.
pBuffer	Pointer of the data buffer. The data buffer refers to the storage area where the successfully transmitted parameters are stored.
szSize	Size of the data buffer (see pBuffer), in bytes.
eMode	This input (enumeration: ETC_CO_MODE) defines the number of written bytes. Possible values include ETC_CO_AUTO (automatic), ETC_CO_EXPEDITED (expedited), and ETC_ CO_ SEGMENTED (segmented). Generally, the ETC_CO_AUTO mode is used because the data length is automatically matched in this mode.
Output parameter	Definition
xDone	The value is "TRUE" when parameter writing is completed.
xBusy	The value is "TRUE" when parameter writing has not been completed.
xError	The value is "TRUE" when an error occurs. The eError parameter displays the error reason.
eError	The output (ETC_CO_ERROR) displays the error reason specified by xError. For example, "ETC_ CO_TIMEOUT" indicates a timeout error.
udiSdoAbort	When an error occurs in device checking, this output provides more error information.
szDataWritten	Number of written bytes. After bytes are successfully written, it is set to szSize.

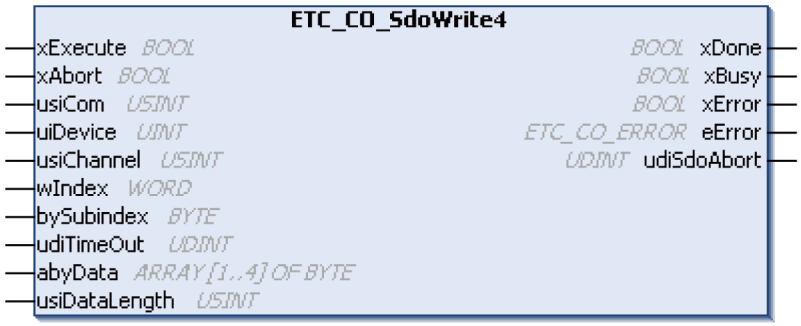
ENUM ETC_CO_MODE

Mode	No.	Description
AUTO	0	The automatic mode is selected.
EXPEDITED	1	The expediting protocol is used.
SEGMENTED	2	The segmented transmission protocol is used.

ETC_CO_SdoWrite4

This function module is provided by IODrvEtherCAT.library to write EtherCAT slave parameters. Different from ETC_CO_SdoWrite, this function module writes only parameters smaller than four bytes. The written parameters are specified by object dictionary indexes and sub-indexes.

ETC_CO_SdoWrite4 Function Block



Input parameter	Definition
xExecute	In the input rising edge, write slave parameters. To obtain the storage unit of internal channel, this instance must be called at least once by "xExecute:= FALSE".
xAbort	If it is set to "TRUE", the write process is aborted.
usiCom	Number of EtherCAT masters: If only one EtherCAT master is used, the usiCom value is 1. If multiple masters exist, the value of the first master is 1, the value of the second one is 2, and so on.
uiDevice	Physical address of the slave. If the automatic configuration mode of the master is disabled, you can configure a special address for the slave. The address selected arbitrarily must be entered here. If the automatic configuration mode is enabled, the first slave obtains address "1001". The current slave address can be checked in the "EtherCAT address" field of the slave configuration dialog box.
usiChannel	Reserved for expansion.
wIndex	Parameter index in the object dictionary.
bySubIndex	Parameter sub-index in the object dictionary.
udiTimeout	You can set the timeout interval in milliseconds. If the write parameter is not executed within the timeout interval, an error is reported.
abyData	This array includes four written data records. The data must be stored in the Intel byte order.
usiDataLength	Number of written bytes (1, 2, or 4).
Output parameter	Definition
xDone	This output is set to "TRUE" when parameter writing is completed.
xBusy	The output is "TRUE" if the write operation is not completed.
xError	If an error occurs, this output is set to "TRUE". eError displays the error reason.

ETC_CO_SdoWrite4 Function Block	
eError	This output (ETC_CO_ERROR) displays the error reason, identified as xError. For example, ETC_CO_TIMEOUT indicates timeout.
udiSdoAbort	If the device has an error, this output provides more error information.

ENUM ETC_CO_MODE

Mode	No.	Description
AUTO	0	The automatic mode is selected.
EXPEDITED	1	The expediting protocol is used.
SEGMENTED	2	The segmented transmission protocol is used.

ETC_CO_SdoWriteDWord

This function block is provided by IODrvEtherCAT.library. Similar to ETC_CO_SdoWrite4, it writes EtherCAT slave parameters. However, the data to be written is output in DWORD (dwData) format, rather than an array. Byte exchange is automatically carried out. The value to be written can be specified.

ETC_CO_SdoWriteAccess

This function block is provided by IODrvEtherCAT.library. Similar to ETC_CO_SdoWrite, it writes EtherCAT slave parameters.

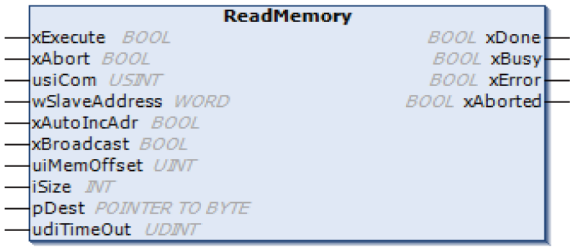
All record indexes can be written by inputting xCompleteAccess (BOOL). Therefore, xCompleteAccess must be set to "TRUE", and bySubIndex must be 0. However, it has an additional input byChannelPriority (BYTE) that specifies the channel and priority in the CoE email box message.

Direct Access to EtherCAT Slave Memory

Choose "EtherCAT configuration editor" > "EtherCAT programming interface" to directly access the EtherCAT slave memory.

Directly access the EtherCAT slave memory through ReadMemory and WriteMemory.

- ReadMemory
  This function block is in IODrvEtherCAT.library to read data in the EtherCAT slave memory.

ReadMemory Function Block		
		
Input parameter	Type	Definition

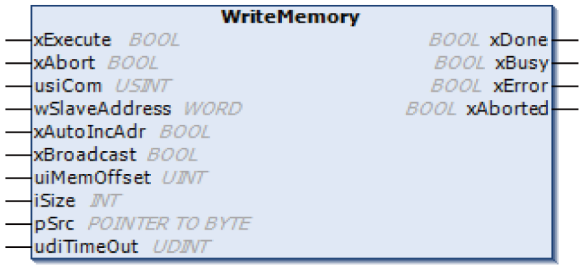
ReadMemory Function Block		
xExecute	BOOL	Rising edge: Action starts. Falling edge: Reset output. If there is a falling edge before the function block completes the action, the output operation is performed in a normal way and a reset is performed when the action is completed or aborted (xAbort). In this case, the related output values (xDone, xError, and iError) are output within a cycle.
xAbort	BOOL	If the input is "TRUE", the action is aborted immediately and all outputs are reset to the initial values.
usiCom	USINT	Master index 1: The first EtherCAT master.
wSlaveAddress	WORD	Automatically created address or device physical address.
xAutoIncAdr	BOOL	Flag confirming that the address mode is used.
xBroadcast	BOOL	If the board mode is used and the value is "TRUE", wSlaveAddress and bAutoIncAdr will not be used.
uiMemOffset	UINT	Memory address offset.
iSize	INT	Read byte.
pDest	POINTER TO BYTE	Data storage and retrieve buffer.
udiTimeOut	UDINT	Operation timeout interval (ms)
Output parameter	Type	Definition
xDone	BOOL	Action completed successfully.
xBusy	BOOL	Function block activated.
xError	BOOL	TRUE: An error is generated and the function block aborts action. FALSE: No error.
xAborted	BOOL	Abort action.

Example: Read register 0x130 (current state)

```
PROGRAM PLC_PRG
VAR
   etcreadmemory : ReadMemory;
   wStatus : WORD;
   xRead : BOOL;
END_VAR

etcreadmemory(xExecute := xRead, usiCom:=1, wSlaveAddress := 1002,
        xAutoIncAdr := FALSE, xBroadcast := FALSE, uiMemOffset := 16#130,
        iSize := 2, pDest := ADR(wStatus), udiTimeout := 500);
```

- WriteMemory
  This function block is in IODrvEtherCAT.library to write data into the EtherCAT slave memory.

WriteMemory Function Block		
		
Input parameter	Type	Definition
xExecute	BOOL	Rising edge: Action starts. Falling edge: Reset output. If there is a falling edge before the function block completes the action, the output operation is performed in a normal way and a reset is performed when the action is completed or aborted (xAbort). In this case, the related output values (xDone, xError, and iError) are output within a cycle.
xAbort	BOOL	If the input is "TRUE", the action is aborted immediately and all outputs are reset to the initial values.
usiCom	USINT	Master index 1: The first EtherCAT master.
wSlaveAddress	WORD	Automatically created address or device physical address.
xAutoIncAdr	BOOL	Flag confirming that the address mode is used.
xBroadcast	BOOL	If the board mode is used and the value is "TRUE", wSlaveAddress and bAutoIncAdr will not be used.
uiMemOffset	UINT	Memory address offset.
iSize	INT	Write byte.
pDest	POINTER TO BYTE	Read data in and retrieve data from the data buffer.
udiTimeOut	UDINT	Operation timeout interval (ms)
Output parameter	Type	Definition
xDone	BOOL	Action completed successfully.
xBusy	BOOL	Function block activated.
xError	BOOL	TRUE: An error is generated and the function block aborts action. FALSE: No error.
xAborted	BOOL	Abort action.

Example: Write register 0x120 (control register)

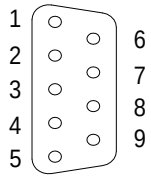
PLC_PRG

```
VAR
   etcwritememory : WriteMemory;
   xWrite : BOOL;
   wCommand : WORD := 4; // set to safe operational
END_VAR
etcwritememory(xExecute := xWrite, usiCom:=1, wSlaveAddress := 1002,
        xAutoIncAdr := FALSE, xBroadcast := FALSE, uiMemOffset := 16#120,
        iSize := 2, pSrc := ADR(wCommand), udiTimeout := 500);
```

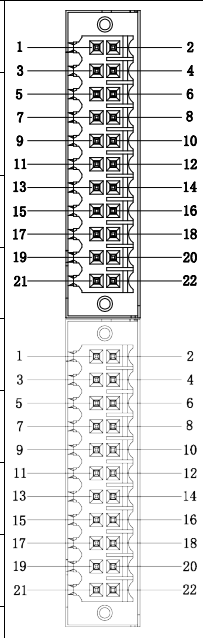
4.5 Modbus Device Editor

4.5.1 Serial Hardware Port

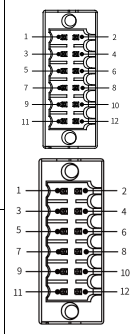
AM600 supports communication using two RS485 serial ports: serial port 0 and serial port 1, both supporting the free protocol.

Terminal	Channel	Pin	Definition
	COM0 (RS485)	1	RS485–
		2	RS485+
		5	GND
	COM1 (RS485)	6	RS485–
		9	RS485+
		3	GND

The AC700-series PLCs support one RS485 communication port and one RS232 communication port connected to the I/O communication port. The RS485 communication port has the pins 1, 3, and 5, and the RS232 communication port has the pins 2, 4, and 6. The following table describes these pins.

Type	Function	Terminal	No.	I/O communication port	No.	Type	Function	Terminal
RS485	RS485+	RS485+	1		2	232R	RS232 receiving	RS232
	RS485–	RS485—	3		4	232T	RS232 transmit-ting	
	Serial port ground	GND	5		6	GND	Serial port ground	
DI	Power-on signal	⏻	7		8	⏹	RUN/STOP	DI
DI	High-speed input 0	X0	9		10	X1	High-speed input 1	DI
DI	High-speed input 2	X2	11		12	X3	High-speed input 3	DI
DI	High-speed input 4	X4	13		14	X5	High-speed input 5	DI
DI	High-speed input 6	X6	15		16	X7	High-speed input 7	DI
DI	Input common terminal	S/S	17		18	COM	Output common terminal	DO
DO	High-speed output 0	Y0	19		20	Y1	High-speed output 1	DO
DO	High-speed output 2	Y2	21		22	Y3	High-speed output 3	DO

The AC800-series PLCs support one RS485 communication port and one RS232 communication port connected to the I/O communication port. The RS485 communication port has the pins 8, 9, and 11, and the RS232 communication port has the pins 8, 10, and 12. The following table describes these pins.

Description	Function	Signal	No.	I/O communication port	No.	Signal	Function	Description
When a pulse of high-level width 500 ms is input and the power key is pressed, the PLC is started.	Power-on signal (used together with the UPS)	⏻	1		2	P_ STATUS	Power-on signal	Output after the controller is powered on and started
Enable memory retention at power failure during ON-OFF switchover	Power failure detection signal	P_OK	3		4	P_ STATUS	Operation status signal	Output after the controller is powered on and started
OFF: RUN ON: STOP	RUN/STOP	RUN	5		6	0V	Output common terminal	-
-	Input common terminal	0V	7		8	GND	Communication reference ground	-
-	RS485+	RS485+	9		10	232R	RS232 receiving	-
-	RS485–	RS485—	11		12	232T	RS232 transmitting	-

The AM300-/AM500-series PLCs support up to three RS485 communication ports (one for the main unit and two for the expansion card). The following table defines the relevant pins.



Signal	Left Terminal	Right Terminal	Signal
RS485 differential pair positive signal	RS485+	+24V	24 VDC power supply positive
RS485 differential pair negative signal	RS485—	0V	24 VDC power supply negative
RS485 communication ground	GND	⏚	PE

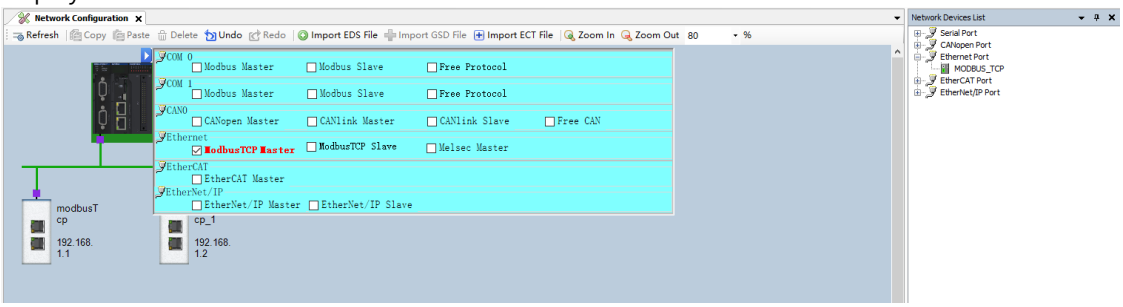
Note

On the software page of the AC700/AC800 series, the COM port 0 corresponds to the RS232 communication port, and the COM port 1 corresponds to the RS485 communication port. Differentiate them during configuration.

4.5.2 Network Configuration

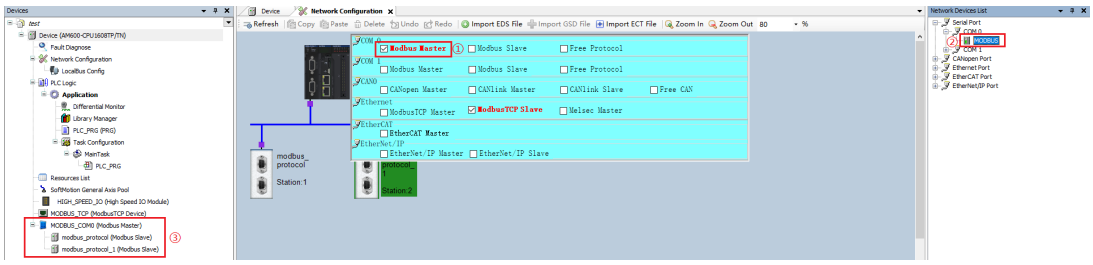
AM400/AM600/AC700/A800 Network Configuration

1. In the left device tree, double-click "Network Configuration". The "Network Configuration" page is displayed.
2. Click the PLC device picture. The enabling states of all masters/slaves supported by the PLC are displayed.

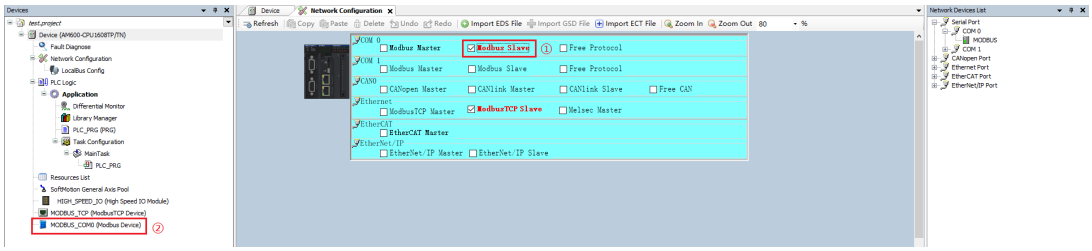


3. Enable the PLC as the Modbus master or Modbus slave.

- Enable the PLC as the Modbus master.
  Select "Modbus Master". In the "Network Device List" on the right, double-click "MODBUS" to enable the PLC as the Modbus master and add the Modbus slave to the network.



- Enable the PLC as the Modbus slave.
  Select "Modbus Slave" to enable the PLC as the Modbus slave.

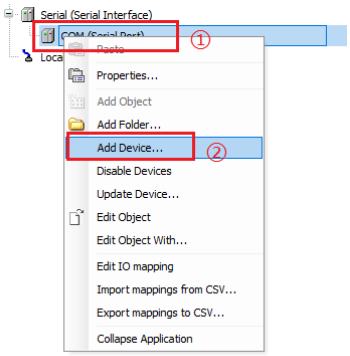


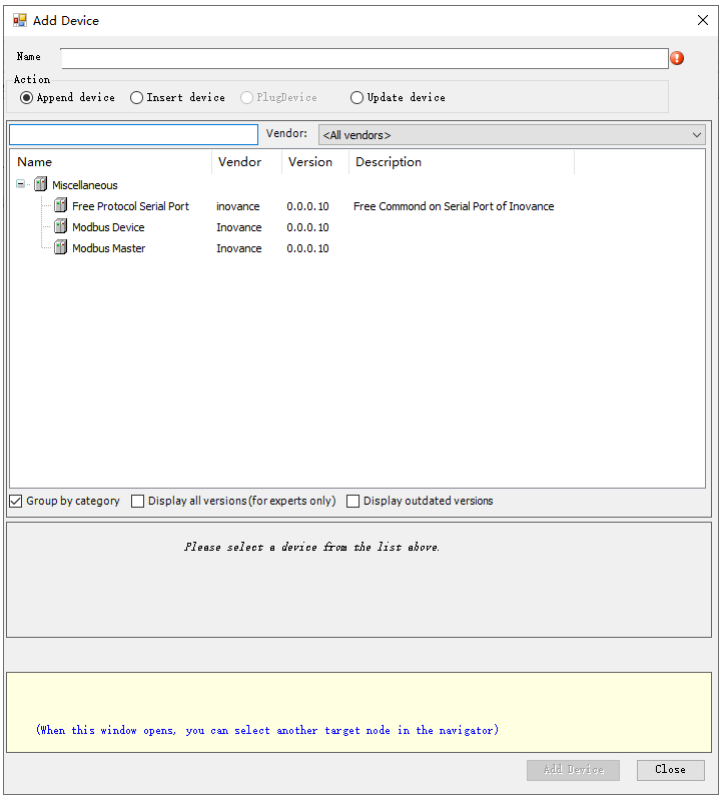
When the PLC serves as the Modbus slave, the addresses that can be accessed by the master are as follows:

a. All bit variable operations (function codes 0x01, 0x02, 0x05, and 0x0F) can read/write 65535 bit variables ranging from %QX0.0 to %QX8191.7.
b. All register variable operations (function codes 0x03, 0x04, 0x06, and 0x10) can read/write 65536 register variables ranging from MW0 to MW65535.
c. Inovance HMI can access system variables SM0 to SM7999 and register variables SD0 to SD7999.

AM300/AM500 Network Configuration

1. In the left device tree, right-click "COM (Serial Port)". In the shortcut menu, select "Add Device". The "Add Device" dialog box is opened.

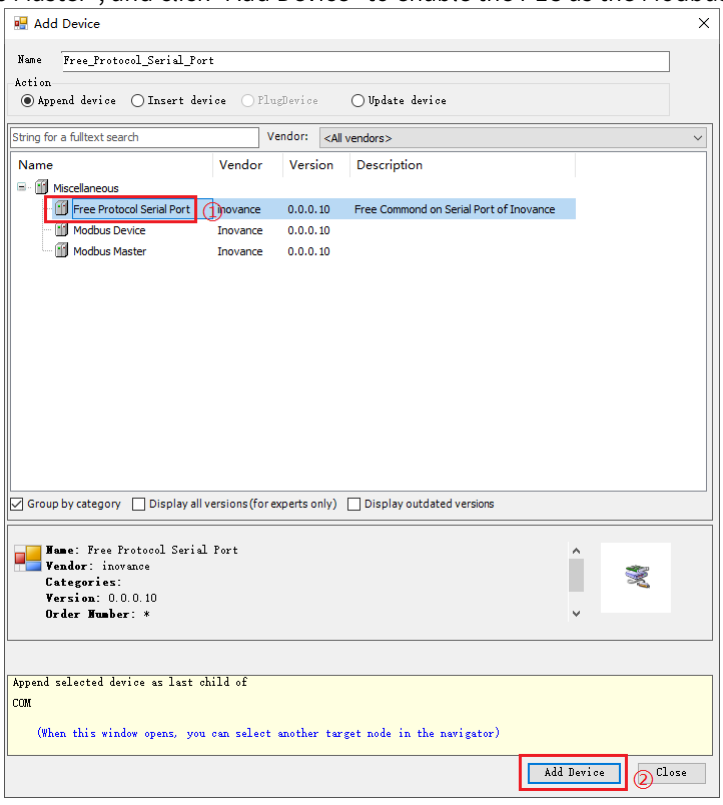




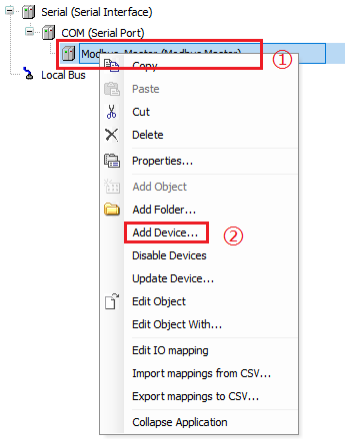
2. Enable the PLC as the Modbus master or Modbus slave.

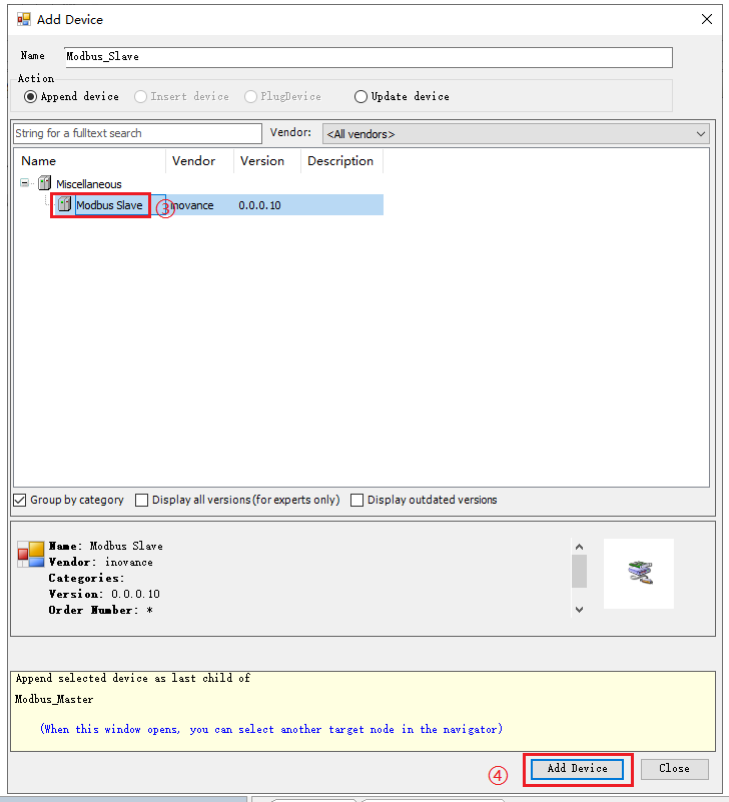
- Enable the PLC as the Modbus master.

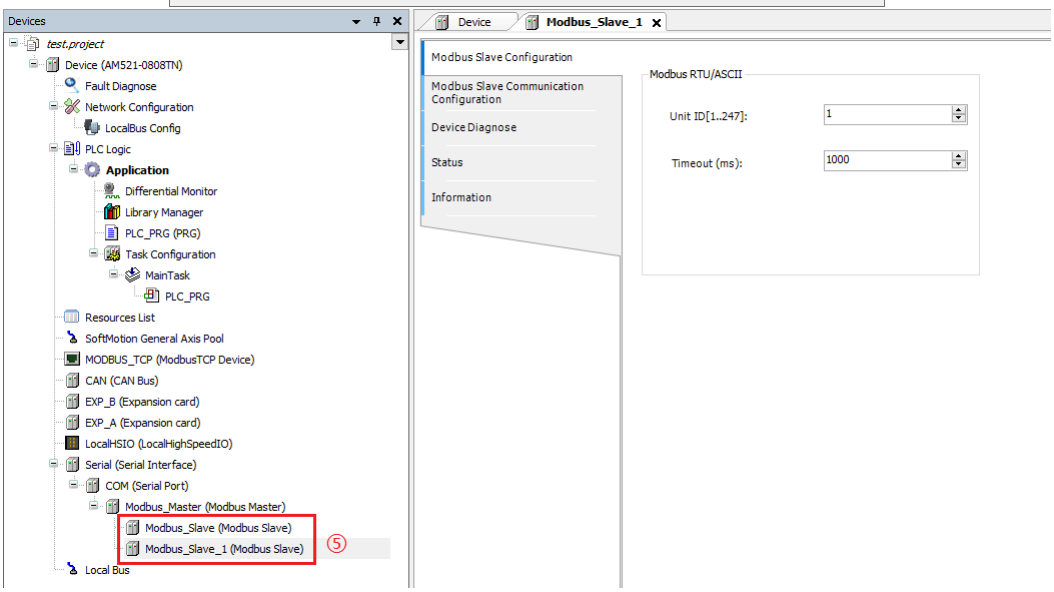
  a. Select "Modbus Master", and click "Add Device" to enable the PLC as the Modbus master.



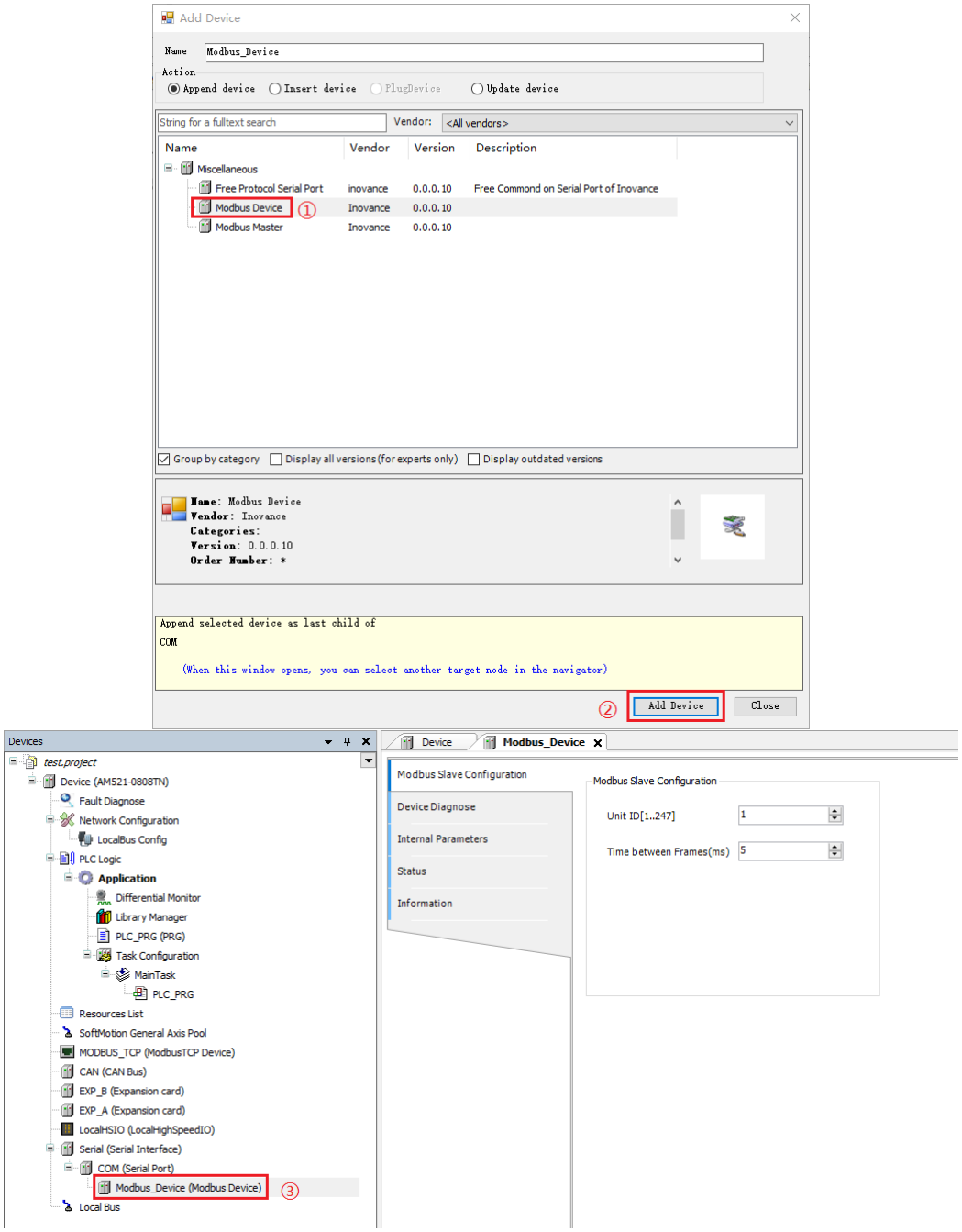
  b. Right-click "Modbus Master" and select "Add Device". In the dialog box displayed, click "Modbus Slave" and then click "Add Device" to add the Modbus slave to the network.


Serial (Serial Interface)
COM (Serial Port)
Local Bus
①
Copy
Paste
Cut
Delete
Properties...
Add Object
Add Folder...
Add Device...
②
Disable Devices
Update Device...
Edit Object
Edit Object With...
Edit IO mapping
Import mappings from CSV...
Export mappings to CSV...
Collapse Application


Add Device
Name Modbus_Slave
Action
Append device
Insert device
PlugDevice
Update device
String for a fulltext search
Vendor: <All vendors>
Name
Vendor
Version
Description
Miscellaneous
Modbus Slave
③
inovance
0.0.0.10
Group by category
Display all versions (for experts only)
Display outdated versions
Name: Modbus Slave
Vendor: inovance
Categories:
Version: 0.0.0.10
Order Number: *
Append selected device as last child of
Modbus_Master
(When this window opens, you can select another target node in the navigator)
④
Add Device
Close


Devices
test.project
Device (AM521-0808TN)
Fault Diagnose
Network Configuration
LocalBus Config
PLC Logic
Application
Differential Monitor
Library Manager
PLC_PRG (PRG)
Task Configuration
MainTask
PLC_PRG
Resources List
SoftMotion General Axis Pool
MODBUS_TCP (ModbusTCP Device)
CAN (CAN Bus)
EXP_B (Expansion card)
EXP_A (Expansion card)
LocalHSIO (LocalHighSpeedIO)
Serial (Serial Interface)
COM (Serial Port)
Modbus_Master (Modbus Master)
Modbus_Slave (Modbus Slave)
Modbus_Slave_1 (Modbus Slave)
⑤
Local Bus
Device
Modbus_Slave_1
Modbus Slave Configuration
Modbus Slave Communication Configuration
Device Diagnose
Status
Information
Modbus RTU/ASCII
Unit ID[1..247]: 1
Timeout (ms): 1000

- Enable the PLC as the Modbus slave.
  Select "Modbus Slave", and click "Add Device" to enable the PLC as the Modbus slave.



When the PLC serves as the Modbus slave, the addresses that can be accessed by the master are as follows:

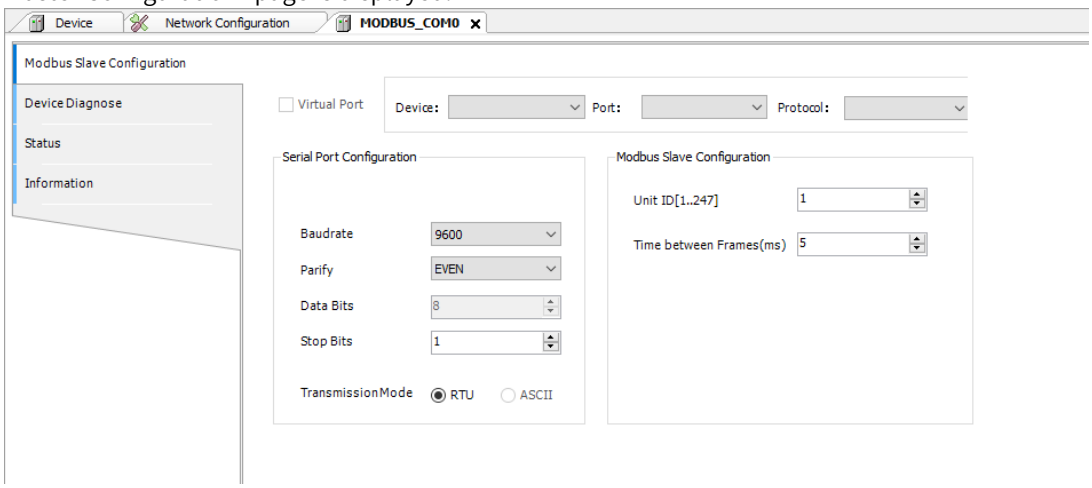
a. All bit variable operations (function codes 0x01, 0x02, 0x05, and 0x0F) can read/write 65535 bit variables ranging from %QX0.0 to %QX8191.7.
b. All register variable operations (function codes 0x03, 0x04, 0x06, and 0x10) can read/write 65536 register variables ranging from MW0 to MW65535.
c. Inovance HMI can access system variables SM0 to SM7999 and register variables SD0 to SD7999.

4.5.3 Modbus Master Configuration

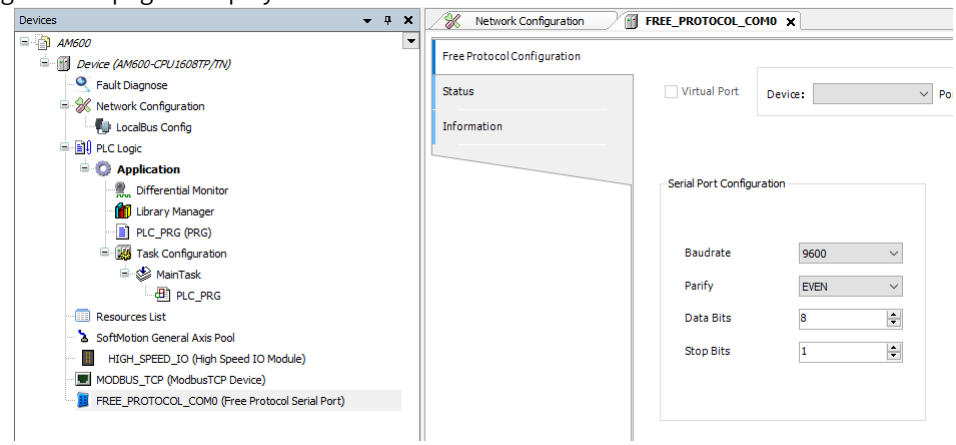
When the PLC serves as the Modbus master, the communication parameters of the Modbus master and slave must be consistent; otherwise, communication may fail.

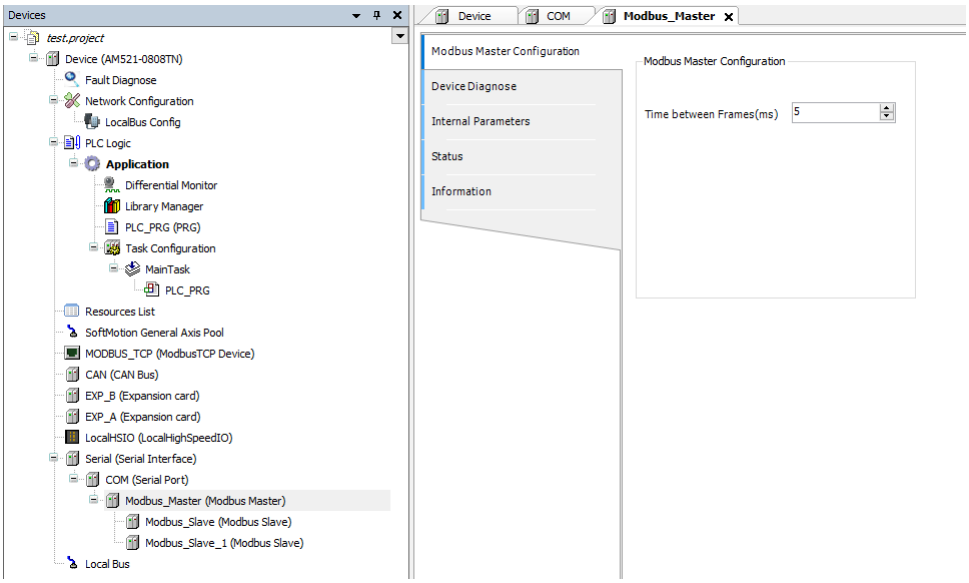
1. Open the "Modbus Master Configuration" page.

- AM400/AM600/AC700/A800: In the left device tree, double-click the Modbus master. The "Modbus Master Configuration" page is displayed.



- AM300/AM500: In the left device tree, double-click "COM (Serial Port)". The "COM Configuration" page is displayed. Double-click "Modbus_Master (Modbus Master)". The "Modbus Master Configuration" page is displayed.





2. Configure the Modbus master parameters. The following table lists the master parameters.

Parameter	Description
COM Port	Serial port 0 or 1, which is used to establish a physical connection to the master
Baud rate	Communication rate
Parity	Method of verifying communication frames
Data Bits	Actual data bits included in communication frames
Stop Bits	Last bit in a single message during communication
Transmission Mode	RTU
Time between Frames	The time for the master to wait for the next request data frame after receiving a response data frame

Example

Parameter	Value
COM Port	0
Baud rate	115200
Parity	Even parity
Data Bits	8
Stop Bits	1
Transmission Mode	RTU
Time between Frames	2 ms

4.5.4 Modbus Master Communication Configuration

When the PLC serves as the Modbus master, double-click a slave in the device tree. The "Modbus Slave Configuration" window is displayed, as shown in the following figure.

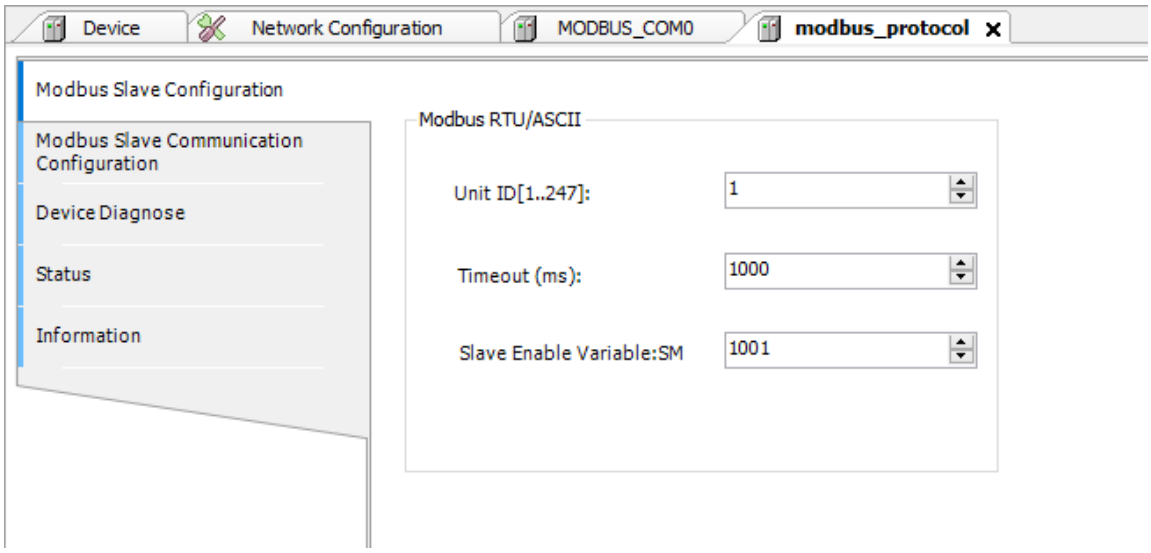


Figure 4-12 Modbus slave configuration when port functions as master

Slave configuration parameters:

Configuration Item	Function
Unit ID	ID of the slave, ranging from 1 to 247
Timeout	After sending frames, the master reports receiving timeout if no data is received from the slave within this timeout period.
Slave Enable Variable	Enables the slave by programming and starts to send frames to the slave.

Example:

Configuration Item	Value
Unit ID	11
Timeout	1000 ms
Slave Enable Variable	1001

Switch to the "Modbus Slave Configuration" window, and add the Modbus master-slave communication configuration. Up to 60 configuration entries can be added, as shown in the following figure.

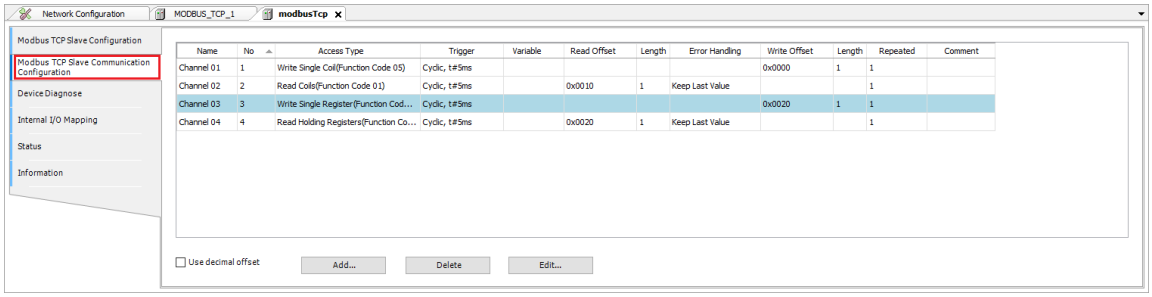


Figure 4-13 Modbus slave communication configuration when port functions as master

In the preceding figure, each channel represents an independent Modbus request. The fourth row defines the cyclic operation on a write-single register (function code 0x06) to write a word to the register with an offset of 0x0020.

After you click "Add...", a dialog box for adding a channel for the Modbus slave is displayed. Click "OK" to create a channel.

Select a channel from the Modbus slave channel list and click "Edit...". The "Modbus Channel Set" dialog box is displayed. Change the values of parameters to modify the channel settings. Click "OK" to update the channel settings. You can set the following parameters to add or edit a channel:

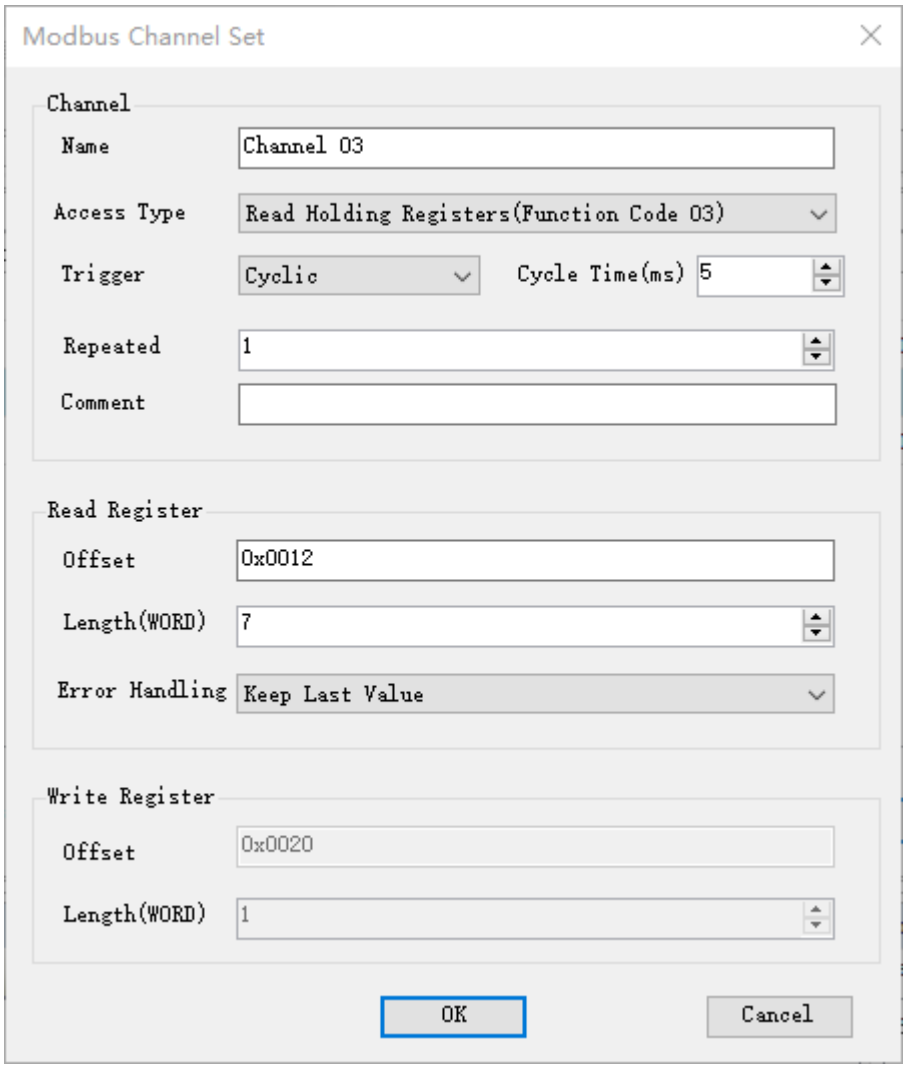


Figure 4-14 Modbus slave communication configuration when port functions as master

Modbus communication parameter settings

Configuration Item	Function
Name	Channel name, in the string format.
Access Type	Read Coils (Function Code 01). Read Discrete Inputs (Function Code 02). Read Holding Registers (Function Code 03). Read Input Registers (Function Code 04). Write Single Coil (Function Code 05). Write Single Register (Function Code 06). Write Multiple Coils (Function Code 15). Write Multiple Registers (Function Code 16).

Configuration Item	Function	
Trigger	Cyclic: Requests are triggered periodically.	Cycle Time: Time for re-execution.
	Level Trigger: Requests are triggered when a change is made during programming.	Trigger Variable (SM): SM element that implements trigger. After trigger is successful, the element is reset automatically.
Repeated	A request is resent for the specified times when no response frame is received from the slave upon a communication error.	
Comment	Brief text description about data.	
Read registers		
Offset	Head address of the registers to be read.	
Length	Number of registers to be read.	
Error Handling	Keep Last Value: The last valid value is kept. 0: All the values are zeroed.	
Write Register		
Offset	Head address of the registers to be written.	
Length	Number of registers to be written.	

The valid range of the "Length" parameter depends on the following parameters:

Function Code	Access Type	Register Count
01	Read Coils	1 to 2000
02	Read Discrete Inputs	1 to 2000
03	Read Holding Registers	1 to 125
04	Read Input Registers	1 to 125
05	Write Single Coil	1
06	Write Single Register	1
15	Write Multiple Coils	1 to 1968
16	Write Multiple Registers	1 to 123

Modbus slave internal I/O mapping

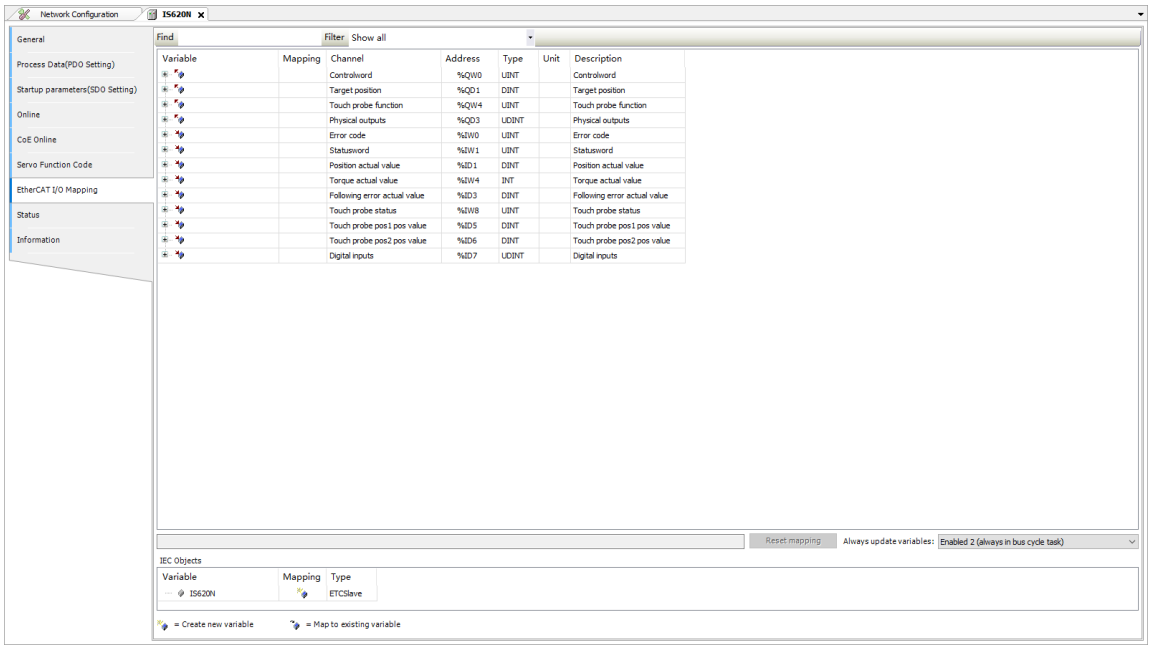


Figure 4-15 Modbus slave internal I/O mapping when port functions as master

After master-slave communication configurations are added on the "Modbus Slave Communication Configuration" page, the mapped address of each configuration is automatically allocated in the Internal I/O Mapping. In the preceding figure, %IW1 in the first row indicates that the read value of a coil to the address %IW1. Besides, you can map the customized variables in the program to the I/O address through the input assistant or directly inputting the example variable path.

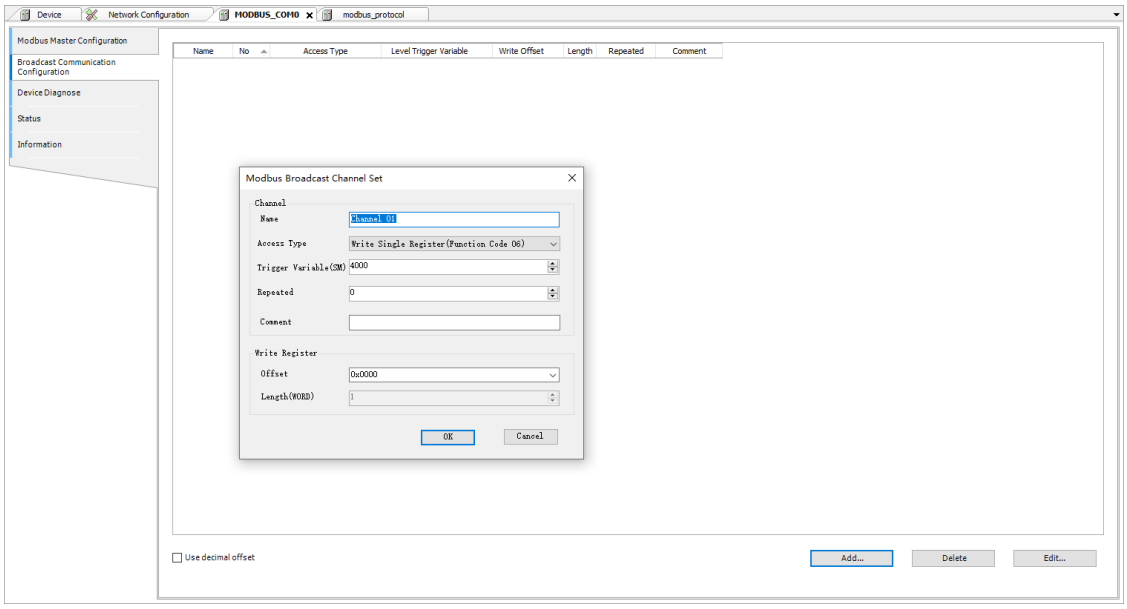
4.5.5 Modbus Master Broadcast Configuration

Caution

Broadcast configuration is only required for Modbus masters of the AM400-/AM600-series PLCs. Skip this section for PLCs of other series.

When the Modbus master is connected to multiple Modbus slaves, and all Modbus slaves receive the write operation, the Modbus master needs to perform broadcast.

1. On the "Modbus Master Configuration" page, click "Broadcast Communication Configuration". The "Broadcast Communication Configuration" page is displayed.
2. Click "Add". The "Modbus Broadcast Channel Set" dialog box is displayed.



3. Set the following parameters: Name, Access Type, Trigger Variable (SM), Repeated, Comment, and Write Register (Offset and Length).
   The access type includes multiple function codes, including Write Single Coil (Function Code 05), Write Single Register (Function Code 06), Write Multiple Coils (Function Code 15), and Write Multiple Registers (Function Code 16).

   - Trigger Variable: Condition that triggers the Modbus master to start communication. The Modbus master can perform broadcast communication only when the trigger variable is "TRUE". The trigger variable needs to be reset during programming.
   - Repeated: Number of resend times after a send operation is completed. The number of send failures can be reduced by setting the number of resend times.

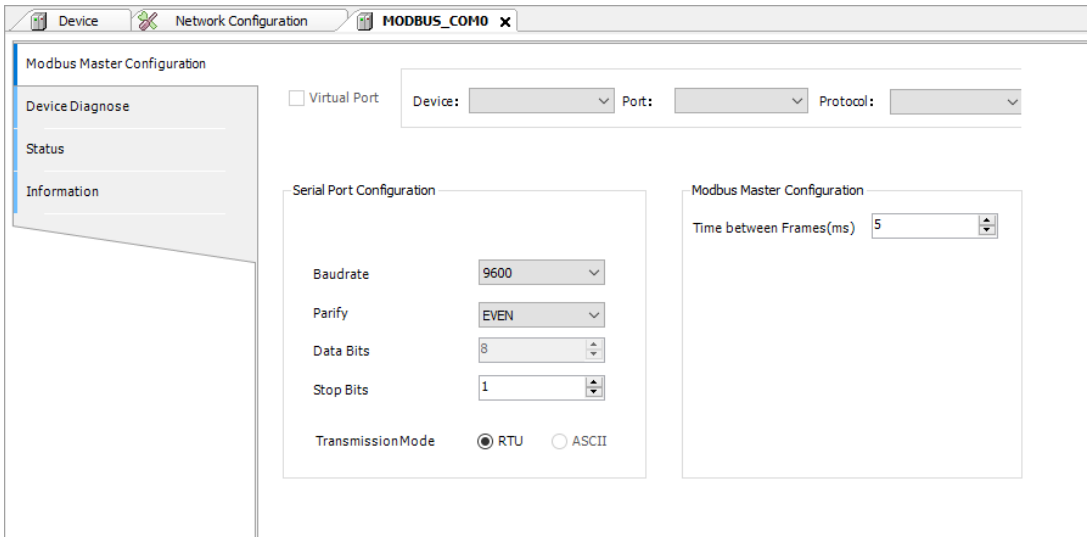
4. Click "OK".

4.5.6 Modbus Slave Configuration

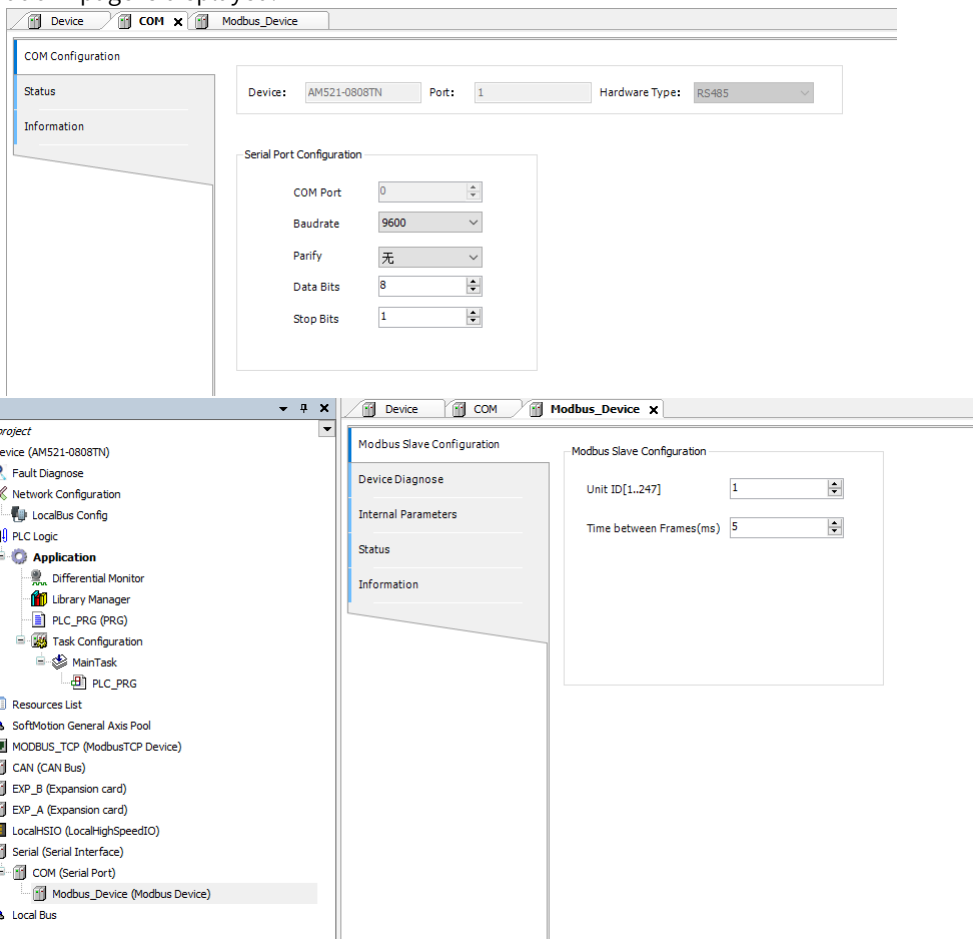
When the PLC serves as the Modbus slave, the communication parameters of the Modbus master and slave must be consistent; otherwise, communication may fail.

1. Open the "Modbus Slave Configuration" page.

   - AM400/AM600/AC700/A800: In the left device tree, double-click the Modbus slave. The "Modbus Slave Configuration" page is displayed.



- AM300/AM500: In the left device tree, double-click "COM (Serial Port)". The "COM Configuration" page is displayed. Double-click "Modbus_Slave (Modbus Slave)". The "Modbus Slave Configuration" page is displayed.



2. Configure the Modbus slave parameters.

   Among these parameters, the serial port configuration parameters have the same meanings as those of the Modbus master. The Modbus slave number refers to the local device station number. "Time between Frames" indicates the delay of responding to the master after the frame from the master is received.

4.5.7 Modbus Device Diagnosis

The Modbus master diagnosis information includes the communication configurations of faulty slaves and the fault.

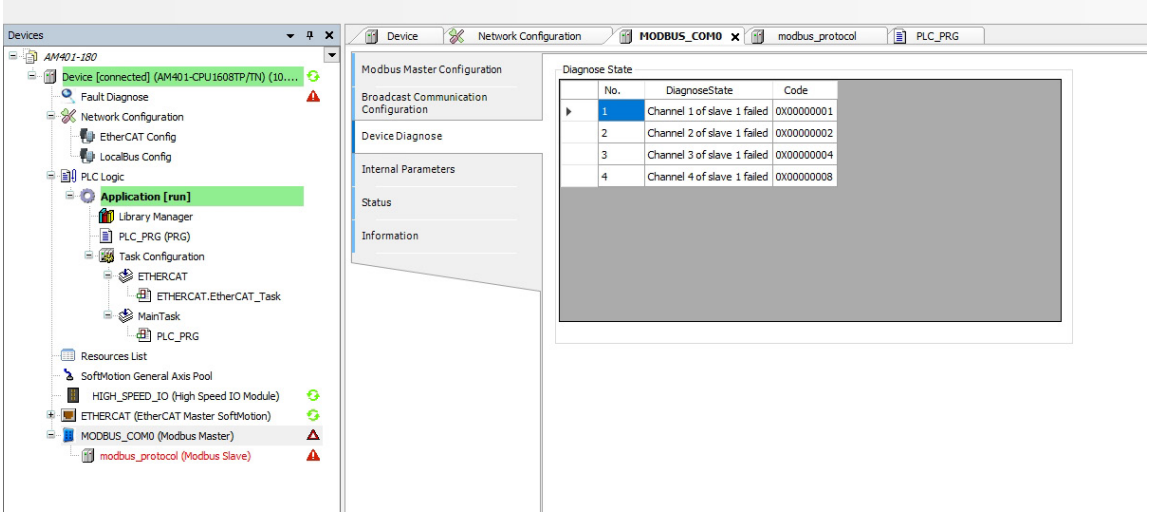


Figure 4-16 Modbus master diagnosis

4.5.8 Common Errors of Modbus

The following errors are frequently encountered during Modbus master-slave connection:

- The configurations of Modbus master and Modbus slave are inconsistent, causing a communication failure between the master and slave.
- An error response is returned when the Modbus master accesses a Modbus slave through an invalid address.
- The Modbus master receives an error response from the Modbus slave when it attempts to write a register of the Modbus slave that only supports the read operation.

Incorrect response frame

An error response consists of a slave address, command code+0x80, error code, and cyclic redundancy check (CRC) code.

The preceding error frame is applicable to all command frames.

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	Command code+0x80	1	Command error code
3	Error code	1	Value range: 1 to 4

4.5.9 Modbus Variable Addressing

Coil: Bit variable, indicated by 0 or 1 The PLC includes Q and SM area variables.

Variable	Command Code	Start Address	Number of Coils	Description
QX0.0-QX1023.7	0X01, 0x05, 0x0f	0	8192	Accessible by standard Modbus protocols.
SM0-SM7999	0x31, 0x35, 0x3f	0	8000	Use different function codes from Inovance HMI dedicated protocol.

Register: 16-bit (word) variable. The PLC includes M and SD area variables.

Variable	Command Code	Start Address	Number of Registers	Description
MW0-MW65535	0x03, 0x06, 0x10	0	65536	Accessible by standard Modbus protocols.
SD0-SD7999	0x33, 0x36, 0x40	0	8000	Use different function codes from Inovance HMI dedicated protocol.

Note:

Inovance HMI dedicated protocol uses different function codes: For the access to SM, use 0x31, 0x35 and 0x3f (0x30 added based on bit variable access command). For the access to SD, use 0x33, 0x36, and 0x40 (0x30 added based on register variable access command).

AM600 soft elements include Q, I, and M areas, which can be accessed by bit, byte, word, and double-word. For example, %QX, %QB, %QW, and %QD are converted as follows:

QB0 = (QX0.0-QX0.7)

QW0 = (QB0-QB1) = ((QX0.0-QX0.7) + (QX1.0-QX1.7))

QD0 = (QW0-QW1) = (QB0-QB3) = ((QX0.0-QX0.7) + (QX1.0-QX1.7) + (QX2.0-QX2.7) + (QX3.0-QX3.7))

Register addressing rules

Addressing by Bit	Addressing by Byte	Addressing by Word	Addressing by DWord		Addressing by Bit	Addressing by Byte	Addressing by Word	Addressing by DWord
QX0.0	QB0	QW0	QD0		MX0.0	MB0	MW0	MD0
QX0.1					MX0.1			
QX0.2					MX0.2			
QX0.3					MX0.3			
QX0.4					MX0.4			
QX0.5					MX0.5			
QX0.6					MX0.6			
QX0.7					MX0.7			
QX1.0	QB1				MX1.0	MB1		
QX1.1					MX1.1			
QX1.2					MX1.2			
QX1.3					MX1.3			
QX1.4					MX1.4			
QX1.5					MX1.5			
QX1.6					MX1.6			
QX1.7					MX1.7			
QX2.0	QB2	QW1			MX2.0	MB2	MW1	
QX2.1					MX2.1			
QX2.2					MX2.2			
QX2.3					MX2.3			
QX2.4					MX2.4			
QX2.5					MX2.5			
QX2.6					MX2.6			
QX2.7					MX2.7			
QX3.0	QB3				MX3.0	MB3		
QX3.1					MX3.1			
QX3.2					MX3.2			
QX3.3					MX3.3			
QX3.4					MX3.4			
QX3.5					MX3.5			
QX3.6					MX3.6			
QX3.7					MX3.7			
QX4.0	QB4	QW2	QD1		MX4.0	MB4	MW2	MD1
QX4.1					MX4.1			

The head address of AM600's Word registers contains an even number of bytes. The head address of DWord registers contains an even number of words. The index number is two times, facilitating address calculation.

4.5.10 Modbus Communication Frame Format

- Read coils
  The 0x01 command code is used to read the Q variable.

  The 0x31 command code is used to read the SM variable.

Request frame format: slave address + 0x01 + head address of coils + number of coils + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x01/0x31 (command code)	1	Read coils
3	Coil head address	2	Upper bits are followed by lower bits. See coil addressing.
4	Number of coils (N)	2	Upper bits are followed by lower bits.
5	CRC code	2	Upper bits are followed by lower bits.

Response frame format: slave address + 0x01 + number of bytes + coil status + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x01/0x31 (command code)	1	Read coils
3	Number of bytes	1	Value: (N + 7)/8
4	Coil status	(N + 7)/8	Every 8 coils are combined into one byte. If the number of coils is not a multiple of 8, undefined bits are filled with 0. The first 8 coils are in the first byte, and the coil with the smallest address is in the least significant bit. This pattern continues for the rest of the coils.
5	CRC code	2	Upper bits are followed by lower bits.

- Read registers

  The 0x03 command code is used to read the M variable.

  The 0x33 command code is used to read the SD variable.

  Request frame format: slave address + 0x03 + head address of registers + number of registers + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x03/0x33 (command code)	1	Read registers
3	Register head address	2	Upper bits are followed by lower bits. See register addressing.
4	Number of registers	2	Upper bits are followed by lower bits (N).
5	CRC code	2	Upper bits are followed by lower bits.

  Response frame format: slave address + 0x03 + number of bytes + register value + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x03/0x33 (command code)	1	Read registers
3	Number of bytes	1	Value: N x 2
4	Register value	N x 2	Every two bytes represent one register value, with upper bits followed by lower bits. The register with the minimum address is in the foremost.
5	CRC code	2	Upper bits are followed by lower bits.

- Write a single coil.
  The 0x05 command code is used to write the Q variable.

  The 0x35 command code is used to write the SM variable.

  Request frame format: slave address + 0x05 + coil address + coil status + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x05/0x35 (command code)	1	Write a single coil
3	Coil address	2	Upper bits are followed by lower bits. See coil addressing.
4	Coil status	2	Lower bits are followed by upper bits. Active when the value is other than 0
5	CRC code	2	Upper bits are followed by lower bits.

  Response frame format: slave address + 0x05 + coil address + coil status + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x05/0x35 (command code)	1	Write a single coil
3	Coil address	2	Upper bits are followed by lower bits. See coil addressing.
4	Coil status	2	Lower bits are followed by upper bits. Active when the value is other than 0
5	CRC code	2	Upper bits are followed by lower bits.

- Write a single register.
  The 0x06 command code is used to write the M variable.

  The 0x36 command code is used to write the SD variable.

  Request frame format: slave address + 0x06 + register address + register value + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x06/0x36 (command code)	1	Write a single register
3	Register address	2	Upper bits are followed by lower bits. See register value addressing.
4	Register value	2	Upper bits are followed by lower bits. Active when the value is other than 0
5	CRC code	2	Upper bits are followed by lower bits.

  Response frame format: slave address + 0x06 + register address + register value + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x06/0x36 (command code)	1	Write a single register
3	Register address	2	Upper bits are followed by lower bits. See register addressing.

No.	Meaning of Data (Byte)	Number of Bytes	Description
4	Register value	2	Upper bits are followed by lower bits. Active when the value is other than 0
5	CRC code	2	Upper bits are followed by lower bits.

- Write multiple coils

  The 0x0f command code is used to write multiple consecutive Q variables.

  The 0x3f command code is used to write multiple consecutive SM variables.

  Request frame format: slave address + 0x0f + head address of coils + number of coils + number of bytes + coil status + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x0f/0x3f (command code)	1	Write multiple coils
3	Coil head address	2	Upper bits are followed by lower bits. See coil addressing.
4	Number of coils (N)	2	Upper bits are followed by lower bits. The maximum value is 1968.
5	Number of bytes	1	Value: (N + 7)/8
6	Coil status	(N + 7)/8	Every 8 coils are combined into one byte. If the number of coils is not a multiple of 8, undefined bits are filled with 0. The first 8 coils are in the first byte, and the coil with the smallest address is in the least significant bit. This pattern continues for the rest of the coils.
7	CRC code	2	Upper bits are followed by lower bits.

  Response frame format: slave address + 0x05 + head address of coils + number of coils + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x0f/0x3f (command code)	1	Write multiple coils
3	Coil head address	2	Upper bits are followed by lower bits. See coil addressing.
4	Number of coils	2	Upper bits are followed by lower bits.
5	CRC code	2	Upper bits are followed by lower bits.

- Write multiple registers

  The 0x10 command code is used to write multiple consecutive M variables.

  The 0x40 command code is used to write multiple consecutive SD variables.

  Request frame format: slave address + 0x10 + head address of registers + number of registers + number of bytes + register value + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x10/0x40 (command code)	1	Write multiple registers
3	Register head address	2	Upper bits are followed by lower bits. See register addressing.

No.	Meaning of Data (Byte)	Number of Bytes	Description
4	Number of registers	2	Upper bits are followed by lower bits (N).
5	Number of bytes	1	Value: N x 2
6	Register value	N x 2 (N x 4)	
7	CRC code	2	Upper bits are followed by lower bits.

Response frame format: slave address + 0x05 + head address of coils + number of coils + CRC code

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Slave address	1	Value range: 1 to 247
2	0x10/0x40 (command code)	1	Write multiple registers
3	Register head address	2	Upper bits are followed by lower bits. See register addressing.
4	Number of registers	2	Upper bits are followed by lower bits (N).
5	CRC code	2	Upper bits are followed by lower bits.

4.6 Application of Free Protocols on Serial Ports

4.6.1 Overview

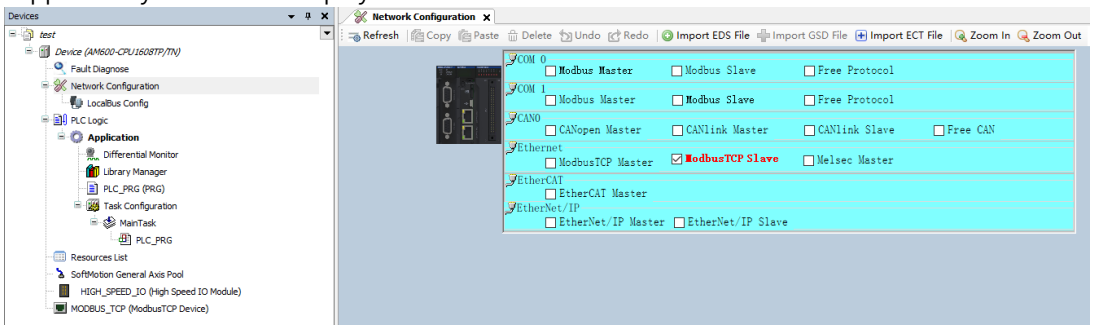
This section describes communication using free protocols on the serial ports of a medium-sized PLC. The applicable versions are as follows:

Device Name	Version
InoProShop	1.2.0 or later
PLC firmware	1.19.70 or later

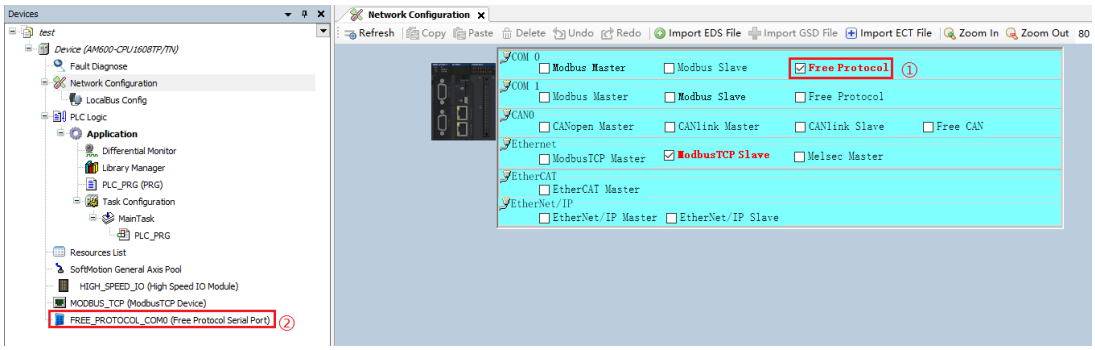
4.6.2 Serial Port Configuration

- Enable the free protocol for the AM400/AM600/AC700/A800 series

1. In the left device tree, double-click "Network Configuration". The "Network Configuration" page is displayed.
2. Click the PLC device picture. The enabling states of all masters/slaves and free protocols supported by the PLC are displayed.

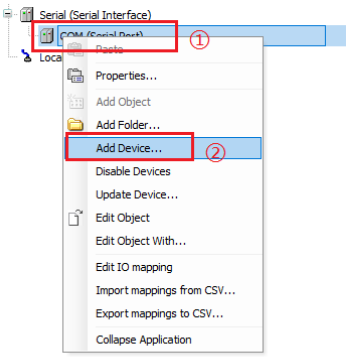


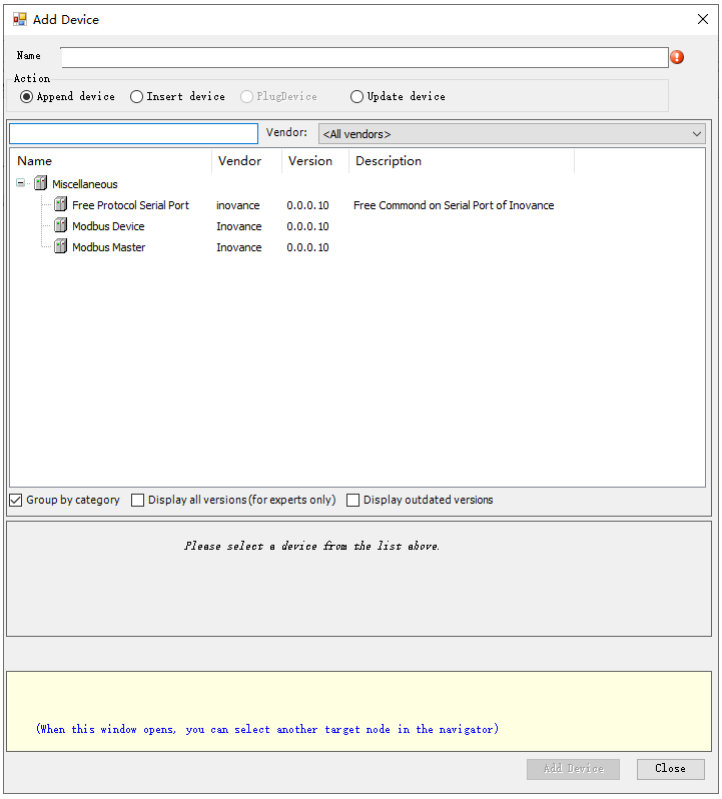
3. Take the COM0 port as an example. Select "Free Protocol" for the COM0 port.



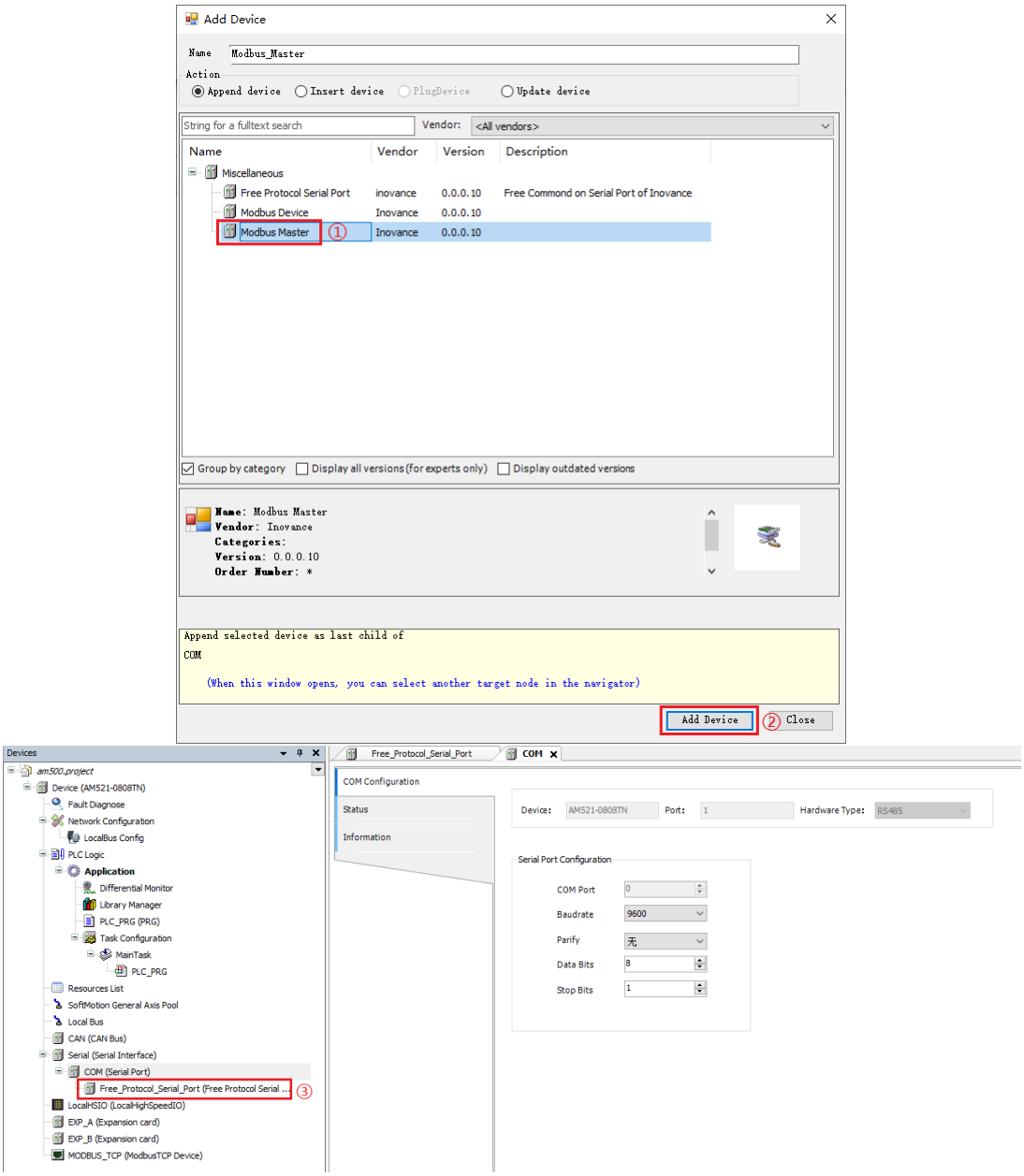
- Enable the free protocol for the AM300/AM500 series

1. In the left device tree, right-click "COM (Serial Port)". In the shortcut menu, select "Add Device". The "Add Device" dialog box is opened.

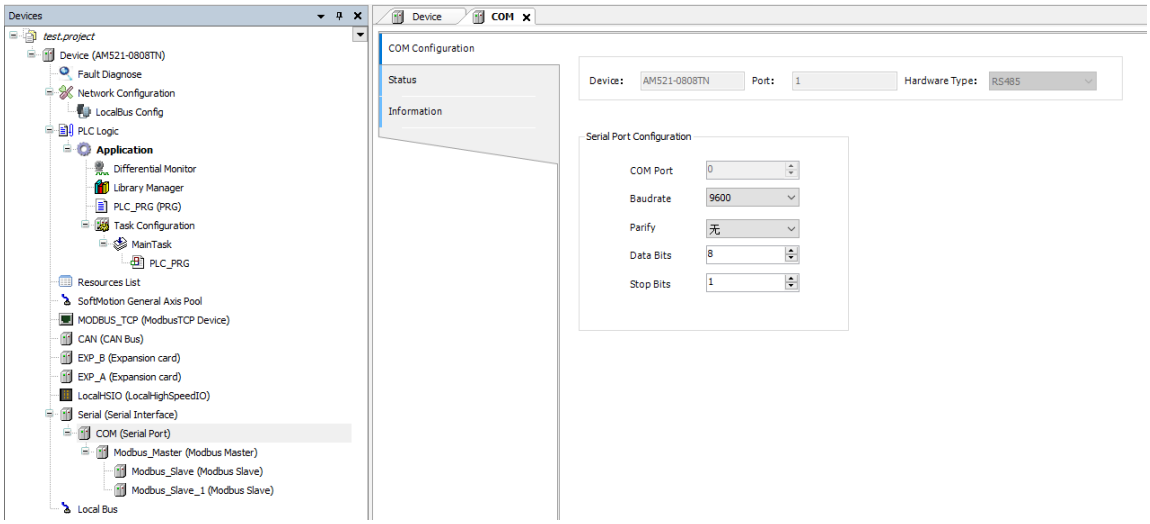




2. Select "Free Protocol Serial Port" and then click "Add Device". The free protocol is enabled for the PLC.


Add Device
Name Modbus_Master
Action
Append device
Insert device
PlugDevice
Update device
String for a fulltext search
Vendor: <All vendors>
Name Vendor Version Description
Miscellaneous
Free Protocol Serial Port inovance 0.0.0.10 Free Commond on Serial Port of Inovance
Modbus Device Inovance 0.0.0.10
Modbus Master ① Inovance 0.0.0.10
Group by category
Display all versions (for experts only)
Display outdated versions
Name: Modbus Master
Vendor: Inovance
Categories:
Version: 0.0.0.10
Order Number: *
Append selected device as last child of
COM
(When this window opens, you can select another target node in the navigator)
Add Device
② Close
Devices
am500.project
Device (AM521-0808TN)
Fault Diagnose
Network Configuration
LocalBus Config
PLC Logic
Application
Differential Monitor
Library Manager
PLC_PRG (PRG)
Task Configuration
MainTask
PLC_PRG
Resources List
SoftMotion General Axis Pool
Local Bus
CAN (CAN Bus)
Serial (Serial Interface)
COM (Serial Port)
Free_Protocol_Serial_Port (Free Protocol Serial ...
③
LocalHSIO (LocalHighSpeedIO)
EXP_A (Expansion card)
EXP_B (Expansion card)
MODBUS_TCP (ModbusTCP Device)
Free_Protocol_Serial_Port
COM
COM Configuration
Status
Information
Device: AM521-0808TN
Port: 1
Hardware Type: RS485
Serial Port Configuration
COM Port 0
Baudrate 9600
Parify 无
Data Bits 8
Stop Bits 1

4.6.3 Communication Configuration



The following table lists the parameters that need to be set.

Configuration Item	Function
COM Port	Serial port 0 or 1, which is used to establish a physical connection to the master
Baud rate	Communication rate
Parity	Method of verifying communication frames
Data Bits	Actual data bits included in communication frames
Stop Bits	Last bit in a single message during communication

The communication format used by the serial port with "Free Protocol" selected must be consistent with that of the connected slave, for example, 9600 8 E 1.

4.6.4 Registers for Data Sending and Receiving

The following figure shows the configurations of a port enabled with free protocol.

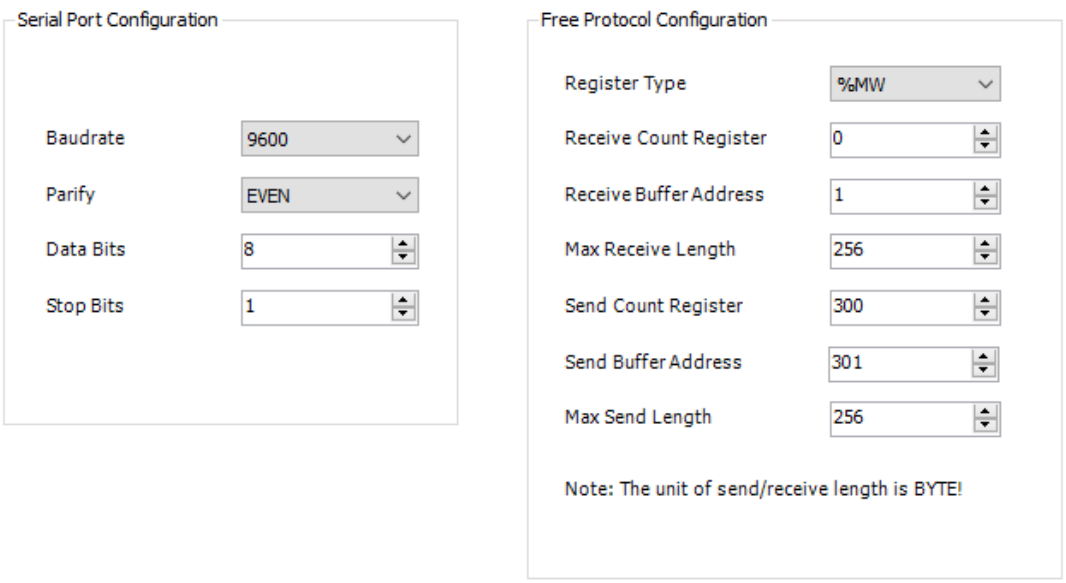


Figure 4-17 Free Protocol Configuration

Configuration Item	Function
Register Type	Options are %MW and SD.
Receive Count Register	This register displays the number of received bytes when data is received.
Receive Buffer Address	This parameter indicates the head address of the buffer that receives data.
Max Receive Length	This parameter indicates the maximum number of buffered bytes.
Send Count Register	Data is sent when this parameter is not set to 0. Its value is automatically reset after data is sent.
Send Buffer Address	This parameter indicates the head address of the buffer that sends data.
Max Send Length	This parameter indicates the maximum number of data bytes sent at a time.

In the preceding figure, the option "%MW0" is set and indicates the length of data frames received from external devices. "MW0" needs to be cleared manually; otherwise, its value keeps increasing until it is cleared when the value of "Max Receive Length" is reached. In this case, the receive buffer is overwritten from the start.

For example, if the receive length is "%MW0 = 10", the receive buffer ranges from %MW1 to %MW5. One %MW occupies 2 bytes.

For example, if the send length is "%MW300 = 8", the send buffer ranges from %MW301 to %MW304. Data is automatically sent when %MW300 is not 0. %MW300 is automatically cleared after data is sent.

Application example:

The process based on the preceding settings is as follows:

1. When data is received, %MW0 displays the number of received data bytes, and the received data is stored in the registers starting from the head address %MW1.
2. Each time after data is received, %MW0 must be cleared manually so that data is buffered all over again starting from %MW1. If %MW0 is not cleared, data is buffered in sequence.

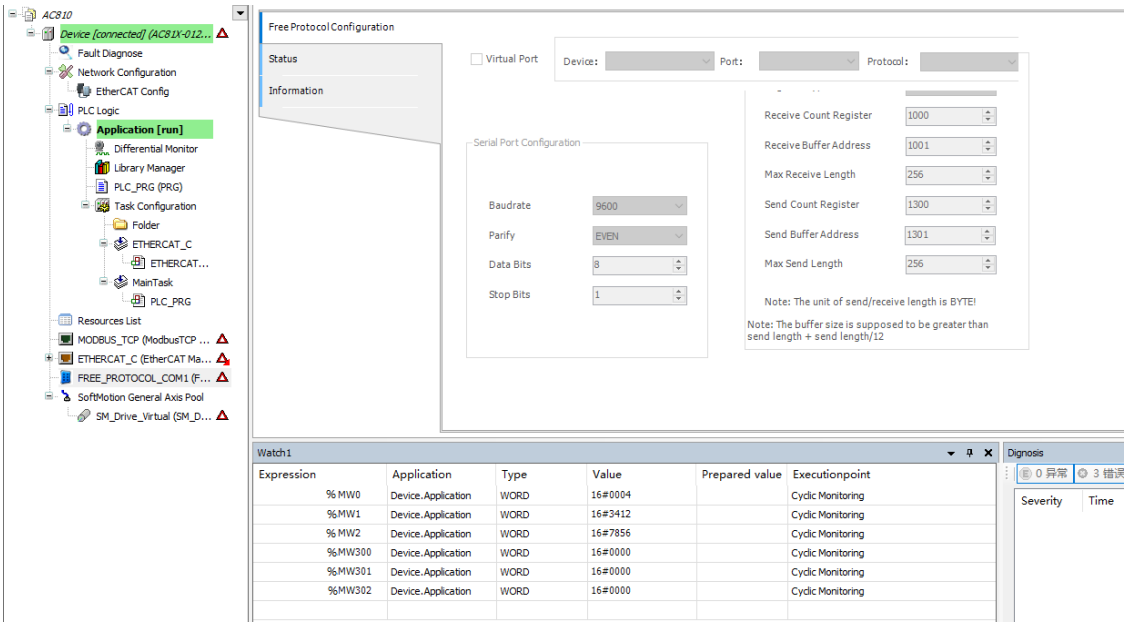
3. When the length of received data exceeds the value 256 (bytes) of "Max Receive Length", %MW0 starts counting again and the received data is stored from the head address %MW1.
4. Before data is sent, the data is written to the registers starting from the head address %MW301.
5. After the data (bytes) to be sent is written to %MW300, data starting from %MW301 is sent.
6. %MW300 is automatically cleared after data is sent.
7. When the number of bytes written to %MW300 exceeds the value 256 of "Max Send Length", data is sent based on "Max Send Length".

4.6.5 Communication Tests Through the Serial Port Commissioning Assistant

1. Receive data by the PLC

   Open "Serial Port Commissioning Assistant" and set the serial port communication parameters as those on the programming software as follows: baud rate = 9600, check mode = NONE, data bit = 8, and stop bit = 1.

   Use the serial port commissioning assistant to send the 4-byte data "12 34 56 78" in hexadecimal. The length %MW0 of the data received by the PLC is 4 bytes. The buffer that receives data ranges from %MW1 to %MW2. %MW0 must be manually cleared before data is received by the PLC again through the serial port commissioning assistant; otherwise, the buffer receiving the data is not updated.



2. Send data from the PLC

   If the PLC wants to send data of 4 bytes, the data value is written to the data sending buffers %MW301 to %MW302 and 4 is written to the buffer %MW300. Then, the data is automatically sent. After data is sent, %MW300 is automatically cleared and the receiving area of the serial port commissioning assistant can receive the data sent by the PLC.

4.7 Modbus TCP Device Editor

4.7.1 Overview

Click a PLC on the "Network Configuration" page. A window is displayed, allowing you to enable the master and slaves supported by the PLC, as shown in the following figure. Select the check box before the master or slave you want to enable, and double-click "MODBUS_TCP" in "Network Devices List" on the right to add the slave to the network.

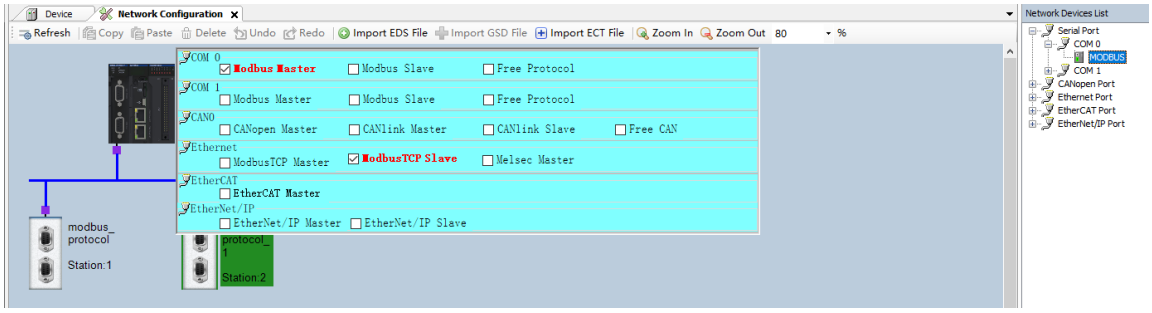


Figure 4-18 Modbus TCP configuration example

The device tree corresponding to Modbus TCP configuration is displayed on the left, as shown in the following figure.

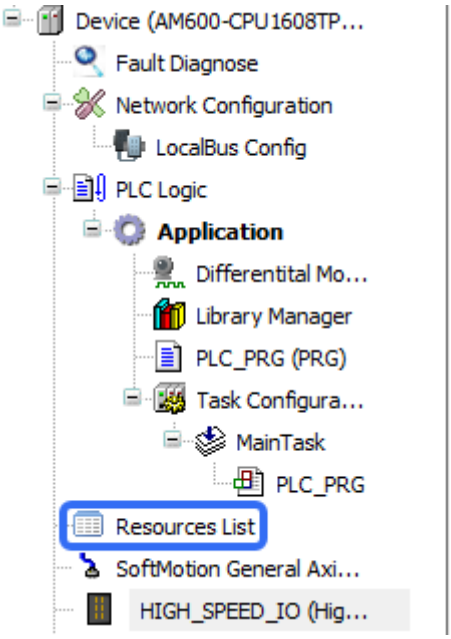


Figure 4-19 Device tree corresponding to Modbus TCP configuration

For Modbus TCP communication specifications of medium-sized PLCs, see the *user guide* of the specific product.

4.7.2 Modbus TCP Master Configuration

When the PLC serves as the Modbus TCP master, double-click a master in the device tree. The "Modbus Master Configuration" window is displayed, as shown in the following figure.

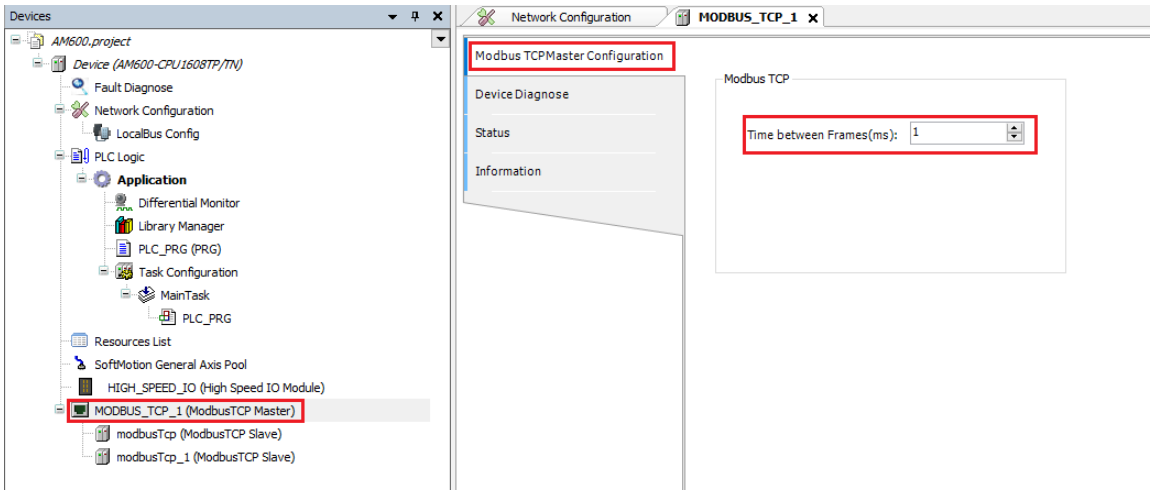


Figure 4-20 Modbus TCP Master Configuration

"Time between Frames" refers to the time for the master to wait for the next request data frame after receiving a response data frame. This parameter can be used to adjust the data exchange rate.

4.7.3 Modbus TCP Master Communication Configuration

Modbus TCP slave configuration

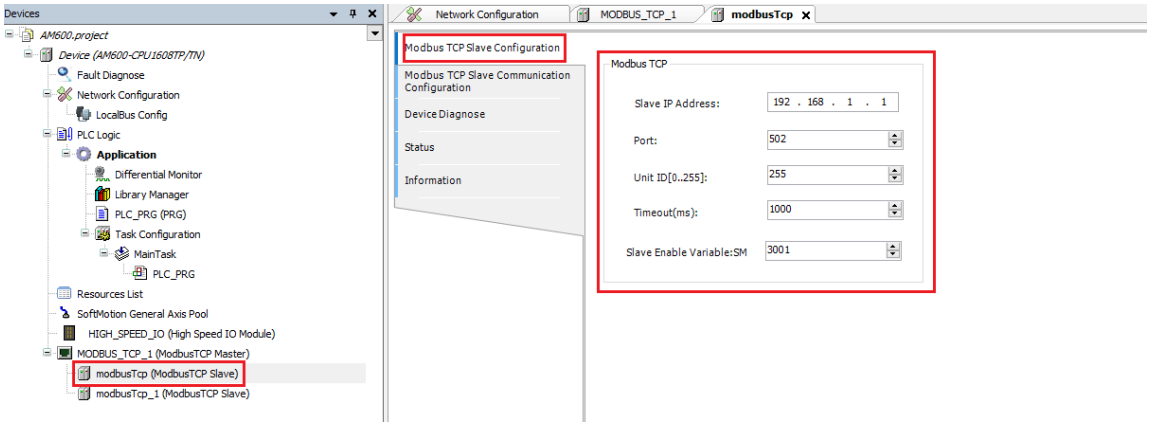


Figure 4-21 Modbus TCP master-slave connection configuration

Configuration parameters:

Configuration Item	Function
Slave IP Address	IP address of the Modbus TCP slave used to connect to the master.
Port	TCP port of the Modbus TCP slave used to connect to the master.
Unit ID	Protocol station address of the Modbus TCP slave used to connect to the master.
Timeout	After sending frames, the master reports receiving timeout if no data is received from the slave within this timeout period.
Slave Enable Variable	Enables the slave by programming and starts to send frames to the slave.

Example:

Configuration Item	Value
Slave IP Address	192.168.10.16
Port	502
Unit ID	05
Timeout	1000
Slave Enable Variable	3001

Configuring Modbus TCP master-slave communication

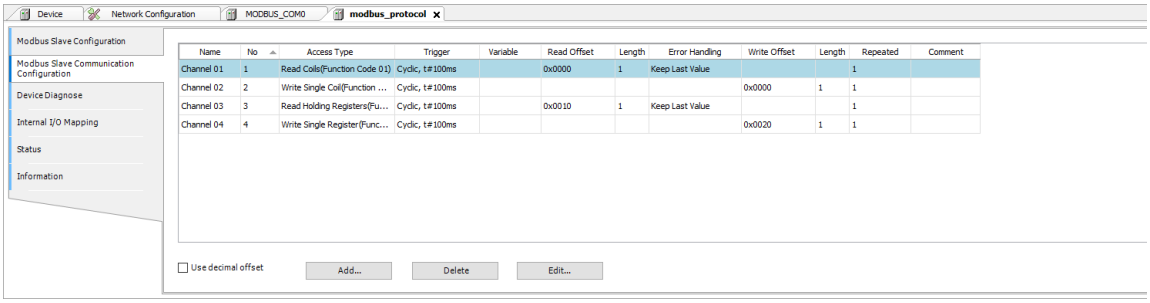


Figure 4-22 Configuring Modbus TCP master-slave communication

In the preceding figure, each channel represents an independent Modbus TCP request. The "Access Type" column defines the 5-ms cyclic operation triggered by "Write Single Register (function code 06)" to write 2-byte data (one register occupies 2 bytes) to the register whose offset address is 0x0020.

After you click "Add...", a dialog box for adding a channel for the Modbus TCP slave is displayed. Complete settings and click "OK" to create a channel.

Select a channel from the Modbus TCP slave channel list and click "Edit...". The "Modbus Channel Set" dialog box is displayed. Change the values of parameters to modify the channel settings. Click "OK" to update the channel settings.

You can set the following parameters to add or edit a channel:

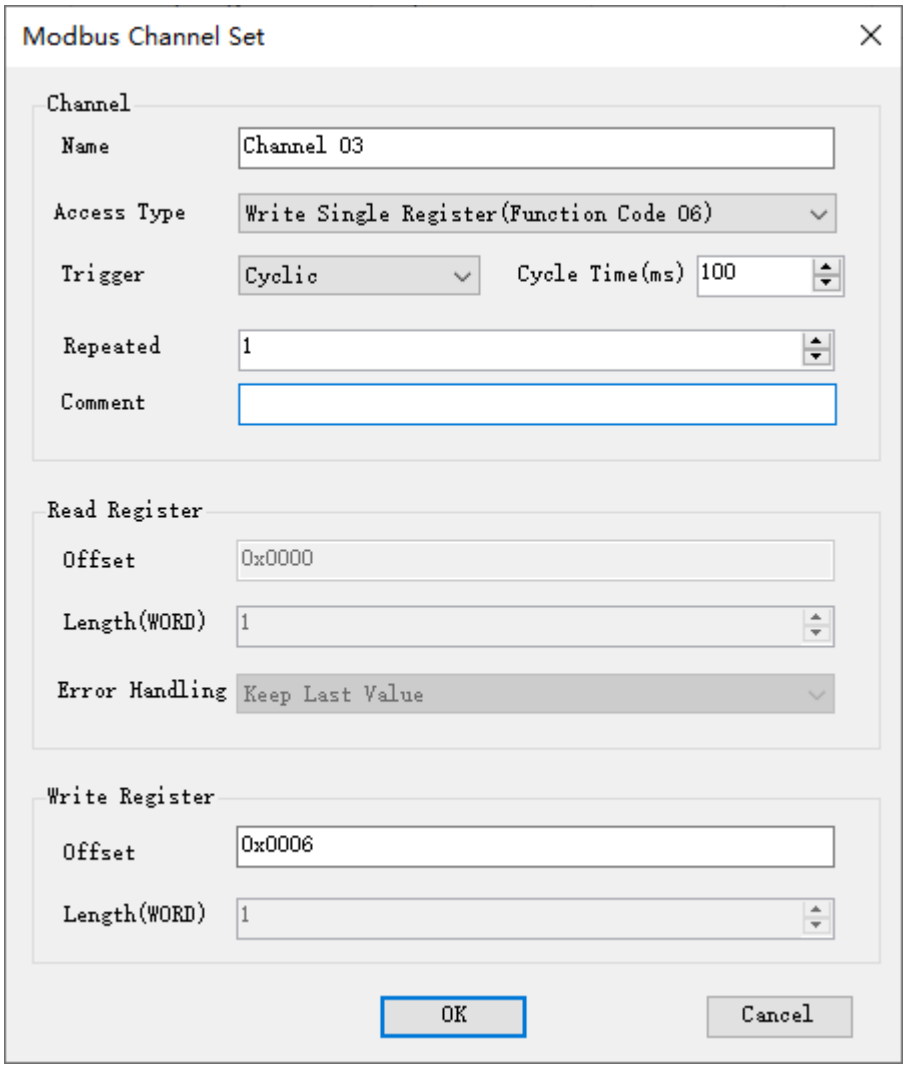


Figure 4-23 Dialog box for Modbus TCP master-slave communication configuration

Modbus communication parameter settings

Configuration Item	Function	
Name	Channel name, in the string format.	
Access Type	Read Coils (Function Code 01). Read Discrete Inputs (Function Code 02). Read Holding Registers (Function Code 03). Read Input Registers (Function Code 04). Write Single Coil (Function Code 05). Write Single Register (Function Code 06). Write Multiple Coils (Function Code 15). Write Multiple Registers (Function Code 16).	
Trigger	Cyclic: Requests are triggered periodically.	Cycle Time: Time for re-execution.
	Level Trigger: Requests are triggered when a change is made during programming.	Trigger Variable (SM): SM element that implements trigger.
Repeated	A request is resent for the specified times when no response frame is received from the slave upon a communication error.	

Configuration Item	Function
Comment	Brief text description about data.
Read Register	—
Offset	Head address of the registers to be read.
Length	Number of registers to be read.
Error Handling	Keep Last Value: The last valid value is kept. 0: All the values are zeroed.
Write Register	—
Write Register	Head address of the registers to be written.
Length	Number of registers to be written.

The valid range of the "Length" parameter depends on the following parameters:

Function Code	Access Type	Register Count
01	Read Coils	1 to 2000
02	Read Discrete Inputs	1 to 2000
03	Read Holding Registers	1 to 125
04	Read Input Registers	1 to 125
05	Write Single Coil	1
06	Write Single Register	1
15	Write Multiple Coils	1 to 1968
16	Write Multiple Registers	1 to 123

Modbus TCP slave internal I/O mapping

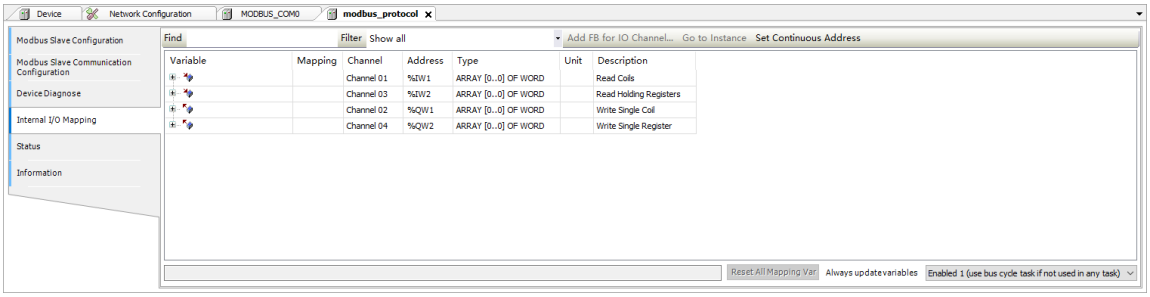


Figure 4-24 Internal I/O mapping of Modbus TCP master-slave connection

After master-slave communication configurations are added on the "Modbus TCP Slave Communication Configuration" page, the mapped address of each configuration is automatically allocated in "Internal I/O Mapping". In the preceding figure, %IW1 in the first row indicates that the read value of a coil to the address %IW1. Besides, you can map the customized variables in the program to the I/O address through the input assistant or directly inputting the example variable path.

4.7.4 Modbus TCP Slave Configuration

When the PLC serves as the Modbus TCP slave, double-click a slave in the device tree. The "Modbus TCP Slave Configuration" window is displayed, as shown in the following figure.

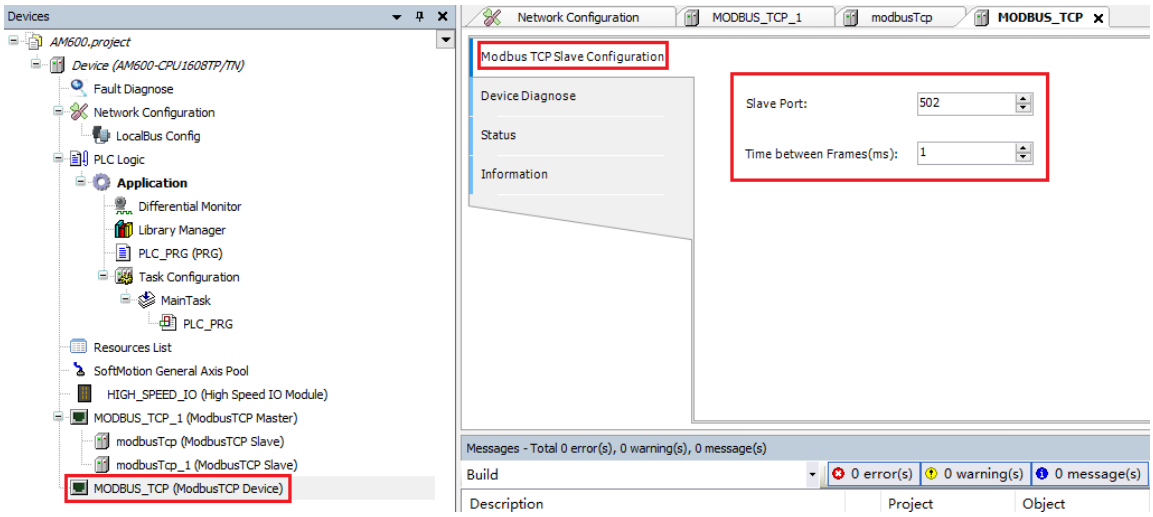


Figure 4-25 Modbus TCP slave configuration

Configuration parameters:

Configuration Item	Function
Slave Port	TCP port of the Modbus TCP slave
Time between Frames	Delay for the Modbus TCP slave to return a response frame after receiving a communication frame

Example:

Configuration Item	Value
Slave Port	502
Time between Frames	1

4.7.5 Modbus TCP Device Diagnosis

Modbus TCP master diagnosis

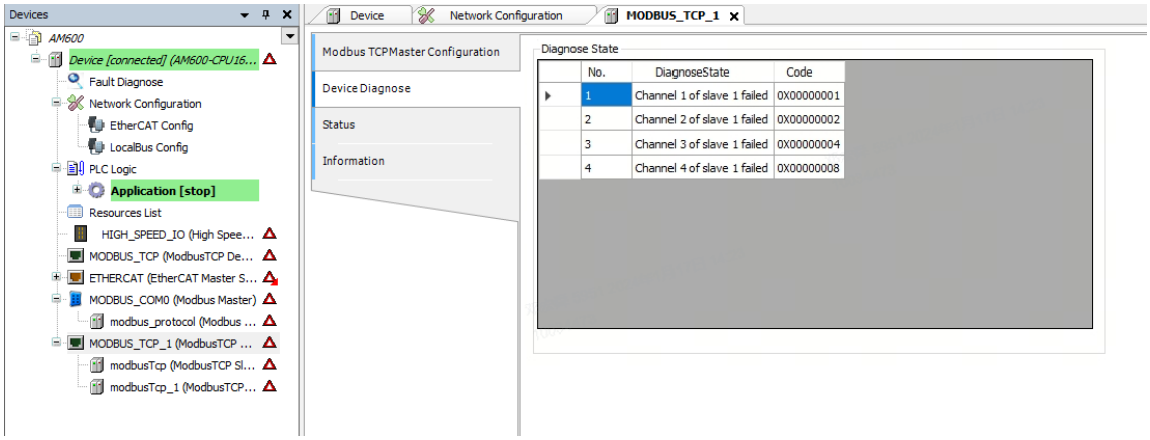


Figure 4-26 Modbus TCP master diagnosis

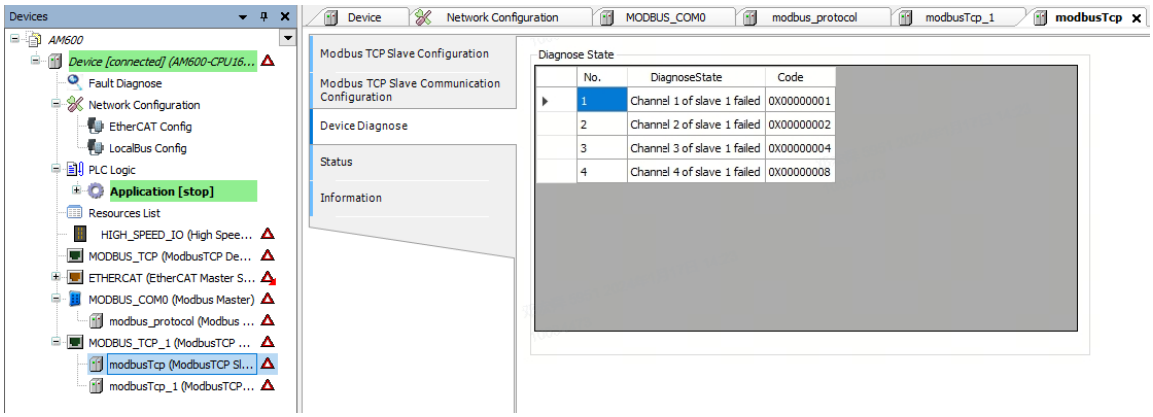


Figure 4-27 Modbus TCP master-slave connection diagnosis

Modbus TCP slave diagnosis

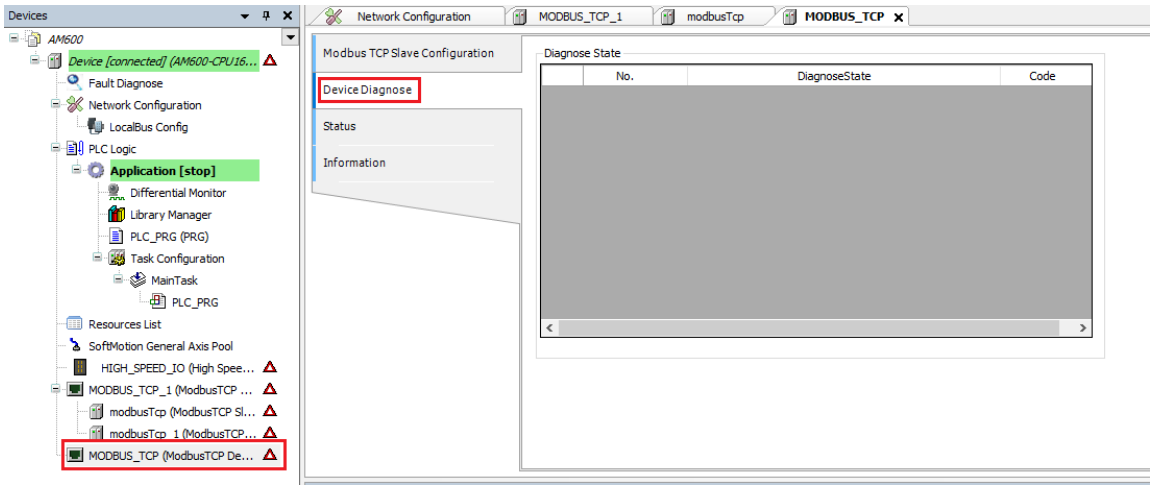


Figure 4-28 Modbus TCP slave diagnosis

4.7.6 Common Errors of Modbus TCP

The following errors are frequently encountered during Modbus TCP master-slave connection:

- The IP addresses configured for Modbus TCP master-slave connection are incorrect, causing a communication failure.
- An error response is returned when the Modbus TCP master accesses a Modbus TCP slave through an invalid address.
- The Modbus TCP master receives an error response from the Modbus TCP slave when it attempts to write a register of the Modbus TCP slave that only supports the read operation.

An error frame consists of a transaction meta identifier, protocol identifier, length, slave address, (command code+0x80), error code, and cyclic redundancy check (CRC) code.

The preceding error frame is applicable to all command frames.

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/ response transaction being processed
2	Protocol identifier	2	0: Modbus protocol

No.	Meaning of Data (Byte)	Number of Bytes	Description
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	Command code+0x80	1	Command error code
6	Error code	1	1 to 4

4.7.7 Modbus TCP Communication Frame Format

- Read coils

  The 0x01 command code is used to read the Q variable.

  The 0x31 command code is used to read the SM variable.

  Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x01 + head address of coils + number of coils

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x01/0x31 (command code)	1	Read coils
6	Coil head address	2	Upper bits are followed by lower bits. See coil addressing.
7	Number of coils (N)	2	Upper bits are followed by lower bits.

  Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x01 + number of bytes + coil status

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x01/0x31 (command code)	1	Read coils
6	Number of bytes	1	Value: (N + 7)/8
7	Coil status	(N + 7)/8	Every 8 coils are combined into one byte. If the number of coils is not a multiple of 8, undefined bits are filled with 0. The first 8 coils are in the first byte, and the coil with the smallest address is in the least significant bit. This pattern continues for the rest of the coils.

- Reads registers

  The 0x03 command code is used to read the M variable.

  The 0x33 command code is used to read the SD variable.

Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x03 + head address of registers + number of registers

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x03/0x33 (command code)	1	Reads registers
6	Register head address	2	Upper bits are followed by lower bits. See register addressing.
7	Number of registers (N)	2	Upper bits are followed by lower bits.

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x03 + number of bytes + register value

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x03/0x33 (command code)	1	Reads registers
6	Number of bytes	1	Value: N x 2
7	Register value	(N x 2) bytes	Every two bytes represent one register value, with upper bits followed by lower bits. The register with the minimum address is in the foremost.

- Writes a single coil.
  The 0x05 command code is used to write the Q variable.

  The 0x35 command code is used to write the SM variable.

  Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x05 + coil address + coil status

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x05/0x35 (command code)	1	Write a single coil
6	Coil address	2	Upper bits are followed by lower bits. See coil addressing.
7	Coil status	2	Lower bits are followed by upper bits. A non-zero value is valid.

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x05 + coil address + coil status

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x05/0x35 (command code)	1	Write a single coil
6	Coil address	2	Upper bits are followed by lower bits. See coil addressing.
7	Coil status	2	Lower bits are followed by upper bits. A non-zero value is valid.

- Writes a single register.
  The 0x06 command code is used to write the M variable.

  The 0x36 command code is used to write the SD variable.

  Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x06 + register address + register value

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x06/0x36 (command code)	1	Write a single register
6	Register address	2	Upper bits are followed by lower bits. See register value addressing.
7	Register value	2	Upper bits are followed by lower bits. A non-zero value is valid.

  Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x06 + register address + register value

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x06/0x36 (command code)	1	Write a single register
6	Register address	2	Upper bits are followed by lower bits. See register addressing.
7	Register value	2	Upper bits are followed by lower bits. Active when the value is other than 0

- Writes multiple coils

  The 0x0f command code is used to write multiple consecutive Q variables.

  The 0x3f command code is used to write multiple consecutive SM variables.

  Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x0f + head address of coils + number of coils + number of bytes + coil status

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x0f/0x3f (command code)	1	Write multiple coils
6	Coil head address	2	Upper bits are followed by lower bits. See coil addressing.
7	Number of coils (N)	2	Upper bits are followed by lower bits. The maximum value is 1968.
8	Number of bytes	1	Value: (N + 7)/8
9	Coil status	(N + 7)/8	Every 8 coils are combined into one byte. If the number of coils is not a multiple of 8, undefined bits are filled with 0. The first 8 coils are in the first byte, and the coil with the smallest address is in the least significant bit. This pattern continues for the rest of the coils.

  Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x05 + head address of coils + number of coils

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x0f/0x3f (command code)	1	Write multiple coils
6	Coil head address	2	Upper bits are followed by lower bits. See coil addressing.
7	Number of coils	2	Upper bits are followed by lower bits.

- Writes multiple registers

  The 0x10 command code is used to write multiple consecutive M variables.

  The 0x40 command code is used to write multiple consecutive SD variables.

  Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x10 + head address of registers + number of registers + number of bytes + register value

No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol

No.	Meaning of Data (Byte)	Number of Bytes	Description
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x10/0x40 (command code)	1	Writes multiple registers
6	Register head address	2	Upper bits are followed by lower bits. See register addressing.
7	Number of registers	2	Upper bits are followed by lower bits.
8	Number of bytes	1	Value: N x 2
9	Register value	N x 2 (N x 4)	

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x05 + head address of coils + number of coils

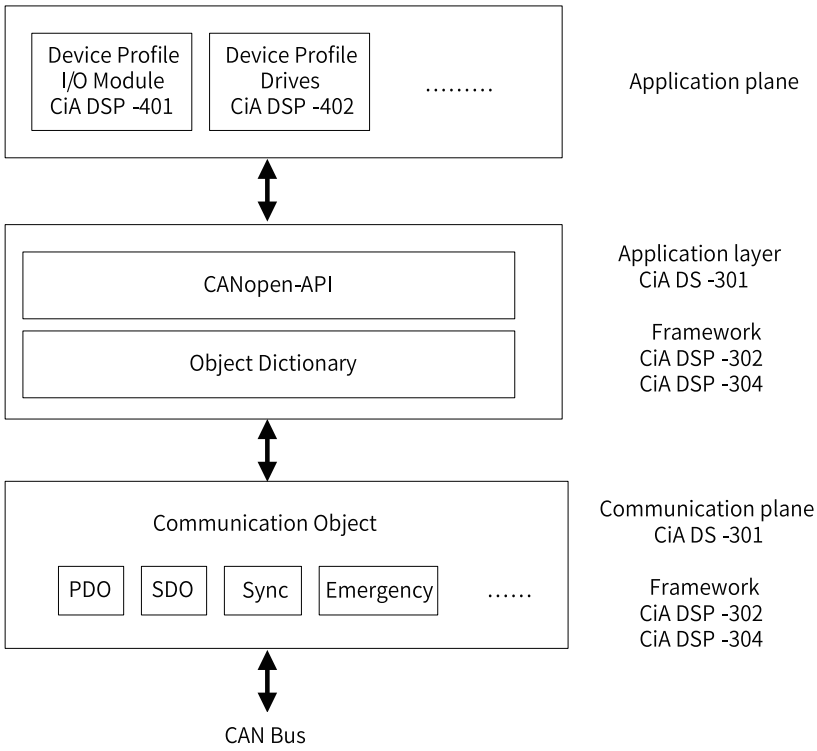
No.	Meaning of Data (Byte)	Number of Bytes	Description
1	Transaction meta identifier	2	Identifier of the Modbus request/response transaction being processed
2	Protocol identifier	2	0: Modbus protocol
3	Length	2	Number of the following bytes
4	Slave address	1	Value range: 1 to 247
5	0x10/0x40 (command code)	1	Writes multiple registers
6	Register head address	2	Upper bits are followed by lower bits. See register addressing.
7	Number of registers	2	Upper bits are followed by lower bits.

4.8 CANopen Network

4.8.1 Overview of CANopen Communication

CANopen bus

CANopen is the industrial communication protocol family for distributed automatic control devices based on the CAN bus. CANopen is promoted by CAN in Automation (CiA) and was standardized at the end of 2002, with a standard number of CENELEC EN 50325-4. CANopen defines application-layer protocols, communication-layer protocols, and multiple application protocols. The following figure shows the overall architecture of CANopen.



The application layer provides applications with acknowledged and unacknowledged services and defines communication objects. Those services can be used to request data from a server.

Communication objects are used during data exchange. They are used to exchange process data and service data, manage processes or synchronize the system clock, manage error statuses, and control and monitor node statuses. A communication object is defined by a structure, transmission type, and CAN identifier. Typical parameters of communication objects include the CAN identifier for data transmission, message transmission type, and disable time or event time. The parameters are defined in communication protocols.

For each CANopen device, the main data structure is an object dictionary, which is the primary data exchange medium for the communication between the application layer and CAN bus. Object dictionary portals can be accessed from the application layer and CAN bus through special messages. Those portals can be considered as a variable or a programmer-defined area.

Each portal has an index or a sub-index. An object dictionary portal can be accurately located by using the index structure. The CANopen protocol stack provides standard API functions to define object dictionary portals, including their read and write attributes. Object dictionaries can be accessed by using communication objects through the CAN bus.

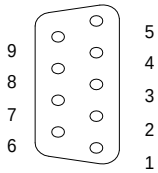
In object dictionaries, attributes of each portal must be defined, including the data type, access permission, data transmission for process data objects (PDOs), and variable range.

- PDOs are used to transmit the real-time data of processes.
- Service data objects (SDOs) are used to access the object dictionary of a device, set parameters, or transmit non-real-time data.
- SYNC messages do not contain data. They are sent by the producer to the CAN network periodically to trigger transmission of PDO data on nodes.
- Emergency messages are triggered by the internal critical errors of devices. They consist of an error code, error register, and manufacturer specific error.

CiA DS-301 defines the application layer and communication protocols. CiA DSP-302 defines the framework of programmable CANopen devices. CiA DSP-304 defines the framework of secure redundant data transmission, whereas the data description of specific devices is defined in the application protocol consisting of respective device protocols. For example, CiA DSP-401 defines the data format of the I/O module, and CiA DSP-402 defines the data format for drive control.

Hardware port

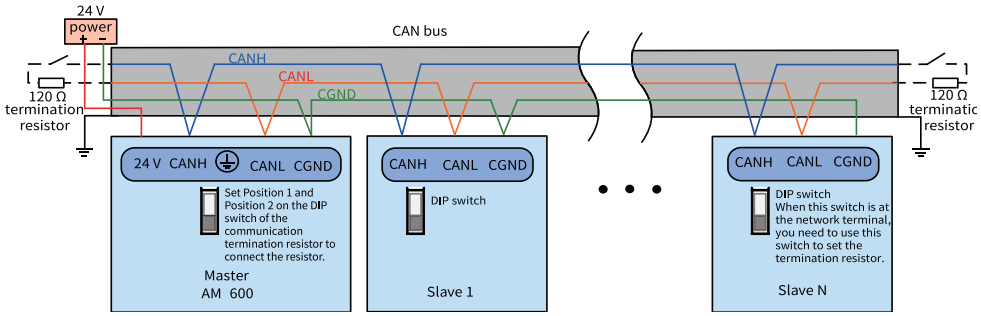
CANopen transmits data through the DB9 connector. The following table lists pins of the DB9 connector.

Diagram	Pin	Signal Definition
	PIN2	CANL
	PIN7	CANH
	PIN3	CGND

It is recommended to use shielded twisted pairs to connect the CAN bus. Connect two 120 Ω termination resistors at each end of the bus to prevent signal reflection. Typically, ground the shield in the single-point grounding mode. Do not bundle the bus cables together with AC power cords and high voltage cables; otherwise, communication signals may be affected by interference.

Networking diagram

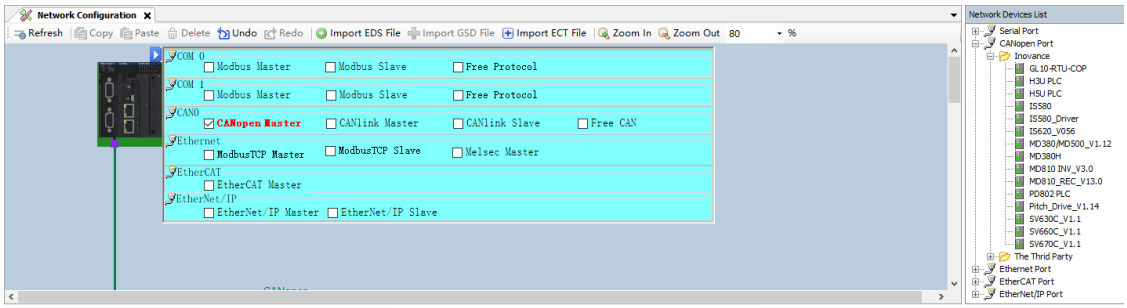
The following figure shows the CAN bus topology.



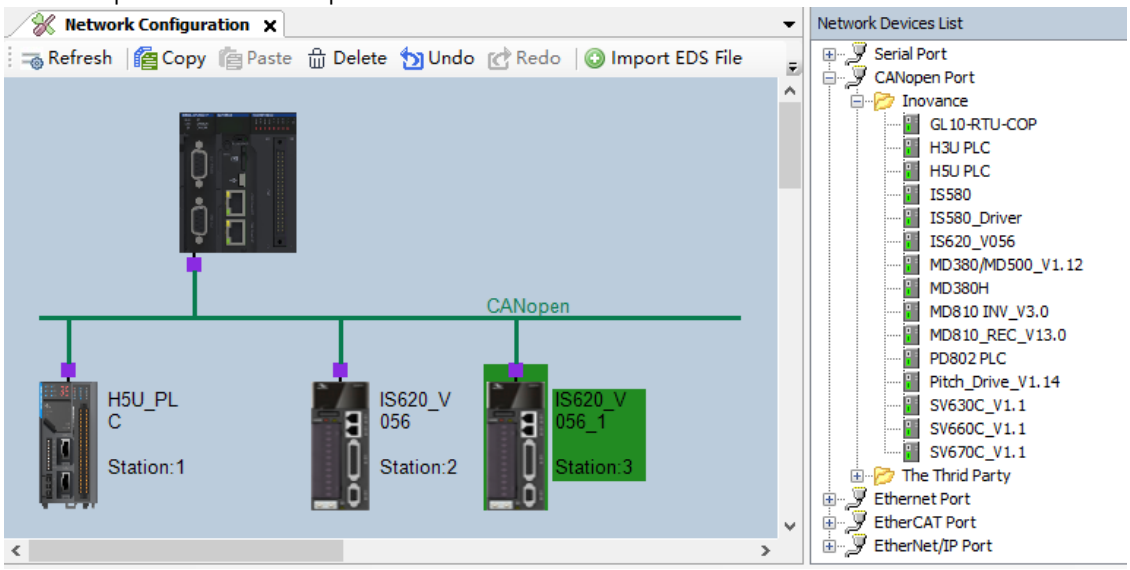
Process of using CANopen

The general process of using CANopen is as follows:

1. Design the CANopen hardware network structure.
2. On the "Network Configuration" page, activate the CANopen bus. After the CANopen bus is activated, the CANopen master is automatically added, and the CANopen bus task named "CANopen" is also added. By default, the CANopen bus uses the task to refresh I/O.
3. On the "Network Configuration" page, add CANopen slaves and modules based on the hardware structure. Before adding a third-party slave, import an EDS file to import the third-party slave on the "Network Configuration" page.

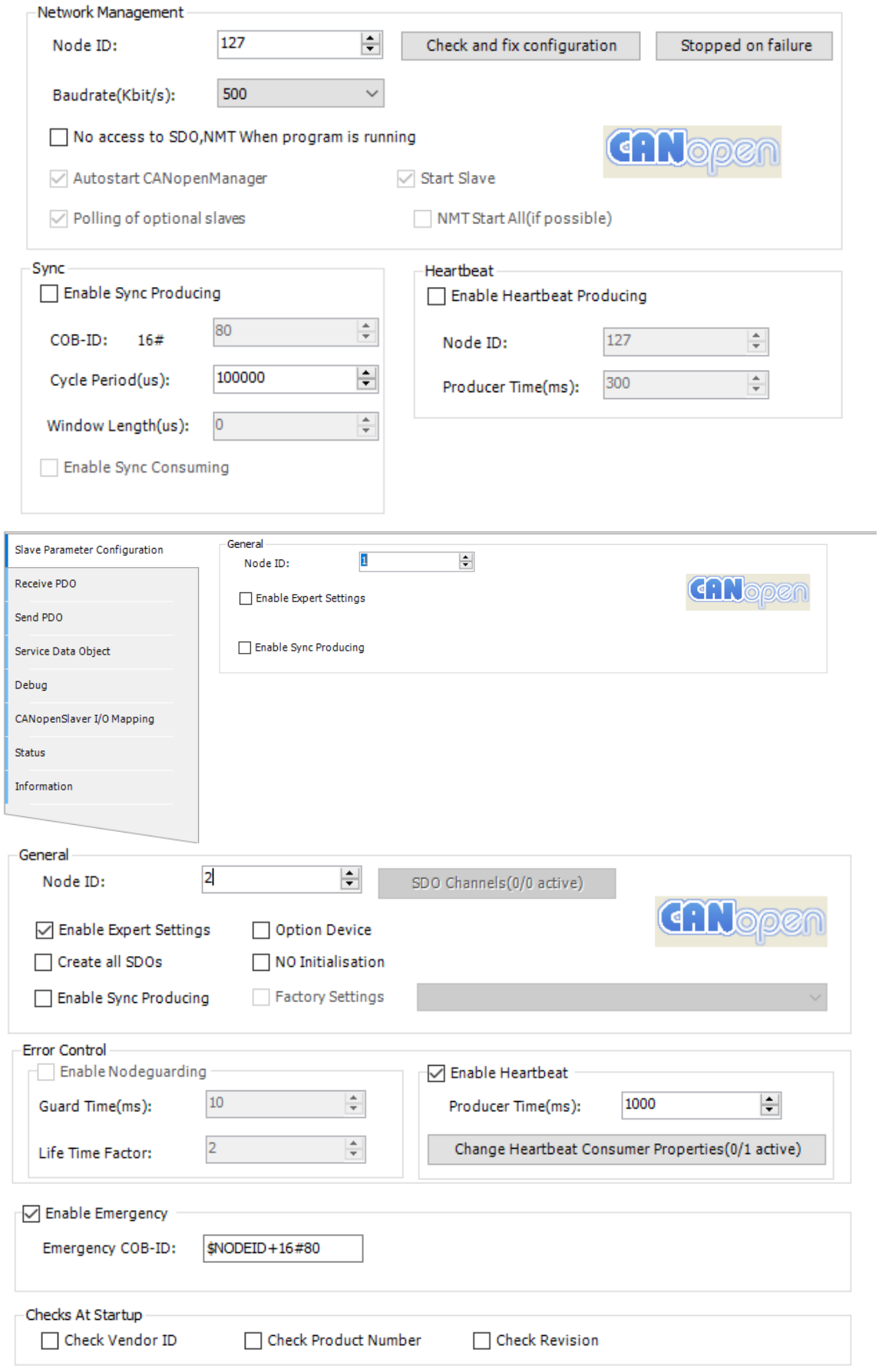


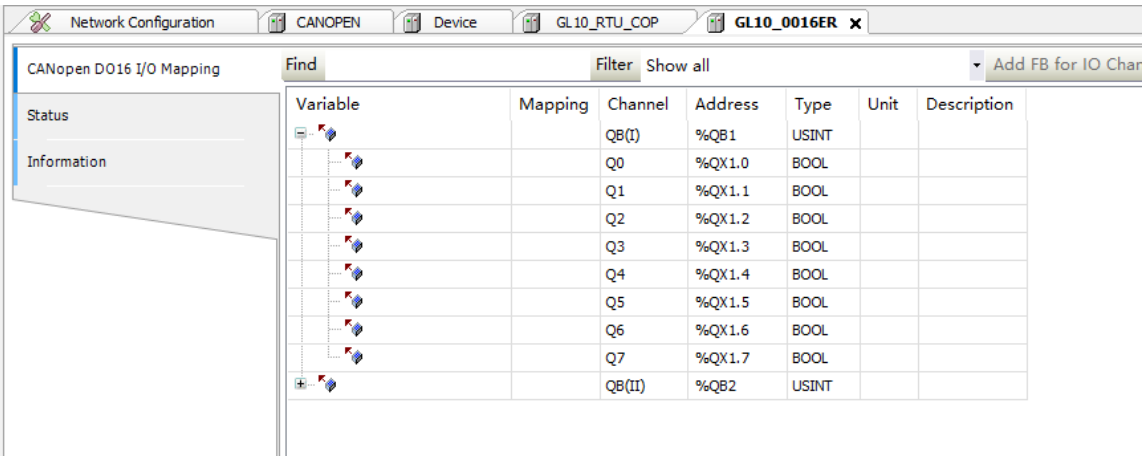
4. To add an AM600 slave, you need to add an I/O module on the "Hardware Configuration" page. Double-click the module in "In\Output Module List" on the right, as shown in the following figure. The CANopen slave is a CANopen remote device.



5. Set the master parameters, slave parameters, and module parameters properly. In normal cases, the slave node ID is automatically generated, PDOs and mapping are automatically generated based on the EDS file, and some special settings need to be modified manually.

When setting the parameters of the master and slave, ensure that the master baud rate and slave node ID match with the slave baud rate, slave node ID, and DIP switch, respectively.


Network Management
Node ID:
127
Check and fix configuration
Stopped on failure
Baudrate(Kbit/s):
500
No access to SDO,NMT When program is running
Autostart CANopenManager
Start Slave
Polling of optional slaves
NMT Start All(if possible)
Sync
Enable Sync Producing
COB-ID: 16#
80
Cycle Period(us):
100000
Window Length(us):
0
Enable Sync Consuming
Heartbeat
Enable Heartbeat Producing
Node ID:
127
Producer Time(ms):
300
Slave Parameter Configuration
Receive PDO
Send PDO
Service Data Object
Debug
CANopenSlaver I/O Mapping
Status
Information
General
Node ID:
1
Enable Expert Settings
Enable Sync Producing
General
Node ID:
2
SDO Channels(0/0 active)
Enable Expert Settings
Option Device
Create all SDOs
NO Initialisation
Enable Sync Producing
Factory Settings
Error Control
Enable Nodeguarding
Guard Time(ms):
10
Life Time Factor:
2
Enable Heartbeat
Producer Time(ms):
1000
Change Heartbeat Consumer Properties(0/1 active)
Enable Emergency
Emergency COB-ID:
$NODEID+16#80
Checks At Startup
Check Vendor ID
Check Product Number
Check Revision



The software provides soft elements to obtain the CANopen slave status and the CiA-DSP405 library for slave management and operation.

4.8.2 CANopen Master Configuration

Master configuration

If an AC800-series PLC serves as the master, the conversion module must be configured, as shown in the following figure.

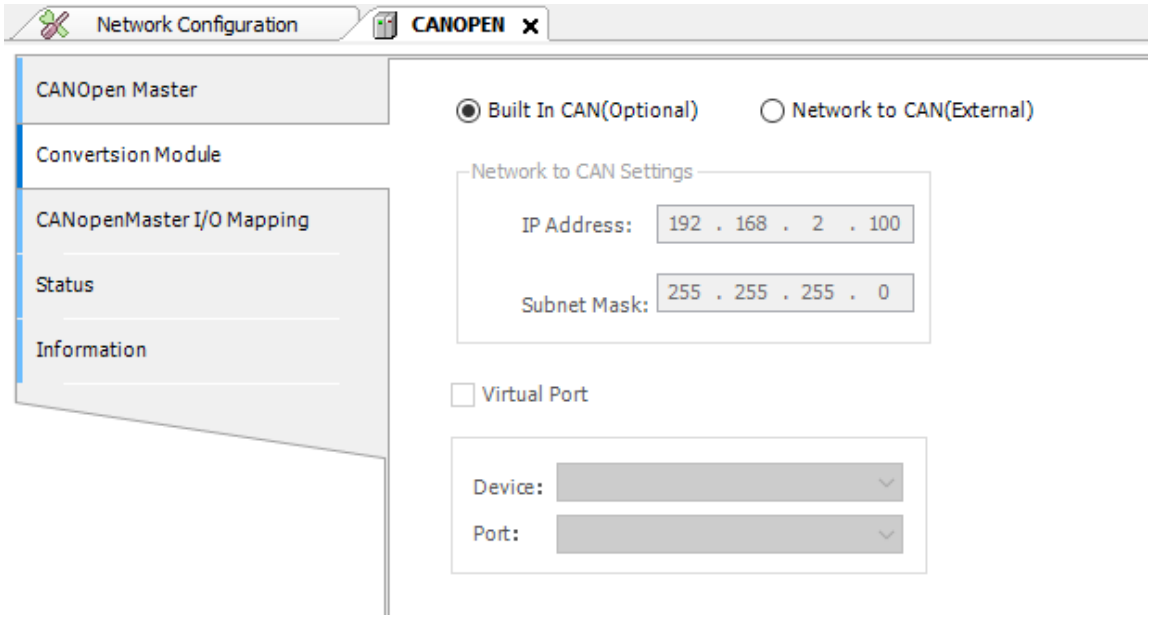
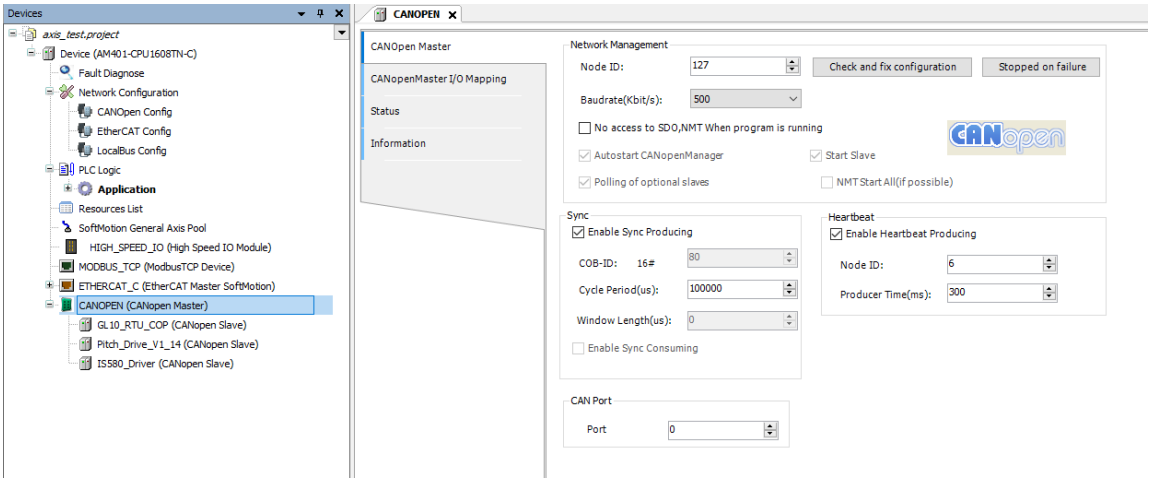


Figure 4-29 CANopen master configuration page

Two conversion modules are available: built-in CAN card and network-to-CAN module. If the built-in CAN card is used, select "Built In CAN". When the network-to-CAN module is used, configure the module type, IP address, and subnet mask. The network segment of the network-to-CAN module must be consistent with that of the port A or B of the AC800; otherwise, the network-to-CAN module cannot be scanned out by the AC800.

Note

After the IP address and subnet mask of the network-to-CAN module are modified, the modification takes effect and the network-to-CAN module is restarted and re-connected upon next login. Start the application program after the network-to-CAN module is restarted and the SYS indicator turns from red to green and blinks for 10s.



Network management

- Node ID: The unique identifier of the master in the CANopen network. The default value is 127, and the value range is 1 to 127, in the decimal format.
- Check and fix configuration: See the section "Check and fix configuration."
- Stopped on failure: See the section "Stopped on failure."
- Baudrate: The baud rate of transmission along the bus. The unit is kbps. Options are 10, 20, 50, 100, 125, 250, 500, 800, and 1000. The default value is 500.

Note

If the CPU module is at the head end or tail end of the network, turn the termination resistor of the CANopen port to ON. Set a proper baud rate because the communication distance is related to the baud rate.

- No access to SDO, NMT when program is running: If this option is selected, the slave cannot be accessed through SDO and NMT in the user program or on the slave commissioning page when the application program is running.
- Network load: The real-time load of the CANopen network when the bus is running. The network load is displayed only after you log in to the PLC.

Sync

- Enable Sync Producing: If this option is selected, the master sends synchronization information. It is deselected by default. This function can be enabled on only one station of the CANopen bus system. The PDO indicating the synchronization type sends information based on the preset type after synchronization information is sent.
- COB-ID: The communication object identifier, which is also the synchronization message ID. It is set to 16#80 and cannot be changed. This COB-ID is also used if "Enable Sync Producing" is selected on the slave.
- Cycle Period (μs): The interval at which synchronization information is sent. The value ranges from 2000 to 4294967000, in microseconds (μs). It is an integral multiple of the bus task time.

- Window Length (μs): Used for PDO synchronization. The unit is microseconds. It is invariably set to 0 and cannot be changed.

Heartbeat

Heartbeat is a node guarding mechanism. Different from node daemon, heartbeat can be triggered by the master or slave. In normal cases, the master sends heartbeat information to the slave, which is configured with the master node ID for consumption so that the slave can monitor the master.

- Enable Heartbeat Producing: If this option is selected, the master sends heartbeat information. It is deselected by default.
- Node ID: The unique identifier of the sent heartbeat information. By default, it is set to the master node ID. The value ranges from 1 to 127.
- Producer Time (ms): The interval at which heartbeat information is sent. The value ranges from 2 to 32767, in milliseconds (ms). It is an integral multiple of the bus task time.
- Window Length (μs): Used for PDO synchronization. The unit is microseconds. It is invariably set to 0 and cannot be changed.

If an AC800-series PLC serves as the master, the conversion module must be configured, as shown in the following figure.

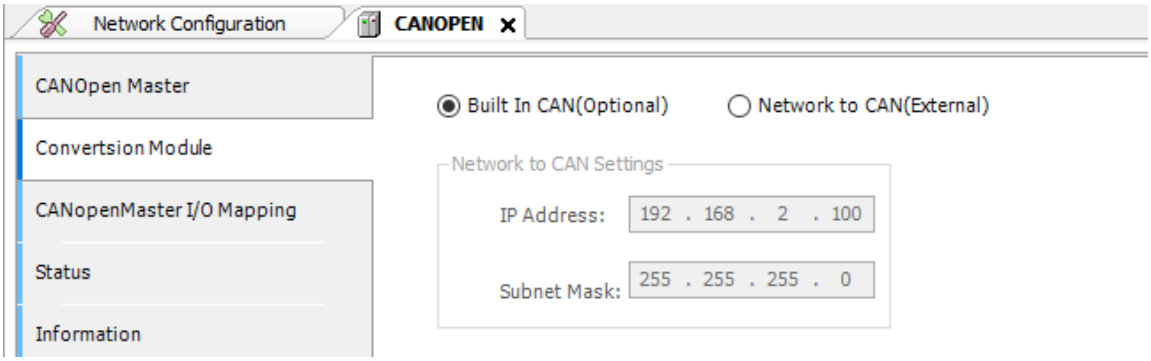


Figure 4-30 CANopen external module configuration page

Check and fix configuration

When multiple slaves are added to the CANopen system, the master or slave node IDs may be repeated or the COB-IDs may conflict with each other because the EDS files of different slaves may contain a configured COB-ID by default or the slave node ID is modified. In this case, click "Check and fix configuration" on the CANopen master configuration page to solve the problem of repeated node IDs or conflicting COB-IDs.

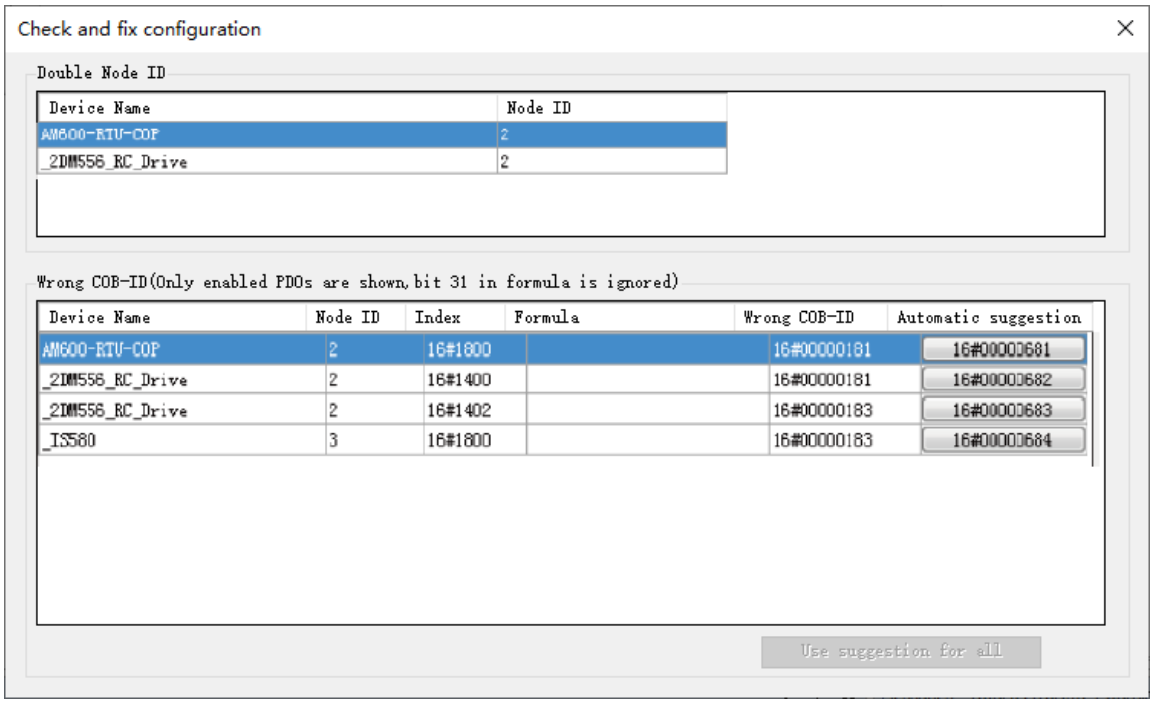


Figure 4-31 "Check and fix configuration" page

Double Node ID

The "Double Node ID" section lists the slaves with the same node ID. You can edit the Node ID column to re-allocate node IDs. Then, repeated node IDs are automatically canceled.

Wrong COB-ID

The "Wrong COB-ID" section lists the slaves with conflicting and invalid COB-IDs. You can modify COB-IDs in the following three ways:

- Edit items in the "Wrong COB-ID" column to manually modify the COB-ID corresponding to the current slave index.
- Click the button in the "Automatic suggestion" column and modify the COB-ID corresponding to the current slave index based on the displayed value.
- Click "Use suggestion for all" and modify all the incorrect COB-IDs based on the displayed values in the "Automatic suggestion" column.

After modification, slaves with correct COB-IDs disappear from the page.

Stopped On Failure

The "Stopped on FailureStopped On Failure" function determines whether to stop slave operation when a slave or module is faulty or the configuration is inconsistent. This function is only applicable to AM600 CANopen slaves.

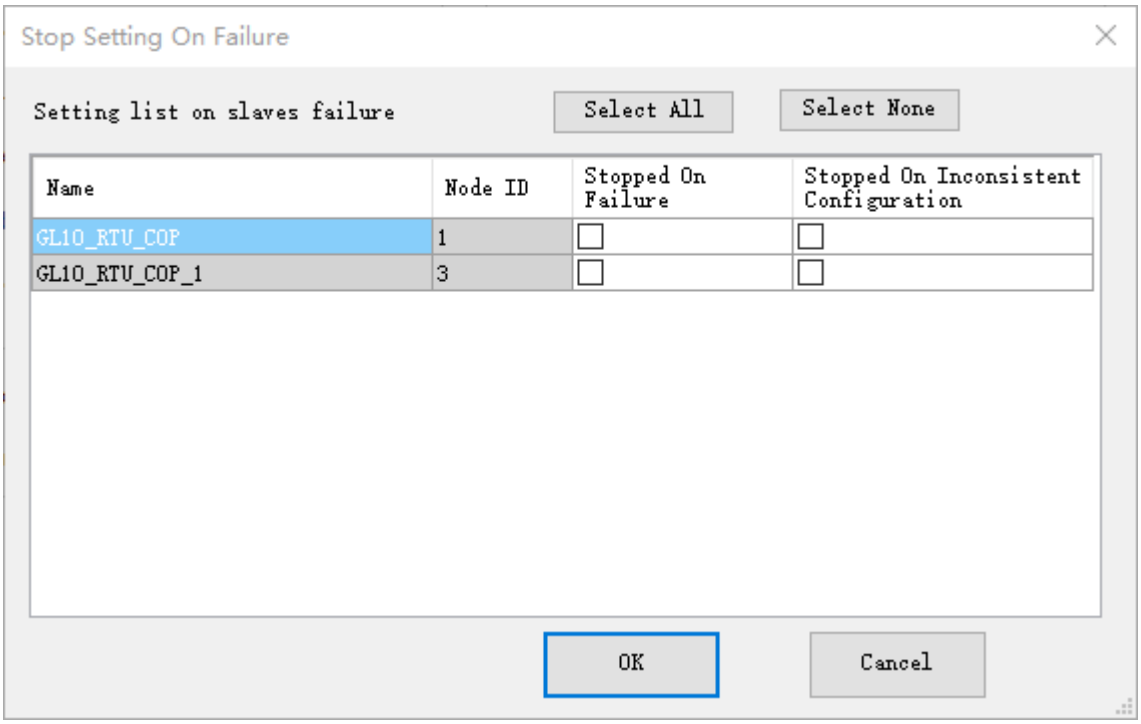


Figure 4-32 Stopped On Failure

- Setting list on slaves failure: You can view and set whether to stop operation upon slave failure or inconsistent configuration.
- In the "Stopped On Failure" column, you can set whether to stop slave operation when the specified slave or module is faulty. If the check box under "Stopped On Failure" is selected, the slave stops running when it is faulty or when the I/O module with the diagnosis and report function enabled is faulty.
- In the "Stopped on Inconsistent Configuration" column, you can set whether to stop slave operation when the I/O module of the slave has inconsistent configuration. If the check box is selected, the slave stops running when the I/O type does not match or the number of modules is more or less than the actual quantity.
- Click "Select All" or "Select None" to activate or deactivate all the slave settings in the section "Setting list on slaves failure".
- Click "OK" or "Cancel" to save or cancel the settings on the "Stopped On Failure" page.

CANopen Master I/O Mapping

For the general description of I/O mapping and instructions on this dialog box, see "I/O mapping".

State

The state configuration editor for the CANopen bus devices or modules displays state information (such as "Running" and "Stopped") and the state of the internal bus system.

Information

The following basic information about the currently available device is displayed: Name, Vendor, Categories, Version, Module Number, and Description.

4.8.3 CANopen Slave Configuration

Main items of CANopen slave configuration include the basic parameters, PDO settings, SDO settings, and commissioning.

Slave parameter setting

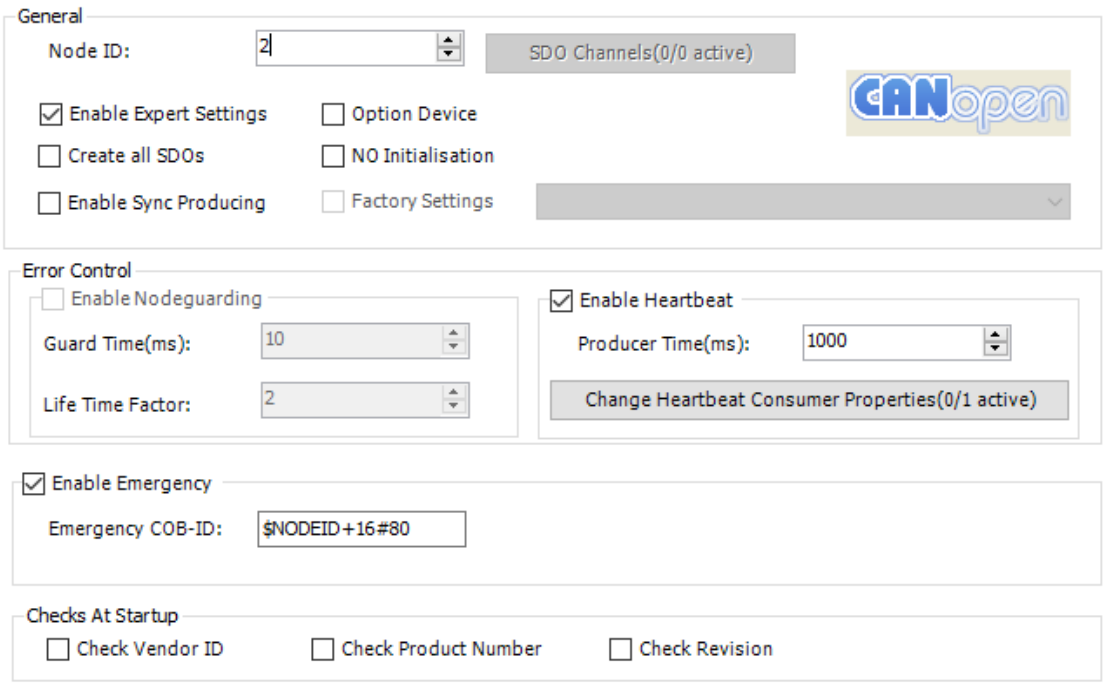


Figure 4-33 Slave parameter setting

General

- Node ID: The unique identifier of a slave in the CANopen network. The value ranges from 1 to 127, in the decimal format. The node ID must be consistent with the slave identifier (such as the DIP switch).
- SDO Channels: Not supported.
- Enable Expert Settings: If this option is selected, you can set expert parameters, such as slave node protection, heartbeat generation, emergency message, restart check, PDO mapping operation, system SDO display, and SDO abnormal jump.
- Option Device: Not supported.
- Create all SDOs: Select this option to create writable SDOs in the object dictionary. For example, the object access attributes are RW, WO, RWR, and RWW. The created SDOs are displayed on the "Service Data Object" page.
- NO Initialisation: Not supported.
- Enable Sync Producing: If this option is selected, the slave sends synchronization information. It is deselected by default. This function can be enabled on only one station of the CANopen bus system. Synchronous sending adopts the synchronization parameter settings of the master.
- Factory Settings: If this option is selected, the slave parameter settings are restored before you download configuration or configure the slave. The type of parameter restoration depends on the option selected from the restoration type list. The options are as follows:

1. sub:001: Restores all the parameters.
2. sub:002: Restores communication-related parameters (manufacturer-specified communication parameters indexed from 1000h to 1FFFh).
3. sub:003: Restores application-related parameters (manufacturer-specified application parameters indexed from 6000h to 9FFFh).
4. sub:004 to sub:127: Restores manufacturer-defined parameters.
5. sub:128 to sub:254: Reserved.

Options in the parameter restoration type list are based on the current object dictionary (EDS file) and come from the parsing results of EDS file index 1011. The sub-indexes have one-to-one correspondence with the preceding options.

Error Control

In the "Error Control" area, you can configure to monitor the node online status. The configuration items include node guarding and heartbeat.

The node guarding function enables the master to monitor the online status of the slave. The master sends slave daemon information periodically, and the slave is supposed to return a response to the master. If the slave fails to respond within the node daemon time (equal to the guard time multiplied by the life time factor), the master considers that the slave is lost.

Heartbeats can be produced by the master or slave. The producer broadcasts heartbeat messages to the CAN bus, and the heartbeat consumer consumes the heartbeat messages. If a node is configured with heartbeat consumption and no heartbeats corresponding to the node ID are detected within the configured time, the node is considered lost. Generally, the slave consumes the heartbeat messages of the master to monitor the online status of the master.

- Enable Nodeguarding: Select this check box to enable the node guarding function. Node guarding and heartbeat are mutually exclusive. The master sends a node guard matrix periodically within the guard time. If the slave fails to return a response containing a specific COB-ID (communication object identifier) within the node daemon time (which is equal to the guard time multiplied by the life time factor), the slave is considered offline.
- Guard Time: The interval at which the master sends node guard frames periodically. The value is an integral multiple of the bus task time and ranges from 10 to 65535, in ms.
- Life Time Factor: Used with "Guard Time". If the slave does not return a response within the node daemon time (equal to the guard time multiplied by the life time factor), the master considers that the slave is lost. The value ranges from 1 to 255.
- Enable Heartbeat: Select this check box to enable the heartbeat function on the slave so that the slave sends heartbeat frames periodically at an interval indicated by "Producer Time". Heartbeat and node guarding are mutually exclusive.
- Producer Time: The interval at which the slave sends heartbeat frames. The value is an integral multiple of the bus task time and ranges from 10 to 32767, in ms.
- Change Heartbeat Consumer Properties: Click this button to configure a heartbeat producer for the slave. You can configure heartbeat consumption for a slave so that the slave monitors the online status of the slave that produces heartbeats. Generally, the slave consumes the heartbeats of the master.

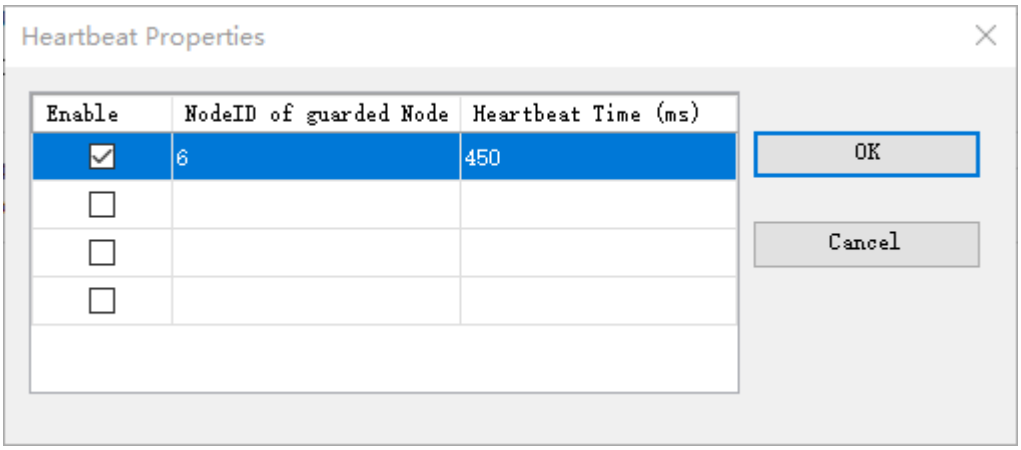


Figure 4-34 "Heartbeat Properties" page

The heartbeat consumption configuration list is used to configure the producer of the consumed heartbeat. You can configure a heartbeat producer after selecting the "Enable" check box.

By default, "NodeID of guarded Node" is set to the ID of the heartbeat producer node of the master. If the heartbeat producer function is not enabled on the master, this parameter is set to 0. The value ranges from 1 to 127. By default, "Heartbeat Time" is equal to the master heartbeat producer time multiplied by 1.5. The value ranges from 1 to 65535.

Emergency message

- Enable Emergency: Select this check box so that the slave sends emergency messages through the emergency message COB-ID. The emergency messages can be obtained through the functions provided by the CiA405 library (RECV_EMCY_DEF and RECV_EMCY) function library.
- Emergency COB-ID: COB-ID for the slave to send emergency messages. The default value is $NODEID+16#80, where NODEID is the node ID of the slave. The COB-ID is in the format of $NODEID +16#+hexadecimal number, 16#+hexadecimal number, or decimal number. (Example)

Checks At Startup

- Check Vendor ID: Select this check box to enable vendor ID checking. The slave checks whether the vendor ID (index: 1018; sub-index: 01) in the object dictionary matches with the vendor ID of the slave. The slave may not run properly if they do not match.
- Check Product Number: Select this check box to enable product number checking. The slave checks whether the product number (index: 1018; sub-index: 02) in the object dictionary matches with the product number of the slave. The slave may not run properly if they do not match.
- Check Revision: Select this check box to enable version checking. The slave checks whether the version (index: 1018; sub-index: 03) in the object dictionary matches with the version of the slave. The slave may not run properly if they do not match.

Receive PDO

PDOs are used for real-time data transmission between master and slave. "Receive PDO" is the real-time data that the master sends to the slave.

A PDO includes communication parameters and mapping parameters. The communication parameters include the unique communication identifier COB-ID, transmission type, and transmission control. The

PDO mapping parameters indicate the indexes and sub-indexes in the object dictionary from which the PDO transmitted data comes.

The receive PDO comes from the objects starting from index 1400 to index 1600 in the object dictionary. The default communication parameter values of each PDO come from the corresponding sub-indexes. The objects mapped to the receive PDO come from the writable objects in the object dictionary, such as the RW, RWW, and WO access permissions.

Note

- In non-expert mode, you can only change the values of the PDO communication parameters, but cannot add or delete PDOs and PDO mappings.
- AM600 slaves can only change the values of the PDO communication parameters, but cannot add or delete PDOs and PDO mappings. PDO mappings increase as AM600 I/Os are added.

The receive PDO is mapped to the AM600 output module. Each module corresponds to an invariable index, as shown in the following table.

Module Type	Index (Hexadecimal)	Sub-index (Hexadecimal)	Data Type
DO16	6300	01-10	Unsigned short int
DA4	6411	01-10	Short int

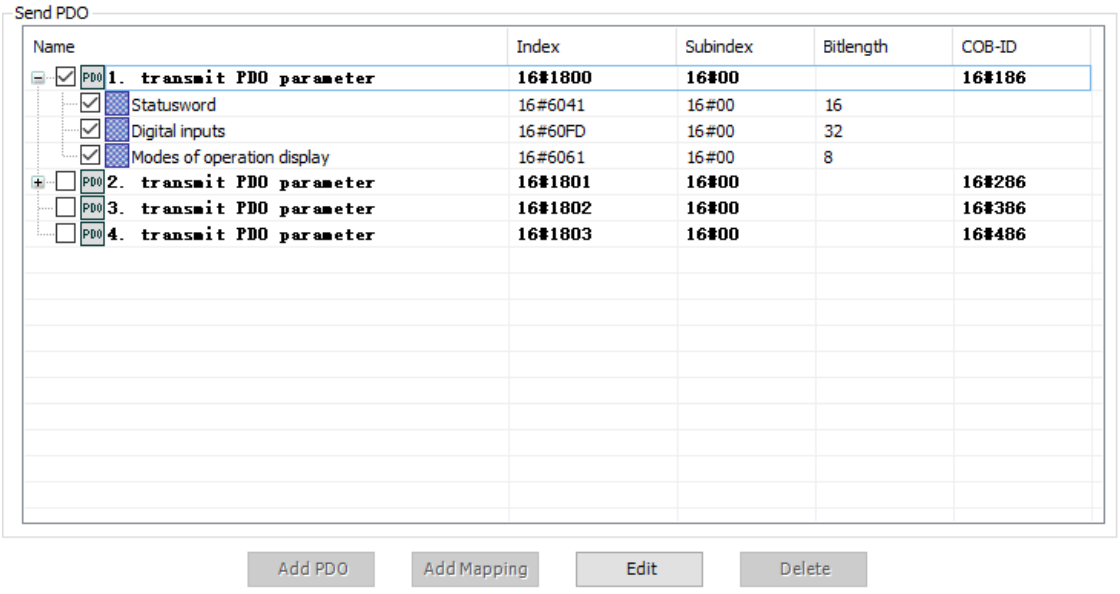


Figure 4-35 "Receive PDO" page

- Click "Add PDO" to add a PDO. The new PDO appears at the end of the PDO list. The maximum number of receive PDOs of the slave is determined by the number of indexes from 1400 to 1600 in the object dictionary. No more PDOs can be added when the maximum number is exceeded. The added PDO name and index are automatically obtained from the object dictionary in the usage sequence, and they cannot be modified.
  After you click "Add PDO", a dialog box is displayed, where you can set PDO attributes. For details, see "PDO attributes".
- Click "Add Mapping" to add a PDO mapping to the selected PDO. The new PDO mapping appears after the current PDO. A PDO mapping contains a maximum of 64 bits. If this limit is exceeded, the

PDO mapping cannot be added. PDO mappings come from the object dictionary. Receive PDOs are mapped to the writable objects in the object dictionary, such as the RW, RWW, and WO access permissions. For non-AM600 slaves, when you click "Add Mapping" to add a receive PDO mapping, the "Select Item From Object Dictionary" dialog box is displayed. For details, see "Adding an object".

- Click "Edit" to change the value of the selected PDO communication or mapping parameter. If a PDO is selected, you can change the values of PDO communication parameters. If a PDO mapping is selected, you can modify the PDO mapping. For AM600 slaves, you can only change the values of communication parameters.
- Click "Delete" to delete the selected PDO or PDO mapping. If a PDO is selected, this PDO is deleted. If a PDO mapping is selected, you can modify the PDO mapping. AM600 slaves do not support the delete operation.

Send PDO

PDOs are used for real-time data transmission between the master and slave. Send PDO is the real-time data that the slave sends to the master.

Send PDO comes from the objects starting from index 1800 to index 1A00 in the object dictionary. The default communication parameter values of each PDO come from the corresponding sub-indexes. The objects mapped to the send PDO come from the readable objects in the object dictionary, such as the RW, RWR, RO, and CONST access permissions.

Note

- In non-expert mode, you can only change the values of the PDO communication parameters, but cannot add or delete PDOs and PDO mappings.
- AM600 slaves can only change the values of the PDO communication parameters, but cannot add or delete PDOs and PDO mappings. PDO mappings increase as AM600 I/Os are added.

The send PDO is mapped to the AM600 input module. Each module corresponds to an invariable index, as shown in the following table.

Module Type	Index (Hexadecimal)	Sub-index (Hexadecimal)	Data Type
DI16	6100	01-10	Unsigned short int
AD4	6401	01-10	Short int

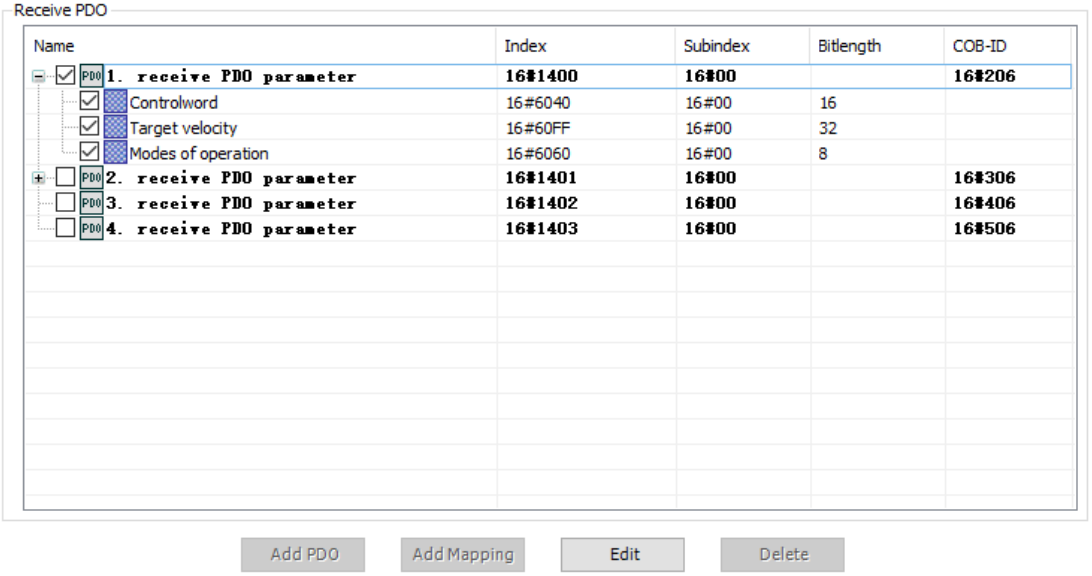


Figure 4-36 "Send PDO" page

- Click "Add PDO" to add a PDO. The new PDO appears at the end of the PDO list. The maximum number of send PDOs of the slave is determined by the number of indexes from 1800 to 1A00 in the object dictionary. No more PDOs can be added when the maximum number is exceeded. The added PDO name and index are automatically obtained from the object dictionary in the usage sequence, and they cannot be modified.
  After you click "Add PDO", a dialog box is displayed, where you can set PDO attributes. For details, see "PDO attributes".
- Click "Add Mapping" to add a PDO mapping to the selected PDO. The new PDO mapping appears after the current PDO. A PDO mapping contains a maximum of 64 bits. If this limit is exceeded, the PDO mapping cannot be added. PDO mappings come from the object dictionary. Send PDOs are mapped to the readable objects in the object dictionary, such as the RW, RWR, RO, and CONST access permissions.
  When you click "Add Mapping" to add a PDO mapping, the "Select Item From Object Dictionary" dialog box is displayed. For details, see "Adding an object."
- Click "Edit" to change the value of the selected PDO communication or mapping parameter. If a PDO is selected, you can change the values of PDO communication parameters. If a PDO mapping is selected, you can modify the PDO mapping. For AM600 slaves, you can only change the values of communication parameters.
- Click "Delete" to delete the selected PDO or PDO mapping. If a PDO is selected, this PDO is deleted. If a PDO mapping is selected, you can modify the PDO mapping. AM600 slaves do not support the delete operation.

Service Data Object

SDOs are used to transmit data during slave initialization and operation. Settings on the "Service Data Object" page are written to the slave during slave initialization.

On the "Service Data Object" page, you can configure the selected SDO, modify the SDO transmission sequence, and define an error handling method to be applied during SDO transmission.

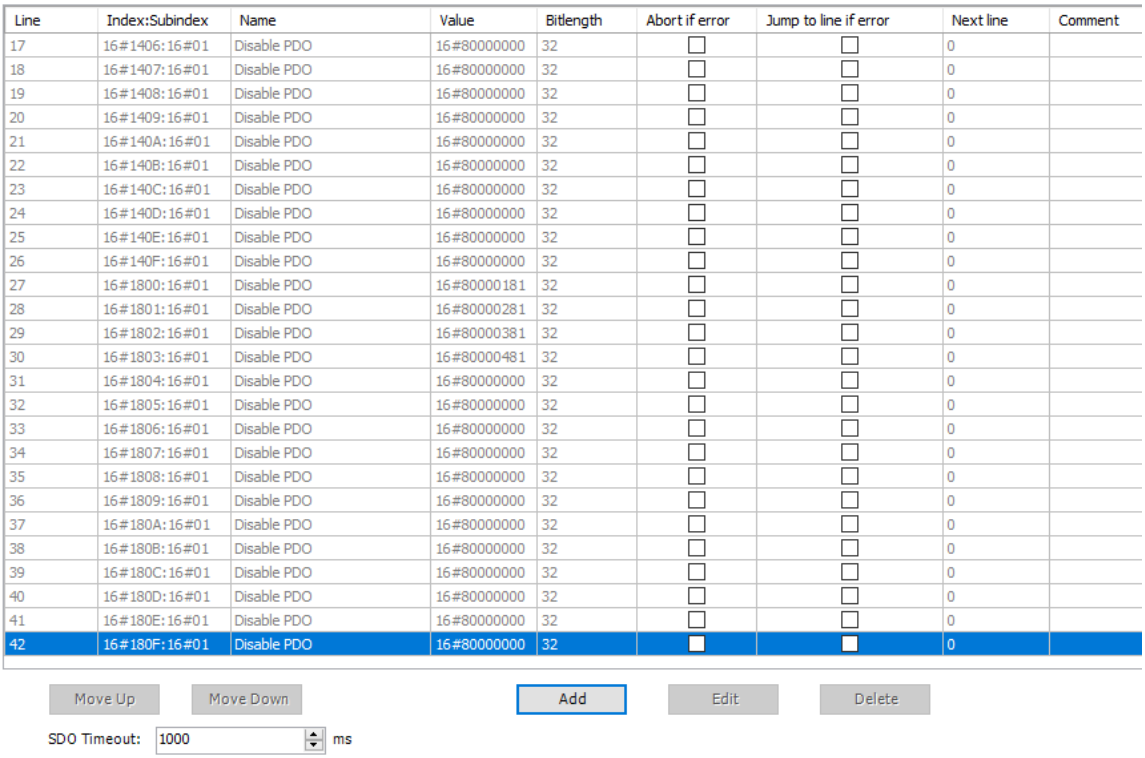


Figure 4-37 Service data object list page

The SDO list displays all the SDOs written to the slave during slave initialization. The SDOs in gray are automatically added and displayed on the top of the list. They are configured preferentially and automatically generated based on the parameter settings on the slave configuration page, such as heartbeat, node daemon, emergency message, PDO, and PDO mapping. You can click "Add" to add user-defined SDOs. User-defined SDOs can be modified and their positions in the list can be changed.

Double-click a value in the "Value" column of an SDO to change the value of the corresponding SDO.

You can define an error handling method to be applied during SDO configuration. If "Abort if error" is selected, SDO configuration is aborted in the case of an error, and the subsequent SDOs are not configured. If "Jump to line if error" is selected, the system jumps to the specified line and you can configure the subsequent SDOs. If "Abort if error" and "Jump to line if error" are not selected, the default error handling method applies, where the next SDO is configured.

Note

- SDOs and the error handling method are displayed only in expert mode.
- Be cautious when selecting "Jump to line if error". SDO configuration may encounter infinite loop if the system jumps to a previous line.

Item	Description
Move Up	Move the selected SDO to the previous line. Only user-defined SDOs can be moved to the previous line.
Move Down	Move the selected SDO to the next line. Only user-defined SDOs can be moved to the next line.

Item	Description
Add	Add an SDO before the selected SDO. When you click "Add", the "Add Object" dialog box is displayed, The "Add Object" page is similar to the "Add PDO" page, but the "Add SDO Object" page displays the SDO value text box and comment, allowing you to edit the SDO value and modify the SDO comment.
Edit	Modify the selected SDO. In the "Add Object" dialog box, modify the SDO information. Only user-defined SDOs can be modified.
Delete	Delete the selected SDO. Only user-defined SDOs can be deleted.
SDO Timeout	Set the timeout period of an SDO. The value ranges from 0 to 4294967, in ms. The default value is 1000 ms.

Commissioning

The commissioning page provides the functions of slave NMT control, SDO read and write, and diagnosis information acquisition.

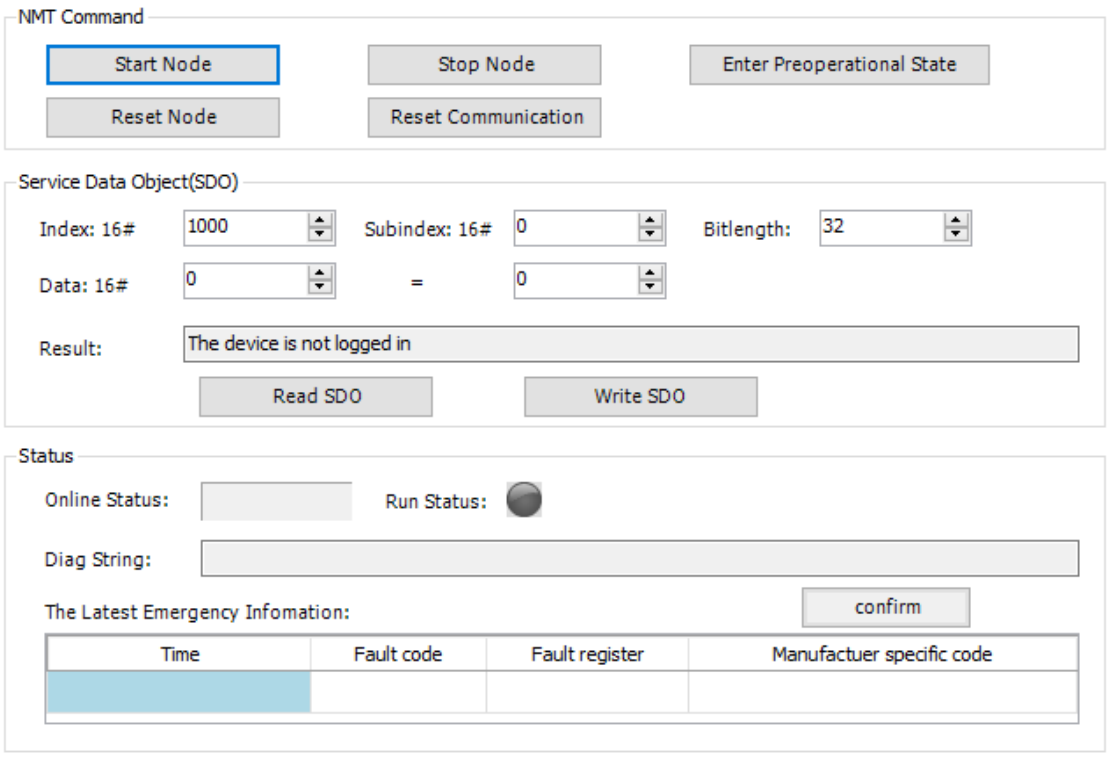


Figure 4-38 Commissioning page

NMT

NMT provides network management services, such as initialization, node start/stop, and failed node detection. These services adopt the master-slave communication mode, in which only one NMT master node or station exists.

The following figure shows the state transition of a slave during startup.

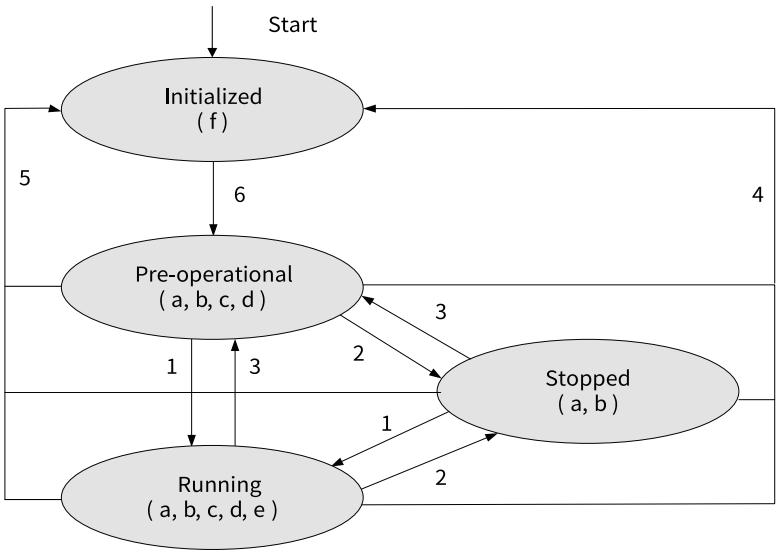


Figure 4-39 State transition of a slave during startup

Note:

a. NMT; b. Node Guard; c. SDO; d. Emergency; e. PDO; f. Boot-up

State transition (1 to 5 are initiated by NMT) sequence, NMT command words (enclosed by brackets):

1: Start_Remote_node (0x01, Start Node)

2: Stop_Remote_Node (0x02, Stop Node)

3: Enter_Pre-Operational_State (0x80, Enter Pre-operational State)

4: Reset_Node (0x81, Reset Node)

5: Reset_Communication (0x82, Reset Communication)

6. The device completes initialization, enters the Pre_Operational state, and sends a Boot-up message.

Initialization includes application data initialization and communication initialization. During node reset, all the data of slave nodes is reset. During communication reset, only communication data is reset.

NMT can enable all or some nodes to enter different states at any time.

- Start Node: Click this button to run slave nodes. PDO communication can be implemented only when slave nodes are running. When the slave is in the pre-operational or stopped state, clicking "Start Node" sets the slave to the running state (state 1 in *“Figure 4–39 State transition of a slave during startup” on page 258*).
- Stop Node: Click this button to stop slave nodes from running, and all communication except node daemon and heartbeat stops. When the slave is in the pre-operational or running state, clicking "Stop Node" sets the slave to the stopped state (state 2 in *“Figure 4–39 State transition of a slave during startup” on page 258*).
- Enter Preoperational State: Click this button to enable the slave to enter the pre-operational state, in which the slave can initiate SDO communication, but not PDO communication. The slave enters the pre-operational state after initialization. When the slave is in the running or stopped state, clicking "Enter Preoperational State" sets the slave to the pre-operational state (state 3 in *“Figure 4–39 State transition of a slave during startup” on page 258*).

- Reset Node: Click this button to reset the configuration of the slave. The application configuration and communication configuration are reset in sequence. The slave enters the pre-operational state (state 4 in *"Figure 4–39 State transition of a slave during startup" on page 258*).
- Reset Communication: Click this button to reset the communication configuration of the slave. Only the communication configuration is reset. The slave enters the pre-operational state (state 5 in *"Figure 4–39 State transition of a slave during startup" on page 258*).

Service data object (SDO)

SDOs are used to transmit a large volume of low-priority data between devices. A typical usage of SDOs is to configure devices in a CANopen network. On this page, you can read or write SDO object values when slave nodes are running. When reading or writing an SDO object, you need to determine the index, sub-index, and bit length of the SDO object. Also, specify the value to be written.

- Index: SDO read/write index, ranging from 16#0 to 16#FFFF.
- Sub-index: SDO read/write sub-index, ranging from 16#0 to 16#FF.
- Bit length: SDO read/write bit length. The optional values are 8, 16, 24, and 32.
- Data: SDO value read or written. Enter a hexadecimal value in the first text box, and enter a decimal value after the equal sign (=). The range of the written SDO value is related to the bit length. The minimum value is 0, and the maximum value is indicated by the bit length.
- Result: SDO read/write result. An error message is displayed when read/write is abnormal.

Status

This section displays the running status and emergency message about slave nodes.

- Online Status: Indicates whether the slave is online. If the slave is online, a green icon with the text "Online" is displayed. If the slave is offline, a red icon with the text "Offline" is displayed.
- Run Status: Indicates the running status of the slave with an icon, followed by descriptive text. The following table lists relevant icons and their descriptive texts.

State	Icon	Descriptive Text
Running		The slave is running.
Stopped		The slave is stopped.
Pre-operational		The slave is in the pre-operational state.
Initialized		The slave has been initialized.
Not connected		The slave is disconnected.
Connecting		The slave is being connected.
Preparing		The slave is in the preparation state.
Reset node		A node is being reset.
Reset communication		Communication is being reset.
Scan node		The slave is being scanned.
Configure node		The slave is being configured.

State	Icon	Descriptive Text
Start node		The slave is being started.
Unknown state		Unknown state

- Diag String: Displays the current diagnosis information of the slave.
- The Latest Emergency Information: Displays the first unconfirmed emergency message. Incoming emergency messages are not displayed until the current emergency message is confirmed.
  An emergency message contains 8 bytes and is in the following format:

sender receiver(s) COB-ID	Byte 0-1	Byte 2	Byte 3-7
0x080+Node_ID	Emergency error code	Error register (Object 0x1001)	Manufacturer specific code

The following table lists hexadecimal fault codes. The xx part is defined by the corresponding device profile.

Emergency Fault Code (Hexadecimal)	Function
00xx	Error Reset or No Error
10xx	Generic Error
20xx	Current
21xx	Current, device input side
22xx	Current, inside the device
23xx	Current, device output side
30xx	Voltage
31xx	Mains voltage
32xx	Voltage inside the device
33xx	Output voltage
40xx	Temperature
41xx	Ambient temperature
42xx	Device temperature
50xx	Device hardware
60xx	Device software
61xx	Internal software
62xx	User software
63xx	Data set
70xx	Additional modules
80xx	Monitoring
81xx	communication
8110	CAN overrun
8120	Error Passive
8130	Life Guard Error or Heartbeat Error
8140	Recovered from Bus-Off
82xx	Protocol Error
8210	PDO no processed Due to length error
8220	Length exceeded
90xx	External error
F0xx	Additional functions
FFxx	Device specific

The emergency message includes the time, error code, error register, and manufacturer specific code.

- Time: The time when the emergency message is obtained, rather than the time when a fault occurs.
- Error code: Hover the cursor over an error code to view the emergency message.
- Error register: The register that stores emergency message. See the following table.

Error Register Definition (Bit)	Error Type
0	Generic
1	Current
2	Voltage
3	Temperature
4	Communication
5	Device profile specific
6	Reserved(=0)
7	Manufacturer

- Manufacturer specific code: The code of the emergency message manufacturer.
- Confirm: Click this button to confirm the emergency message. Only one emergency message is retained. Incoming emergency messages are not displayed until the current emergency message is confirmed.

PDO attributes

PDO attributes are used to set PDO communication parameters, including COB-ID (communication object identifier), transmission type, inhibit time, and event time. The "Send PDO Properties" dialog box is displayed when you modify or add a PDO.

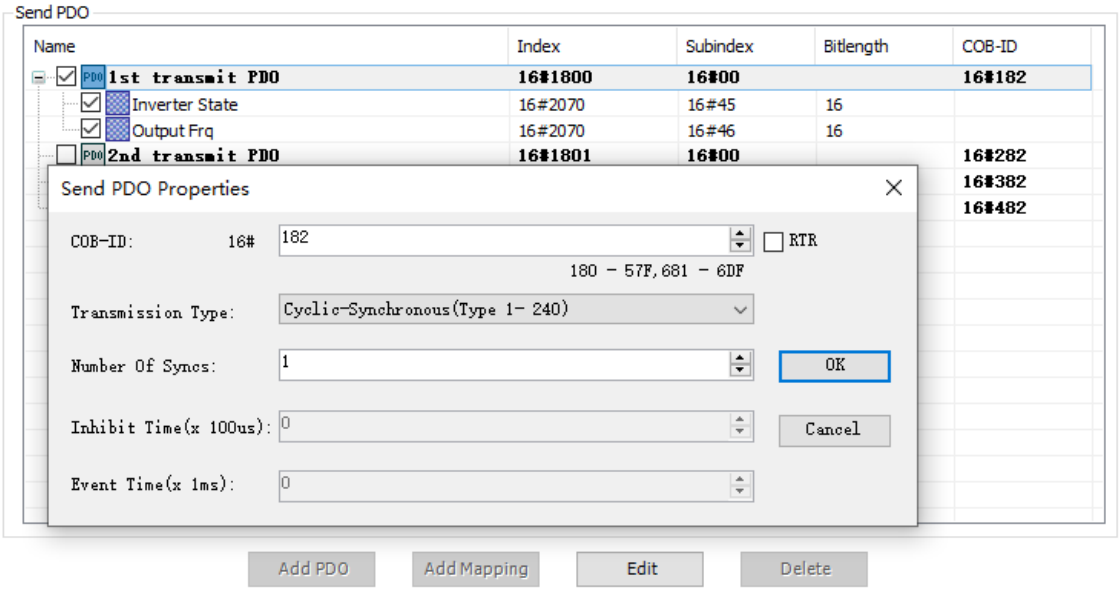


Figure 4-40 "Send PDO Properties" dialog box

- COB-ID: The communication object identifier of the PDO. Each COB-ID is unique within the CANopen bus and cannot be the same as other PDO COB-IDs, emergency COB-IDs, and heartbeat COB-IDs. The value range of PDO COB-ID is 16#180-57F and 681-6DF. If it is invalid, you can modify it

manually or by using the "check and fix configuration" function on the master. The default COB-ID of each PDO comes from the object of sub-index 01 of the corresponding PDO in the object dictionary. If the object dictionary format is $NodeID+value, the COB-ID changes with the slave node ID. After the COB-ID is changed manually, the original emergency code does not change with the new ID. The COB-ID cannot be changed if the object dictionary corresponding to the COB-ID does not have the write permission.

- RTR: Whether to enable remote frame reception. PDO sending is triggered when remote frames are received. Only sent PDOs are displayed.
- Transmission Type: The transmission mode to be applied during PDO communication.
  PDO supports the following transmission modes:

  1. Synchronous (synchronization is implemented by receiving SYNC objects)

     Aperiodic: Sending is triggered by remote frames or object-specific events specified in the device profile.

     Periodic: Sending is triggered after 1 to 240 SYNC messages.
  2. Asynchronous

     Sending is triggered by remote frames or object-specific events specified in the device profile.

     The following table lists the different PDO transmission modes defined by transmission types, which are part of the PDO communication parameter objects and defined by an eight-digit unsigned integer.

Transmission Type	PDO Communication Trigger Condition (B = both needed; O = one or both)			PDO Transmission
	SYNC	RTR	Event	
0	B	-	B	Synchronous and non-cyclic
1-240	O	-	-	Synchronous and cyclic
241-251	-	-	-	Reserved
252	B	B	-	Synchronous, after RTR
253	-	O	-	Asynchronous, after RTR
254	-	O	O	Asynchronous, manufacturer specific event
255	-	O	O	Asynchronous, specific event defined in the device profile
Note: SYNC: SYNC-object received. RTR: remote frame received. Event: Value changed or timer interrupted. Transmission type: A value ranging from 1 to 240 indicates the number of SYNC objects between two PDOs.				

     The default transmission type of each PDO comes from the object of sub-index 02 of the corresponding PDO in the object dictionary. If the object dictionary does not have the write permission, the transmission type cannot be changed.

- Number Of Syncs: This parameter is related to the transmission type and can be modified only when the value of "Transmission Type" is within the range from 1 to 240. The slave starts processing transmitted PDO data after receiving the number of synchronization frames specified by this parameter.

- Inhibit Time: The value of this parameter can be changed only when the value is equal to the product of the minimum interval at which the same PDO transmits two data records and 100 μs. This parameter prevents frequent PDO sending when a value is changed. The value ranges from 0 to 65535. The default value is 0. This parameter can be set only when PDOs are sent and "Transmission Type" is set to 254 or 255. The default value of "Inhibit Time" of each PDO comes from the object of sub-index 03 of the corresponding PDO in the object dictionary. The value of "Inhibit Time" cannot be changed if the object dictionary corresponding to "Inhibit Time" does not have the write permission.
- Event Time: The interval at which the same PDO transmits two data records. The value ranges from 0 to 65535, in ms. The default value is 0. This parameter can be set only when PDOs are sent and "Transmission Type" is set to 254 or 255. The default value of "Event Time" of each PDO comes from the object of sub-index 05 of the corresponding PDO in the object dictionary. The value of "Event Time" cannot be changed if the object dictionary corresponding to "Event Time" does not have the write permission.

Adding an Object

In the "Select Item From Object Dictionary" dialog box, you can add and modify receive PDO mappings, send SDO mappings, or SDOs. During SDO operation, this dialog box adds the "SDO Value" text box and the "SDO Comment" text box.

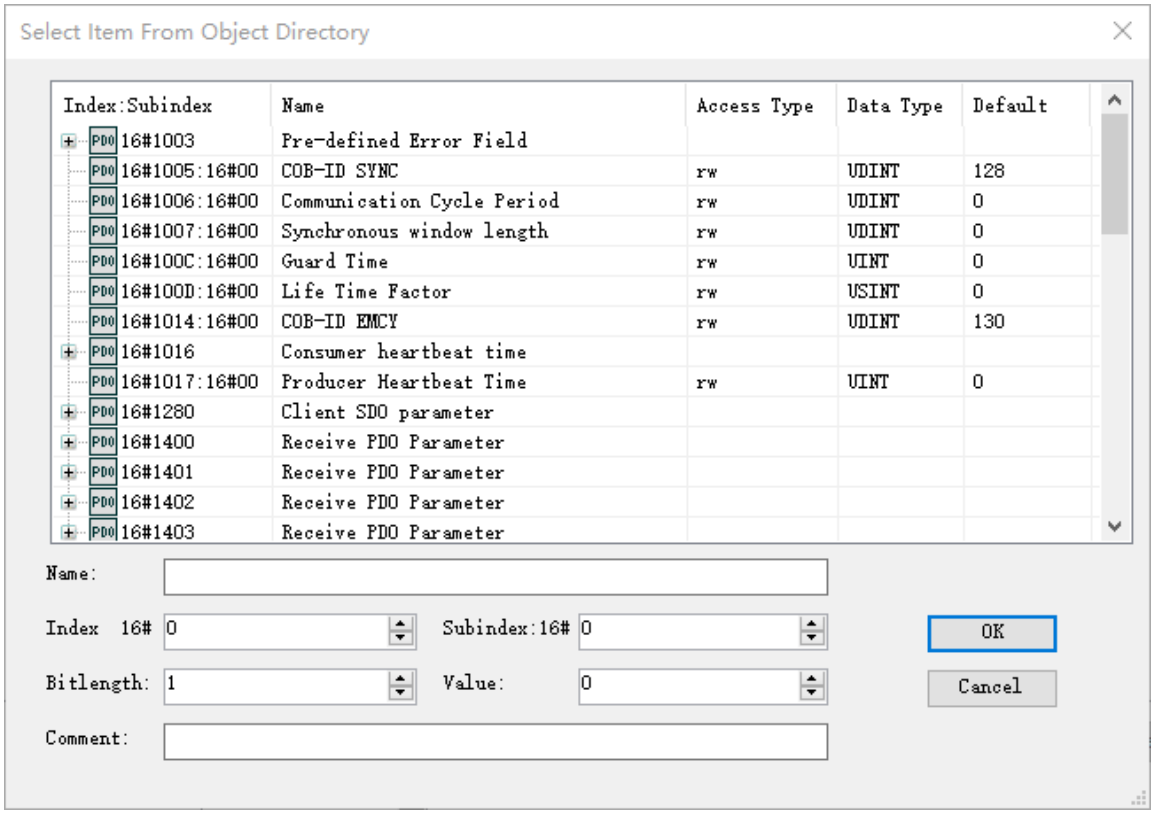


Figure 4-41 "Select Item From Object Dictionary" dialog box

The object list displays the objects in the EDS file. When receive PDO mappings are modified, only the objects with the RW, RWW, or WO access permission and with an index greater than 16#2000 are displayed. When send PDO mappings are modified, only the objects with the RW, RWR, RO, or CONST

access permission and with an index greater than 16#2000 are displayed. When SDOs are modified, only the objects with the RW, RWW, RWR, or WO access permission are displayed.

Note

When the SDOs of AM600 slaves are modified, the objects with an index within the range from 16#2000 to 16#40df cannot be displayed. These parameters are used by module configuration and set in the module.

- Index: The index of an object, ranging from 16#0 to 16#FFFF. After an object in the object list is selected, its index is displayed.
- Sub-index: The sub-index of an object, ranging from 16#0 to 16#FF. After an object in the object list is selected, its sub-index is displayed.
- Bit length: The bit length of an object, ranging from 0 to 32. After an object in the object list is selected, its bit length is displayed.
- Value: The SDO value. It is displayed only when SDOs are modified. Its value range is related to the data type of the selected object. After an object in the object list is selected, its value is displayed.
- Comment: The SDO comment. It is displayed only when SDOs are modified. The value contains a maximum of 50 characters.

CANopen slave I/O mapping

This page is displayed only when the slave is not of the AM600 series. The I/O mappings of AM600 slaves correspond to the AM600 I/O modules and are not displayed here. For the general description of I/O mapping and instructions on this page, see "I/O mapping".

State

The state configuration editor for the CAN bus or modules displays state information (such as "Running" and "Stopped") and the state of the internal bus system.

Information

The following basic information about the currently available device is displayed: Name, Vendor, Categories, Version, Module Number, Description, and Image.

4.8.4 CANopen Module

Modular device and non-modular device

In CANopen slave configuration, you can connect CANopen slave nodes to the following two types of modular devices:

Modular device: It is connected to a CANopen slave node and provides an I/O mapping list. The "CANopen Slave I/O Mapping" dialog box is not required. The PDO mappings of slave nodes increase as modules are added. Currently, the AM600 I/O module is a type of modular device.

Non-modular device: The slave node dialog box includes the I/O mapping dialog box. PDO mappings cannot be configured automatically.

AM600 CANopen I/O module

The AM600 CANopen I/O module is added in hardware configuration. A PDO mapping is automatically added when an I/O is added. For details about the relationship with PDO mappings, see "Receive PDO and Send PDO". After an I/O is added, you can set I/O parameters and add mappings to refresh data. For details, see "CPU-specific I/O module".

4.8.5 CANopen Parameter Configuration

Configuration is an important part of Inovance InoProShop programming software. Currently, InoProShop supports IS620P-CO models only. To use drives of other vendors, import EDS files of other vendors first. In addition, confirm that drive programs of other vendors are designed in strict compliance with CANopen communication flags and CiA402 standard. Before calling the CANopen402 function block, set parameters correctly. The details are as follows:

Master configuration

Major configuration items for the master include communication baud rate, synchronization mode, synchronization time, heartbeat, and heartbeat interval.

- Communication baud rate: The communication efficiency is improved as the baud rate increases. However, the communication distance decreases as the baud rate increases. Normally, the baud rate is set to 500 Kbits/s.
- Synchronization mode: The "Enable Sync Producing" function must be selected; otherwise, the function block cannot work.
- Synchronization cycle time: If only CANopen axis is available, it is recommended that you set the parameter to 4 ms, set the number of slaves to 3, and set the size of PDO configurations sent and received to less than 8 bytes. It is recommended that the cycle time be equal to the CANopen task scan cycle time.
- Heartbeat: The master sends heartbeat frames at intervals of the heartbeat time so that slaves can monitor whether the master is disconnected. The function must be used with slaves. Some slaves do not have the heartbeat check function, which is not required by default.
- Heartbeat time: The master sends heartbeat frames to slaves at intervals of the preset time. The heartbeat time is 300 ms by default and takes effect when "Enable Heartbeat" is selected.

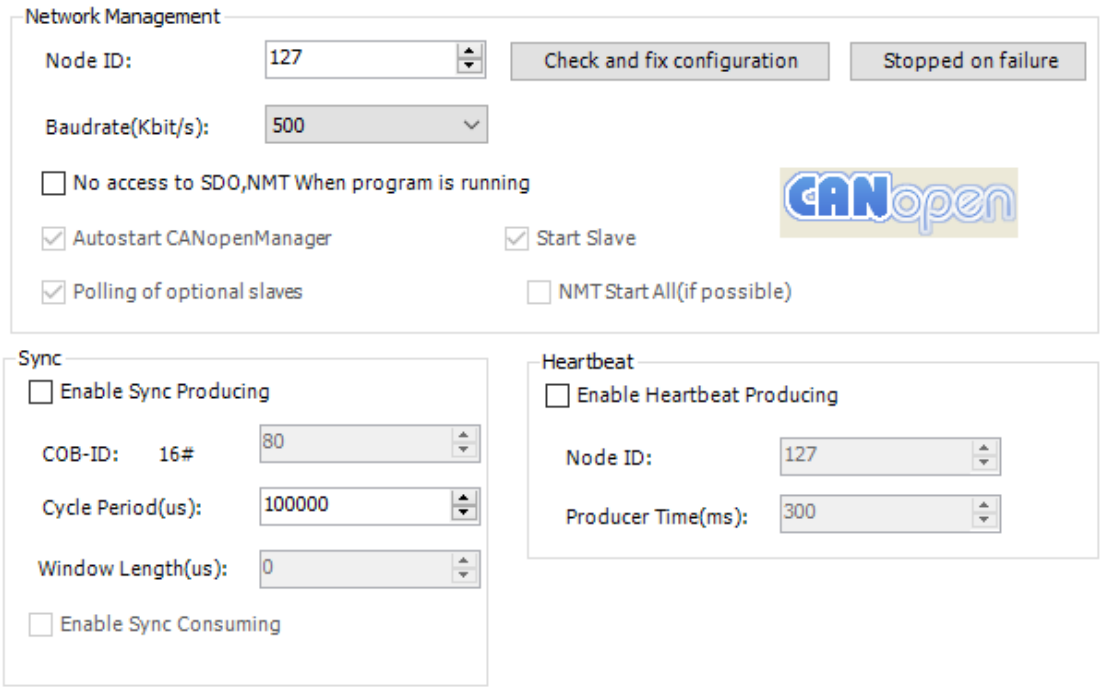


Figure 4-42 CANopen master configuration

Slave configuration

Major configuration items for a slave include node ID, heartbeat, heartbeat time, PDO synchronization mode, and synchronization object dictionary.

- Node ID: Also called the station number, it is a basic parameter for slave communication. It must be consistent with the physical station number.
- Heartbeat: The slave sends heartbeat frames to specified stations so that other stations can monitor its communication state. "Enable Heartbeat" is selected by default.
- Heartbeat time: The slave sends heartbeat frames to the master at intervals of the preset time. The heartbeat time is 1000 ms by default and takes effect when "Enable Heartbeat" is selected.
- PDO synchronization mode: The asynchronous mode is selected by default. You need to change it to the cyclic synchronization mode.
- PDO object dictionary configuration: The PDO object dictionary ensures that slaves exchange data with the master every bus cycle. The more PDOs are, the more efficient slaves exchange data with the master is. However, the more PDOs are, the more the bus load is. Excessive PDOs may delay bus data transmission and even cause disconnection. Select 6040, 6041, 6060, 6061, 607A, 6064 (6063), 60FF, 606C, and 6081 for sending and receiving PDOs for axis control devices. Options are 6083, 6084, 607C, and 6098.

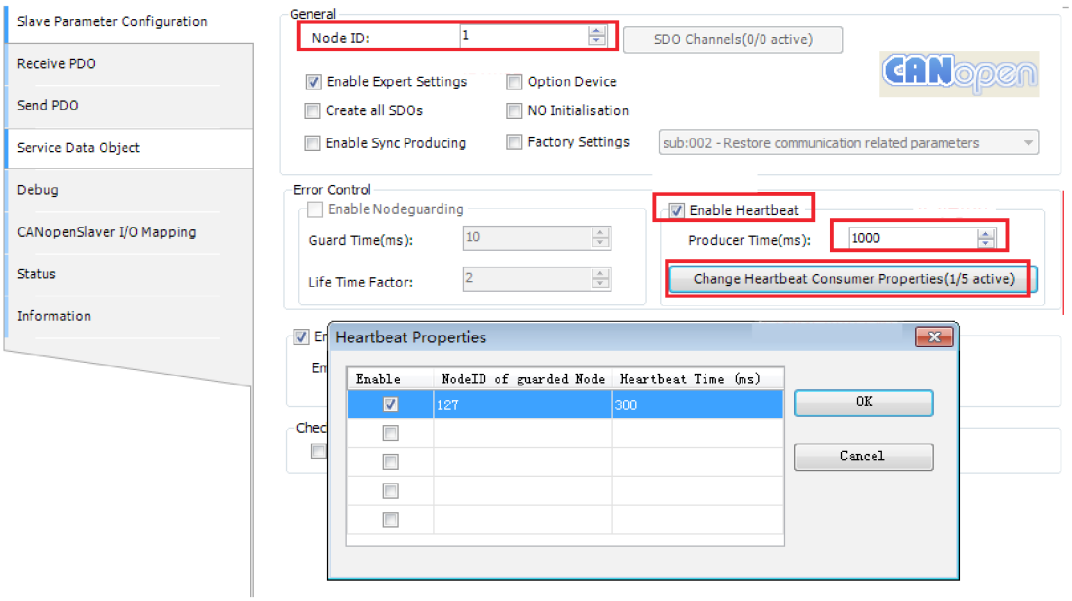


Figure 4-43 CANopen slave parameter configuration

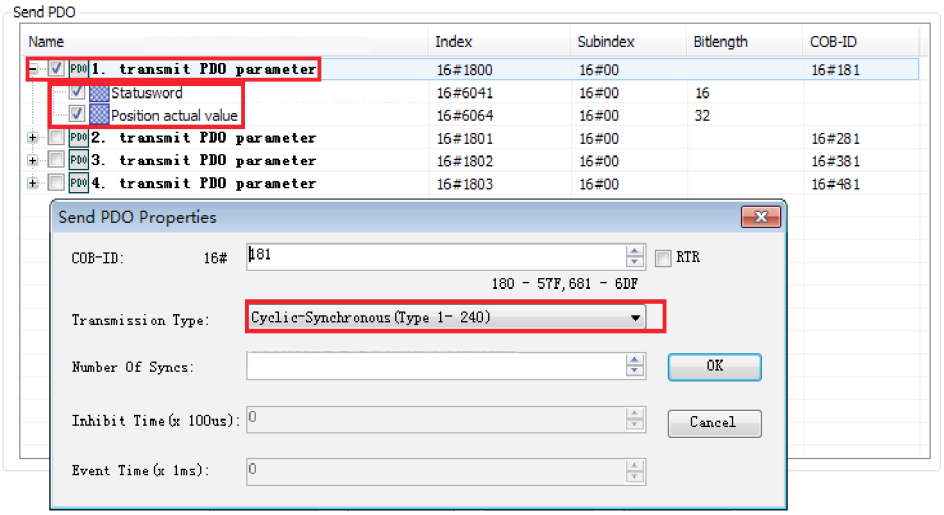


Figure 4-44 Configuration for CANopen slave sending PDOs

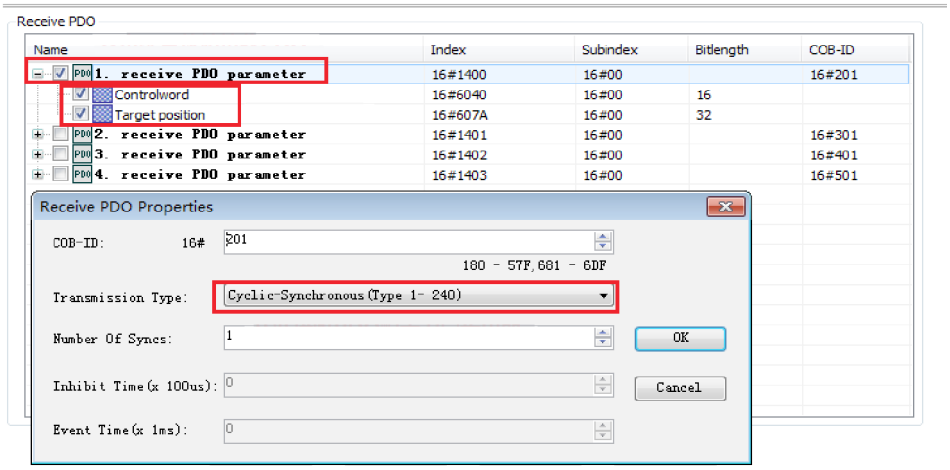


Figure 4-45 Configuration for CANopen slave receiving PDOs

CANopen axis configuration

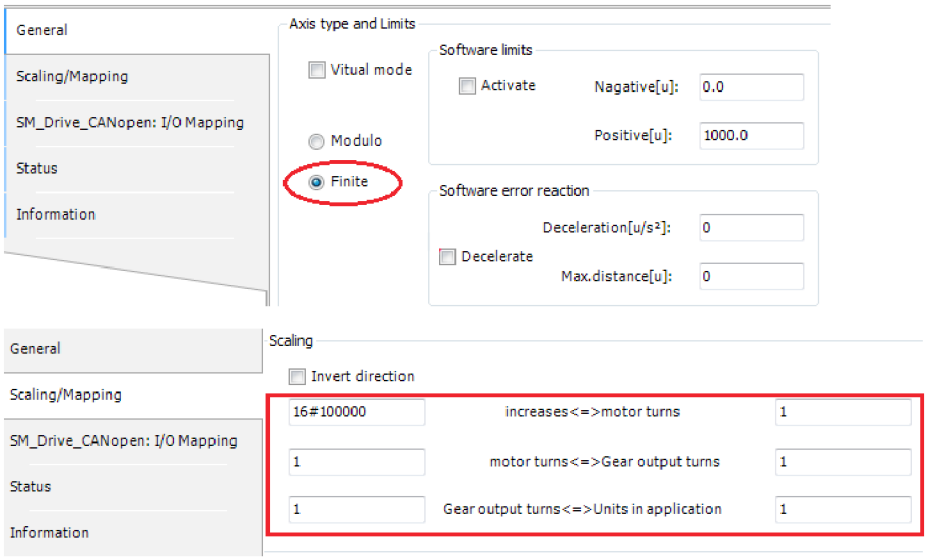


Figure 4-46 CANopen axis configuration

4.8.6 Programming Interface

For details, see section 6.3 "CANopen Communication Instructions" in the "Medium-Sized PLC Instruction Guide".

4.9 CANlink 3.0 Configuration Editor

4.9.1 Overview

The CANlink protocol is a real-time CAN bus application-layer protocol developed by Inovance based on the CAN 2.0 bus protocol. CANlink is mainly used for high-speed and real-time data exchange between Inovance products, such as PLCs, AC drives, servo controllers, and remote expansion modules. Read this section carefully before using the CANlink function of AM600-series PLCs.

CANlink 3.0 adopts the master/slave mode. A network must have a single master and may have 1 to 62 slaves. The master and slave numbers range from 1 to 63, and each number must be unique. It supports the following functions:

1. Master/slave running state monitoring through heartbeat
2. Bus usage warning and real-time bus usage monitoring
3. Reconnection upon disconnection
4. Hot access
5. 256 configuration records (including time trigger, event trigger, and synchronization trigger) sent by the master

6. 16 configuration records (including time trigger, event trigger, and synchronization trigger) sent by a single slave, amounting to 256 configuration records sent by all slaves
7. Point-to-multi-point data received by each station from other eight stations
8. Master-slave data exchange and slave-slave data exchange
9. Up to 128 data records written synchronously to the master

4.9.2 CANlink3_en.0 网络组成

Hardware port

For details about CANlink communication ports, see *" Hardware port" on page 242*.

Communication distance

A CANlink 3.0 network consists of 1 master and 1 to 62 slaves. The specific number of slaves is related to the baud rate.

Baud Rate	Maximum Communication Distance	Communication Cable Diameter	Number of Connected Stations
1000 kbps	20 m	⩾ 0.3 mm^2	18
500 kbps	80 m	⩾ 0.3 mm^2	32
250 kbps	150 m	⩾ 0.3 mm^2	63
125 kbps	300 m	⩾ 0.5 mm^2	63
100 kbps	500 m	⩾ 0.5 mm^2	63
50 kbps	1000 m	⩾ 0.7 mm^2	63

The preceding data is obtained under the premise of using standard shielded twisted pairs. The maximum number of connected stations (master and slaves) is determined based on the current baud rate.

Supported CANlink 3.0 devices

A CANlink 3.0 network must have a single master, which is a PLC of the AM400, AM600, or AC800 series. The network may have 1 to 62 slaves, including AM400, AM600, or AC800 (300 of D8280 indicates support), remote expansion modules (51210 or versions later than 52210), 214 non-standard IS500 servos (H00-02 = 214.xx), IS620P (H01-00 = 6.0 or greater), IS700 (H01-00 = 301.05), MD310 (F7-11 = u37.18 or greater), and MD380 (F7-11 = 4.71.06 or greater). Some products must be configured with a CANlink communication expansion card to use the CANlink function. For details, see the user guide of the specific product.

Special elements of CANlink 3.0 supported by AM600

Note

The CANlink function of AM600 uses the SD and SM soft elements, which are similar to the D and M elements of small-sized PLCs. However, they do not have a mapping relationship.

SD Range	Function	SM Range	Function
0-7000	User-specific word soft element area (available range in the CANlink configuration table)	0 to 3071	User-specific bit soft element area (available range in the CANlink configuration table)
7000-7999	User-specific word soft element area	3072-7999	User-specific bit soft element area
8000 to 8999	System-defined special register element area	8000 to 8999	System-defined special bit element area, which is currently only used by CANlink
9000 to 9999	System-defined special register element area, which is currently only used by high-speed I/O	9000 to 9999	System-defined special register element area, which is currently only used by high-speed I/O

The following table lists the special soft elements involved in the AM600 CANlink function (for details, see the CANlink 3.0 standard).

Special Soft Element	Attribute
SD8100-SD8163	SD8100 indicates the current state of the local node. SD81xx indicates the current state of the node with the station number xx. For example, SD8101 indicates the current state of the node with station number 1. Meanings of register values: 0: unconfigured; 1: configured; 2: online; 5: offline
SD8164-SD8239	Reserved
SD8240	Bus loading capacity (monitoring in the programming software)
SD8241	CAN 3.0 slave state
SD8242	CAN 3.0 slave state
SD8243	CAN 3.0 slave state
SD8244	CAN 3.0 slave state
SD8245	Number of received frames
SD8246	Receive error count
SD8247	Send error count
SD8287	Vendor code
SD8288	Product series
SD8289	Product model
SD8290	Firmware version
SD8307	CAN 3.0 error (command error code)
SD8308	CAN 3.0 error (configuration error code)

4.9.3 General Process of Using CANlink

The general process of using CANlink is as follows:

1. Design the CANlink hardware network structure.
2. On the "Network Configuration" page, activate the CANlink bus. The AM600 CPU can serve as the CANlink master or a CANlink local slave. Bus devices are automatically added after the bus is activated.
3. If the AM600 CPU serves as the CANlink master, in the CANlink network configuration wizard, you can set the master parameters and add or delete slaves. Slaves refer to remote slaves. If the AM600 CPU serves as a CANlink local slave, you can set other parameters.

4. Set the send parameters, receive parameters, and synchronization parameters properly.
5. Control CANlink transmitted data through soft elements in programs.
6. Control master and slave start/stop and monitor the master/slave running state on the "Network Management" page.

4.9.4 CANlink Network Configuration

Activate the CANlink bus in network configuration before configuring a CANlink network. After the CANlink master is activated, the CANlink master node is added to the device tree. When configuring the first CANlink network, double-click the node to display the network configuration wizard. After a CANlink slave is activated, the CANlink slave node is added to the device tree. Double-click the node to display the local slave configuration page.

CANlink 3.0 network configuration wizard

The network configuration wizard is displayed when you configure the CANlink bus for the first time or click "Site Management" on the "Network Management" page. Set the master and slave parameters in the network configuration wizard.

Master configuration

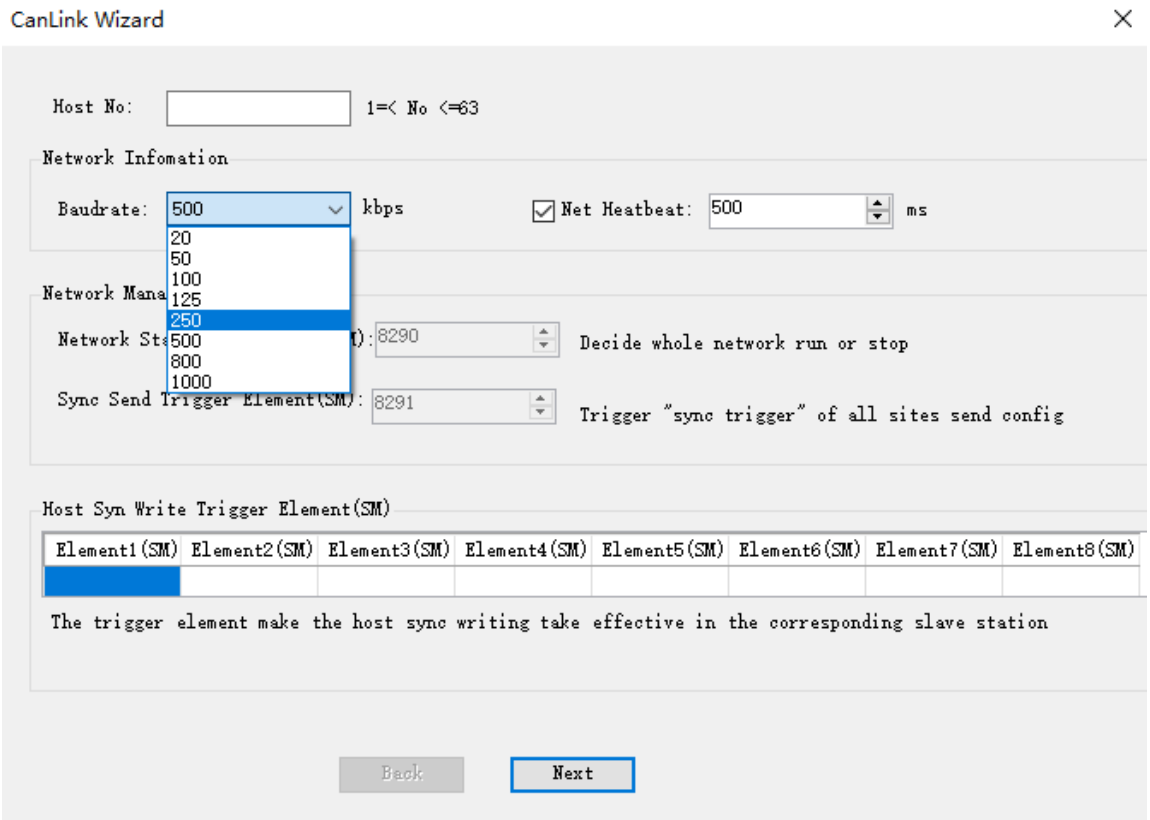


Figure 4-47 Master parameter setting page

- Host No.: Identifies a station as the master in the network. The master manages and monitors the entire CANlink 3.0 network. The master number must be consistent with the number of the configured PLC.

Network Information

- Baudrate: Select a baud rate to be used by the network. The baud rate of each station must be consistent with the selected one. If no baud rate is selected here, the baud rates of the connected stations must be consistent.
- Net Heartbeat: The value ranges from 10 to 20000, in ms. The master and slaves send heartbeats to the network periodically based on this value to monitor each other. When a communication error occurs, the master or slave triggers an alarm and handles the error. When "Net Heartbeat" is set to a smaller value, monitoring is more sensitive and the network usage of heartbeats is higher. An alarm is automatically triggered when the network usage of heartbeats exceeds 10%.

Note

If you deselect this parameter, the heartbeat function is disabled and the system cannot monitor the network.

Network Management

- Network Start/Stop Element: Control the CANlink network to start and stop by using the SM8290 soft element. When SM8290 is set to "TRUE", the CANlink network starts; otherwise, the CANlink network stops.
- Syn Send Trigger Element: Control the synchronous write function for the configuration sent by all stations by using the SM8291 soft element. For details, see *" Synchronous trigger configuration" on page 279*.

Host Syn Write Trigger Element

This element is used by the master to synchronize configuration. Up to eight synchronous write trigger elements can be configured. You can leave this parameter unspecified when the master does not need to synchronize configuration. For details, see *"4.9.8 Synchronous Write by the Master" on page 281*.

Adding and deleting a slave

This page allows you to add, modify, and delete CANlink slaves.

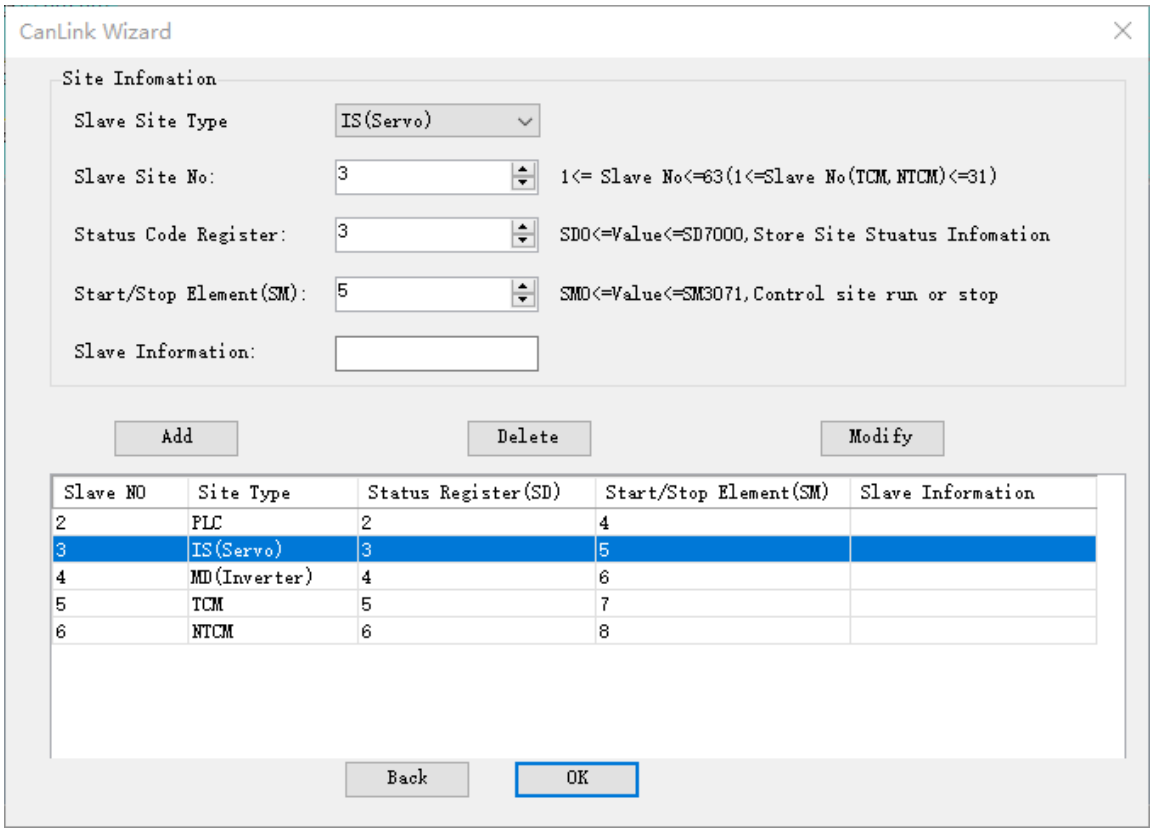


Figure 4-48 Page for adding and deleting a slave

Site Information

- Slave Site Type: Indicates supported type of the slaves, which can be the PLC, servo, AC drive, temperature control module, and non-temperature control module.
- Slave Site No.: Identifies a slave in the network. It must be different from the number of the master or any other slave.
- Status Code Register: Refers to the SD element that indicates the running state of the corresponding slave. These elements cannot be the same as other slave state register elements. The state may be "Running" or "Faulty" (excluding offline).
- Start/Stop Element: Indicates the SM element used by the master to start and stop slaves. These SM elements can be used to start and stop slave communication during network runtime. These elements cannot be the same as other slave start/stop elements and synchronous write trigger elements.
- Slave Information: Indicates the comment about a slave, containing up to 32 characters.
- Add: Click this button to add the slave with the configured information to the slave list. The system checks whether "Status Code Register" and "Start/Stop Element" are unique.
- Delete: Click this button to delete the selected slave from the slave list.
- Modify: Click this button to modify the settings of the selected slave. These settings are obtained from the site information. The system checks whether "Status Code Register" and "Start/Stop Element" are unique.

After the slave settings are complete, click "OK" to enter the "Network Management" page.

4.9.5 Network Management

On the "Network Management" page, you can start and stop the network, trigger synchronous sending, start and stop monitoring, start and stop slaves, enter the station configuration wizard, modify the network information of stations, and clear all the configurations.

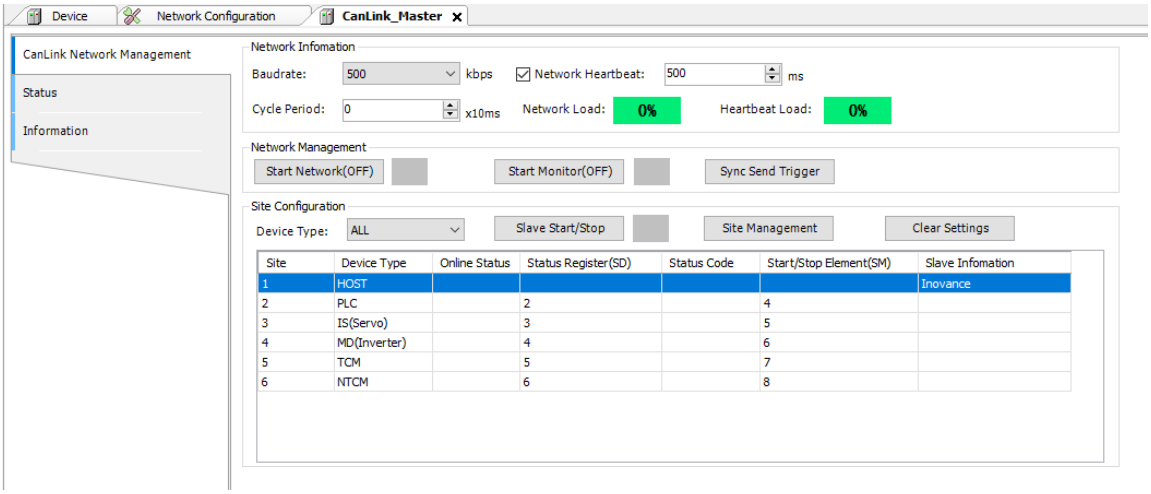


Figure 4-49 "Network Management" page

Network Information

- Baudrate: Select a baud rate to be used by the network. The baud rate of each station must be consistent with the selected one. If no baud rate is selected here, the baud rates of the connected stations must be consistent.
- Network Heartbeat: The value ranges from 10 to 20000, in ms. The master and slaves send heartbeats to the network periodically based on this value to monitor each other. When a communication error occurs, the master or slave triggers an alarm and handles the error. When "Network Heartbeat" is set to a smaller value, monitoring is more sensitive and the network usage of heartbeats is higher. An alarm is automatically triggered when the network usage of heartbeats exceeds 10%.

Note

If you deselect this parameter, the heartbeat function is disabled and the system cannot monitor the network.

- Cycle Period: Used to estimate the loading capacity of CANlink.
- Network Load: Indicates the CANlink network load, including CANlink receiving and sending, all heartbeat loads, real-time load of the CANlink bus obtained during monitoring (SD8240 register value), and estimated network load under non-monitoring conditions.
- Background color of "Network Load":

Color	Range (%)	Load Description
Green	0 to 50	Load percentage, such as 10%
Yellow	51 to 75	Load percentage, such as 55%
Red	76 to 90	Load percentage, such as 78%
Red	> 90	ERR

- Heartbeat Load: Indicates the heartbeat load of CANlink, which is calculated through estimation. An estimated value is obtained after login.
- Background color of "Heartbeat Load":

Color	Range (%)	Load Description
Green	0 to 10	Load percentage, such as 10%
Red	> 10	ERR

Network Management

The network management function is available only after login to the PLC.

- Start Network: Click this button to start and stop the CANlink network by using the SM8290 element.
- Start Monitor: Click this button to start CANlink network monitoring and obtain the CANlink master/slave running state periodically. The online state of stations is updated in the station list.
- Sync Send Trigger: Click this button to trigger a synchronous write operation.

Site Configuration

- Device Type: Select the device type to be displayed in the station list.
- Slave Start/Stop: Set the slave to the running state by setting the slave start/stop element to "TRUE".
- Site Management: Click this button to display the network configuration wizard dialog box, where you can configure a CANlink network.
- Clear Settings: Click this button to clear all the CANlink configurations and restore the master configuration to the default.
- Site list: Displays the information and state of CANlink stations. The following columns are displayed: "Site", "Device Type", "Online Status", "Status Register", "Status Code", "Start/Stop Element", and "Slave Information". Double-click a slave in the list to display the page for send configuration, receive configuration, or synchronous master write.
- Site: Indicates the unique identifier of a station.
- Device Type: Indicates the device type of the station, which can be the PLC, servo, AC drive, temperature control module, and non-temperature control module.
- Online Status: Click "Start Monitor" after login to the PLC to display the slave state. The online state of a slave is obtained through the SD8241, SD8242, SD8243, and SD8244 online status registers. If the slave is online, the system determines whether the slave is running or stopped by obtaining the value of the slave start/stop element. The online status register is read-only and cannot be modified. The following table lists the mapping relationships between each bit of the online status register and the station number.

Soft Element	Bit	Slave Number
SD8244	Bit 15 to bit 0	32 to 47
SD8243	Bit 15 to bit 0	48 to 63
SD8242	Bit 14 to bit 0	1 to 15
SD8241	Bit 15 to bit 0	16 to 31

- Status Register: Indicates the register that stores the slave running state. The running status register is read-only and cannot be modified.
  The following table lists each bit of the running status register.

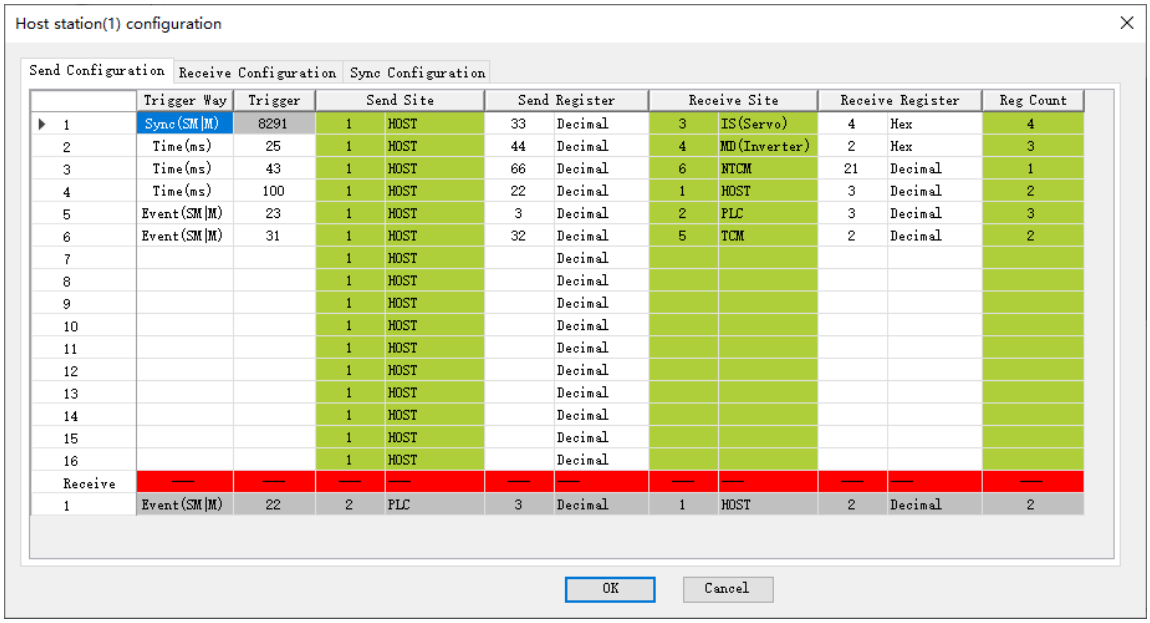
Bit	Description
Bit 0	Indicates the fault status. 1 indicates that the node is faulty, and 0 indicates that the node is normal.
Bit 1	Indicates the running status. 1 indicates that the node is running, and 0 indicates that the node is stopped.
Bit 2	Indicates whether the device is ready. 1 indicates that the device is ready, and 0 indicates that the device is not ready. This bit is only applicable to servos.
...	Reserved
Bit 15	Reserved

- Status Code: Indicates the value of the slave status register, which indicates the status code defined by the slave.
- Start/Stop Element: Starts or stops the element.
- Slave Information: Indicates the comment about the slave.

4.9.6 Send Configuration

CANlink send configuration is divided into time trigger configuration, event trigger configuration, and synchronous trigger configuration. Event trigger configuration is divided into PLC event trigger and non-PLC event trigger.

Send configuration editor



	Trigger Way	Trigger	Send Site		Send Register		Receive Site		Receive Register		Reg Count
1	Sync(SM|M)	8291	1	HOST	33	Decimal	3	IS(Servo)	4	Hex	4
2	Time(ms)	25	1	HOST	44	Decimal	4	MD(Inverter)	2	Hex	3
3	Time(ms)	43	1	HOST	66	Decimal	6	NTCM	21	Decimal	1
4	Time(ms)	100	1	HOST	22	Decimal	1	HOST	3	Decimal	2
5	Event(SM|M)	23	1	HOST	3	Decimal	2	PLC	3	Decimal	3
6	Event(SM|M)	31	1	HOST	32	Decimal	5	TCM	2	Decimal	2
7			1	HOST		Decimal					
8			1	HOST		Decimal					
9			1	HOST		Decimal					
10			1	HOST		Decimal					
11			1	HOST		Decimal					
12			1	HOST		Decimal					
13			1	HOST		Decimal					
14			1	HOST		Decimal					
15			1	HOST		Decimal					
16			1	HOST		Decimal					
Receive	—	—	—	—	—	—	—	—	—	—	—
1	Event(SM|M)	22	2	PLC	3	Decimal	1	HOST	2	Decimal	2

Figure 4-50 Send configuration editor

On the "Send Configuration" page, you can set the site send parameters. The send list is divided into two parts, of which the upper part displays the configuration sent by the local station to other stations, while the lower part displays the configuration sent by other stations to the local station.

For the master, up to 256 data records can be configured for sending; for slaves, up to 16 data records can be configured for sending.

The send list contains the following columns: "Trigger Way", "Trigger", "Send Site", "Send Register", "Receive Site", "Receive Register", and "Reg Count".

Item	Description
Trigger Way	Indicates the trigger type of data sending by stations. The options are time trigger, event trigger, and synchronous trigger.
Trigger	Indicates the trigger condition of data sending by stations. Configuration is sent when the trigger condition is met. When the trigger type is time trigger and the range is 0 to 30000, the station sends configuration periodically based on the trigger value. When the trigger type is event trigger and the trigger condition is "TRUE", the station sends configuration if it belongs to the host or PLC type, the trigger is an SM soft element, and the range is 0 to 3071. The station sends configuration periodically if it is a servo, AC drive, temperature control module, or non-temperature control module, and the range is 0 to 30000. Sending is executed when the trigger mode is synchronous trigger and soft element SM8291 is set to "TRUE".
Send Site	Indicates the station that sends configuration. It cannot be changed.
Send Register	Indicates the register corresponding to the send station. If the send station belongs to the PLC type, the range is 0 to 7000 (which indicates the register value plus the number of registers). If the send station is a temperature control module, the range is 0 to 499 or 700 to 722. If the send station is a non-temperature control module, the range is 0 to 63 or 700 to 722. If the send station is a servo or an AC drive, the range is 0 to 65535.
Receive Site	Indicates the station that receives configuration. It must be an existing station.
Receive Register	Indicates the register of the station that receives configuration. For point-to-multi-point communication (the send station and receive station are the same), the range is 0 to 65535. If the receive station is a PLC, the range is 0 to 7000. If the receive station is a temperature control module, the range is 0 to 499 or 721 to 722. If the receive station is a non-temperature control module, the range is 0 to 63 or 721 to 722. If the receive station is a servo or an AC drive, the range is 0 or 65535.
Reg Count	Indicates the number of send or receive registers. The value range is 1 to 4. The total length of "Reg Count" and "Send Register" or "Receive Register" cannot exceed the specified limit.

Time trigger configuration

Time trigger configuration is a common configuration. The send station writes the source address to the target address of the target station based on the configured time cycle. That is, the target station reads the source address of the send station cyclically and saves the source address to the target address.

If the send station and receive station are the same, it is a point-to-multi-point configuration. All the stations in the network configured to receive point-to-multi-point data from this station can receive such data. For details, see *"4.9.7 Receive Configuration" on page 281*.

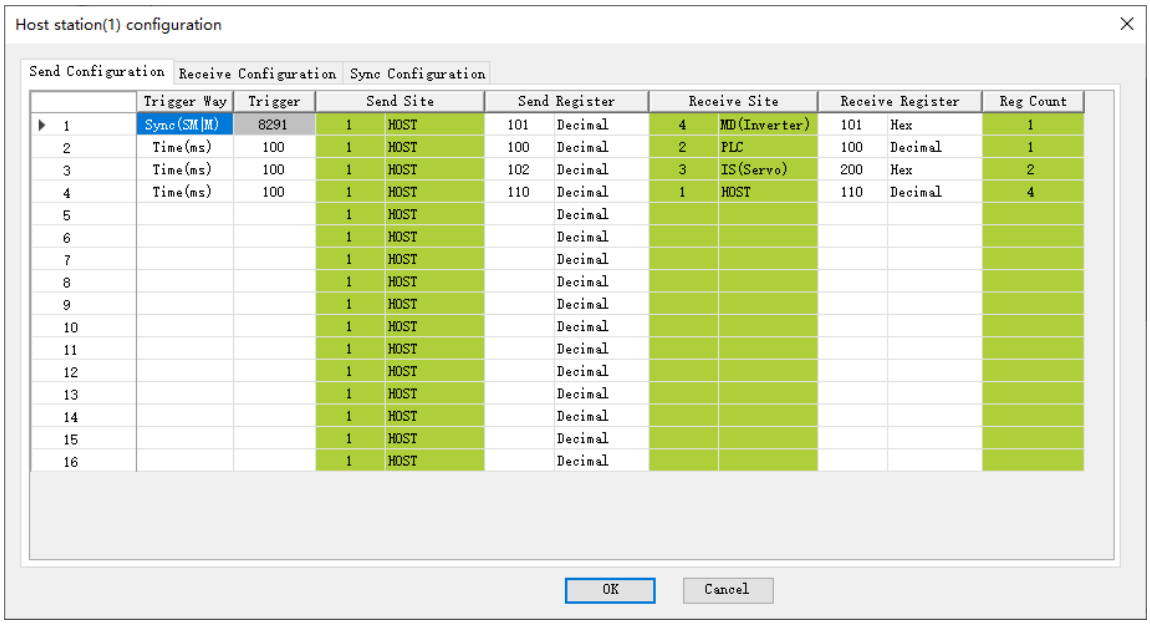


Figure 4-51 Time trigger configuration page

In case of time trigger, the cycle time ranges from 1 to 30000, in ms. The shorter the time, the higher the data update speed and the network load are.

In the 3rd row of the preceding figure, station 1 writes the values of registers SD102 and SD103 to registers H0200 and H0201 of station 3's servo every 100 ms. In the fourth row of the preceding, the send station and receive station are both station 1, so the configuration is a point-to-multi-point configuration. Station 1 sends the data of registers SD110 to SD113 to the network in point-to-multi-point mode every 100 ms. All the stations configured to receive point-to-multi-point data from station 1 can receive the frame.

Event trigger configuration

Event trigger is a real-time trigger configuration. Sending is triggered immediately when conditions are met. If conditions are not met, sending is not triggered, regardless of the interval from the last sending. The event trigger of PLCs is slightly different from that of other products based on product features.

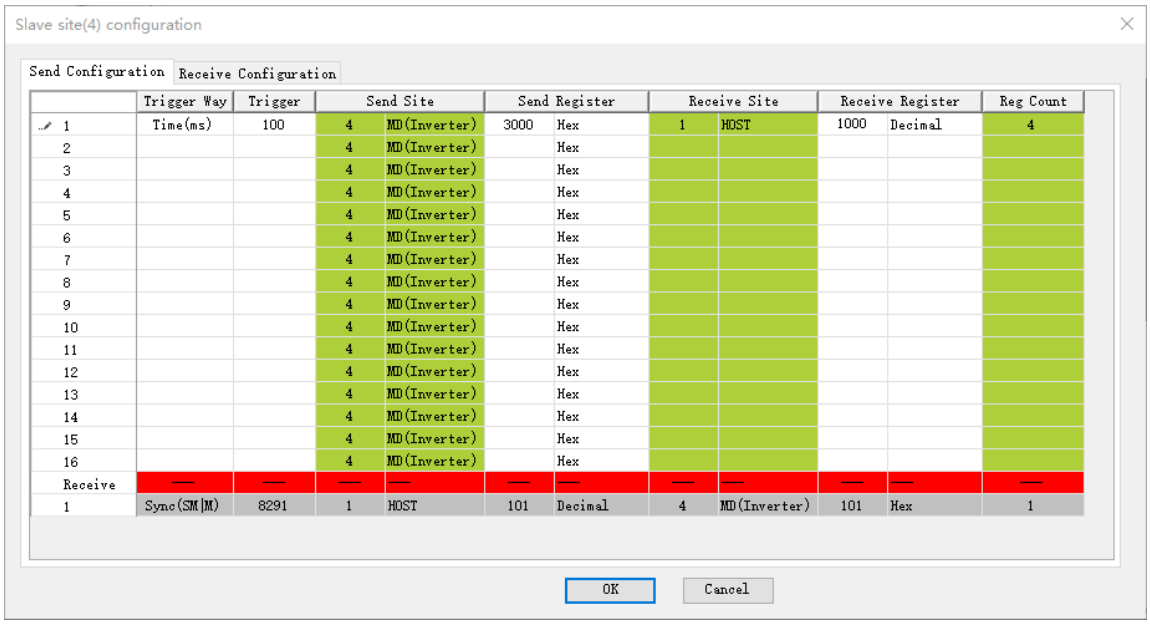
PLC event trigger configuration

The corresponding event is triggered when SM used as the trigger condition is set to "ON". The event trigger range is SM0 to 3071. A prompt is displayed when the range is exceeded.

In the preceding figure, when SM100 is set to 1, station 1 sends the value of SD100 to SD100 of the PLC of station 2. Then, SM100 is automatically reset.

Non-PLC event trigger configuration

Except PLCs, the event trigger of AC drives, servos, and expansion modules is based on register value change and the minimum interval from the last sending.



	Trigger Way	Trigger	Send Site		Send Register		Receive Site		Receive Register		Reg Count
1	Time(ms)	100	4	MD(Inverter)	3000	Hex	1	HOST	1000	Decimal	4
2			4	MD(Inverter)		Hex					
3			4	MD(Inverter)		Hex					
4			4	MD(Inverter)		Hex					
5			4	MD(Inverter)		Hex					
6			4	MD(Inverter)		Hex					
7			4	MD(Inverter)		Hex					
8			4	MD(Inverter)		Hex					
9			4	MD(Inverter)		Hex					
10			4	MD(Inverter)		Hex					
11			4	MD(Inverter)		Hex					
12			4	MD(Inverter)		Hex					
13			4	MD(Inverter)		Hex					
14			4	MD(Inverter)		Hex					
15			4	MD(Inverter)		Hex					
16			4	MD(Inverter)		Hex					
Receive	—	—	—	—	—	—	—	—	—	—	—
1	Sync(SM|M)	8291	1	HOST	101	Decimal	4	MD(Inverter)	101	Hex	1

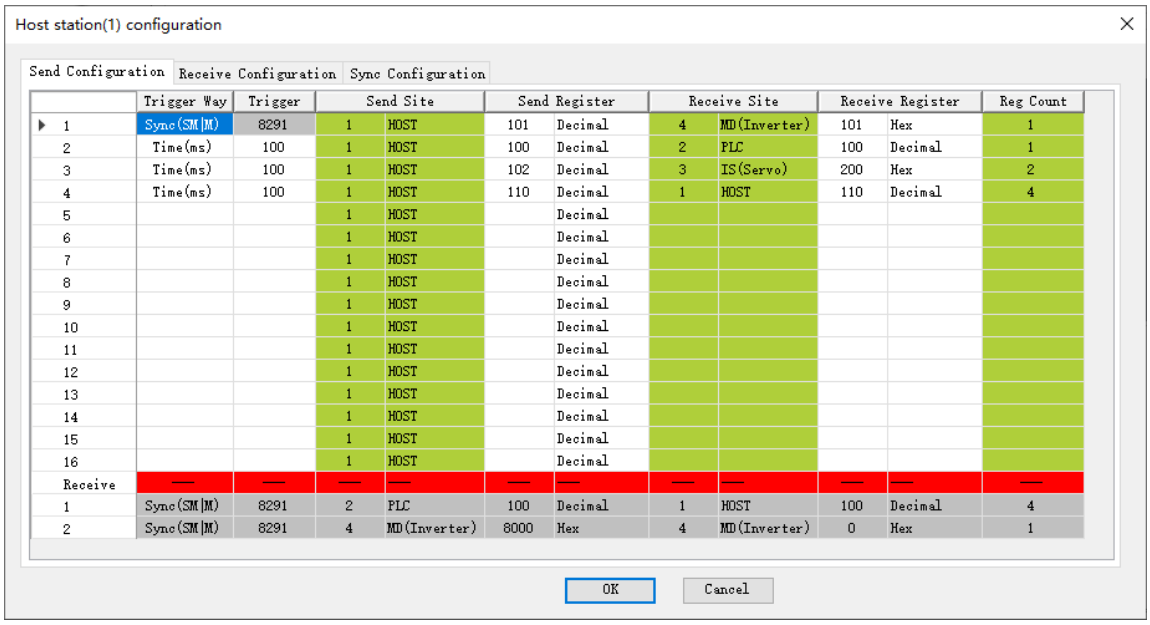
Figure 4-52 Non-PLC event trigger configuration

As shown in the preceding figure, when H3000 of the AC drive of station 4 is changed, sending is triggered immediately if the interval from the last sending of the configuration reaches 100 ms. If the interval from the last sending is less than 100 ms, sending is not triggered until the specified time is reached. A shorter interval results in better real-time effect but has greater impact on the network.

The minimum interval ranges from 1 to 30000, in ms. A prompt is displayed when the range is exceeded.

Synchronous trigger configuration

When the master configured with synchronous trigger detects that the synchronous send trigger element SM8291 is reset, the master broadcasts commands to the network to require all the stations in the network to trigger synchronous send in sequence. The trigger condition is configured by SM8291. The master resets SM8291 after sending a command frame. As shown in the following figure, master 1, PLC slave 2, and MD380 slave 4 are configured with synchronous trigger. If SM8291 of the master is reset, the three stations send synchronous trigger data.



	Trigger Way	Trigger	Send Site		Send Register		Receive Site		Receive Register		Reg Count
1	Sync(SM|M)	8291	1	HOST	101	Decimal	4	MD(Inverter)	101	Hex	1
2	Time(ms)	100	1	HOST	100	Decimal	2	PLC	100	Decimal	1
3	Time(ms)	100	1	HOST	102	Decimal	3	IS(Servo)	200	Hex	2
4	Time(ms)	100	1	HOST	110	Decimal	1	HOST	110	Decimal	4
5			1	HOST		Decimal					
6			1	HOST		Decimal					
7			1	HOST		Decimal					
8			1	HOST		Decimal					
9			1	HOST		Decimal					
10			1	HOST		Decimal					
11			1	HOST		Decimal					
12			1	HOST		Decimal					
13			1	HOST		Decimal					
14			1	HOST		Decimal					
15			1	HOST		Decimal					
16			1	HOST		Decimal					
Receive	—	—	—	—	—	—	—	—	—	—	—
1	Sync(SM|M)	8291	2	PLC	100	Decimal	1	HOST	100	Decimal	4
2	Sync(SM|M)	8291	4	MD(Inverter)	8000	Hex	4	MD(Inverter)	0	Hex	1

Figure 4-53 Synchronous trigger configuration

SM8291 is used in the same way as the SM element of PLC event trigger. You can perform the corresponding operation by clicking "Sync Send Trigger" on the CANlink 3.0 main screen.

Only SM8291 of the master can operate SM8291 of the slave PLC without impact.

In essence, synchronous trigger is a type of event trigger in which SM8291 of the master is used as an event by all the stations in the network.

Differences among time trigger, event trigger, and synchronous trigger

Item	Time Trigger	Event Trigger	Synchronous Trigger
Sending mode	Scheduled and cyclic sending.	Sending is triggered in the case of an event or data change. The event is automatically cleared after sending.	Sending is enabled by the master. The event is automatically cleared after sending.
Real-time effect	The real-time effect is related to the configured time. The shorter the time, the better the real-time effect is.	Sending is triggered in the case of an event, with good real-time effect.	Sending is triggered when M8291 of the master is reset. The real-time effect is better than that of event trigger.
Execution times	Scheduled sending.	Triggered once in the case of an event.	Triggered once in the case of an event.
Programming in PLC	Scheduled sending, without program design.	Sequential logic design in program.	Sequential logic design in program.
Network usage	High	Low	Low
Applicable scenario	There are no special time sequence requirements. Data can be written continuously.	There are time sequence requirements. Data is written once, or continuous writing may cause a fault.	The master requires data returned by multiple slaves in special scenarios.

4.9.7 Receive Configuration

On the "Receive Configuration" page, you can configure stations to receive point-to-multi-point data from the network.

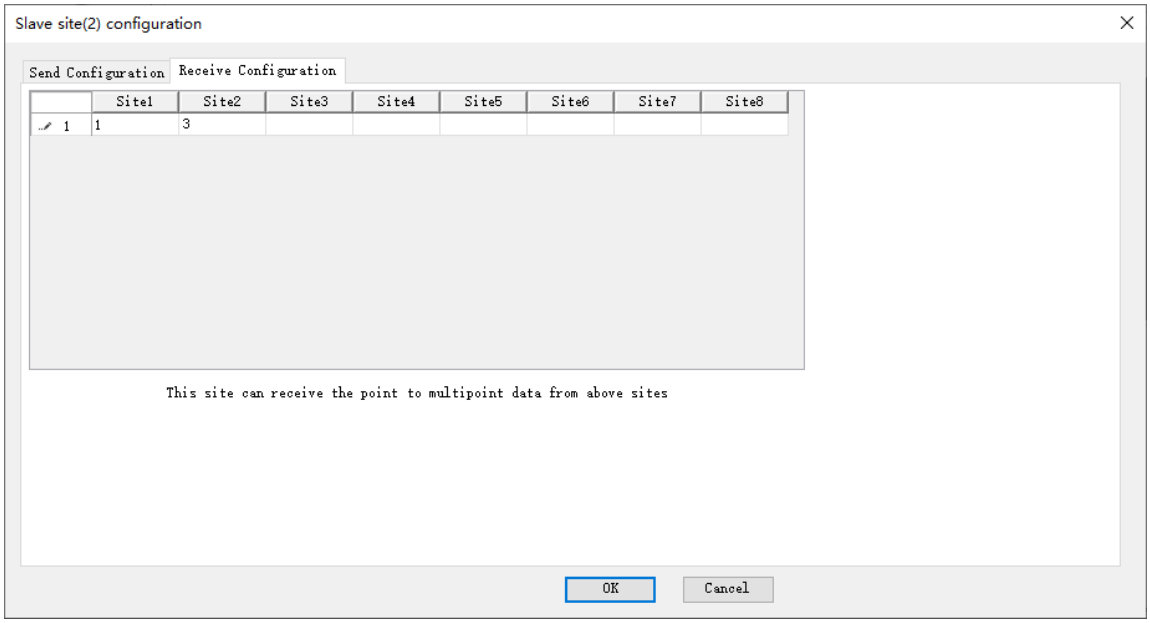


Figure 4-54 "Receive Configuration" dialog box

In the preceding figure, the receive configuration of slave 2 contains stations 1 and 3, indicating that slave 2 only receives point-to-multi-point data frames from stations 1 and 3. If the destination register address of the received point-to-multi-point data meets the definition of station 2, the received data takes effect; otherwise, the received data is discarded.

Each station can receive point-to-multi-point data from up to eight stations. Each send station does not limit the number of stations that receive point-to-multi-point data.

The point-to-multi-point sending function allows a station to modify the same parameter number on multiple stations and implement synchronization.

4.9.8 Synchronous Write by the Master

Synchronization configuration is specific to the master and allows the master to write the multiple registers or parameters of one or more slaves. For example, the master can control multiple servos to start or stop simultaneously by writing the parameter H31-00.

Trigger element

For example, after you enter a trigger element in "Host Syn Write Trigger Element (SM)" in the configuration wizard, double-click the master on the main screen to enter master configuration. On the "Sync Configuration" page, select a configuration condition from the "Trigger Condition" drop-down list. If no trigger element is entered in the configuration wizard, you can still open the "Sync Configuration" page but the "Trigger Condition" drop-down list is unavailable. In this case, click "Site Management" on the main screen to open the configuration wizard again and add or delete trigger elements.

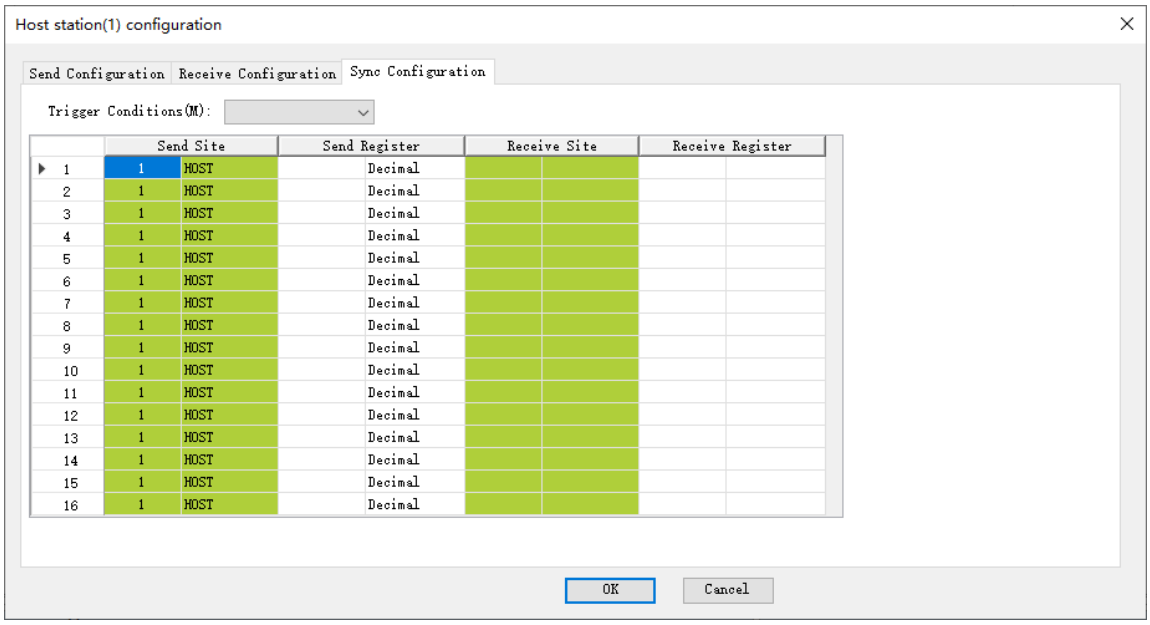


Figure 4-55 "Sync Configuration" page for the master

When a synchronous write trigger element of the master is set to "ON", all the configuration data under the trigger element is sent in sequence. The slaves receive the data and store it in the buffer. When the master detects that all the synchronous write operations under the trigger condition are successfully sent, the master broadcasts an effectiveness command so that all the receive slaves retrieve data from the buffer and make the data effective.

The master supports synchronous write based on up to 8 different trigger conditions, each of which can be configured with 16 synchronous write operations. A single slave can receive up to 8 synchronous write operations. A prompt is displayed when this limit is exceeded.

Operation on the 32-bit register of the servo

CANlink 3.0 supports operations on 16-bit registers and parameters. To operate 32-bit registers or parameters, use the following method:

Write the upper and lower 16-bit addresses of 32-bit registers or parameters through the synchronous write function of the master.

For example, for the parameter H11-12 (1st displacement) of the servo, the following operation writes SD1000 and SD1001 of the master to H11-12 of servo 3 as lower 16 bits and upper 16 bits, respectively.

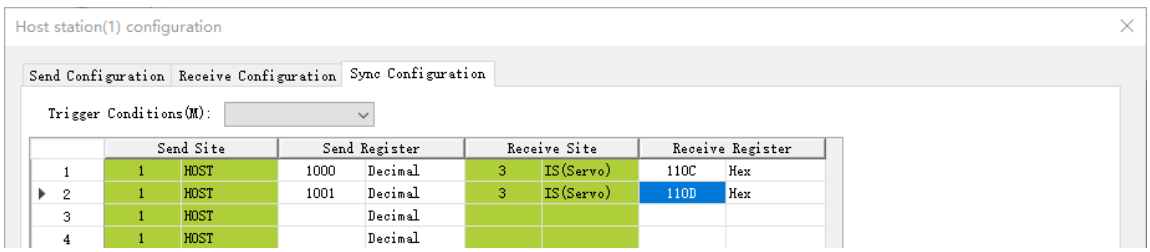


Figure 4-56 Two 16-bit registers combined in the 32-bit format

It is recommended that you use the SET statement of the upper- and lower-edge conducting trigger element of the M element. When operating the parameters of 32-bit addresses, you cannot only

operate the addresses of the lower and upper 16 bits or split the addresses of the lower and upper 16 bits of 32-bit parameters to different trigger conditions.

You can use general send configuration to write the values of two consecutive SD elements to the lower address bits of the 32-bit parameters of the servo, as shown in the following figure:

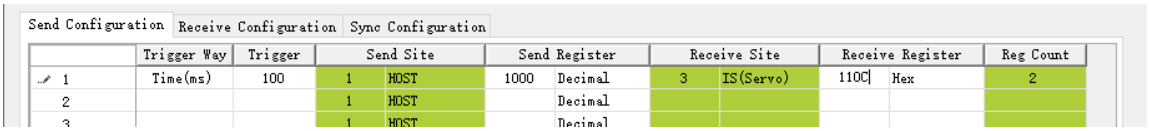


Figure 4-57 Writing a 32-bit value to two 16-bit registers

4.9.9 Local Slave Configuration

A local slave is a PLC serving as a CANlink slave. A remote slave is attached to the master, such as a servo, AC drive, temperature control module, or PLC. A PLC can serve as a master or local slave.



Figure 4-58 Dialog box for local slave configuration

Site No.: Identifies a local slave. It cannot be the same as the number of any other station in the CANlink network. The value ranges from 1 to 63.

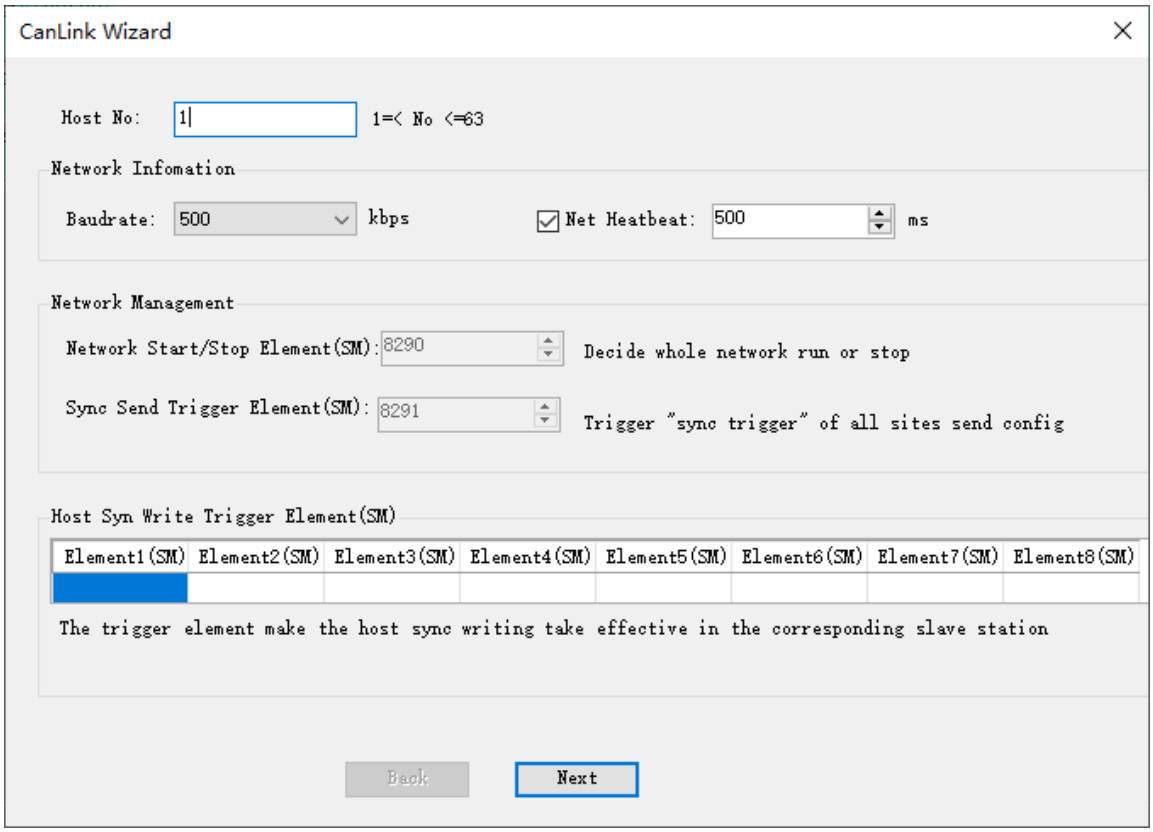
Baudrate: Indicates the baud rate of data communication of the local slave. It must be consistent with the baud rate of the master.

4.9.10 设备接入CANlink3_en.0 网络

Connecting an AM600-series PLC to the CANlink 3.0 network

The station number and baud rate of AM600 can be set only in software. To make the settings take effect, you need to restart the machine or download the running program again.

Each station number in the same network must be unique. If station numbers are repeated, subsequently connected stations disable communication. Station number 0 is not allowed in the network.

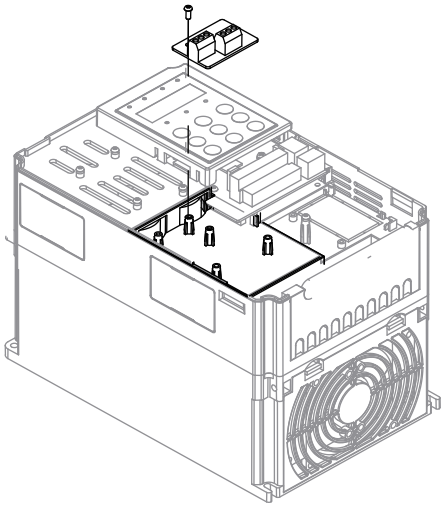


Connecting the MD380/500 AC drive to the CANlink 3.0 network

Installing the MD38CAN1 expansion card of the AC drive

Insert the MD38CAN1 expansion card into the Inovance AC drive. The expansion card (CANlink) cannot be installed or uninstalled in the power-on state. Power off the AC drive before installation, and wait 10 minutes until the power indicator of the AC drive is off. Install the expansion card in accordance with the following figure.

Insert the MD38CAN1 expansion card into the AC drive and fasten screws.



Configuring the MD380/500 AC drive

To configure start/stop control of the AC drive through the CANlink network, set the command source of the AC drive to be the communication command channel. That is, set F0-02 to 2. If start/stop control is not required, set F0-02 based on the actual situation.

Setting the station number

Set the station number through Fd-02. The value ranges from 1 to 63. A value beyond the range may result in access failure of CANlink 3.0. The modified value takes effect immediately. Each station number in the network must be unique. If station numbers are repeated, subsequently connected stations may disable communication.

Setting the baud rate

Set the baud rate through Fd-00. The thousands position indicates the baud rate of CANlink. The following table lists the mapping relationships.

Parameter Address	Name	Value Range	Default
HFd-00	Baud rate	Ones position: Modbus Tens position: PROFIBUS-DP Hundreds position: Reserved Thousands position: CANlink 0: 20 kbps 1: 50 kbps 2: 100 kbps 3: 125 kbps 4: 250 kbps 5: 500 kbps 6: 1000 kbps	5005

The baud rates of all the stations in the CANlink 3.0 network must be consistent; otherwise, communication may be abnormal.

Note

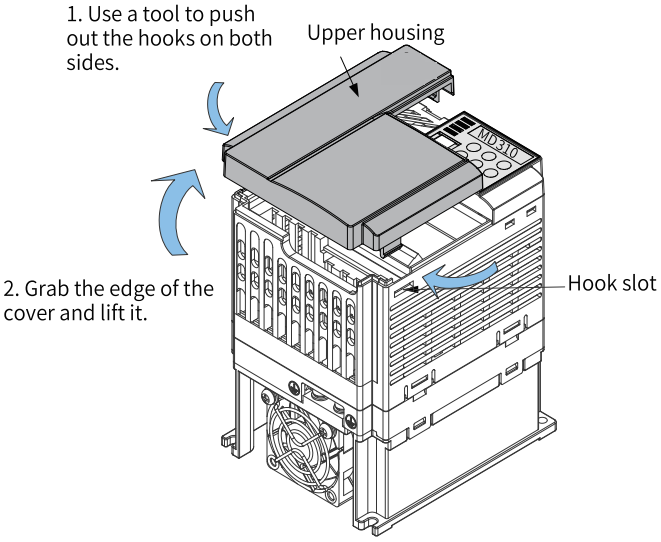
MD380/500 AC drives do not support the 800 kbps baud rate.

Connecting the MD310 AC drive to the CANlink 3.0 network

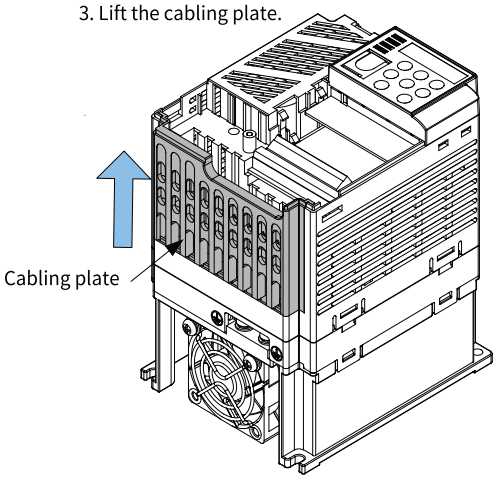
Installing the MD310-CANL card

Insert the MD310-CANL card into the Inovance AC drive. The card cannot be installed or uninstalled in the power-on state. Power off the AC drive before installation, and wait 10 minutes until the power indicator of the AC drive is off. Install the expansion card in accordance with the following figure.

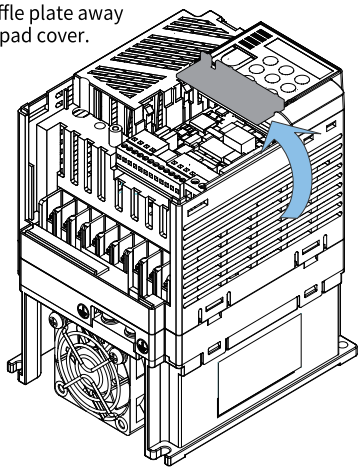
1. Use a tool to push out the hooks on both sides.



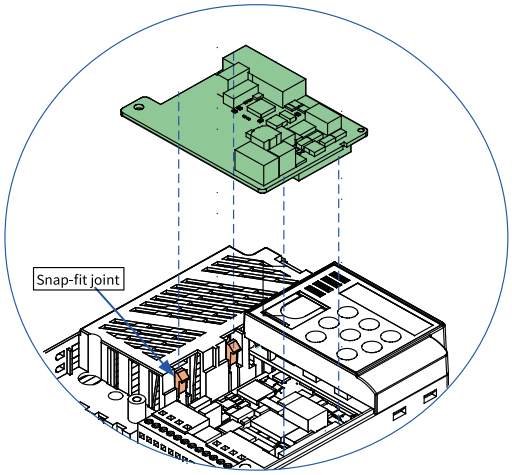
3. Lift the cabling plate.



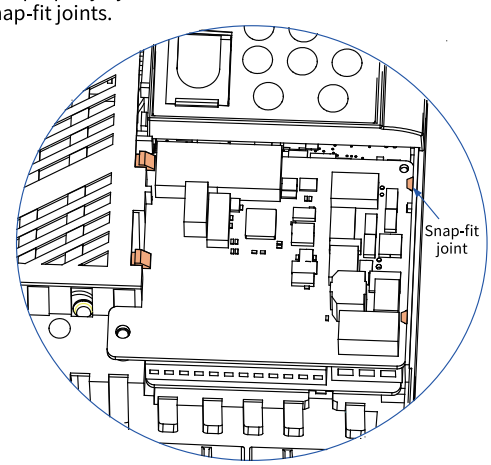
4. Lift the baffle plate away from the keypad cover.



5. Insert the expansion card below the two snap-fit joints on the left, pry open the right side of the middle housing, and then press the right edge of the expansion card to place.



6. Ensure that the expansion card is fastened properly by the four snap-fit joints.



Configuring the MD310

When the MD310 uses CANlink 3.0, the station number and baud rate are set by the DIP switch of the CAN expansion card. The S1 and S2 of the MD310-CANL DIP switch form a 10-bit DIP switch used to set the communication baud rate of the CAN bus and the communication device address. The following figure shows the numbers on the DIP switch. Bd1, 2, and 3 are used to set the baud rate, and Adr1 to 7 are used to set the CANlink address. The "ON" position of the DIP switch indicates 1, and the lower position indicates 0. The modified baud rate and station number take effect immediately.

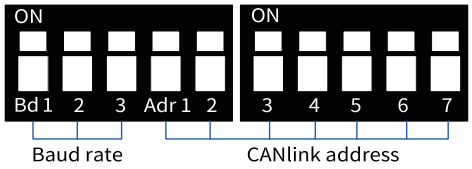


Figure 4-59 MD310-CANL DIP switch

Setting the baud rate

The following table lists the mapping relationships between the DIP switch and the baud rate. Up to eight baud rates can be set.

MD310-CANL baud rates

DIP Switch Bd			Baud Rate
1	2	3	
0	0	0	20 kbps
0	0	1	50 kbps
0	1	0	100 kbps
0	1	1	125 kbps
1	0	0	250 kbps
1	0	1	500 kbps
1	1	0	800 kbps
1	1	1	1 Mbps

CANlink device address

MD310-CANL provides a 7-bit DIP switch for setting the CANlink communication address. Adr1 of the DIP switch indicates the highest bit, and Adr7 indicates the lowest bit. Adr1 to 7 correspond to b6 to b0 of a station number. The valid address setting range of the DIP switch is 1 to 63, as shown in the following table. 0 and 64 to 127 are reserved addresses and cannot be used. The MD310-CANL card is not functional when a reserved address is set.

MD310-CANL DIP switch addresses

DIP Switch Adr							Address
1	2	3	4	5	6	7	
0	0	0	0	0	0	0	Reserved
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	2
0	0	0	0	0	1	1	3
...							...
0	1	1	1	1	1	1	63
1	x	x	x	x	x	x	Reserved

For the usage instructions of the MD310-CANL expansion card, see MD310-CANL Communication Expansion Card Guide.

Connecting the servo to the CANlink 3.0 network

Setting the station number

Modify the station number of the servo through H0C-00. The value ranges from 1 to 63. A value beyond the range may result in access failure of CANlink 3.0. The modified value takes effect immediately. Each station number in the network must be unique. If station numbers are repeated, subsequently connected stations may disable communication.

Setting the baud rate

Modify the baud rate through H0C-08. See the following table.

Parameter Address	Value Range	Default
H0C-08	0: 20 kbps 1: 50 kbps 2: 100 kbps 3: 125 kbps 4: 250 kbps 5: 500 kbps 6: 800 kbps[1] 7: 1000 kbps	5

IS620P does not support the 800 kbps baud rate. When 6 is selected, the 1000 kbps baud rate is used.

The baud rate takes effect immediately after setting. The baud rates of all the stations in the network must be consistent; otherwise, communication may be abnormal.

4.10 CAN Free Protocol

4.10.1 Overview

Caution

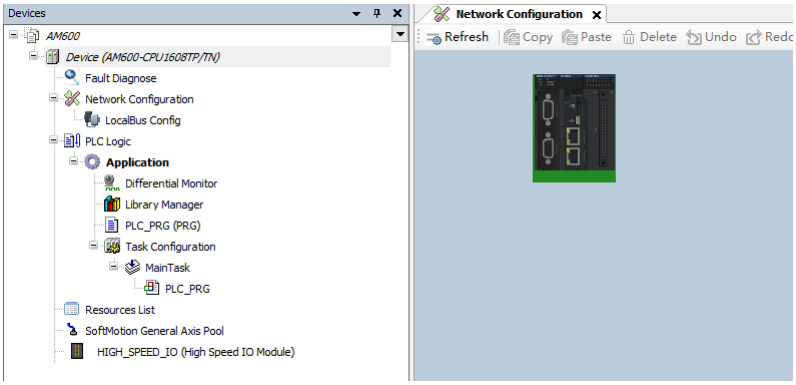
Only AM400/AM600/AM300/AM500 support the CAN free protocol.

AM400/AM600/AM300/AM500 adopts the CAN2.0B protocol. The protocol supports standard frames and expansion frames and is applicable to scenarios where the application-layer protocol is customized. The baud rate ranges from 10 kbps to 1000 kbps.

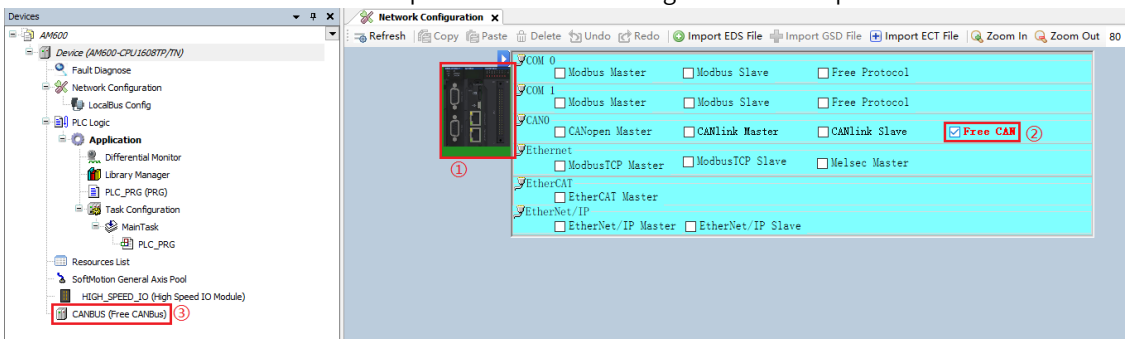
4.10.2 Network Configuration

AM400/AM600 network configuration

1. In the left device tree, double-click "Network Configuration". The "Network Configuration" page is displayed.

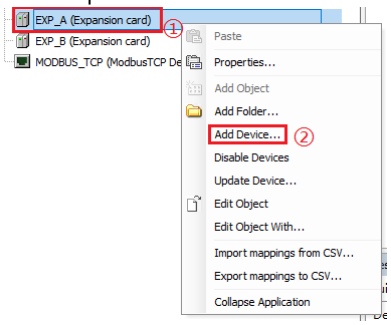


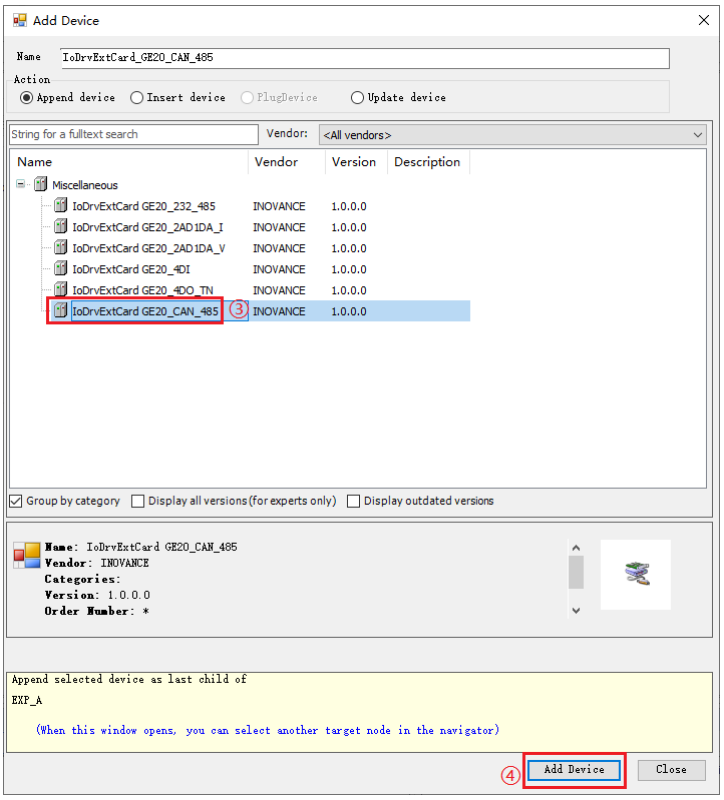
2. Click the PLC picture, select "Free CAN". Then, "CANBUS (Free CANBus)" is automatically added to the left device tree and the CAN free protocol network configuration is completed.



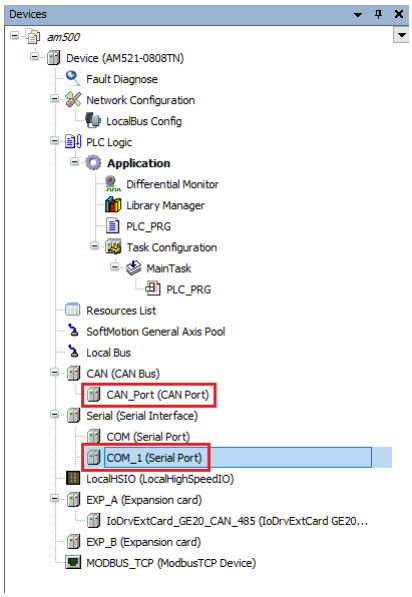
AM300/AM500 Network Configuration

1. In the left device tree, right-click "EXP_A (Expansion card)". In the shortcut menu, select "Add Device". The "Add Device" dialog box is opened.

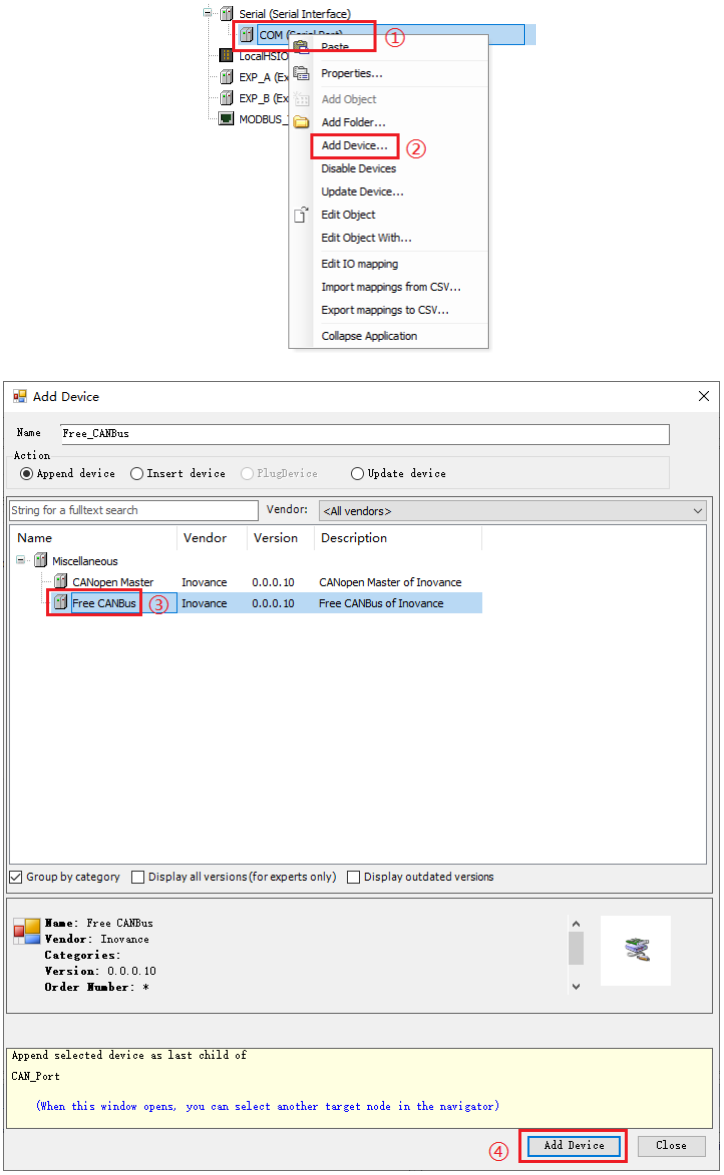




2. Select "GE20_CAN_485" and click "Add Device".
   After the expansion card is added, the "CAN_Port (CAN Port)" and "COM_1 (Serial Port)" are automatically added to the left device tree.



3. Right-click "CAN_Port (CAN Port)". In the shortcut menu, select "Add Device". The "Add Device" dialog box is opened.

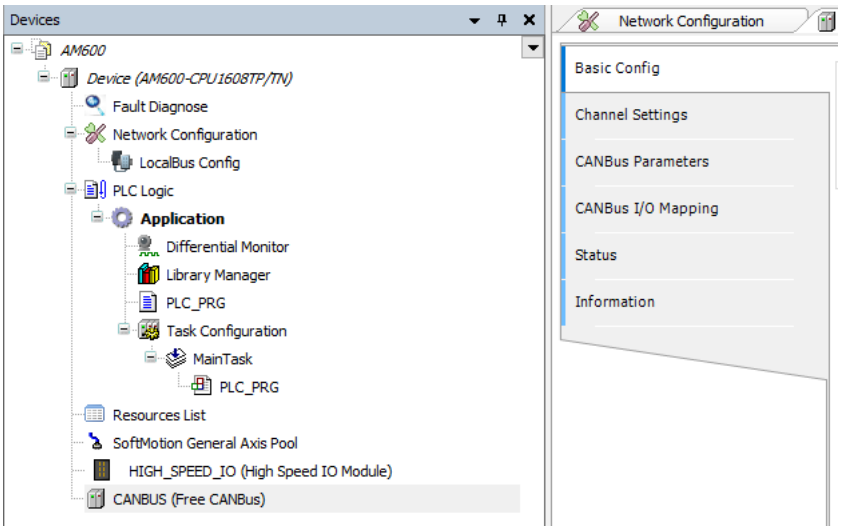


4. Select "Free CANbus" and click "Add Device". The CAN free protocol network configuration is completed.

4.10.3 CAN Free Protocol Configuration

The methods to configure the AM400/AM600 and AM300/AM500 series PLCs are different. This section takes the AM600 series PLC as an example to describe how to configure the protocol for the device.

1. In the left device tree, double-click "CANBUS (Free CANBus)" (for AM300/AM500, "CAN Free Protocol (CAN Free Protocol)") is displayed in the device tree. The CAN free protocol configuration page is displayed.



2. Configure the network.

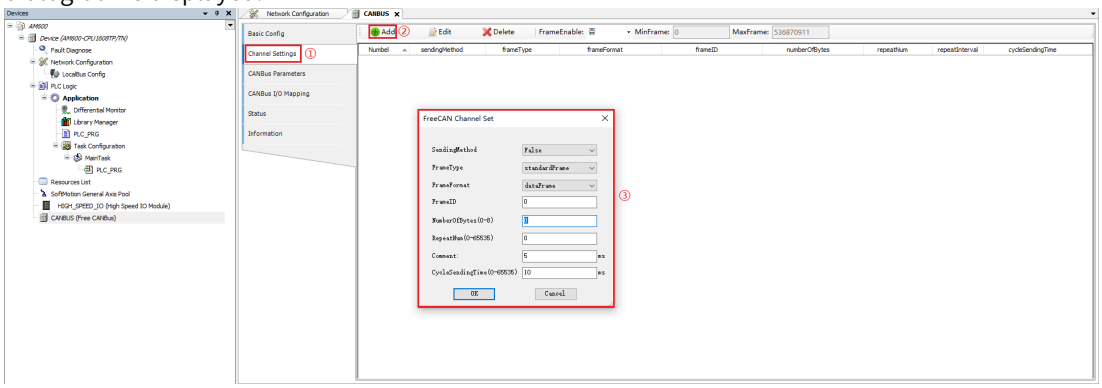
   Set the network number and baud rate.

Caution

- When a function block is used, the network number is the unique ID used to identify specific CANBus device in the function block.
- Skip this step for the AM300/AM500 series PLCs.

3. Configure the channel.

   a. Click "Channel Settings". On the tab page displayed, click "Add". The "FreeCAN Channel Set" dialog box is displayed.



   b. Configure the CAN free protocol communication parameters. The following table lists the relevant parameters.

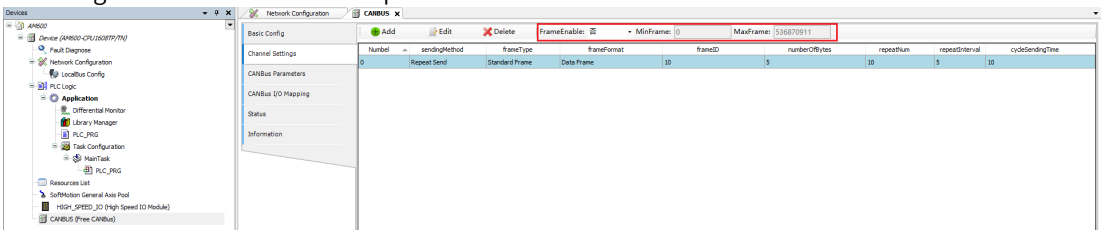
Parameter	Description	Setting
SendingMethod	Indicates the data sending method. **Note:** The trigger mode is implemented by the CANBus_ DevTxMsg function block.	Set it to "Cyclic". Default: "Cyclic".
FrameType	The frame type includes standard frames and expansion frames.	Set as required. Default: Standard frame

Parameter	Description	Setting
FrameFormat	The frame format includes data frames and remote frames. **Caution:** When the remote frame is selected, 0-byte data is sent no matter the value of "Number of Bytes".	Set as required. Default: Data frame
FrameID	It is in the decimal format and supports range detection. For standard frames, 11 bits are effective. For expansion frames, 29 bits are effective.	Set as required. Default: 0
NumberOfBytes (0-8)	Indicates the number of bytes.	Set as required. Value range: 0 to 8 Default: 0
RepeatNum (0-65535)	Indicates the number of repeat data sending attempts. 0 indicates that the data is not re-sent upon a sending failure.	Set as required. Value range: 0 to 65535 Default: 0
Repeat Interval (0-65535)	Indicates the interval of two consecutive data sending attempts.	Set as required. Value range: 0 to 65535 Default: 5
CycleSendingTime (0-65535)	Indicates the cyclic sending interval. **Caution:** A too small cycle sending time will result in high CPU usage.	Set as required. Value range: 0 to 65535 Default: 10

Note

- CANBus is a non-real-time task affected by other high-priority real-time tasks such as MainTask. Therefore, there is an error in the CAN message sending time. When the execution time of high-priority tasks (such as MainTask) is less than 1 ms, the frame interval error does not exceed ±1 ms.
- In some applications, the CAN free protocol frame interval error will be more than ±1 ms, which is actually caused by the high-priority real-time tasks seizing CPU resources during the system scheduling process, and the frame interval time error can be reduced by the following ways:
  - Increase the cycle of high-priority tasks (such as MainTask) to reduce the CPU loading capacity.
  - Optimize the application program to reduce the execution time of each high-priority task (such as MainTask).

c. (Optional) Configure "FrameEnable" to "Yes" or "No", as shown in the following figure. The following table describes relevant parameters.



Parameter	Description	Setting
FrameEnable	Enables or disables the filtering mechanism of the received frame IDs.	Set as required. Default: No
MinFrame	This setting is effective when "FrameEnable" is set to "Yes".	Set as required. Default: 0
MaxFrame	This setting is effective when "FrameEnable" is set to "Yes".	Set as required. Default: 536870911

d. (Optional) Download the program and log in to the PLC. Then, click "CANBus Parameters", "CANBus I/O Mapping", or "Status" to view the CANBus parameters, CANBus I/O mappings, and the status information, respectively.

4.10.4 CANBus Library

4.10.4.1 Enumeration Types of CANBus Library

CANBus_BaudrateType_Enum: Enumeration type of CANBus baud rate

ENUM CANBus_BaudrateType_Enum

Name	Type	Inherited from	Address	Initial	Comment
CANBUS_BAUDRATE_10KBPS	UINT			10	
CANBUS_BAUDRATE_20KBPS	UINT			20	
CANBUS_BAUDRATE_50KBPS	UINT			50	
CANBUS_BAUDRATE_100KBPS	UINT			100	
CANBUS_BAUDRATE_125KBPS	UINT			125	
CANBUS_BAUDRATE_250KBPS	UINT			250	
CANBUS_BAUDRATE_500KBPS	UINT			500	
CANBUS_BAUDRATE_800KBPS	UINT			800	
CANBUS_BAUDRATE_1000KBPS	UINT			1000	

CANBus_CtrlCmdType_Enum: Enumeration type of CANBus control command

ENUM CANBus_CtrlCmdType_Enum

Name	Type	Inherited from	Address	Initial
CANBUS_CTRLCMD_INVALID	UINT			0
CANBUS_CTRLCMD_RESTART	UINT			1
CANBUS_CTRLCMD_START	UINT			2
CANBUS_CTRLCMD_STOP	UINT			3
CANBUS_CTRLCMD_CLEAR	UINT			4

CANBus_ErrorType_Enum: Enumeration type of CANBus error

ENUM CANBus_ErrorType_Enum

Name	Type	Inherited from	Address	Initial
CANBUS_ERR_OK_NOERR	UINT			0
CANBUS_ERR_CFG_ERR	UINT			1
CANBUS_ERR_DRV_ILLEGAL_ERR	UINT			2
CANBUS_ERR_DRV_NO_ACK_ERR	UINT			3
CANBUS_ERR_DRV_BUSOFF_ERR	UINT			4
CANBUS_ERR_BUS_LOAD_OV_ERR	UINT			5
CANBUS_ERR_TX_BUSY_ERR	UINT			16
CANBUS_ERR_TX_TIMEOUT_ERR	UINT			17
CANBUS_ERR_TX_BUFF_FULL_ERR	UINT			18
CANBUS_ERR_RX_BUFF_EMPTY_ERR	UINT			32
CANBUS_ERR_RX_BUFF_FULL_ERR	UINT			33
CANBUS_ERR_OTHER_UNDEF_ERR	UINT			65535

CANBus_FBErrorType_Enum: Enumeration type of CANBus function block error

ENUM CANBus_FBErrorType_Enum

Name	Type	Inherited from	Address	Initial
CANBUS_FBERR_NO_ERROR	UINT			0
CANBUS_FBERR_EXE_FAILED	UINT			1
CANBUS_FBERR_INVAILD_DEVICEID	UINT			2
CANBUS_FBERR_DEVICE_FAULT	UINT			3
CANBUS_FBERR_WRONG_PARAMETER	UINT			4
CANBUS_FBERR_BUSY_ERROR	UINT			5
CANBUS_FBERR_ABORT_ERROR	UINT			6
CANBUS_FBERR_TIME_OUT_ERROR	UINT			7
CANBUS_FBERR_POINTER_NULL	UINT			8
CANBUS_FBERR_BUFF_FULL	UINT			9
CANBUS_FBERR_UNKNOWN_ERROR	UINT			255

CANBus_FBStateType_Enum: Enumeration type of CANBus function block state

ENUM CANBus_FBStateType_Enum

Name	Type	Inherited from	Address	Initial
CANBUS_FBSTATE_DORMANT	UINT			
CANBUS_FBSTATE_EXECUTING	UINT			
CANBUS_FBSTATE_ABORTING	UINT			
CANBUS_FBSTATE_DONE	UINT			
CANBUS_FBSTATE_ERROR	UINT			
CANBUS_FBSTATE_RESETTING	UINT			

CANBus_StateType_Enum: Enumeration type of CANBus state

ENUM CANBus_StateType_Enum

Name	Type	Inherited from	Address	Initial
CANBUS_STATE_UNKNOWN	UINT			0
CANBUS_STATE_INITIALIZE	UINT			1
CANBUS_STATE_RUNNING	UINT			2
CANBUS_STATE_STOPPING	UINT			3
CANBUS_STATE_FAULTRUN	UINT			4
CANBUS_STATE_FAULTSTOP	UINT			5

4.10.4.2 Structure Types of CANBus Library

CANBus_IoDrvVer_t: Version of the drive component used by CANBus

STRUCT CANBus_CmpDrvVer_t

Name	Type	Inherited from	Address	Initial
dwCmpDrvID	DWORD			
dwCmpDrvVer	DWORD			
dwCmpDrvDate	DWORD			

Parameters of the CANBus_IoDrvVer_t type are all in the hexadecimal format.

CANBus_DiagInfo_t: Structure of the CANBus diagnosis information

STRUCT CANBus_DiagInfo_t

Name	Type	Inherited from	Address	Initial
xEnableFlag	BOOL			
xErrorFlag	BOOL			
uiNetId	UINT			
eStateType	CANBus_StateType_Enum			
eErrorType	CANBus_ErrorType_Enum			
uiBaudrate	UINT			
uiBusload	UINT			
uiTxErrCnt	UINT			
uiRxErrCnt	UINT			
udiTxMsgNumInQueue	UDINT			
udiTxMsgNumFinish	UDINT			
udiTxBitCntPerSecond	UDINT			
udiRxMsgNumInQueue	UDINT			
udiRxMsgNumFinish	UDINT			
udiRxBitCntPerSecond	UDINT			

CANBus_FrameMsg_t: Structure of the CANBus frame information

STRUCT CANBus_FrameMsg_t

Name	Type	Inherited from	Address	Initial
xExtFlag	BOOL			
xRtrFlag	BOOL			
dwCanId	DWORD			
usiDataLen	USINT			
abyDataStr	ARRAY [0..7] OF BYTE			

CANBus_RxFilter_t: Structure of the CANBus receive frame filter

STRUCT CANBus_RxFilter_t

Name	Type	Inherited from	Address	Initial
xExtFlag	BOOL			
xRtrFlag	BOOL			
dwCanId	DWORD			
xFilterFlag	BOOL			
xFilterExt	BOOL			
xFilterRtr	BOOL			
dwFilterCanId	DWORD			

4.10.4.3 CANBus Function Blocks

The following table lists the name, meaning, and trigger mode of CANBus function blocks.

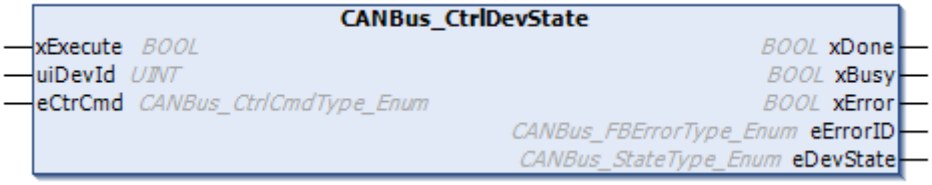
Name	Meaning	Trigger Mode
CANBus_CtrlDevState	State of the control device	Edge trigger (rising edge)
CANBus_DevRxMsg	Information about the single frame received by the device	Level trigger (high level), singe frame cached
CANBus_DevRxMultiMsg	Information about the multiple frames received by the device	Level trigger (high level), multiple frames cached
CANBus_DevTxMsg	Device send information	Edge trigger (rising edge)
CANBus_GetDevDiagInfo	Get the device diagnosis information	Level trigger (high level)

Name	Meaning	Trigger Mode
CANBus_GetIoDrvVer	Get the version of the CANBus component	Level trigger (high level)
CANBus_SetDevBaud	Set the device baud rate	Edge trigger (rising edge)

The following describes CANBus function blocks:

- The input parameter uiDevId (device ID) in the function block must correspond to the network ID of the CANBus device.
- The input condition of the edge trigger function block is "xExecute", indicating that the function block is triggered at the rising edge and is aborted or reset at the low level.
- The input condition of the level trigger function block is "xEnable", indicating that the function block is triggered at the high level and is aborted or reset at the low level.
- Up to 128 instantiated receive function blocks can be enabled at a time, including CANBus_DevRxMsg and CANBus_DevRxMultiMsg.
- Only one instantiated CANBus_DevTxMsg function block can be enabled at a time; otherwise, the error "Sending busy" is reported.

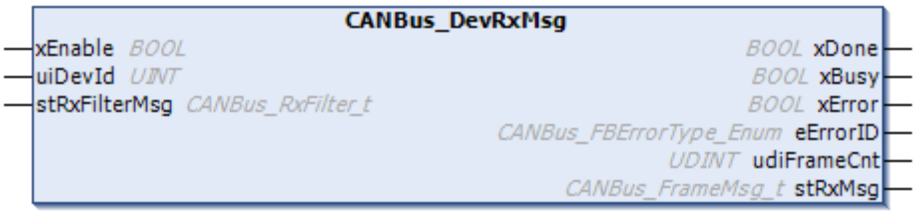
CANBus_CtrlDevState



FUNCTION_BLOCK CANBus_CtrlDevState

Name	Type	Inherited from	Address	Initial
xExecute	BOOL			
uiDevId	UINT			
eCtrCmd	CANBus_CtrlCmdType_Enum			
xDone	BOOL			FALSE
xBusy	BOOL			FALSE
xError	BOOL			FALSE
eErrorID	CANBus_FBErrorType_Enum			
eDevState	CANBus_StateType_Enum			

CANBus_DevRxMsg



FUNCTION_BLOCK CANBus_DevRxMsg

Name	Type	Inherited from	Address	Initial
xEnable	BOOL			FALSE
uiDevId	UINT			
stRxFilterMsg	CANBus_RxFilter_t			
xDone	BOOL			FALSE
xBusy	BOOL			FALSE
xError	BOOL			FALSE
eErrorID	CANBus_FBErrorType_Enum			
udiFrameCnt	UDINT			
stRxMsg	CANBus_FrameMsg_t			

Parameter	Description
xEnable	The function block is enabled at high level and reset at the low level.
uiDevId	The device ID, which must correspond to the network ID of the CANBus device.
stRxFilterMsg	Set the receive parameters and filter parameters.
udiFrameCnt	The frame counter, indicating the number of received frames that meet the conditions.
stRxMsg	Save information of the received frames that meet the conditions (only the last received CAN message is cached).

CANBus_DevRxMultiMsg



FUNCTION_BLOCK CANBus_DevRxMultiMsg

Name	Type	Inherited from	Address	Initial
xEnable	BOOL			FALSE
xBuffResetFlag	BOOL			FALSE
xBuffReadFlag	BOOL			FALSE
uiDevId	UINT			
stRxFilterMsg	CANBus_RxFilter_t			
xDone	BOOL			FALSE
xBusy	BOOL			FALSE
xError	BOOL			FALSE
eErrorID	CANBus_FBErrorType_Enum			
uiBuffFrameMaxNum	UINT			0
uiBuffFrameNum	UINT			0
udiFrameCnt	UDINT			0
stRxMsg	CANBus_FrameMsg_t			
astRxFrameBuff	POINTER TO CANBus_FrameMsg_t			

Parameter	Description
xEnable	The function block is enabled at high level and reset at the low level.
xBuffResetFlag	The reset buffer flag. The value "True" indicates clearing the buffer. The buffer is filled again and the function block is automatically zeroed. • Manually reset the received buffer data. • Triggered when the function block is enabled. • When executed at high level, xBuffResetFlag is automatically reset.
xBuffReadFlag	The read buffer flag. The value "TRUE" indicates that the application layer reads data from the buffer. This state can prevent the function block from writing data to the buffer. • During buffer reading, you can reset this flag to prevent new data from overwriting existing data in the buffer. • Note that during buffer reading, the internal register only saves the last received message. • To avoid frame loss, reset this flag timely at the possible smallest internal.
uiDevId	The device ID, which must correspond to the network ID of the CANBus device.
stRxFilterMs	Set the receive parameters and filter parameters.
uiBuffFrameMaxNum	Indicates the maximum number of received and buffered frames, which is automatically calculated based on the structure array.
uiBuffFrameNum	Indicates the number of received and buffered frames. After the buffer is used up, you can reset xBuffResetFlag to clear the buffer.

Parameter	Description
udiFrameCnt	The frame counter, indicating the number of received frames that meet the conditions.
stRxMsg	Save information of the received frames that meet the conditions (only the last received CAN message is cached).
astRxFrameBuff	Indicates the received frame buffer. It is a one-dimensional length-variable structure array. The number of array elements cannot exceed 65535. • The interface type of astRxFrameBuff is a structure pointer (POINTER TO CANBus_FrameMsg_t). Actually, it is a variable-length structure array (ARRAY[*] OF CANBus_FrameMsg_t). • Structure arrays are necessary for parameter transmission. If other data types are used, an error will be reported during programming.

Up to 128 instantiated receive function blocks can be enabled at a time, including CANBus_DevRxMsg and CANBus_DevRxMultiMsg.

CANBus_DevTxMsg



FUNCTION_BLOCK CANBus_DevTxMsg

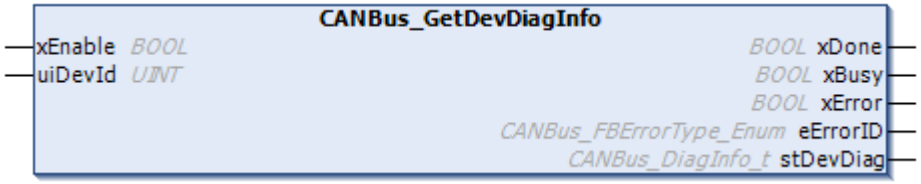
Name	Type	Inherited from	Address	Initial
xExecute	BOOL			
uiDevId	UINT			
uiTimeOutMs	UINT			
stTxMsg	CANBus_FrameMsg_t			
xDone	BOOL			FALSE
xBusy	BOOL			FALSE
xError	BOOL			FALSE
eErrorID	CANBus_FBErrorType_Enum			

Parameter	Description
xEnable	The function block is enabled at rising edge, and aborted and reset at falling edge and low level.
uiDevId	The device ID, which must correspond to the network ID of the CANBus device.

Parameter	Description
uiTmieOutMs	Set the send timeout time, in ms.
stTxMsg	Indicate the send information, including the expansion frame flag, remote frame flag, CANID, data length, and data.

- If the timeout time is set to 0, the send timeout is detected at an interval of 50 ms by default during function block running.
- Only one instantiated CANBus_DevTxMsg function block can be enabled at a time; otherwise, the error "Sending busy" is reported.

CANBus_GetDevDiagInfo

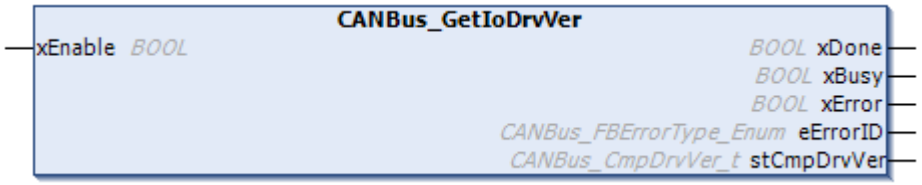


FUNCTION_BLOCK CANBus_GetDevDiagInfo

Name	Type	Inherited from	Address	Initial
xEnable	BOOL			FALSE
uiDevId	UINT			
xDone	BOOL			FALSE
xBusy	BOOL			FALSE
xError	BOOL			FALSE
eErrorID	CANBus_FBErrorType_Enum			
stDevDiag	CANBus_DiagInfo_t			

Parameter	Description
stDevDiag	The diagnosis information is updated in real time, including but not limited to the running state, error state, DevID, running baud rate, bus loading capacity, and send/receive counter.

CANBus_GetIoDrvVer

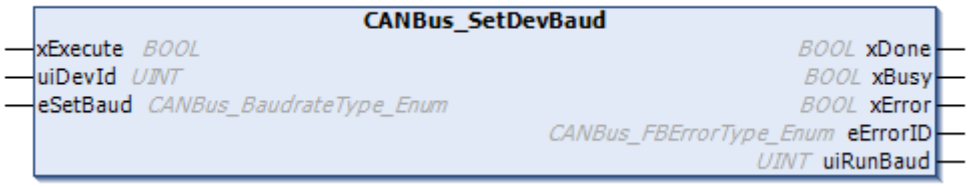


FUNCTION_BLOCK CANBus_GetIoDrvVer

Name	Type	Inherited from	Address	Initial
xEnable	BOOL			FALSE
xDone	BOOL			FALSE
xBusy	BOOL			FALSE
xError	BOOL			FALSE
eErrorID	CANBus_FBErrorType_Enum			
stCmpDrvVer	CANBus_CmpDrvVer_t			

Parameters of the stCmpDrvVer structure are all in the hexadecimal format.

CANBus_SetDevBaud



FUNCTION_BLOCK CANBus_SetDevBaud

Name	Type	Inherited from	Address	Initial
xExecute	BOOL			
uiDevId	UINT			
eSetBaud	CANBus_BaudrateType_Enum			
xDone	BOOL			FALSE
xBusy	BOOL			FALSE
xError	BOOL			FALSE
eErrorID	CANBus_FBErrorType_Enum			
uiRunBaud	UINT			

eSetBaud CAN is used to set the bus baud rate. Use the enumeration type to avoid baud rates not supported by the device.

4.10.4.4 Error Codes of CANBus Function Blocks

The following table lists error codes of CANBus function blocks.

Fault codes of CANBus_CtrlDevState

Error Type	Error Code	Meaning	Solution
CANBUS_FBERR_NO_ ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_ FAILED	0x0001	The function block failed to run.	The input DevID is invalid.
CANBUS_FBERR_INVAILD_ DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_ FAULT	0x0003	The device is faulty.	The device is being started.
CANBUS_FBERR_WRONG_ PARAMETER	0x0004	The input parameters of the function block are incorrect.	The input CtrlCmd is invalid.
CANBUS_FBERR_BUSY_ ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_ OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_ POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_ FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_ UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

CANBus_DevRxMsg

Error Type	Error Code	Meaning	Solution
CANBUS_FBERR_NO_ ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_ FAILED	0x0001	The function block failed to run.	The parameter address is modified.
CANBUS_FBERR_INVAILD_ DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_ FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_ PARAMETER	0x0004	The input parameters of the function block are incorrect.	-
CANBUS_FBERR_BUSY_ ERROR	0x0005	The function block is busy.	The instantiated function block is out of the limit.
CANBUS_FBERR_ABORT_ ERROR	0x0006	An error occurs when the function block is aborted.	-

Error Type	Error Code	Meaning	Solution
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

CANBus_DevRxMultiMsg

Error type	Error code	Meaning	Solution
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	The parameter address is modified.
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	The maximum number of buffers is 0.
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	The instantiated function block is out of the limit.
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	The input/output parameters or the buffer address cannot be specified.
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	The receive buffer is used up.
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

CANBus_DevTxMsg

Error Type	Error Code	Meaning	Solution
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	The input DevID is invalid.
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	The device is not running or faulty.

Error Type	Error Code	Meaning	Solution
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	The CANID is invalid and the data length exceeds the limit (8 for data frames and 0 for remote frames).
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	The send requests are conflicting.
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	The data sending times out and is not successful.
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

CANBus_GetDevDiagInfo

Error type	Error code	Meaning	Solution
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	-
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

CANBus_GetIoDrvVer

Error type	Error code	Meaning	Solution
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	-

Error type	Error code	Meaning	Solution
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	-
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	-
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

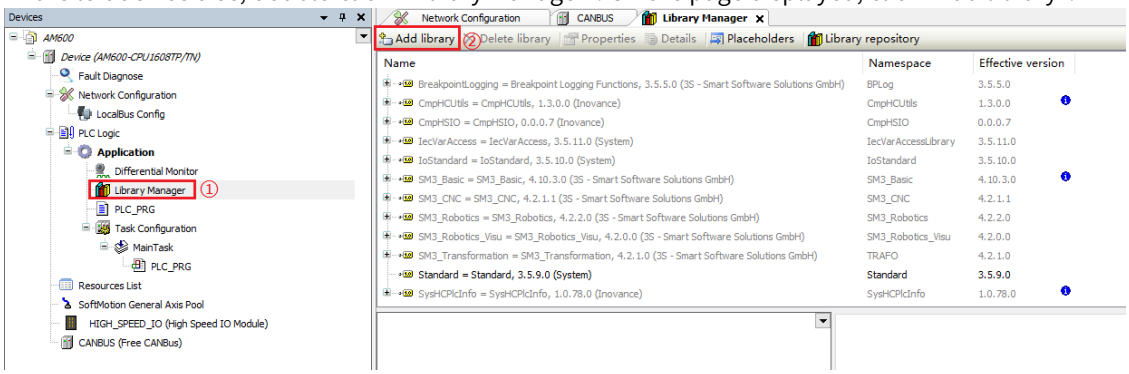
CANBus_SetDevBaud

Error type	Error code	Meaning	Solution
CANBUS_FBERR_NO_ERROR	0x0000	The function block runs normally or has no error.	The function block runs normally.
CANBUS_FBERR_EXE_FAILED	0x0001	The function block failed to run.	Modification of the baud rate failed.
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	The device ID is invalid because this ID is not enabled.	The input DevID is invalid.
CANBUS_FBERR_DEVICE_FAULT	0x0003	The device is faulty.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	The input baud rate is incorrect.
CANBUS_FBERR_BUSY_ERROR	0x0005	The function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	An error occurs when the function block is aborted.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	The function block times out.	-
CANBUS_FBERR_POINTER_NULL	0x0008	The function block pointer is null.	-
CANBUS_FBERR_BUFF_FULL	0x0009	The function block buffer is used up.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	The function block has an unknown error.	-

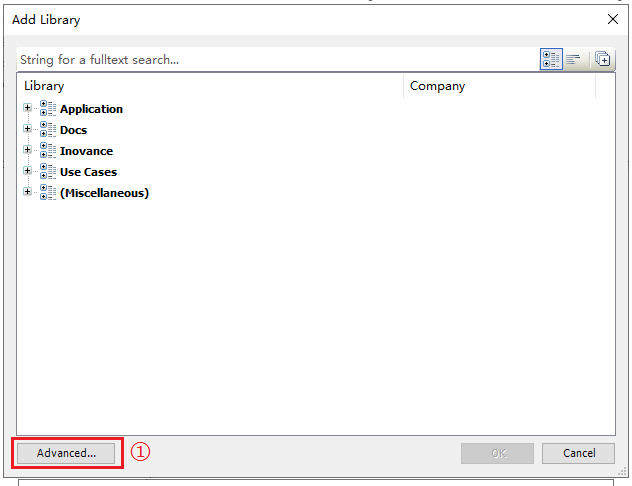
4.10.4.5 Example of Using CANBus Function Blocks

Adding a CANBus library

1. In the left device tree, double-click "Library Manager". On the page displayed, click "Add library".

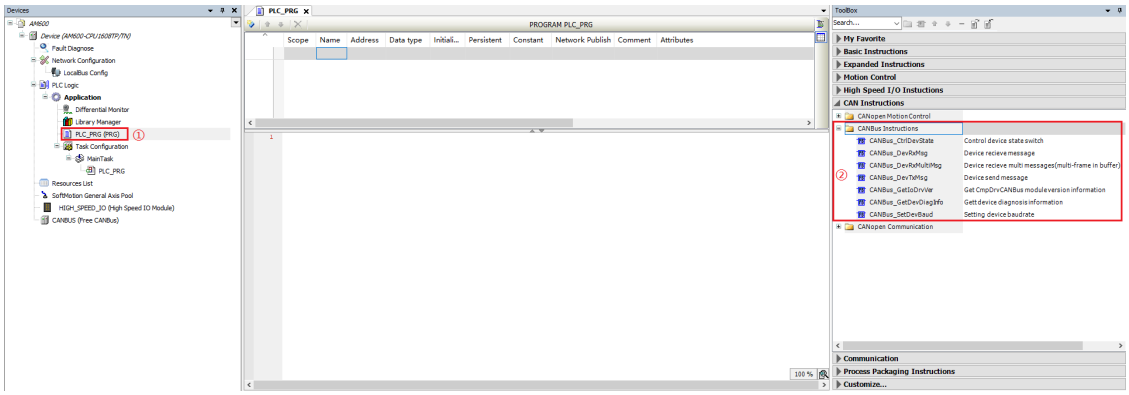


2. In the dialog box displayed, click "Advanced". On the page displayed, unfold the item "Inovance", select "HC_CANBus", and click "OK". The CANBus library is added to the library manager.

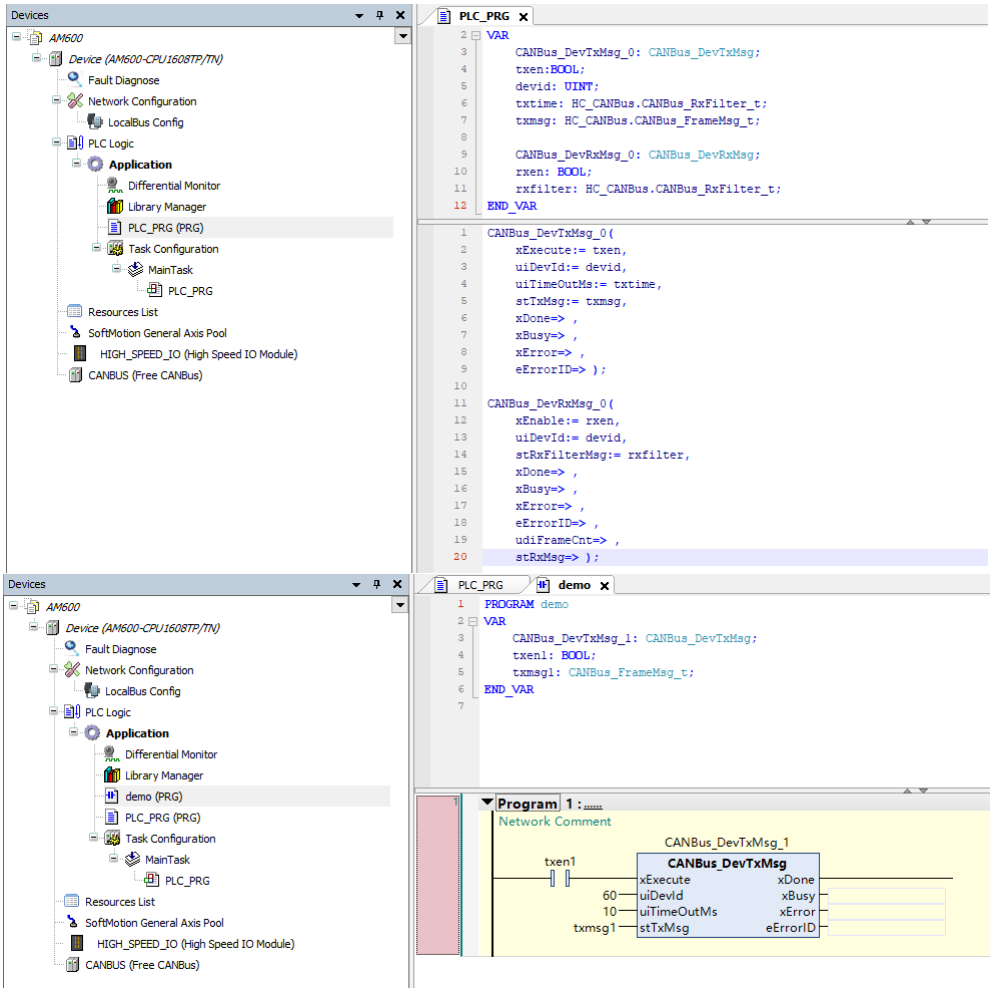


Using the CANBus function block instructions

1. In the left device tree, double-click "PLC_PRG". The program editor page is displayed.



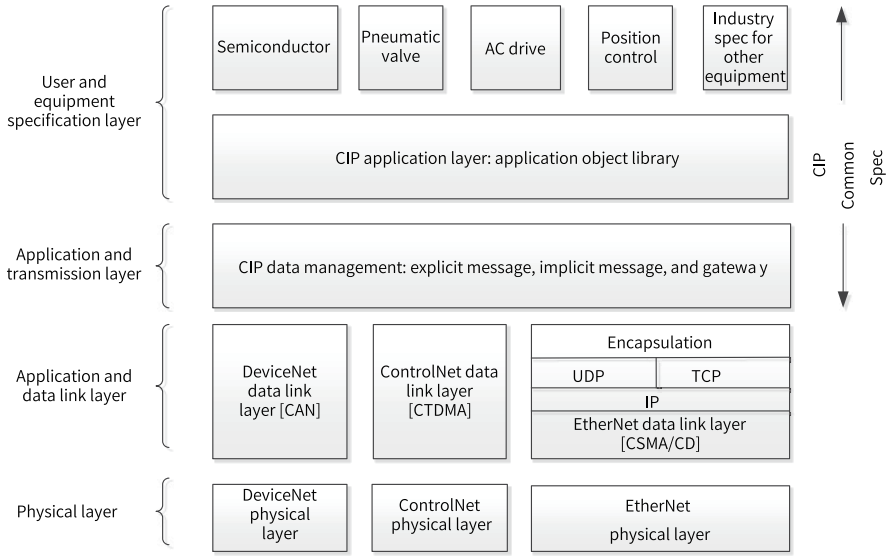
2. In the toolbox section, choose "CAN Instructions" > "CANBus Instructions", and then click the corresponding CANBus function block instruction to use this instruction and create an instance for it. CANBus function block instructions support the ST language and LD language, as shown in the following figure.



4.11 EtherNet/IP Communication

4.11.1 Overview of the Protocol

The EtherNet/IP communication protocol is a protocol system suitable for industrial environments (IP refers to "Industrial Protocol"), based on the traditional Ethernet protocol and the standard TCP/IP protocol, which enables efficient exchange of application information between industrial devices. The EtherNet/IP application layer protocol uses the standard object-oriented CIP protocol [1]. The following figure shows the structure of each layer.



Technical characteristics of the EtherNet/IP protocol:

1. Standard. EtherNet/IP is built on top of the standard TCP/UDP protocol and is fully compliant with the standard IEEE802.3U standard, and all devices with standard Ethernet nodes can join this network.
2. Real-time [2]. EtherNet/IP data transmission is divided into implicit messaging and explicit messaging. Implicit messaging is used to transmit real-time data cyclically. Explicit messaging is used to transmit non-real-time data (such as configuration information), using a request-response approach.
3. Efficient communication. EtherNet/IP uses the producer/consumer technology, which allows nodes on the network to access data from the same source at the same time. In the producer/consumer mode, each data is assigned a unique identifier, and each data source sends the data to the network at one time, and the other nodes read the data selectively, which greatly improves the communication efficiency of the system.
4. Wide application of EtherNet/IP devices. EtherNet/IP communication devices include simple I/O devices, sensors (scanners/cameras), activators, and complex control devices (such as robots, PLCs, and welders).

InoProShop 1.5.0 and later versions support EtherNet/IP function with the following communication specifications:

- All medium-sized PLCs support one EtherNet/IP master and one slave at the same time.

- Both the master and slave support Class-1 tag or instance path connection, and the slave supports Class-3/UCMM service message tag connection.
- The minimum cycle communication period (RPI) is 5 ms.
- One connection supports up to 1400 bytes of data reading and writing.
- An EtherNet/IP master supports a maximum of 64 slaves.
- An EtherNet/IP slave supports a maximum of 32 connections.

Note

- [1]: For protocol details, see the official standard documents EIP-CIP-V1-1.0 and EIP-CIP-V2-1.0.
- [2]: When EtherCAT and EtherNet/IP networks both exist in a networking project, the real-time communication performance of EtherNet/IP network is reduced because EtherCAT has the highest priority by default.

4.11.2 EtherNet/IP Communication Specifications

Item				AM300 Series	AM400 Series	AM500 Series	AM600 Series	AC700 Series	AC800 Series
CIP service	Implicit (I/O) messaging	Number of I/O connections at the originator[1]		16	16	16	16	32	64
		Number of I/O connections at the target end[2]		16	16	16	16	32	64
		RPI (communication cycle)		5 to 50000 (unit: ms)					
		Band-width allowable for implicit (I/O) messaging	(@4 Byte)	6400 (pps)[3]	6400 (pps)	6400 (pps)	6400 (pps)	12800 (pps)	25600 (pps)
			(@250 Byte)	4800 (pps)	4800 (pps)	4800 (pps)	4800 (pps)	12800 (pps)	25600 (pps)
			(@500 Byte)	3200 (pps)	3200 (pps)	3200 (pps)	3200 (pps)	12800 (pps)	25600 (pps)
			(@1400 Byte)	1000 (pps)	1000 (pps)	1000 (pps)	1000 (pps)	12800 (pps)	25600 (pps)
		Maximum data size per connection[4]		1400 bytes	1400 bytes	1400 bytes	1400 bytes	1400 bytes	1400 bytes
		Multicast filter[5]		Supported (IGMP client function)					
	Explicit messaging	Number of Class-3 tags at the target end[6]		128	128	128	128	128	128
		Number of UCMM tags at the target end[6]		128	128	128	128	128	128

[1] I/O connections at the originator include:

- Consumer tags: Connections requested by the originator with the name of the producer tag as the connection path.
- Originator generic I/O connections: Connections requested by the originator with the instance ID of the generic I/O connections of the target end as the connection path.

[2] I/O connections at the target end include:

- Consumer tags: Responses from the target end to the connection requests with the name of the producer tag as the connection path.
- Target end generic I/O connections: Responses from the target end to connection requests with the instance ID of the generic I/O connections of the target end as the connection path.

[3] pps refers to Packet Per Second. pps is the unit of network throughput rate. Here, it means the sum of the number of grouping packets sent and received that can be processed in one second. It is calculated according to the following formula: Communication bandwidth pps = 1000 ms/RPI x Number of connections x 2.

[4] Data simultaneity within a connection is guaranteed. The device used supports Large Forward Open (CIP option specification) when the data size is greater than 509 bytes.

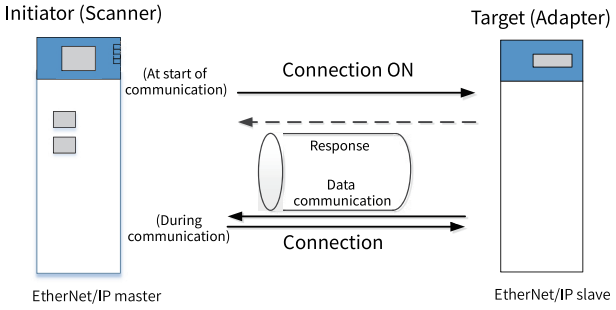
[5] The EtherNet/IP unit supports the IGMP client function, so the use of an Ethernet switch that supports IGMP Snooping allows filtering out unattended multicast packets.

[6] The number of tags is as follows:

- Initiator tags include the consumer tags, initiator generic I/O connections, Class-3 tags at the originator, and UCMM tags at the initiator. The maximum number is 32.
- Target end tags include the producer tags, target end generic I/O connections, Class-3 tags at the target end, and UCMM tags at the target end. The maximum number is 32. The maximum number of target end generic I/O connections is 16.

4.11.3 Configuration of PLC as the EtherNet/IP Master

When communication starts, the end that opens the connection is called the initiating device, also called the scanner, which is usually the EtherNet/IP master. The open end is called the target device, also called the adapter, which is usually the EtherNet/IP slave.

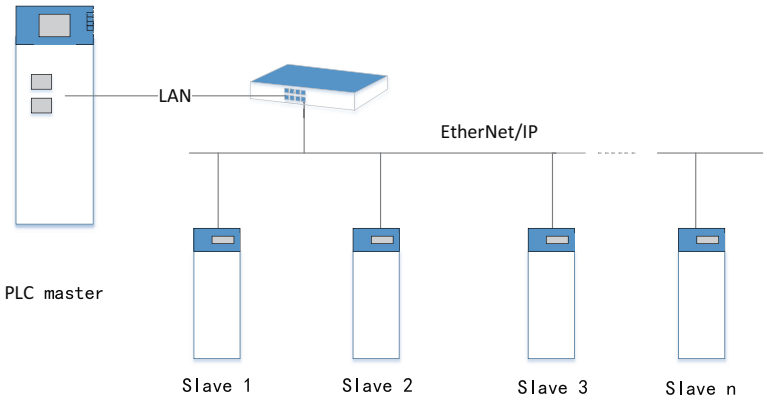


This section describes in detail how to configure the InoProShop software with a medium-sized Inovance PLC as the EtherNet/IP master.

EtherNet/IP device IP settings

EtherNet/IP supports bus topology, star topology, hybrid topology, and ring topology.

In a star topology, all nodes are connected to the network hub, and nodes are easily added, deleted, and maintained. Such topology is often used due to its cost-effectiveness, easy connection, and availability of required devices. In a star topology, IP addresses of all devices must be unique and in the same EtherNet/IP network segment.



For example, IP addresses of devices in the network can be set to the following formats:

IP address of the EtherNet/IP master: 192.168.1.100

IP address of the EtherNet/IP slave 1: 192.168.1.101

IP address of the EtherNet/IP slave 2: 192.168.1.102

...

IP address of the EtherNet/IP slave n: 192.168.1.XXX

Adding EtherNet/IP Remote Slave

1. Import the EDS description file of the slave.
   The InoProShop software provides two methods for importing the EDS files of EtherNet/IP slaves. Method 1: In the device library, choose Tools > Device Library in the menu tool. Click "Install" and then import the file. Method 2: On the "Network Configuration" page, click "Import EDS File" to import the file.

   Take KEYENCE N-L20 EtherNet/IP slave module as an example. You can import the EDS description file of a third-party device according to method 1.

2. Select the "EtherNet/IP" master.
   Click the master. On the page displayed, select "EtherNet/IP Master".

3. Add an EtherNet/IP remote slave.

   - Method 1: Add the slave on the "Network Devices List" page.
     On the "Network Devices List" page on the right, locate the third-party vendor device imported in step 1 under "EtherNet/IP Port List", and right-click the device to add this slave device to the network configuration.

     As shown in the following figure, in the right device tree, the slave N_L20 is generated under the master. If there are multiple EtherNet/IP remote slaves, the slaves can be added one by one.

     Under the task configuration behind the EtherNet/IP device, the system automatically generates two tasks EIPMaster.IOTask and EIPMaster.ServiceTask, which are used to update the cyclic communication data and service data of EtherNet/IP. The default priority of the task EIPMaster.IOTask is 0, which can be adjusted as required. For example, if the priority of the task EtherCAT in the project must be set to 0 (the highest), then the priority of EIPMaster.IOTask can be modified to 1.

- Method 2: Add the slave through the EtherCAT scanning function.
  After adding the EDS file of the slave, log in to the PLC, select "EtherNetIPMaster" from the device tree. In the shortcut menu displayed, right-click "Scan Devices" to scan EtherNet/IP devices in the current network, and then click "Copy All Scanned Devices" to add slaves. For details, see the EtherCAT scanning function.

EtherNet general settings

Before configuring EtherNet general settings, configure the PLC gateway to connect the PLC through the software.

Double-click the Ethernet device to enter the "General" page, as shown in the following figure. Access the "Network Adapters" page.

Select the master IP address of the corresponding network interface.

Note: This IP address refers to the IP address through which the PLC serving as the master and external devices connect to the EtherNet/IP network. The AC800 series PLCs have two network interfaces. Make sure that the IP address of the connected device corresponds to the network interface. The AM600-series PLCs have only one Ethernet port. Ensure that the IP address is consistent with that of the logged-in PLC. For PLCs not scanned on the device initialization page, IP address and other information of the network interface adapters cannot be obtained.

Besides, on the general settings page of the master, "Auto reestablish connections" is selected by default, so the connection is reestablished when the slave link is restored.

EtherNet/IP remote slave settings

Double-click the EtherNet/IP slave "N-L20" you want to set. On the page displayed, set the IP address of the slave as that of the remote slave.

In the "Electronic Keying" section, the EtherNet/IP device information loaded from the slave EDS file is displayed. During communication connection request, the strict identity check is performed by default, that is, the system checks whether the actual product information and the information of the requested connection are consistent. If not, the connection is failed. During actual application, you should select or modify relevant information for product check.

Connection settings

1. Add a connection.
   The EtherNet/IP remote slave supports connections based on the instance ID path and tag path, among which connections based on the instance ID path can be created through "Add Connection". The EDS description file of an EtherNet/IP remote slave generally contains one or more connection information entries. After the EtherNet/IP network configuration is added, the connection page of the remote slave loads the first connection as the default connection

   On this page, the "Assembly" section displays the default defined parameter names, parameter type, and bit length. In the EDS connection whose parameter name is not defined, you can set the data type and parameter command freely based on the data transmission length.

   You can click "Add Connection" and set the default connection path pre-defined in the EDS file.

2. Edit the connection.

Click "Edit" on the preceding page to enter the connection settings page. Generally, parameters RPI (communication cycle) and number of transmission bytes of the default pre-defined path connection need to be modified based on the application, and other parameters directly use the default values.

The "Edit Connection" page contains main configuration parameters of the connection requested by the EtherNet/IP master. The following describes some important parameters.

- Connection Path: Specifies the format and connection instance of a byte stream frame. For example, 20 04 2C C6 2C 68 (for details, see EIP-CIP-V1-1.0, Appendix C: Data Management).
  20: Logical Segment, ClassID, 8-bit logical address

  04: Assembly Object (04H)

  2C: Logical Segment, Connection Point, 8-bit logical address

  C6: ID-C6H of the Assembly Object instance

  2C: Logical Segment, Connection Point, 8-bit logical address

  68: ID-68H of the Assembly Object instance

  Note: The connection path varies with the vendor and must be configured based on the guide of the specific slave.
- RPI (ms): Requested Packet Interval. It indicates the communication transmission interval in ms. The RPI of each node can be set individually without affecting each other.

Caution

The master RPI period must be an integer multiple of the task period.

- Transmission byte size
  O->T size (bytes): Indicates the amount of data transferred from the producer (initiator) to the consumer (target device), in bytes.

  T->O size (bytes): Indicates the amount of data transferred from the consumer (target device) to the producer (initiator), in bytes.
- Transport Type
  Exclusive Owner: Allows users to set both data sending from the initiator to the target device and data receiving from the target device to the initiator.

  Redundant Owner: Allows multiple initiators to create independent and identical connections to the same target device.

  Input Only: This connection can only be used to set data receiving from the target device to the initiator.

  Listen Only: EtherNet/IP devices apply this type of connections to listen to multicast data without providing configuration or scheduling information.
- Trigger Type
  Cyclic: Periodically triggers data transmission.

  Change-Of-State: Transmits data when a change in the state of the application object is detected.

Application Object: Transmits data when the application object is triggered.

- Connection Type
  Multicast: Multiple scanners receive data from one target device at the same time.

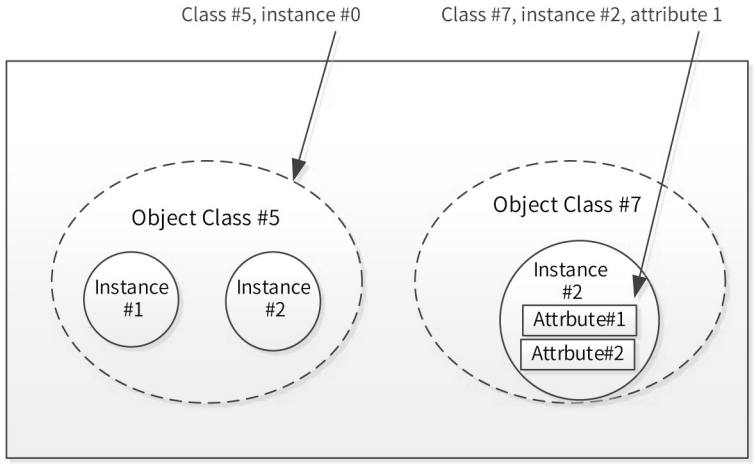
  Point-to-Point: One scanner can receive data from only one target device.

3. Add a generic connection.
   On the "Connection" page, click "Add Device". On the connection settings page, select "Generic connection", as shown in the following figure. The default connection settings are recommended. You can customize a connection path based on the slave specifications based on CIP protocol knowledge. For Inovance PLC slaves, you just need to modify the instance ID of configuration assembly, the dataset instance ID of consuming assembly (O->T), and the dataset instance ID of consuming assembly (T->O).
4. Add a tag connection.
   On the "Connection" page, click "Add Tag Connection" to add a tag path connection. The tag connection type is "Input Only" by default, and the connection only reads the producer tag sent from the consumer slave. You can directly modify the T->O size and connection path (tag name) in the current entry, and modify other connection parameters on the "Edit Connection" page.

Setting user parameters

If you need to set additional EtherNet/IP bus communication parameters required by the slave, you can make configuration through this option with CIP protocol knowledge. In most cases, you do not need to make such configuration. After configuration is completed, each time the communication of the slave is started or restarted, these configured parameters are sent to the master once.

- Name: Indicates the name of the parameter.
- Class: All accessible objects in the network have a unique integer ID, such as classes #5 and #7 in the preceding figure.
- Instance: Indicates the concrete and real (physical) occurrence of an object. For example, the new New Zealand is an instance of the object class "Country", as shown in Instance#1 and Instance#2. (A CIP instance ID value of 0 is used to indicate a reference to a specific instance within the class.)
- Attribute: Indicates the description of an externally visible feature or characteristic of an object. Usually, attributes provide state information or control objects, such as Attribute#1 and Attribute#2 in the preceding figure.



Configuring I/O variable mapping in the program

The following figure shows the EtherNet/IP I/O mapping page of the variable. You can set the input and output variables through address setting. Directly select the address and edit it. You can modify the address automatically allocated by the system to map variables in the program.

Caution

The default value of "Always update variables" is "Enable 2 (always in bus cycle task)" and remains unchanged.

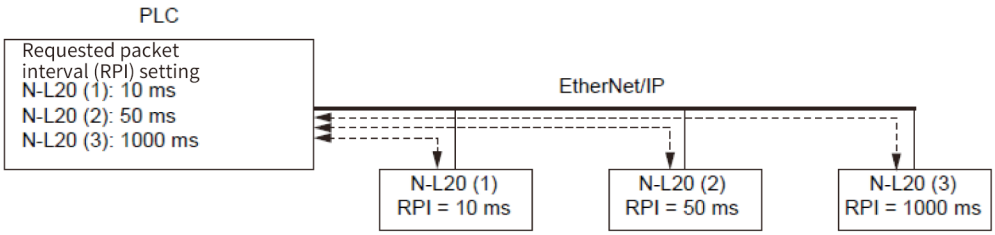
4.11.4 Programming Example for Configuration of PLC as the EtherNet/IP Master

General steps for configuring the EtherNet/IP master project

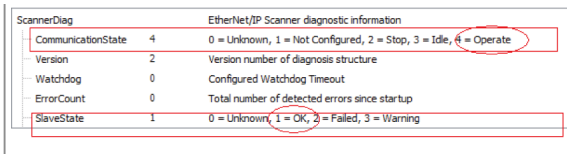
1. Create a project and select the master on the "Network Configuration" page.
2. Import the third-party EDS file and add the EtherNet/IP slave to the networking project.
3. Set the IP address in the EtherNet generic settings. This IP address must be in the same LAN of the IP address set on the slave generic settings page.
4. Add the default pre-defined connection/tag of the slave, set the RPI, task period, and connection parameters.
5. Map parameters in the EtherNet/IP I/O mappings of the remote slave.
6. Compile the user POU program as needed.

Cycle data communication project example of the EtherNet/IP master

In cyclic communication, you can set the RPI (communication cycle) based on the priority of the sent and received data to receive the data after the adjusted communication load, as shown in the following figure.



This instance project uses AC810 as the EtherNet/IP master and KEYENCE NL-20+SR700 scanner as the EtherNet/IP slave. After the configuration is completed, the default connection is established and the program starts to run. When all the devices in the device tree turn to green, the communication connection is established. Besides, in the master state, CommunicationState is 4, indicating the state is OP, SlaveState is 1, indicating the state is OK, and ErrorCount is 0. See the following figure.

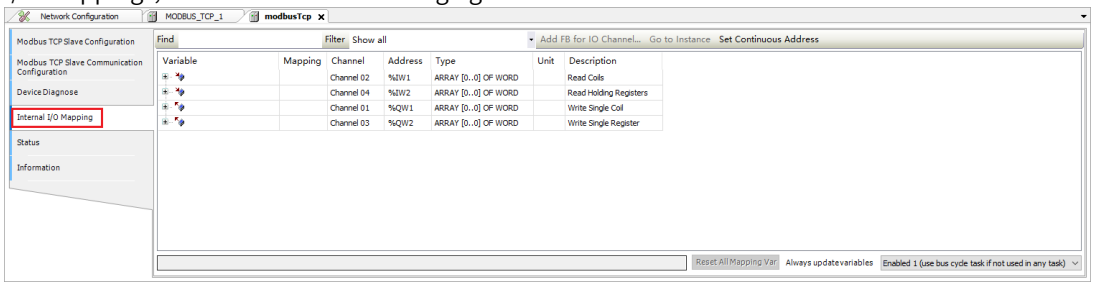


Note

To check whether the master-slave communication in the user program is normal, check whether EtherNetIPMaster.eState is 6 (RUNNING). To check the slave state, check whether the instantiated slave name N_L20.eState is 8 (RUNNING).

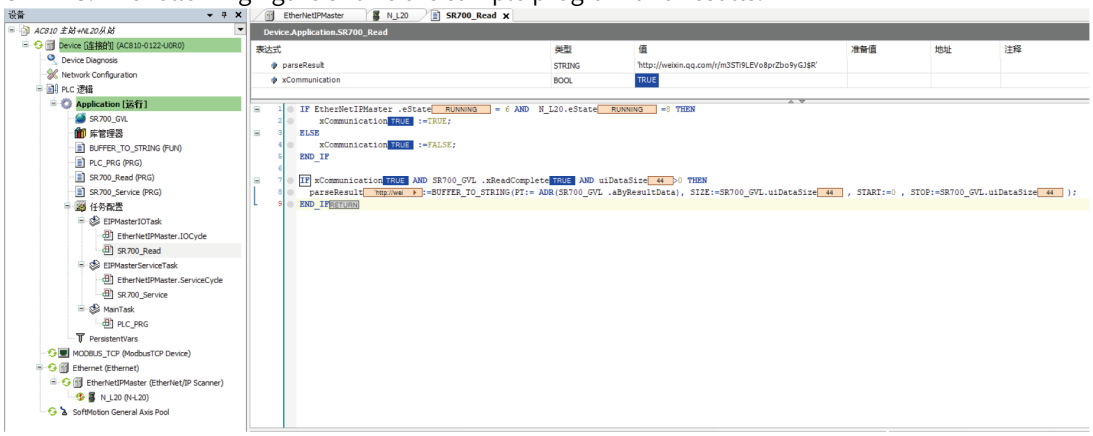
- The test steps of the cyclic communication instance of the KEYENCE NL20 and scanning module are as follows:

  1. Map operation variables of the slave module, and associate global variables with the EtherNet/IP I/O mappings, as shown in the following figure.



  2. During barcode reading, data is written to "Read Data". Then, "Read Complete" changes to "TRUE (1)".
  3. After "Read Complete" changes to "TRUE(1)", set "Read Complete Clear" to "TRUE(1)".
  4. After "Read Complete Clear" is set to "TRUE(1)", "Read Complete" changes to "FALSE(0)"
  5. After "Read Complete" changes to "FALSE(0)", set "Read Complete Clear" to "FALSE(0)".

  The user program judges the communication state in each cycle. When the communication connection is normal and the sensor scanning is completed, the xReadComplete flag is set to "TRUE", the returned scanning data is not 0, and the data results are parsed by BUFFER_TO_STRING. The following figure shows the sample program and results.

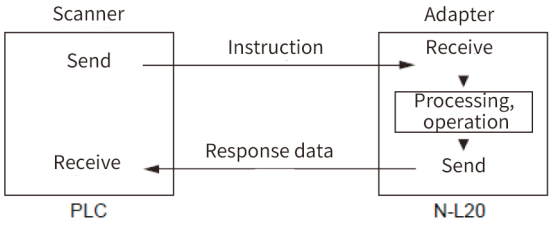


  The following figure shows the QR code strings scanned out.

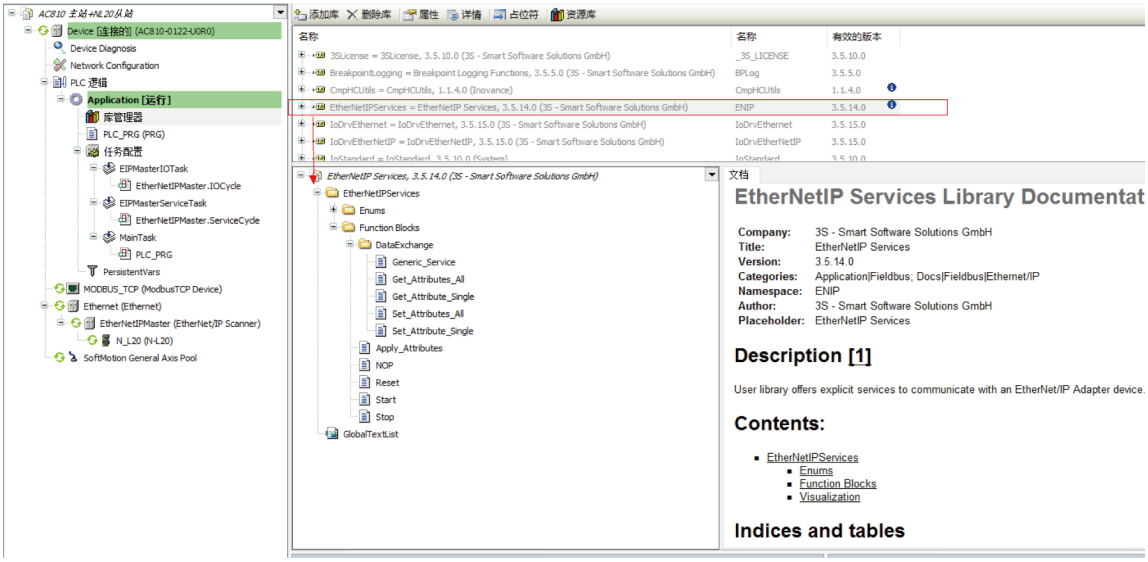
parseResult	STRING	'http://weixin.qq.com/r/m3STi9LEVo8prZbo9yGJ$R'
xCommunication	BOOL	TRUE

Service data communication project example of the EtherNet/IP master

In service message communication, the timing is controlled by command/response, as shown in the following figure.



According to the naming format of service data communication sending command, it is usually necessary to specify "Service code", "Class ID", "Instance", "Attribute ID", and "Service data". This can be realized in the application program by using the EtherNet/IP service function block library EtherNetIPService, as shown in the following figure.



This function block library provides most of common services of the CIP communication protocol (CIP Common Services). The service IDs and names are listed in the following table.

Table 4–1 CIP Common Services IDs and Names

ID	Name
00	Reserved for future use
01	Get_Attributes_All
02	Set_Attributes_All Request
03	Get_Attribute_List
04	Set_Attribute_List
05	Reset
06	Start
07	Stop
08	Create
09	Delete
0A	Multiple Service Packet
0B-0C	Reserved for future use
0D	Apply_Attributes
0E	Get_Attribute_Single
0F	Reserved for future use
10	Set_Attribute_Single
11	Find_Next_Object_Instance

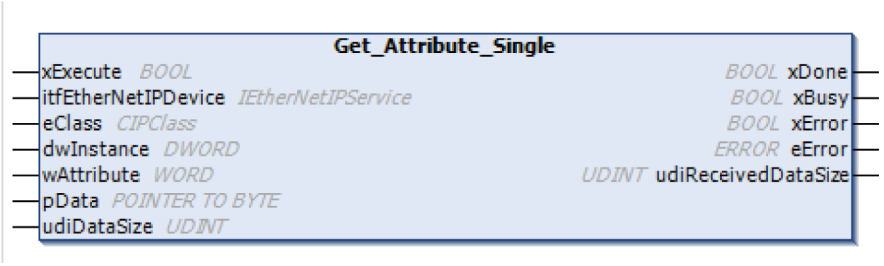
ID	Name
12-13	Reserved for future use
14	Error Response (used by DeviceNet only)

The Get_Attribute_Single and Generic_Service function blocks allow you to get attributes and service data from the user program.

- Get_Attribute_Single
  The KEYENCE NL-20 module guide provides special attribute IDs to get the current state of the reader, as shown in the table below.

Instance ID	Attribute ID	Name	Parameter	
			Qty	Description
1 (0x01)	100 (0x64)	Read Status	UINT	bit0: Error bit1: Result Data Available bit2: Result Data Strobe bit3 to bit5: Reserved bit6: Buffer Overflow Error bit7: General Error bit8: BUSY bit9 to bit10: Reserved bit11: MODE BUSY bit12: ERR BUSY bit14 to bit15: Reserved
			UINT	Bit0: Read Complete
			UINT	Reserved
			UINT	Reserved
1 (0x01)	108 (0x6C)	IN/OUT Status	UINT	bit0: IN1 bit1: IN2 bit2 to bit3: Reserved bit4: OUT1 bit5: OUT2 bit6: OUT3 bit7: OUT4 bit8 to bit15: Reserved
	110 (0x6E)	Result Data Count	UINT	Result Data Ready Count
			UINT	General Error Code
	111 (0x6F)	General Error Code	UINT	General Error Code
	128 (0x80)	Result Data Ready Count	UINT	Result Data Ready Count
	129 (0x81)	Result Data Update Count	UINT	Result Data Update Count

The function block Get_Attribute_Single is used to realize corresponding parameter data in the above table, as shown below.



You can get the returned result count "Result Data Count" (attribute 0x6E) by instantiating the Get_Attribute_Single function block. The example code is as follows. The AutoID Communication Unit Object = 16#69 is not an object in the EtherNet/IP standard, but an object developed by KEYENCE to make N-L20 more user-friendly.

```
PLC_PRG
PROGRAM PLC_PRG
VAR
    getAttributeSingle : ENIP.GET_ATTRIBUTE_SINGLE;
    getAttributeData : ARRAY[1..100] OF UINT;
    xGetAttribute_Trigger : BOOL;

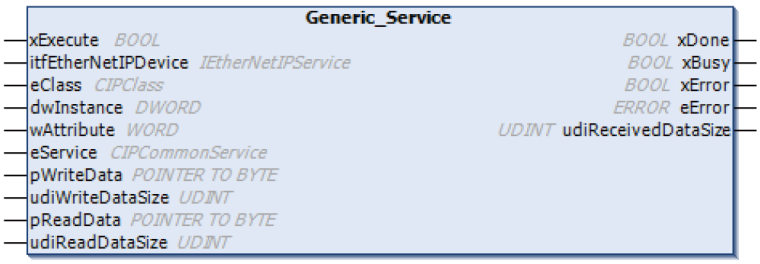
END_VAR
```

```
getAttributeSingle(
    xExecute:= xGetAttribute_Trigger,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    itfEtherNetIPDevice:= NL_20,
    eClass:= 16#69,
    dwInstance:= 1,
    eError=> ,
    wAttribute:= 110,
    pData:= ADR(getAttributeData),
    udiDataSize:= SIZEOF(getAttributeData),
    udiReceivedDataSize=> );
```

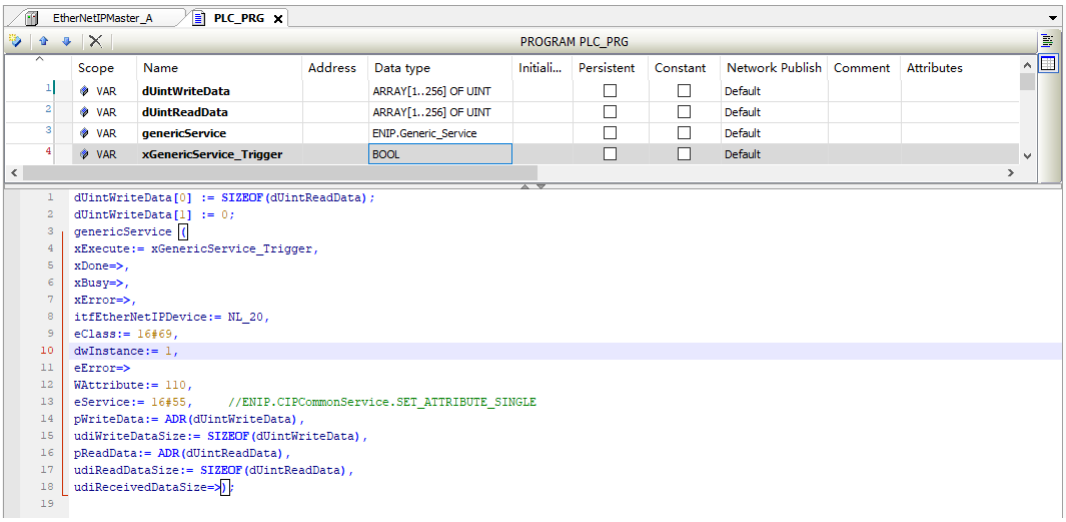
- Generic_Service
  In addition to the common services, there are also special types of services provided by some EtherNet/IP device vendors, such as the KEYENCE NL-20 module, which provides the following service codes for obtaining device-specific interaction data.

Instance ID	Service code	Service Data Data Type: Data	Name	Description
1 (0x01)	14 (0x0E)	-	Get_Attribute _Single	Get an attribute item.
	16 (0x10)	-		Get an attribute item.
	75 (0x4B)	UINT: Bank Number	Read Start	Start to read the data.
	76 (0x4C)	-	Read Stop	The read operation stops.
	83 (0x53)	-	Error Clear	Clear the error.
	85 (0x55)	UINT: Result Data Size UINT: Offset	Get Result Data	Get the read data. Response data UINT: Size of the result data UINT: Size of the remaining data cached to NL-20 BYTE[]: Result data
	86 (0x56)	-	Sequence Reset	Clear the following information: Result Data Ready Count Result Data Update Count Main unit statistical information Buffering data Sequence bit
	90 (0x5A)	-	Read Status Clear	Clear the "Read Complete" and "Read Failed" notifications.

The above service data can be obtained through the function block Generic_Service (function block shown below). The interface data provided by the specified vendor can be obtained by providing the detailed Class ID, instance ID, and service code.



Read the data of the code scanner through the service code 85 (0x55), and the sample code is as follows.

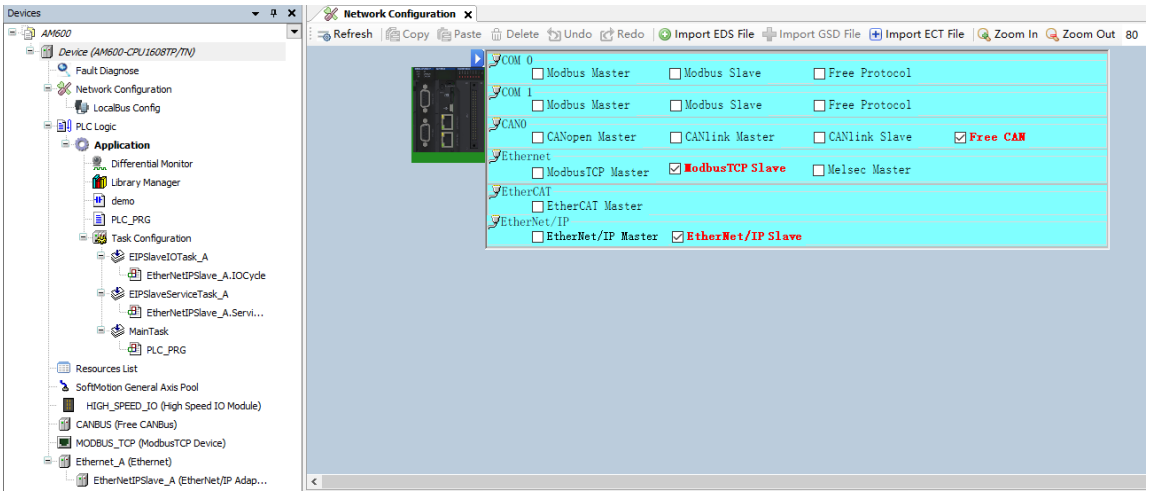


4.11.5 Configuration of PLC as the EtherNet/IP Slave

Inovance medium-sized PLCs support EtherNet/IP local slaves, Class-1 instance ID connection and tag connection response, and Class-3/UCMM service message tag communication (up to 32 tags). The specific configuration of the slave is described below.

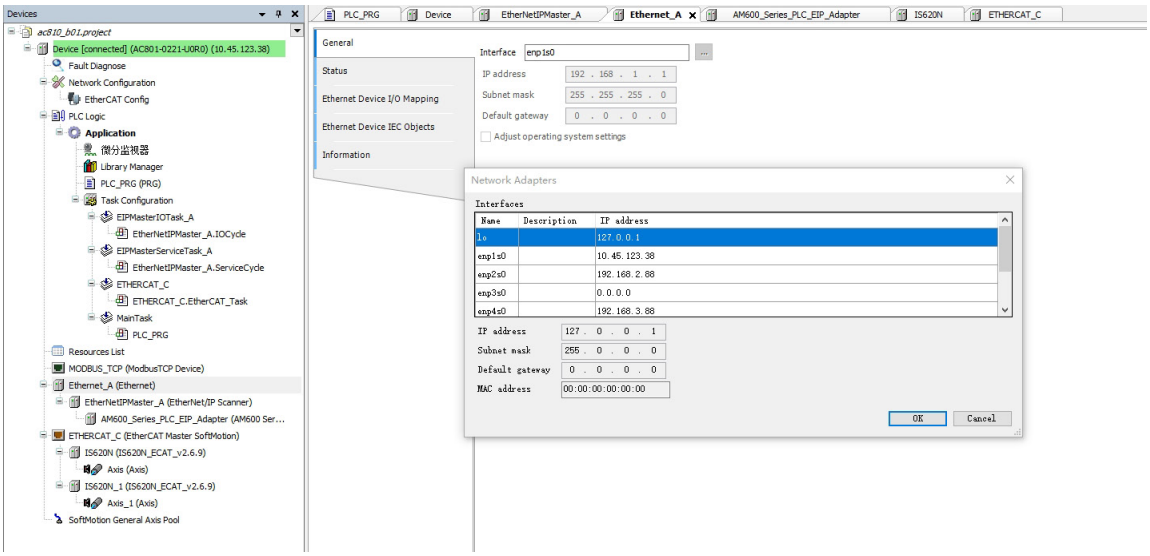
Select "EtherNet/IP Slave".

Click the master. On the page displayed, select "EtherNet/IP Slave".



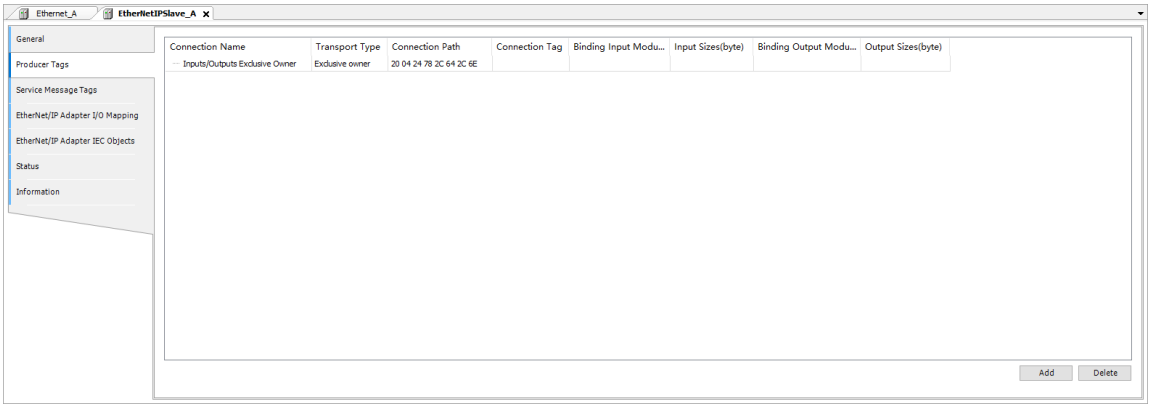
Configure an IP address for the slave.

Log in to the PLC gateway from the software, and double-click the Ethernet module. On the "General" page displayed, set the IP address for the slave. This IP address is the one used to connect the slave to the EtherNet/IP master.

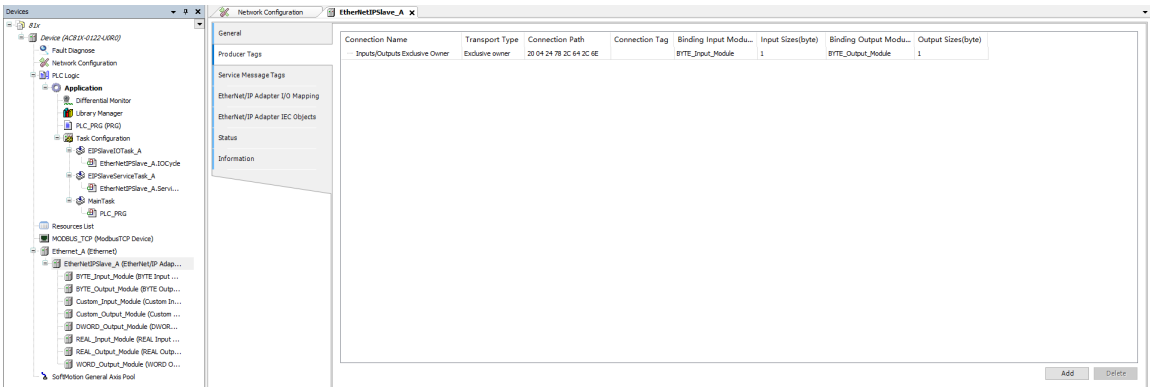


Set the slave producer connection.

The EtherNet/IP slaves support both instance ID path and tag path connections and can respond to connection requests from multiple masters (scanners). As shown in the figure below, there is only one instance ID path connection by default on the "Producer Tags" page, and multiple connections can be added via the "Add" button. For the connection path "20 04 24 78 2C 64 2C 6E", configure the instance ID as 16#78, the input dataset instance ID as 16#64, and the output dataset instance ID as 16#6E.



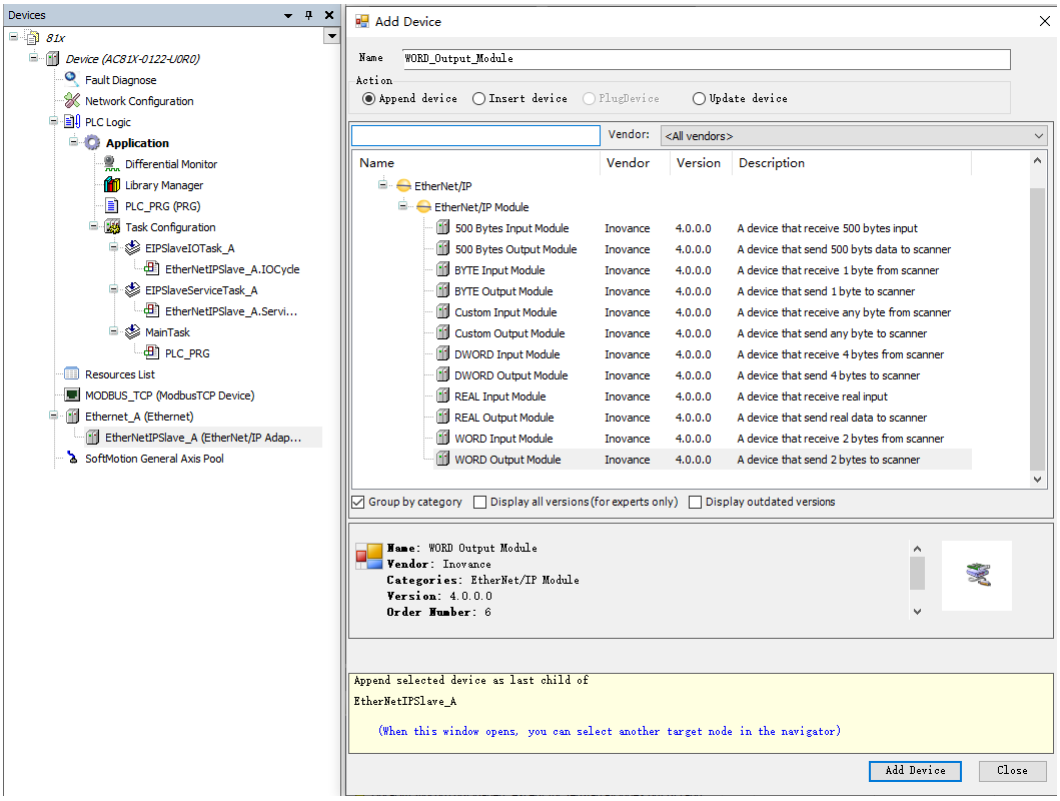
Each connection must be bound to a data input/output module; otherwise, it will not take effect. After the data module is added, this page is automatically refreshed and can display the name of the bound data module and the total size of the data, as shown in the following figure.



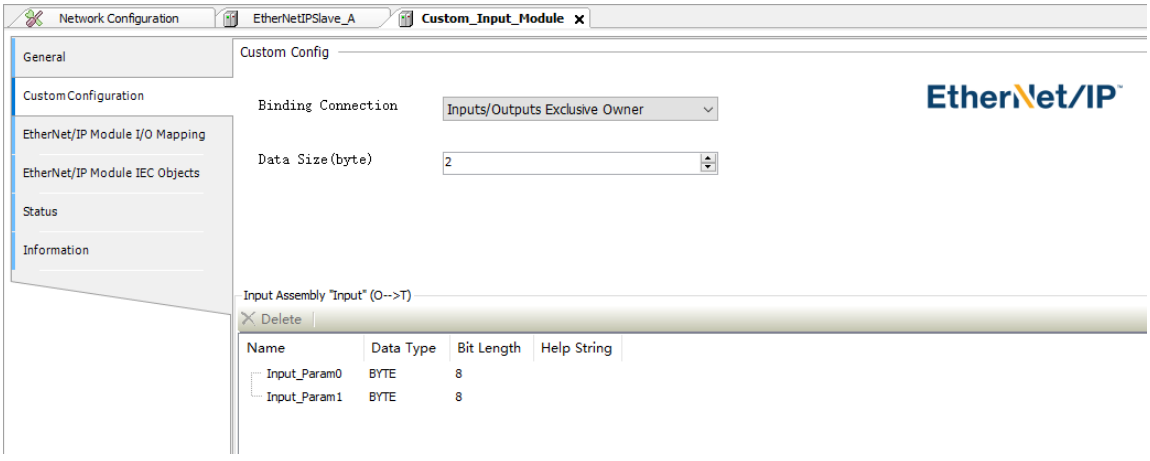
To set the producer tag connection, you can manually enter the producer tag name in the connection path. The default data of the producer tag is output only, so only the output connection can be bound.

Add an input/output module.

Right-click "EtherNetIPSalve". In the shortcut menu displayed, click "Add Device". On the page displayed, add an input/output module for the slave. You can combine the modules according to the need of data parameter transmission. Among the modules, "Custom Input Module" and "Custom Output Module" are user-defined input and output modules. Other data modules are input and output modules with fixed byte size. You can add multiple data modules to set the total data size as required.

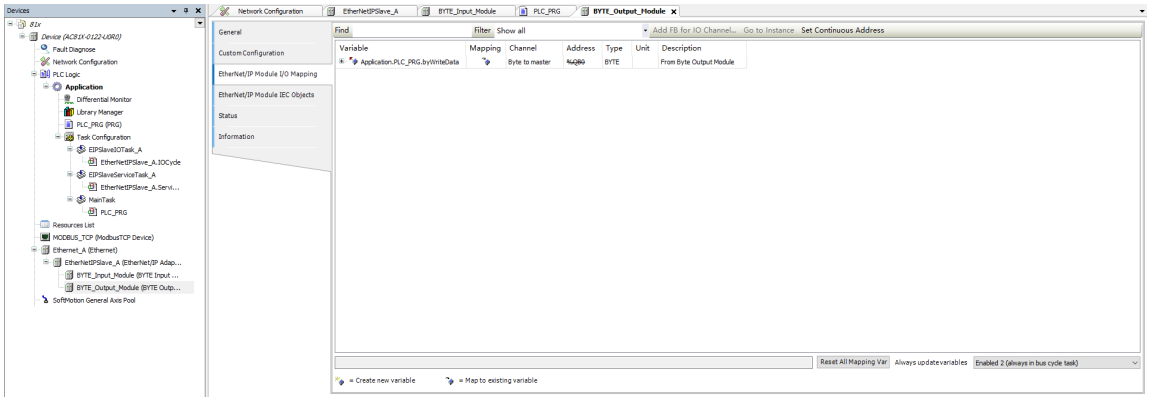


On the "Custom_Input_Module" configuration page, you can freely bind the producer connections and modify the data size bytes, as shown in the following figure. In the "Input Assembly" section, you can customize the parameter name and data type to create user structure parameters flexibly.

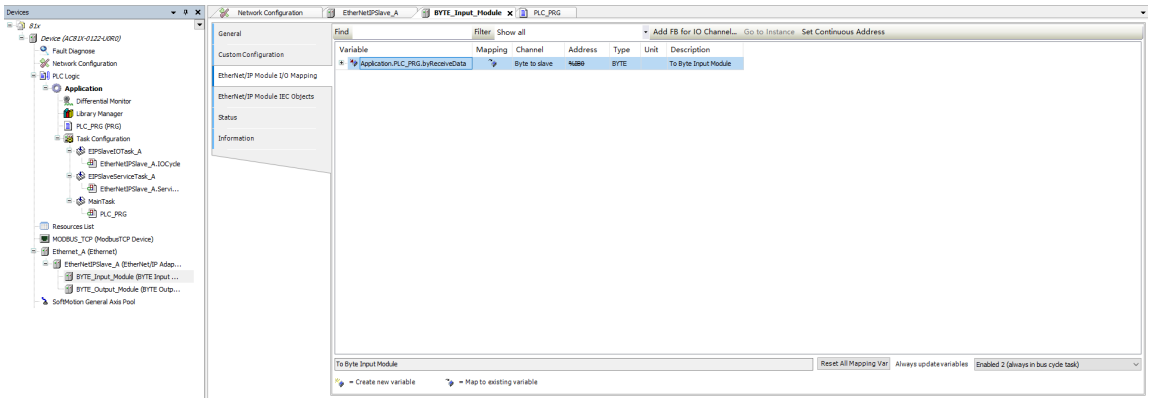


Configure I/O mappings for the data module.

Take "Byte Input Module" and "Byte Output Module" as an example. The PLC automatically assigns the input/output mappings with data type "BYTE". When the EtherNet/IP slave of the PLC receives the data, the mapping variables are refreshed in "InputData". As shown in the following figure, the data type of "byReceiveData" is "BYTE".



When the EtherNet/IP slave of the PLC sends data, then the variable is mapped in "OutputData" to write the data. As shown in the following figure, the data type of "byWriteData" is "BYTE".

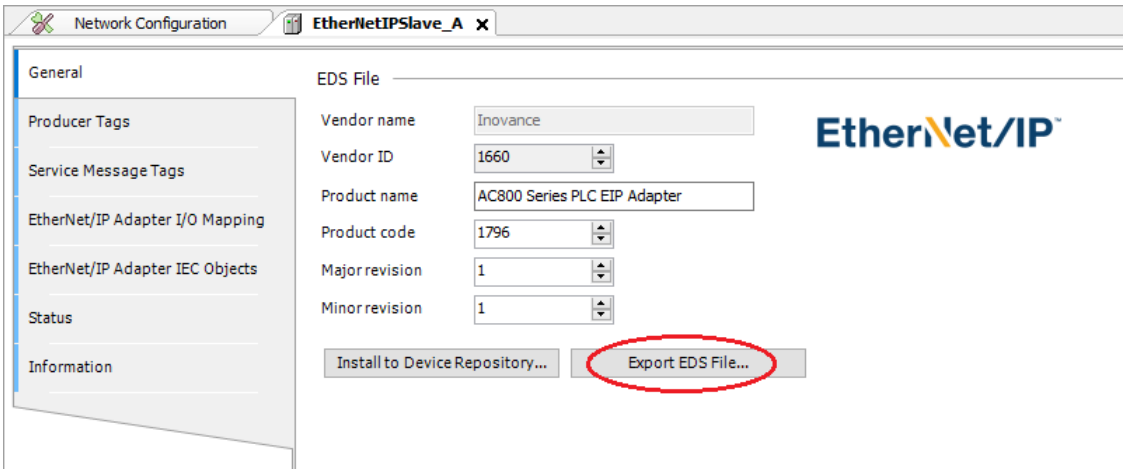


Note

After the slave configuration is selected, the software automatically generates the task configuration. It is recommended to set the task cycle of EIPMasterIOTask in the slave project to be consistent with the RPI, to ensure synchronous data transmission.

Export the EDS file of the slave.

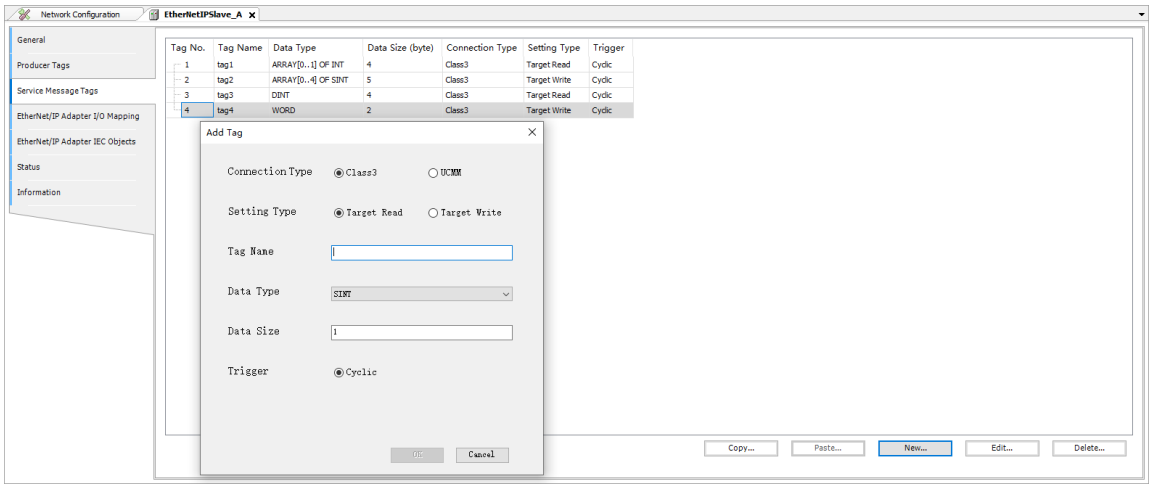
As shown in the following figure, on the "General" page of the EtherNet/IP slave, click "Export EDS File" to export the added connections, input/output datasets, and slave information.



If the export page is grayed, the slave data configuration is not refreshed. In this case, close the slave window and open it again.

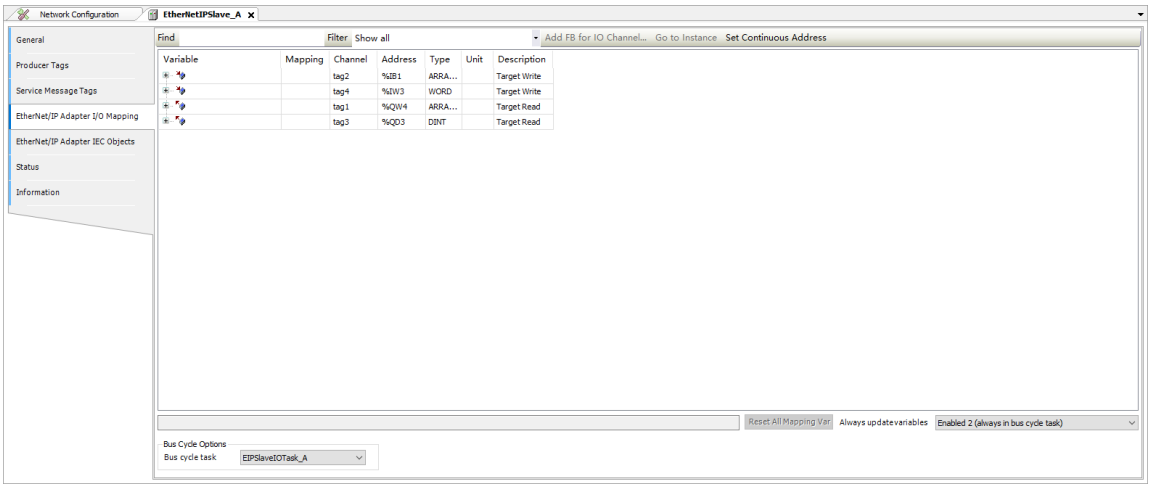
Configure the service message tags.

On the "Service Message Tags" page, you can configure the Class-3/UCMM explicit message communication. After a connection is created, its service tags are displayed, as shown in the following figure.



When "Setting Type" is set to "Target Read", the device can respond to the read data request initiated by the EtherNet/IP master (scanner), and the service code is 4C. When "Setting Type" is set to "Target Write", the device can respond to the write data request initiated by the EtherNet/IP master (scanner), and the service code is 4D. The tag name, data type, and data size must be consistent with those of the initiator; otherwise, a communication error may occur.

The update of read and write data for display message communication is refreshed cyclically in the EtherNet/IP Adapter I/O mapping, and the task refresh period is the period set by the EIPSlaveServiceTask.



EIP configuration of global variable list

Background

Access specifications for tag variables are shown in the following table, and access to the entire structure is not supported.

Data Type Name	Data Type Code	Data Type Description	Number of Supported Array Dimensions	Support to Structure Member Access
BOOL	C1	TRUE/FALSE	3	Yes
SINT	C2	8-bit integer	3	Yes
INT	C3	16-bit integer	3	Yes
DINT	C4	32-bit integer	3	Yes
LINT	C5	64-bit integer	3	Yes
UINT	C7	UInt16	3	Yes
USINT	C6	UInt8	3	Yes
UDINT	C8	UInt32	3	Yes
ULINT	C9	UInt64	3	Yes
REAL	CA	32-bit floating points	3	Yes
LREAL	CB	64-bit floating points	3	Yes
STRING	D0	String	0	Yes
BYTE	D1	8-bit	3	Yes
WORD	D2	16-bit	3	Yes
DWORD	D3	32-bit	3	Yes
LWORD	D4	64-bit	3	Yes

Both slave service message tag configuration and global variable selecting are supported. The difference is shown in the following table.

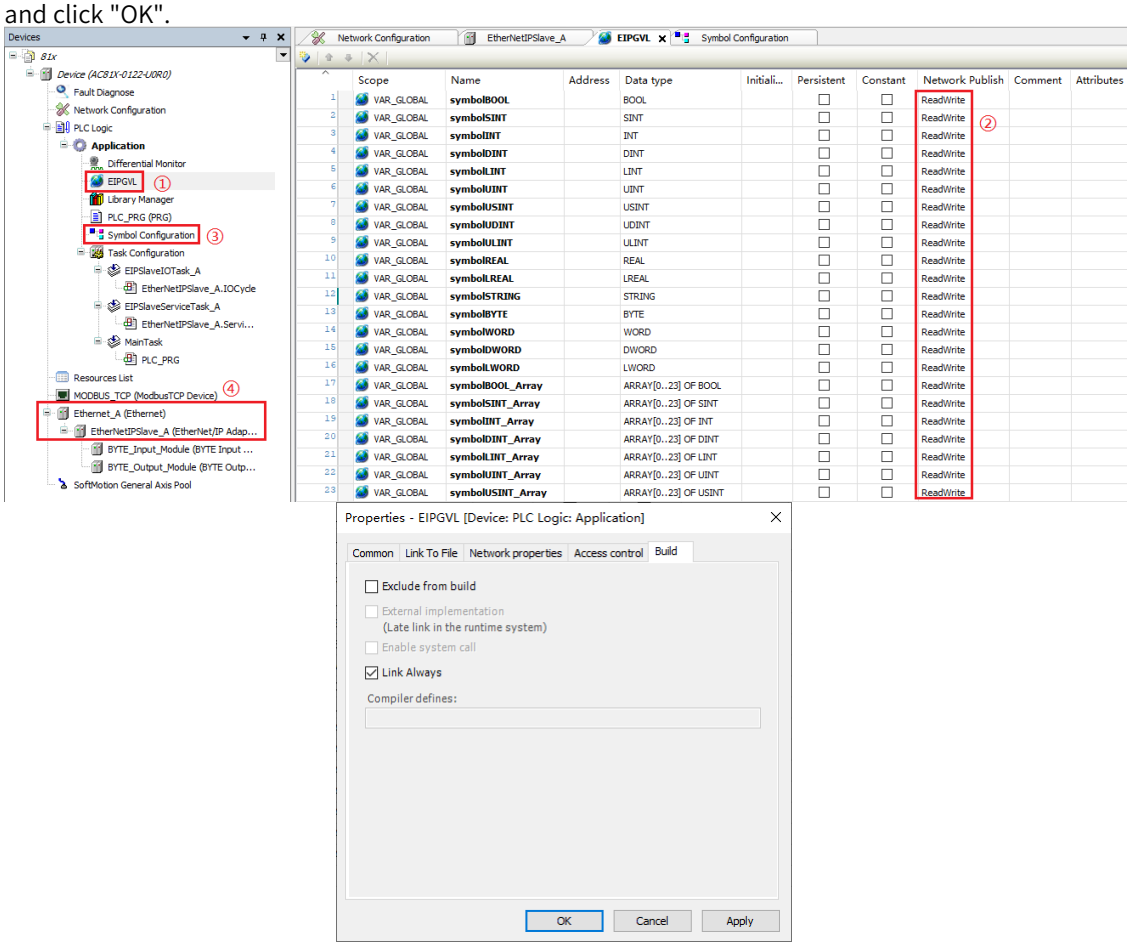
Item	Service Message Tag Configuration	Global Variable List
Access type	Tag name	Application + global variable list name + variable name
Import and export	Not supported	Supported
Data Type	BOOL, STRING, DUT, and multi-dimensional arrays are not supported.	See the table.

Item	Service Message Tag Configuration	Global Variable List
Read-write access	Not supported	Supported
Maximum amount of data for a single access	1400 bytes	1000 bytes
Chinese tag	Not supported	Supported (symbol configuration table, set "Use UTF-8 for character encoding")
Data index	Supported	Supported
Multi-tag service	Supported	Supported

Use case scenarios: HMI, EIP host controller API, PLC, and IM/MES.

Procedure

1. Create a global variable and right-click it. On the "Properties" page displayed, select "Link Always" and click "OK".



2. Set the "Network Public" attribute.

- ReadWrite: The tag variable can be read and written.
- Read-only: The tag variable can only be read.
- Write-only: The tag variable can only be written.

3. Add symbol configuration.

- Method 1: The symbol configuration is automatically added when you compile or download the program.
- Method 2: Manually add the symbol configuration.

4. On the "Network Configuration" page, select "EtherNet/IP Slave" and set the IP address of the EtherNet_A/EtherNet_B network port.

4.11.6 Programming Example for Configuration of PLC as the EtherNet/IP Slave

General steps for configuring the EtherNet/IP slave project

1. Create a project and select the slave on the "Network Configuration" page.
2. Configure the producer connection (Class 1) or explicit service message connection (Class 3/UCMM) based on the application requirements.
3. If the slave is configured with producer connections (instance ID/tag), add input/output modules and bind them to the corresponding connections.
4. Perform variable parameter mapping in the slave input/output modules or in the slave I/O mapping.
5. Compile the user POU program as needed.

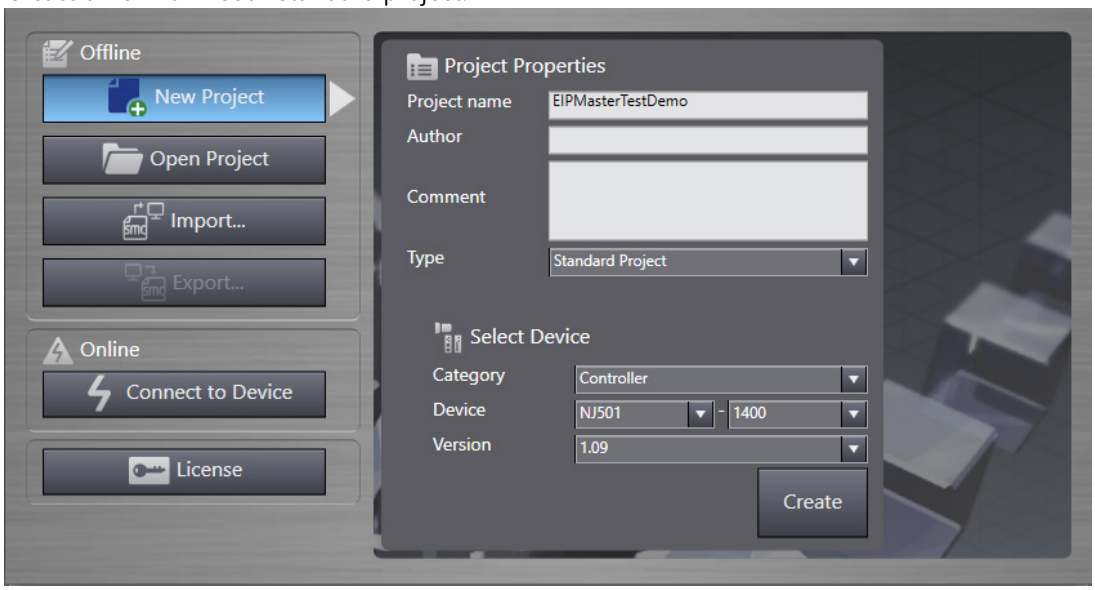
Follow the above EtherNet/IP slave configuration steps to create a slave programming example. This guide tests the communication connection with the AM600 project and the Omron NJ501 EtherNet/IP master project.

Slave project communicating with the Omron NJ501 master

Test the communication between the Omron PLC EtherNet/IP master and the Inovance AM600 slave project.

- NJ501 EtherNet/IP master project configuration

1. Create an Omron NJ501 standard project.



2. Under "Configurations and Setup", select "Built-in EtherNet/IP Port Settings". On the "TCP/IP Settings" window displayed, set the IP address of the master's EtherNet/IP communication port. Ensure that the IP address belongs to the same network segment as that of the Inovance AM600 slave.
3. Create a global variable.

4. Set the connection path.

   a. Import the EDS file of the Inovance slave.In the "Toolbox" section of the "Connection" page, right-click a blank area. In the shortcut menu displayed, select "Display EDS Library (L)". In the displayed EDS library list, click "Install", select the path of the EDS file of the Inovance slave.

   b. Add the slave.After the installation is completed, select "Add Target Device" under the toolbox list. From the drop-down list of "Model name", select the Inovance EtherNet/IP slave imported from the EDS library. Set the IP address and revision of the node connected to the Inovance EtherNet/IP slave (the IP address must be in the same network segment as the Omron PLC master), and then click "Add".

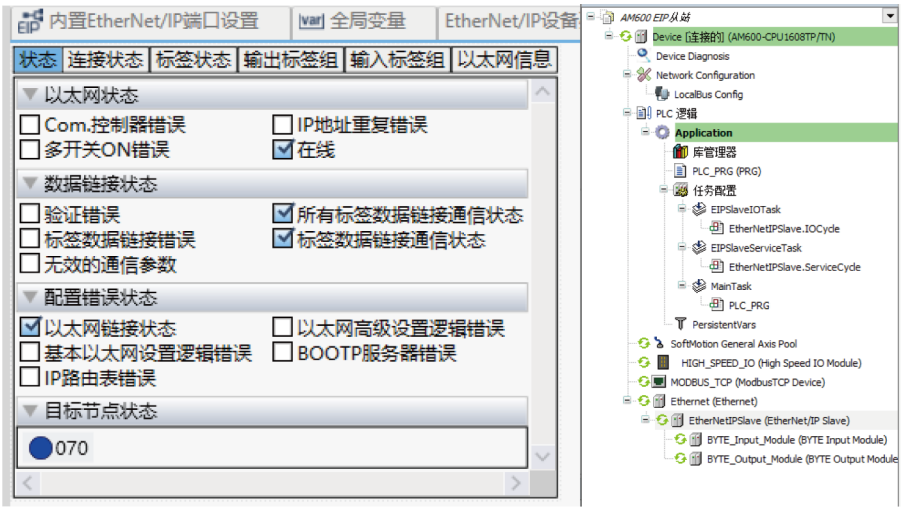
   c. Set the connection.As shown in the following figure, set the target device, connection type, input/output variables, and other parameters.

Note

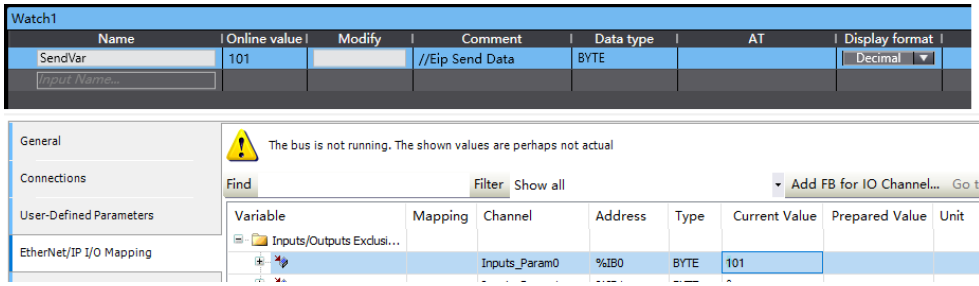
The tag variables input and output 110 and 100 are the instance IDs of the Inovance slave dataset outputs and inputs, respectively.

EtherNet/IP master-slave communication test

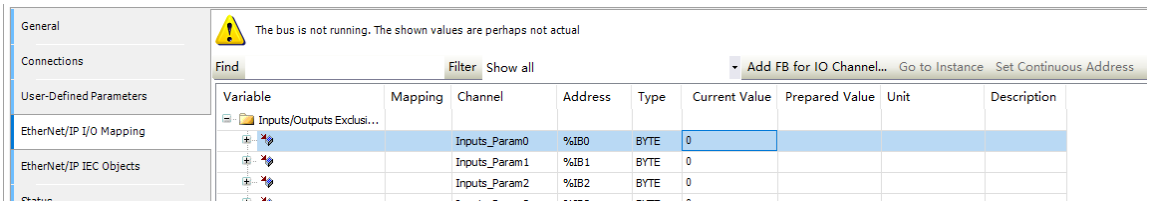
After the above settings are made, compile, synchronize and then run the Omron PLC project, and check whether the communication between the master and slave is normal, as shown in the following figure.



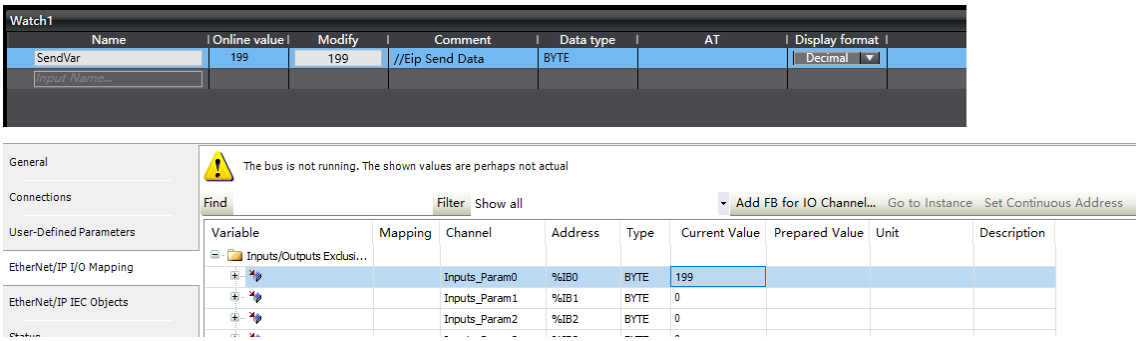
Modify the value of NJ501 EtherNet/IP tag communication SendVar to see the change of AM600 EtherNet/IP slave receive variable. If the variable value can follow the change of the modified value, communication is normal.



Modify the SendVar online value to 199, and the corresponding AM600 mapping variable is also updated to 199.



Modify the "Output Data" of the AM600 EtherNet/IP slave to 52, and the RecvVar value of NJ501 EtherNet/IP communication is also updated to 52.



4.11.7 Diagnosis of EtherNet/IP Communication State

Diagnosis of EtherNet/IP device communication state

Medium-sized PLC EtherNet/IP devices can be used as masters, slaves, or both masters and slaves. After the configuration is completed, log in to the device. If a green icon is displayed in front of the EtherNet/IP device tree, the communication is successful. The state diagnosis in the program can call variables in the protocol library. The communication state of the EtherNet/IP master can be obtained through the program "EtherNetIPMaster.eState", the communication state of the slave can be obtained through the program "EtherNetIPSlave.eState", and the communication state of the remote slave can be obtained through the program "RemoteSlaveName.eState" (for example, AM400_Series_PLC_EIP_Adapter.eState).

The following shows an example program.

```
VAR
    eSlaveState : IodrvethernetIPAdapter.AdapterState;
    eMasterState : Iodrvethernetip.ScannerState;
    eRemoteSlave: Iodrvethernetip.AdapterState;
END_VAR
eSlaveState :=EtherNetIPSlave.eState; //State of the local slave
eMasterState :=EtherNetIPMaster.eState;//State of the local master
eRemoteSlave  := AM400_Series_PLC_EIP_Adapter.eState;//State of the remote slave
//Check whether the local slave state is normal
IF eSlaveState = iodrvethernetIPAdapter.AdapterState.RUNNING THEN
    //TODO ; add the user logic
END_IF

//Check whether the local master state is normal
IF eMasterState =  iodrvethernetip.ScannerState.RUNNING THEN
    //TODO; add the user logic
END_IF

//Check whether the remote slave state is normal
IF  eRemoteSlave = IoDrvEthernetip.AdapterState.RUNNING THEN
    //TODO; add the user logic
END_IF
```

Diagnosis codes of EtherNet/IP log errors

When EtherNet/IP is used as a master, you can add slaves (remote slaves) to the master. On both master configuration page and slave configuration page, the "Device Diagnosis" option is available. The diagnosis information of the master is used to indicate that the configuration item is faulty, and does not contain the specific cause of the fault, so there is no fault code on the "Device Diagnosis" page.

In case of communication sending connection errors, you can find the corresponding device diagnosis information for the EtherNet/IP component in logs on the software. The diagnosis codes and diagnosis information for EtherNet/IP as master or slave are the same. See the table below for EtherNet/IP diagnosis codes.

Diagnosis Information ID	Status Code	Description
SUCCESS	0	The communication is successful.
DUPLICATE_FWD_OPEN	16#100	The connection request is repeated.
TRANSPORTCLASSTRIGGER_NOT_SUPPORTED	16#103	The transmission type and trigger type are not supported.
OWNERSHIP_CONFLICT	16#106	A master conflict occurs.
TARGET_CONNECTION_NOT_FOUND	16#107	The connection to the target slave is not found.
INVALID_NETWORK_CONNECTION_PARAMETER	16#108	The network connection parameter is invalid.
INVALID_CONNECTION_SIZE	16#109	The T->O or /O->T size is invalid.

Diagnosis Information ID	Status Code	Description
TARGET_FOR_CONNECTION_NOT_CONFIGURED	16#110	The target slave for connection is not configured.
RPI_NOT_SUPPORTED	16#111	The RPI is not supported.
RPI_NOT_ACCEPTABLE	16#112	The RPI value is not acceptable.
OUT_OF_CONNECTIONS	16#113	The connection type is out of the range.
VENDORID_OR_PRODUCT_CODE_MISSMATCH	16#114	The vendor/product ID does not match.
PRODUCT_TYPE_MISSMATCH	16#115	The product type does not match.
REVISION_MISSMATCH	16#116	The software version does not match.
INVALID_PATH	16#117	The path is invalid.
INVALID_CONFIGURATION_PATH	16#118	The configuration path is invalid.
NON_LISTEN_ONLY_CONNECTION_NOT_OPEND	16#119	The non-Listen connection path is not available.
TARGET_OBJECT_OUT_OF_CONNECTIONS	16#11A	The maximum number of connections to the target device exceeds the limit.
RPI_SMALLER_THAN_PRODUCTION_INHIBIT_TIME	16#11B	The RPI value is smaller than the producer inhibit time.
TRANSPORT_CLASS_NOT_SUPPORTED	16#11C	The transmission type or trigger parameter is not supported.
PRODUCTION_TRIGGER_NOT_SUPPORTED	16#11D	The trigger type is not supported.
DIRECTION_NOT_SUPPORTED	16#11E	The transmission direction is not supported.
INVALID_OT_FIXVAR_VALUE	16#11F	The O->T network connection parameter fixing/ variable identifier is invalid.
INVALID_TO_FIXVAR_VALUE	16#120	The T->O network connection parameter fixing/ variable identifier is invalid.
INVALID_OT_PRIORITY	16#121	The O->T priority is invalid.
INVALID_TO_PRIORITY	16#122	The T->O priority is invalid.
INVALID_OT_CONNECTION_TYPE	16#123	The O->T connection type is invalid.
INVALID_TO_CONNECTION_TYPE	16#124	The T->O connection type is invalid.
INVALID_OT_REDUNDANT_OWNER_FLAG	16#125	The redundant owner flag is invalid.
INVALID_CONFIGURATION_SIZE	16#126	The configuration size is invalid.
INVALID_OT_SIZE	16#127	The O->T size is invalid.
INVALID_TO_SIZE	16#128	The T->O size is invalid.
INVALID_CONFIG_APPL_PATH	16#129	The configuration path is invalid.
INVALID_CONSUMING_APPL_PATH	16#12A	The consumer path is invalid.
INVALID_PRODUCING_APPL_PATH	16#12B	The producer path is invalid.
CONFIG_SYMBOL_DOES_NOT_EXIST	16#12C	The configuration tag does not exist.
CONSUMING_SYMBOL_DOES_NOT_EXIST	16#12D	The consumer tag does not exist.
PRODUCING_SYMBOL_DOES_NOT_EXIST	16#12E	The producer tag does not exist.
INCONSISTENT_APPL_PATH_COMBINATION	16#12F	The configuration/producer/consumer paths are inconsistent.
INCONSISTENT_CONSUME_DATA_FORMAT	16#130	The consumer data formats are inconsistent.
INCONSISTENT_PRODUCE_DATA_FORMAT	16#131	The producer data formats are inconsistent.

Diagnosis Information ID	Status Code	Description
NULL_FWDOPEN_NOT_SUPPORTED	16#132	The connection request of the null configuration is not supported.
CONNECTION_TIMEOUT_ MULTIPLIER_NOT_ACCEPTABLE	16#133	The connection timeout multiplier is not acceptable.
MISMATCHED_TO_CONNECTION_ SIZE	16#134	The connection data size does not match.
MISMATCHED_TO_CONNECTION_ FIXVAR	16#135	The connection parameter fixing/variable tag does not match.
MISMATCHED_TO_CONNECTION_ PRIORITY	16#136	The priority of the connection parameter does not match.
MISMATCHED_TRANSPORT_CLASS	16#137	The transmission type does not match.
CONNECTION_TIMED_OUT	16#203	Connection times out.
UNCONNECTED_REQUEST_TIMED_ OUT	16#204	The unconnected request times out.
PARAM_ERROR_IN_UNCONNECTED_ REQUEST	16#205	The unconnected request parameter is incorrect.
MESSAGE_TOO_LARGE	16#206	The message length is too large.
UNCONNECTED_ACK_WITHOUT_ REPLY	16#207	The unconnected acknowledge is not responded.
NO_BUFFER_MEMORY_AVAILABLE	16#301	No buffer space is available.
NETWORK_BANDWIDTH_NOT_ AVAILABLE	16#302	The network bandwidth is insufficient.
PORT_NOT_AVAILABLE	16#311	The port is unavailable.
LINK_ADDRESS_NOT_VALID	16#312	The link address of the port data segment is invalid.
INVALID_SEGMENT_IN_ CONNECTION_PATH	16#315	The connection path contains an invalid data segment.

4.12 PROFIBUS-DP Bus

4.12.1 Overview

Bus overview

PROFIBUS is an international and open fieldbus standard independent of device manufacturers. PROFIBUS is widely used by automation in the manufacturing, process, building, traffic, and electricity sectors. It becomes a European industrial standard in 1996, became an international standard in 1999, and was approved to be the fieldbus standard of industrial automation in the People's Republic of China in 2001.

PROFIBUS adopts existing international standards and is based on the Open Systems Interconnection (OSI) model, as shown in Figure 3-82. Therefore, PROFIBUS meets the open and standard requirements. In the OSI models, predefined tasks are implemented accurately during transmission. The physical layer (first layer) defines the physical transmission features. The data link layer (second layer) defines the bus access protocol. The application layer (seventh layer) defines application functions.

PROFIBUS-DP uses the first layer, second layer, and user interfaces. The third to seventh layers are not described here. This flow structure ensures fast and effective data transmission. The direct data link mapper (DDLM) provides a service user interface for easy access to the second layer. The user interface defines the application functions called by users, the system, and different devices, describes the behaviors of different PROFIBUS-DP devices, and provides the RS485 transmission technique or optical fibers.

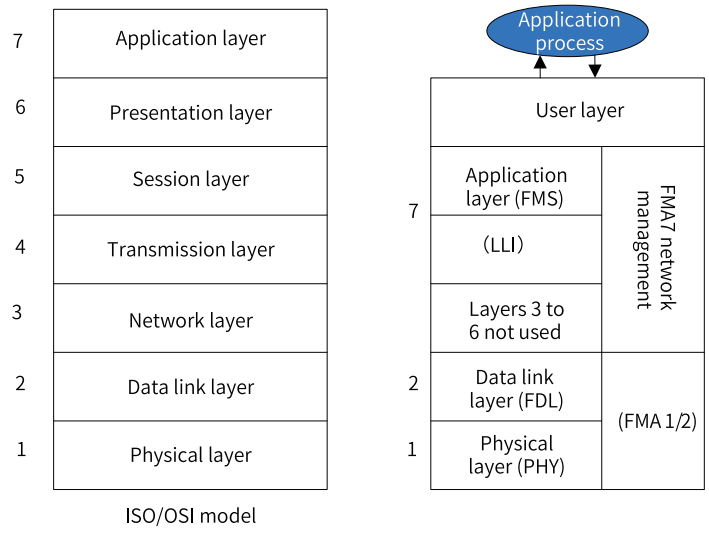


Figure 4-60 PROFIBUS-DP fieldbus model

PROFIBUS-DP is used for field high-speed data transmission. The master reads the input information of slaves and sends output information to slaves periodically. In addition to periodic transmission of user data, PROFIBUS-DP also provides aperiodic communication required by smart field devices for configuration, diagnosis, and alarm handling.

4.12.2 General Process of Using PROFIBUS-DP

The general process of using PROFIBUS-DP is as follows:

1. Design the hardware network structure of PROFIBUS-DP.
2. Activate the PROFIBUS-DP bus in network configuration. The PROFIBUS-DP master is automatically added after the bus is activated.
3. On the "Network Configuration" page, add PROFIBUS-DP slaves and modules based on the hardware structure. Before adding a third-party slave, import a GSD file to import the third-party slave on the "Network Configuration" page. Before adding an AM600 slave, add an I/O module in hardware configuration. A PROFIBUS-DP slave is a remote DP device.
4. Set the master parameters, slave parameters, and module parameters properly. In normal cases, the slave node ID is automatically generated, I/O mapping is automatically generated based on the GSD file, and some special settings need to be modified manually.

When setting the parameters of the master and slave, ensure that the baud rates of the master and slave are adaptive and that the configured slave node ID matches with the DIP switch setting of the actual slave node ID.

4.12.3 PROFIBUS-DP Master Configuration

Master parameter configuration

General

- Node ID: The unique identifier of the master in the PROFIBUS-DP network. The default value is 1, and the value range is 1 to 126, in the decimal format.
- Highest station address: The maximum station address for token transfer. The default value is 126.
- Watch Dog Control: The watch dog time transferred to the slave, used to determine the master-slave connection.

Bus Parameters

Baud rate: The baud rate of transmission along the bus. The unit is kbps. Options are 9.6, 19.2, 45.45, 93.75, 187.5, 500, 1500, 3000, 6000, and 12000. The default value is 1500.

Note

Set the baud rate properly based on different communication distances and the number of communicating stations. The PROFIBUS subnet works properly only with matching parameters of the bus configuration file. You can change the default settings only if you are familiar with the parameter allocation of the PROFIBUS bus configuration file. It is recommended that the default bus parameter settings be used.

Stopped on failure

The "Stopped on failure" function determines whether to stop slave operation when a slave or module is faulty or the configuration is inconsistent. This function is only applicable to AM600 PROFIBUS-DP slaves.

- Setting list on slaves failure: You can view and set whether to stop operation upon slave failure or inconsistent configuration.
  In the "Stopped On Failure" column, you can set whether to stop slave operation when the specified slave or module is faulty. If the check box under "Stopped On Failure" is selected, the slave stops running when it is faulty or when the I/O module with the diagnosis and report function enabled is faulty.
- Click "OK" or "Cancel" to save or cancel the settings on the "Stop Setting on Failure" page.

PROFIBUS-DP master I/O mapping

For the general description of I/O mapping and instructions on this page, see "I/O mapping."

Status

The state configuration editor for the PROFIBUS-DP bus devices or modules displays state information (such as "Running" and "Stopped") and the state of the internal bus system.

Information

The following basic information about the currently available device is displayed: Name, Vendor, Categories, Version, Module Number, and Description.

4.12.4 PROFIBUS-DP Slave Configuration

To configure a PROFIBUS-DP slave, you mainly need to set the basic slave parameters and the slave parameters declared in GSD.

Slave parameter setting

General

- Node ID: The unique identifier of a slave in the PROFIBUS-DP network. The value ranges from 1 to 125, in the decimal format. The default value is 2. The node ID must be consistent with the slave identifier (such as the DIP switch).
- ID number: The unique identifier of the slave, which is determined by the GSD file.
- T_SDR: The minimum response interval of the slave, that is, the minimum interval for the slave to return a response after receiving data from the master.
- Lock/Unlock: The current state of the slave.

User parameters

User parameters are defined by the GSD file and can be displayed in the decimal or hexadecimal format. For details, see the guide provided by the device vendor.

Slave diagnosis

The diagnosis function indicates the running state of slave nodes.

Diagnosis explanation:

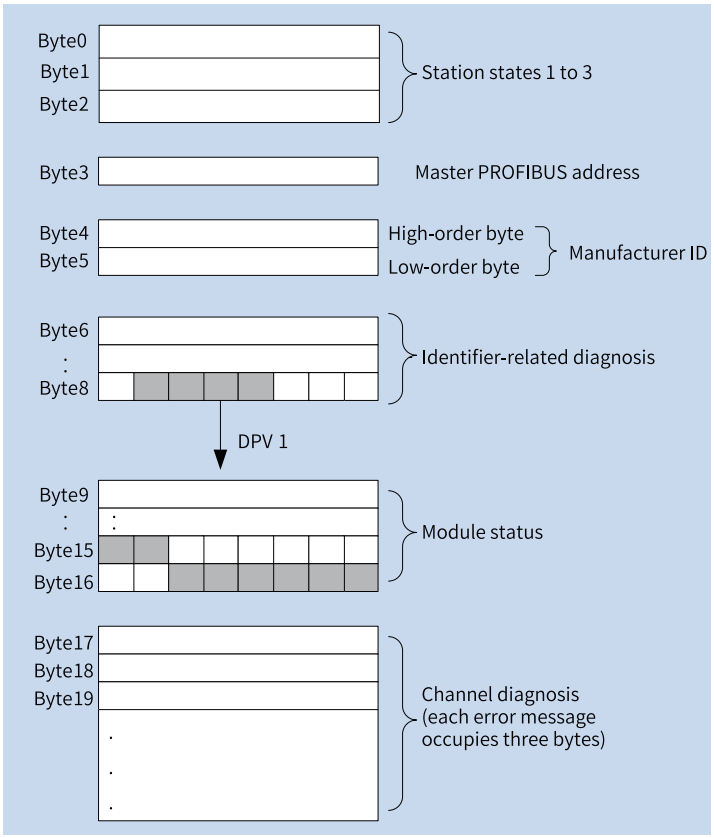


Figure 4-61 Diagnosis message structure

Table 4–2 Meaning of station state 1

Bit	Meaning		Cause and Solution
0	0:	The DP master cannot address the DP slave, and this bit of the DP slave is always 0.	Check whether the PROFIBUS address of the DP slave is correct. Check whether the bus connector and FOC are connected. Check the voltage of the DP slave. Check whether the settings of the RS485 relay are correct. Check whether the DP slave is reset (enabled or disabled).
1	1:	The DP slave is not ready for data exchange.	Wait until the DP slave is started.
2	1:	The configuration data that the DP master sends to the DP slave does not match with the actual configuration of the DP slave.	Check whether the station type entered in the configuration software or the DP slave configuration is correct.
3	1:	External diagnosis is available.	Evaluate the identifier-related diagnosis, module state, and/or channel-related diagnosis. Bit 3 is reset after all the errors are fixed. This bit is reset when a new diagnosis message is generated in the preceding diagnosis bytes.
4	1:	The DP slave does not support the requested function.	Check the configuration.
5	1:	The DP master fails to parse the response of the DP slave.	Check the bus configuration.
6	1:	The DP slave type does not match with the software configuration.	Check whether the configuration software of the station type is configured correctly.
7	1:	Other DP masters (not the DP master currently accessing the DP slave) have been configured for the DP slave.	This bit is always 1 when the DP slave is accessed through a programming device or other DP masters. The PROFIBUS address of the DP master configured for the DP slave is located in the "master PROFIBUS address" diagnosis byte.

Table 4–3 Meaning of station state 2

Bit	Meaning	
0	1:	The DP slave must be reconfigured.
1	1:	The slave is in the startup stage.
2	1:	This bit of the DP slave is always 1.
3	1:	Response monitoring is enabled for the DP slave.
4	1:	The DP slave has received the "FREEZE" control command.
5	1:	The DP slave has received the "SYNC" control command.
6	0:	This bit is always 0.
7	1:	Activation of the DP slave is canceled. That is, the DP slave is removed from current processing.

Table 4–4 Meaning of station state 3

Bit	Meaning	
0 to 6	0:	This bit is always 0.
7	1:	The number of channel-specific diagnosis messages exceeds the number of messages allowed by diagnosis frames.

- The master PROFIBUS address indicates that the PROFIBUS-DP master has been configured for the DP slave and has the read and write permissions on the slave. If the value is FF, the DP master is not configured for the DP slave.
- The manufacturer ID indicates the DP slave type. It is declared by the device manufacturer and is reflected in GSD.

4.12.5 PROFIBUS-DP Module

Modular device and non-modular device

In PROFIBUS-DP slave configuration, you can connect PROFIBUS-DP slave nodes to the following two types of modular devices:

Modular device: It is connected to a DP slave node and provides an I/O mapping list. The "PROFIBUS-DP Slave I/O Mapping" dialog box is not required. The data of slave nodes increases as modules are added. Currently, the AM600 I/O module is a type of modular device.

Non-modular device: The slave node dialog box includes the I/O mapping dialog box. Data cannot be configured automatically.

AM600 PROFIBUS-DP I/O module

Add the AM600 PROFIBUS-DP I/O module in hardware configuration, and set related I/O parameters and add I/O mappings to refresh data. For details, see "CPU > I/O module".

4.13 HMI Communication Configuration

4.13.1 Communication Configuration

This drive member reads and writes the data of various registers of Inovance medium-sized PLCs based on the Modbus TCP/IP protocol through the InoTouch Editor software configuration.

The HMI supports the 01, 31, 03, 33, 05, 35, 06, 36, 0F, 3F, 10, and 40 parameters. For details about the parameters, see the HMI user guide.

Configuration Item	Description
Communication protocol	The Modbus TCP/IP protocol of the Inovance AM600-series PLC is used.
Communication mode	One master and one slave; one master and multiple slaves. The drive member is the master, and devices are the slaves.
Communication device	Ethernet sub-devices must be attached to the general TCP/IP parent devices to work properly.

Hardware connection

Ensure that hardware is connected correctly before configuring communication between the InoTouch Editor software and devices.

Connection method: Use an RJ45 network cable to directly connect the HMI to the PLC (use a straight-through network cable or hub switch).

Device communication parameters

Set the client communication parameters of InoTouch Editor.

When communication setup is in progress, select Ethernet and add a device.

- PLC IP addr.: Enter the IP address of the PLC based on the actual setting.
- Port No.: The port used to send and receive data frames by the host computer and slave computer. The default value is 502. It is recommended that the default value be used. Retain the default settings of other parameters.

Communication configuration

Set the parameters of the Inovance AM600-ModbusTCP sub-device.

Response delay: The interval between frame sending and reception startup. The default value is 0 ms.

Collection channel

Communication state

Communication State Value	Description
Communication normal	The current communication is normal.
Command failed	The read and write commands of the device fail to be executed.
Check failed	An error occurred while checking collected data.
Communication timeout	No collected data is returned.

Internal attributes

You can add a channel by using internal attributes. This drive member supports the Modbus TCP registers of Inovance AM600-series PLC. The following table lists the parameters.

Register	Data Type	Read Parameter	Write Parameter	Operation	Channel Example
[SM area] input coil	BT	31	35 and 3F	Read-write	"SM read-only 0000" indicates SM area address 0.
[Q area] output coil	BT	01	05 and 0F	Read-write	"Q read-write 0001" indicates Q area address 1.

Register	Data Type	Read Parameter	Write Parameter	Operation	Channel Example
[SD area] input register	16-bit-BCD 32-bit-BCD 16-bit-unsigned 16-bit-signed 32-bit-unsigned 32-bit-unsigned 32-bit-float	33	36 and 40	Read-write	"SD read-only 0002" indicates SD area address 2.
[M area] Output register	16-bit-BCD 32-bit-BCD 16-bit-unsigned 16-bit-signed 32-bit-unsigned 32-bit-unsigned 32-bit-float	03	06 and 10	Read-write	"M read and write 0003" indicates M area address 3.

Function code: [SD area] uses the function code 40 when double-word (32-bit) data is written or multiple data records are written in batch.

[M area] uses the function code 10 when double-word (32-bit) data is written or multiple data records are written in batch.

Note

When a channel is added by using internal attributes, the start address is 0 (protocol address), which complies with the Modbus TCP protocol of Inovance AM600.

4.13.2 Communication Example

Bit variable read and write

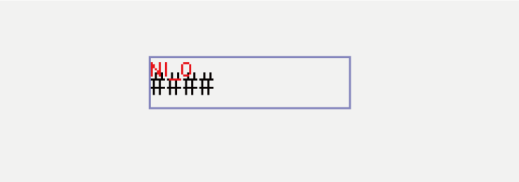
A bit status indicator and a bit status switch are provided, as shown in the following figure.



Modify the general attributes of the bit status indicator. Set the address to Q_bit (0:0).

Word variable read and write

Select the numeric value input control, as shown in the following figure.



Modify the general attributes of the numeric value input control. Set the address to MW0.

4.13.3 Common Faults

Symptom	Analysis	Solution
Communication timeout	An error occurs during collection initialization. No collected data is returned. (Incorrect communication hardware connection and parameter settings)	1. Check whether the parameter settings of network devices are correct.
		2. Check whether the serial port is occupied by other programs.
		3. Check whether the communication cable is connected correctly.
		4. Check whether the data read address exceeds the specified range.
Command failed	The read and write operations failed.	1. Check whether the data read address exceeds the specified range.
		2. Check whether the communication cable is too long, and perform short-distance test.
		3. Check whether onsite interferences are excessive, and prevent interferences in the surrounding environment.

Registers and parameters supported by the drive member

Register	Read Parameter	Write Parameter	Parameter Description
[SM area] input coil	31	35 3F	31: Read the input coil status 35: Enforce a single input coil 3F: Enforce multiple input coils
[Q area] output coil	01	05 0F	01: Read the output coil status 05: Enforce a single output coil 0F: Enforce multiple output coils
[SD area] input register	33	36 40	33: Read the input register 36: Pre-configure a single register 40: Pre-configure multiple registers
[M area] Output register	03	06 10	03: Read the holding register 06: Pre-configure a single register 10: Pre-configure multiple registers

Note:

1. This drive member supports the 01, 31, 03, 33, 05, 35, 06, 36, 0F, 3F, 10, and 40 parameters. Other parameters not used by data communication are not supported.
2. The preceding parameters adopt the hexadecimal format. Parameters 0x0F and 0x10 correspond to 15 and 16 in the decimal format.

3. The [SM area] input coil uses 3F when writing multiple relays in batch.
4. The [Q area] output coil uses 0F when writing multiple relays in batch
5. The [SD area] output register uses 40 when double-word (32-bit) data is written or multiple data records are written in batch.
6. The [M area] output register uses 10 when double-word (32-bit) data is written or multiple data records are written in batch.

Note

When a register channel is added, the start address is 0 (protocol address), which complies with the Modbus protocol of Inovance AM600.

Data type table

BTdd	Bit (dd range: 00 to 15)
16-bit-BCD	16-bit-BCD
32-bit-BCD	32-bit-BCD
16-bit-unsigned	16-bit unsigned
16-bit-signed	16-bit signed
32-bit-unsigned	32-bit unsigned
32-bit-unsigned	32-bit signed
32-bit-float	32-bit floating point number

32-bit data storage in registers

1. Mapping relationship between the variable name and register address in the PLC project (16-bit register)
   - Variable %MW5 corresponds to register address 5.
   - Variable %MD5 corresponds to register address 10.
     Variable name prefix: W - word (16 bits); D - double word (32 bits).
2. The addresses of drive device commands and channel collection registers start from 0, which indicates that the register address is 0.
3. When 32-bit data is read or written, two registers (32 bits) starting from the corresponding register address are occupied.

5 Programming Basics

5.1 Overview

Operands are the objects in user programs related to operators, functions, function blocks, or program operations. They can be used as input, output, and intermediate stored results. Common operands of CoDeSys include direct addresses, constants, and variables.

Similar to other advanced languages, CoDeSys also provides constants and variables. Constants are unchanged numeric values. Variables are user-defined identifiers. Variables are stored in user specified addresses of the %I, %Q, and %M areas. If addresses are not specified, variables are stored in system-allocated addresses. You do not need to concern the variable storage location.

5.2 Direct Address

5.2.1 Syntax

A direct address, also called fixed address or direct variable, has a direct mapping to a specific address of the PLC. The address information includes the variable storage location in the CPU, storage size, and storage position offset.

Syntax: %<storage area prefix><size prefix><number>|.<number>

- <storage area prefix>: The programming system supports the following storage area prefixes:

  1. I: input, physical input, sensor
  2. Q: output, physical output, activator
  3. M: storage location

- <size prefix>: The programming system supports the following size prefixes:

  1. X: bit, 1 bit
  2. B: byte, 1 byte
  3. W: word, 1 word
  4. D: double-word, 4 bytes (double word)

- <number>|.<number>
  The first number indicates the offset address of the storage area prefix. The number following "." indicates the specific bit after the address offset when the variable is of the BOOL type.

Example:

%QX7.5 output area with 7-byte offset, sixth bit (bit 5)

%QB17 output area with 17-byte offset

%IW215 input area with 215-word offset

%MD48 memory area with 48-double-word offset

iVar AT %IW10: WORD;//The variable iVar is of the word type and maps to the location with 10-word offset in the input area.

Note

- When a variable with the X-type size prefix indicates the BOOL data type, the offset address needs to be precise to bits.
- The size prefix matches with the data type. A variable with the B-type size prefix must be declared as a 1-byte data type, such as BYTE, SINT, and USINT. A variable with the W-type size prefix must be declared as a 1-word data type, such as WORD, INT, and UINT. A variable with the D-type size prefix must be declared as a double-word data type, such as DWORD, DINT, and UDINT.

5.2.2 PLC Direct Address Storage Area

The direct address storage area varies with PLCs. PLC data is not retained upon power failure for the %I and %Q areas, but is retained for the %M area.

The AM600, AM610, AM401, and AM402 programming systems provide the 128-KB (byte) input area (I area), 128-KB (byte) output area (Q area), and 512-KB storage area (M area). The first 480 KB of the storage area can be used directly, whereas the last 32 KB are used by the system, mainly as soft elements, and cannot be used directly by users. During programming, users can directly access addresses or define a variable, map the variable to an address, and then access the address. The following table lists storage areas and the address ranges they use.

Area	Use	Size	Address Range
I area (%I) 128 KB	For users	64 K words	%IW0 to %IW65535
Q area (%Q) 128 KB	For users	64 K words	%QW0 to %QW65535
M area (%M) 512 KB	For users	240 K words	%MW0 to %MW245759
	SD element	10000 words	%MW245760 to %MW255759
	SM element	10000 bytes	%MB511520 to %MB521519
	Reserved	2768 bytes	%MB521520 to %MB524287

The AC800-series programming system provides a 128-KB input area (I area), a 128-KB output area (Q area), and a 5-MB storage area (M area). The AC800 series does not support SD and SM soft elements and addresses in the %M can be used without restriction. The following table lists storage areas and the address ranges they use.

Area	Use	Size	Address Range
I area (%I) 128 KB	For users	64 K words	%IW0 to %IW65535
Q area (%Q) 128 KB	For users	64 K words	%QW0 to %QW65535
M area (%M) 5 MB	For users	2.5MWords	%MW0 to %MW2321439

5.3 Variable

5.3.1 Overview

Variables can be defined by the POU, automatic declaration dialog box, and the DUT or GVL editor. Variable types are identified through variable type keywords. For example, VAR and END_VAR identify local variables.

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

5.3.2 Variable Definition

Variables can be edited in the declaration editor. The declaration editor is displayed in the text view or table view. Complex data types such as structures and arrays support variable definitions, array element comments, and recursive display of addresses of complex data types.

The following figure shows the declaration editor of POU in the text view.

```
VAR_GLOBAL
END_VAR
VAR_GLOBAL RETAIN PERSISTENT
    {attribute 'ElemComment':='A_0([(ERT),9()]),A_1([10()])'}
    A AT%MW0:DUT := (A_0 := [2(FALSE), TRUE, 7(FALSE)]);
END_VAR
VAR_GLOBAL
        {attribute 'ElemComment':='A_0([(EEE),9()]),A_1([10()])'}
    B AT%MW200:DUT;
    C AT%MW400:DUT;
END_VAR
```

Figure 5-1 Text declaration

The following figure shows the declaration editor of POU in the table view.

	Scope	Name	Address	Data type	Initialization	Persistent	Constant	Network Publish	Comment	Attributes
1	VAR_GLOBAL RETAIN PERSISTENT	A_0		BOOL		☑	☐	Default	Variable A	
2	VAR_GLOBAL CONSTANT	A_1		INT		☐	☑	Default		
3	VAR_GLOBAL	A_2		BYTE		☐	☐	Default		
4	VAR_GLOBAL	A_3		WORD	100	☐	☐	Default		
5	VAR_GLOBAL	A_4		DWORD		☐	☐	Default		AAA
6	VAR_GLOBAL	A_5		ARRAY[0..9] OF REAL		☐	☐	Default		

Figure 5-2 Table declaration

In the table declaration, you can edit variable attributes. The following table describes items in the table.

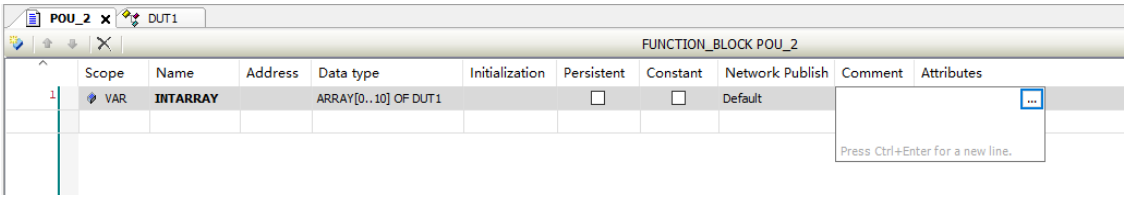
Item	Description
Scope	The variable type (such as local variable, input variable, output variable, or temporary variable)
Name	The variable name
Address	The address of the variable after compilation

Item	Description
Data type	The data type of the variable (such as INT or BOOL)
Initialization	The initial value of the variable
Persistent	Indicate whether the variable is persistent
Constant	Indicate whether the defined variable is a constant
Comment	The variable comment
Attributes	The variable attributes

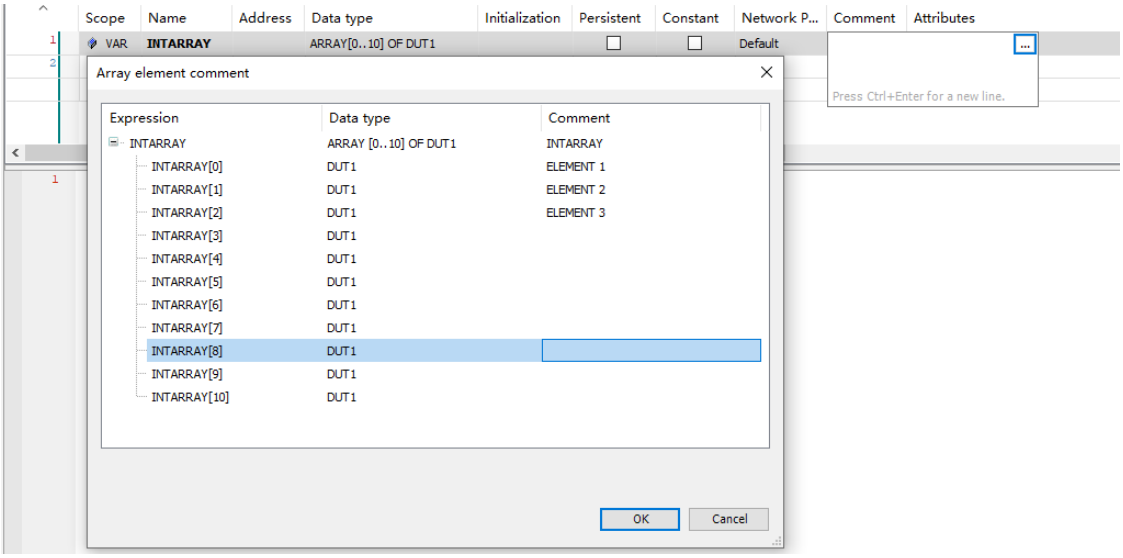
Variable definition supporting array element comments and instance comments

1. **Array element comments**

   The following figure shows a comment setting page in table declaration mode.



   Double-click  at a blank area under "Comment".



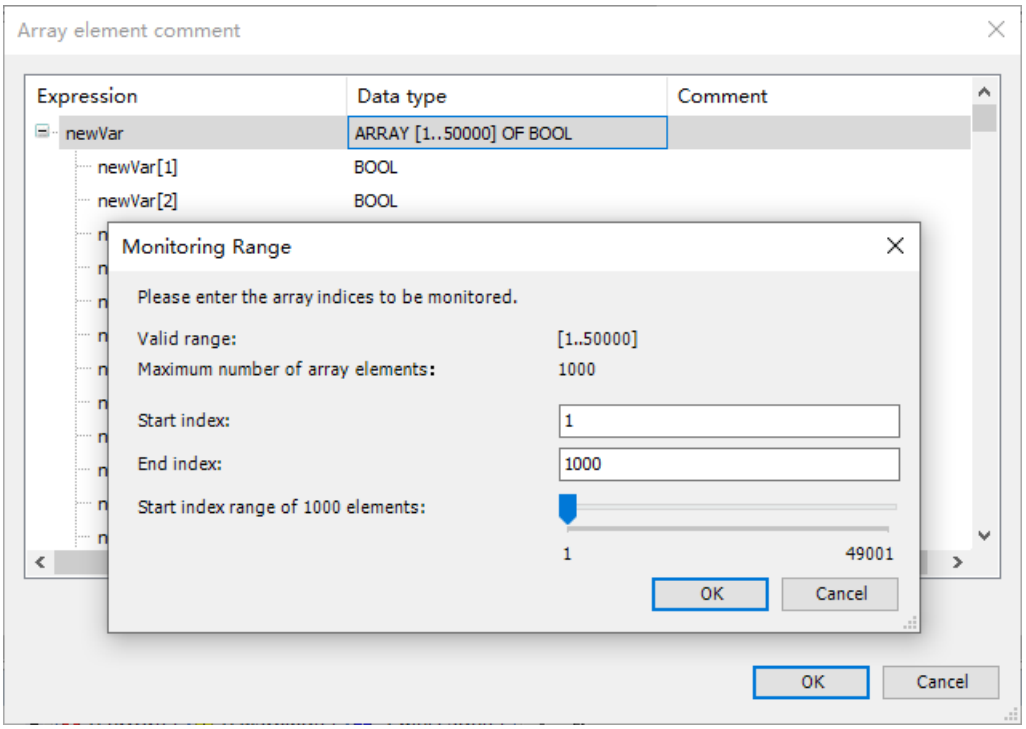
   Set the comments. The following figure shows the text declaration effect (you can make the declaration directly in text).

```
1  FUNCTION_BLOCK POU_2
2  VAR_INPUT
3  END_VAR
4  VAR_OUTPUT
5  END_VAR
6  VAR
7      // 整形数组
8      {attribute 'ElemComment':='(元素1),(元素2),(元素3),7()'}
9      INTARRAY: ARRAY[1..10] OF 结构体1;
10 END_VAR
11
```

- You can edit array element comments in the text view or table view.

- In the table view, the editing page for the current element and sub-element comments is displayed in the "Comment" column on the displayed window (the operation is the same as the initialization operation).
- The edit format of the text editor is as follows:
  - Array: Use the standard comment edit method.
  - Array element: {attribute 'ElemComment':='1(comment of sub-element 1),1(comment of sub-element 2),n(same sub-element comment)'}.
- In the table view, the comment is null (the "Attributes" column is added by default) when the array type variables are declared.
- In the table view, only comments of the arrays are displayed. Comments of elements are not displayed.
- In the "Attributes" column, remove the element comment attribute display (the array element comments are implemented by attributes which are information marked on the variables).
- When the data length in the table view changes, existing array element comments are saved accordingly.
- When the array dimension in the table view changes, the array element comments are migrated and saved according to the smallest sub-index of the expanded dimension.
  If the dimension of the array INT_ARRAY:ARRAY[1..2,2..3] changes to ARRAY[1..2,2..3,3..4], the comments of the original array element INT_ARRAY[1,2] are migrated to the new array element INT_ARRAY[1,2,3].

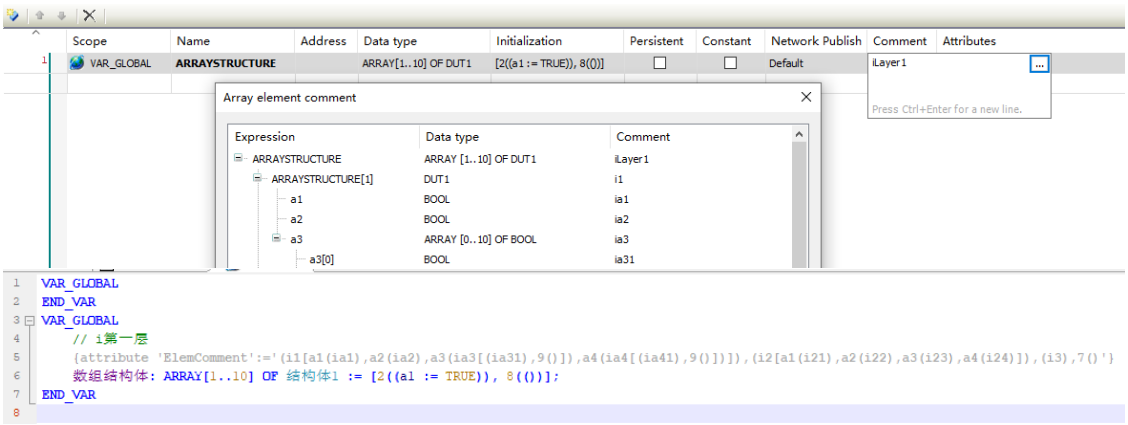
  If the dimension of INT_ARRAY:ARRAY[1..2,2..3] changes to ARRAY[1..2], the comments of the original array element INT_ARRAY[1,2] are migrated to the new array element INT_ARRAY[1].
- In the table view, when the data type changes from an array to a non-array, the array element comments are cleared.
- On the array element comment editing page, up to 1000 elements are displayed. Double-click an item in the "Data type" column of the array to edit the display range.



2. **Instance comment**

A variable declared in PRG (Program) and GVL (Global Variable List) or declared to be of type VAR_STAT (Static) can be expanded without restriction to edit the comments of the internal members of the variable, and all the comments of the internal members will be marked on the variable when the comments are saved, which is known as the instance comment of the variable.

As shown in the following figure, all member comments within a data structure can be marked and saved on the variable array structure.



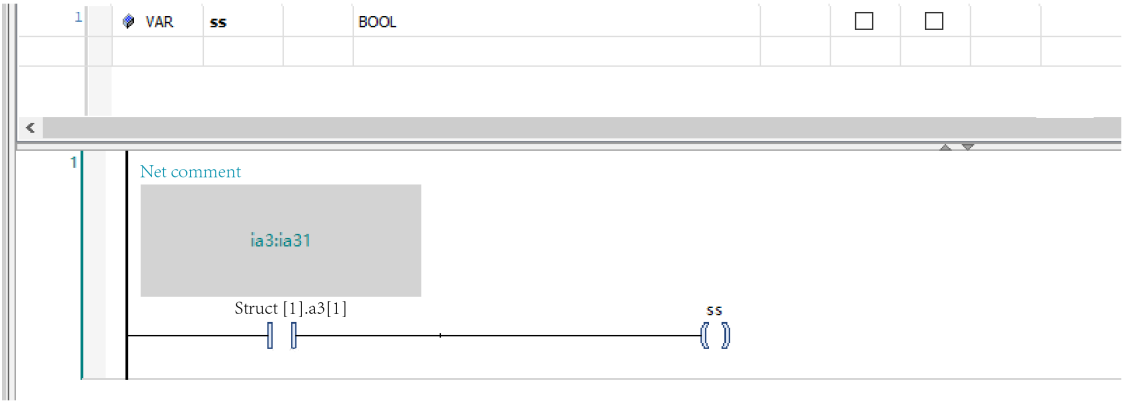
- When the internal member is of type FB, only input, output, and input/output variables are displayed; variables of other types are not displayed.
- In the table, when the data type is changed from non-array to array, the instance comment is cleared.
- The maximum number of elements displayed for the array type member of a variable is 1000, and the display range can be adjusted.

3. **Comment display**

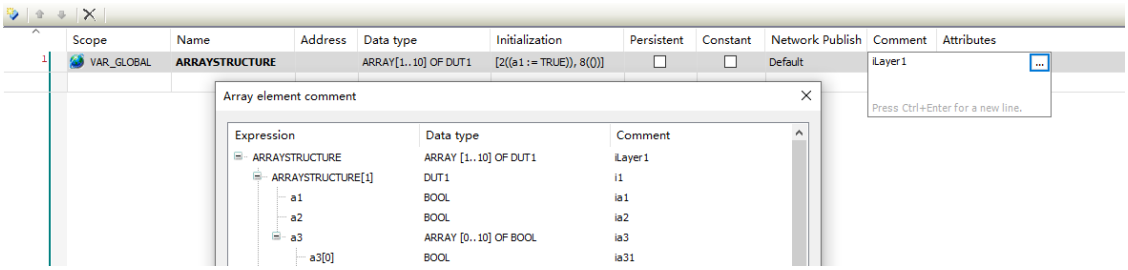
On the initial value editing page, monitoring variable table page, ladder diagram, cursor hovering display comment, and other functions involving the display of variable comments, the display of comments is prioritized by the instance comment, and if the variable does not have an instance comment, the type comment of the variable is displayed.

If the ladder diagram involves the display of the comments of array elements, the array comments and element comments will be displayed together; however, the same rules of prioritization will be used for the display.

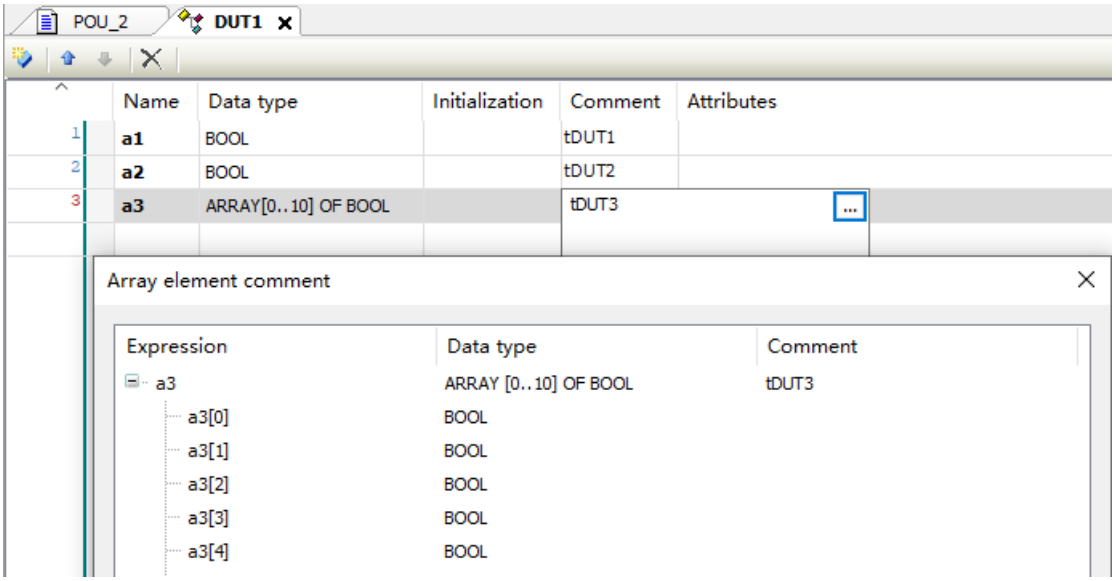
For example, the following figure shows the comments of the array element data structure [1].a3[1].



Instance comment:

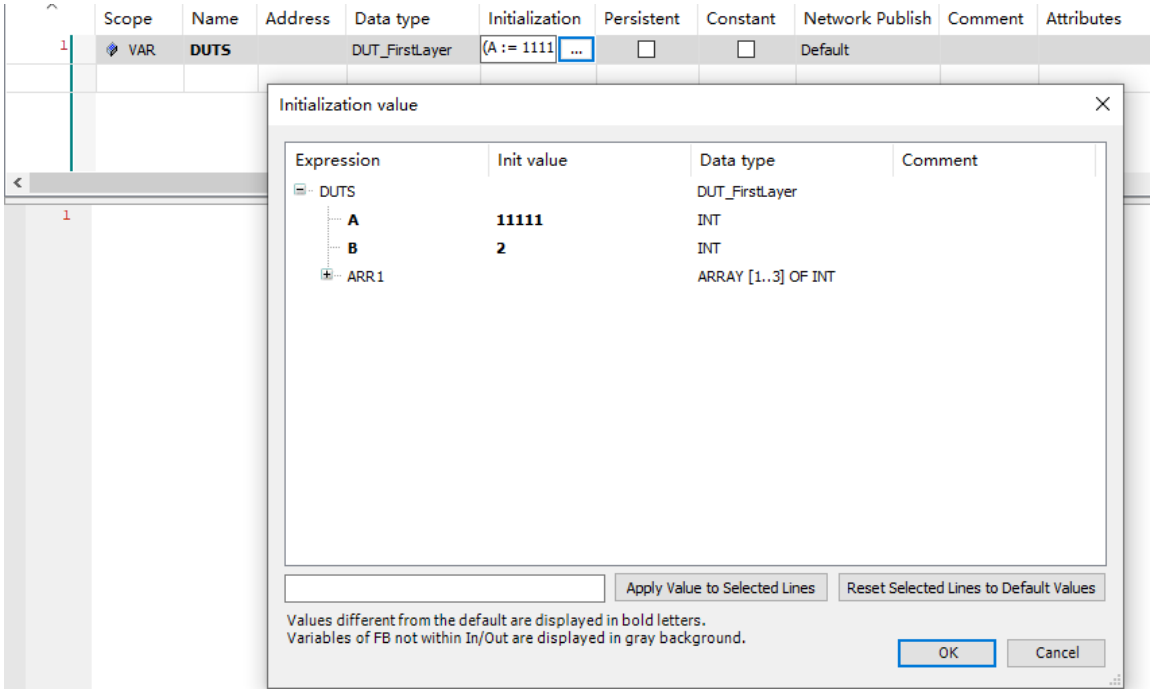


Due to existence of instance comment, the array comments and element comments defined in the type are not displayed at this time, as shown in the following figure.



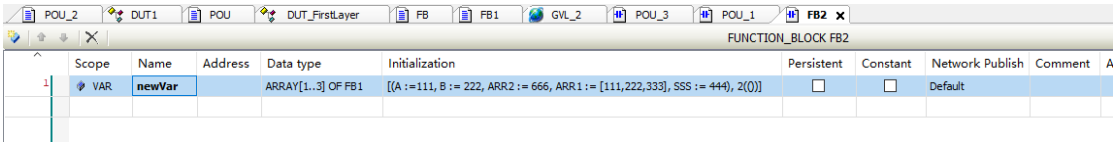
Variable initial value setting

The following figure shows an initial value setting page.

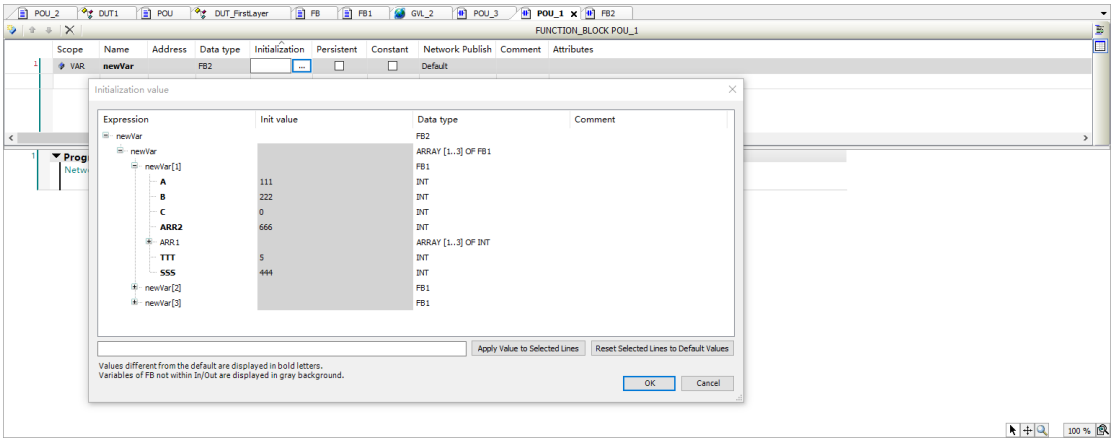


- In the table view of variable declaration, click a value in the "Initialization" column to directly edit the initial value. You can also click the "..." icon to expand the menu and edit the value.
- When the initial value setting is different from the default value, the initial value is displayed in bold.
- The initial value setting page will display members of the variables layer by layer. The intermediate variables cannot be set directly. Only the initial value of terminal variables can be set. Items in the "Expression" column for which can set the initial value are displayed in bold.
- Members inside the function block that are not input or output variables are displayed in gray and cannot be edited.

  For example, the newVar member in FB2 is not an input or output variable.



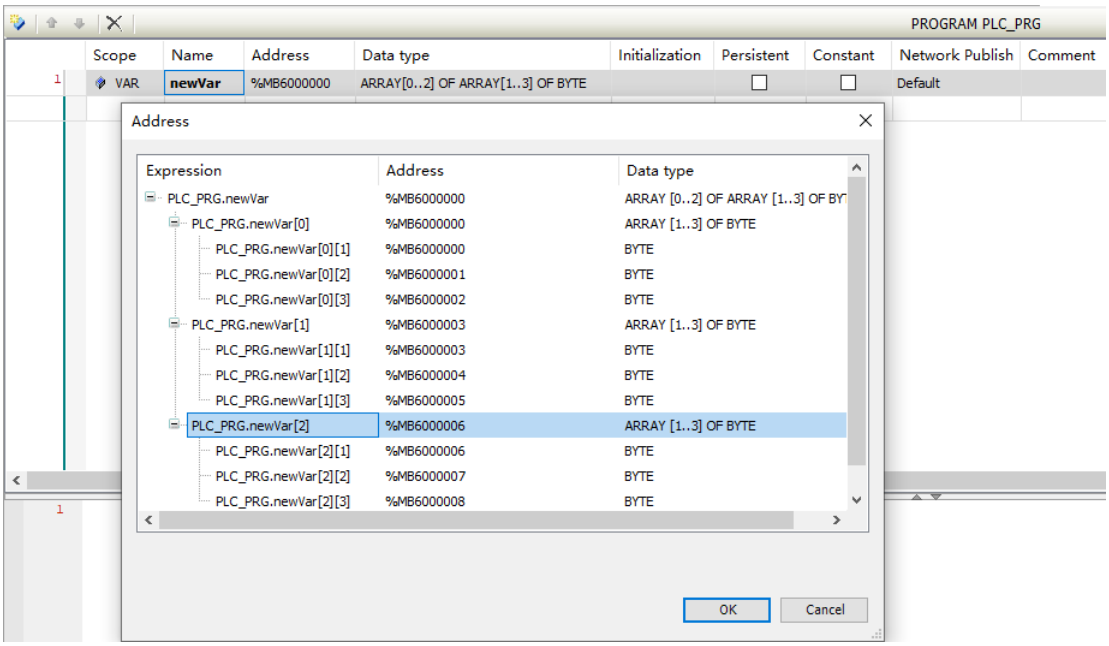
  When the newVar member of FB2 type is expanded, the initial value cannot be edited (with a gray background).



- If there is an array in the expanded item, the maximum number of elements displayed in the array is 1000. You can double-click the "Data type" cell in the row where the array is located to adjust the display range.

Address information display of sub-elements in variable definition

The following figure shows an address display page.



- In the table view of variable declaration, click an item in the "Address" column. If the variable is a block type that does not contain a function block (such as array, structure, union, or alias), a text box is displayed after the click operation, which displays the address of the sub-element.
- Enter the address in the "Address" column text box. "Read only" is displayed on the address display page.
- Modify the address. Then, the modification is synchronized to the variable address.
- The maximum number of elements to be displayed (up to 1000 elements in the array, adjustable display range).

Identifier

An identifier is the name of a variable. The variable naming conventions are as follows:

- The name cannot contain spaces or special characters.
- The name cannot contain predefined keywords.
- The name is case-insensitive.
- The name length is unlimited.
- The name cannot be defined repeatedly.

A local variable name can be the same as a global variable name. By default, the local variable is used and can indicate a global variable. A specific variable can also be indicated by a full path variable name. Example: local variable iVar: = 1; global variable .iVar: = 2; full path variable globlist1.iVar: = 3.

Consider naming suggestions when you name a variable. For example, a variable name must accurately indicate the meaning and data type of the variable, and the Hungarian notation (variable name = attribute + type + object description) is recommended.

AT address

An AT address is a direct address. For details, see *"5.2.1 Syntax" on page 346*.

Data type

Data types are classified into standard data type and user-defined data type.

1. Standard data type

   Standard data types are classified into boolean, integer, floating point, string, and time.

Type	Keyword	Range	Memory Usage
Boolean	BOOL	TRUE, FALSE, 0, and 1	8-bit
Bit type	bit	TRUE, FALSE, 0, and 1, only used in structures or function blocks	1-bit
Integer	Byte	0 to 255	8-bit
	WORD	0 to 65535	16-bit
	DWORD	0 to 4294967295	32-bit
	LWORD	0 to 2^{64} - 1	64-bit
	SINT	–128 to +127	8-bit
	USINT	0 to 255	8-bit
	INT	–32768 to +32767	16-bit
	UINT	0 to 65535	16-bit
	DINT	–2147483648 to +2147483647	32-bit
	UDINT	0 to 4294967295	32-bit
	LINT	-2^{63} to $+2^{63}$ - 1	64-bit
	ULINT	0 to 2^{64} - 1	64-bit
Floating point	REAL	1.401e-45 to 3.403e+38	32-bit
	LREAL	2.2250738585072014e to 308 - 1.7976931348623158e+308	64-bit
String	STRING	Only ASCII characters are supported. Chinese characters are not supported. By default, the maximum length is 80 characters. The part exceeding the maximum length is truncated. The maximum character length can be declared, for example, str:STRING(35):='This is a String'. A string function supports up to 255 characters.	Strings are stored in the ASCII format. The terminator is stored as one byte.
	WSTRING	Only Unicode characters (including Chinese characters) are supported. By default, the maximum length is 80 characters. The part exceeding the maximum length is truncated. The maximum character length can be declared, for example, wstr:WSTRING(35):="This is a WString";.	Strings are stored in the Unicode format. The terminator is stored as two bytes.
Time	TIME	Time constant, such as day, hour, minute, second, and millisecond	32-bit, processed according to double-word internally
	TIME_OF_DAY(TOD)	Time constant within one day	32-bit, processed according to double-word internally
	DATE	Date constant, starting from January 1, 1970	32-bit, processed according to double-word internally
	DATE_ADN_TIME(DT)	Date and time constant, starting from January 1, 1970	32-bit, processed according to double-word internally

2. User-defined data type

User-defined data types include array, structure, enumeration, union, alias, subset, reference, and pointer. In programming software InoProShop of medium-sized PLCs, right-click an application and choose "Add Object" > "DUT" from the shortcut menu to add the following four user-defined data types: structure, enumeration, union, and alias.

Array

Syntax: <Array_Name>:ARRAY [<ll1>..<ul1>,<ll2>..<ul2>,<ll3>..<ul3>] OF <elem. Type>

ll1, ll2, and ll3 define the lower limit of the area, whereas ul1, ul2, and ul3 define the upper limit. The numeric value must be an integer. "elem. Type" indicates 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 value 0,0,4,4,4,4,2,3*)

arr1 : ARRAY [1..10] OF INT := [1,2]; (*Array initialization. Uninitialized elements adopt the default value 0*)

Example of array structure initialization

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)];

Syntax of access union elements:

<Array-Name>[Index1,Index2].

Example:

Card_game [9,2]

Structure

Syntax:

TYPE <structurename> | EXTENDS DUTTYPE:

STRUCT

<declaration of variables 1>

...

<declaration of variables n>

END_STRUCT

END_TYPE

<structurename> is a type and can be used as a data type. EXTENDS DUTTYPE is optional and indicates inheritance from the members of DUTTYPE. Variables of the structure name type can be used to access the members of DUTTYPE. DUTTYPE is of the structure, union, or alias type.

Initialization and example

Polygonline 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]);

Syntax of access structure elements:

<structurename>.<variable>

Example:

Poly_1.Start

Enumeration

An enumerated value consists of several constants.

Syntax:

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

END_TYPE

identifier: user-defined enumeration type; enum_n: constant value of the enumeration type. Each constant can declare its value. If no value is declared, the default value is used. The data type of enumerated constants is base data type. The value may not be declared and is an integer by default.

Note

When an enumeration variable exists in multiple libraries, you need to add the library name prefix; otherwise, an error is reported during compilation.

Union

Syntax:

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 of the union type share the same storage location and are allocated with space same as that of the variable that occupies the largest space.

Example

TYPE union1: UNION

a : LREAL;

b : LINT;

END_UNION

END_TYPE

Syntax of access array elements:

< unionname >.<variable>

Example

union1.a

Alias

A data type can be expressed by an alias.

Syntax:

TYPE <aliasname>:basetype END_TYPE

"aliasname" indicates the alias type and is used as a data type. "basetype" is a standard or user-defined data type.

Example

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

The initialization and access mode are consistent with the basic type.

Subset

The subset data type is a subset of the defined basic data type. A subset type can be added by adding a DUT. A variable can be directly declared as a subset type.

Syntax of DUT objects:

TYPE <name> : <Inttype> (<ug>..<og>) END_TYPE;

name: valid IEC identifier.

Inttype: a data type, such as SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, and DWORD (LINT, ULINT, and LWORD).

ug: a constant, which must be compatible with the basic type and sets the lower boundary of the range types. The lower boundary itself is included in this range.

og: a constant, which must be compatible with the basic type and sets the upper boundary of the range types. The upper boundary itself is included in this basic type.

Example of DUT object declaration

```
TYPE
SubInt : INT (-4095..4095);
END_TYPE
```

Example of direct variable declaration

```
VAR
  i : INT (-4095..4095);
  ui : UINT (0..10000);
END_VAR
```

Reference

Reference is the alias of an object. Operating references is equivalent to operating objects.

Syntax:

<identifier> : REFERENCE TO <data type>

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

Example and initialization

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

Note

The bit type cannot be referenced. That is, ref1:REFERENCE TO BIT cannot be defined.

Pointer

A pointer stores the address of an object and can point to any data type (except the bit type).

Syntax:

<identifier>: POINTER TO <data type>;

identifier: pointer identifier. data type: data type pointed to by a pointer.

Pointers are operated by using address operators. Address operators include ADR (variable address acquisition) and ^ (value of a variable address).

Example and initialization

```
VAR
  pt:POINTER TO INT; (* Declares the pointer pt of the INT type*)
  var_int1:INT := 5;
  var_int2:INT;
END_VAR

pt := ADR(var_int1); (* Allocates the address of the varint1 variable to the pointer pt *)
var_int2:= pt^; (* Uses the ^ address operator to obtain the value of the pointer*)
pt^:=33; (*Assigns a value to the var_int1 variable corresponding to the pointer*)
```

Initial value

By default, the initial value of a variable is 0. You can add user-defined initial values by using the valuation operator ":=" during variable declaration. An initial value is a valid ST expression. An ST expression consists of operators, operands, and a valuation expression. Operators mainly include addition (+), subtraction (–), multiplication (*), and division (/). Operands mainly include constants, variables, and functions. A valuation expression is the operator in the ST expression used to assign values to variables. Therefore, constants, variables, or functions can be initialized. Ensure that the used variables have been initialized.

Example:

```
VAR
var1:INT := 12; (* The initial value of the integer variable is 12*)
x : INT := 13 + 8; (*Defines the initial value of the constant expression*)
y : INT := x + fun(4); (*Includes function call in the initial value*)
z : POINTER TO INT := ADR(y); (*Initializes the pointer by using the address function ADR*)
END_VAR
```

Note

- The global variable list (GVL) is initialized before POU local variables are defined.
- If the default value is modified online, the pointer is not initialized during definition and still points to the variable before online modification.

5.3.3 Variable Type

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

Declaration syntax of the variable type: <type_key> |attribute_key

variable1;

variable2;

...

END_VAR

type_key: type keyword, which may be VAR (local variable), VAR_INPUT (input variable), VAR_OUTPUT (output variable), VAR_IN_OUT (I/O variable), VAR_GLOBAL (global variable), VAR_TEMP (temporary variable), VAR_STAT (static variable), and VAR_CONFIG (configuration variable).

attribute_key: attribute keyword, which may be RETAIN, PERSISTENT, or CONSTANT. It defines the range of a variable.

Note

- For details about the variables RETAIN and PERSISTENT, see *"5.5.3 Persistent Variable Table" on page 372*.
- For details about CONSTANT, see *"5.4 Constants" on page 368*.

Local variable (VAR)

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

Valuation format:

Local variable:=Value

Example

```
VAR
  iLoc1:INT; (* Local variable*)
END_VAR
```

Input variable (VAR_INPUT)

The variables between VAR_INPUT and END_VAR within POU are input variables and assigned values in the call location.

POU call format:

Local variable:=Value input by the caller

Example

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

Note

Input variables can be modified within POU, even when the CONSTANT attribute is added.

Output variable (VAR_OUTPUT)

The variables between VAR_OUTPUT and END_VAR within POU are output variables. Output variables can be returned to the caller during the call process for further processing.

POU call format:

Output variable=>variable of the caller-matched type

Example

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

Note

- For FUNCTION and METHOD, return values and output variables are supported, but a caller needs to be allocated for receiving the variables during the call process. Example: fun(iIn1 := 1, iIn2 := 2, iOut1 => iLoc1, iOut2 => iLoc2);
- The output variables of function blocks can be assigned to the caller after the call process.

I/O variable (VAR_IN_OUT)

The variables between VAR_IN_OUT and END_VAR within POU are I/O variables. I/O variables can be transferred to the called POU and modified within the called POU. In the actual situation, the variables transferred to the called POU are referenced by the caller.

Example

```
VAR_IN_OUT
  iInOut1:INT; (* I/O variable*)
END_VAR
```

Note

- As the variables transferred to the called POU are referenced by the caller, the I/O variables in function block instances cannot be accessed directly. That is, <FBinstance>.<InOutVariable> cannot be used directly. The reason is that input variables are referenced by the caller and have been changed.
- I/O variables cannot be constants or direct variables of the bit type, such as xBit0 AT %I2.0:BOOL. Add the CONSTANT attribute (VAR_IN_OUT CONSTANT) to declare I/O variables. To use direct variables of the bit type, you need to add an intermediate variable as an I/O variable and assign the value of the intermediate variable to the direct variable of the bit type.

Example of a direct variable of the bit type:

```
VAR_GLOBAL
   xBit0 AT %MX0.1 : BOOL;(*Declare a direct variable of the bit type*)
   xTemp : BOOL; (*Intermediate variable*)
END_VAR
//Function block with an I/O variable (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
 //A compiling error is returned when a direct address variable of the bit type is used.
//I1(xIn:=xIn, xInOut:=xBit0);
//Use the intermediate variable xTemp to transfer the value of xBit0 to the function block and assign the value of the intermediate variable to xBit0.
```

```
xTemp := xBit0;
I2(xIn:=xIn, xInOut:=xTemp);
xBit0 := xTemp;
```

I/O constants (VAR_IN_OUT CONSTANT) are read-only. Input variables can be modified in the current version, even when the constant attribute is added. Therefore, the variable attribute can be changed to non-modifiable by using an I/O constant.

I/O constant example:

```
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); (* The string can be read and written within the POU *)
   iVarReadWrite : DWORD; (*The variable can be read and written within the POU*)
END_VAR
VAR_IN_OUT CONSTANT
   scReadOnly : STRING(16); (*The string is read-only within the POU*)
END_VAR
sReadWrite := 'string_from_POU';
iVarInPOU := STRING_TO_DWORD(scReadOnly);
```

Global variable (VAR_GLOBAL)

The variables between VAR_GLOBAL and END_VAR are global variables. Common variables, constants, and reserved variables can be declared as global variables. In the AM600 programming software InoProShop, right-click an application and choose "Add Object" > "Add Global Variable List" from the shortcut menu to add a global variable list, and add global variables to the list.

Example

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

Note

- If a local variable has the same name as a global variable, the local variable is operated when an operation is performed on the variable name. You can add the global range operator (.) before the variable name to operate the global variable, such as ". iGlobVar1".
- Global variables are always initialized before local variables.

Temporary variable (VAR_TEMP)

The variables between VAR_TEMP and END_VAR are temporary variables, which are initialized when being called.

Example

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

Note

- Temporary variables are declared only in programs and function blocks.
- Temporary variables are used only in declared programs or function blocks.

Static variable (VAR_STAT)

The variables between VAR_STAT and END_VAR are static variables. Static variables are initialized when being called for the first time. The variable values are returned after the POU is called each time.

Example

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

Note

- Static variables are declared only in function blocks, functions, and methods, but cannot be declared in programs.
- Static variables are used only in the declared POU.

Configuration variable (VAR_CONFIG)

The variables between VAR_CONFIG and END_VAR are configuration variables. Configuration variables are direct variables that are mapped to the direct variables with indefinite addresses in function blocks. A variable with an indefinite address can be defined in a function block. The indefinite address (arbitrary address) is indicated by "*". Add a configuration variable list (by adding a global variable list)

to add the variables with indefinite addresses in all the function block instances to the configuration variable list, which defines all the indefinite addresses. This allows you to manage the variables with indefinite addresses in all the function blocks.

Syntax for variables with indefinite addresses in function blocks:

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

Addresses are finally defined 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, a local input variable (%I*), and a local output variable (%Q*) are defined.

A global variable list (GVL) is added. The specific addresses declared by instance variables are entered between the keywords VAR_CONFIG and END_VAR. The instance variables include the complete instance path of the POU. The specific addresses correspond to indefinite addresses (%I* and %Q*) in function blocks. The data type must be consistent with that declared by function blocks.

Syntax of configuration variables:

<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
```

Note

- Configuration variables are not required in normal cases. The reason is that for I/O address input and output, variables can be mapped to I/O addresses by using an input assistant or directly entering the instance variable path on the I/O mapping page of the corresponding module.
- Configuration variables are mapped to variables with indefinite addresses in function blocks or programs.
- A compiling error is returned when only variables with indefinite addresses or configuration variables exist. The two types of variables must be used in combination.

5.3.4 Variable Import and Export

Variables can be imported and exported from/to an XLS worksheet (.xls) in the format of an Excel file. You can add, delete, or edit variables and then import the settings to the InoProShop programming software.

See the following figure.

	Scope	Name	Address	Data type	Initialization	Persistent	Constant	Network Publish	Comment	Attributes
1	VAR_GLOBAL RETAIN PERSISTENT	**A_0**		BOOL		☑	☐	Default	变量A	
2	VAR_GLOBAL CONSTANT	**A_1**		INT		☐	☑	Default		
3	VAR_GLOBAL	**A_2**		BYTE		☐	☐	Default		
4	VAR_GLOBAL	**A_3**		WORD	100	☐	☐	Default		
5	VAR_GLOBAL	**A_4**		DWORD		☐	☐	Default		AAA
6	VAR_GLOBAL	**A_5**		ARRAY[0..9] OF REAL		☐	☐	Default		
7	VAR_GLOBAL RETAIN PERSISTENT	**A_6**		BYTE		☑	☐	Default		
8	VAR_GLOBAL CONSTANT	**A_7**		WORD		☐	☑	Default		
9	VAR_GLOBAL	**A_8**		DWORD	200	☐	☐	Default		
10	VAR_GLOBAL	**A_9**		ARRAY[0..9] OF REAL		☐	☐	Default	变量A	

Add some variables to the variable table. Right-click and select Excel or CSV as the export format. CSV files are plain text files, whereas Excel files contain format information. CSV files are small, so they are easy to create, distribute, and read, suitable for storing structured information. CSV files are opened in Excel by default in the Windows operating system. They are text files in essence. There is no difference between the Excel and CSV formats in terms of variable editing.

Open the exported file, edit it (add new variables A_6, A_7, A_8, and A_9), and then import it to the variable table. The following figure shows the effect.

Type	Name	Address	DataType	InitValue	Comment	Attribute
VAR_GLOBAL RETAIN PERSISTENT	A_0		BOOL		变量A	
VAR_GLOBAL CONSTANT	A_1		INT			
VAR_GLOBAL	A_2		BYTE			
VAR_GLOBAL	A_3		WORD	100		
VAR_GLOBAL	A_4		DWORD			
VAR_GLOBAL	A_5		ARRAY [0..9] OF REAL			
VAR_GLOBAL RETAIN PERSISTENT	A_6		BYTE			
VAR_GLOBAL CONSTANT	A_7		WORD			
VAR_GLOBAL	A_8		DWORD	200		
VAR_GLOBAL	A_9		ARRAY [0..9] OF REAL		变量B	

	Scope	Name	Address	Data type	Initialization	Persistent	Constant	Network Publish	Comment	Attributes
1	VAR_GLOBAL RETAIN PERSISTENT	A_0		BOOL		☑	☐	Default	变量A	
2	VAR_GLOBAL CONSTANT	A_1		INT		☐	☑	Default		
3	VAR_GLOBAL	A_2		BYTE		☐	☐	Default		
4	VAR_GLOBAL	A_3		WORD	100	☐	☐	Default		
5	VAR_GLOBAL	A_4		DWORD		☐	☐	Default		AAA
6	VAR_GLOBAL	A_5		ARRAY[0..9] OF REAL		☐	☐	Default		
7	VAR_GLOBAL RETAIN PERSISTENT	A_6		BYTE		☑	☐	Default		
8	VAR_GLOBAL CONSTANT	A_7		WORD		☐	☑	Default		
9	VAR_GLOBAL	A_8		DWORD	200	☐	☐	Default		
10	VAR_GLOBAL	A_9		ARRAY[0..9] OF REAL		☐	☐	Default	变量A	

5.4 Constants

In PLC programming, constants are parameters with unchanged values, such as timer time and conversion ratio.

Constant declaration syntax:

VAR CONSTANT

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

END_VAR

Example

VAR CONSTANT

c_iCon1:INT:=12;

END_VAR

CoDeSys supports constants of multiple data types, such as boolean, integer, time, and string. The following table lists specific constants.

Type	Description	Example
Boolean	The optional values are "TRUE" and "FALSE" (or 1 and 0). 1 indicates "TRUE", and 0 indicates "FALSE".	TRUE, FALSE, and 1
Bit type	Similar to the boolean type, the bit type is used only in structures (number of occupied bits) or function blocks (direct addresses of the boolean type).	TRUE, FALSE, and 0
Integer	Integer constants support values in the binary, decimal, octal, and hexadecimal formats. If an integer value is not in the decimal format, add the format number and the # symbol before the value. 10 to 15 in the decimal format are indicated by A to F in the hexadecimal format.	Decimal format: 66 Binary format: 2#101 Octal format: 8#72 Hexadecimal format: 16#3A Type constants: INT#22 BYTE#204
Floating point	Floating point constants are expressed by decimal numbers and exponents in scientific notation.	7.4 2.3e+9 REAL#3.12

Type	Description	Example
ASCII string	An ASCII string constant is located between two single quotation marks and can include spaces and special characters. A character is expressed by a byte. Only ASCII characters are supported. Chinese characters are not supported. By default, the maximum length is 80 characters. The part exceeding the maximum length is truncated. The maximum character length can be declared, for example, str:STRING(35):='This is a String'. A string function supports up to 255 characters.	Example of $ used as an escape character: '$30': 0, character 0, ASCII character corresponding to 30 in the hexadecimal format $$: $, US dollar character $': ', single quotation mark
Unicode string	A Unicode string constant is located between two double quotation marks. A character occupies two bytes. Only Unicode characters (including Chinese characters) are supported. By default, the maximum length is 80 characters. The part exceeding the maximum length is truncated. The maximum character length can be declared, for example, wstr:WSTRING(35):="This is a WString";.	"Unicode string"
Time	Time constants are generally used by time-related operations and consist of "T#" (or "t#") and a time value, in the units of days (d), hours (h), minutes (m), seconds (s), and milliseconds (ms).	t#12h34m15s;
Time	Time range within a day; syntax: TOD#time value.	TOD#15:36:30.123
Date	Starting from January 1, 1970; syntax: d#date.	d#2015-02-12
Date and time	Date constants and time constants are collectively called date and time constant, which starts from January 1, 1970; syntax: dt#date.	dt#2004-03-29-11:00:00

Note

Constants not of the BOOL, BIT, and string types are indicated in the format "keyword#constant value".

5.5 Persistent Variable

5.5.1 Overview

Caution

The persistent variable is only supported in the editor 3.5.11.71 and later versions. In the toolbar, choose "Project" > "Project Settings". In "Project Settings" dialog box displayed, view the editor version on the "Editing Options" page.

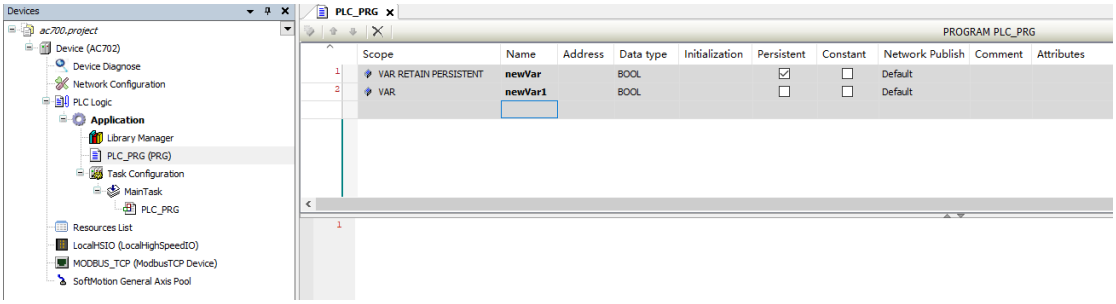
The original value of the persistent variable is retained upon PLC power failure or after the program is downloaded. This variable is used to define important parameters in the project to prevent the loss of important parameters due to sudden power failure of the PLC or program download.

5.5.2 Variable Definition

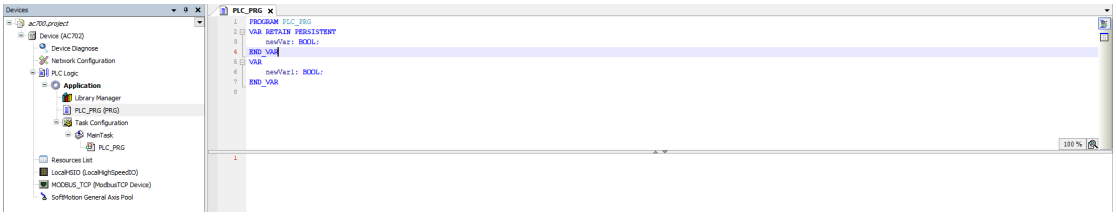
Define variables

The persistent variable can be defined in Global Variable List (GVL), Program (PRG), Function Block (FB), and Function (FUN, static variables only), but not in Method (METH), Property (Prop), Structure (STRUCT), Unions (Union), Enumeration (Enum), and Alias (Alias), which can be defined by both table and text. The following takes the definition in Program (PRG) as an example.

- **Table view**
  In the table on the programming page of "PLC_PRG(PRG)", select the box next to a variable under "Persistent" to define the variable as a persistent variable.



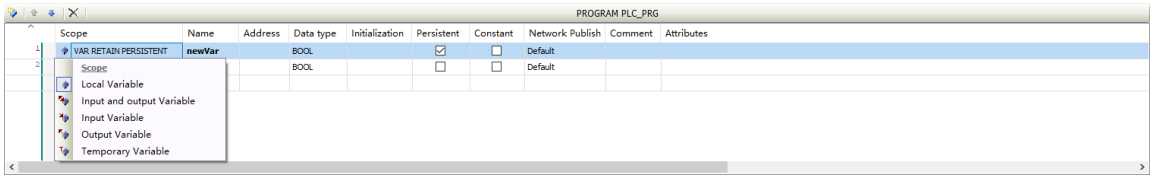
- **Text view**
  In the text on the programming page of "PLC_PRG(PRG)", add the keywords "RETAIN PERSISTENT" or "PERSISTENT RETAIN" to define the variable as a persistent variable.



Variable type

Persistent variables can be set as local variables, input variables, output variables, and static variables, but cannot be set as I/O variables, temporary variables, constants, and configuration variables.

As an example, to set the variable type in a table in the program (PRG), double-click the "Scope" column corresponding to the variable row, and then set the supported variable types on the page displayed.

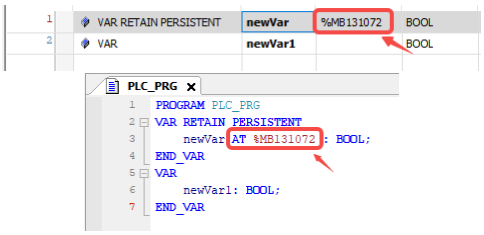


Variable type

Persistent variables can be defined as any types other than pointers, references, and function blocks (FB), including nested pointers, nested references, and nested function blocks.

Variable mapping address

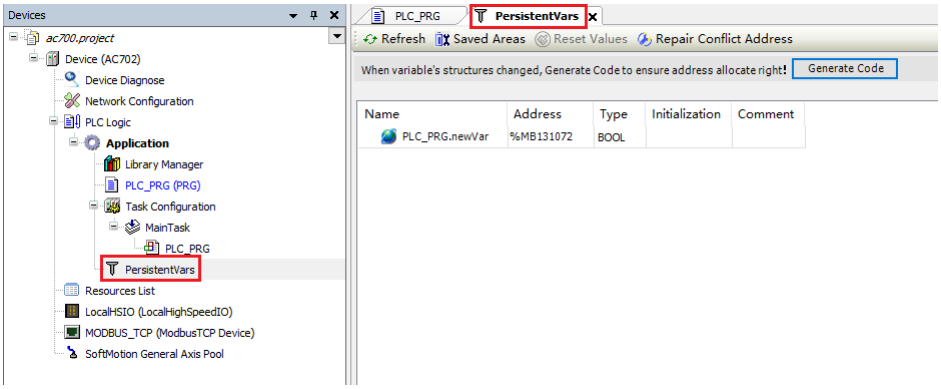
After a variable is defined as a persistent variable, an address mapped to the M area is automatically generated after compilation, which can be manually edited.



Caution

Persistent variables can only be set with addresses in the M area. Addresses in I and Q areas are not supported.

Besides, if the target node "PersistentVars" does not exist in the original device tree, then the target node "PersistentVars" is generated. If the target node "PersistentVars" exists in the original device tree, the internal persistent variable is updated and all persistent variables in the project are added to the object view.



Variable response action

The following table lists response actions of different persistent variables upon execution reset and power failure.

Action	VAR	VAR PERSISTENT RETAIN or VAR RETAIN PERSISTENT	VAR RETAIN
Power failure	Initialization	Original value retained	Original value retained
Hot reset	Initialization	Original value retained	Original value retained
Cold reset	Initialization	Original value retained	Initialization
Initial value reset	Initialization	Initialization	Initialization
Program download	Initialization	Original value retained	Initialization
Online modification	Original value retained	Original value retained	Original value retained

Note

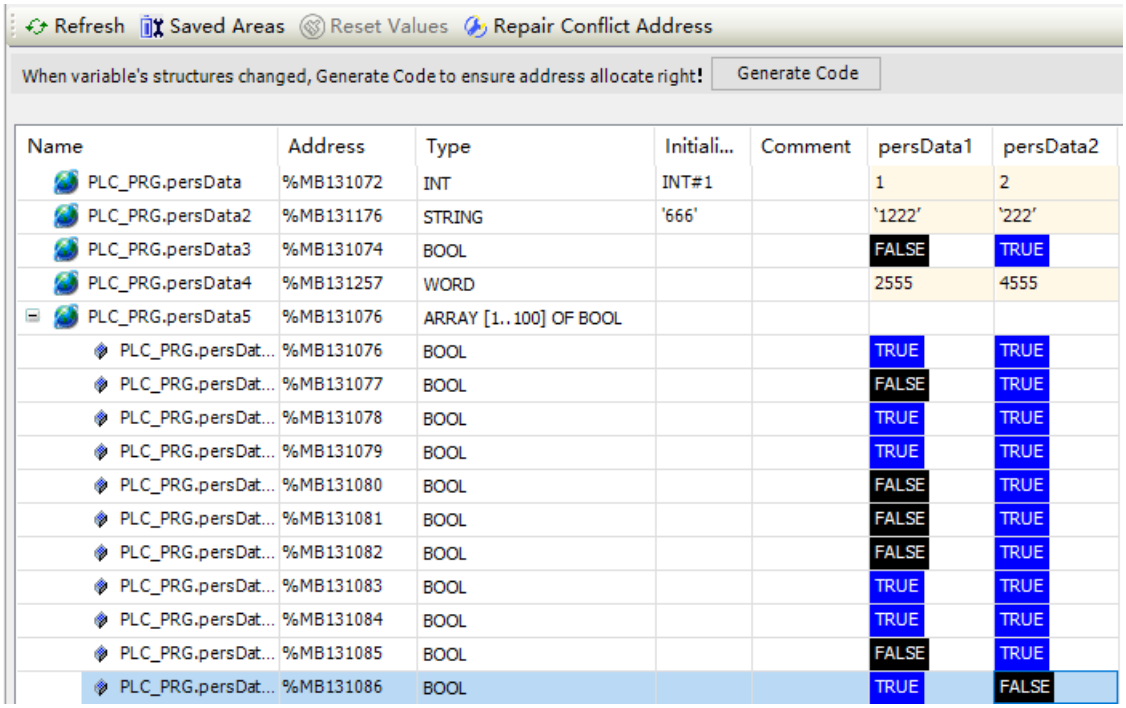
- Both RETAIN and PERSISTENT RETAIN are retain variables, but their retain characteristics are different.
- The direct variables mapped to the %M address can be declared as persistent variables, whereas the direct variables mapped to %I and %Q cannot be declared as persistent variables.

5.5.3 Persistent Variable Table

If a persistent variable is defined in the project, a persistent variable table must be generated; otherwise, the defined variable does not have the persistent function. The persistent variable table can be generated in the following two ways:

- Manually add: Right-click "Application". In the shortcut menu displayed, choose "Add Object" > "Persistent Variables" to add a persistent variable table.
- Automatically add: When a persistent variable is declared, the persistent variable table will be created automatically during compilation.

The persistent variable table has two modes: legacy mode and standard mode (recommended). The legacy mode is the old mode, and the specific use remains unchanged from the original. The persistent variable table in standard mode is shown below:



Name	Address	Type	Initiali...	Comment	persData1	persData2
PLC_PRG.persData	%MB131072	INT	INT#1		1	2
PLC_PRG.persData2	%MB131176	STRING	'666'		'1222'	'222'
PLC_PRG.persData3	%MB131074	BOOL			FALSE	TRUE
PLC_PRG.persData4	%MB131257	WORD			2555	4555
PLC_PRG.persData5	%MB131076	ARRAY [1..100] OF BOOL				
PLC_PRG.persDat...	%MB131076	BOOL			TRUE	TRUE
PLC_PRG.persDat...	%MB131077	BOOL			FALSE	TRUE
PLC_PRG.persDat...	%MB131078	BOOL			TRUE	TRUE
PLC_PRG.persDat...	%MB131079	BOOL			TRUE	TRUE
PLC_PRG.persDat...	%MB131080	BOOL			FALSE	TRUE
PLC_PRG.persDat...	%MB131081	BOOL			FALSE	TRUE
PLC_PRG.persDat...	%MB131082	BOOL			FALSE	TRUE
PLC_PRG.persDat...	%MB131083	BOOL			TRUE	TRUE
PLC_PRG.persDat...	%MB131084	BOOL			TRUE	TRUE
PLC_PRG.persDat...	%MB131085	BOOL			FALSE	TRUE
PLC_PRG.persDat...	%MB131086	BOOL			TRUE	FALSE

The following table lists options in the toolbar.

Option	Function	Description
Refresh	Add persistent variables of an external project to the persistent variable table and assign addresses to persistent variables that have not been assigned with addresses. Refresh variable recipe data structures. Detect address legitimacy of all persistent variables.	-
Saved Areas	Enable mode switching and standard mode storage area address range settings.	For area address range settings, see "*Address area setting*" *on page 377*
Reset Values	Clear the data in the device memory for persistent variables in the online state.	-
Repair Conflict Address	Reallocate addresses for persistent variables with conflict addresses.	-

The following describes relevant list items:

Item	Function	Attribute
Name	Display the source and variable name of the persistent variable.	Not editable
Address	Display the address of the persistent variable.	Editable
Type	Display the type of the persistent variable.	Not editable
Initialization	Display the initial value of the persistent variable upon declaration. In the "Initialization" column, when the menu command "current value -> initial value" is executed, the online value of the persistent variable is written to the "Initialization" column and synchronized to the persistent variable declaration in the project.	Not editable
Comment	Display the comments of the persistent variable.	Not editable
Recipe	Display and save the recipe value of the persistent variable. Added as required.	Editable

5.5.4 Persistent Rules



These rules are supported in the firmware of AC700/AC800 1.26.14.0 and later versions, and firmware of AM400/AM600 1.40.8.0 and later versions.

The rules for the variable value when the attribute related to the persistent variables is changed are as follows:

- The value of the persistent variable will be initialized to the initial value when the variable name is changed.
- The original value of the persistent variable will be retained when the variable address, initial value, comment, variable type, and attributes are changed.
- Variable type changes include changes to the variable type name, variable type member, and variable type member type.

- The value of the persistent variable will be initialized to the initial value when the type name of the persistent variable is changed.

Note

If you modify the type name of the FB function block without changing the FB function block member name, type, and instance object name, the original value of the FB member variable will be retained.

- When the type name of the persistent variable is not changed but its type member is changed:
  - Structure/function block: The original values of the unchanged members will be retained, and the values of the changed members will be initialized to their initial values.
  - Union/enumeration: The value will be initialized to its initial value when its type size decreases.
  - Array: The values of array variables will be initialized to their initial values when their base type is changed (when the base type of an array type variable is a structure and only structure members are added or deleted, the original values of unchanged structure members will be retained); when the base type of an array type variable is unchanged, the original values of the elements with the same index will be retained.

- When the copy data caused by the persistent variable exceeds 200000 lines of code, it will cause the value of the copied persistent variable to be initialized to the initial value.

5.5.5 Persistent Mode

Mode comparison

To maintain the compatibility of the project with the software version, the old editing mode of the persistent variable, that is, legacy mode, is retained in the project.

When an old project that contains "persistent variables" is opened, the system can still display and edit the project according to the legacy mode; if the original project does not have "persistent variables", the newly created persistent variables will follow the "standard mode".

At present, the persistent variables support standard mode and legacy mode (old mode), and the differences between the two are shown in the table below:

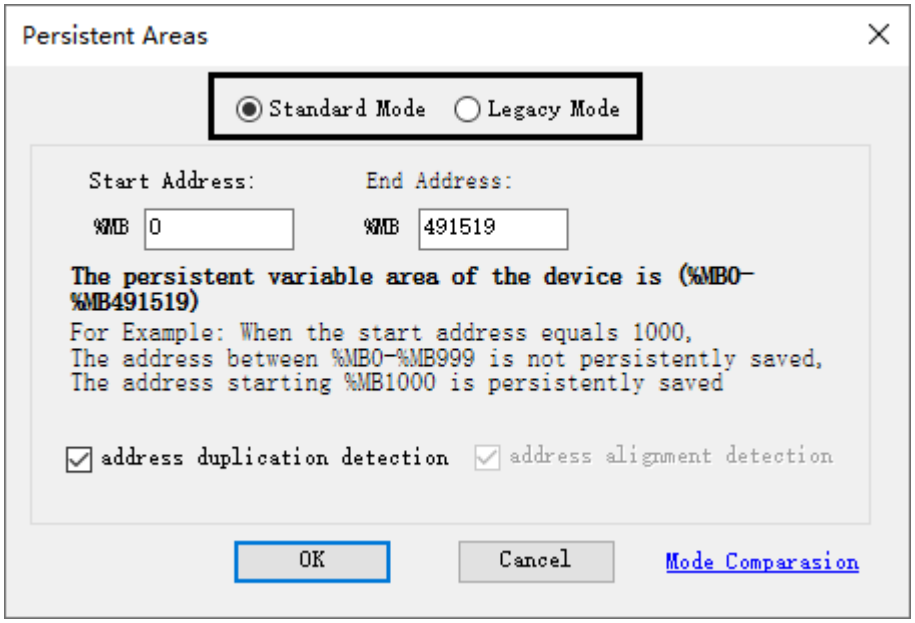
Option	Standard Mode	Legacy Mode (Old Mode)
Data source	All from variables with "Persistent Retain" attribute outside the persistent variable table or assigned to the M-area persistent area.	Variables and variables with "Persistent Retain" attribute outside the table are defined inside the persistent variable table.
Address mapping	All variables need to be strongly correlated to the persistent area of the M area.	The variables do not need to be strongly correlated to the persistent area.
Intermediate insertion or deletion of variables	Variables are mapped by address, so insertion or deletion of a variable has no effect on the saved values of other variables.	Accidental operations may cause data loss (such as clearing all data, PLC device update, and compilation option modification).
Recipe function	The recipe function is built-in.	Only external recipe function can be used.

Note

In the simulation mode, if the persistent variables are in the standard mode, the variable value will be initialized after the "cold reset" command is executed.

Switch from legacy mode to standard mode

On the "Persistent Areas" tab page in the toolbar, you can switch the mode for persistent variables. The following figure shows a mode switching page.



When the mode is changed from "Legacy Mode" to "Standard Mode", user-defined persistent variables in the legacy mode will be saved to the newly generated GVL program. When you compile the program again or click "Refresh", the system adds the persistent variables in GVL to the data table of the standard mode and assigns addresses for the variables.

Note

If the mode is switched to "Legacy Mode" again, the original data remains unchanged. However, the newly generated GVL program is not deleted. Therefore, when the mode is switched to "Legacy Mode" again, you need to delete the newly generated GVL program manually.

Mode compatibility

For the persistent modes, the compatibility of the PLC firmware and project versions is as follows:

1. If the PLC supports persistent mode switching, after the project is downloaded, the persistent mode of the PLC is automatically changed to that set in the project.

Note

If the project version does not match the PLC mode, the PLC automatically switches the persistent mode when the project is downloaded for the first time, which may result in undesired data.

The following table lists PLC firmware versions supporting the persistent mode.

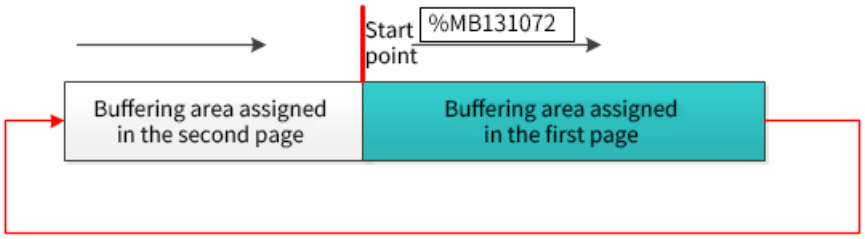
PLC Model	Version Restriction
AM600	1.24.20.0 (included) and later versions
AM401	21.24.20.0 (included) and later versions
AM402	41.24.20.0 (included) and later versions
AM403	81.24.20.0 (included) and later versions
AP700	1.13.30.0 (included) and later versions
AC810/AC801/AC802	1.13.30.0 (included) and later versions

2. If the PLC does not support persistent mode switching, you need to upgrade the PLC firmware or switch the project persistent mode to the legacy mode.
   Jump to the mode switching page: Enter the persistent variable table and open the mode switching page.

5.5.6 Address Assignment

If a persistent variable is defined in the user program but is not assigned with an address, the system will automatically assign an address to the variable by clicking "Compile" in the toolbar or "Refresh" on the persistent page in the standard mode.

Considering the user's Modbus address usage range, the initial address assignment will avoid its regular usage area (%MB0 to %MB131071) and start from %MB131072. Only when the end address is used up or no more address can be assigned to the variable, then the address will be assigned from the %MB0 to %MB131071 range from the start.



For example, the maximum address available is %MB50000, and you want to assign an address for Var, a Real type variable:

Since the space size of Real type variable is 4 bytes, the start address is calculated as follows. First select the address "%MB131072" as the start address from the range of "%MB131072 to %MB50000", and then detect the conflict with the existing assigned addresses. If an address conflict occurs, the address is discarded, the address is recalculated from the next available address, and the test is performed again until no address conflict occurs. If a complete area cannot be found in the range "%MB131072 to %MB50000" to save the variable, then start searching in the range of "%MB0 to %MB1310721" and select "%MB0" as the start address. Perform address conflict detection in the same way, until the legitimate address is found.

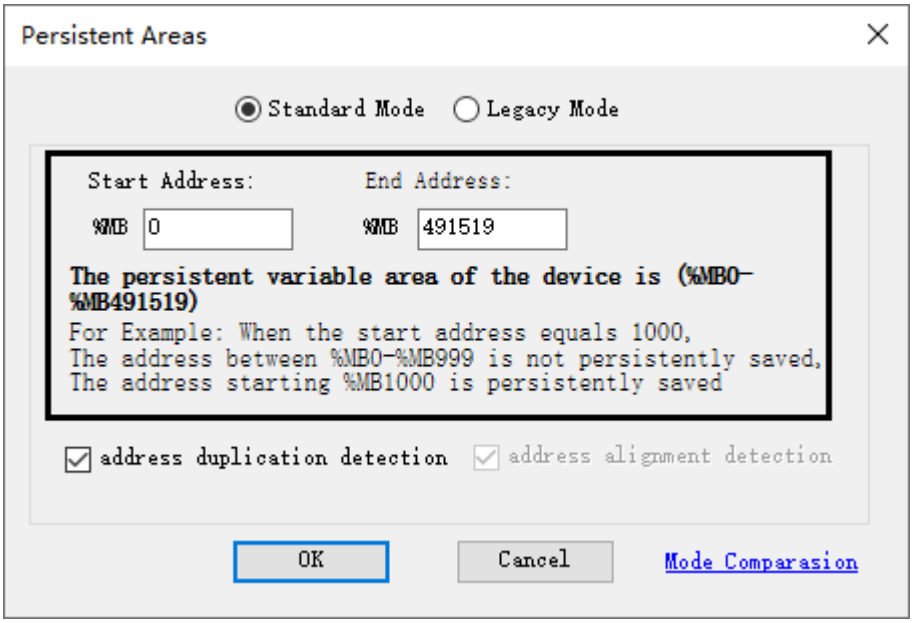
For address assignment of variables, there are several data types that need to satisfy the four-byte alignment principle, which means that the initial address of the variable is divisible by four. Variable types include:

- User-defined data structures.
- Enumeration type.
- Real or LReal type.

- Array types where the base type data is an array of the above three types.

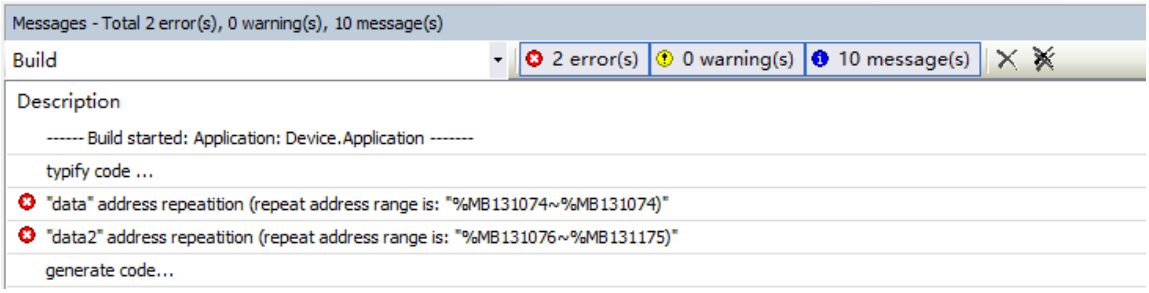
Address area setting

The address assignment area is the M area of PLC, and the specific address range can be viewed and set through the "Persistent Areas" option in the toolbar, as shown in the following figure.

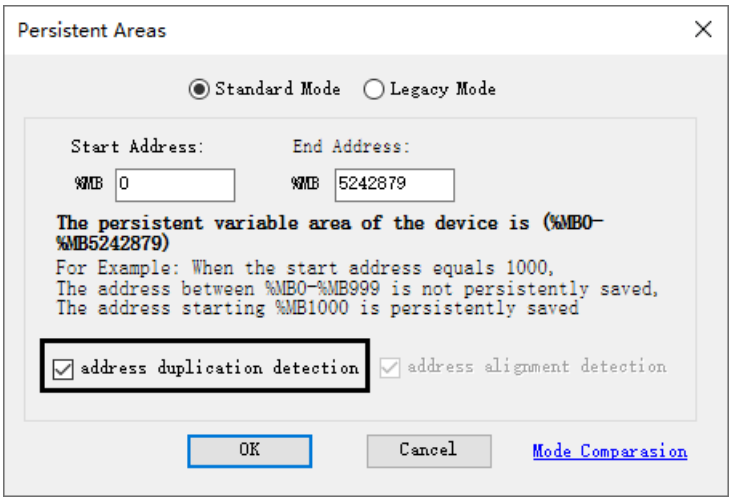


Address duplication check

In the standard mode, to ensure the correctness of the data saving of persistent variables, in principle, each variable should correspond to a unique M-area address. By default, the system will perform address duplication check for all persistent variables. When two or more variables have conflict addresses, an error will be indicated in the output window.



If you want duplicated variable addresses, on the "Persistent Areas" tab page in the toolbar, deselect "address duplication detection", as shown in the following figure.



Repair conflict addresses

When the persistent variable addresses are conflicting, you can manually modify the variable addresses. Besides, you can also repair address conflict on the "Conflict Address Repair" tab page of the toolbar.

The table in the preceding figure displays all variables with conflict addresses. Columns of the table indicate the variable name, address, type, address range, and suggested address, respectively.

The suggested address is an address repair function for a single variable. To repair all addresses, you can click "Repair All".

Delete variable addresses in batch

When you need to adjust the persistent area range or want to reassign all variable addresses because there are too many conflict addresses, you can click "Clear All Variables Address" in the shortcut menu to clear all variable addresses, and then assign addresses for all variables again through compilation.

Note

When the persistent variable data has been downloaded to the PLC, modification of the address of a persistent variable that already exists in the PLC, either manually or through an operation such as address repair, can result in the data saved at its original address not being used. Also, if data exists at the newly assigned address, that data will be applied to the newly associated variable, and unforeseen problems may occur. To ensure the validity of subsequent data, the initial value must be initialized.

Addresses needing adjustment are as follows:

- Array size change

```
VAR PERSISTENT RETAIN
    dataArray AT %MB131072 : ARRAY [1..100] OF INT;
    data AT %MX131272.0 : BOOL;
END_VAR
```

The array variable "dataArray" has an initial range of 1 to 100 and is assigned the address %MB131072. The bool type variable "data" is assigned the address %MX131272.0. The addresses of the two variables are consecutive. When the range of the "dataArray" variable is adjusted to 1 to 200 because of practical needs, the start address is not changed.

```
VAR PERSISTENT RETAIN
    dataArray AT %MB131072 : ARRAY [1..200] OF INT;
    data AT %MX131272.0 : BOOL;
END_VAR
```

At this time, the address range of the "dataArray" variable is %MB131072 to %MB131471, which is duplicated with the address %MX131272 of the data variable.

- Data structure member change
  Define the data structure variable "dataDUT", where the data structure of DUT is shown below:

```
VAR PERSISTENT RETAIN
    dataDUT AT %MB131072 : DUT;
    dataBool AT %MX131158.0 : BOOL;
END_VAR
```

```
TYPE DUT :
STRUCT
    subData1:INT;
    subData2:WORD;
    subData3:STRING;
END_STRUCT
END_TYPE
```

Initially, the DUT contains three members with a start address of %MB131072. When subsequent program adjustments are made and two member variables are added to the DUT data members, the variable "dataDUT" actually takes up more space and conflicts with the address of the subsequent dataBool variable.

```
TYPE DUT :
STRUCT
    subData1:INT;
    subData2:WORD;
    subData3:STRING;
    subData4:BOOL;
    subData5:BOOL;
END_STRUCT
END_TYPE
```

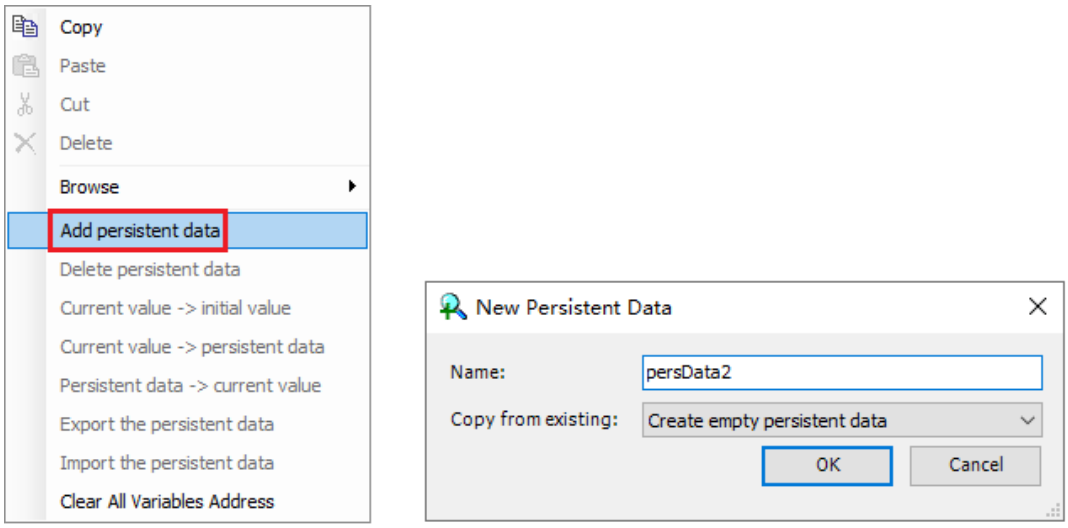
5.5.7 Recipe Operations

Recipes can hold data for a set of variables and write values as persistent variables.

In the persistent standard mode, you can right-click a recipe entry in the persistent variable table to perform operations such as "Add persistent data", "Delete persistent data", "Current Value -> persistent data", "Persistent data -> current value", "Export the persistent data", and "Import the persistent data".

Add a recipe

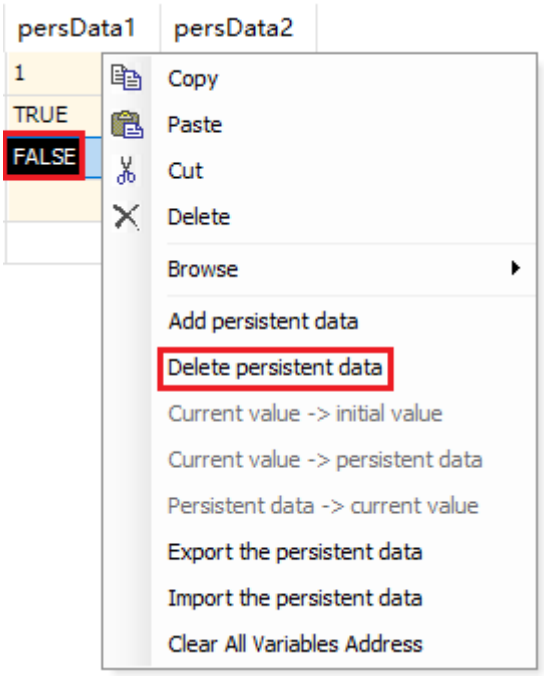
In the persistent variable table, select "Add persistent data" from the short-cut menu.



- Function: This operation is used to add a new recipe column next to the last column of the persistent variable table. In the "New Persistent Data" dialog box displayed, set a name for the new recipe column or copy an existing name.
- Enabling condition: Persistent variable tables in the standard mode are available.
- Note: When you add a recipe for the first time, click "Refresh" first; otherwise, a dialog box is displayed, prompting you to refresh the data.

Delete a recipe

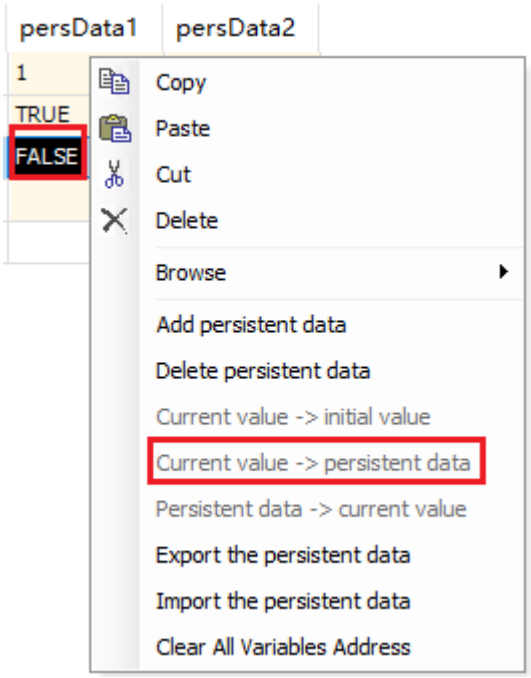
In the persistent variable table, right-click a recipe and select "Delete persistent data" from the shortcut menu displayed.



- Function: This operation is used to delete the selected recipe.
- Enabling condition: Persistent variable tables in the standard mode are available and a recipe is selected.

Current value -> persistent data

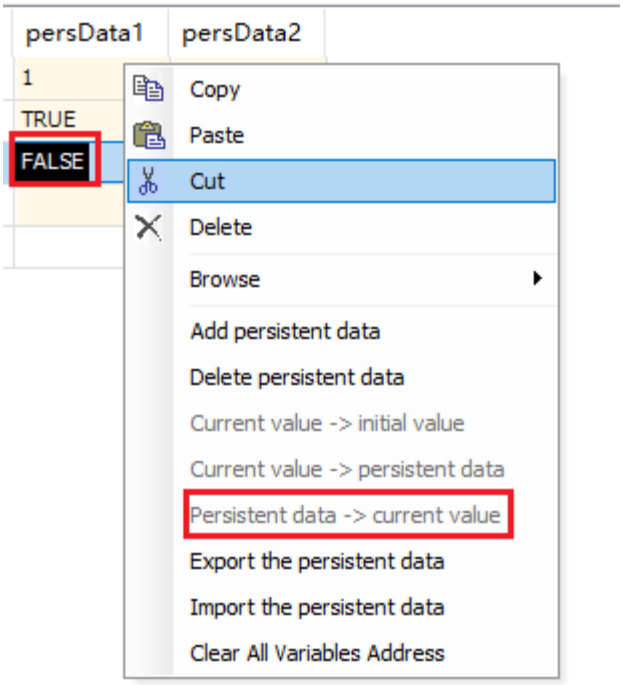
In the persistent variable table, right-click a recipe and select "Current value -> persistent data" from the shortcut menu displayed.



- Function: This operation is used to save the online value of the persistent variable to the selected recipe.
- Enabling condition: The project is logged in, persistent variable tables in the standard mode are available, and a recipe is selected.
- Note: The data structure in the initial value column must be consistent with that in the recipe column; otherwise, a dialog box is displayed, prompting you to refresh the data.

Persistent data -> current value

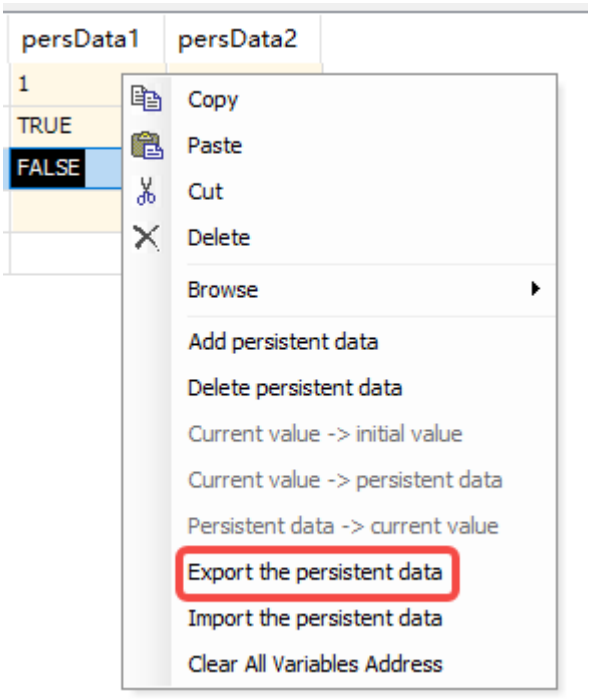
In the persistent variable table, right-click a recipe and select "Persistent data -> current value" from the shortcut menu displayed.



- Function: This operation is used to write the recipe value of the persistent variable in the selected recipe to the online value column.
- Enabling condition: The project is logged in, persistent variable tables in the standard mode are available, and a recipe is selected.
- Note: The data structure in the initial value column must be consistent with that in the recipe column; otherwise, a dialog box is displayed, prompting you to refresh the data.

Export the persistent data

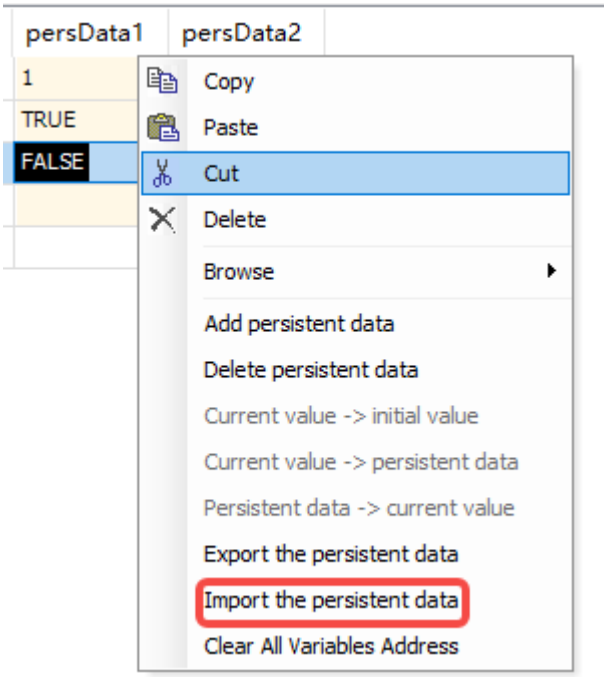
In the persistent variable table, right-click a recipe and select "Export the persistent data" from the shortcut menu displayed.



- Function: This operation is used to export the value of the selected recipe to a new file with the suffix ".txtrecipe", and save this file to the specified position.
- Enabling condition: Persistent variable tables in the standard mode are available, and a recipe is selected.

Import the persistent data

In the persistent variable table, right-click a recipe and select "Import the persistent data" from the shortcut menu displayed.



- Function: This operation is used to import the recipe value in the selected file (with the suffix ".txtrecipe") at the specified position to the column of the selected recipe.
- Enabling condition: Persistent variable tables in the standard mode are available, and a recipe is selected.
- Note: Variables in the recipe to be imported must be in the persistent variable table; otherwise, a mismatch prompt is displayed.

5.5.8 Description

Standard mode

In the standard mode, when the variable structure changes and the variable start address is %MB131072 (Modbus common address range is %MB0 to %MB131071), the system will save the existing data of the original persistent variable and initialize new variables. Example:

Define the data structure persistent variable "dtData" as follows.

```
VAR_GLOBAL PERSISTENT RETAIN
    dtData AT %MB131072 : DUTData;
END_VAR
```

The data structure of the variable "dtData" is shown below.

```
TYPE DUTData :
STRUCT
    perData:INT;
    perData1:BOOL;
    perData2:WORD;
END_STRUCT
END_TYPE
```

After downloading the project to the PLC, the system stores the data of the variable "dtData" upon power-down.

When a new BOOL type variable "perData3" is added to "DUTData" and the latest program is downloaded for the first time, the data of this structure variable is changed, as shown below.

```
TYPE DUTData :
STRUCT
    perData:INT;
    perData1:BOOL;
    perData2:WORD;
    perData3:BOOL;
END_STRUCT
END_TYPE
```

Original persistent data retained

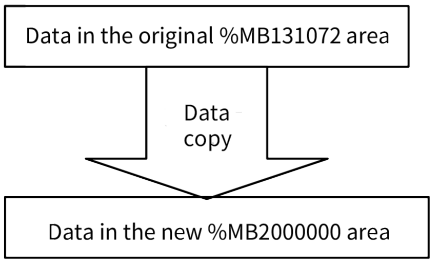
Data initialized upon first download

Variable address change

When the address of the persistent variable changes, the system copies the data from the original address to the new address without affecting the persistent variable data change.

```
VAR_GLOBAL PERSISTENT RETAIN
    dtData AT %MB131072 : DUTData;
END_VAR
```

```
VAR_GLOBAL PERSISTENT RETAIN
    dtData AT %MB2000000 : DUTData;
END_VAR
```



FB persistent variable

FB-type variables do not support persistent characteristics, but FB child member variables can have persistent characteristics.

In the following function block "FBData", sub-member variables "fbData", "fbData1", and "fbData2" can have the "PERSISTENT RETAIN" characteristic. However, the FBData-type variable "fbVar" does not support the "PERSISTENT RETAIN" characteristic.

```
FUNCTION_BLOCK FBData
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR PERSISTENT RETAIN
    fbData:INT;
    fbData1:BOOL;
    fbData2:STRING;
END_VAR
```

```
VAR_GLOBAL
    fbVar:FBData;
END_VAR
```

6 Programming Languages

6.1 Programming Languages Supported by InoProShop

The programming software supports the following PLC programming languages:

· Ladder diagram (LD)

· Function block diagram (FBD)

· Structured text (ST)

· Sequential function chart (SFC)

· Continuous function chart (CFC)

The LD, FBD, ST, and SFC are based on the IEC 61131-3 standard, and CFC is an expansion of the IEC 61131-3 standard.

The basic edit methods on the programming page are applicable regardless of the selected language, which greatly facilitates programming.

- Functions of the standard editor are supported, such as the copy (Ctrl+C), paste (Ctrl+V), and delete (Del) shortcut keys.
- The standard Ctrl and Shift keys can be used to select multiple options.
- The function key F2 can be used to start the input assistant. The system provides input tips or options based on the specific environment.

6.2 Structured Text (ST)

6.2.1 Overview

The ST is a text-based advanced language and similar to PASCAL or C. The program code consists of instructions comprising keywords and expressions. Different from the IL, the ST can include multiple statements during a statement cycle, allowing for complex structure development.

Example:

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

6.2.2 Expressions

An expression is a type of structure. Its calculated value can be used in instructions.

An expression consists of operators and operands. An operand can be a constant, a variable, a function call, or other expressions. Example:

- Constant, such as 20, t#20s, and 'string'
- Variable, such as iVar and Var1[2,3]
- Function call, whose value is the return value of a call, such as Fun1(1,2,4)
- Other expressions: such as 10+3, var1 OR var2, (x+y)/z, and iVar1:=iVar2+22

In an expression, operands are evaluated using operators in sequence defined by a particular operator priority. In an expression, operands are evaluated using operators in sequence defined by a particular operator priority. Operators with top priority must be first used for evaluation. Other operators with lower priority are used by priority in descending order. Operators with the same priority must be used in order from left to right in the expression.

For example, if A, B, C, and D are INT variables and are set to 1, 2, 3, and 4, respectively, A+B-C*ABS(D) must be –9 and (A+B-C)*ABS(D) must be 0.

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

The following table lists the operators of the ST language.

Operation	Symbol	Priority
Parentheses	(Expression)	High ↓ Low In descending order
Function call	Function name (parameter list, separated by commas)	
Exponentiation	EXPT	
Negation value calculation	-	
Complementing	NOT	
Multiply	*	
Divide	/	
Modulo	MOD	
Plus	+	
Minus	-	
Compare	<, >, <=, >=	
Equal to	=	
Not equal	<>	
AND	AND	
XOR	XOR	
OR	OR	

6.2.3 ST Instruction

The overall ST program consists of instructions separated by semicolons (;). The instructions consist of keywords and expressions. The following table lists the ST instructions.

Instruction	Description	Example
:=,S=,R=	Assign value, set, and reset	A:=B; C S= cond0; b1 R=cond1;
Function block call	Function block call and output	CMD_TMR :TON (CMD_TMR.Q is the timer output state) CMD_TMR(IN := %IX5, PT := 300); A:=CMD_TMR.Q
RETURN	Return (exit the current POU)	RETURN;

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

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

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

WHILE	WHILE loop	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	REPEAT loop	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT loop	EXIT;
CONTINUE	Continue the next execution loop	CONTINUE;
JMP	Jump	label: i:=i+1; JMP label;
;	Empty statement	;

Valuation

A valuation instruction assigns values to variables. In a valuation keyword, the left part is a variable and the right part is the value to be assigned by the keyword. Three types of keywords are provided: ":=", "S=", and "R=".

- ":=" indicates general valuation, in which the right-side value is directly assigned to the left-side value and the two values are equal.
  Example: Var1 := Var2 * 10;

  After execution, the value of Var1 is 10 times that of Var2.
- "S=" indicates set valuation, in which the left-side variable is changed to "TRUE" (set) if the right-side value is "TRUE", until it is initialized using the R= instruction.
- "R=" indicates reset valuation, in which the left-side variable is changed to "FALSE" (reset) if the right-side value is "TRUE". It is used to reset the variables set by the S= instruction.
  Example: a S= b;

  Once the value of b changes to "TRUE", the value of a remains "TRUE", even after the value of b changes to "FALSE".

Function block calls

Syntax: <FB instance name>(FB input variable:=<value and address>|, <more FB input variables:=<value and address>|...more FB input variables);

Call syntax: A delay function block (TON) is called, and the IN and PT parameters are assigned. The result variable Q is assigned to variable A.

Note: TON is instantiated through "TMR:TON".

Instantiation syntax: <FB instance name> :<FB variable >;

TMR(IN := %IX5, PT:= T#300MS, Q=> q1, ET=>et1);

A:=TMR.Q;

RETURN

The RETURN instruction indicates exiting the POU when the predefined condition is "TRUE".

Syntax:

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 POU will be immediately returned.

IF

The IF keyword is used to determine the condition for executing instructions.

Syntax:

IF <boolean expression1> THEN

<IF_instruction>

{ELSIF <boolean expression2> THEN

<ELSIF_instruction1>

ELSIF <boolean expression n> THEN

<ELSIF_instruction-1>

ELSE

<ELSE_instruction}

END_IF;

The content inside {} is optional.

If <boolean expression 1> is "TRUE", only <IF_instruction> is executed whereas other instructions are not; otherwise, the boolean condition expressions starting from <boolean expression 2> are calculated one by one until the value of an expression is "TRUE". Then, the instructions of the expression are executed. If no expression has the value "TRUE", the instruction corresponding to <ELSE_instruction> is executed.

Example

IF temp<17

THEN heating_on := TRUE;

ELSE heating_on := FALSE;

END_IF;

Here, heating starts when the temperature is below 17°C; otherwise, heating remains disabled.

CASE

The CASE instruction lists and processes the instructions corresponding to the multiple values of a conditional variable. Conditional variables must be integers.

Syntax:

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;

The CASE instruction implements the following processing:

- If the value of the <Var1> variable is <valueI>, <Instruction I> is executed.
- If no value matches with <Var1>, <ELSE Instruction> is executed.
- If the same instruction is executed in multiple variable values, you can write the values one by one and separate them with commas (,) so that they are executed simultaneously.
- If the same instruction is executed within a variable range, you can write the start and end values and separate them with two periods (.).

Example

```
CASE iVar1 OF
    2:c1:=c1+1;
    1,6:c1:=c1-7;
    7..20:c1:=c1+5;
    ELSE
        c1:=c1-1;
END_CASE
```

FOR loop

The FOR loop can be used to compile the iterative processing logic.

Syntax:

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

<instructions>

END_FOR;

The content inside {} is optional.

INT_Var is a counter of the integer type. <Instructions> is executed as long as <INT_Var> is not greater than <END_VALUE>. Check the condition before executing <Instructions>. <Instructions> is not executed if <INIT_VALUE> is greater than <END_VALUE>.

<INT_Var> automatically increases by <Step size> each time after <Instructions> is executed. <Step size> can be any integer. The default value 1 applies if this parameter is not set. 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;

Assume that the default value of Var1 is 2. The value changes to 32 after FOR loop.

WHILE loop

Like the FOR loop, the WHILE loop can be used by loop processing. Different from the FOR loop, the WHILE loop supports loop conditions of arbitrary boolean expressions. The loop is executed once the loop condition is met; otherwise, it is exited.

Syntax:

WHILE <boolean expression> DO

<instructions>

END_WHILE;

When <Boolean_expression> is "TRUE", <Instructions> is executed, until <Boolean_expression> changes to "FALSE". <Instructions> is never executed if the initial value of <Boolean_expression> is "FALSE". If <Boolean_expression> is always "TRUE", <Instructions> is executed without stop. This results in infinite loop, which is not allowed during programming.

Example:

WHILE Counter<>0 DO

Var1:= Var1*2;

Counter := Counter-1;

END_WHILE

In a sense, the WHILE loop and REPEAT loop functions are more powerful than the FOR loop function because cycle times do not need to be counted before loop execution. Therefore, only the WHILE loop and REPEAT loop are required in some cases. However, if the cycle times is clear, the FOR loop is better.

REPEAT loop

Different from the WHILE loop, the REPEAT loop checks the loop condition after the loop instruction is executed. This means the loop is executed at least once, regardless of the loop condition.

Syntax:

REPEAT

<instructions>

UNTIL <Boolean expression>

END_REPEAT;

The execution logic is as follows:

<Instructions> is executed until <Boolean expression> is "TRUE". <Instructions> is executed only once if the initial value of <Boolean expression> is "TRUE". If the value of <Boolean_ expression> is always "FALSE", <Instructions> is executed permanently, resulting in infinite loop.

Example:

REPEAT

Var1:=Var1*2;

```
Counter:=Counter-1;
UNTIL Counter=0;
END_REPEAT;
```

CONTINUE

The CONTINUE instruction is used to end a FOR, WHILE, or REPEAT loop in advance and start the next loop.

Example:

```
FOR Counter:=1 TO 5 BY DO
INT1:=INT1/2;
IF INT1=0 THEN
CONTINUE;
END_IF
Var:=Var1/UBT1L
END_FOR;
Erg:=Var1;
```

EXIT

The EXIT instruction is used to exit the FOR, WHILE, or REPEAT loop.

JMP

The JMP instruction is used to jump to the code line marked by the specified label.

Syntax:

```
<label>:
JMP <label>;
```

<label> is at the start of the program line. The JMP instruction requires a jump target, that is, a predefined label. When the JMP instruction is reached, the program jumps to the code line marked by the specified label for execution.

Example:

```
aaa :=0;
_label11:aaa:=aaa+1;
(*instructions*)
IF (aaa < 10) THEN
JMP _label1;
END_IF;
```

The initial value of the variable "aaa" is 0. As long as the value is less than 10, the program jumps to the code line marked by label 1 for execution. This affects the repeated execution of the program between the JMP instruction and label.

Such a function can also be implemented by the WHILE or REPEAT loop. Be cautious when using the JMP instruction because it reduces code readability.

Comment

Structured text can be commented in two ways:

- Single-line comment: Start with "//" and end with "//", for example, "// This is a comment.".
- Multi-line comment: Start with "(*" and end with "*)", for example, "(*This is a comment.*)".

Comments can be added to the ST editor declaration or any part in implementation.

Comment nesting: A comment can be inserted into another comment.

Example:

(*

a:=inst.out; (*to be checked*)

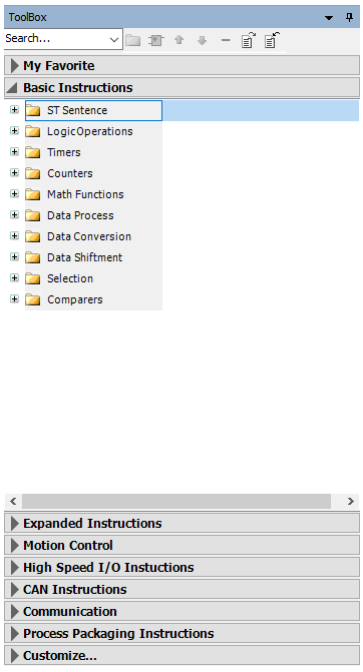
b:=b+1;

*)

6.2.4 ST Editor

6.2.4.1 ST Tool Kit

The following figure shows the "ToolBox" page.



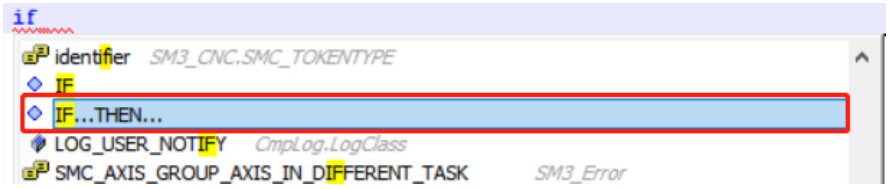
This page provides functions such as ST statement, logic operation, timer, counter, math function, data process, data conversion, and data shiftment. You can drag an ST, IF, FOR, WHILE, REPEATED,

CASE, CONTINUE, JMP, EXIT, or RETURN statement and drop it in the programming area, and then the statement template is automatically inserted.

6.2.4.2 Intelligent Input

- Keyword matching
  After you input ST statement keywords, then IF, WHILE, FOR, CASE, and REPEATED statements containing the keywords are automatically matched. For the formatting template, see appendix statement template.

  As shown in the figure below, after entering IF, statements containing "if" are displayed.



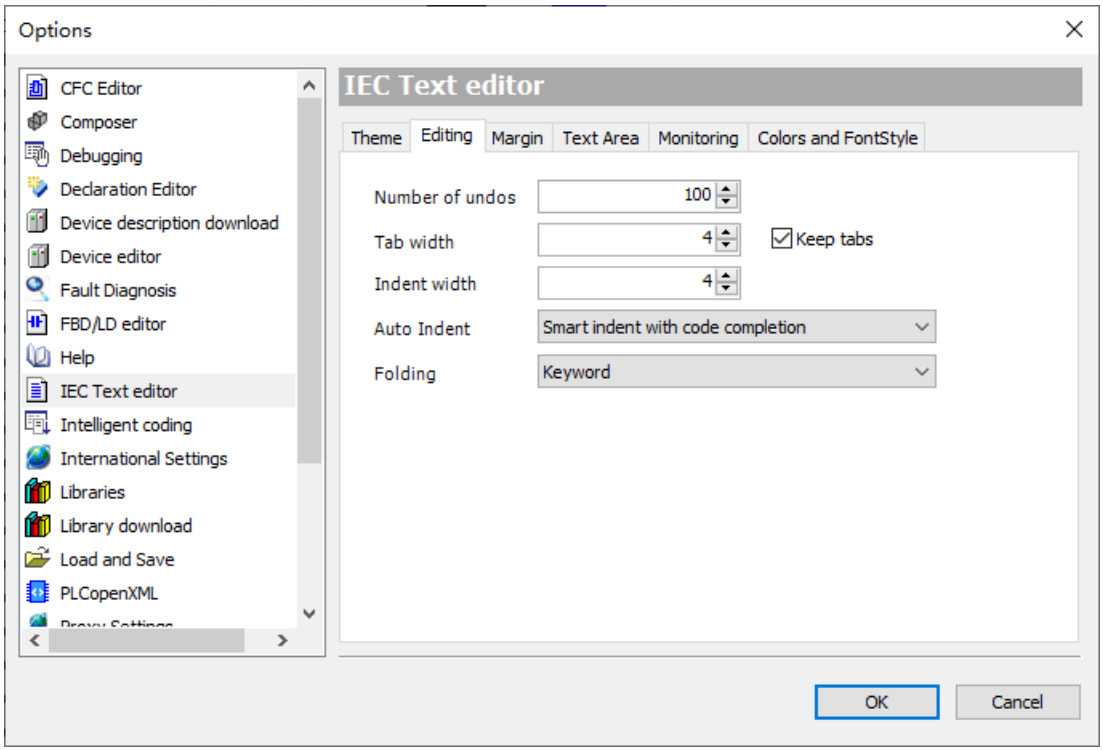
- Shortcut functions of the Tab key
  - Automatically format input and output for function blocks, functions, methods, actions, and programs.
  - Automatically format input and output for function block instances and their methods and actions.
  - Automatically format IF, WHILE, FOR, CASE, and REPEATED statements as well as the input function type names and function block instances. For the formatting template, see appendix statement template.

Note

Format requirements: When an input is made with the Tab key, the part from the beginning of the line to the cursor position is used as a keyword as a whole. If no statement matches with the keyword, it will not be automatically complemented.

6.2.4.3 Folding and Indenting Functions

- Folding can be performed with keywords such as VAR, VAR_INPUT, VAR_GLOBAL, VAR_OUTPUT, VAR_IN_OUT, VAR_TEMP, VAR_STAT, VAR_ESTERNAL, CASE, FOR, REPEATED, IF/ELSE/ELSIF, WHILE, STRUCT, UNION, TYPE, __TRY, __CATCH, and __FINALLY.
- If "Smart indent" is selected for "Auto indent", the Tab length increases automatically based on the keyword. If "Smart indent with code completion" is selected for "Auto indent", an end such as VAR, FOR, and WHILE is automatically added to the keywords. Keywords can be nested.
- In case of "Smart indent", if the upper line is a keyword, the Tab character is automatically added after a line break, and if the upper line is not a keyword, the next line is indented the same as the upper line.
- The block is highlighted. Block highlighting information is displayed between categories such as brackets, parentheses, WHILE, FOR, IF, ElSE, CASE, REPEAT, STRUCT, UNION, TYPE, and TRY, with highlighting markers at text boundaries and in text areas.



6.2.4.4 Page Colors of IEC Text Editor

Each color on the ST page indicates a template. You can set the color through the template or choose "Tools" > "Options" > "IEC Text editor" > "Theme" to set the color. The color configurations include basic configuration, IEC61131 type and identifier configuration, online configuration, and font style configuration.

You can make configuration as needed. The following table lists relevant parameters.

Type	Description
Basic configuration	Mainly set the basic color of the page, mainly containing: Background color Foreground color (default text color) Line highlight color Text block highlight color Symbol highlight color Cursor color Background color of the selected text in focus state Background color of the selected text in non-focus state Default border text color Border background color Border expansion background Border line highlight color Focus state split line Non-focus state split line Folding state color Incremental search color

Type	Description
IEC61131 types and identifiers	Mainly set the data type and identifier color, including BOOL type constants Time type constants Integer type constants Floating point type constants String type constants Comment type Attribute type Direct Addresses Global variables Static variables Input variables Output variables Input/output variables Constants Temporary variables Persistent variables Retain variables External variables Configuration variables FB Methods Actions Functions Structures Enumeration types Enumerated values Unions Interfaces Operators Keywords Errors
Online configuration	Monitoring box background and text Streaming monitoring box background and text
Font styles	Data type and identifier font format, such as bold and underline

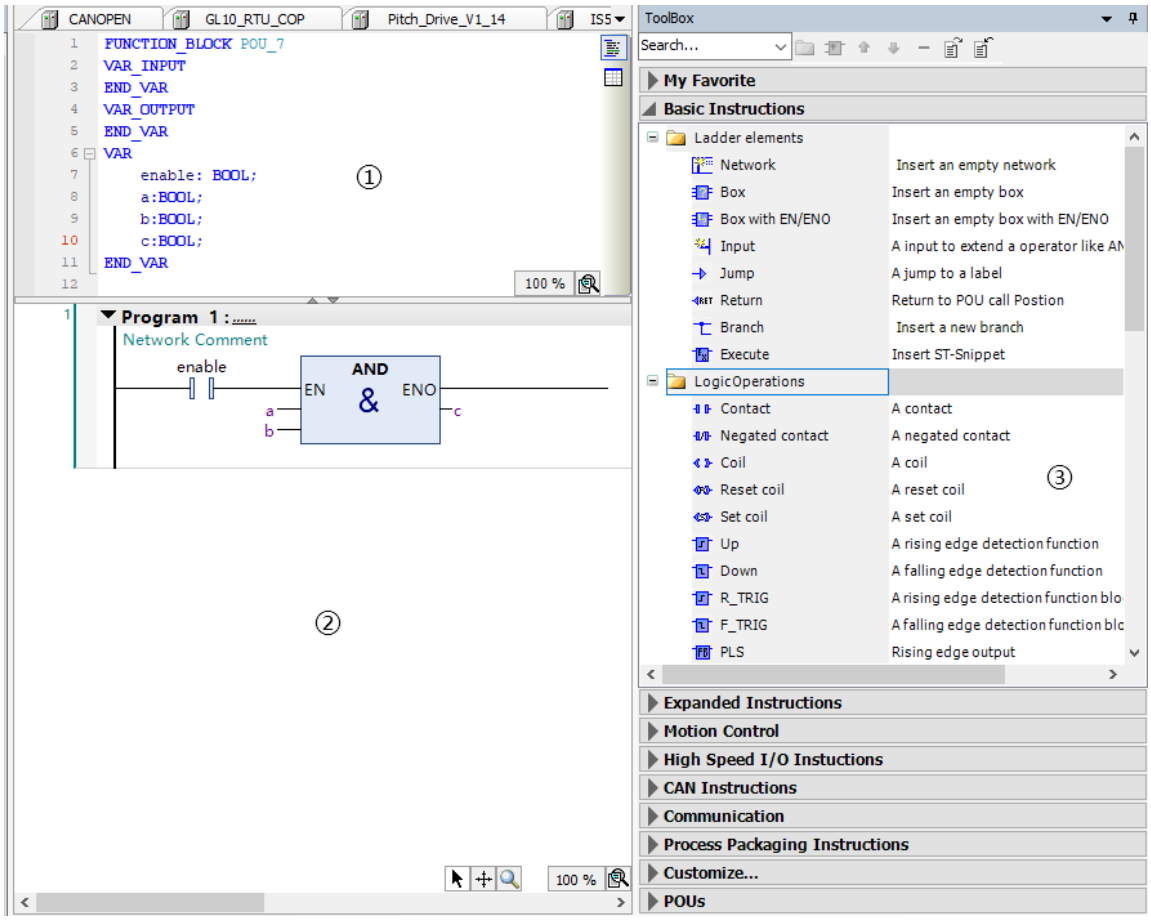
Persistent, retain, and constant types and variables are modifier relationships, and only one of the two can be selected. The priority relationship is as follows: global variables, input, output, input/output persistent (retain) local, temporary constants static variables, configuration variables, external variables, and enumerated variables support color settings. Since persistent and constant are attributes and are juxtaposed with variable types, they need to be displayed in priority control.

6.3 Ladder Diagram (LD)

6.3.1 Overview

LD is a graphical programming language. Its structure is similar to that of the circuit diagram. LD contains a series of networks (also called nodes), and each network starts from the vertical line on the left (the power rail and power flow line). A network consists of contacts, coils, operation blocks (functions, function blocks, programs, execution blocks, actions, and methods), jump instructions, labels, and connection lines.

The bus on the left of the network is the power flow line and is always "TRUE". Contacts, operation blocks, and coils are connected after the bus. Each contact is allocated with a boolean variable. If the variable is set to "TRUE", the switch is turned on and the condition is transferred along the connection line from left to right. If the variable is not "TRUE", the switch is turned off. The coil on the right of the network receives the "On" or "Off" signal transmitted from the left side. "TRUE" or "FALSE" is written to the boolean variable associated with the coil. The following figure shows the LD edit page.

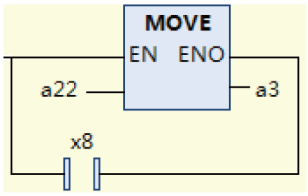


Note

- ① - variable definition area; ② - LD programming area; ③ - toolbox
- LD supports EXECUTE block nesting used to insert the ST snippet.

LD mainly includes contacts, coils, operation blocks, branches, and comments. These elements are inserted, dragged, scribed, and copied and pasted to networks to form the LD execution logic. To set the page font, operands, and comment display of the LD, choose "Tools" > "Options" > "FBD/LD editor".

LD provides online commissioning functions such as monitoring, written values, force values, and breakpoints.



6.3.2 LD Elements

LD elements include network, contact, coil, operation block, execution block, branch, jump, label, and return.

The input and output of contacts, coils, and operation blocks are related to operands, which can be variables, constants ("TRUE" or "FALSE"; 1 or 2), and addresses. For details, see the variable definition.

LD elements are displayed under "ToolBox" (choose "View" > "ToolBox"), as shown in the following figure. The toolbox includes general elements, LD elements, IEC standard operators (such as boolean operators and math operator), function blocks, and POU defined in the current program.

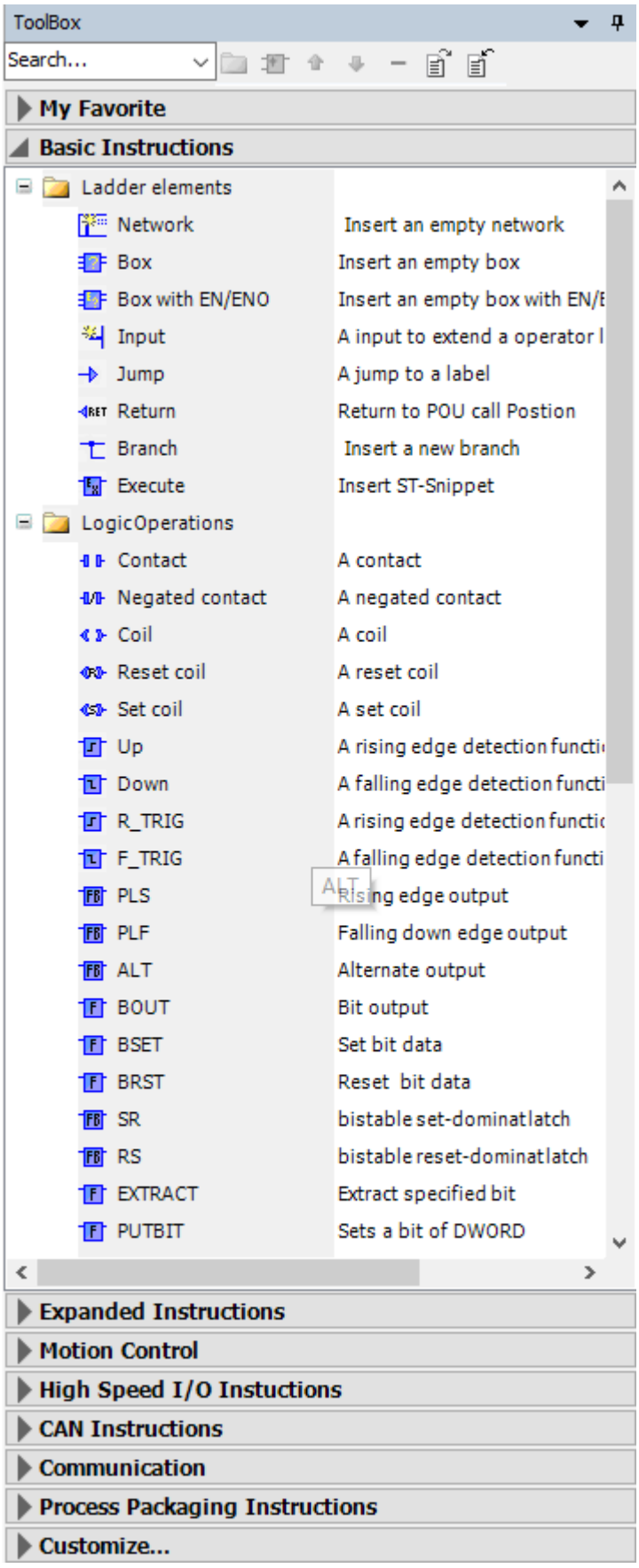


Figure 6-1 LD toolbox

Network

Icon -

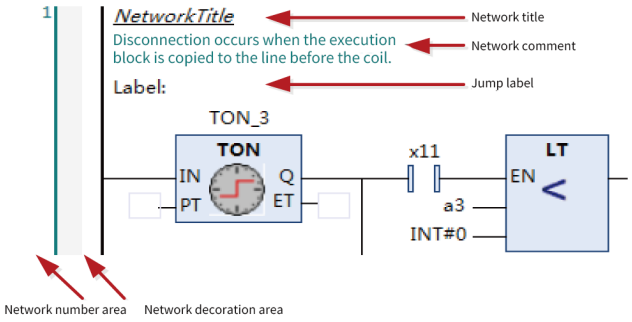
The LD consists of a series of networks. All the other LD elements are within networks. Each network is indicated by a serial number on the left.

You can insert a network title (summary about the network) and comment (detailed description about the network). Choose "Tools" > "Options" > "FBD/LD editor" > "General" to show or hide the network title and comment.

You can insert a label below the network title and comment to indicate the jump target.

Use the menu command "Toggle network comment state" to enable or disable the network.

The network decoration area between the network serial number and network content displays the breakpoint mark and bookmark position.



Contact

Icon -

Contacts are classified into normally open (NO) and normally closed (NC) contacts. Contacts are boolean variables and used to transfer "ON (TRUE)" and "OFF (FALSE)" values. If the variable value is "TRUE", the NO contact transfers "ON (TRUE)" to the right; otherwise, it transfers "OFF (FALSE)". The NC contact transfers reverse values.

You can add the edge signal function to contacts. Right-click a contact and select "Edge Detection" from the shortcut menu to change the contact to a rising-edge trigger contact or a falling-edge trigger contact. The rising-edge trigger contact transfers "ON" to the right when the variable value of the contact changes from "FALSE" to "TRUE". The falling-edge trigger contact transfers "ON" to the right when the variable value of the contact changes from "TRUE" to "FALSE".

Coil

Icon -

Coils are located at the end of the network. The logical operation results on the left are assigned to coil variables. Coil variables are only of the BOOL type. "TRUE" indicates "ON" and "FALSE" indicates "OFF". Parallel coils can only be inserted upward or downward.

Coils are classified into coils, negated coils, set coils, and reset coils. You can switch between the four coil types by using the right-click menu command or shortcut.

- Coil: Assigns the logical operation results on the left to coil variables directly.

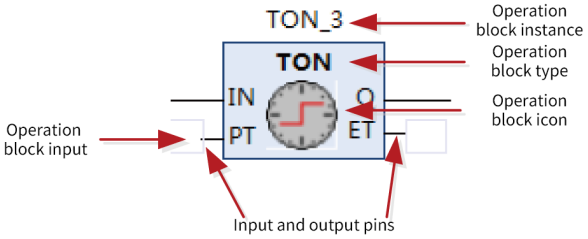
- Negated coil: Negates the logical operation results on the left and assigns the values to coil variables.
- Set coil: Sets the value of the coil variable to "ON (TRUE)" if the left-side state value of the coil is "ON (TRUE)", and keeps the value until the variable is reset to "OFF (FALSE)" by the reset coil.
- Reset coil: Resets the set coil.

Operation block

Icon - 

An operation block can be an operator, function, function block, program, action, or method. If the operation block is of the function block type, a text box is displayed above the operation block box to display the function block instance.

An operation block includes at least one input and one output. The following figure shows the components of an operation block.



Operation blocks are classified into common operation blocks and EN/ENO operation blocks.

- EN/ENO operation block: Contains EN input and ENO output, in addition to the input and output provided by an operation block. The execution logic of the EN/ENO operation block is as follows: The operation block logic is executed when EN is "TRUE". ENO is "TRUE" after execution. The operation block is not executed if EN is "FALSE". In this case, ENO is "FALSE". Note that the input line of the EN/ENO operation bock can only be connected to the EN pin, and the output line can only be connected to the ENO pin.
- Input and output pins of the operation block: The negation, rising edge, and falling edge signals can be added to the input pin of the BOOL type. The negation signal can be added to the output pin of the operation block.
- Multi-input line operation block: The operation block contains multiple inputs that are connected to the power flow line. The following figure shows an operation block with two input lines. As a multi-input line operation block provides multiple lines connected to the power flow line, it is only located in the first branch and cannot be connected to branches in parallel.

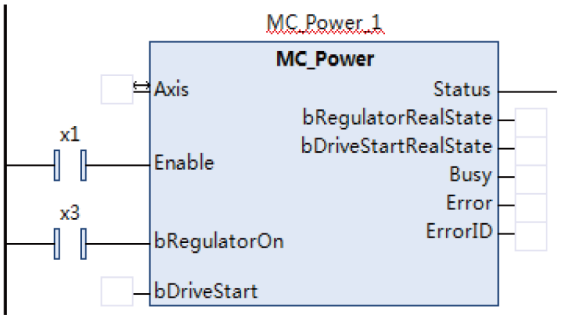


Figure 6-2 Multi-input line operation block

Execution block

Icon - 

An execution block can be inserted into the embedded ST. ST statements can be edited within the block. The execution block can be zoomed in or out. The maximum size is (1000 x 400).

Branch

Icon - 

Branches form a non-closed parallel logic.

Label

Icon - 

A label indicates a jump location and is at the head of the network. The system jumps to the label position through jump elements. A jump label is a string and must comply with the naming conventions.

Jump

Icon - 

When the input on the left of a jump element is "TRUE", the system jumps to the specified label position for execution. The jump element is on the rightmost of the network.

Return

Icon - 

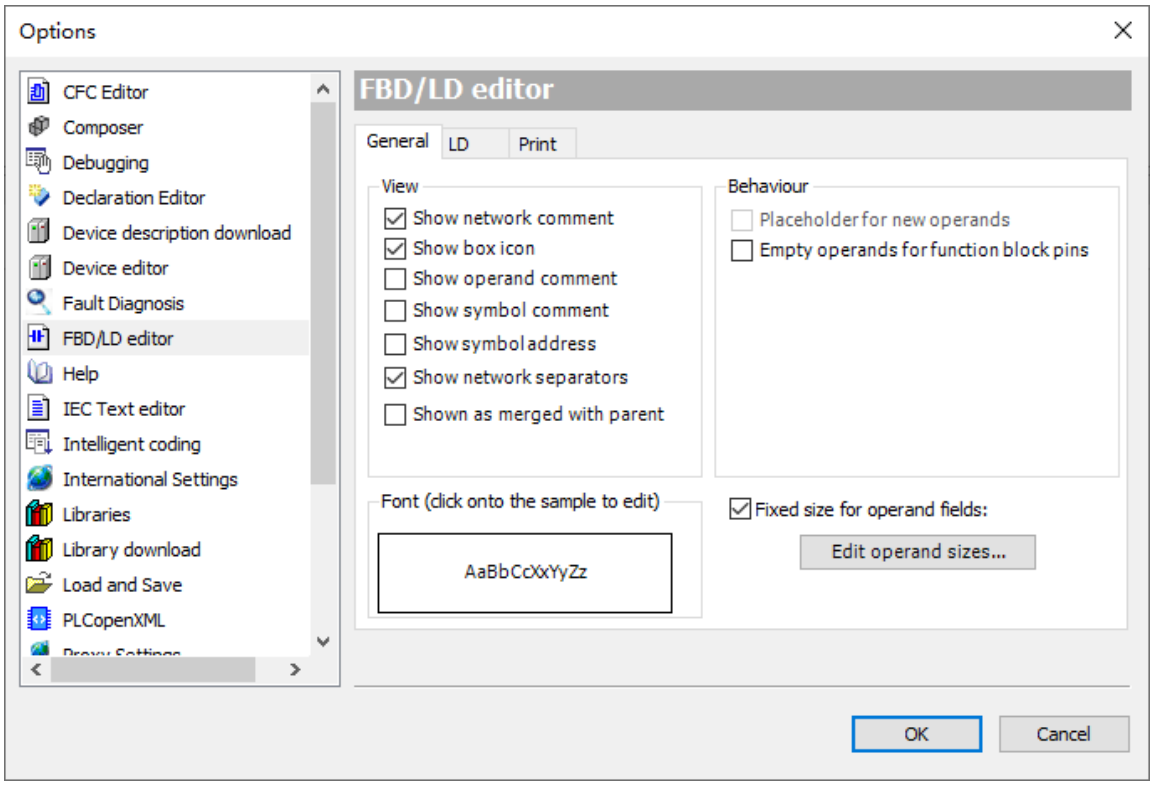
When the input on the left of the return element is "TRUE", the current program exits execution immediately. The return element is on the rightmost of the network.

6.3.3 LD Editor Options

LD editor options are used to control the LD page display, single-key command setting, and print display mode. Access the LD editor options by choosing "Tools" > "Options" > "FBD/LD editor". The LD editor provides three tabs: "General", "LD", and "Print".

General

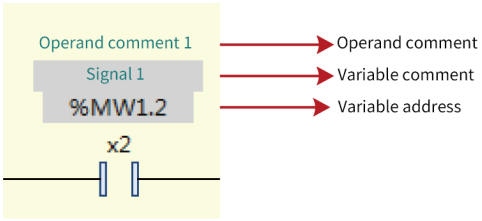
The following figure shows the "General" tab page.



"General" tab page of the LD editor

View

- Show network title: If this option is selected, you can insert and edit the title of each network of the LD. The inserted title is displayed above the current network. If no title exists, the title line is not displayed. A title is inserted by using a menu command.
- Show network comment: If this option is selected, you can edit the comment of each network of the LD. If a network comment is added, it is displayed below the network title. If no network comment exists, the comment line is not displayed. A network comment is edited by using a menu command.
- Show box icon: If this option is selected, the icon defined for the operation block is displayed in the middle of the block. Icons are defined for standard operands (such as ADD and SUB) and function blocks (such as TON and TOF). To add images as operation block icons to user-defined functions, function blocks, or programs, right-click an object and choose "Properties" > "Bitmap", and then click the area to select project-related bitmap.
- Show operand comment: If this option is selected, you can edit and display the comment of each operand on the LD page. An operand is a programming concept. Variables, constants, and addresses are operands. As the LD does not necessarily use variables, when constants or addresses are used, you can describe them by using operand comments. When editing an operand comment, select an operand string and right-click it.
- Show symbol comment: If this option is selected, the comments during variable declaration are displayed for the variables on the LD page. Variable comments come from variable declaration and cannot be edited.



Font

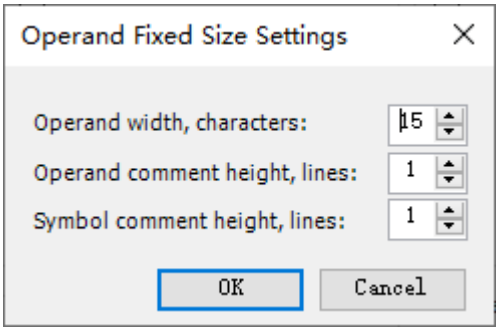
Click the sample text. The editor font selection box is displayed, where you can set the font of the LD. The default font is Microsoft YaHei and the default font size is 9. The font range mainly covers operands and comments. The execution block font uses the text editor font (ST text and variable declaration text).

Behaviour

- Placeholder for new operands: not implemented.
- Empty operands for function block pins: If this option is selected, the input and output pins of a new operation block use empty characters. If this option is not selected, the input and output pins of the operation block use "???".

Operand Fixed Size Settings

If this option is selected, you can set operand width, operand comment height, and symbol comment height, as shown in the following figure.

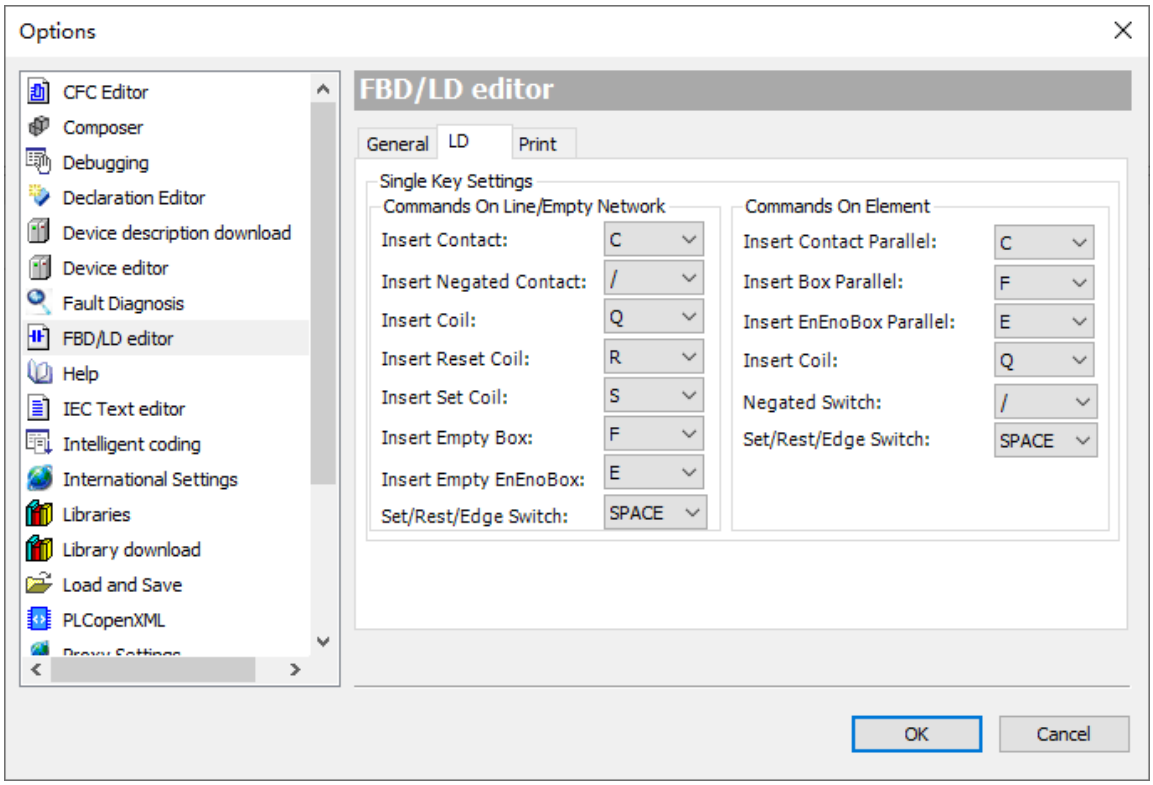


Operand Fixed Size Settings

- Operand width: Sets the number of fixed characters of an operand. The default value is 15.
- Operand comment height: Sets the number of operand comment lines of fixed length. The default value is 1.
- Symbol comment height: Sets the number of variable comment lines of fixed length. The default value is 1.

LD option settings

The following figure shows the "LD" tab page.



Single Key Settings

The single key settings function enables the edit operation through a single key, including single keys executed on lines and those executed on elements. The single-key commands executed on lines insert serial elements, whereas the single-key commands executed on elements insert parallel elements.

You can switch element functions when selecting elements, such as negation switching, edge signal switching, and set/reset switching. Negation switching is applied to contacts and coils and uses the "/" key. Edge signal switching is applied to contacts and uses space. Set/reset switching is applied to coils and uses space.

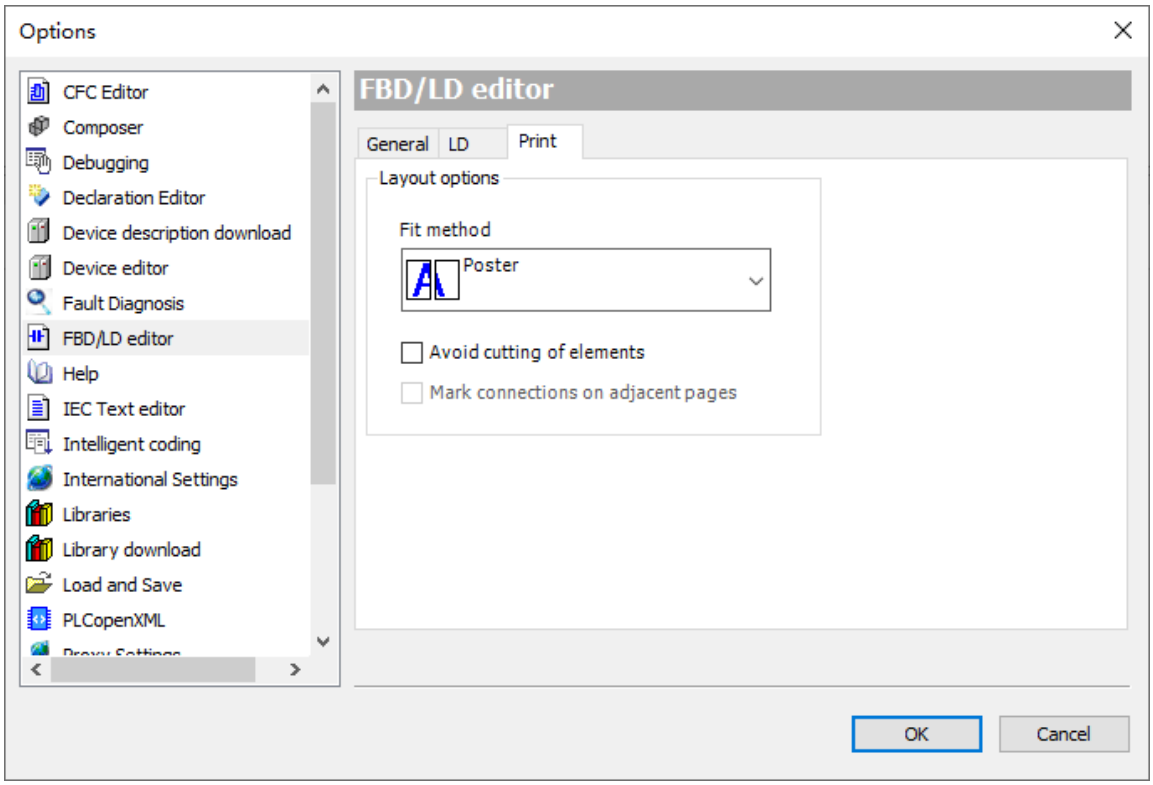
You can set a single key for implementing a function. Each function has a default single-key character. You can set keys as needed.

Note

The same key cannot be used repeatedly under "Commands on Line" or "Commands on Element", but "Commands on Line" and "Commands on Element" may share the same key.

Print

The following figure shows the "Print" tab page.



Layout options

Fit method:

- Poster: Print based on the normal proportion. If the current page is not high enough to display the entire network, the next page is printed. If the page is wide enough to display the entire network, the remaining part is printed on the next page.
- Shrink to fit the widest network: The displayed content is compressed during printing so that all the networks are displayed within the width of the page. If a page is not high enough to display the entire network, the remaining part of the network is displayed on the next page.
- Avoid cutting of elements: If this option is selected, when an element is displayed between two pages, the element is placed on the next page for display. This option is available only when "Poster" is selected for "Fit method".
- Mark connections on adjacent pages: If this option is selected, the connection between two pages is marked. This option is available only when "Avoid cutting of elements" is selected.

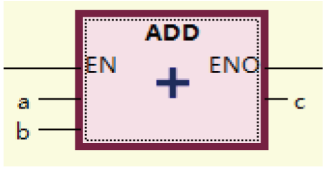
6.3.4 Element Selection

Selection is the basis of the edit operation. You can select elements or lines, select one object at a time or select multiple objects simultaneously by pressing Ctrl or Shift, and select consecutive or nonconsecutive objects.

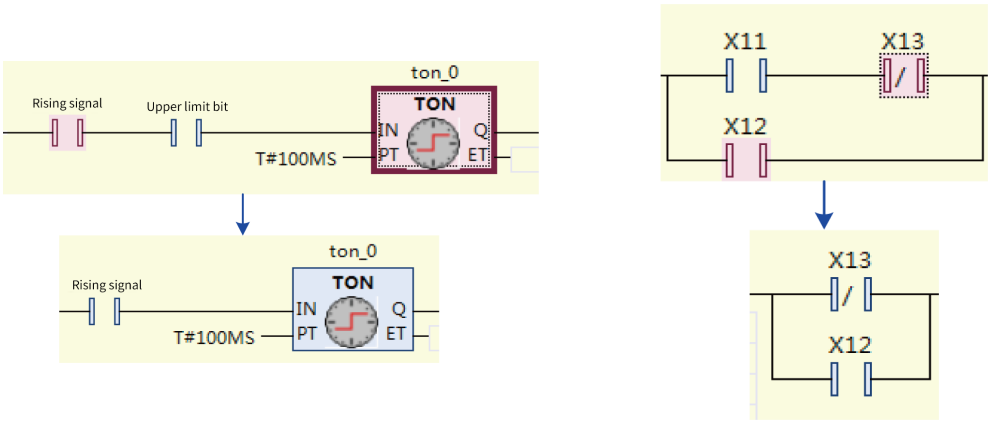
A selected element is highlighted. For example, a selected contact is displayed as . The external dotted box indicates that the contact is focused. You can select an element and paste it to another element in parallel mode, or drag and drop it to another element in serial or parallel mode. As multiple selected elements may not be consecutive, you need to describe the result logic formed by the selected elements. The selection result logic is intended to keep the original logic consistent. You can select an element through box select, and select multiple or all elements by pressing Ctrl or Shift.

Result logic formed by selected elements

- Single-element selection: When you select a contact or coil, only the selected contact or coil is included. When you select an operation block, the operation block and its line-free input and output operands are included. As shown in the following figure, when the ADD operation block is selected, the ADD operation block, inputs a and b, and output c are pasted.

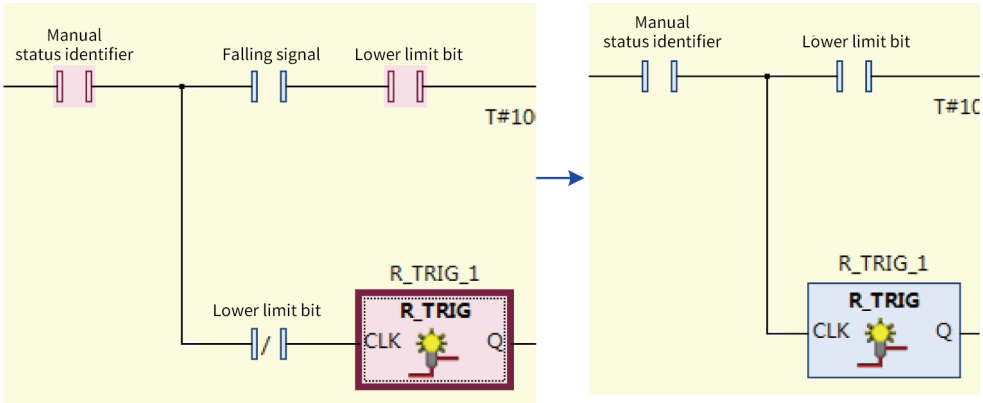


- Multi-element selection: Select the serial connection on a line (including nonconsecutive selection) and select elements on a parallel line (including nonconsecutive selection) to form parallel results.

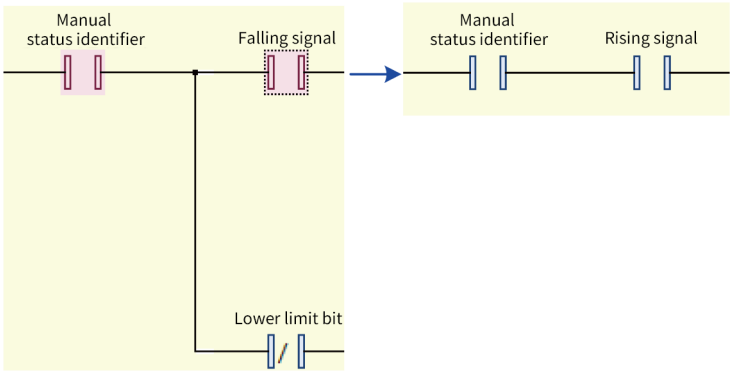


Parallel logic formation

- Multi-element selection: If the selected elements span multiple branches, the results also span branches, and the original logic remains unchanged. If the selected elements span two branches, the elements are connected serially.

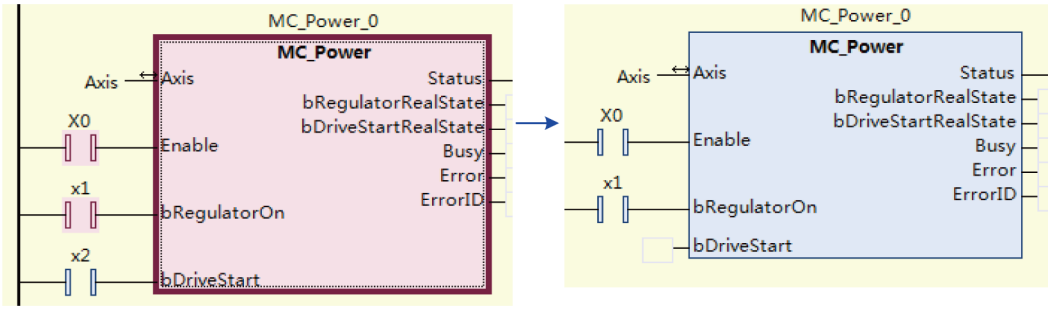


The results span multiple branches if the selected elements come from more than two branches.

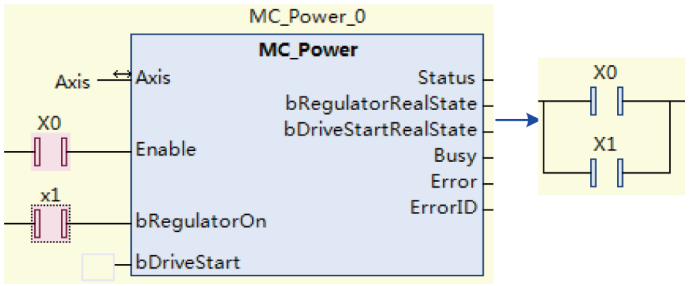


If the selected elements come from two branches, the results of the two branches are connected serially.

- Multi-element selection: If you select the multi-input line operation block and the elements on multiple input lines, the results are the same as the selected ones. If you only select the elements on multiple input lines but do not select the operation block, the elements on the input lines are connected in parallel to form results.



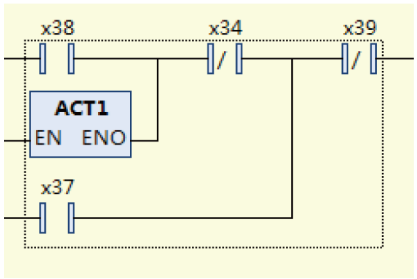
If you select the operation block and input elements, the results are the same as the selected ones.



If you only select input elements but do not select the operation block, a parallel logic is formed.

Box selection

The elements within the rectangle that starts from where the mouse is pressed and ends where the mouse is released, are selected, as shown in the following figure.



Note

Box selection is only applicable to single network selection. Selection starts from the blank area where the mouse is pressed.

Multi-element selection through Ctrl and Shift

Multi-element selection through Ctrl and Shift complies with the standard multi-element selection method.

- When you press Ctrl, if the current element is not selected, it is added to the selection list. If the current element is already selected, it is removed from the selection list.
- Press Shift to select elements within the rectangle from the last selected element to the currently selected element.

Select all

Press Ctrl+A to select all networks.

6.3.5 Standard Edit Commands

The LD supports standard and common edit operations, such as copy, paste, delete, cut, undo, and restore. Standard edit shortcut keys are used.

Copy

It is used to copy the selected element. The copy result is the selected element. For details, see the section about element selection.

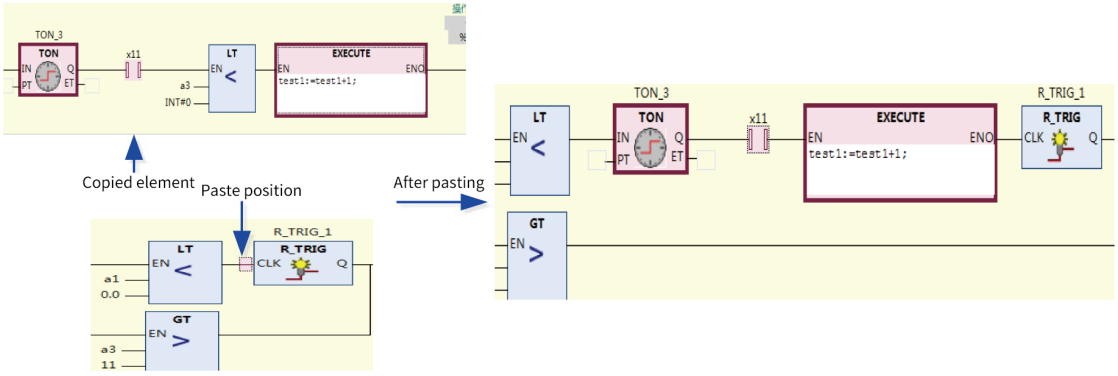
The following elements of the LD can be copied: network, contact, coil, operation block, string, and branch line.

The copied elements may be consecutive or not, depending on the selection. The elements to be copied must be within a single network. If elements across networks are selected, only the data selected in the focused network is copied.

Paste

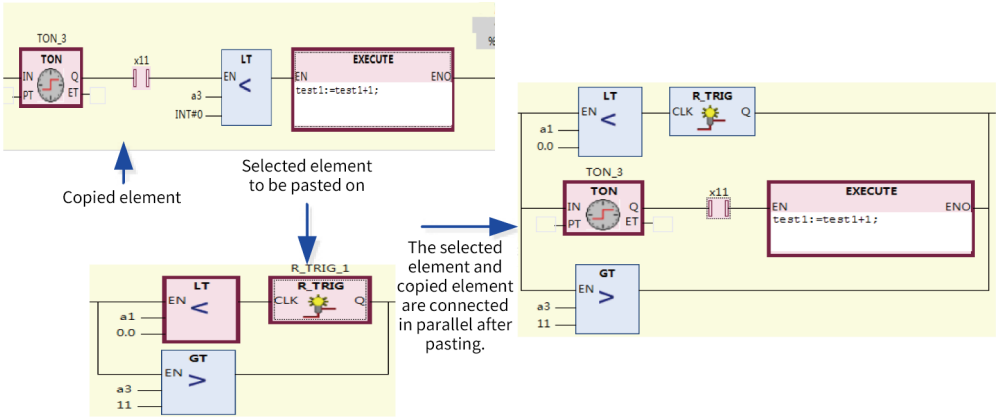
It is used to paste the copied elements. The paste rules are as follows:

- Paste on line
  When "Paste on line" is selected, the copied element is inserted in the line position to form a serial relationship.



- Paste on element

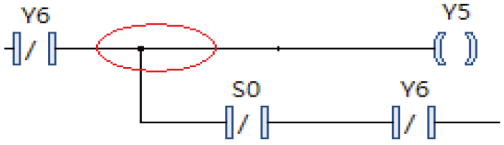
  When "Paste on element" is selected, the elements selected in batch form a parallel relationship. When a parallel relationship is formed, the selected elements must meet the parallel conditions for the paste operation. That is, the selected elements must be on the same line and consecutive and cannot span branches, and the start element and end element cannot connect to the branch internally and externally in parallel.



- Paste at coil position

  As no elements exist on the right of a coil, special paste rules must be observed. The rules for jump elements and return elements are the same as those for coils. Coil is used as an example here.

  1. If a coil is selected (parallel element connection is selected), the pasted elements are located in a new branch below the coil, as shown in the following figure. Select Y5 to paste S0 and Y6. S0 and Y6 are located in a new branch below the Y5 coil.

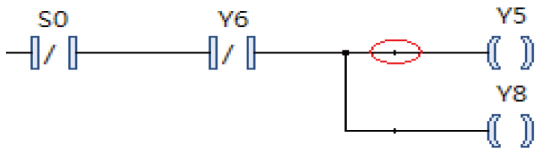


  2. If a line before the coil is selected (serial line connection is selected), the copied content may include only one branch or multiple branches, depending on the copied elements.

  Only one branch in the copied content

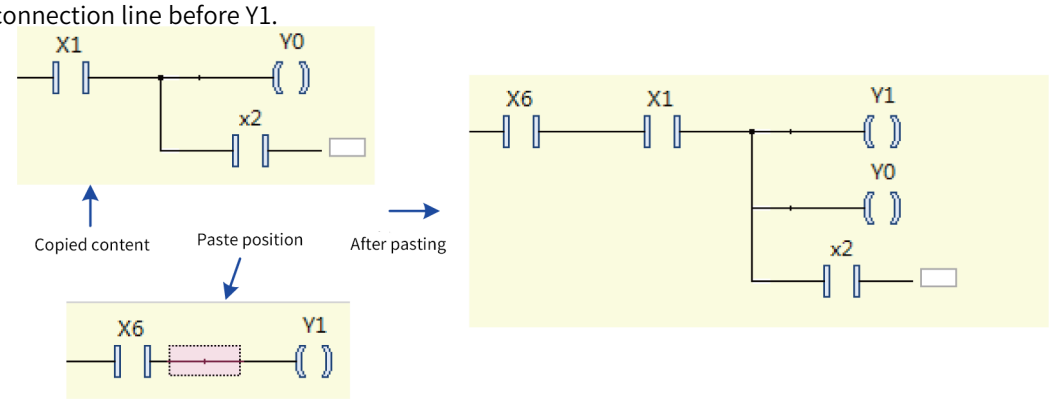
  1. If the copied content does not include the coil, the copied elements are connected to the coil serially.
  2. If the copied element includes the coil, the data before the copied coil is inserted at the line selection position, and the copied coil and the coil after the selection line form a non-closed parallel connection, as shown in the following figure. Select the connection line before Y5 to paste

S0, Y6, and Y8. After pasting, S0 and Y6 before Y8 are serially connected to the line before Y5. Y8 and Y5 are connected in parallel through a branch.

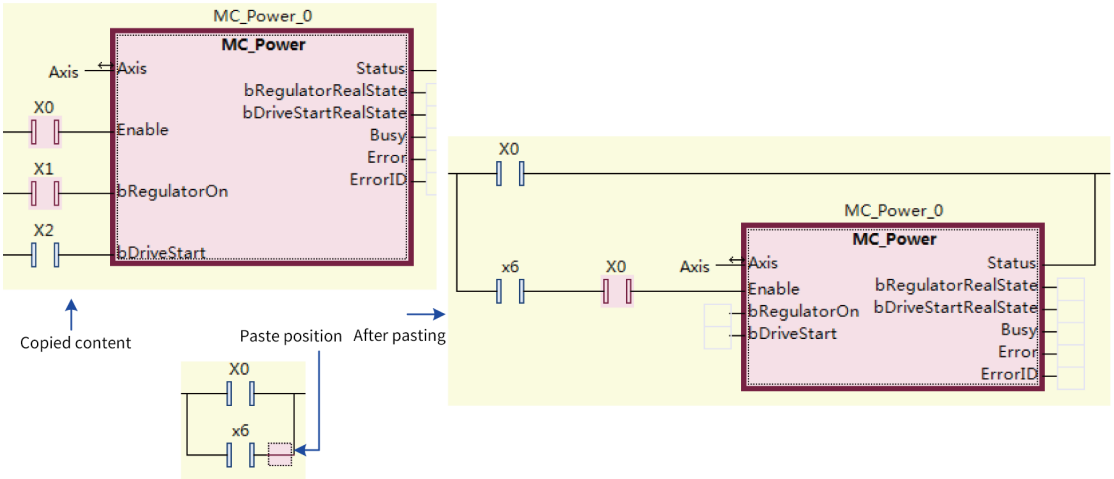


Multiple branches in the copied content

1. If the last branch horizontal to the copied content does not include the coil, the copied content is inserted at the line selection position.
2. If the last branch horizontal to the copied content includes a coil, connect this branch to the coil after the connection line in parallel mode, and keep other data connected in serial or parallel mode unchanged, as shown in the following figure. Paste the copied X1, Y0, and X2 to the connection line before the Y1 coil. Y0 is located below the Y1 coil, and X1 is located on the connection line before Y1.

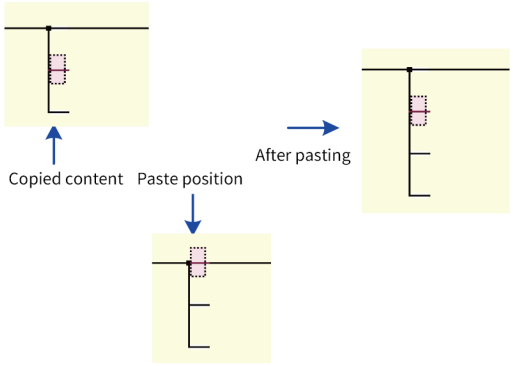


- Paste multi-input line operation block
  The multi-input line operation block is only located at the non-parallel position of the first branch, that is, the paste position. If the multi-input line operation block is not included, the connection elements of the copied operation block starting from the second input line are deleted, as shown in the following figure. The X0 and X1 contacts of the copied operation block are copied, and X1 is deleted when being pasted.



- Paste network
  You can select one or more networks for the copy and paste operations. Selection is required in advance.

- Paste a single branch line
  This function is used to add a single branch. It is the same as the function of inserting a branch upward or downward. You can copy a single branch line and paste it at the branch output position. The branch line is pasted below the selected branch, as shown in the following figure.

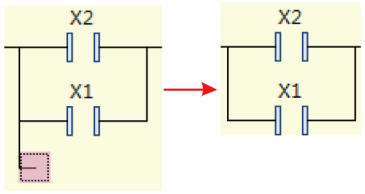


Delete

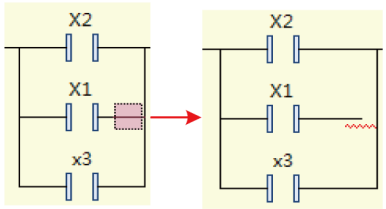
It is used to delete the selected element. After the current element is deleted, the next element is selected to ensure operation continuity.

Both elements and lines can be deleted.

- Delete element: When this option is selected, the element is deleted.
- Delete line: When this option is selected, the line as well as its branch lines, vertical lines, and lines connected to the vertical lines on the left are deleted.
  Delete branch line: When this option is selected, after the blank branch line is deleted, this branch is also deleted.



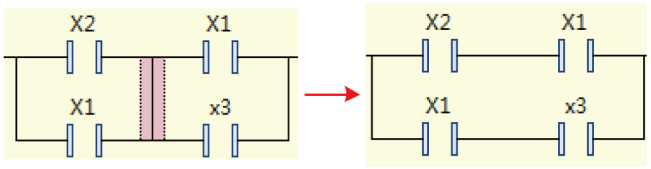
Delete lines connected to the vertical lines on the left: When this option is selected, the lines connected to the vertical lines on the left are deleted, and the connection and branch are disconnected.



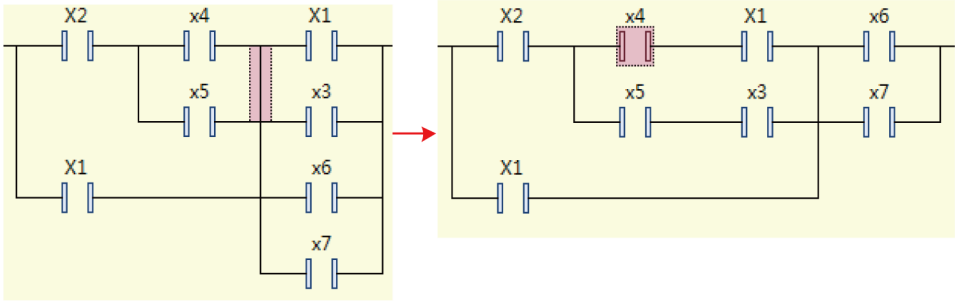
Delete vertical line: In principle, after a vertical line is deleted, this line no longer exists. Three results may occur: The branches are merged, the branch is disconnected, or the branch is moved left.

1. Branch merging

   If both the left and right parts of the selected vertical line are inside a branch, after the vertical line is deleted, the branches on the two sides will be merged.

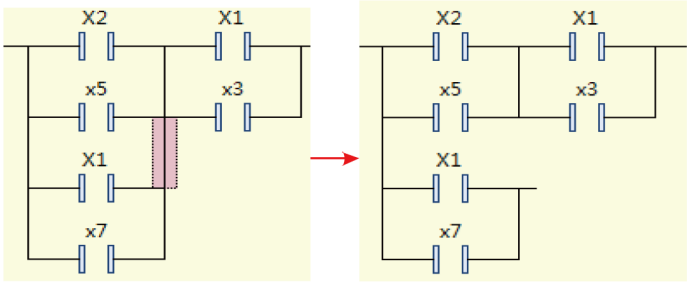


Note that if a bridge circuit is generated after the vertical line is deleted, the upper and lower branches on the right will be moved to the left and merged, and other branches on the right will be moved down. As shown in the following figure, x1 and x3 are moved to the left and merged, while x6 and x7 are moved up.



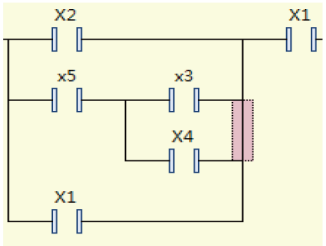
2. Branch disconnected

If the left part of the selected vertical line is inside a branch, there is a branch in the right upper part but no in the right lower part, after the vertical line is deleted, the branch on the left is disconnected.



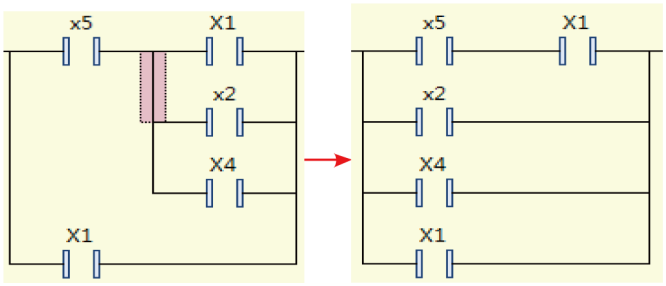
Note

If a bridge circuit is generated after a vertical line is deleted, undo the deletion.



3. Branch moved to the left

If the right part of the selected vertical line is inside a branch, there is a branch in the left upper part but no in the left lower part, after the vertical line is deleted, the branch on the right is moved to the previous vertical line.



Cut

It is used to copy, paste, and then delete the selected element.

Undo/Restore

Undo: It is used to return to the previous edit state and restore the previously selected element.

Restore: It is used to restore the next edit state and the next selected element.

6.3.6 LD Menu Commands

Menu commands include the shortcut menu commands and the LD commands in the LD toolbox.

Insert network

The "Insert Network" and "Insert Network (below)" menu commands are provided. You can drag a network in the toolbox to insert a network.

Command execution condition: A network is selected and another network is inserted above or below the selected network.

Insert Network: icon - ; shortcut key: Ctrl+I, which inserts an empty network above the selected network.

Insert Network (below): icon - ; shortcut key: Ctrl+T, which inserts an empty network below the selected network.

Toggle network comment state

Icon - ; Ctrl+O, which switches a network between the comment state and non-comment state.

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

Command execution condition: A network is selected.

Insert network header information

A network header mainly includes the network title, network comment, and label.

The commands for inserting a network header include "Insert Label", "Edit Network Title", and "Edit Network Comment".

- Edit Network Title: icon - , which edits the title of the selected network.

- Command execution condition: A network is selected and "Show network title" is selected.
- Edit Network Comment: icon - , which edits the comment of the selected network.
- Command execution condition: A network is selected and "Show network comment" is selected.
- Insert label: icon - , which inserts a jump label to the selected network to indicate the jump position of a jump element.
- Command execution condition: A network is selected.

Insert operation block

The following five menu commands are provided: "Insert Box", "Insert Empty Box", "Insert Box with EN/ENO", "Insert Empty Box with EN/ENO", and "Insert Box Parallel (below)". The commands are used to insert operators, functions, function blocks, and programs. You can also drag and drop the operation block or EN/ENO operation block from the toolbox to insert an operation block.

The first four commands are used to insert serial operation blocks, and the last command is used to insert operation blocks parallel to selected elements.

Insertion position:

1. Select a horizontal line and insert an operation block on the horizontal line.
2. Select an element and insert an operation block to the left of the element.

- Insert Box: icon - ; Ctrl+B, which displays the input assistant for you to select the operation block to be inserted.
- Insert Empty Box: icon - ; shortcut key: Ctrl+Shift+B, which inserts an empty operation block, without using the input assistant. You can specify the operation block type in the corresponding location.
- Insert Box with EN/ENO: icon - ; Ctrl+Shift+E, which displays the input assistant for you to select the operation block to be inserted. The operation block provides EN/ENO input and output. The operation block with EN/ENO is executed only when EN is "TRUE". It is not executed when EN is "FALSE". ENO has the same result as EN.
- Insert Empty Box with EN/ENO: Inserts an empty operation block, without using the input assistant. The operation block provides EN/ENO input and output.
- Insert Box Parallel (below): Inserts an empty operation block below the selected element. The selected element can be a contact or operation block.

Insert execution block

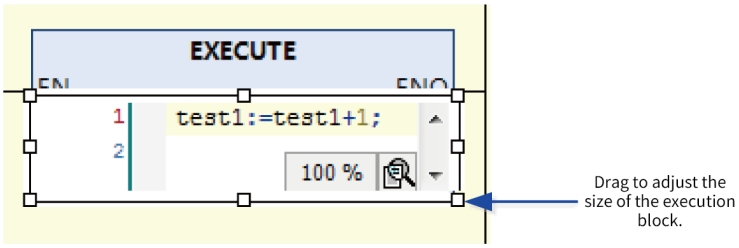
Insert Execute Box: icon - , which inserts a serial execution block in the selected location. You can also drag and drop an execution block from the toolbox.

Insertion position:

1. Select a horizontal line and insert an execution block on the horizontal line.
2. Select an element and insert an operation block to the left of the element.

An execution block can be used to edit ST statements. Click the text area for editing. The execution block only provides EN/ENO input and output.

Drag to change execution block size: Drag the execution block frame in the editable state to control the block size, as shown in the following figure.



Insert input

Insert Input: icon - ; shortcut key: Ctrl+Q, which adds input to a variable-input operation block.

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

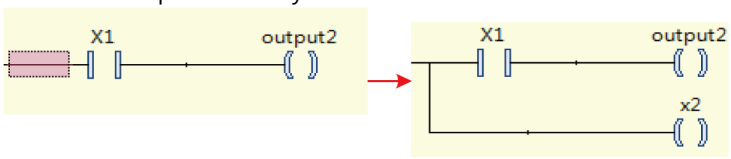
Insertion position: When an input pin is selected, an input is added before the input pin. When an operation block is selected, the added input pin is located at the end.

Insert coil

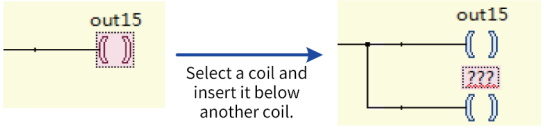
Three menu commands are provided: "Insert Coil", "Insert Set Coil", and "Insert Reset Coil". You can also insert coils by dragging and dropping "Coil", "Set coil", and "Reset coil" from the toolbox.

- Command execution condition: The selected position cannot be located on the parallel branch or at the input position of the multi-input line operation block.
- Insert Coil: icon - ; shortcut key: Ctrl+Shift+A, which outputs a coil at the current position.
- Insertion position:

  1. Select a horizontal line or element and insert a coil on the horizontal line or at the left of the element. The coil and line are processed by a non-closed branch.



  2. If coil, return, or jump is selected, a new coil is inserted below the selected element.



The default variable of the inserted coil is "???". You need to enter the required variable or constant. You can use the input assistant (press F2) to select an input from the variable list.

- Insert Set Coil: icon - , which inserts a set coil at the current position. The operation is the same as "Insert Coil".
- Insert Reset Coil: icon - , which inserts a reset coil at the current position. The operation is the same as "Insert Coil".

Insert contact

The following four menu commands are provided: "Insert Contact", "Insert Negated Contact", "Insert Contact Parallel (below)", and "Insert Contact Parallel (above)". You can also insert a contact by dragging and dropping "Contact" or "Negated contact" from the toolbox.

- Insert Contact: icon - ; shortcut key: Ctrl+K, which inserts an NO contact at the current position in serial mode.
  Insertion position:

  1. Select a horizontal line and insert a contact on the horizontal line.
  2. If a network is selected, the new contact is inserted at the end.
  3. If an element is selected, the new contact is inserted on the left of the element.

The default variable name of the contact is "???". Click the variable or constant required by text input. You can use the input assistant (press F2) to select an input from the variable list.

- Insert Negated Contact: icon - , which inserts an NC contact at the current position in serial mode. The operation is the same as "Insert Contact".
- Insert Contact Parallel (below): icon - ; shortcut key: Ctrl+R, which inserts an NO contact below the selected element in parallel mode. The selected element can be a contact or operation block.
- Insert Contact Parallel (above): icon - ; shortcut key: Ctrl+P, which inserts an NO contact above the selected element in parallel mode. The operation is the same as "Insert Contact Parallel (below)".

Insert branch

Two menu commands are provided: "Insert Branch" and "Insert Branch above". You can also insert branches by dragging and dropping "Branch" from the toolbox. The inserted branch is a non-closed line.

- Insert Branch: icon - ; shortcut key: Ctrl+Shift+V, which inserts a branch at the selected line position.
  Insertion position:

  1. If a line is selected, the branch is inserted below the line.
  2. If a contact or coil is selected, the branch is inserted before the selected element.

  As shown in the following figure, each selected position indicates a branch.

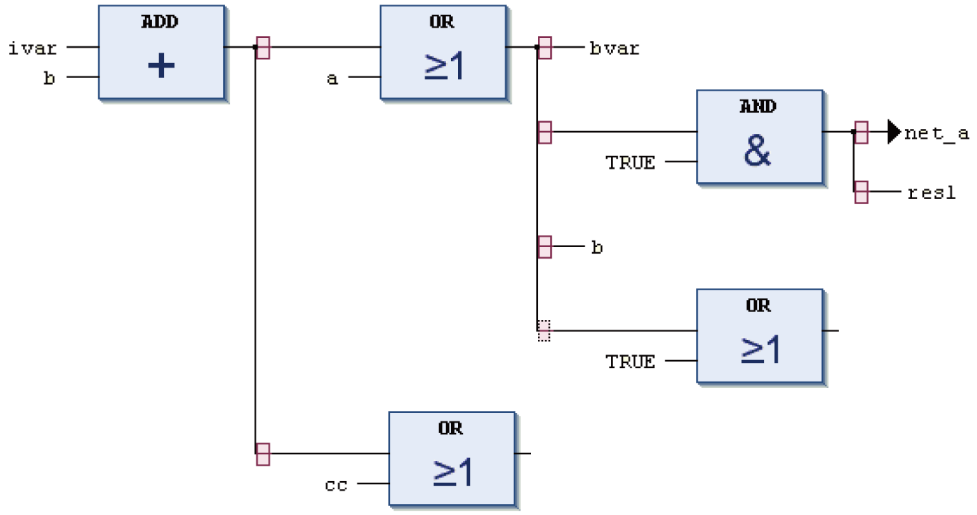


Figure 6-3 Branch label

- Insert Branch above: icon - , which adds a branch above the selected branch. A branch line is selected before this command can be executed.

Jump and return

Two menu commands are provided: "Insert Jump" and "Insert Return". Jump and Return are used to control the program execution sequence. In normal cases, programs are executed from top down and from left to right based on the network sequence. To add a jump or return element, drag and drop "Jump" or "Return" from the toolbox.

Like coils, the jump and return elements must be located on the rightmost side. Therefore, the rules for inserting jump and return elements are the same as those for inserting coils. For details, see the "Insert Coil" command.

- Insert Jump: icon - ; shortcut key: Ctrl+L, which inserts a jump element to jump to the specified label position.
  The jump position is marked by a label in the network. That is, jump across networks is supported. Jump is executed only when the pre-jump input condition is met.
- Insert Return: icon - , which inserts a return element. When the input condition is met, the current POU executes the return command and returns results to the caller POU.

Negation

Icon - ; shortcut key: Ctrl+N, which negates the operation block input, operation block output, jump condition, return condition, contact value, or coil.

The negation command can be executed at the following two positions:

1. Element negation: The main elements are contact and coil. A slash (/) is added to the contact and coil after negation.
2. Line negation: The main elements are operation block input line, operation block output line, coil input line, jump input line, and return input line. A circle is added on the line.

The following figure shows the negation position.

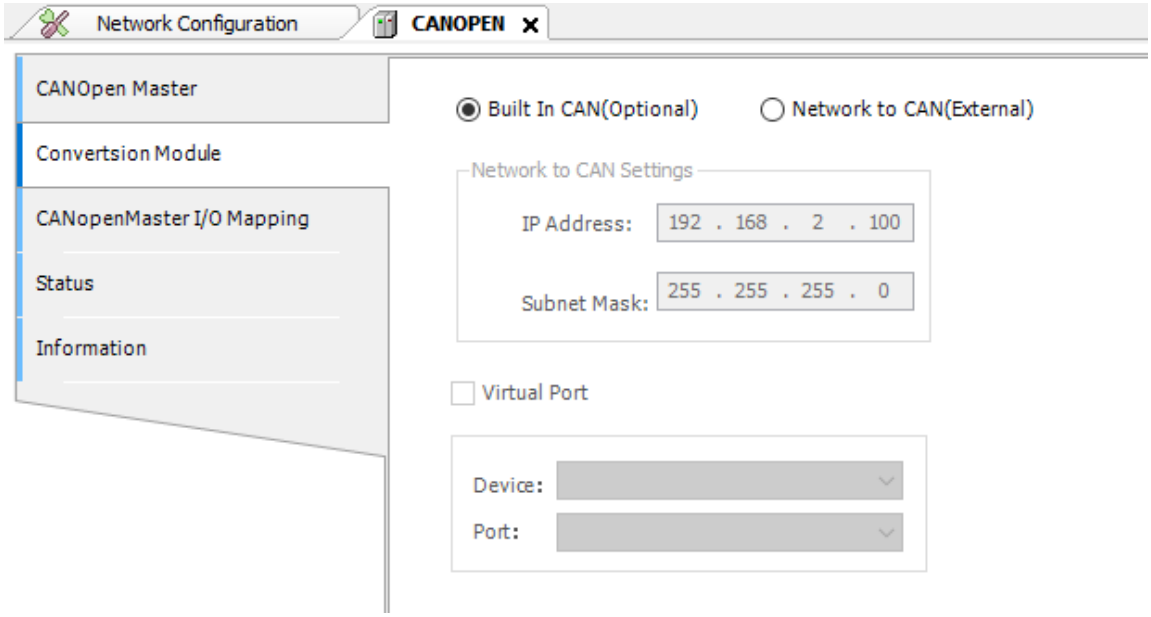


Figure 6-4 Negation position

The negation state is switched back when the negation command is executed again.

Detect edge

Icon - ; shortcut key: Ctrl+E, which adds the edge trigger function to contacts, operation block input lines, coil input lines, jump element input lines, and return element input lines.

Rising edge detection is equivalent to the R_TRIG function block, and falling edge detection is equivalent to the F_TRIG function block.

The edge detection command can be executed at the following two positions:

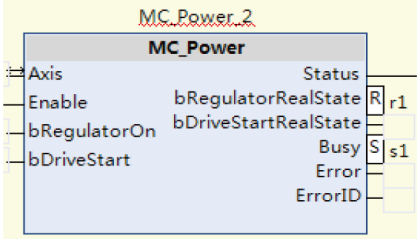
1. Contact edge detection: Select a contact to run the edge detection command. The edge detection function is added to the contact. indicates the rising edge, and indicates the falling edge.
2. Add edge detection to line: The edge detection command of the execution block is applicable to the operation block input line, coil input line, jump element input line, and return element input line. The edge signal symbol is added to the line. The rising edge detection symbol is , and the falling edge detection symbol is . The edge detection function is added only to input lines of the BOOL type.

Set and reset

Icon - ; shortcut key: Ctrl+M, which adds the set or reset output function. Set output is displayed as S, and reset output is displayed as R. The command can be executed multiple times and switches among set, reset, and normal output.

The set/reset command can be executed at the following two positions:

1. Coil selection. This command sets or resets a coil. Set coil: Reset coil:
2. Select the BOOL-type output line (non-main output) of the operation block and configure the set or reset function, as shown in the following figure.



Set output connection

Icon - ; shortcut key: Ctrl+W, which modifies the pin of the main output when the operation block contains multiple outputs. An operation block has only one main output, which is linked to subsequent elements, as shown in the following figure.

Select the output pin to be modified and run this command to modify the output connection.

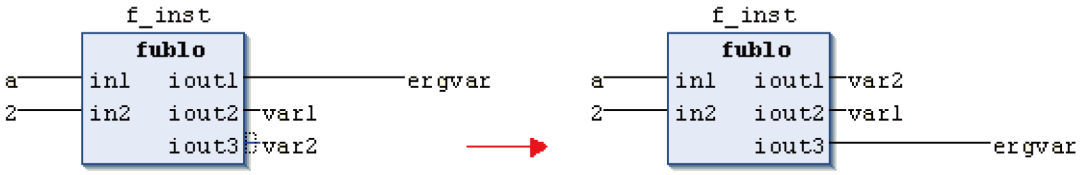


Figure 6-5 Output connection modification

Modify the input and output pin display

The following two menu commands are provided: "Update parameters" and "Remove unused FB call parameters".

Update parameters: icon - ; shortcut key: Ctrl+U, which updates the input and output parameters of the selected operation block. When the input or output parameters of the operation block are changed, run the "Update parameter" command to update the parameters.

Remove unused FB call parameters: icon - , which deletes the input and output pins of the unused operation block. That is, when the input or output of the operation block is "???" or empty, the input or output is not displayed.

Batch update block calls

When an LD editor is used, if any input or output parameter of any operation block in the editor changes, click "Batch update block calls" in the toolbar. Then the input and output parameters of the operation block in the current editor are updated in batch. When this command is executed, the system checks parameters of all operation blocks in the current editor before updating them.

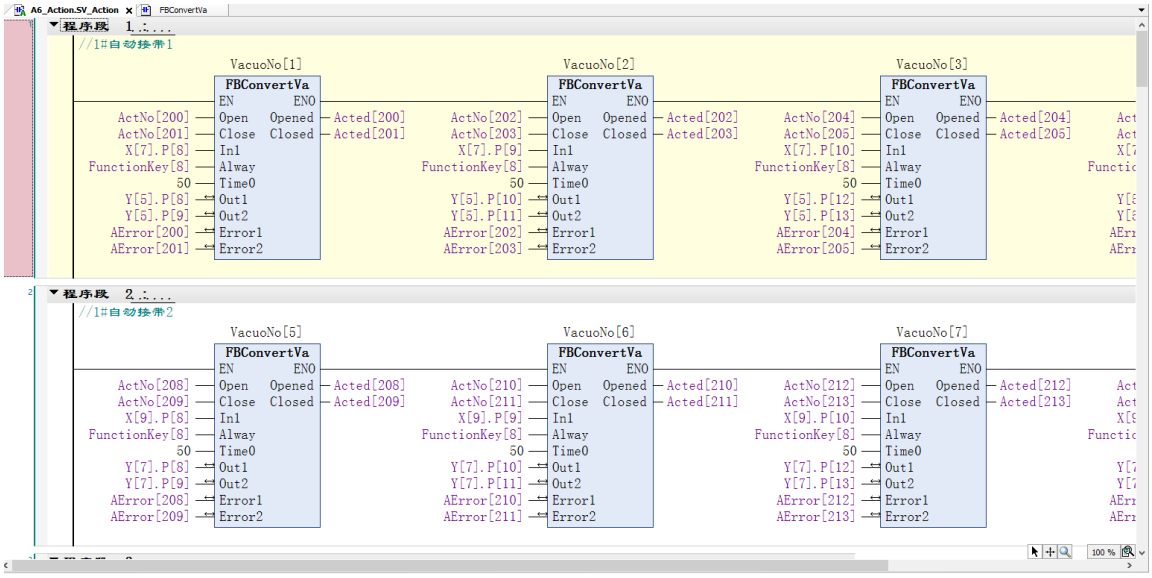
Example:

Modify the input and output parameters of FBConVerTVa: Delete input parameters Time1, Time2, Time3, and Alway1 and output parameters Signal1 and Signal2.

Before update:



After update:



Note

When the "Batch update block calls" command is executed, the system updates input, output, and input/output pin of operation blocks that can be updated in the editor for which LD is activated in batch based on the latest input, output, and input/output parameter definitions. Block operations that can be updated include PRG, FB, FUN, Action, and Meth.

Update block calls

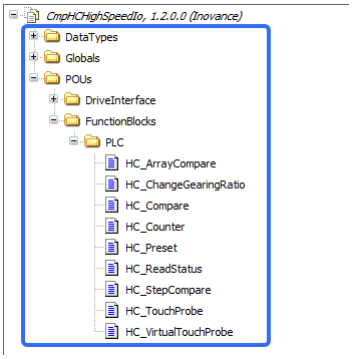
Right-click a POU node (only containing functions, function blocks, function block methods, and programs) in the device tree. In the shortcut menu displayed, select "Update block calls" to update input and output pins of operation blocks calling this POU in all LDs (not supported for ST) of the project.

After "Update block calls" is selected, the system searches for POUs that call this POU and recalculates their pins. If update is required, the system displays a modification confirmation window, listing all

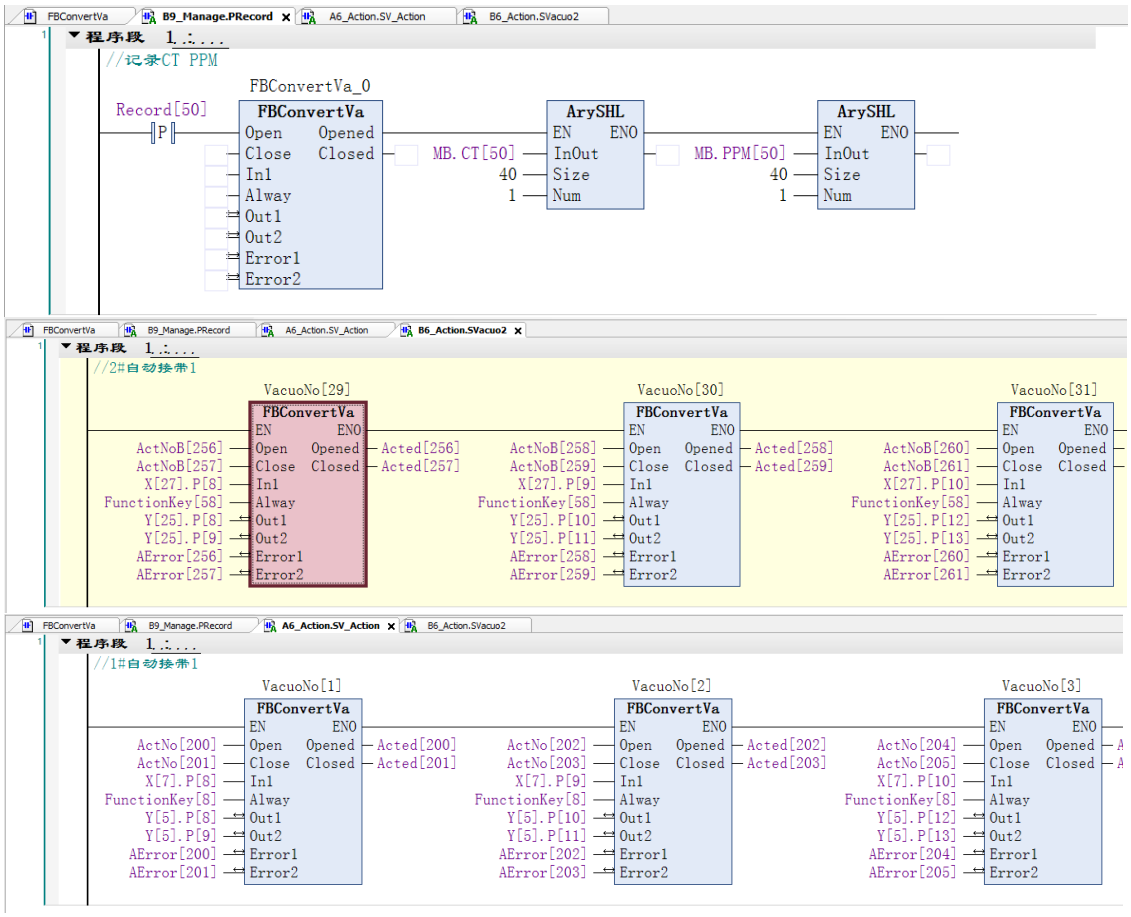
POUs to be modified. Click "OK" to perform the update or click "Cancel" to cancel the update. If no POU needs to be updated, this window is not displayed.

Example:

1. Right-click "FBConvertVa" in the device tree. In the shortcut menu, select "Update block calls". The system lists POUs requiring update in B9_Manage.PRecord, B6_Action.SVacuo2, and A6_Action.SV_Action.



2. In the dialog box displayed, click "OK".
   The input and output pins of all operations blocks (B9_Manage.PRecord, B6_Action.SVacuo2, and A6_Action.SV_Action) that call FBConvertVa are updated.



Note

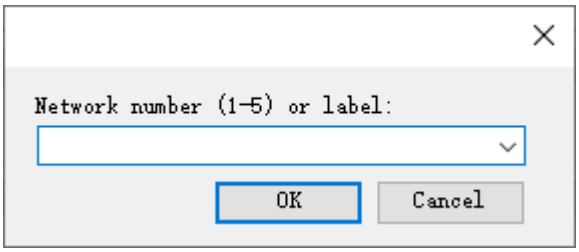
Only pins of operation blocks of the selected POUs are updated. If an FB function block with multiple Actions is selected, those calling the FB and calling the FB actions are updated together (because the input and output of Actions are the same as those of the FB function block by default).

Convert to LD language

View as ladder logic: shortcut key Ctrl+2. Convert FBD/IL to LD language: As FBD and IL are no longer supported, use this command to convert FBD and IL in the LD language for old projects.

Jump to network

Go To...: Jumps to the specified network. Specify the target network number in the "Network number (1-5) or label:" text box.



Edit Operand Comment

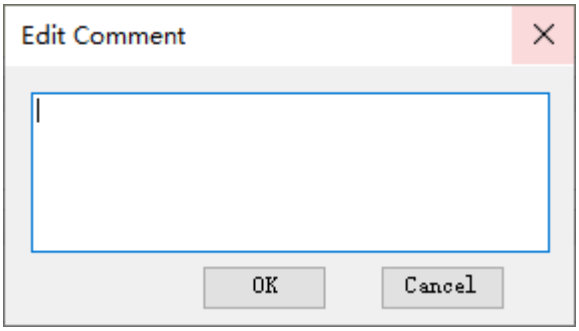
Edit Operand Comment: Edits the comment of the selected operand.

Command execution condition:

- FBD/LD is selected, and "Show operand comment" is selected.
- An operand string is selected.

Operand is a logical concept. Input variables, constants, and addresses are operands. Examples are operation block input variable, contact union variable, coil union variable, and operation block instance.

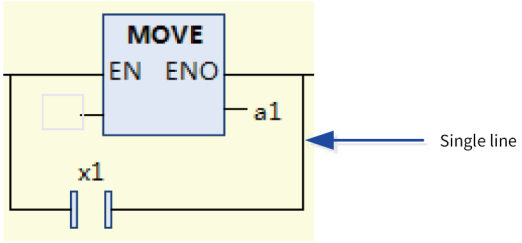
Select an operand string and run this command. The "Edit Comment" dialog box is displayed, as shown in the following figure. Edit the operand comment.



Toggle parallel mode

Toggle Parallel Mode: Switches the parallel mode of a parallel branch. The parallel mode is divided into the sequential parallel branch and the short-circuit-type parallel branch.

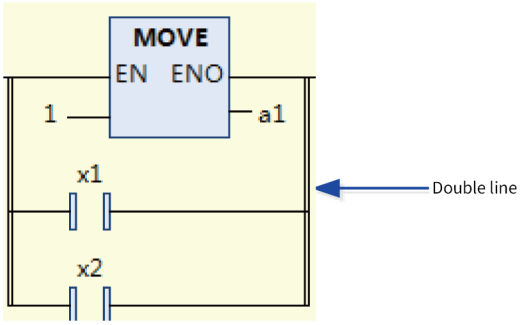
- The sequential parallel bracket uses a single line. The output of a single branch is subjected to the OR operation to obtain the branch output result, as shown in the following figure. Branch results are obtained through OR.



- The short-circuit-type bracket uses double line. The branch output result must consider whether each branch includes a non-operation block.
  The branch of the non-operation block is used as a condition. If a result of the branch is "TRUE", the branch with an operation block is not executed. It can be considered as an operation block of the contact short circuit type. As shown in the following figure, the first "Move" branch instruction is executed only when the results of the X1 and X2 branches are not "TRUE".

  The branch of the non-operation block must meet the following conditions:

  1. The branch only includes a contact or operator block.
  2. The contact does not include the edge signal.
  3. The operator block is not of the EN/ENO type, and its input line does not contain the negation or edge signal.



Note

Branches of the short circuit type are not recommended considering the branch complexity.

Set Branch Start/End Point

Set Branch Start/End Point: icon - ; shortcut key: Ctrl+D.

The "Set Branch Start/End Point" command is used to connect the start and end connections, functioning as the line drawing function.

To connect two points, set a start point (displayed the start point, indicating the start point of the connection), select an end point, and then execute this command. The system automatically connects the two points. For the specific connection logic, see the line drawing function.

6.3.7 Single-Key Command

A single-key command enables fast editing by using a single-character shortcut key. The single-key command can be executed on a line or an element. Run a single-key command on a line to insert a serial element. A single-key command on an element is used to insert parallel elements or switch the element function.

To set a character for each command, choose "Options" > "FBD/LD" > "LD".

Single-key commands on lines

- Insert Contact: The default single key is C.
- Insert Negated Contact: The default single key is "/".
- Insert Coil: The default single key is Q.
- Insert Reset Coil: The default single key is R.
- Insert Set Coil: The default single key is S.
- Insert Empty Box: The default single key is F.
- Insert Empty Box with EN/ENO: The default single key is E.
- Set/Reset/edge signal switching: The default single key is space. It is used to switch the BOOL-type input and output lines of the operation block. When a BOOL-type input line of the operation block is selected, delay signal switching is performed. When a BOOL-type output line of the operation block which is not the main output line is selected, "Set/Reset" switching is performed.

Single-key commands on elements

- Insert Contact Parallel: The default single key is C. The selected element can be a contact or operation block.
- Insert Box Parallel: The default single key is F. The selected element can be a contact or operation block.
- Insert EnEnoBox Parallel: The default single key is E. The selected element can be a contact or operation block.
- Insert Coil: The default single key is Q. The selected element can be a coil, return element, or jump element.
- Negated Switch: The default single key is "/". Select a contact to switch between NO and NC. Select a coil to switch negation.
- Switch element set/reset/edge signal: The default single key is space. Select a contact to switch among the rising edge signal, falling edge signal, and normal signal. Select a coil to switch among set coil, reset coil, and normal coil.

6.3.8 Line Drawing Function

This function connects the start and end points of a line. To use this function, meet the following conditions:

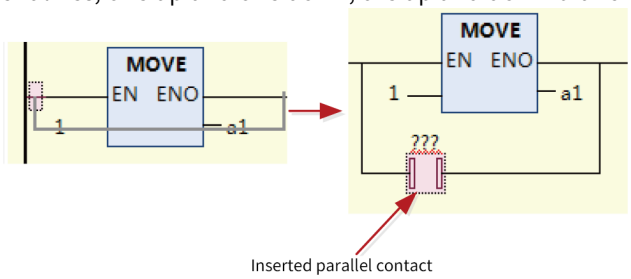
- The start and end points must be on the same line and can be selected (power flow line excluded).
- Dragging the input and output pins of an operation block can swap their positions. Areas (approximately 11 pixels) around pins of operation blocks are used for such dragging and therefore lines cannot be drawn in these areas.

Line drawing functions are divided by result into parallel line connection (a parallel branch added), branch closing, and branch splitting.

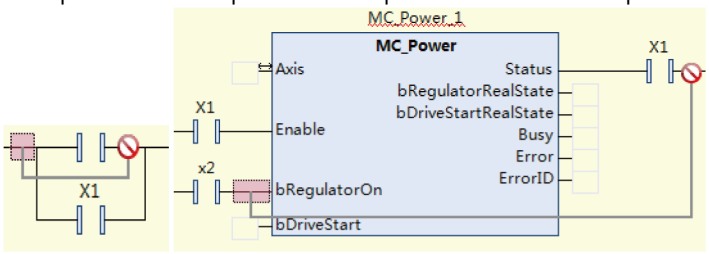
Parallel line connection

When the start point and end point of a line are on the same branch, a contact is automatically connected to the start and end points in parallel, as shown in the following figure.

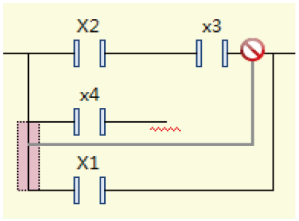
- If a line is drawn inside a branch, a contact is automatically connected between the start and end points in parallel.
- If you draw a line starting of the end point of a disconnected line to another line, the disconnected line is closed.
- If you draw two adjacent lines, one up and one down, the up and down branches are split.



- The start point and end point must meet the conditions of parallel connection. That is, the start point and end point cannot span the inside and outside of the parallel branch, cannot span branches, and cannot span from the input to the output of the multi-line operation block.

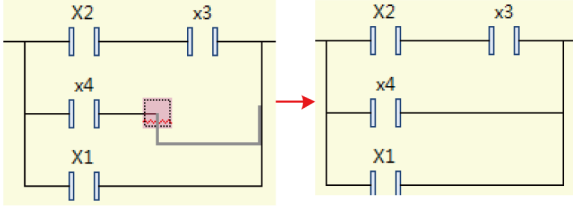


- The start and end point cannot span a disconnected branch.

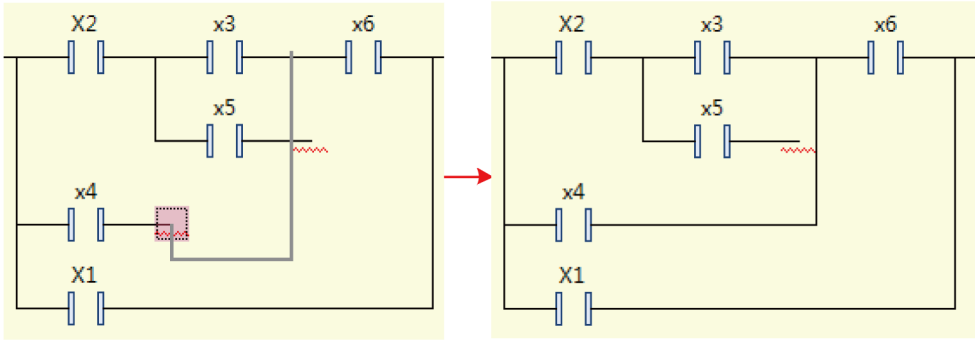


Branch closing

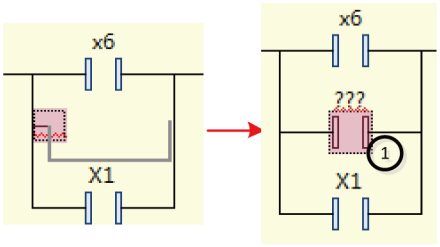
When the start point of a line is the end line of a disconnected branch and the other end is connected to a line that can be closed, the current disconnected branch is closed, as shown in the following figure.



- The branch closing function allows spanning a disconnected branch.

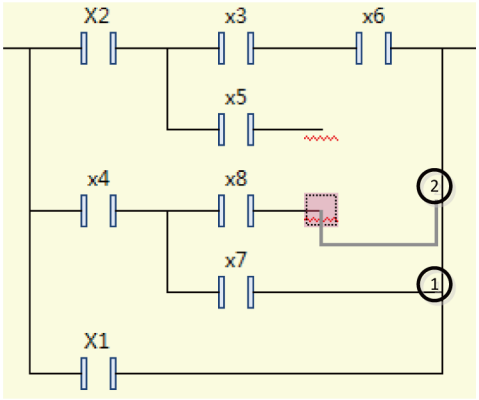


- If the disconnected branch only has an end point, an empty contact is automatically added when the branch is closed.



①: An empty contact is automatically added.

- If the branch of the hierarchy corresponding to the disconnected branch is closed, the disconnected branch can only be closed with the right vertical line.

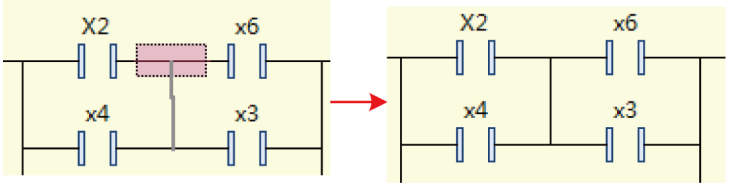


①: The branch of the hierarchy corresponding to the disconnected branch is closed.

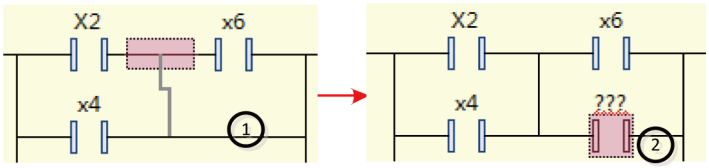
②: The disconnected branch can only be closed with the right vertical line.

Branch splitting

When the start and end points of a line are on two adjacent branches, the parts at two sides of the start and end points are split into two branches in parallel.



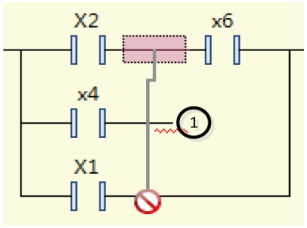
- If no element exists at the left or right side of the start or end point, an empty contact is automatically added at the side without element.



①: No element at the right

②: Added empty contact

- Disconnected branches cannot be split.



①: Disconnected branch

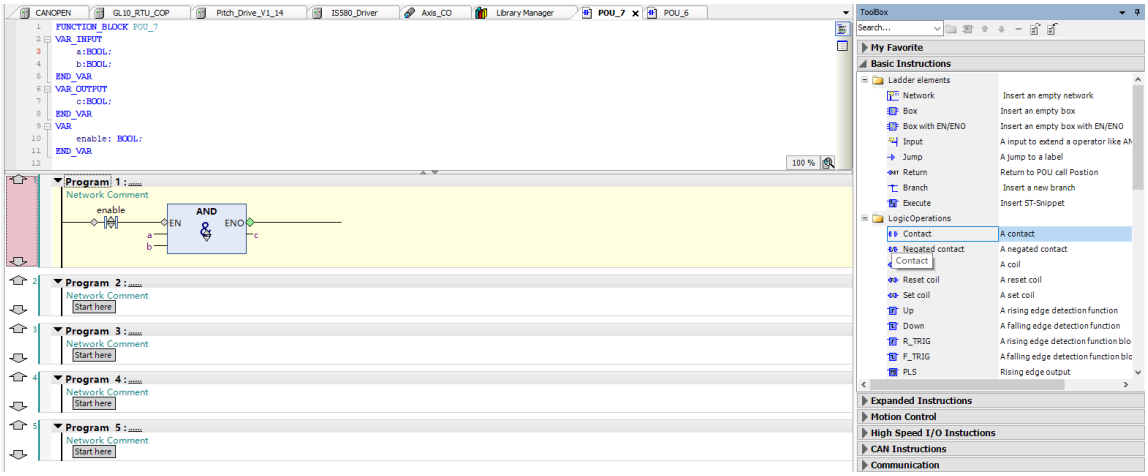
6.3.9 Drag and Drop

The LD supports the drag and drop operation on elements. You can drag the elements in the toolbox and drop them onto the network, and drag and drop elements on the LD page or across pages.

When you drag and drop an element, the LD page displays the available positions. The available positions are displayed in the following three modes:

- Diamond: . You can drag and drop an element onto the current position and insert it in serial mode.
- Upper and lower triangles: . You can insert a parallel element above or below the current element.
- Upper and lower arrows: . You can add a network in the upward or downward direction.

When you drag and drop an element to the insertion position, each figure changes to green inside, for example, , indicating the insertion position. The following figure shows the drag and drop process.



① Insert a parallel connection. ② After dragging, the color changes to green to indicate the insertion position. ③ Insert a contact.

Drag and drop elements from the toolbox

You can drag and drop elements from the toolbox to the LD editor.

The toolbox provides basic instructions, expansion instructions, motion control, high-speed I/O, CANopen axis control instructions, communication instructions, and POUs. You can customize types and instructions, or add instructions to the toolbox.

LD elements are provided in basic instructions.

POUs mainly contain the program, function block, function, method, and action defined in the current project. Up to 200 POUs can be displayed. If this limit is exceeded, POUs are not displayed, to avoid content disorder.

When you drag and drop elements, the available positions are displayed. Observe the following rules:

- You can drag and drop contacts to contacts and operation blocks (including execution blocks) for parallel connection, and to lines for serial connection.
- You can drag and drop operation blocks to contacts and operation blocks (including execution blocks) for parallel connection, and to lines for serial connection.
- You can drag and drop coils to non-parallel closed branches and input lines of non-multi-line operation blocks for serial connection. You can also drag and drop coils above or below other coils, return elements, and jump elements.

Drag and drop elements on the edit page

On the LD page, you can drag and drop the selected element from one position to another. You can drag and drop elements within the current edit page or to another LD edit page.

You can select one or more elements for the drag and drop operation.

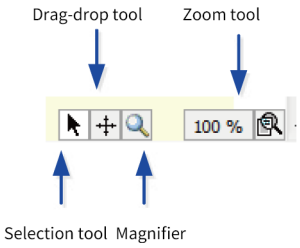
The drag and drop operation is divided into normal drag and drop and copy-type drag and drop (press Ctrl for drag and drop). During normal drag and drop, after the selected element is dragged and dropped, it is deleted from the original position. During copy-type drag and drop, after the selected element is dragged and dropped, it is still retained at the original position.

The drag and drop function is implemented in the standard manner.

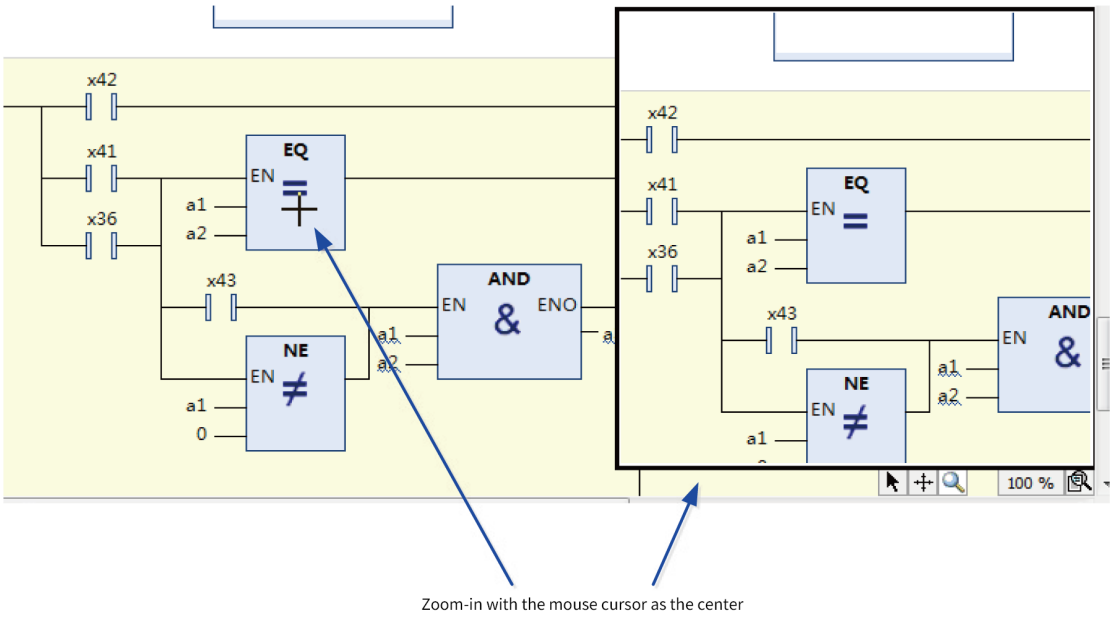
The drag and drop rules for one or more selected elements are the same as those of the paste operation by the standard edit command.

6.3.10 Graphic Display Tool

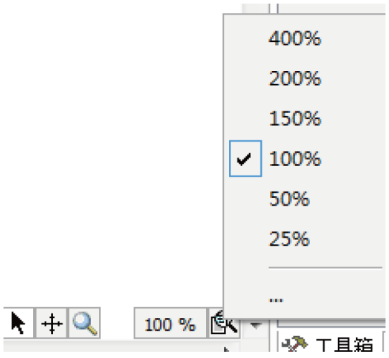
The LD graphic display tools are used to control the LD display mode, including the selection tool, drag and drop tool, magnifier tool, and zooming tool. By default, the LD adopts the selection tool. The graphic display tools are displayed in the lower right corner of the LD page, as shown in the following figure.



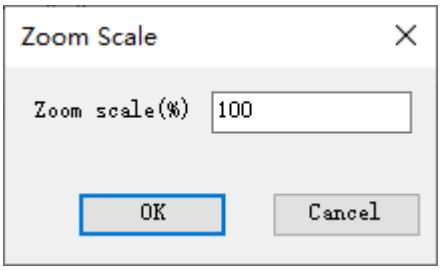
- Selection tool
  The selection tool is the default displayed tool. In selection tool mode, the cursor is displayed as . You can select elements for editing.
- Drag and drop tool
  In drag and drop tool mode, the cursor is displayed as . You can perform the drag and drop operation in areas.
- Magnifier
  In magnifier mode, the cursor is displayed as . Content is magnified with the cursor at the center, as shown in the following figure.



- Zooming tool
  The zooming tool displays the zoom scale of the current page and allows you to set the zoom scale, as shown in the following figure.



Click "...". The "Zoom Scale" dialog box is displayed. Enter a zoom scale, as shown in the following figure.

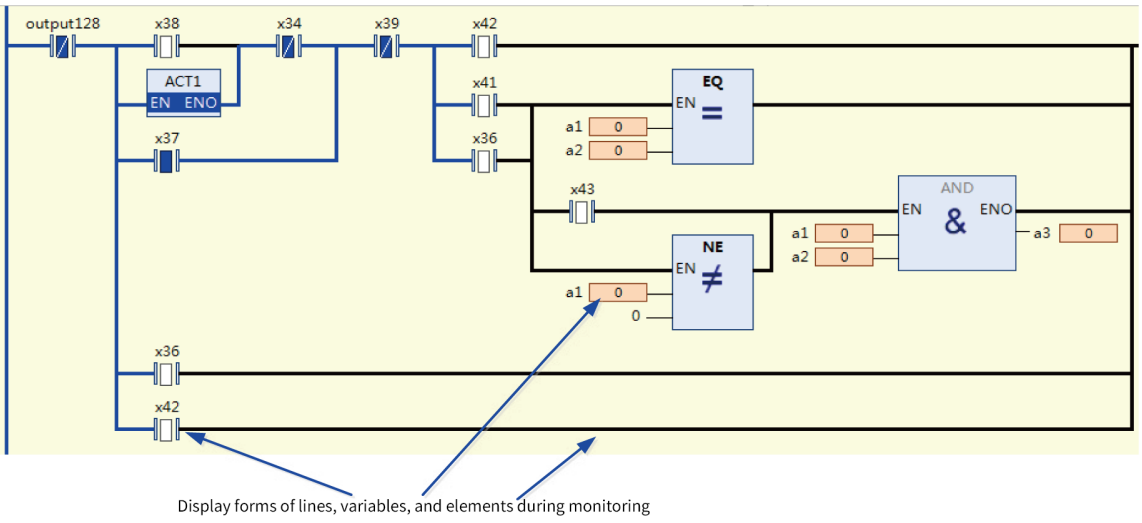


6.3.11 LD Debugging

The LD provides powerful debugging functions. In addition to the existing monitoring table, the LD also provides online monitoring, operand writing, mandatory value writing, breakpoint, and single step debugging.

Monitoring

In online mode, the LD page displays the execution results of lines, elements, and operand variables in specific forms, as shown in the following figure.



- Line monitoring

  1. BOOL-type lines are displayed in blue in the bold form in the conducting state (the value is "TRUE"); otherwise, they are displayed in black in the bold form.
  2. Non-BOOL-type lines (operation block input, output integer variable, time-type variable, and floating point variable) are displayed as fine lines. When the value is 0, they are displayed as black fine lines. When the value is not 0, they are displayed as blue fine lines.

- Element monitoring

  1. When the contact is energized, the NO contact is displayed as , or the NC contact is displayed as ; When the contact is not energized, the NO contact is displayed as , or the NC contact is displayed as .

2. When the coil is energized, the normal coil is displayed as , or the negated coil is displayed as . When the coil is not energized, the normal coil is displayed as , or the negated coil is displayed as .
3. The logic of the EN/ENO operation block is executed only when EN is "TRUE". To allow you to understand the execution state of the EN/ENO operation block (whether it is enabled), we differentiate the text of the operation block type. If the operation block is executed (EN is "TRUE"), the operation block type is displayed in black text. If the operation block is not executed, the operation block type is displayed in gray text (the block is disabled), as shown in the following figure.

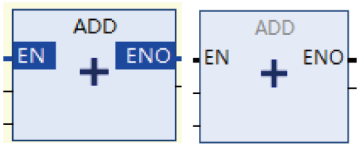


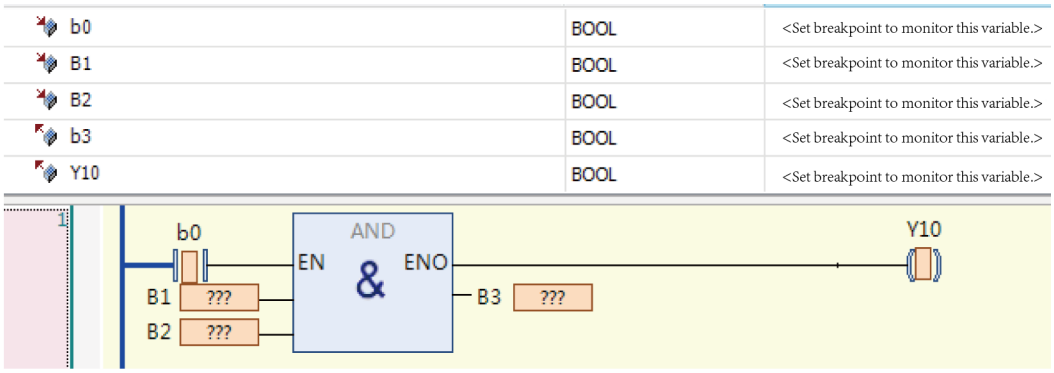
Figure 6-6 Operation block executed Operation block not executed

- Variable monitoring

1. Monitored variables are displayed in different widths, depending on the specific type, to reduce space usage. For variable-length elements such as strings and enumerated elements (the enumeration name is displayed), the default length is 12 characters. If the displayed content is incomplete, "..." is displayed, and the complete content is displayed in a prompt. For fixed-length elements such as integers and floating point numbers, content is displayed based on the maximum length.
2. You can drag and drop monitored variables to the monitored variable list.
3. To change the variable display mode, choose "Debug" > "Display Mode".

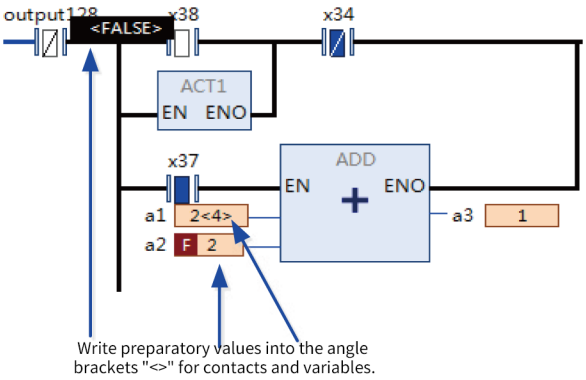
Note

Functions and methods are executed instantly, resulting in only temporary data. Therefore, functions and methods cannot be monitored directly after login. To monitor functions and methods, you need to add breakpoints to the functions and methods to interrupt execution before monitoring, as shown in the following figure.

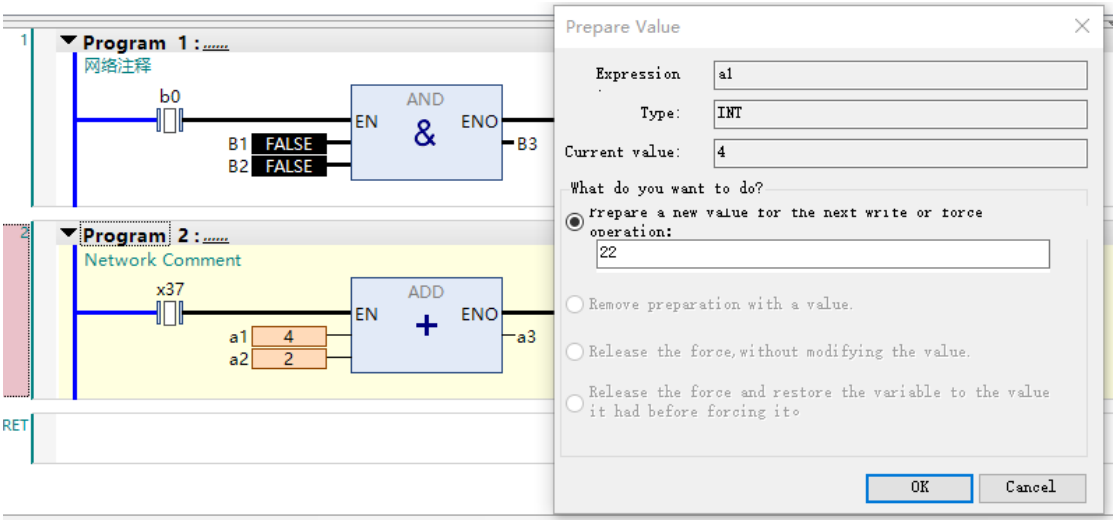


Mandatory value writing

You can write preparatory values to contacts, coils, and variables of the LD. Then, run the "Write values" or "Force values" command in the "Debug" menu to write or enforce values to variables. Before writing or enforcing values, you need to write preparatory values, as shown in the following figure.



- For contacts, coils, and BOOL-type variables, double-click the element or variable value position to switch between the "TRUE" and "FALSE" preparatory values. For example, you can double-click in the middle of a contact or coil to switch the preparatory value.
- Double-click the value position of a non-BOOL variable. The "Prepare Value" dialog box is displayed. Enter a preparatory value, as shown in the following figure.

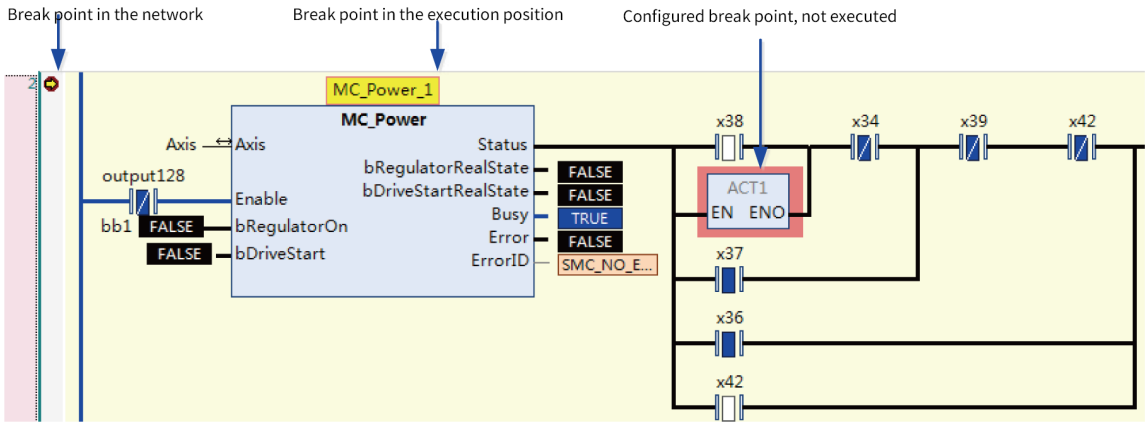


- After a mandatory value is written, the F identifier is added before the value to indicate it is a mandatory value.
- To release a mandatory value, choose "Online" > "Unforce values".

Breakpoint

The LD supports the breakpoint function. After a breakpoint is added, program execution automatically stops at the breakpoint, and you can debug the program. Operations such as jump-in, skip, jump-out, and run-to-cursor are supported.

After a breakpoint is added, the breakpoint position (element) is marked by a rectangular box in light red. When program execution reaches the breakpoint, the breakpoint position is marked by a rectangular box in yellow. If a breakpoint exists in the network, a circle is displayed in the network decoration area, as shown in the following figure.



The LD is graphical and a breakpoint can be added only at a position with a logical statement. Logical statements exist only in limited areas of the LD for optimized performance. That is, breakpoints can be added only in limited areas. For example, breakpoints cannot be added at the contact position or non-EN/ENO operator block position.

Breakpoints exist in places with possible variable value change, program branches, POU call position, and places where output variables are assigned values. Choose "View" > "Breakpoints" to open the "Breakpoints" dialog box and view all possible breakpoint positions.

Breakpoints can be added at the following positions:

- Network start position, which is the position of the first possible breakpoint in the network. When a breakpoint is added to a network, it is added to the first breakpoint position.
- Operation blocks not including the EN/ENO operator, such as FB, action, program call, and execution block.
- Coil, return, and jump element positions.

6.3.12 LD Data Update

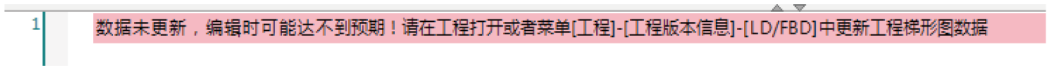
For InoProShop V1.4.0 and earlier versions, such as InoProShop V0.0.9.10, InoProShop V1.1.0, InoProShop V1.2.0, InoProShop V1.2.60.0, and InoProShop V1.2.70.1, the accessed LD data must be updated before you can use the functions of the optimized LD version.

LD data can be updated in the following two ways:

- In the "Project Version Information" dialog box that is displayed when a project is opened, click the LD/FBD tab. Select all the update marks shown in the following figure, and then click "OK".
- Choose "Project" > "Project Version Information". In the "Project Version Information" dialog box displayed, click the LD/FBD tab. Select all the update marks, and then click "OK".



If the LD data is not updated, update description is displayed in the first network.



Note

LDs must be updated before use.

7 Diagnosis

7.1 Overview

Diagnosis aims to quickly locate errors while the PLC is running so that you can find solutions based on error information and states. The InoProShop diagnosis page can be accessed and displayed only when you log in to the PLC.

The InoProShop programming system can diagnose various communication devices and generate messages indicating faults, disconnection state, and other errors based on the running state of devices.

Modules involved in fault diagnosis include CPU module, Modbus module, Modbus TCP module, EtherCAT module, CANopen module, CANlink module, and PROFIBUS-DP module.

The InoProShop programming system allows you to obtain diagnosis information through the configuration diagnosis, list of diagnosis information, list of device self-diagnosis information, or diagnosis programming interface.

All diagnosis information is parsed and obtained through diagnosis codes. Diagnosis codes correspond to diagnosis programming interfaces.

7.2 Configuration Diagnosis

7.2.1 Overview

Configuration can be classified into network configuration and hardware configuration. The corresponding diagnosis can be classified into network configuration diagnosis and hardware configuration diagnosis. In configuration, the diagnosis state of each communication module is displayed through different icons: "Running", "Stopped", "Disconnected", and "Faulty".

: Running state: The device is running without faults.

: Stopped state: The device is stopped.

: Disconnected state: The device is disconnected or the device does not exist.

: Faulty state: The device is faulty and cannot run.

The device state is displayed on the configuration page.

7.2.2 Network Configuration Diagnosis

You can configure a PLC bus system, activate the bus, and add slaves on the "Network Configuration" page. Log in to the system and access the "Network Configuration" page. The diagnosis state of each communication device is displayed, as shown in the following figure.

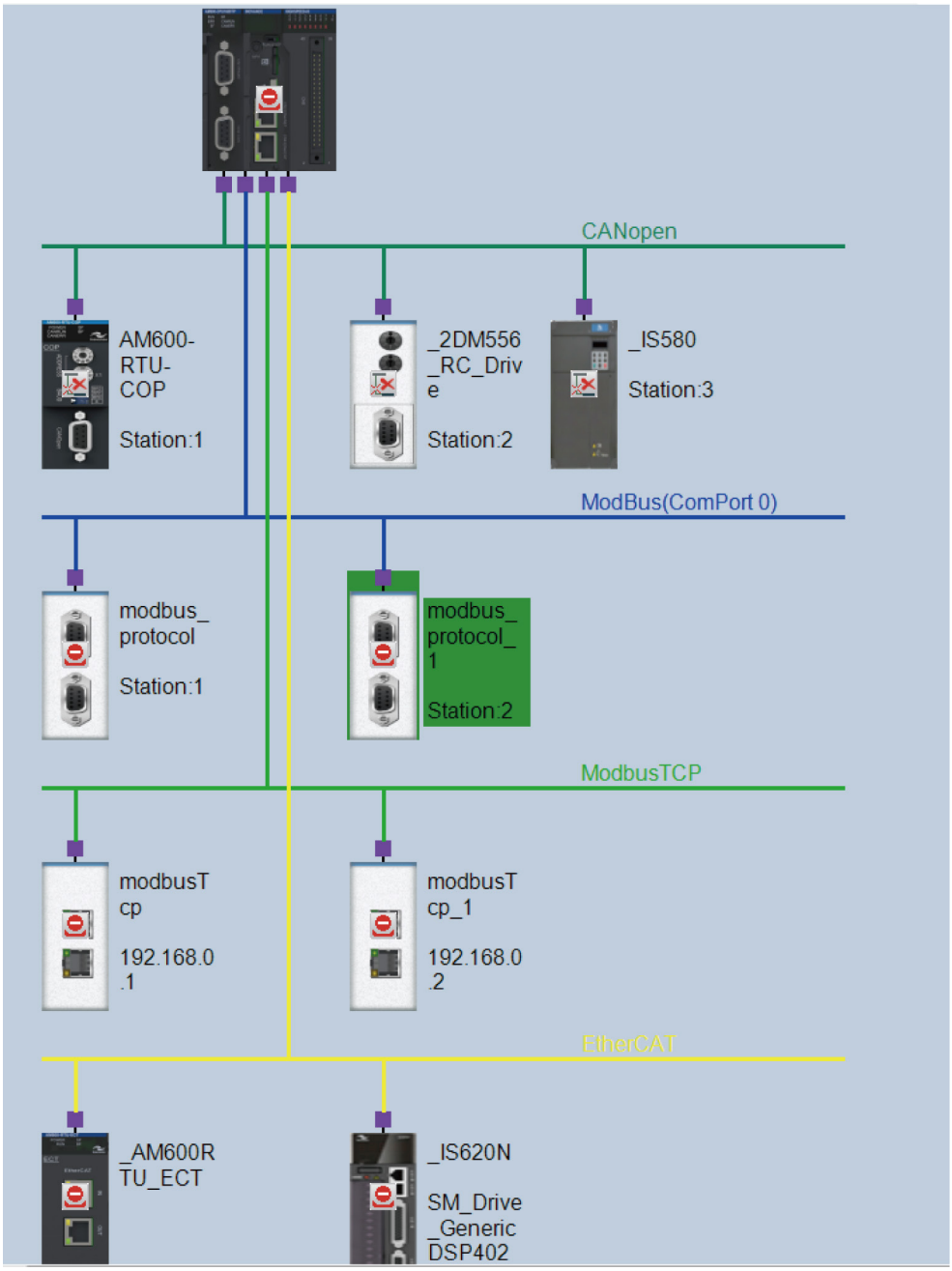


Figure 7-1 Network configuration diagnosis

After login, the state of each slave or CPU is displayed on the "Network Configuration" page: "Running", "Faulty", or "Disconnected". For details about network configuration, see hardware configuration.

7.2.3 Hardware Configuration Diagnosis

Hardware configuration is mainly used to add expansion modules corresponding to the bus, including local I/O hardware configuration, EtherCAT hardware configuration, and CANopen hardware configuration. CANlink, Modbus, and Modbus TCP modules are displayed only on the "Network Configuration" page. You can double-click a network configuration sub-node or slave module to access the "Hardware Configuration" page, or select another hardware configuration mode on the "Hardware

Configuration" page. The hardware configuration diagnosis is similar to the network configuration diagnosis. The following figure shows CANopen hardware configuration diagnosis.

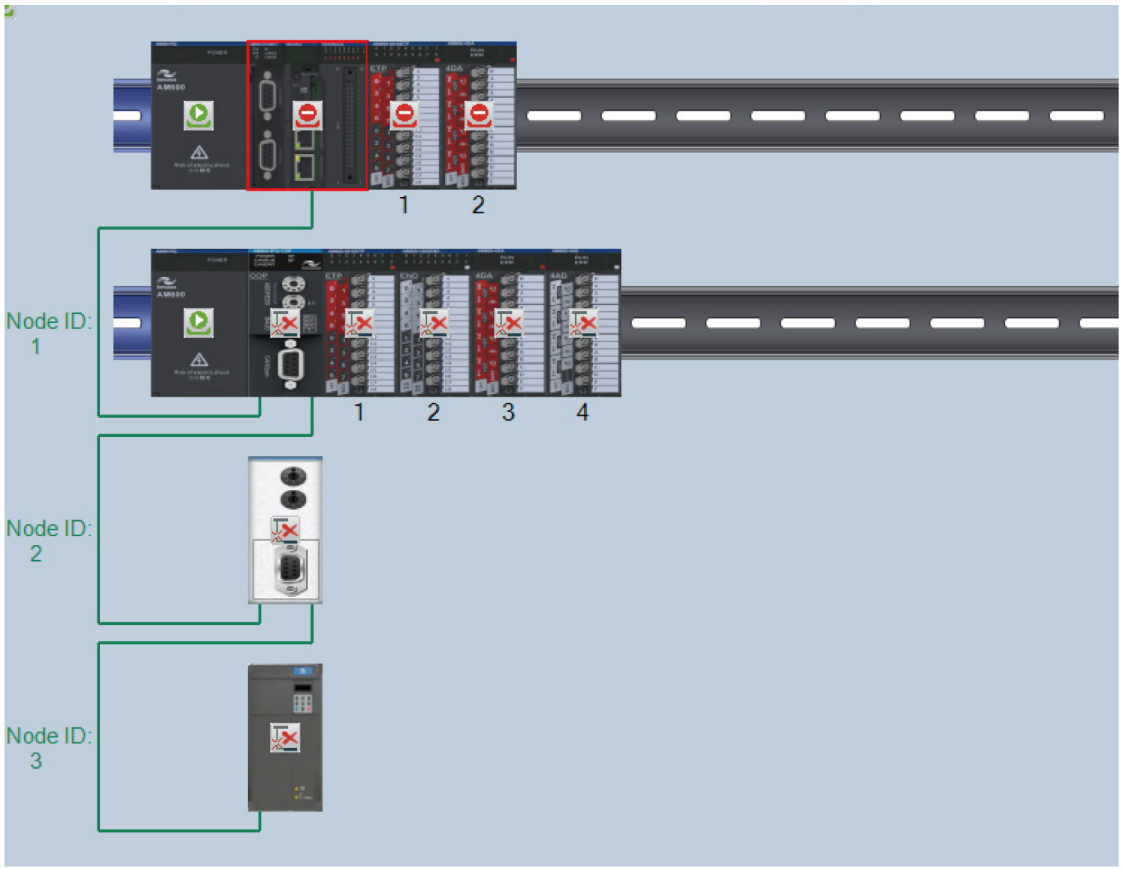
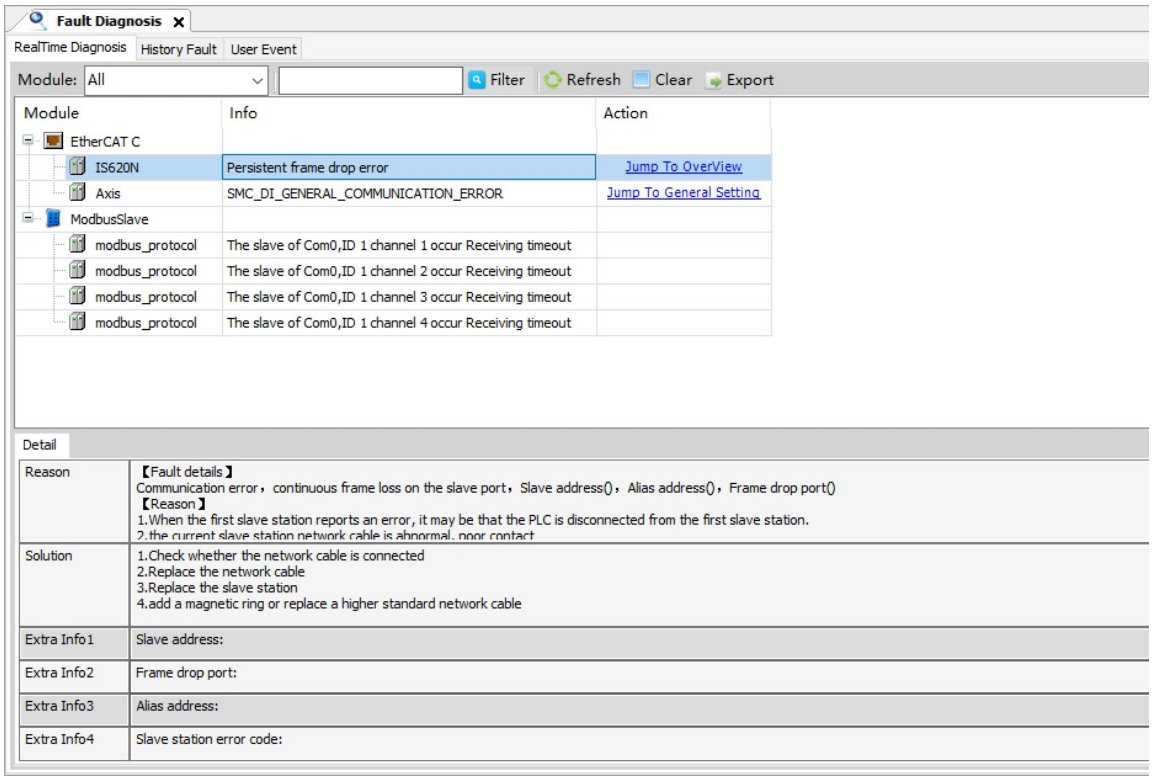


Figure 7-2 CANopen hardware configuration diagnosis

7.3 Fault Diagnosis

Fault diagnosis displays all device fault information, provides details about faults and troubleshooting, and provides diagnosis details under special circumstances.

After the device is connected, you can choose "Tools" > "Troubleshooting" to access the "Fault Diagnosis" page. The following figure shows the "Fault Diagnosis" page.



Three tabs are provided on this page:

- RealTime Diagnosis: Displays all faults of the device to which you have logged in.
- History Fault: Displays all history faults of the device to which you have logged in.
- User Event: Displays all history operation records made by users to the device to which you have logged in.

RealTime Diagnosis

1. Click "RealTime Diagnosis". The "RealTime Diagnosis" page is displayed.
2. On this page, you can set a filter to display modules of which fault information you want to display.

   - From the drop-down list of "Module", select a module type. Options are "All", "CPU Module", "Modbus Module" "Modbus TCP Module", "Local Module", and "EtherCAT Module". The default option is "All".
   - In the text box, enter the name of a faulty module or keywords in fault information, and then click "Filter".

   The fault information list displays relevant information such as the module name, fault information, and action based on the selected module type or keywords.

Note

- To refresh the module fault information list, click "Refresh". The latest module fault information is displayed in the list.
- To clear the module fault information list, click "Clear".
- To export module fault information from the list, click "Export".

3. In the fault information list, click a fault record. The "Detail" and "Deep Diagnosis" tabs are displayed in the lower part of the page, through which you troubleshoot the fault based on the diagnosis information.

- Click "Detail". On the tab page displayed, the fault reason, solution, and more information are displayed.
- Click "Deep Diagnosis". On the tab page displayed, the diagnosis information of complex faults is displayed.

Note

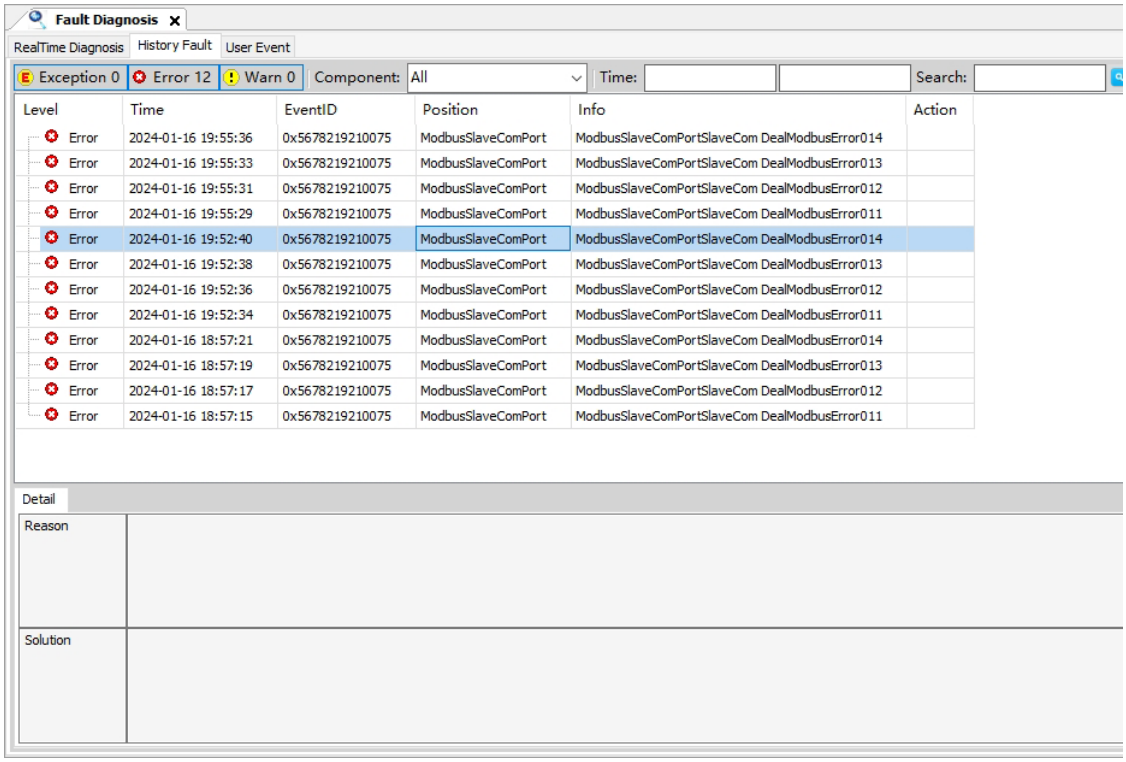
The "Deep Diagnosis" tab is available for some faults of EtherCAT, such as the GL10 and GL20 modules.

---

4. (Optional) In the "Action" column, click the link to jump to the corresponding page.

History Fault

1. Click "History Fault". The "History Fault" page is displayed.
   At the top of the page, the number of faults of different fault levels is displayed. Fault levels include "Exception", "Error", and "Warn".



Note

If no fault information is displayed, click "Refresh".

---

2. Set a filter to display modules of which fault information you want to display.

- Click the corresponding fault level button to display or hide information of faults of this level.

- From the drop-down list of "Component", select a module type. Options are "All", "CPU Module", "Modbus Module" "Modbus TCP Module", "Local Module", and "EtherCAT Module". The default option is "All".
- In the text box of "Time", enter a start time and an end time. In the text box of "Search", enter keywords in the fault information, and then click "Filter".

The fault information list displays relevant information such as the fault level, occurrence time, event ID, device location, fault information, and action based on the module level, module type, time range, or keywords entered.

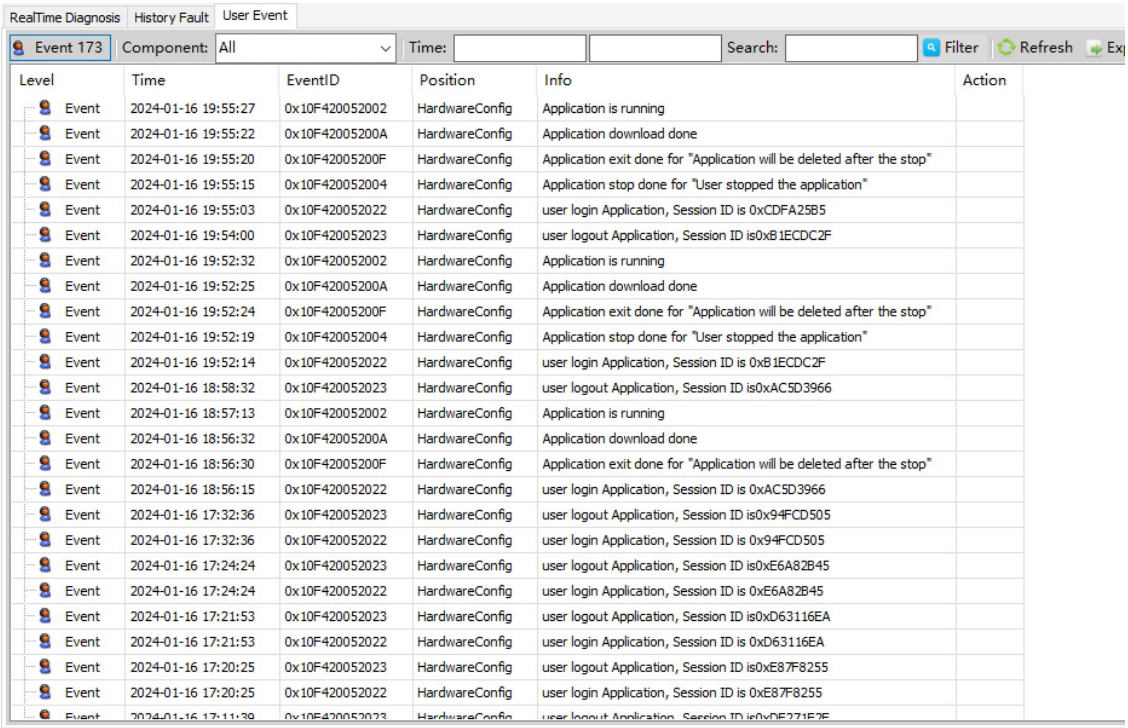
Note

- To refresh the module fault information list, click "Refresh". The latest module fault information is displayed in the list.
- To export module fault information from the list, click "Export".

3. In the fault information list, click a fault record. The fault details are displayed in the lower part of the page, including the cause and solution. Troubleshoot the fault based on such information.
4. (Optional) In the "Action" column, click the link to jump to the corresponding page.

User Event

1. Click "User Event". The "User Event" page is displayed.
   At the top of the page, the number of events of different levels is displayed. Event levels are classified into "Event" and "Information". The information level is not displayed by default.



Level	Time	EventID	Position	Info	Action
Event	2024-01-16 19:55:27	0x10F420052002	HardwareConfig	Application is running	
Event	2024-01-16 19:55:22	0x10F42005200A	HardwareConfig	Application download done	
Event	2024-01-16 19:55:20	0x10F42005200F	HardwareConfig	Application exit done for "Application will be deleted after the stop"	
Event	2024-01-16 19:55:15	0x10F420052004	HardwareConfig	Application stop done for "User stopped the application"	
Event	2024-01-16 19:55:03	0x10F420052022	HardwareConfig	user login Application, Session ID is 0xCDFA25B5	
Event	2024-01-16 19:54:00	0x10F420052023	HardwareConfig	user logout Application, Session ID is0xB1ECDC2F	
Event	2024-01-16 19:52:32	0x10F420052002	HardwareConfig	Application is running	
Event	2024-01-16 19:52:25	0x10F42005200A	HardwareConfig	Application download done	
Event	2024-01-16 19:52:24	0x10F42005200F	HardwareConfig	Application exit done for "Application will be deleted after the stop"	
Event	2024-01-16 19:52:19	0x10F420052004	HardwareConfig	Application stop done for "User stopped the application"	
Event	2024-01-16 19:52:14	0x10F420052022	HardwareConfig	user login Application, Session ID is 0xB1ECDC2F	
Event	2024-01-16 18:58:32	0x10F420052023	HardwareConfig	user logout Application, Session ID is0xAC5D3966	
Event	2024-01-16 18:57:13	0x10F420052002	HardwareConfig	Application is running	
Event	2024-01-16 18:56:32	0x10F42005200A	HardwareConfig	Application download done	
Event	2024-01-16 18:56:30	0x10F42005200F	HardwareConfig	Application exit done for "Application will be deleted after the stop"	
Event	2024-01-16 18:56:15	0x10F420052022	HardwareConfig	user login Application, Session ID is 0xAC5D3966	
Event	2024-01-16 17:32:36	0x10F420052023	HardwareConfig	user logout Application, Session ID is0x94FCD505	
Event	2024-01-16 17:32:36	0x10F420052022	HardwareConfig	user login Application, Session ID is 0x94FCD505	
Event	2024-01-16 17:24:24	0x10F420052023	HardwareConfig	user logout Application, Session ID is0xE6A82B45	
Event	2024-01-16 17:24:24	0x10F420052022	HardwareConfig	user login Application, Session ID is 0xE6A82B45	
Event	2024-01-16 17:21:53	0x10F420052023	HardwareConfig	user logout Application, Session ID is0xD63116EA	
Event	2024-01-16 17:21:53	0x10F420052022	HardwareConfig	user login Application, Session ID is 0xD63116EA	
Event	2024-01-16 17:20:25	0x10F420052023	HardwareConfig	user logout Application, Session ID is0xE87F8255	
Event	2024-01-16 17:20:25	0x10F420052022	HardwareConfig	user login Application, Session ID is 0xE87F8255	

Note

If no event information is displayed, click "Refresh".

2. Set a filter to display events of which information you want to display.

- Click "Event" to display or hide information of events of the "Event" level.
- From the drop-down list of "Component", select the type of events you want to display. The default option is "All".
- In the text box of "Time", enter a start time and an end time. In the text box of "Search", enter keywords in the event information, and then click "Filter".

The event information list displays relevant information such as the event level, occurrence time, event ID, component location, event information, and action based on the event level, component type, time range, or keywords entered.

Note

- To refresh the event information list, click "Refresh". The latest event information is displayed in the list.
- To export event information from the list, click "Export".

3. (Optional) In the "Action" column, click the link to jump to the corresponding page.

7.4 Online Diagnosis

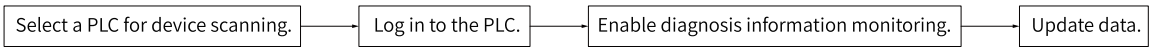
7.4.1 Overview

Online diagnosis is used to display the diagnosis information of programs, devices, and systems in real time after the PLC is connected. This function helps quickly locate and fix errors to ensure normal device running.

7.4.2 Diagnosis Procedure

General procedure

Scan devices and select a communication device. Wait for login and enable diagnosis information monitoring. After receiving the diagnosis information, refresh the data.



Scan procedure

Communication channels for the diagnosis function are divided into **standard communication channels** and **diagnosis communication channels**. Standard communication channels are existing communication channels and support all communication functions. Diagnosis communication channels are channels used for diagnosis when the PLC failed to communicate through standard communication channels during a system runtime error but can communicate with the network.

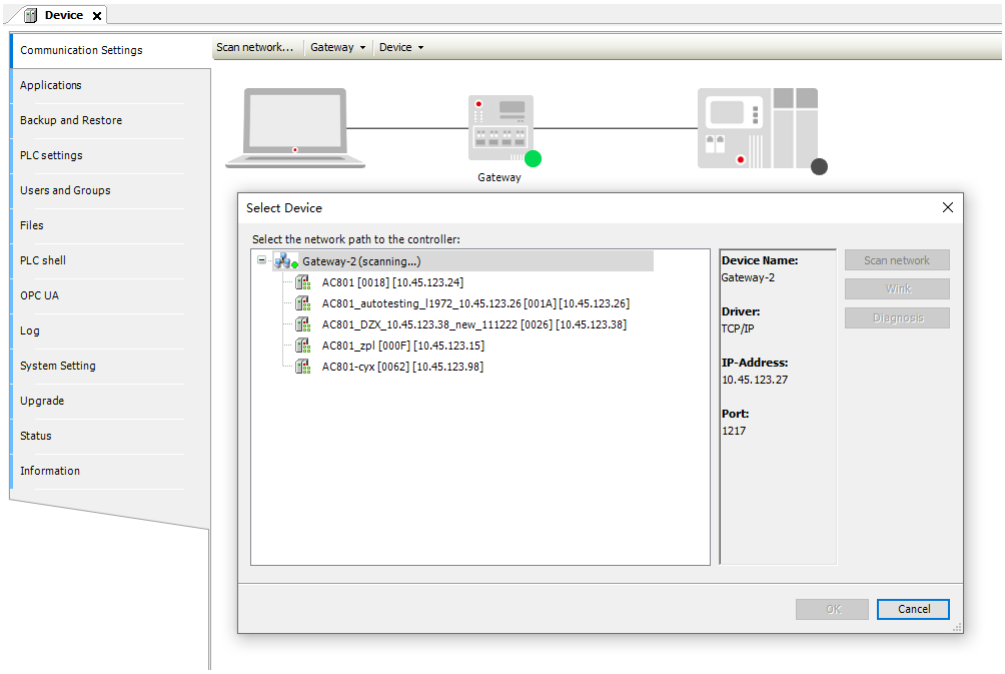
During device scanning, the system enables both the standard and diagnosis communication channels, and displays all devices scanned by standard communication channels as well as devices that are not scanned by standard communication channels but can be displayed in diagnosis communication channels.

After you select a device, the system performs communication based on the communication channel of the device. Currently, diagnosis communication channels only support online diagnosis and login state switching. The following table lists communication functions supported by different communication channels.

Communication Function	Standard Communication Channel	Diagnosis Communication Channel
Program download	Supported	Not supported
Online modification	Supported	Not supported
Monitoring	Supported	Not supported
Log refreshing	Supported	Not supported
Online diagnosis	Supported	Supported
Login state switching	Supported	Supported

7.4.3 Scanning Devices

Click "Scan network". On the "Select Device" page displayed, device scanning is enabled for both standard and diagnosis communication channels and the detected devices are displayed on the page, as shown in the following figure.



Information in black indicates the scanning results of standard communication channels, while information in light green indicates the scanning results of diagnosis communication channels. If a detected device is in a diagnosis communication channel, click "Diagnosis" to upload the diagnosis logs and troubleshoot the problem accordingly.

Note

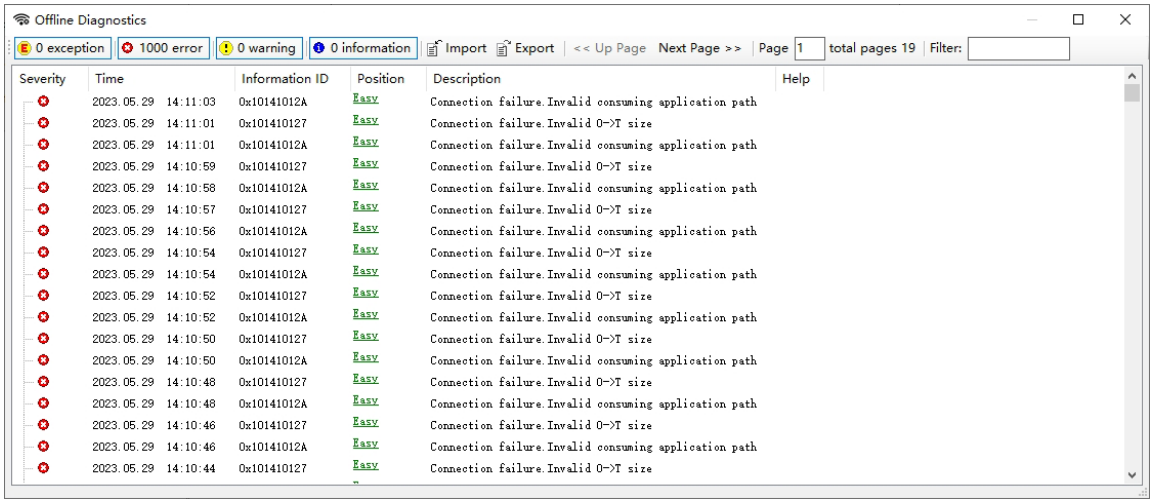
- Only the latest version (5) supports the device diagnosis and scanning function.
- Diagnosis communication channels do not support USB-based communication.

When a device detected by a diagnosis communication channel is selected, the communication mode in the status bar is displayed as "Diagnostic communication". After you log in to the PLC and communication is established between the device and the PLC, the system starts diagnosis communication, "Under diagnosis..." is displayed next to the node on the device tree, and the system gets the diagnosis data.

If a fault occurs on a device under a diagnosis channel before communication, the fault information may not be displayed. To view the fault information, view the history diagnosis information.

7.4.4 Logging in to PLC

If the software is connected to the PLC, after you click , the system starts diagnosis and the "Diagnosis" page is displayed, as shown in the following figure.



The following table lists the tabs on this page and their descriptions.

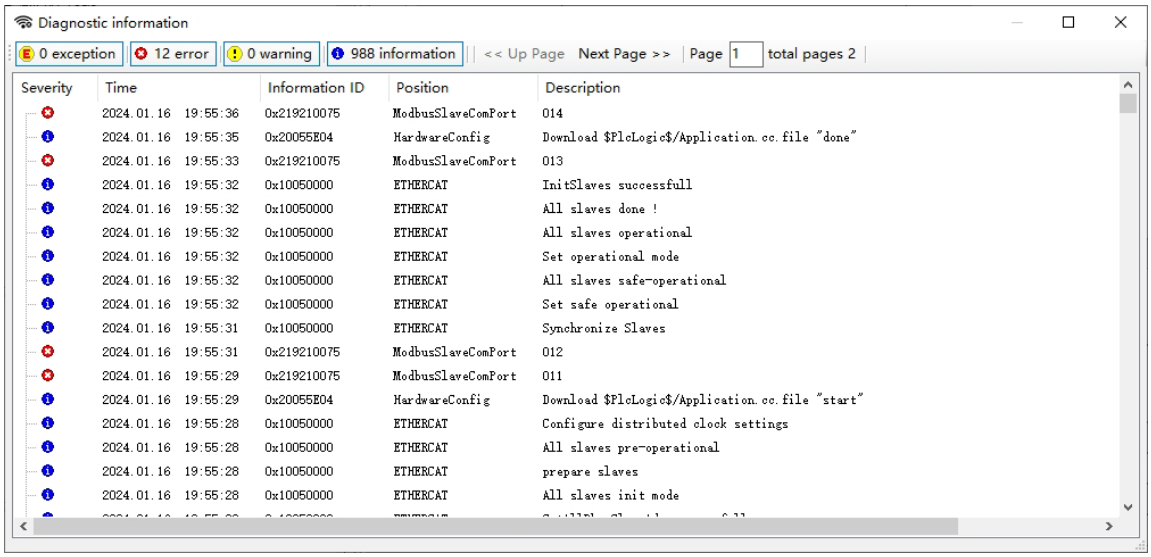
Parameter	Description
Exception/error/warning/information	Displays and filters information of different levels.
Clear	Clears the displayed diagnosis information. If new diagnosis information is generated after the clearing operation, the system displays the latest information.
Export	Exports all the diagnosis information to a CSV file.
History DIAG INFO	Displays the diagnosis information generated before login.
System DIAG INFO	Displays system diagnosis information that needs to upload exceptions among the diagnosis information.
Show All	Displays all the diagnosis information.
Offline Diagnosis	Imports diagnosis information from the saved CSV file in the offline state.

The following table lists the parameters on this page and their descriptions.

Parameter	Description
Severity	Displays the information level.
Time	Displays the information occurrence time.
Information ID	Displays the event ID in the information.

Parameter	Description
Position	Displays the fault occurrence position. When a diagnosis information supports jumping, the information in the "Position" column is underlined. Double-click the line to jump to the corresponding position. When an exception file exists and needs to be uploaded, you can double-click the line to upload the exception file.
Description	Describes the symptoms and causes of the faults.
Help	Displays the fault solution or handling procedure.

In the online state, you can export all the online diagnosis information to a CSV file. Then, you can import the saved information on the "Diagnostic information" page to display all the diagnosis information. The following page shows the "Diagnostic information" page.



7.5 List of Device Self-Diagnosis Information

7.5.1 CPU Diagnosis

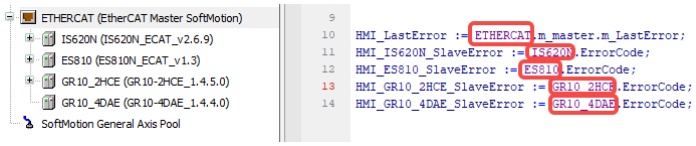
No diagnosis page is available for a CPU. You can check diagnosis information on the list of diagnosis information.

For CPU diagnosis codes and diagnosis information, see *"9.8.2 CPU Diagnosis Code" on page 509*.

7.5.2 EtherCAT Diagnosis

EtherCAT diagnosis is used to record and describe bus errors, including master diagnosis, slave diagnosis, slave module diagnosis, and slave servo drive diagnosis. EtherCAT diagnosis only parses errors of Inovance slaves. For details about diagnosis methods, see *"7.3 Fault Diagnosis" on page 440*. Error IDs are listed in *"9.8.7 EtherCAT Diagnosis Code" on page 515*.

In some application scenarios, error IDs are displayed on the touchscreen. You only need to assign the variable "m_LastError" (the EtherCAT bus error ID) and the EtherCAT slave error ID to a variable associated with the HMI address. "HMI_LastError" and "HMI_IS620N_SlaveError" are word variables associated with the HMI address. The bus error ID and slave error ID diagnosed for EtherCAT are displayed on the touchscreen.



AM600 EtherCAT slave diagnosis

The following table lists the CANopen Emergency frame formats corresponding to the EtherCAT AM600 slave.

Emergency Error Code		Error Register	Manufacturer-Specific Code				
BYTE0	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	BYTE7
BaseInfo	SlaveError	0x80	InterCommEr-ror	ConformenceError		IOModulePosError	

Note

The error register value 0x80 indicates the "Emergency" frame of the slave.

"BaseInfo" is not in use. For other diagnosis codes and diagnosis information, see *" " on page* . If the device is diagnosed in this format, the data is parsed into corresponding diagnosis information. Otherwise, the data is parsed into a message "Device faulty".

7.5.3 I/O Diagnosis

The I/O module can be added to the CPU, CANopen AM600 slave, PROFIBUS-DP AM600 slave, or EtherCAT AM600 slave. They share the same diagnosis information. For diagnosis codes and diagnosis information, see *"9.8.3 I/O Module Diagnosis Code" on page 512*.

For descriptions of the self-diagnosis page, see the overview of the list of device self-diagnosis information.

7.5.4 PROFIBUS-DP Diagnosis

The PROFIBUS DP diagnosis refers to the PROFIBUS-DP slave diagnosis. Data is included in the diagnosis array. Each slave has a "Slave Diagnosis" page, as shown in the following figure, on which slave diagnosis information is displayed. For non-AM600 modules of the slave, diagnosis information is displayed on the "Slave Diagnosis" page. For the I/O module of the AM600 PROFIBUS-DP slave, diagnosis information is displayed on the diagnosis page of the I/O module. For details, see *"7.5.3 I/O Diagnosis" on page 448*.

The PROFIBUS-DP channel diagnosis can be classified into defined channel diagnosis and GSD file-defined channel diagnosis. For details, see "PROFIBUS-DP Diagnosis Overview".

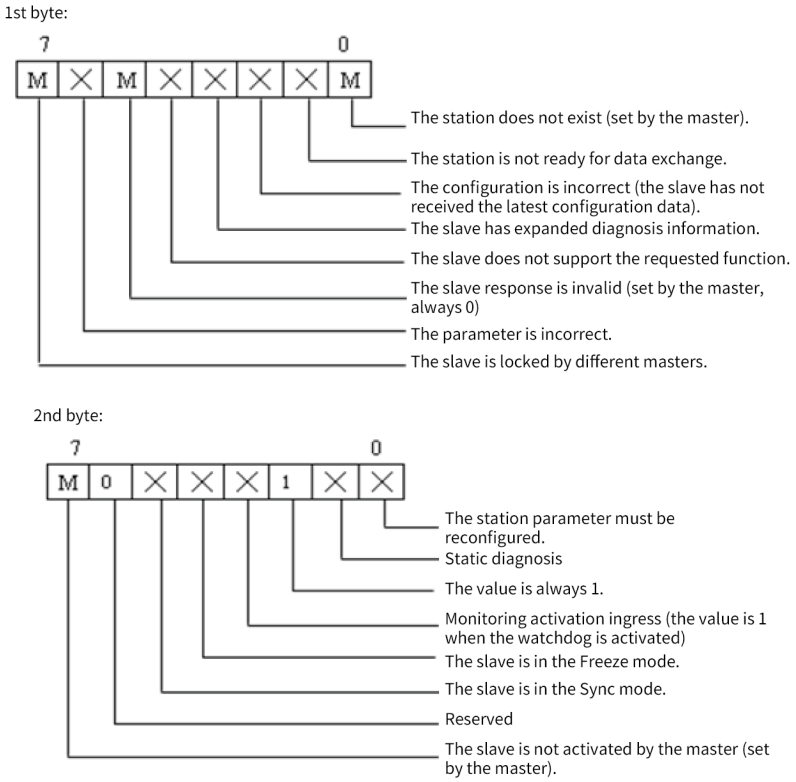
- Master address: Indicates the address of the master, which corresponds to the 4th byte in the diagnosis array.
- ID: Indicates the slave-defined ID, which corresponds to the 5th and 6th bytes in the diagnosis array. The GSD file also defines the ID.
- Hex format: Diagnosis array data is displayed in hexadecimal format.
- Standard diagnosis: Indicates state diagnosis and identifier diagnosis included in the basic diagnosis and expanded diagnosis of slaves. For details, see "PROFIBUS-DP Diagnosis Overview".
- Channel diagnosis: Indicates the channel diagnosis included in the expanded diagnosis of slaves, including defined channel diagnosis and GSD file-defined channel diagnosis. For details, see "PROFIBUS-DP Diagnosis Overview".

The following table shows the diagnosis array structure.

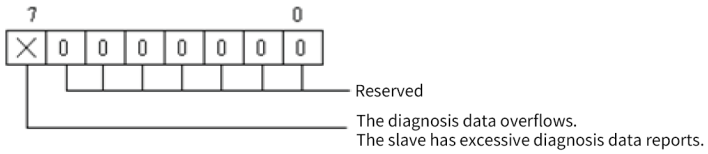
Table 7–1 Diagnosis array structure

First Six Bytes Indicating Basic Diagnosis Information (Mandatory)	Alarm or State Information Block (4-63 Bytes) (Optional)	Flag Module Diagnosis Information Block (Optional)	Channel Diagnosis Information Block (3 Bytes per Channel) (Optional)
1————————————- ————————————6 basic diagnosis information	7————————————————————————————————————- ————————————244 expanded diagnosis information		
A data unit (DU) contains a minimum of 6 bytes and a maximum of 244 bytes.			

1. Basic diagnosis information



3rd byte:



4th byte: the master address, ranging from 0 to 7Dh (0 to 125). The value FFh (255) indicates that the slave is not controlled by any other slaves or is not set.

5th byte: the high-order byte of the slave PROFIBUS ID, ranging from 0 to FFh (0 to 255).

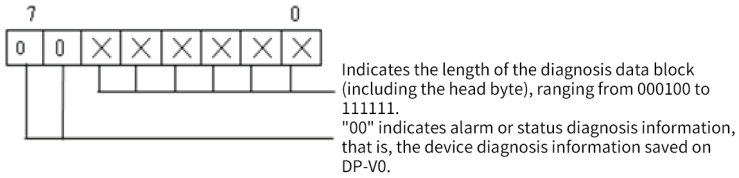
6th byte: the low-order byte of the slave PROFIBUS ID, ranging from 0 to FFh (0 to 255).

2. Expanded diagnosis information

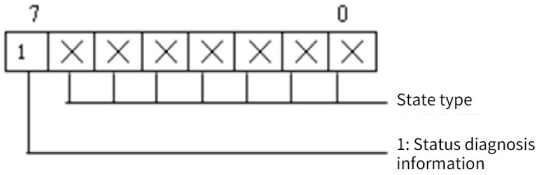
Expanded diagnosis includes state diagnosis, identifier diagnosis, and channel diagnosis.

- State diagnosis

  7th byte



  8th byte



  When bit 7 is 1, it indicates state diagnosis information, and bits 0 to 6 correspond to the following state information types, respectively.

  0: Reserved

  1: Indicates that the byte corresponding to state details is followed by state information.

  2: Indicates that the byte corresponding to state details is followed by module state information (affecting the bytes following the 9th byte).

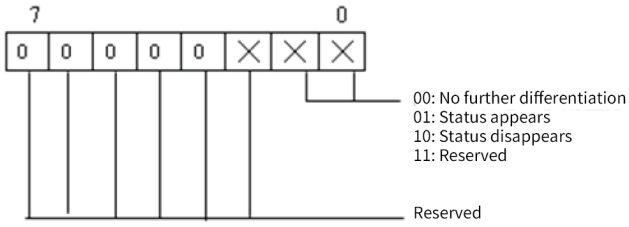
  3 to 31: Reserved

  32 to 126: Indicates that the byte corresponding to state details is followed by special manufacturer data.

  127: Reserved

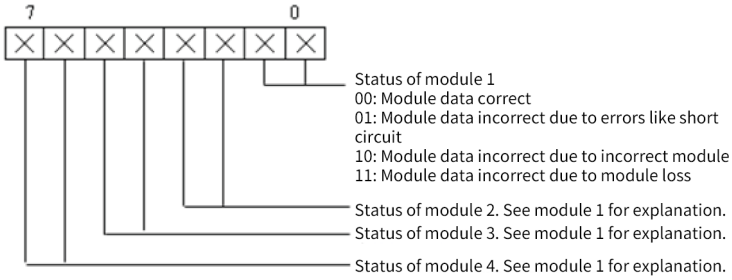
  9th byte: Indicates the slot number of the slave reporting an error, ranging from 0 to 254.

  10th byte: Indicates detailed features of a state.

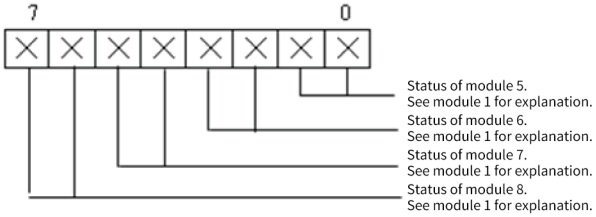


Byte following the 11th byte: Indicates a user data byte.

If the 8th byte corresponds to state type 2, that is, module state information, the 9th byte is 0; that is, the slave slot number is 0. Therefore, bytes following the 11th byte are no longer user data bytes. The following describes the structure and definition.



12th byte: Indicates the state of modules 5 to 8.

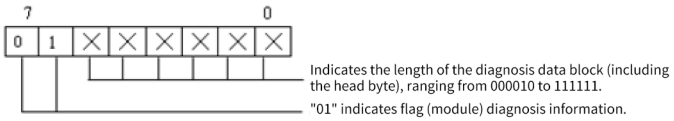


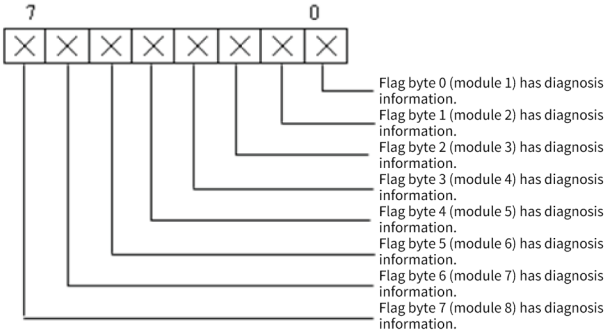
Subsequent bytes can be arranged based on the preceding rule until information on all modules is entered.

- Identifier diagnosis

Head Byte	Diagnosis Data Byte Relating to Flag (Module)
01××××××	Bytes 1 to 62

Similar to the preceding head byte, this head byte indicates the type and length (number of bytes, including the head byte itself) of the flag diagnosis information in the following format.



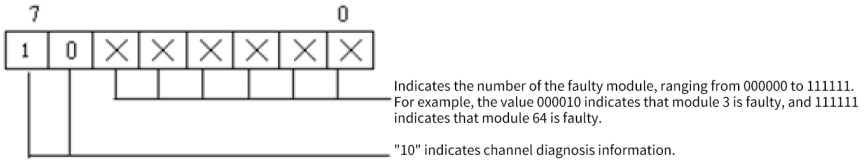


If the number of modules exceeds 8, you can continue using bytes that follow to specify flag byte numbers (or module numbers).

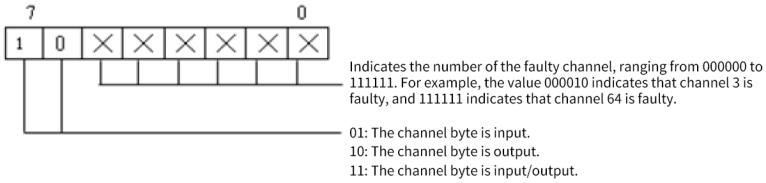
- Channel diagnosis
  Each piece of channel diagnosis information contains 3 bytes. Channel diagnosis includes diagnosis information for multiple channels. The following table shows the structure of diagnosis information for one channel.

Head Byte	Diagnosis Data Byte Relating to Channel
10××××××	2 bytes (3 bytes if the head byte is included)

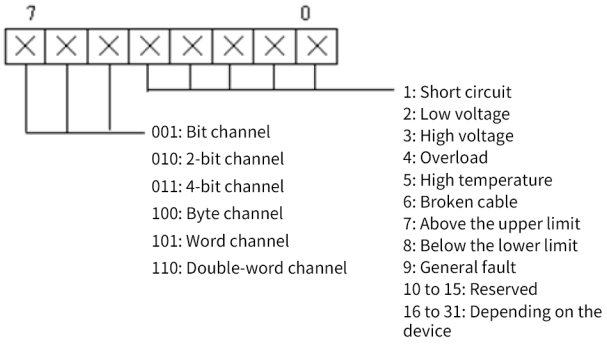
The head byte specifies the type of channel diagnosis information and the number of a faulty module. The following describes the structure and definition of the head byte.



The 2nd byte indicates the channel byte type in the following format:



The 3rd byte indicates the channel byte length and fault type in the following format:



7.5.5 Modbus RTU Diagnosis

Modbus RTU supports buses of Modbus serial ports 0 and 1. Modbus serial port 0 or 1 can serve as a Modbus master or Modbus slave.

When a Modbus serial port serves as a master, you can add slaves (remote) to the master. On both master configuration page and slave configuration page, the "Device Diagnosis" option is available. Master diagnosis information is used to identify slave configuration faults without fault causes. Therefore, no fault codes are displayed on the "Device Diagnosis" page. On the "Device Diagnosis" page for a slave, fault information corresponding to specific configuration items is displayed.

When a Modbus serial port serves as a slave, a "Device Diagnosis" page is available, on which master-slave communication faults are displayed. For details, see the list of device self-diagnosis information.

The diagnosis codes and diagnosis information for a Modbus serial port remain unchanged whether it serves as a master or a slave. For details, see *"9.8.6 Modbus Diagnosis Code" on page 514*.

7.5.6 Modbus TCP Diagnosis

An AM600 PLC can serve as a Modbus TCP master or a Modbus TCP slave.

When a Modbus TCP device serves as a master, you can add slaves (remote) to the master. On both master configuration page and slave configuration page, the "Device Diagnosis" option is available. Master diagnosis information is used to identify slave configuration faults without fault causes. Therefore, no fault codes are displayed on the "Device Diagnosis" page. On the "Device Diagnosis" page for a slave, fault information corresponding to specific configuration items is displayed.

When a Modbus TCP device serves as a slave, a "Device Diagnosis" page is available, on which master-slave communication faults are displayed. For details, see the list of device self-diagnosis information.

The diagnosis codes and diagnosis information for a Modbus TCP device remain unchanged whether it serves as a master or a slave. For details, see *"9.8.6 Modbus Diagnosis Code" on page 514*.

7.5.7 CANlink Diagnosis

No "Device Diagnosis" page is available for CANlink devices. However, you can enable the monitoring function on the CANlink network management page to check the online state and running state of the slave. For details, see "CANlink Network Management".

You can check the state of CANlink stations through soft elements. For details, see *"4.9.4 CANlink Network Configuration" on page 271*.

After logging in to the PLC, you can view CANlink diagnosis information from the list of diagnosis information. For CANlink diagnosis codes and diagnosis information, see *"4.9.2 CANlink3_en.0 网络组成" on page 269*.

For descriptions of the self-diagnosis page, see the overview of the list of device self-diagnosis information.

7.6 Diagnosis Programming Interface

7.6.1 Overview

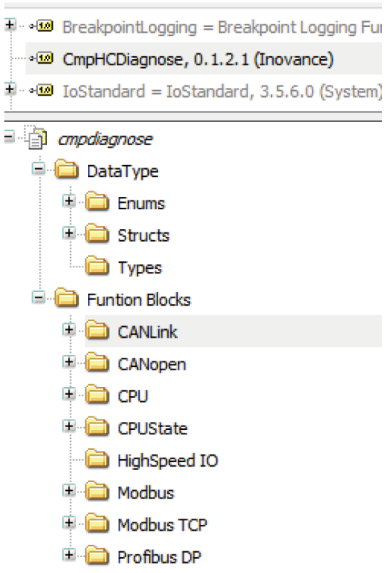
The diagnosis interface library "CmpHCDiagnose" only supports PLCs of the AM400 and AM600 series. The CANopen diagnosis function block "GET_CANOPEN_SALVE_DIAGNOSE" is not applicable to software of 1.3.0 or later versions. To diagnose slave states, you need to add the "CmpHCCiA402" library and call the "GET_STATE" interface.

The "SysHCPlcInfo" library can get master fault diagnosis information and system hardware and software information, and supports Modbus and Modbus TCP diagnosis for PLCs of the AM400/AM600/AC800 series.

7.6.2 Overview

A diagnosis programming interface allows you to obtain diagnosis information from user programs. You can check diagnosis information for modules in user programs and take action based on the information.

A diagnosis programming interface exists in library form. You can add it on the "Library Manager" page, as shown in the following figure.



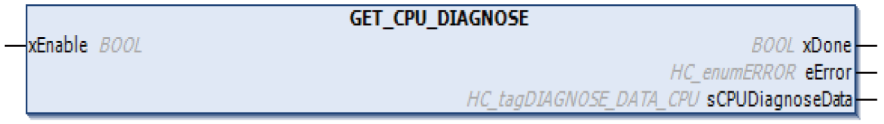
Programming interfaces include diagnosis programming interfaces corresponding to CANlink, CANopen, CPU, Modbus, Modbus TCP, and PROFIBUS-DP modules, respectively. Each diagnosis corresponds to one function block so that you can obtain corresponding diagnosis codes.

User-defined diagnosis results and diagnosis states are defined in "DataType". "HC_enumERROR" indicates whether diagnosis succeeds, as shown in the following table.

Enumerator	Value (Decimal)	Description
NO_ERROR	0	No error
WRONG_PARAMETER	1	Parameter error
UNKNOWN_DEVICEID	2	Unknown device ID
INVAILD_DEVICEID	3	Invalid device ID
INVAILD_IO_POS	4	Invalid I/O position
UNSUPPORT_DIAGNOSE	5	Unsupported diagnosis
TIME_OUT	6	Timeout
INTERNAL_FB_ERROR	7	Internal function block error
UNKNOWN_ERROR	8	Unknown error
INVAILD_IP	9	Invalid IP address

7.6.3 CPU Diagnosis Programming Interface

CPU diagnosis programming interface

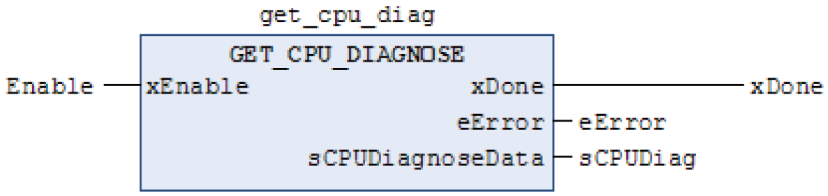
Obtaining CPU Diagnosis Data: GET_CPU_DIAGNOSE			
			
Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
Output parameter			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
sCPUDiagnoseData	HC_tagDIAGNOSE_DATA_CPU		Indicates CPU diagnosis data.

The "HC_tagDIAGNOSE_DATA_CPU" data is structure data, as shown in the following table. For details about the correlation between the diagnosis code and diagnosis information, see *"9.8.2 CPU Diagnosis Code" on page 509*.

Name	Type
SDCardError	BYTE
FlashError	BYTE
SystemError	BYTE
InterCommError	BYTE
ConformenceError	WORD
IOModulePosError	WORD
FunctionErrorCode	WORD

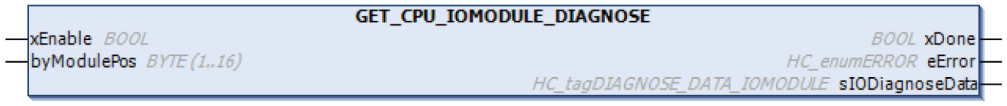
Example

```
PROGRAM POU
VAR
    get_cpu_diag: GET_CPU_DIAGNOSE;
   Enable: BOOL;
    eError: HC_enumERROR;
    xDone: BOOL;
    sCPUDiag: HC_tagDIAGNOSE_DATA_CPU;
END_VAR
```



CPU local I/O expansion module

Obtaining CPU I/O Diagnosis Data: GET_CPU_IOMODULE_DIAGNOSE



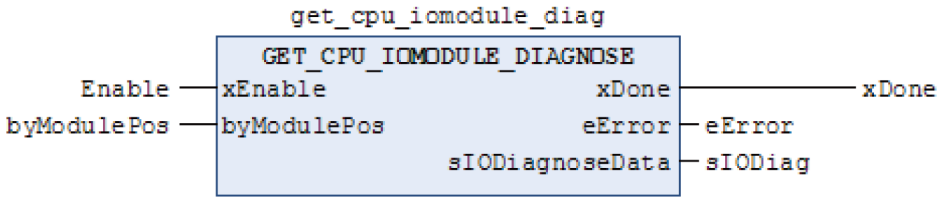
Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
byModulePos	BYTE (1..16)	0	Indicates the obtained I/O position.
Output parameter			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
sIODiagnoseData	HC_tagDIAGNOSE_DATA_IOMODULE		Indicates I/O diagnosis data.

The "HC_tagDIAGNOSE_DATA_IOMODULE" data is structure data, as shown in the following table. For details about the correlation between the diagnosis code and diagnosis information, see *"9.8.3 I/O Module Diagnosis Code" on page 512*.

Structure Member	Type	Description
ModuleError	BYTE	Indicates the module error.
ChannelError	ARRAY[0..3] OF BYTE	Indicates the channel error.

Example

```
PROGRAM POU
VAR
    get_cpu_iomodule_diag: GET_CPU_IOMODULE_DIAGNOSE;
   Enable: BOOL;
    eError: HC_enumERROR;
    xDone: BOOL;
    byModulePos: BYTE (1..16);
    sIODiag: HC_tagDIAGNOSE_DATA_IOMODULE;
END_VAR
```



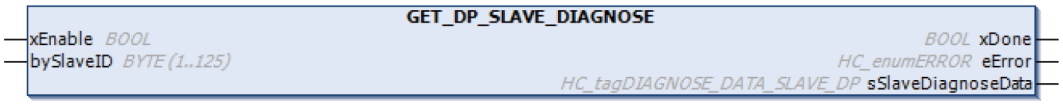
7.6.4 CANopen Diagnosis Programming Interface

To diagnose the communication state of a CANopen slave, add the "CmpHCCiA405" library to the library manager and call the "GET_STATE" interface to get the state.

7.6.5 PROFIBUS-DP Diagnosis Programming Interface

PROFIBUS-DP slave diagnosis programming interface

Obtaining DP Slave Diagnosis Data: GET_DP_SLAVE_DIAGNOSE



Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
bySlaveID	BYTE (1..125)	0	Indicates the obtained slave address, ranging from 1 to 125.
Output parameter			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
sSlaveDiagnoseData	HC_tagDIAGNOSE_DATA_SLAVE_DP		Indicates DP slave diagnosis data.

The "HC_tagDIAGNOSE_DATA_SLAVE_DP" data is structure data, as shown in the following table. For details about the correlation between the diagnosis code and diagnosis information, see *"9.8.4 PROFIBUS-DP Diagnosis Code" on page 513*.

Structure Member	Type	Description
Length	BYTE	Indicates the diagnosis data length.
ExtDiagData	ARRAY[0..243]OF BYTE	Indicates diagnosis data.

Example

PROGRAM POU

VAR

get_dp_slave_diag: GET_DP_SLAVE_DIAGNOSE;

Enable: BOOL;

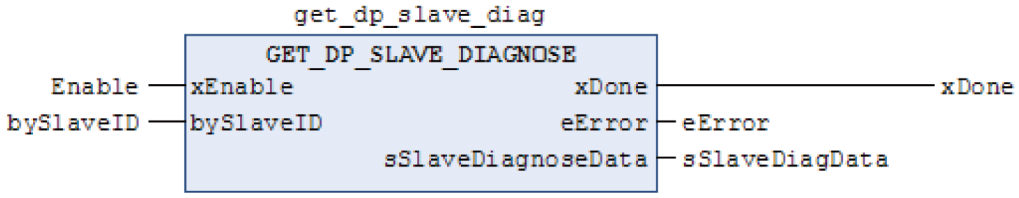
eError: HC_enumERROR;

xDone: BOOL;

bySlaveID: BYTE (1..125);

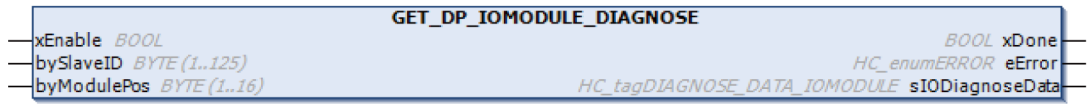
sSlaveDiagData: HC_tagDIAGNOSE_DATA_SLAVE_DP;

END_VAR



PROFIBUS-DP slave I/O diagnosis programming interface

Obtaining DP Slave I/O Diagnosis Data: GET_DP_IOMODULE_DIAGNOSE



Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
bySlaveID	BYTE (1..125)	0	Indicates the slave address, ranging from 1 to 125.
byModulePos	BYTE (1..16)	0	Indicates the diagnosed I/O position.
Output parameter			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
sIODiagnoseData	HC_tagDIAGNOSE_DATA_IOMODULE		Indicates I/O diagnosis data.

The "HC_tagDIAGNOSE_DATA_IOMODULE" data is structure data, as shown in the following table. For details about the correlation between the diagnosis code and diagnosis information, see *"9.8.3 I/O Module Diagnosis Code" on page 512*.

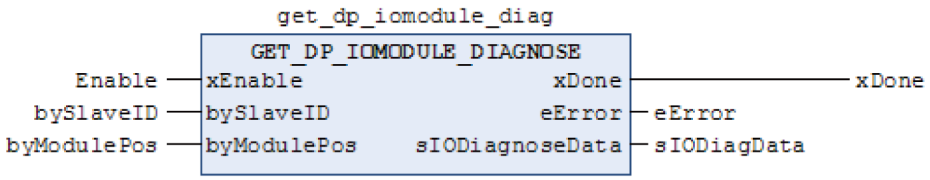
ModuleError	BYTE
ChannelError	ARRAY [0..3] OF BYTE

Example

PROGRAM POU

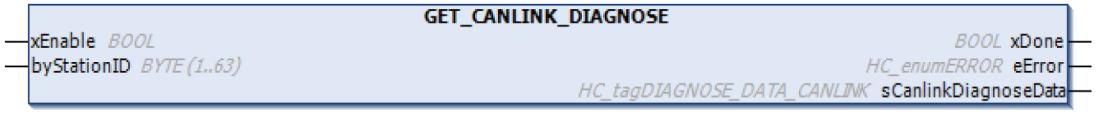
VAR

```
    get_dp_iomodule_diag: GET_DP_IOMODULE_DIAGNOSE;
   Enable: BOOL;
    eError: HC_enumERROR;
    xDone: BOOL;
    bySlaveID: BYTE (1..125);
    byModulePos: BYTE (1..16);
    sIODiagData: HC_tagDIAGNOSE_DATA_IOMODULE;
END_VAR
```



7.6.6 CANlink Diagnosis Programming Interface

Obtaining CANlink Diagnosis Data: GET_CANLINK_DIAGNOSE



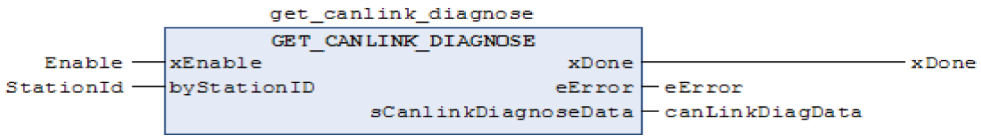
Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
byStationID	BYTE (1..63)	0	Indicates the obtained station node ID, ranging from 1 to 63.
Output parameter			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
sCanlinkDiagnoseData	HC_tagDIAGNOSE_DATA_CANLINK		Indicates CANlink station diagnosis data.

The "HC_tagDIAGNOSE_DATA_CANLINK" data is structure data, as shown in the following table. For details about the correlation between the diagnosis code and diagnosis information, see *"9.8.5 CANlink Diagnosis Code" on page 513*.

Structure Member	Type	Description
IsUsed	BOOL	Indicates whether it is used.
IsMaster	BOOL	Indicates whether it is the master.
StationStatus	WORD	Indicates the CANlink station state.
CfgFrameError	WORD	Indicates the configuration frame error.
CmdFrameError	WORD	Indicates the command frame error.

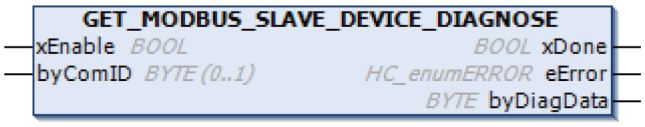
Example

```
PROGRAM POU
VAR
    get_canlink_diagnose:GET_CANLINK_DIAGNOSE;
 Enable: BOOL;
    StationId: BYTE (1..63);
    eError: HC_enumERROR;
    canLinkDiagData: HC_tagDIAGNOSE_DATA_CANLINK;
    xDone: BOOL;
END_VAR
```



7.6.7 Modbus Diagnosis Programming Interface

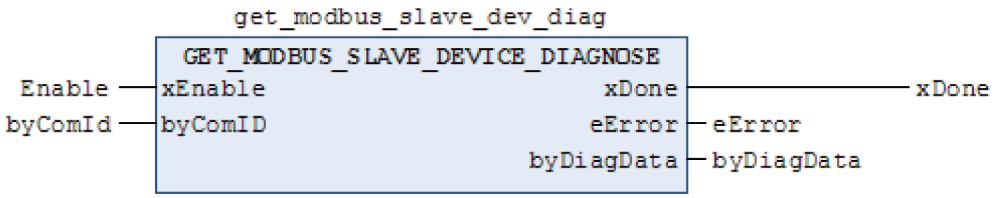
Modbus local slave diagnosis programming interface

Obtaining Modbus Local Slave Diagnosis Data: GET_MODBUS_SLAVE_DEVICE_DIAGNOSE			
			
Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
byComID	Byte (0..1)	0	Indicates the serial port number for the local slave, ranging from 0 to 1.
Output parameter			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.

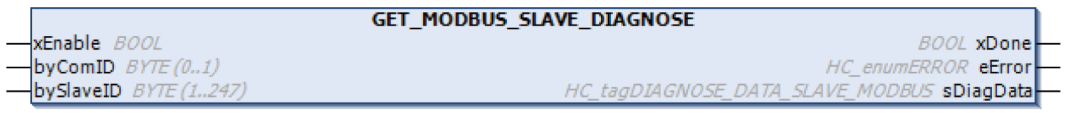
Obtaining Modbus Local Slave Diagnosis Data: GET_MODBUS_SLAVE_DEVICE_DIAGNOSE			
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
byDiagData	Byte		Indicates the diagnosis code. For details about the correlation between the diagnosis code and diagnosis information, see "Modbus Diagnosis Code".

Example

```
PROGRAM POU
VAR
    get_Modbus_slave_dev_diag: GET_Modbus_SLAVE_DEVICE_DIAGNOSE;
   Enable: BOOL;
    eError: HC_enumERROR;
    xDone: BOOL;
    byComId: Byte (0..1);
    byDiagData: Byte;
END_VAR
```



Modbus remote slave diagnosis programming interface

Obtaining Modbus Remote Slave Diagnosis Data: GET_MODBUS_SLAVE_DIAGNOSE			
			
Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
byComID	Byte (0..1)	0	Indicates the serial port number for the master, ranging from 0 to 1.
bySlaveID	Byte (1..247)	0	Indicates the slave address, ranging from 1 to 247.
Output parameter			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.

Obtaining Modbus Remote Slave Diagnosis Data: GET_MODBUS_SLAVE_DIAGNOSE			
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
sDiagData	HC_tagDIAGNOSE_DATA_SLAVE_ Modbus		Indicates slave diagnosis data.

The "HC_tagDIAGNOSE_DATA_SLAVE_Modbus" data is structure data, as shown in the following table. For details about the correlation between the diagnosis code and diagnosis information, see *"9.8.6 Modbus Diagnosis Code" on page 514*.

Name	Type
ChannelNum	Byte
DiagData	Byte

Example

PROGRAM POU

VAR

get_Modbus_slave_diag: GET_Modbus_SLAVE_DIAGNOSE;

Enable: BOOL;

eError: HC_enumERROR;

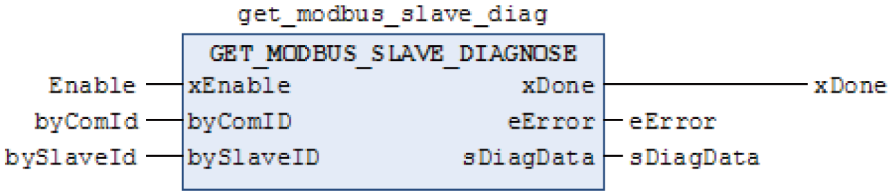
xDone: BOOL;

byComId: Byte (0..1);

bySlaveId: Byte (1..247);

sDiagData: HC_tagDIAGNOSE_DATA_SLAVE_Modbus;

END_VAR



7.6.8 Modbus TCP Diagnosis Programming Interface

Modbus TCP local slave diagnosis programming interface

Obtaining Modbus TCP Local Slave Diagnosis Data: GET_MODBUSTCP_SLAVE_DEVICE_DIAGNOSE			
			
Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
Output parameter			

Obtaining Modbus TCP Local Slave Diagnosis Data: GET_MODBUSTCP_SLAVE_DEVICE_DIAGNOSE			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
byDiagData	Byte		Indicates the diagnosis code. For details about the correlation between the diagnosis code and diagnosis information, see "Modbus Diagnosis Code".

Example

PROGRAM POU

VAR

get_Modbus TCP_slave_dev_diag: GET_Modbus TCP_SLAVE_DEVICE_DIAGNOSE;

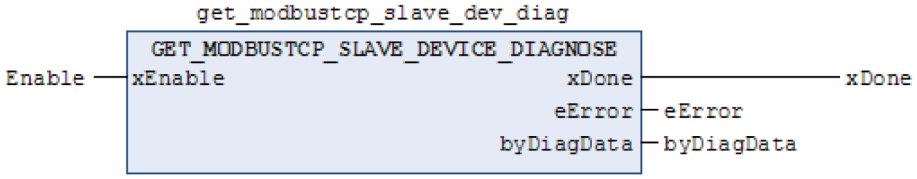
Enable: BOOL;

eError: HC_enumERROR;

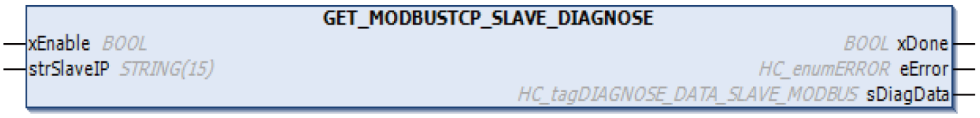
xDone: BOOL;

byDiagData: BYTE;

END_VAR



Modbus TCP remote slave diagnosis programming interface

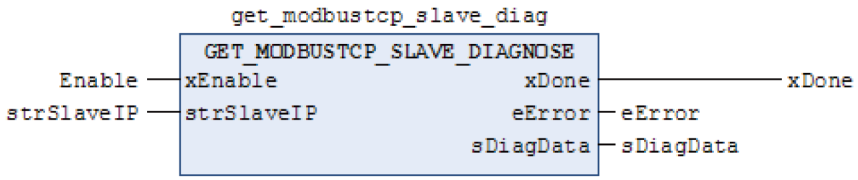
Obtaining Modbus TCP Remote Slave Diagnosis Data: GET_MODBUSTCP_SLAVE_DIAGNOSE			
			
Parameter	Type	Initial Value	Function
Input parameter			
xEnable	BOOL	FALSE	Indicates the enabling bit, triggered by level.
strSlaveIP	STRING(15)	''	Indicates the remote slave IP address.
Output parameter			
xDone	BOOL	FALSE	Indicates whether the diagnosis result is obtained.
eError	HC_enumERROR	NO_ERROR	Indicates whether the diagnosis result is obtained.
sDiagData	HC_tagDIAGNOSE_DATA_SLAVE_MODBUS		Indicates slave diagnosis data.

The "HC_tagDIAGNOSE_DATA_SLAVE_MODBUS" data is structure data, as shown in the following table. For details about the correlation between the diagnosis code and diagnosis information, see *"9.8.6 Modbus Diagnosis Code" on page 514*.

Name	Type
ChannelNum	Byte
DiagData	Byte

Example

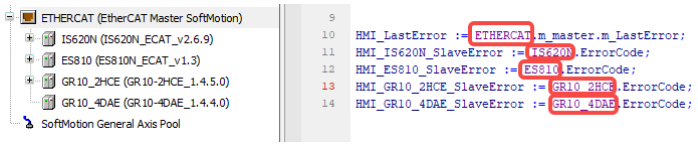
```
PROGRAM POU
VAR
    get_Modbus TCP_slave_diag: GET_Modbus TCP_SLAVE_DIAGNOSE;
   Enable: BOOL;
    eError: HC_enumERROR;
    xDone: BOOL;
    sDiagData: HC_tagDIAGNOSE_DATA_SLAVE_MODBUS;
    strSlaveIP: STRING(15);
END_VAR
```



7.6.9 EtherCAT Diagnosis Programming Interface

EtherCAT diagnosis is used to record and describe bus errors, including master diagnosis, slave diagnosis, slave module diagnosis, and slave servo drive diagnosis. EtherCAT diagnosis only parses errors of Inovance slaves. For details about diagnosis methods, see *"7.3 Fault Diagnosis" on page 440*. Error IDs are listed in appendices to this guide.

In some application scenarios, error IDs are displayed on the touchscreen. You only need to assign the variable "m_LastError" (the EtherCAT bus error ID) and the EtherCAT slave error ID to a variable associated with the HMI address. "HMI_LastError" and "HMI_IS620N_SlaveError" are word variables associated with the HMI address. The bus error ID and slave error ID diagnosed for EtherCAT are displayed on the touchscreen.



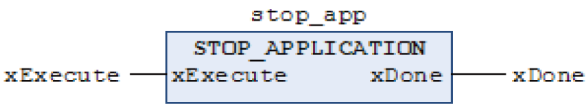
7.6.10 CPU Stop Control

Description of function blocks

Stopping Application Program: STOP_APPLICATION			
			
Parameter	Type	Initial Value	Function
Input parameter			
xExecute	BOOL	FALSE	Indicates the enabling bit of the function block, triggered by rising edges.
Output parameter			
xDone	BOOL	FALSE	Indicates execution complete output.

Function block example

```
PROGRAM POU
VAR
    stop_app: STOP_APPLICATION;
    xExecute: BOOL;
END_VAR
```



7.6.11 Axis Diagnosis

Error code

The axis diagnosis function is used to record and describe faults occurred during axis initialization, startup, and running, such as slave communication errors and axis internal errors during axis running (including soft limits, servo alarms, and improper function block uses). For error IDs of axis diagnosis, see appendices to this guide.

Axis faults are classified by severity into errors, warnings, and information.

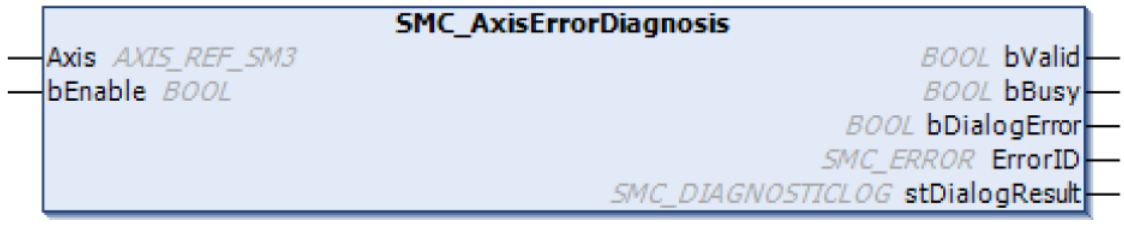
- **Error (serious)**: Including running errors with axis stopped, axis moving, interruption during running, bus errors, servo faults, function block running errors, and SDO communication errors.
- **Warning (general)**: Including running errors without axis stopped, errors caused by input of motion instructions such as "standstill", "errorstop", and "poweroff", and other internal errors that do not affect the action function or the current state.
- **Information (record)**: Including startup records and records about other key processes.

Diagnosis page

The following table describes the diagnosis result structure "SMC_DIALOGRESULT" (quick diagnosis).

Variable Member	Data Type	Initial Value	Variable Description
DeviceType	UINT	-	SoftMotion
uiModleID	UINT	-	Axis ID
tTimeStamp	TIME	0	Running time stamp
ErrorID	SMC_ERROR	-	Error ID
eErrorClass	SMC_ERRORCLASS	-	Fault level (serious, medium, and minor)
eErrorType	SMC_ERRORTYPE	-	Error type
StErrorInstance	STRING(60)	-	Error source
strErrorNotes	STRING(60)	-	Error description
strErrorShooting	STRING(255)	-	Solution

The following figure shows the diagnosis function block.



- Function
  The axis real-time diagnosis information is saved in the diagnosis result structure "SMC_DIALOGRESULT".
- Input and output parameters
  The following table lists input and output variables.

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Mapped to the axis, that is, an instance of AXIS_REF_SM3.

The following table lists input variables.

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Execution condition	BOOL	TRUE, FALSE	FALSE	The diagnosis function of the function block is enabled when a high level signal is input.

The following table lists output variables.

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Compliant	BOOL	TRUE, FALSE	FALSE	"TRUE" is output when the instruction is executed.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	It is set to "TRUE" when the current instruction is being executed.
Error	Error	BOOL	TRUE, FALSE	FALSE	It is set to "TRUE" when an error occurs.
ErrorID	Error code	DWORD	-	0	The error code is output when an error occurs.
stDialogResult	Diagnosis result	SMC_ DIALOGRESULT	-	-	Diagnosis result structure

8 FAQ

8.1 CPU Utilization Too High

8.1.1 CPU Usage Definition

0% to 89%: The PLC runs stably. Logic execution, bus synchronization, I/O refresh, data synchronization, and data saving are all guaranteed with time.

90% to 100%: The PLC runs less stably. Main impacts:

- It is difficult to ensure the EtherCAT operation stability, and EtherCAT slave disconnection and synchronization loss may occur.
- The PLC may enter the false state and you cannot scan or log in to the PLC.
- Data cannot be stored at power-off.
- The CANopen, CANlink, Modbus, and Modbus TCP modules are subjected to data refresh and disconnection.
- Online modification or download of a PLC program may slow down and possibly fail.
- There is a risk of slow or failed refresh of monitored PLC variable values.

Note

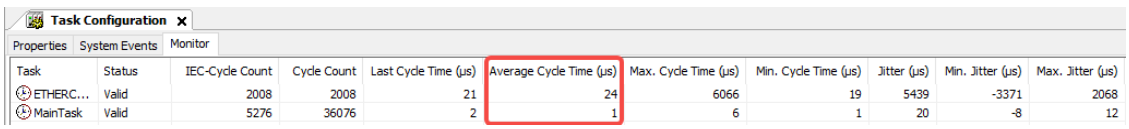
CPU usage is defined for PLCs of the AM600 and AM400 series but not the AC800 and AP700 series.

8.1.2 Analysis Procedure

1. Check the CPU usage of the PLC.
   Log in to the PLC and check the CPU usage based on the background status bar, as shown in the figure below.



2. Check the task execution time and calculate the proportion of the execution time in the task.
   Log in to the PLC, access the "Task Configuration" page, click the "Monitor" tab, and check the average cycle time of each task.



Note

If the page has been displayed before login, when you re-access the page after login, you need to right-click the task and choose "Reset" to restore the initial calculation status.

As shown in the figure above, the cycle periods for both the EtherCAT and MainTask tasks are 4 ms, and the proportion of the MainTask task is about 89% (3575/4000). This indicates that the MainTask task occupies excessive execution logic.

3. Optimize programs within tasks.
   Locate a program with too long execution time, and then find the code segments that take a long time to execute in the program.

   Identify a program that consumes too much CPU time, usually by assessing the impact of removing programs under tasks. If the task execution time significantly decreases after a program is removed from a task, it suggests that the program may need to be optimized.

   After identifying the program, locate the code segment that takes a long time to execute by assessing the impact of removing code from the program.

8.1.3 Common Optimization Methods

- Increase the task scan cycle.
  After the task scan cycle is increased, the program execution count within the task decreases, leading to less CPU usage.
- Optimize code for batch data processing.
  Programs often involve loops for processing data in batches. Consider processing batches of data over multiple cycles, especially for tasks like initialization code or logic with low real-time requirements.
- Introduce additional IF conditions.
  Without additional conditions, program blocks and functions will be executed continuously in each cycle. Introduce IF conditions to determine when a block or function should be executed based on specific criteria. Consider to introduce IF conditions in ST and convert operation blocks to EN/ENO types in LD.
- Upgrade the PLC to a higher-performance PLC.

8.2 Abnormal PLC Running

8.2.1 Overview

Inovance medium-sized PLCs of the AM and AC series are developed using a compiled language designed based on the international standard IEC 61131-3.

A compiled language, unlike an interpreted language (commonly used in small PLCs), requires a specific compilation process to compile the program into machine language files before program execution. During runtime, the compilation results are used directly without the need of reinterpretation. Programs of compiled languages are flexible in coding and efficient in execution, but demand relatively higher programming skills from developers (having a background in C/C++ is beneficial).

When writing programs, users need to be cautious about unauthorized pointer access, division by zero, array out-of-bounds, implicit data type conversion, infinite loops, global variable protection, and other avoidance measures. Failure to address these faults may lead to PLC operation failures and even system breakdown.

This document focuses on the main causes, locating steps, and solutions for operation failures of user programs on PLCs, such as download failure and system breakdown.

8.2.2 Symptoms

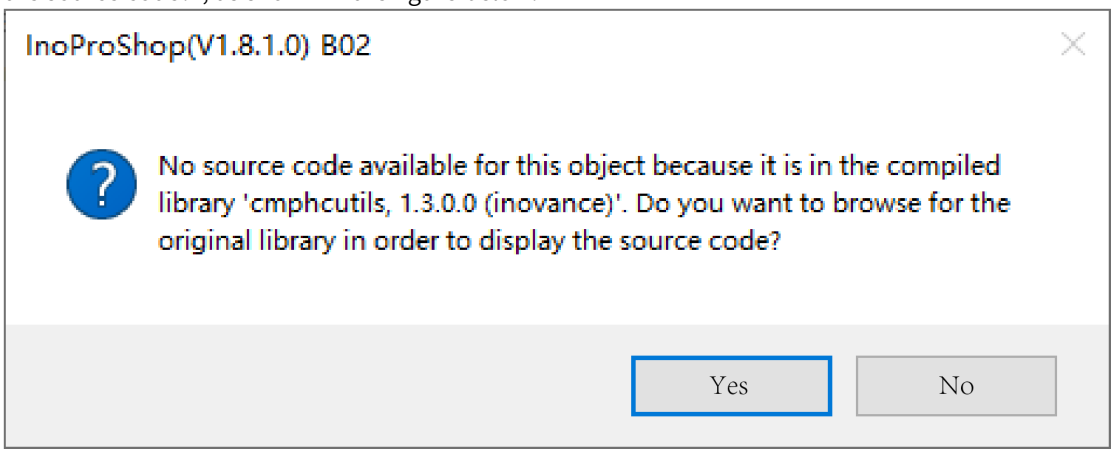
Note

- After identifying the fault, it is recommended to manually remove all implicit check functions, as these functions consume a portion of CPU resources.
- The firmware version for the AM series PLC must be V1.22.0.0 or above. The InoProShop version for the programming software must be V1.3.2 or above.

- The AM series LED display stops refreshing (normally showing "00"), and the AC series LCD display shows the message "Runtime crash".
- The programming software cannot scan the corresponding PLC device. After the PLC is powered on again and operates normally for a while, the PLC cannot be scanned again.
- After program downloading or login to the PLC that operates for a period, the information display bar shows the program as "Stopped" with an error message "Program loaded - EXCEPTION," as shown in the figure below.



- After program downloading or login to the PLC that operates for a period, a prompt box pops up, displaying "No source code available for this object because it is in the compiled library 'cmphcutils, 1.3.0.0 (inovance)'. Do you want to browse for the original library in order to display the source code?", as shown in the figure below.



8.2.3 Cause Analysis and Solutions

Unauthorized Pointer Access

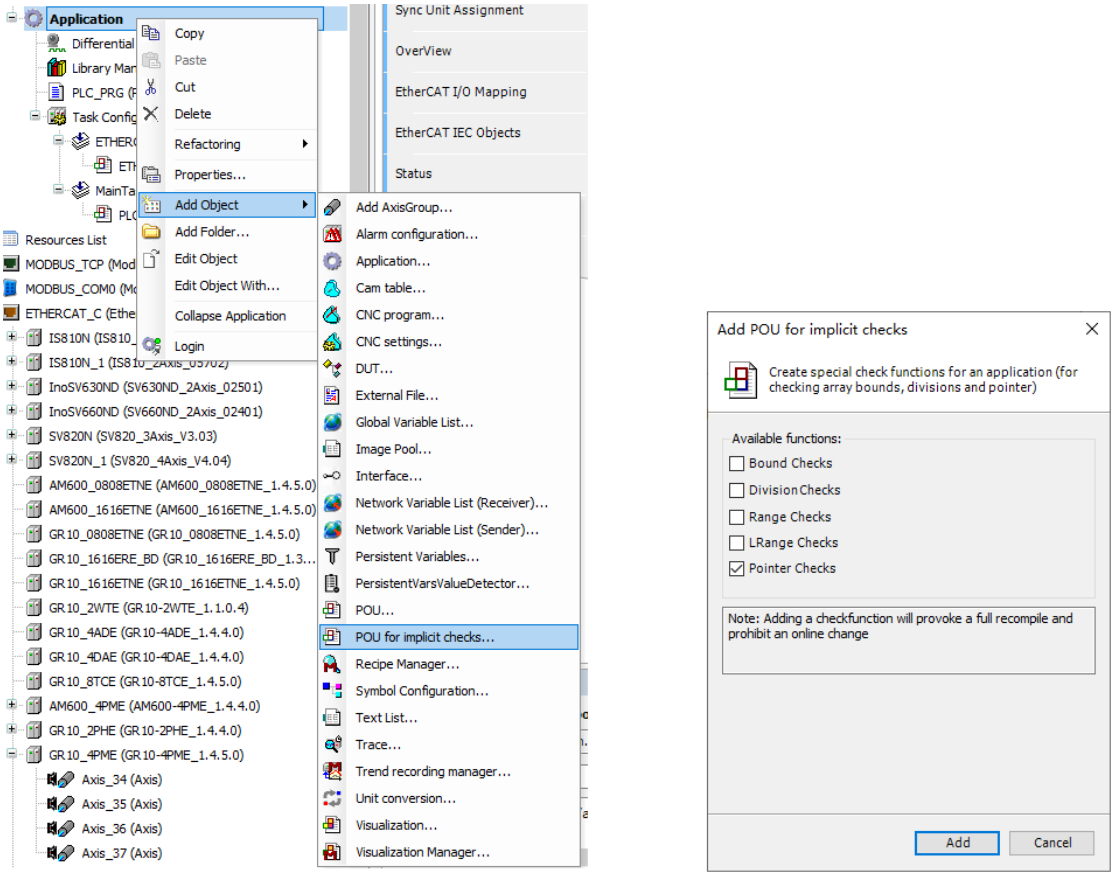
Unauthorized pointer access includes null pointer access (the address value pointed to by the pointer is 0x00000000) and pointer access to unauthorized areas (the address pointed to by the pointer conflicts with the internal address of the operating system).

The PLC operating system cannot execute a null pointer (the address 0x00000000 is a startup address for some microcontroller systems, leading to a soft restart of the microcontroller). Null pointers are relatively easy to locate.

Pointer access to unauthorized areas may conflict with other programs running in the system, causing execution failure. It is more challenging to identify the reasons for such pointers. It is recommended for novice users to use arrays rather than pointers.

- Locating steps

  In the user program device tree, right-click "Application" and choose "Add Object" > "POU for implicit checks". In the "Add POU for implicit checks" dialog box, select "Pointer Checks", as shown in the figure below.



  After "Pointer Checks" is added, the programming software automatically adds the "CheckPointer" function under the "Application" device tree, and you can manually add debug code to the function.

  Each time the user program calls a pointer, the system automatically executes the implicit check function "CheckPointer" once. By adding program breakpoints inside the function, you can pinpoint the specific location where a null pointer is being used in the user program (refer to the appendix for breakpoint debugging methods).

  Log in to the PLC, right-click "CheckPointer := ptToTest;" where code is added to add and activate a breakpoint, and manually start to run the PLC (during debugging, switch the PLC run switch to "STOP" for the AM series, and set "Program Startup Running" on the small screen for the AC series). Currently, this method can only pinpoint the situation where the pointer is 0.

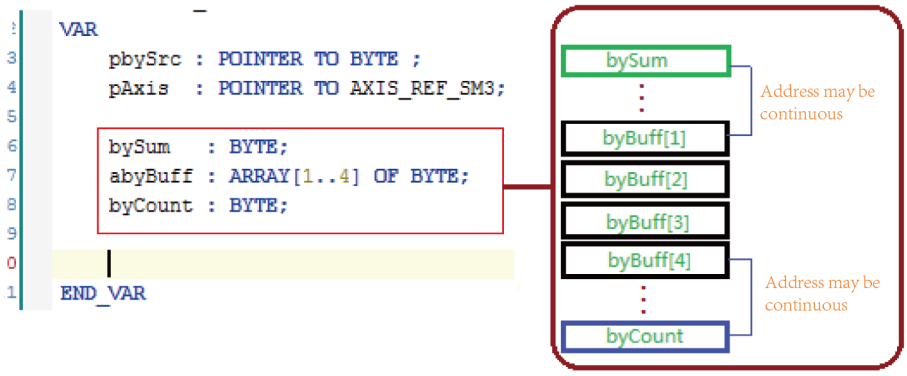
- Solution

Add a non-zero check condition for pointer calls in the user program.

Array Out-of-bounds

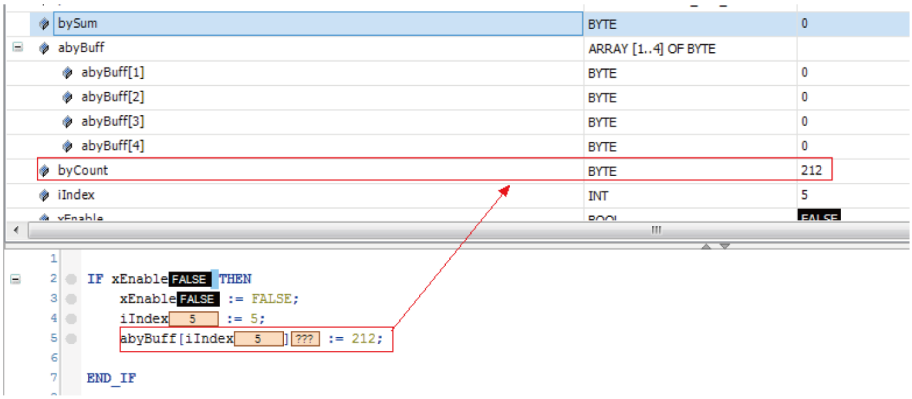
Array out-of-bounds can be categorized as out-of-upper bounds and out-of-lower bounds. Calling an out-of-bounds array in the program may result in the values of adjacent variables being overwritten by the out-of-bounds array values.

In the program, the memory layout of POU variables and arrays in the system is roughly indicated by the relationship between "bySum", "abyBuff", and "byCount", as shown in the figure below.



In the variable definition area, "byBuff[1]" is adjacent to the variable "bySum", and "byBuff[4]" is adjacent to "byCount". If the user program writes data to "byBuff[0]" and "byBuff[5]", the values of "bySum" and "byCount" may be overwritten by the values of "byBuff[0]" and "byBuff[5]", sharing the same memory address (during compilation, the system has already allocated memory addresses for all variables). Due to the optimization algorithm for memory address allocation, the addresses of variables may be contiguous or non-contiguous. Generally, it is more likely that the upper bound of an array is contiguous with memory addresses of variables.

For example, "iIndex" is used as an array variable. When "iIndex" is equal to 5 and "byBuff[iIndex]" is set to 212, "byCount" changes to 212 because the address pointed to by "byCount" is contiguous with the address of "byBuff[5]".



However, when "iIndex" is equal to 0 and "byBuff[iIndex]" is set to 212, "bySum" changes to 0 because the address pointed to by "bySum" is not contiguous with the address of "byBuff[0]".

- Locating steps
  The steps are similar to those of locating the pointer failure. In the user program device tree, right-click "Application" and choose "Add Object" > "POU for implicit checks". In the "Add POU for

implicit checks" dialog box, select "Bound Checks". Then the system automatically adds the "CheckBounds" function under the "Application" device tree.

Log in to the PLC, and add and activate breakpoints at the code lines "CheckBounds := lower;" and "CheckBounds := upper;". Perform single-step debugging (F10) to locate the failure in the user program and check whether the current array variable is within the defined range. For example, "abyBuff[5]" is not within the defined range of "byBuff[1..4]".

- Solution
  There is no direct solution. The fault must be addressed in code.

Division by Zero

Causes of division by zero include uninitialized variables, variable initialization after being called, and global variables being set in multiple tasks or POUs.

- Locating steps
  In the user program device tree, right-click "Application" and choose "Add Object" > "POU for implicit checks". In the "Add POU for implicit checks" dialog box, select "Division Checks".

  The system automatically adds the "CheckDivDInt", "CheckDivLInt," "CheckDivLReal", and "CheckDivReal" functions under the "Application" device tree.

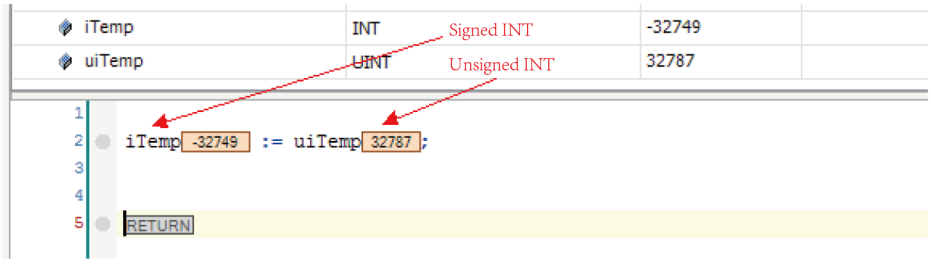
  Log in to the PLC, add and activate breakpoints at the code lines "CheckDivDInt:=1;", "CheckDivLInt:=1;", "CheckDivLReal:=1;", and "CheckDivReal:=1;" for the four functions. Perform single-step debugging (F10) to locate the failure in the user program (refer to the previous debugging steps of pointer failure for specific operations).

- Solution
  Add a condition check of "division by zero" in the code where the operation takes place. For 32-bit variables, a threshold value of 10E-6 (0.000001) is commonly used. For 64-bit variables, the precision is adjusted as needed, with a minimum of 10E-15, as shown in the figure below.

```
10  IF ABS(fDiv) >= 0.000001 THEN
11      fRet := (fSum / fDiv);
12  END_IF
```

Implicit Data Type Conversion

Assignments between signed and unsigned variables with the same data width may lead to unexpected values if forced, potentially causing program execution failure when variables are referenced in other program segments, as shown in the figure below.



- Locating steps
  None

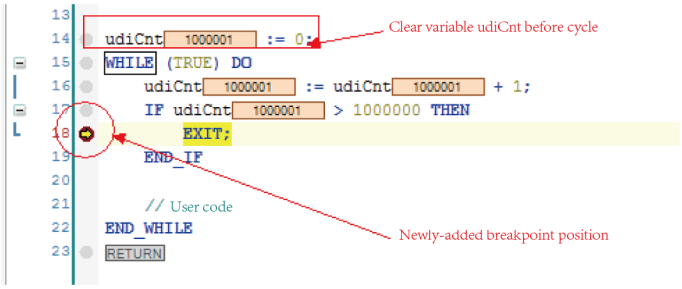
- Solution
  During the compilation process, the user program will generate warning information. Pay attention to the specific content of the compilation warning information and ensure that the data types on both sides of assignment operation are the same.

Infinite Loop

If the for, while, and repeat loop conditions are improperly used, the system will continuously execute the code segments within the loop body, causing the failure to execute code outside the loop body, for example, failure in LED display refresh and communication between the PLC and programming software. In the case of EtherCAT tasks, an infinite loop program could disrupt the data retention upon power failure, leading to the loss of retained data.

- Locating steps
  Introduce a count variable in the for, while, or repeat loop. When the count variable reaches a certain value, the loop should exit. Log in to the PLC and add a breakpoint at the loop exit position "EXIT". If the loop count value exceeds the expected value, the program will run to the breakpoint. For details about how to use breakpoints, see the user guide.

  As shown in the figure below, the variable "udiCnt" specifies the loop execution count. After the code within the while loop body is executed 100001 times, the program runs to the "EXIT" point.



- Solution
  Follow the locating steps provided above, or add a timer within the loop body. When the timer reaches a specific time limit, the program should exit the loop.

Function Block Instance Calling in Multiple Tasks

Internal variables of a function block instance retain their values of the first execution after the second execution within a scan cycle (similar to static variables in C/C++). When tasks A and B simultaneously call the same function block instance (including methods, actions, attributes, and transitions), a situation may arise where the function block instance of task A is interrupted by task B that has a higher priority. If task A resumes after task B completes, values of internal variables within the function block instance may not match the values before the interruption, potentially affecting the normal execution of the function blocks in tasks A and B.

- Locating steps
  None

- Solution
  There is no direct solution. The fault must be prevented during the initial code design phase.

Global Variable Calling in Multiple Tasks

When both tasks A and B writes data to the same variable, a situation may arise where task A is writing data to the global variable but task B that has a higher priority preempts CPU resources and writes values of global variables. If task A resumes after task B completes, the values of global variables may not match the values before the preemption, potentially affecting the normal execution of the function blocks.

Example: "wTemp" is a global variable consisting of 2 bytes, with "wTempH" and "wTempL" as the higher byte and lower byte of "wTemp", respectively. When task A completes writing data to "wTempL" but has not written data to "wTempH", task B with higher priority preempts CPU resources. After task B writes data to both "wTempL" and "wTempH", it releases the CPU resources and task A continues to write data to "wTempH". Since "wTempL" has been modified by task B, the value of "wTemp" in task A may not be within the expected range.

- Locating steps
  None
- Solution
  Only one task can write data to global variables. This fault must be addressed during the code design phase. (Inovance medium-sized PLCs can potentially address this type of faults using mutex semaphores. However, this is only suitable for advanced developers with deep understanding of operating systems. Ordinary users should not attempt this without proper knowledge, as misuse can lead to PLC breakdown.)

Others

For user program breakpoints, single-step, and other debugging methods, see the user manual. The relevant content can be found in the user manual.

8.3 Failure to Obtain Folders

Symptoms

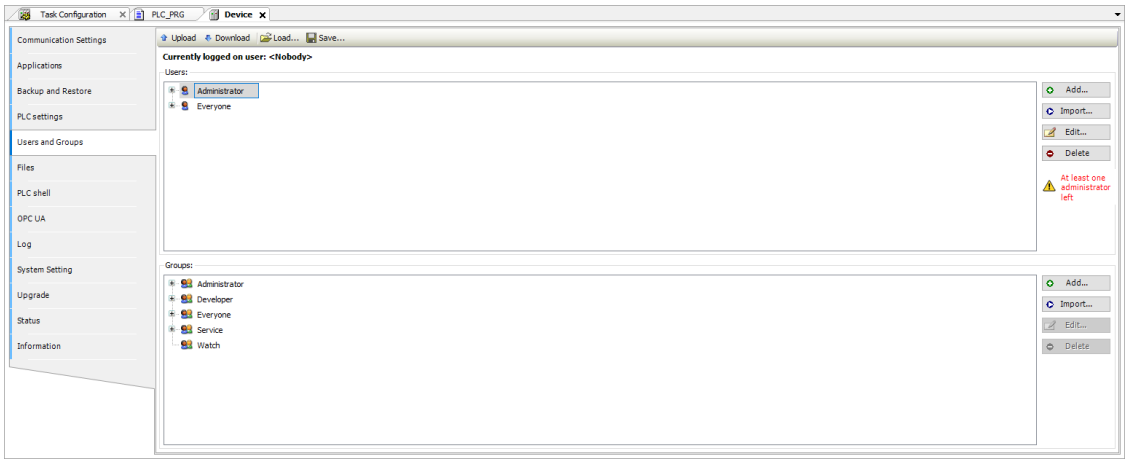
When you attempt to check folders on the "Device" page of the InoProShop software, the "Get directory-entries failed!" message pops up.

Cause Analysis

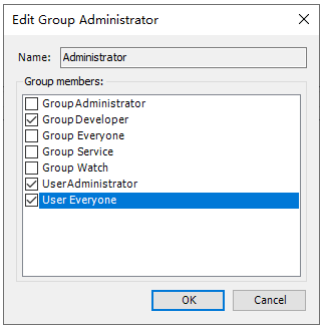
A user logs in to the PLC as the default user "Everyone". Before configuration on the "Users and Groups" page, the Everyone user has the administrator privileges to access folders. After configuration on the page, the Everyone user no longer has the privileges.

Solution

1. On the "Device" page, click "Users and Groups".



2. In the "Groups" area, select "Administrator" and click "Edit". The "Edit Group Administrator" dialog box is displayed.



3. Select "User Everyone" and click "OK" to add the "Everyone" user to the "Administrator" group.
4. Click "Download" for the configurations to take effect.

9 Appendix

9.1 Communication Protocols for Communication Ports

9.1.1 Overview

The medium-sized PLCs are provided with Mini-USB ports, serial communication ports, Ethernet ports, EtherCAT ports, Mini-SD card slots, CAN communication ports, PROFIBUS DP communication ports, high-speed I/O interfaces, and local bus expansion interfaces.

9.1.2 Mini-USB Port and Built-in Communication Protocol

The Mini-USB port is used to download PLC user programs and monitor and debug the system. Therefore, the port has a specific communication protocol, and you do not need to select one. As long as a USB drive is installed on the PC, you can use InoProShop to download or monitor user programs of the medium-sized PLCs on the PC at any time.

As the built-in download protocol for the Mini-USB port is a proprietary protocol of Inovance, you cannot download programs of medium-sized PLCs through third-party programming software.

After the InoProShop programming software is installed for the first time, the USB drive is automatically installed. To install different versions of InoProShop on one PC, you need to install them in different directories.

9.1.3 COM Communication Ports and Built-in Protocols

COM0 and COM1 ports are basic ports of a PLC for external communication. They are integrated on one DB9 physical port, mainly used for RS485 or Modbus communication.

The following table lists protocols supported by COM0 and COM1 ports as well as definitions of set units.

<table>
<tr><th>COM0/COM1 Protocol</th><th>Half/Full Duplex Mode</th><th>Communication Format</th><th>Baud Rate</th><th>Data Bit</th><th>Stop Bit</th><th>Parity Bit</th></tr>
<tr><td>Modbus-RTU master</td><td>Half duplex</td><td>Fixed</td><td rowspan="3">4800 bps
9600 bps
19200 bps
38400 bps
57600 bps
115200 bps</td><td>7-bit
8-bit</td><td rowspan="3">1-bit
2-bit</td><td rowspan="3">None
Odd
Even</td></tr>
<tr><td>Modbus-RTU slave</td><td>Half duplex</td><td>Fixed</td><td>8-bit</td></tr>
<tr><td>RS485 free protocol</td><td>Half duplex</td><td>Unfixed</td><td>8-bit</td></tr>
<tr><td>Modbus-ASCII master</td><td>Not supported</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Modbus-ASCII slave</td><td>Not supported</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table>

The following describes these COM communication protocols.

- Modbus master protocol

As a control host, the PLC usually uses this protocol to communicate with AC drives, servos, and other slave computers, or to read data from smart meters and sensors. PLCs communicate with each other through the Modbus protocol, improving the flexibility of communication.

- Modbus slave protocol
  A host computer uses the Modbus slave protocol to read internal data from PLCs that serve as slaves. When you set the port as a Modbus slave on the PLC, the PLC automatically processes and responds based on the communication instruction from the host computer.
- Free communication protocol
  Protocols other than PLC built-in communication protocols are called "free communication protocols". To enable communication through a free protocol, programmers must fully understand the frame structure definition of the protocol. Programmers prepare data strings to be sent (stored in the register) in user programs in advance based on the slave communication protocol and required communication operations. The system automatically sends the data in the specified register area to serial ports successively. Then, the serial ports receive the data and save the received data in specified areas. Upon receipt of data of the specified length, the serial ports notify user programs through system flags so that user programs can parse received data based on the protocol.

  The register can be operated through the AM600 series free communication protocols. That is, user programs can directly access the communication buffer to process the sent and received data in the buffer, implementing communication through user-defined protocols. You need to configure and prepare for serial communication during programming so that communication can be conducted based on requirements. Related tasks include configuring the data sending and receiving mode of serial ports, baud rate, bits, parity bit, software protocols, and timeout conditions; preparing data for the sent and received data buffer; processing sending and receiving labels.

9.1.4 CANopen Communication Protocol

The CANopen communication protocol adopts the function block to read and write SDOs/PDOs, implementing communication. This protocol assigns communication variables (object dictionary data in the protocol) to corresponding input parameters of the function block and triggers execution conditions to access slave data. The AM600 series PLCs can function as the CANopen master only.

9.1.5 CANlink Communication Protocol

The CANlink communication protocol presets communication variables, communication frequency, and trigger conditions in a configuration table to implement communication. When a PLC serves as the CANlink master, it can be connected to various Inovance remote expansion modules, MD380 or MD500 AC drives, IS620 servos, and other slaves. When a PLC serves as the CANlink slave, it can be connected to other devices.

The CANlink3.0 communication protocol provides the following communication frames:

- Communication frames triggered in timed or conditional mode are used to exchange communication data between ordinary slaves.

- Synchronous trigger frames are used to control multiple highly real-time devices that can be controlled synchronously, for example, synchronous position control of multiple servos.
- Heartbeat frames are used to monitor the communication status of each CANlink slave to promptly respond to exceptions of the control system, avoiding further losses.

9.1.6 Ethernet Ports and Communication Protocols

The Ethernet port mainly provides the following two functions:

- Download PLC user programs and monitor and debug the system (like the Mini-USB port).
- Implement Ethernet communication using the TCP/IP Modbus communication protocol or free communication protocols.

You can configure communication parameters and address registers in the background for the Modbus protocol. The PLCs can access register values in user programs to exchange data with remote Modbus devices. Free communication protocols implement data exchange only by operating the function block of standard sockets.

9.1.7 EtherCAT Port and Communication Protocol

The EtherCAT port is used for full duplex communication with a baud rate of 1 Mbps in the linear topology through the standard EtherCAT protocol. The maximum communication distance between slave nodes is 100 m. The master supports synchronization events and DC mode. The maximum task jitter of the AM600 system is 120 μs (typical value).

9.1.8 High-Speed I/O Interface

The high-speed I/O interface has high-speed pulse control and high-speed pulse counting functions.

- The high-speed pulse control function is used to control pulse servo drives and stepper drives.
- The high-speed pulse counting function is used to collect A/B-phase, single-phase, CW/CCW pulse signal frequency and counting.

9.1.9 Mini-SD Card Slot

The Mini-SD card slot is mainly used to upgrade the underlying PLC firmware (unavailable to external users) and PLC user programs (available to external users).

9.1.10 Local Bus Expansion Interface

A PLC can be directly connected to the I/O module through a local bus expansion interface. The PLC updates addresses of I/O module data mapped to the PLC based on the internal bus cycle.

You can access the mapped addresses to operate the I/O module.

9.1.11 PROFIBUS-DP Port

The PROFIBUS-DP port and CAN port are integrated on one DB9 hardware port. Currently, the DP function is used only on the AM610 series and reserved for other products.

9.2 Soft Elements

Soft elements are global variables predefined on the programming system, which can be used directly. As direct variables, soft elements are mapped to the M area (%M) and are retentive at power failure (RETAIN). The AM600 programming system includes SD soft elements and SM soft elements. SD soft elements are INT direct global variables. SM soft elements are BOOL direct global variables.

The M area (%M) has a capacity of 512 KB, the first 480 KB of which is for users and the last 32 KB is for the system. Do not use the address directly. The first 30,000 bytes of the 32 KB space are used for SD and SM soft elements to implement special functions, such as CANlink, CANopen, high-speed I/O instructions, and Modbus, as detailed in the following table. You can access the soft elements.

SD Range	Function	SM Range	Function
0 to 7999	Register elements for users: 0 to 7000 for CANlink (CANlink configuration, compatible with small-sized PLCs)	0 to 7999	Bit elements for users: 0 to 3071 and 8000 to 8511 for CANlink (CANlink configuration, compatible with small-sized PLCs) 0 to 7999 indicates Modbus/Modbus TCP-triggered variables, which are used by slaves.
8000 to 8999	Register elements for the system: CANlink and CANopen	8000 to 8999	Bit elements for the system: CANlink and CANopen
9000 to 9999	Register elements for the system: high-speed I/O interface only	9000 to 9999	Bit elements for the system: high-speed I/O interface only

Note

- The system is automatically reset after Modbus-triggered variables are set.
- You can read but not write system soft elements. Otherwise, a system error may occur.

For details about how to use soft elements, see descriptions of CANlink soft elements, Modbus soft elements, and Modbus TCP soft elements.

9.3 Cheat Sheet of Basic Instructions

Type	Description	Name	Category
Arithmetic operation instruction	Addition	ADD	Function
	Multiplication	MUL	Function
	Subtraction	SUB	Function
	Division	DIV	Function
	Remainder	MOD	Function
Data processing instruction	Valuation	MOV	Function
	Batch data transmission	BMOV	Function
	One-to-many data transmission	FMOV	Function
	Obtaining the status of specific data bit	BON	Function
	Sum of ON bits	SUM	Function
	Conversion from byte to word	BTOW	Function
	Conversion from word to byte	WTOB	Function
	Byte swap	SWAP	Function
	Data exchange	XCH	Function
Word logic instruction	AND operation	AND	Function
	OR operation	OR	Function
	XOR operation	XOR	Function
	NOT operation	NOT	Function
Bit logic instruction (CmpHCUtils)	Rising edge output	PLS	Function block
	Falling edge output	PLF	Function block
	Alternate output	ALT	Function block
	Bit data output	BOUT	Function
	Bit data setting	BSET	Function
	Bit data reset	BRST	Function
Shift instruction	Shift left	SHL	Function
	Shift right	SHR	Function
	Rotation left	ROL	Function
	Rotation right	ROR	Function
	Rotation right with carry	RCR	Function
	Rotation left with carry	RCL	Function
	Copy bit data left	SFTL	Function
	Copy bit data right	SFTR	Function
	Copy word data left	WSFL	Function
	Copy word data right	WSFR	Function
	Data read (FIFO)	SFRD	Function
	Data write (FIFO)	SFWR	Function
Selection instruction	Either one	SEL	Function
	Maximum value	MAX	Function
	Minimum value	MIN	Function
	Limit	LIMIT	Function
	One from multiple	MUX	Function

Type	Description	Name	Category
Comparison instruction	Greater than	GT	Function
	Less than	LT	Function
	Greater than or equal to	GE	Function
	Less than or equal to	LE	Function
	Equal to	EQ	Function
	Not equal to	NE	Function
Basic mathematical operation instruction	Absolute value	ABS	Function
	Square root	SQRT	Function
	Natural logarithm	LN	Function
	Common logarithm	LOG	Function
	Exponent	EXP	Function
	Sine	SIN	Function
	Cosine	COS	Function
	Tangent	TAN	Function
	Arcsine	ASIN	Function
	Arccosine	ACOS	Function
	Arctangent	ATAN	Function
	Exponential	EXPT	Function
	Conversion from angle to radian	RAD	Function
Auxiliary mathematical operation instruction (Util library)	Differential	DERIVATIVE	Function block
	Integral	INTEGRAL	Function block
	Integral statistics	STATISTICS_INT	Function block
	Real number statistics	STATISTICS_REAL	Function block
	Variance	VARIANCE	Function block
Type conversion instruction Note: The following data types are supported: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, and DATE_ADN_TIME(DT).	Boolean conversion	BOOL_TO_<TYPE>	Function
	Byte conversion	BYTE_TO_<TYPE>	Function
	Date conversion	DATE_TO_<TYPE>	Function
	Long integer conversion	DINT_TO_<TYPE>	Function
	Date time conversion	DT_TO_<TYPE>	Function
	Double word conversion	DWORD_TO_<TYPE>	Function
	Integer conversion	INT_TO_<TYPE>	Function
	Word conversion	WORD_TO_<TYPE>	Function
	Real number conversion	REAL_TO_<TYPE>	Function
	Short integer conversion	SINT_TO_<TYPE>	Function
	Character conversion	STRING_TO_<TYPE>	Function
	Clock conversion	TIME_TO_<TYPE>	Function
	Time conversion	TOD_TO_<TYPE>	Function
	Long unsigned integer conversion	UDINT_TO_<TYPE>	Function
Address operation instruction	Address	ADR	Function
	Address content	^	Function
	Bit address	BITADR	Function
	Index	INDEXOF	Function
	Data size	SIZEOF	Function
Call instruction	Call	CAL	Function
Initialization instruction	Initialization	INI	Function

Type	Description	Name	Category
String processing instruction (standard library)	String length	LEN	Function
	Left string	LEFT	Function
	Right string	RIGHT	Function
	Middle string	MID	Function
	String concatenation	CONCAT	Function
	String insertion	INSERT	Function
	Character deletion	DELETE	Function
	String replacement	REPLACE	Function
	String finding	FIND	Function
Bistable instruction (standard library)	Setting-prior bistable trigger	SR	Function block
	Reset-prior bistable trigger	RS	Function block
Trigger instruction (standard library)	Rising edge inspection trigger	R_TRIG	Function block
	Falling edge inspection trigger	F_TRIG	Function block
Counter (standard library)	Incremental counter	CTU	Function block
	Decremental counter	CTD	Function block
	Incremental/decremental counter	CTUD	Function block
Timer instruction (standard library)	Timer	TP	Function block
	Power-on delay timer	TON	Function block
	Power-off delay timer	TOF	Function block
	Real-time clock	RTC	Function block
BCD conversion instruction (Util library)	BCD-to-integer conversion	BCD_TO_INT	Function
	Integer-to-BCD conversion	INT_TO_BCD	Function
Bit/byte operation instruction (Util library)	Bit extraction	EXTRACT	Function
	Bit packing	PACK	Function
	Bit unpacking	UNPACK	Function block
	Bit valuation	PUTBIT	Function
Controller instruction (Util library)	PD controller	PD	Function block
	PID controller	PID	Function block
	PID controller	PID_FIXCYCLE	Function block
Signal generator instruction (Util library)	Pulse signal generator	BLINK	Function block
	Cyclic signal generator	GEN	Function block
Robot operation instruction (Util library)	Characteristic curve	CHARCURVE	Function block
	Integral speed limit	RAMP_INT	Function block
	Real number speed limit	RAMP_REAL	Function block
Analog processing instruction (Util library)	Hysteresis	HYSTERESIS	Function block
	Upper/lower limit alarm	LIMITALARM	Function block

Type	Description	Name	Category
PLC system information instruction (SysHCPlcInfo library. For details, see help information about instruction use.)	Obtaining the system hardware information	SysHC_HWInfo	Function block
	Obtaining the system software information	SysHC_SWInfo	Function block
	Obtaining the CPU information	SysHC_CPUInfo	Function block
	Obtaining the CPU-related fault diagnosis information	SysHC_CPUDiagnose	Function block
	Obtaining the fault diagnosis information about the ModbusRTU slave	SysHC_ ModbusRtuDeviceDiagnose	Function block
	Obtaining the fault diagnosis information about the access from the ModbusRTU master to slave	SysHC_ ModbusRtuSlaveDiagnose	Function block
	Obtaining the fault diagnosis information about the ModbusTCP slave	SysHC_ ModbusTcpDeviceDiagnose	Function block
	Obtaining the fault diagnosis information about the access from the ModbusTCP master to slave	SysHC_ ModbusTcpSlaveDiagnose	Function block
	Setting the network information	SysHC_NetworkConfig	Function block
	Obtaining the network information	SysHC_NetworkInfo	Function block
	Obtaining the path information of the USB flash drive	SysHC_UDiskPath	Function block
	Obtaining the Boot version	GetBootVersion	Function
	Obtaining the PLC version	GetPLCVersion	Function
	Obtaining the device name	GetProductName	Function
	Obtaining the runtime version	GetRuntimeVersion	Function
	Obtaining the SN	GetSerialNumber	Function
	Saving the RETAIN information	SysHC_SaveAllRetains	Function
Time and date (CmpHCUtils)	Setting the current system clock	SetSystemDate	Function block
	Obtaining the current system clock and time zone	GetSystemDate	Function block
	Obtaining the operation time of the system, in milliseconds, microseconds, or nanoseconds	GetSystemTime	Function block
Table and range (CmpHCUtils)	Dead zone control	BZAND_TAB	Function
	Mean calculation	MEAN_TAB	Function
	Zone control	ZONE_TAB	Function
	Full data reset	ZRST_TAB	Function
	Obtaining the table coordinates	SCL_TAB	Function
	Sorting the table data	SORT_TAB	Function
	Ramp	RAMP_TAB	Function block
	Data summarization and calculation	WSUM_TAB	Function

Type	Description	Name	Category
Communication instruction (CmpHCUtils)	Creating a communication service on the TCP server	TCP_Server	Function block
	Creating a communication service on the TCP client	TCP_Client	Function block
	Creating a TCP connection and connecting to the server	TCP_Connect	Function block
	Receiving TCP communication data	TCP_Receive	Function block
	Sending TCP communication data	TCP_Send	Function block
	Creating a UDP communication connection	UDP_Peer	Function block
	Receiving UDP communication data	UDP_Receive	Function block
	Sending UDP communication data	UDP_Send	Function block
Filter instruction (CmpHCUtils)	Limiting filter	LimitingFilter	Function block
	Median filter	MedianFilter	Function block
	Arithmetic average filter	ArithmeticAverageFilter	Function block
	Recursive average filter	RecursiveAverageFilter	Function block
	Median average filter	MedianAverageFilter	Function block
	Limiting average filter	LimitingAverageFilter	Function block
	First-order lag filter	FirstOrderLagFilter	Function block
	Weighted recursive average filter	WeightRecursiveAverageFilter	Function block
	Debounce filter	DebounceFilter	Function block
	Limiting debounce filter	LimitingDebounceFilter	Function block
	Obtaining the operation time of the system, in milliseconds, microseconds, or nanoseconds	GetSystemTime	Function block
Queue (CmpHCUtils)	FIFO queue	FIFO	Function block

9.4 PLC Programming Software Upgrade

9.4.1 Version

- The programming software InoProShop V1.1.0 and earlier versions are incompatible with the latest version in terms of persistent variable, hard disk partition, high-speed I/O function, and EtherCAT bus I/O module. It is recommended to upgrade the software to the latest version. In addition, versions (earlier than V1.3.2) not mentioned in this section are not recommended. Contact local vendors if necessary.
- Slave files, for example, EtherCAT description file (.xml), CANopen description file (.eds), and PROFIBUS DP description file (.gds), must match the slave firmware version. If you have any questions, contact local vendors. Slaves not installed for V1.3.2 by default can be supported by installing corresponding device files.
- For details about how to use the AM400, AM600, and AC800 series, see hardware manuals or contact vendors.

9.4.2 Upgrade Method

- Application software installation
  A Windows 7 or Windows 10 operating system is required, and the memory must not be less than 4 GB. It is recommended to use a 64-bit instead of a 32-bit Chinese-English operating system.

  Install the software based on the wizard or set the installation path during installation as required. The default installation path is C:\Inovance Control\InoProShop.

Note

Do not install the software in the same folder with other versions.

- User project
  When you open a project in an earlier version, the "Project version information" window is displayed. If you do not want to update the project, select "Not update" to edit or use it directly. However, the ladder diagram must be updated.

  You can open the "Project version information" window using either of following methods:

  - Open an existing project to pop up the window automatically.
  - Choose "Project" > "Project version information".

  Two project update modes are supported:

  1. Full update: In the "Project version information" window, select "Set All to Latest Version" and then click "OK".
  2. Partial update: In the "Project version information" window, select the options to be updated and then click "OK".

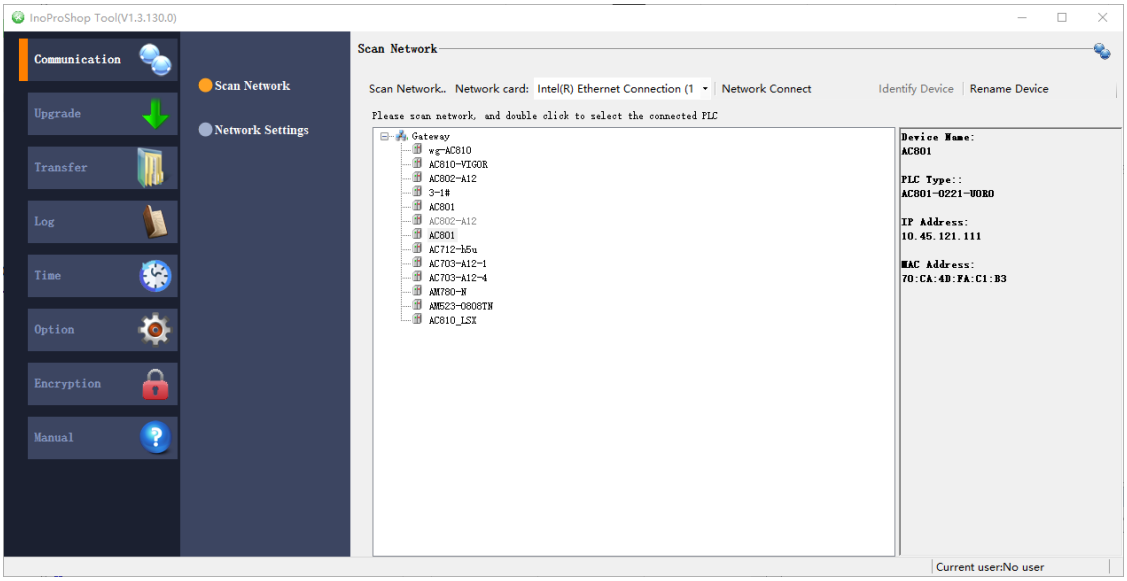
Note

The ladder diagram must be updated.

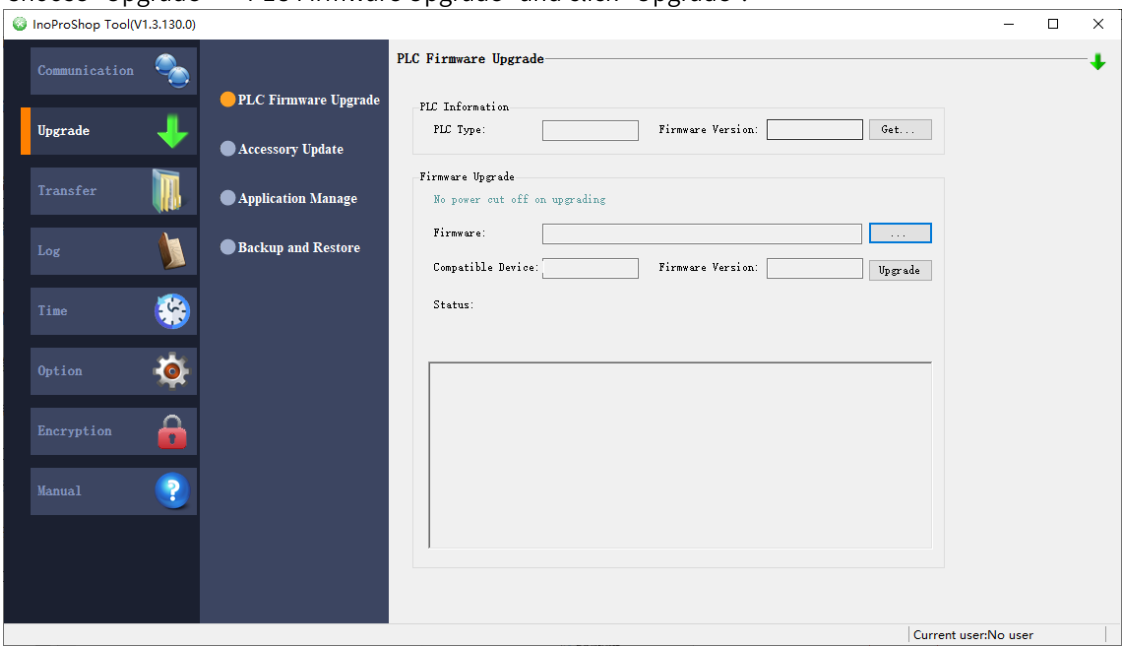
Online firmware upgrade

PLC (CPU module) upgrade

1. Choose "Tool" > "InoProShop Tool" > "Scan Network" and select a device.

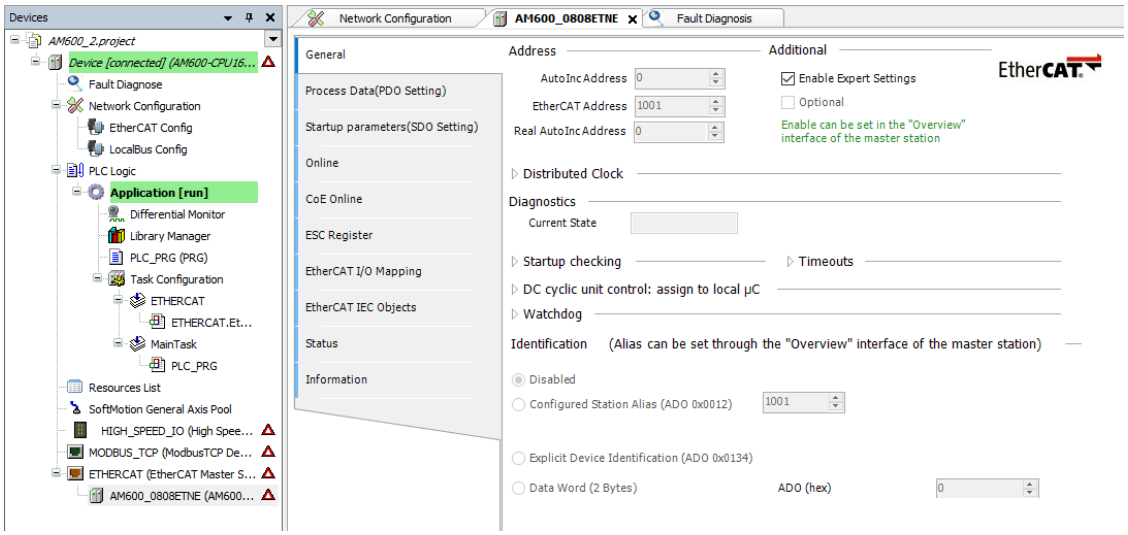


2. Choose "Upgrade" > "PLC Firmware Upgrade" and click "Upgrade".

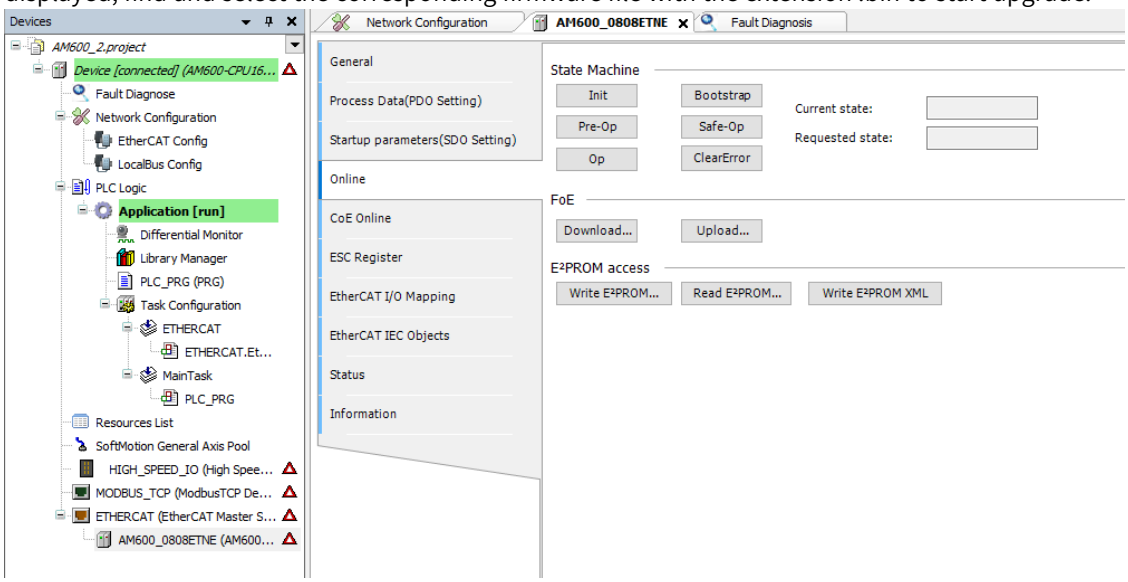


EtherCAT module upgrade

1. Choose "Device" > "General", select "Enable Expert Settings", and then choose "Download" > "Run".



2. Choose "Device" > "Online" and then click "Bootstrap" in the "State Machine" section. After the device enters the Bootstrap state, click "Download" in the "FoE" section. In the dialog box that is displayed, find and select the corresponding firmware file with the extension .bin to start upgrade.

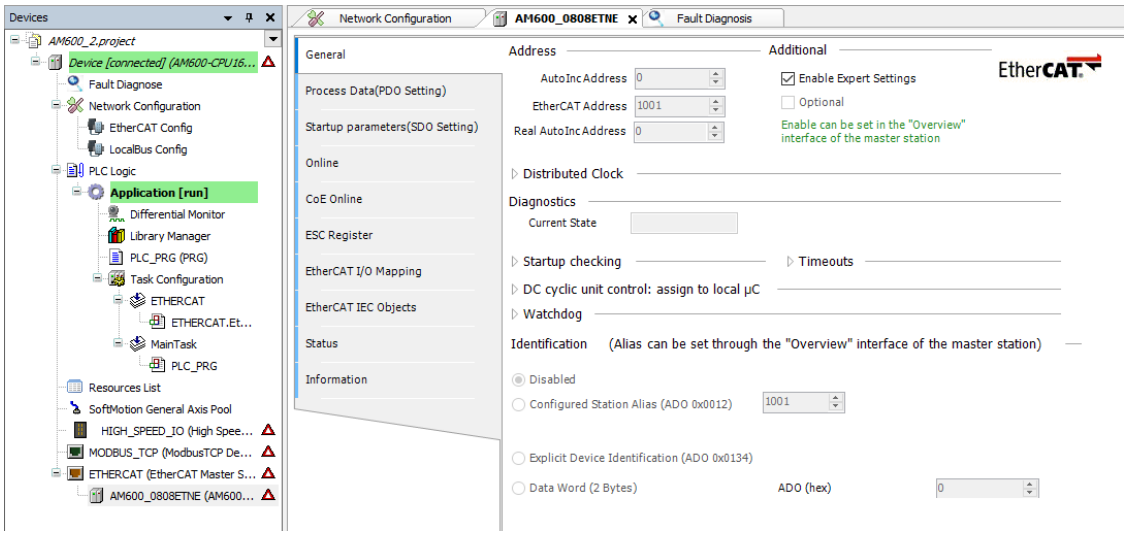


Library upgrade

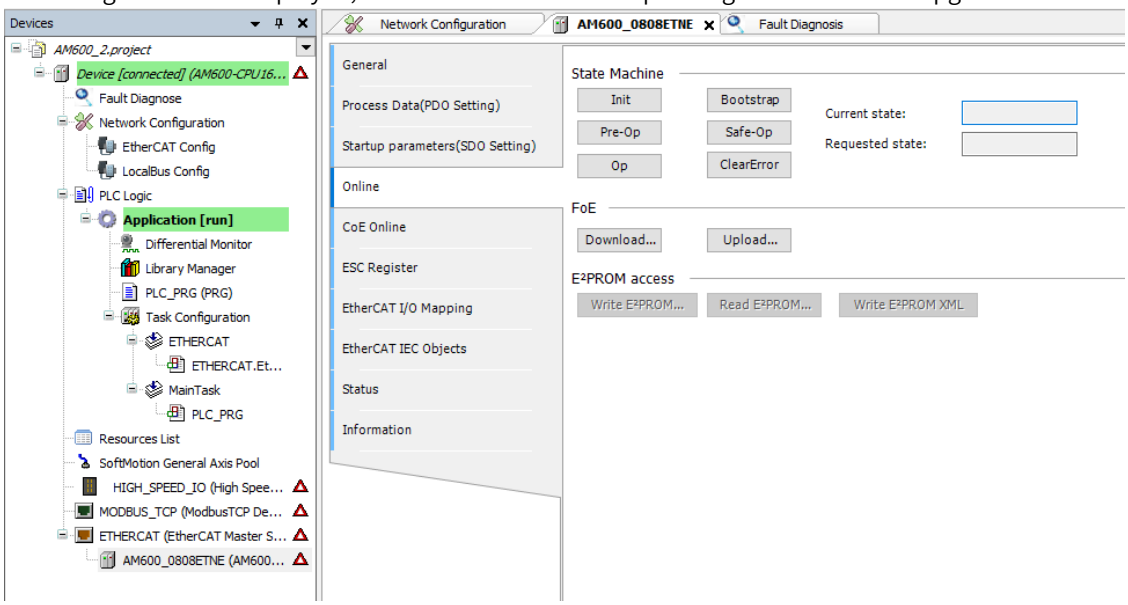
See "How do I add a compiled library to a project?" in FAQs.

EtherCAT device file upgrade

1. Choose "Device" > "General", select "Enable Expert Settings", and then choose "Download" > "Run".



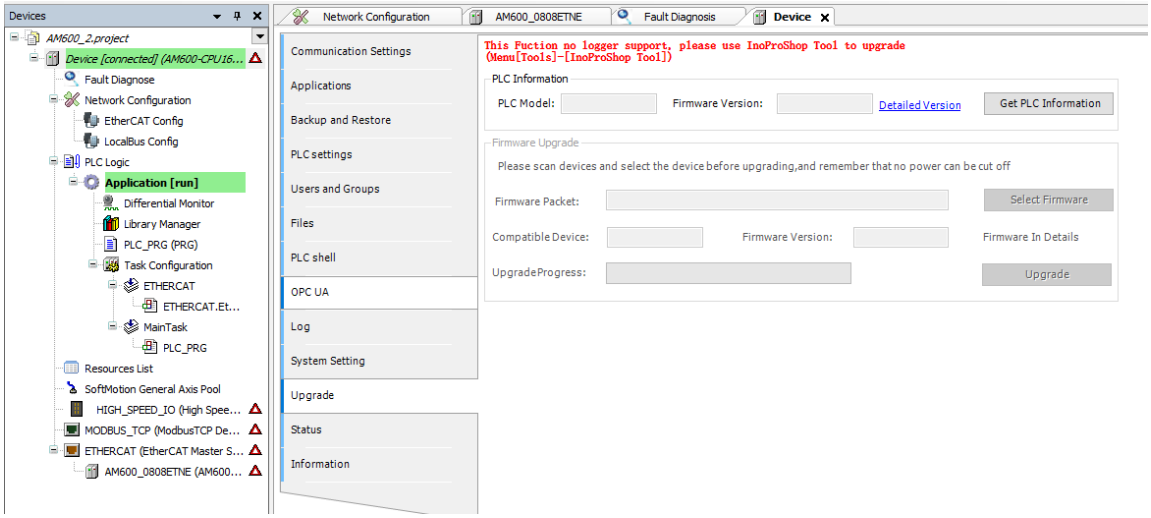
2. Choose "Device" > "Online" and then click "Write E2PROM XML" in the "E2PROM access" section. In the dialog box that is displayed, find and select the corresponding XML file to start upgrade.



9.4.3 FAQs

How do I check the version?

Double-click "Device (*XXX*)" in the device tree, click "Upgrade", and then click "Get PLC Information".



What can I do if the target system does not match the connected device?

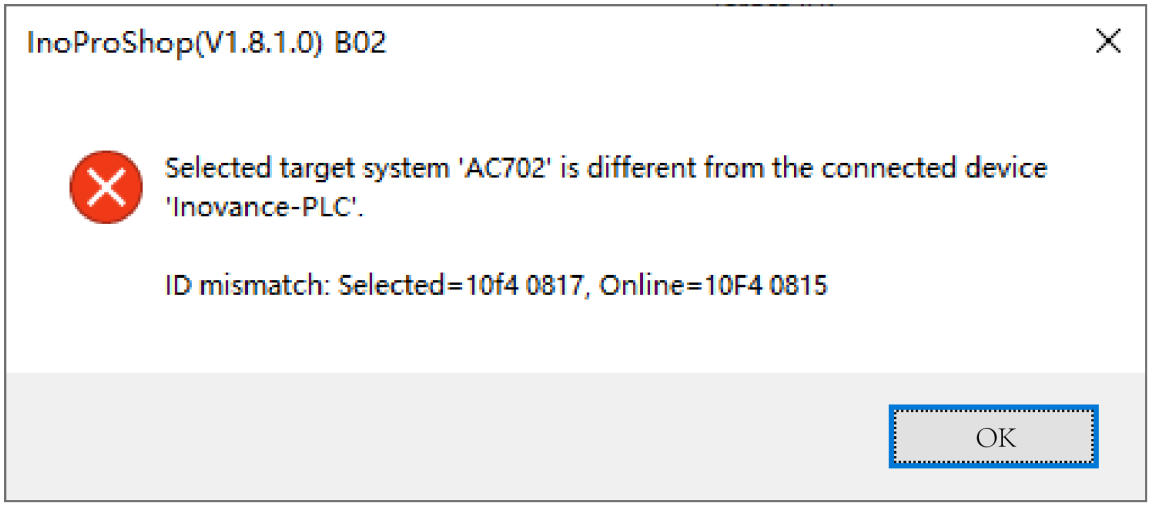


Figure 9-1 Target system not matching the connected device

Cause: The PLC version in InoProShop is V3.5.11.10, while the actual PLC version is V3.5.10.20. The device version in InoProShop cannot be later than the actual version.

- **Solution 1**: Upgrade the PLC firmware to match the PLC device version (V3.5.11.10).

  1. Right-click "Device (*XXX*)" and select "Update device". On the window that is displayed, select "Display all versions (for experts only)" to find the corresponding version and click "Update Device". If no matching version is found from the device list, you can select a version carrying the same first three numbers.

     As shown in the following figure, the version "3.5.10.20" does not exist in the list. In this case, you can select "3.5.10.40" (carrying the same first three numbers) and update the device.

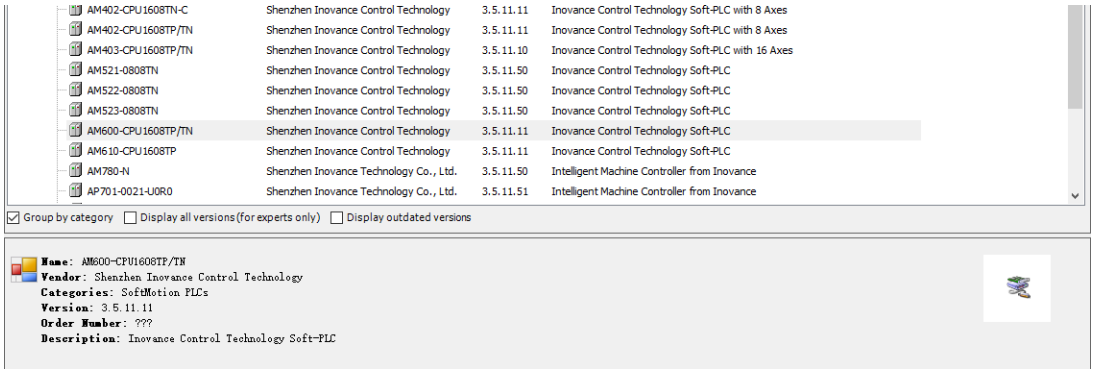


Figure 9-2 Updating the PLC version

2. Rescan and select the corresponding device. No error is reported.
3. Click "Upgrade" and upgrade the firmware in online mode in the "Firmware Upgrade" section.

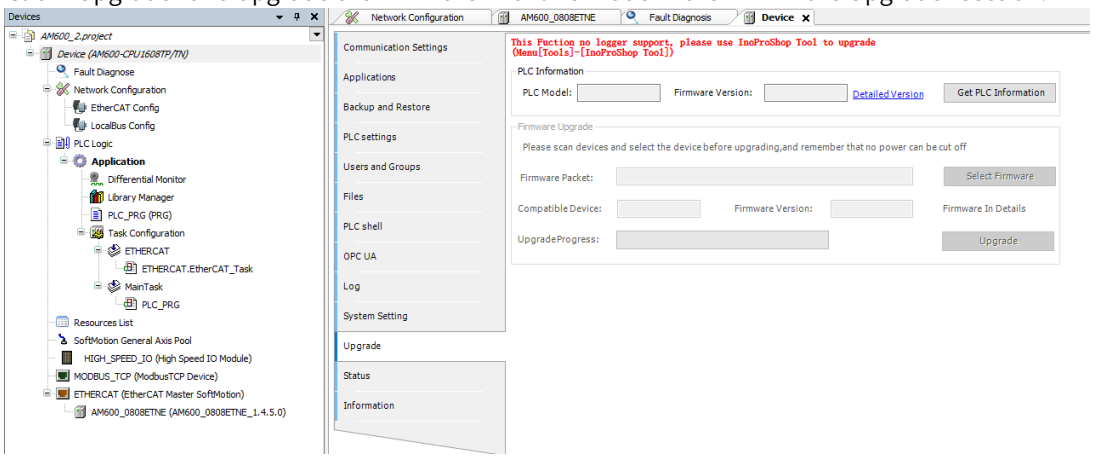


Figure 9-3 Online firmware upgrade

4. After the firmware is upgraded, upgrade the PLC to V3.5.11.10 as described in Step 1. Then, you can use the latest version of PLC and its firmware.

- **Solution 2**: Degrade the device to match the firmware version.
  Take Step 1 in Solution 1. However, device files of PLCs of earlier versions can be used only with IEC libraries that match the PLC versions.

  When an IEC library is added to the project, as the latest version of the IEC library is added by default, a compilation error may occur during program compilation because the library version does not match the PLC version. In this case, you can change the IEC library version manually.

What can I do if a compilation error occurs when I add programming software of the latest version to a library?

Use V1.3.2 to open a project created by software in versions earlier than V1.3.0 (V1.2.0 as an example), add the IEC library CmpBasic and use the MC_ResetDrive function block. A compilation error occurred.

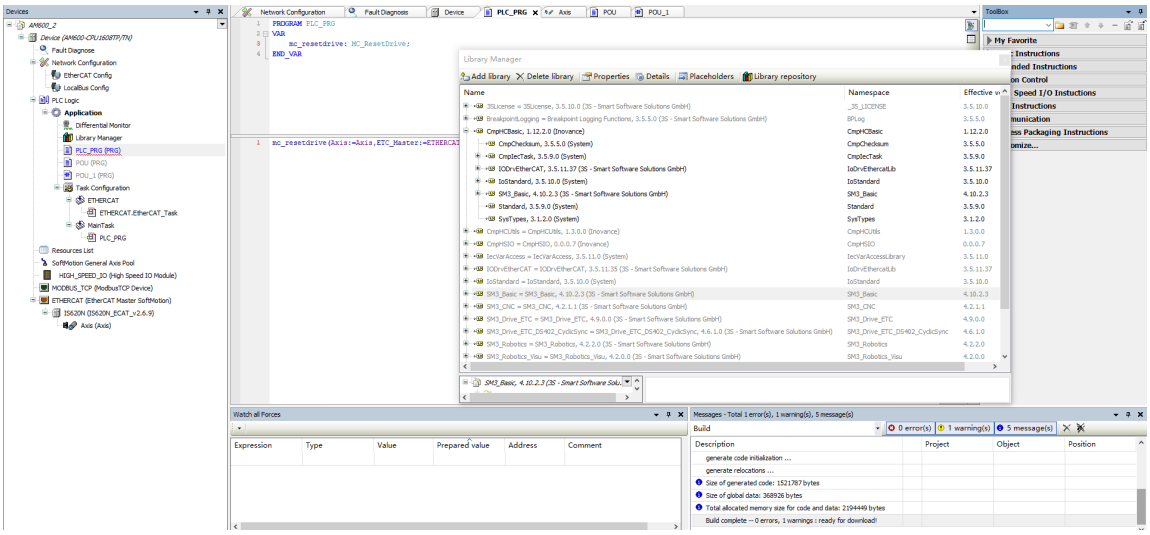


Figure 9-4 Compilation error occurred while adding a library

Cause: The version of specific libraries that the CmpBasic library depends on (SM3_Basic and IODrvEtherCAT libraries) does not match the version of dependent specific libraries in Library Manager. The IODrvEtherCAT version that CmpBasic (V1.8.0.0) depends on is V3.5.11.10, while the version referenced by the project is V3.5.10.0. The SM3_Basic version that CmpBasic (V1.8.0.0) depends on is V4.2.2.0, while the version referenced by the project is V4.2.1.0.

Solution:

1. Double-click "Library Manager" to display the Library Manager page. Select CmpBasis from the library list. The library version is V1.8.0.0.
2. On the Library Manager page, select "Properties". In the window that is displayed, select 1.6.0.0 (project created by V1.2.0) from the "Specific version" drop-down list in the "Version" section, which is the IEC library version matching the PLC, and then click "OK".

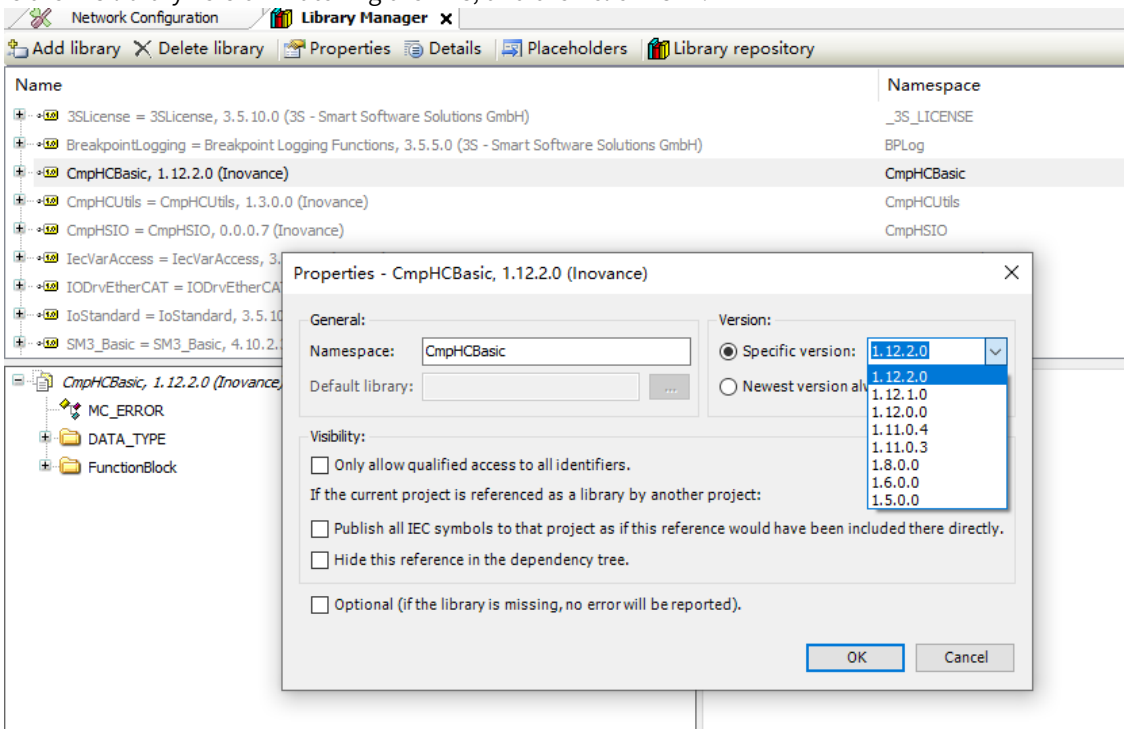


Figure 9-5 Updating the IEC library manually

3. Compile the project, as shown in the following figure.

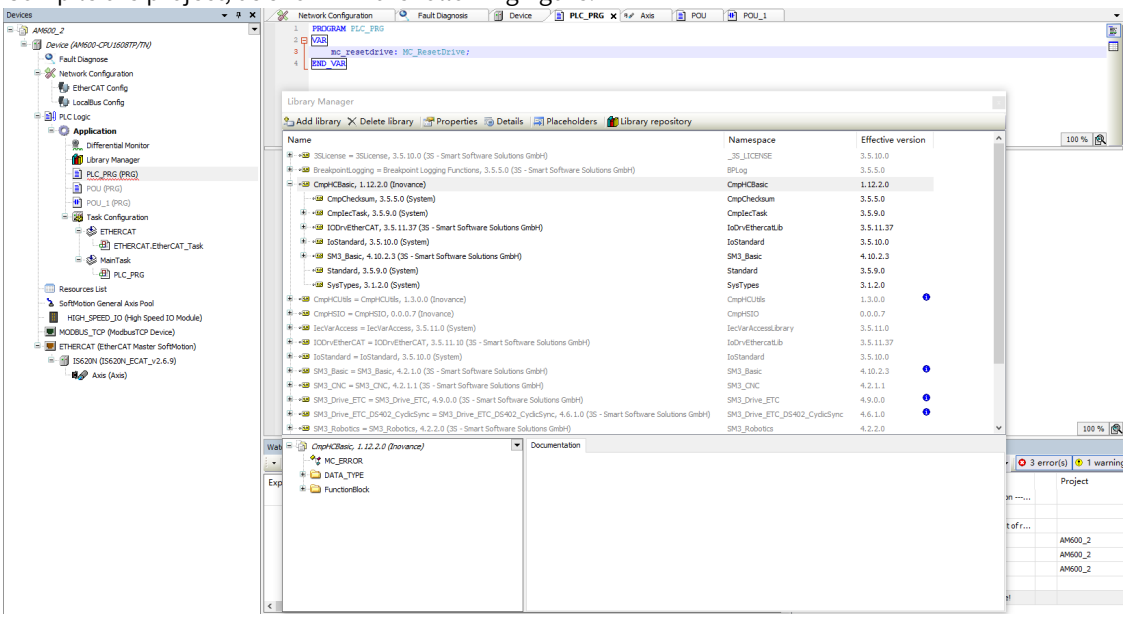


Figure 9-6 Updated IEC library and compilation information

Note

The major cause for library incompatibility is that the version of the system library that the library depends on does not match the version of the system library included in the previous project. Common libraries include IODrvEtherCAT and SM3_Basic.

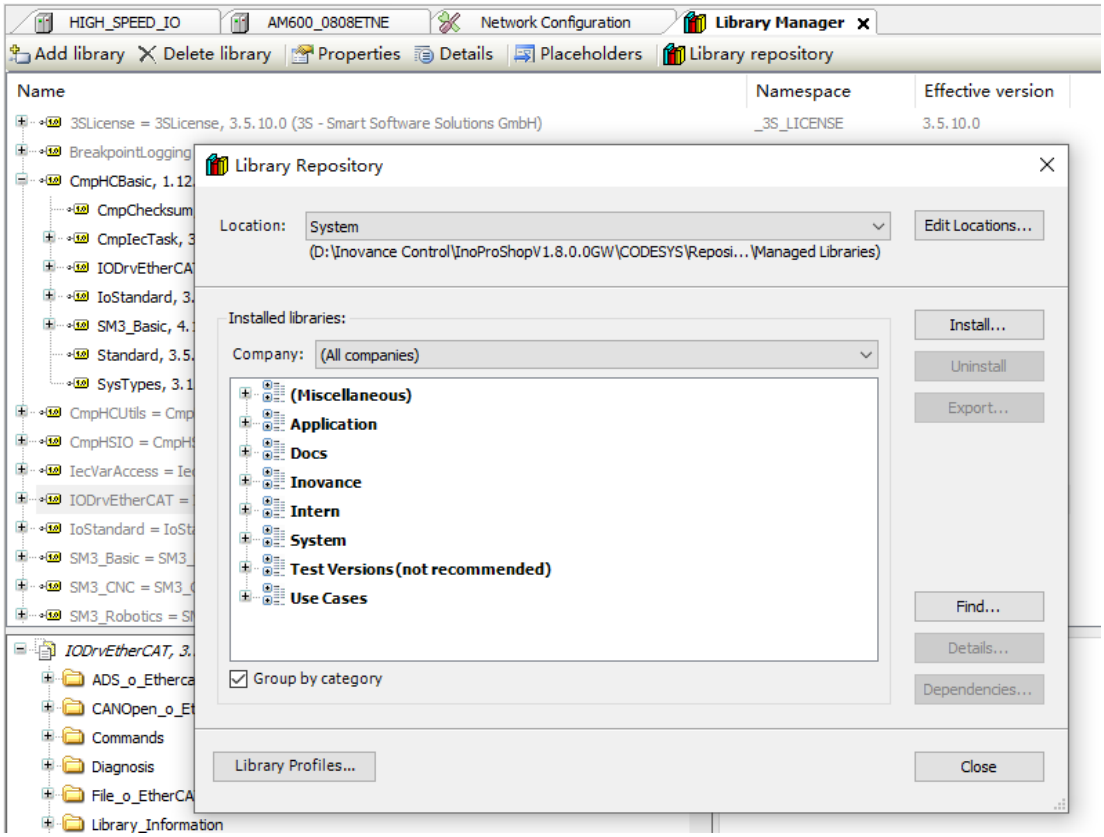
How do I add a compiled library to a project?

The following section uses the compiled library CmpBasic. compiled-library of V1.11.0.0 and software tool of V1.2.60 as an example. The operations are applicable to other versions of software tool.

1. Install a compiled library.

   a. Open a project and double-click "Library Manager".

   b. On the "Library Manager" page, click "Library repository". In the dialog box that is displayed, click "Install".

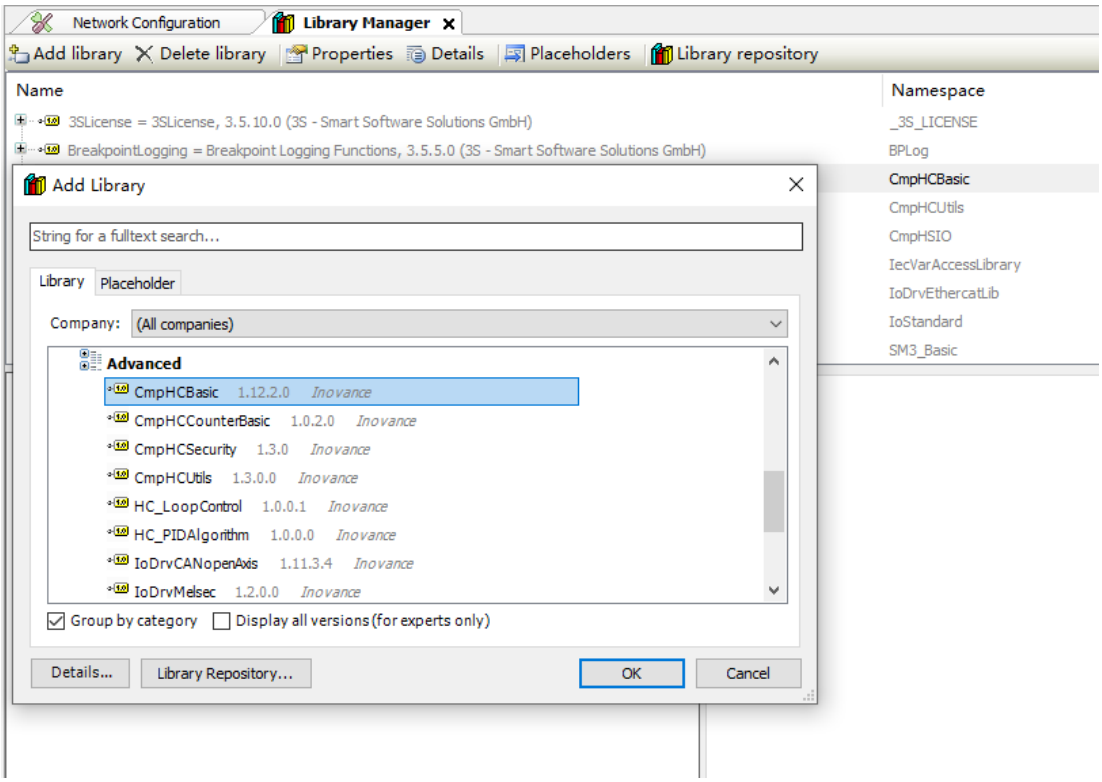


c. Find the compiled library (CmpBasic.compiled-library V1.11.0.0) and open it.

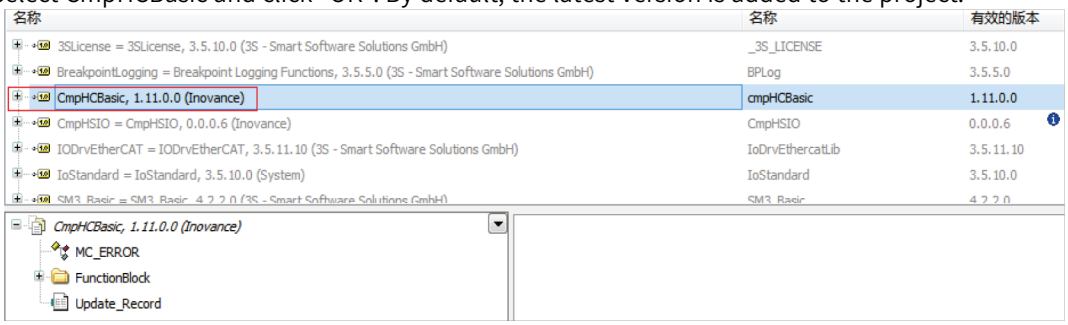


2. Add the library to the project.

a. On the "Library Manager" page, click "Add library". In the dialog box that is displayed, click the Add sign before "Miscellaneous".

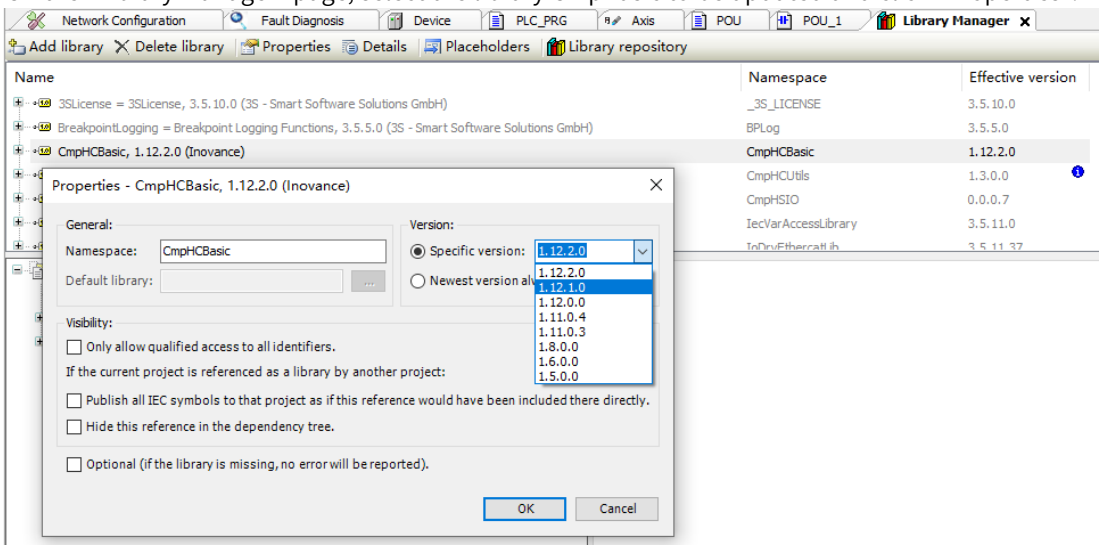


b. Select CmpHCBasic and click "OK". By default, the latest version is added to the project.



3. Select the library version manually.

a. On the "Library Manager" page, select the library CmpBasic to be updated and click "Properties".



b. In the dialog box that is displayed, select a version from the "Specific version" drop-down list in the "Version" section. (Confirm that the library version matches the software tool. Otherwise, the system reports a compilation error when you use the library).

Note

- To add a library to the project or update a library, choose "Build" > "Clear all" first.
- After the library is compiled, log in and download it again (a PLC error may be caused by online download).

9.5 PLC User Program Upgrade

9.5.1 Upgrade Using the InoProShop

Procedure:

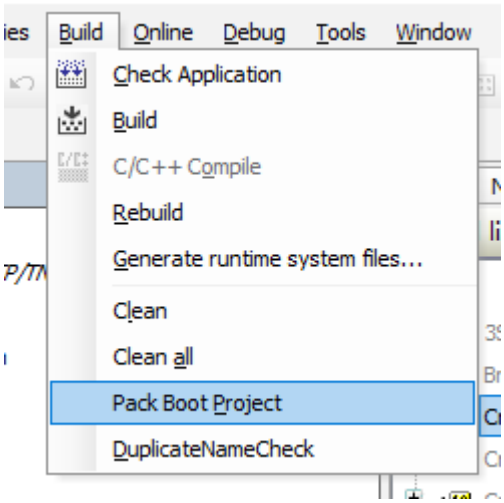
1. Choose "Tools" > "InoProShop Tool" to access the "InoProShop Tool" page.
2. Click "Help" to open the help manual.
3. Upgrade the PLC user program. For details, see the help manual.

9.5.2 Upgrade Using an SD Card

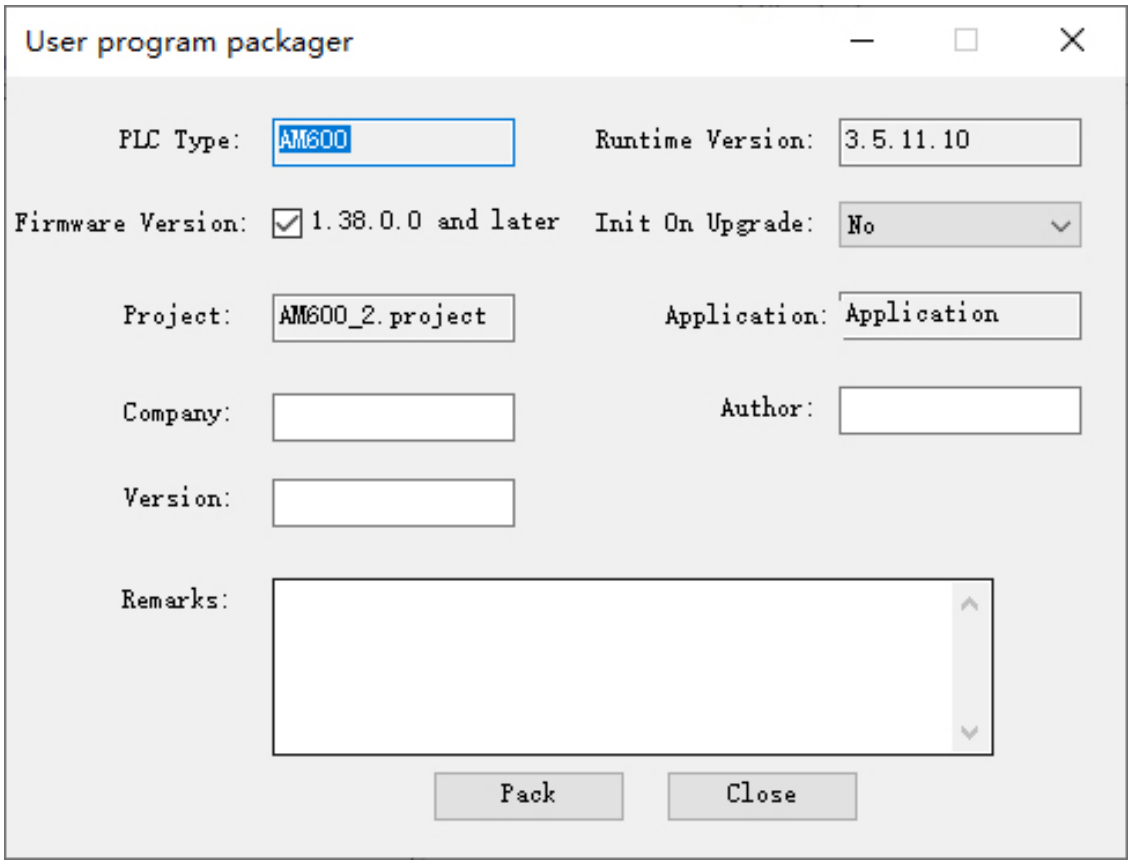
Prerequisites: A TF card (⩽ 32 GB) is available.

Procedure:

1. Open a compiled project and choose "Build" > "Pack Boot Project".



2. On the "User program packager" page, click "Pack" to pack the user program.



3. The Application.userprg file is generated in the directory for storing the user program. Copy this file to the root directory of the SD card.

Note

The file cannot be identified if it is stored to other directories of the SD card.

4. When the PLC is powered off, insert the SD or TF card into the SD card slot of the PLC.
5. When the DIP switch is set to Stop, power on the PLC. The user program starts upgrade automatically.
   In this case, the LED flashes 0 alternatively. This process lasts for approximately 20 seconds.



6. The upgrade process is completed when the LED stops flashing 0 alternatively. Remove the SD or TF card, and then power off and restart the PLC. The user program update is completed.
7. When the DIP switch is set to Run, upload and run the upgraded user program.

9.6 AM400 or AM600 High-Speed I/O Wiring

The CPU module of the AM400, AM600, and AM610 systems supports the high-speed I/O data processing function. The module has a built-in high-density port that provide 16 channels of high-speed inputs. The first six channels support 24 V single ended input or differential input, while the

latter 10 channels support 24 V single ended input. The following section describes how to wire the high-speed I/O signal interfaces and adapter terminals.

The following figure shows the high-density port (screenprint: CN5).

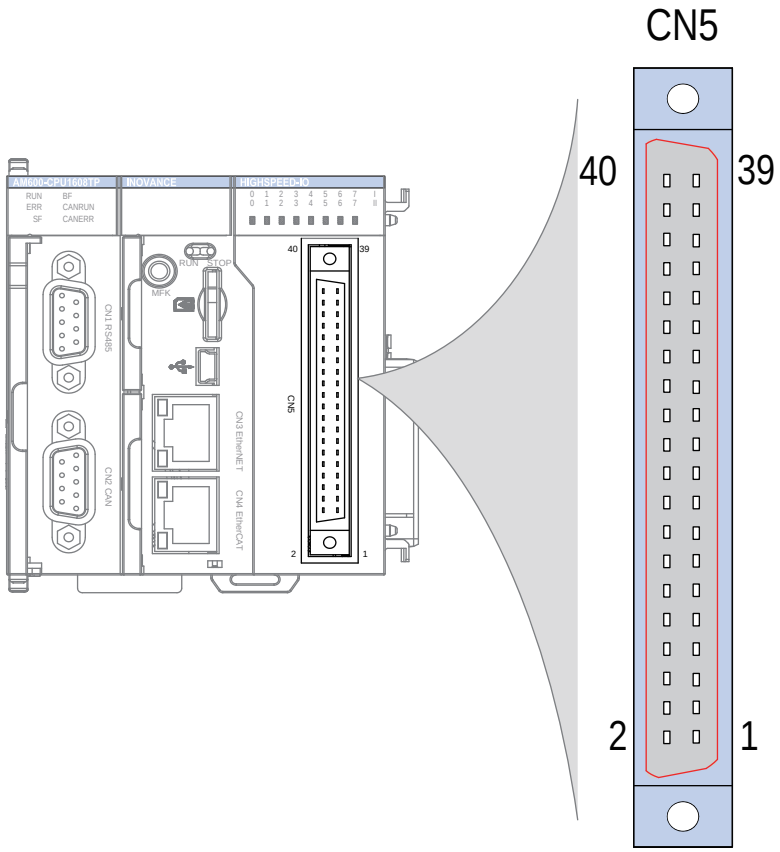
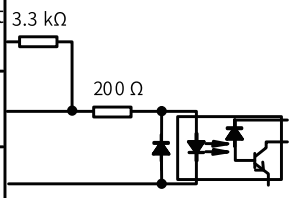
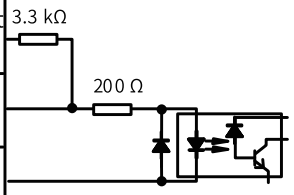
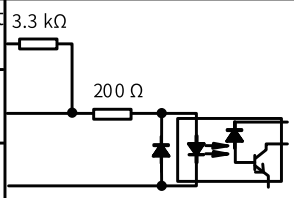
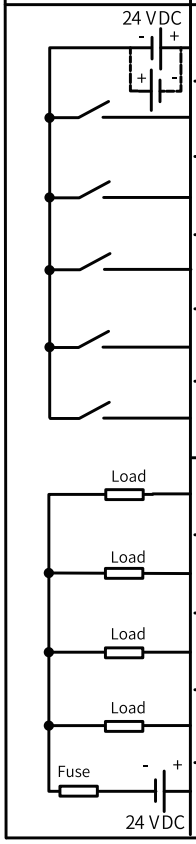
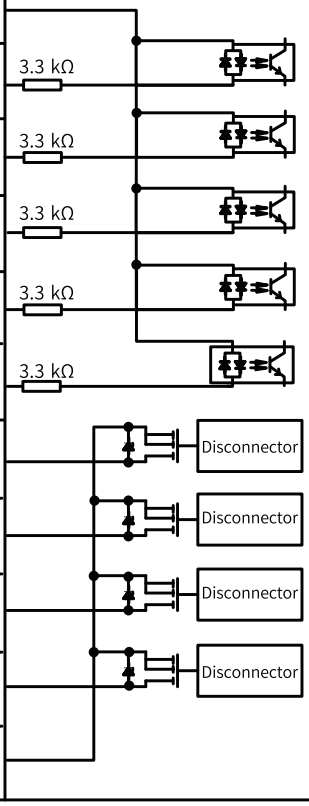


Figure 9-7 High-speed port on the CPU module of the AM600 and AM610 systems

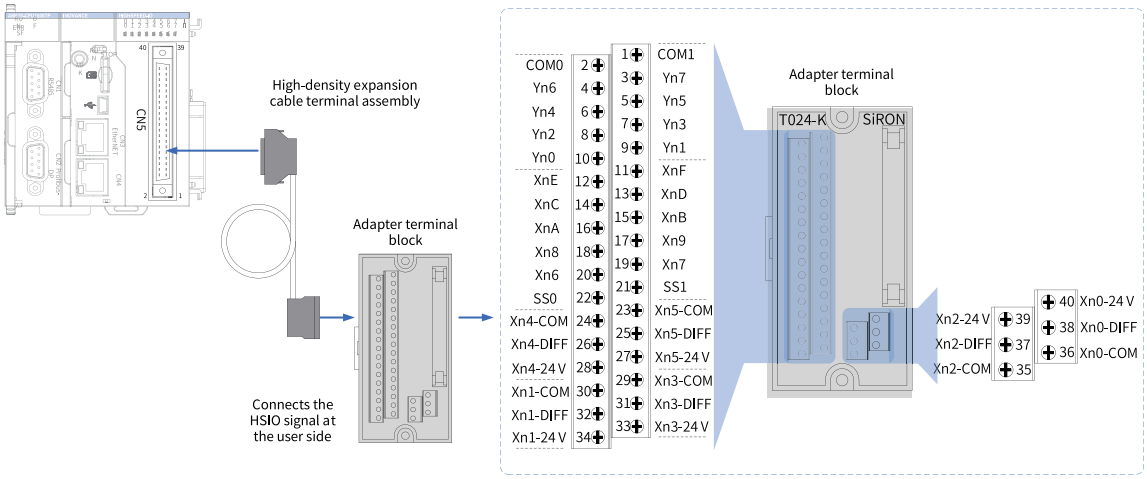
You need to connect the internal circuits and complete external wiring for the port based on actual needs.

External wiring	Signal Name Column B	CN5 Pin No.		Signal Name Column A	Internal circuit
See the application example.	High-speed 24 V input (Xn 0-24 V)	40	39	High-speed 24 V input (Xn 2-24 V)	
	High-speed differential input (Xn 0-DIFF)	38	37	High-speed differential input (Xn 2-DIFF)	
	High-speed input common terminal (Xn 0-COM)	36	35	High-speed input common terminal (Xn 2-COM)	
See the application example.	High-speed 24 V input (Xn 1-24 V)	34	33	High-speed 24 V input (Xn 3-24 V)	
	High-speed differential input (Xn 1-DIFF)	32	31	High-speed differential input (Xn 3-DIFF)	
	High-speed input common terminal (Xn 1-COM)	30	29	High-speed input common terminal (Xn 3-COM)	
See the application example.	High-speed 24 V input (Xn 4-24 V)	28	27	High-speed 24 V input (Xn 5-24 V)	
	High-speed differential input (Xn 4-DIFF)	26	25	High-speed differential input (Xn 5-DIFF)	
	High-speed input common terminal (Xn 4-COM)	24	23	High-speed input common terminal (Xn 5-COM)	
	Input common terminal (SS 0)	22	21	Output common terminal (SS 1)	
	Standard input (Xn 6)	20	19	Standard input (Xn 7)	
	Standard input (Xn 8)	18	17	Standard input (Xn 9)	
	Standard input (XnA)	16	15	Standard input (XnB)	
	Standard input (XnC)	14	13	Standard input (XnD)	
	Standard input (XnE)	12	11	Standard input (XnF)	
	Output (Yn 0)	10	9	Output (Yn 1)	
	Output (Yn 2)	8	7	Output (Yn 3)	
	Output (Yn 4)	6	5	Output (Yn 5)	
	Output (Yn 6)	4	3	Output (Yn 7)	
	Output common terminal (COM 0)	2	1	Output common terminal (COM 1)	

Note

To avoid wiring errors, connect the high-speed DI pins of the first six channels by referring to the application example below.

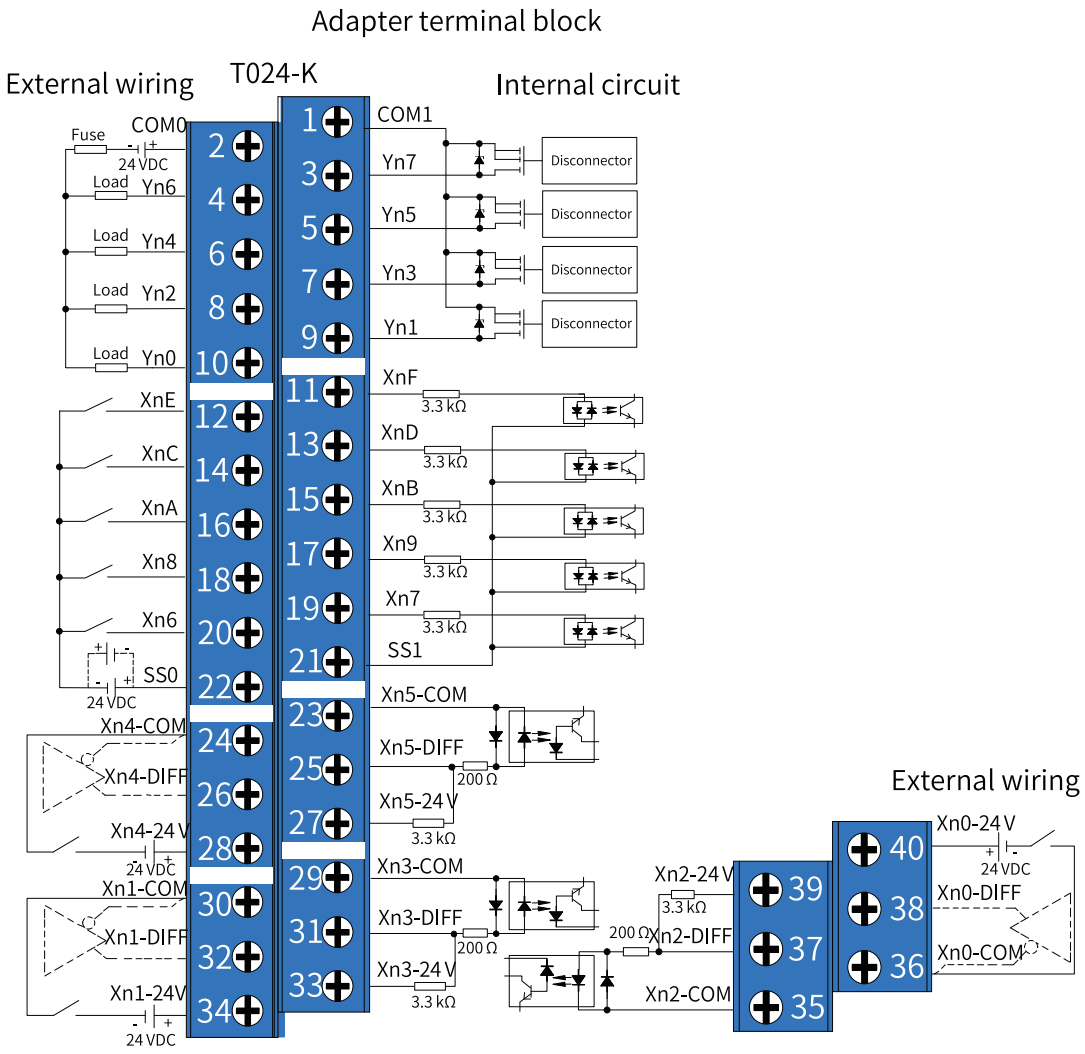
If adapter wiring is completed using the SIRON T024-K adapter terminals, refer to the mappings between the terminal numbers and CN5 pin numbers of the module, as shown in the following figure.



The following components in the preceding figure can be purchased from Inovance: ① high-density expansion cable; ② connecting plug (required for customized cable); ③ adapter terminal block. The order information is as follows.

No.	Part Number	Description
①	15300119	40-pin FCN-to-MIL high-density expansion cable (500 mm, including two 40-pin FCN connecting plugs)
②	15050180	40-pin FCN connecting plug (If you do not purchase a high-density expansion cable, you can purchase this plug to make a cable by yourself.)
③	15020452	40-pin MIL-to-screw wiring terminal block

SIRON T024-K adapter terminal block



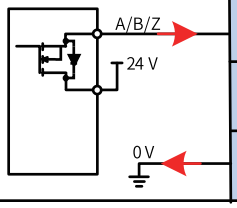
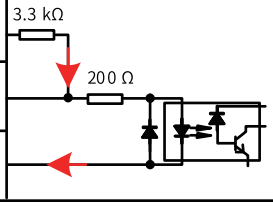
Note

The pin definition and wiring instructions for the high-density port on the CPU module are described above. Read the information carefully before you perform wiring operations.

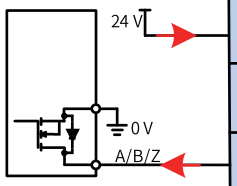
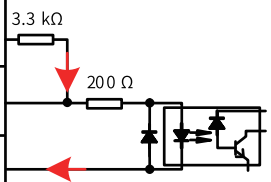
Application example

The first 4 channels of the high-speed I/O interface support single ended and differential signals. Pay attention to wire them correctly. The following uses Xn0 as an example to describe the wiring.

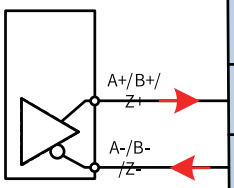
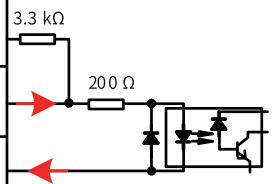
1. Wiring for 24 V level input of the PNP type

Input Type	External Wiring	No.	Signal Name	Internal Circuit
PNP collector (24 V level)		40	High-speed 24 V input (Xn0-24V)	
		38	High-speed differential input (Xn0-DIFF)	
		36	High-speed input common terminal (Xn0-COM)	

2. Wiring for 24 V level input of the NPN type

Input Type	External Wiring	No.	Signal Name	Internal Circuit
NPN collector (24 V level)		40	High-speed 24 V input (Xn0-24 V)	
		38	High-speed differential input (Xn0-DIFF)	
		36	High-speed input common terminal (Xn0-COM)	

3. Wiring for 5 V level input of differential signals

Input Type	External Wiring	No.	Signal Name	Internal Circuit
Differential signal (5 V level)		40	High-speed 24 V input (Xn 0-24 V)	
		38	High-speed differential input (Xn 0-DIFF)	
		36	High-speed input common terminal (Xn 0-COM)	

9.7 High-Speed I/O Compatibility

9.7.1 Introduction to Earlier and Latest UIs

CmpHCBuiltinIo and CmpHSIO represent the high-speed I/O function block libraries for earlier versions and the latest version, respectively.

InoProShop in V1.2.0 (temporary version: 1.1.60.0), AM600 firmware in V1.19.70.0, and FPGA in A624 and later versions support the latest high-speed I/O function block library.

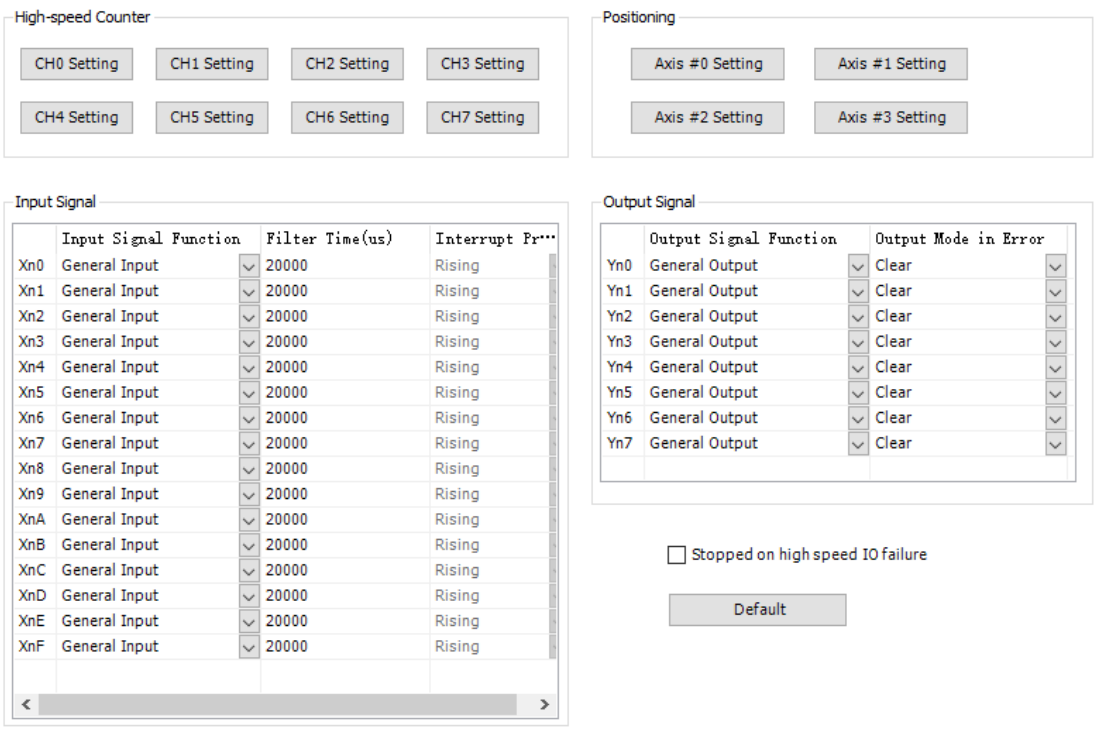


Figure 9-8 UI for high-speed I/O function block library in earlier versions

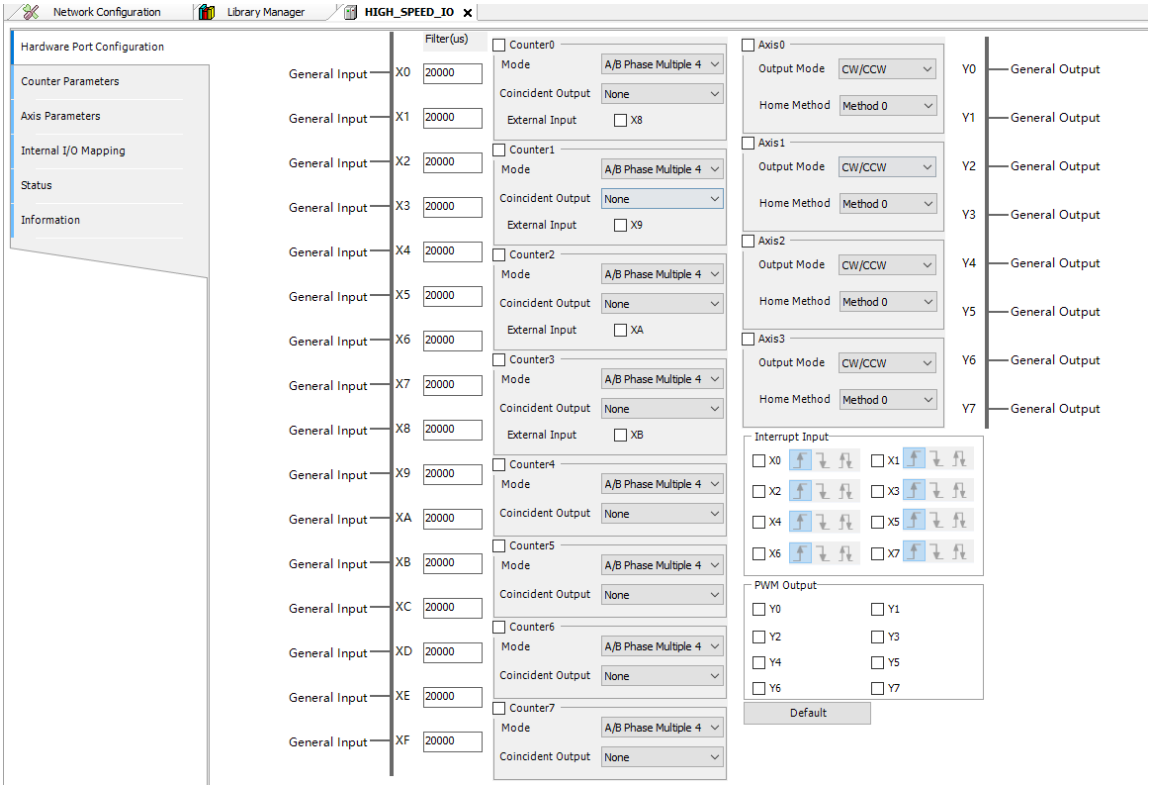
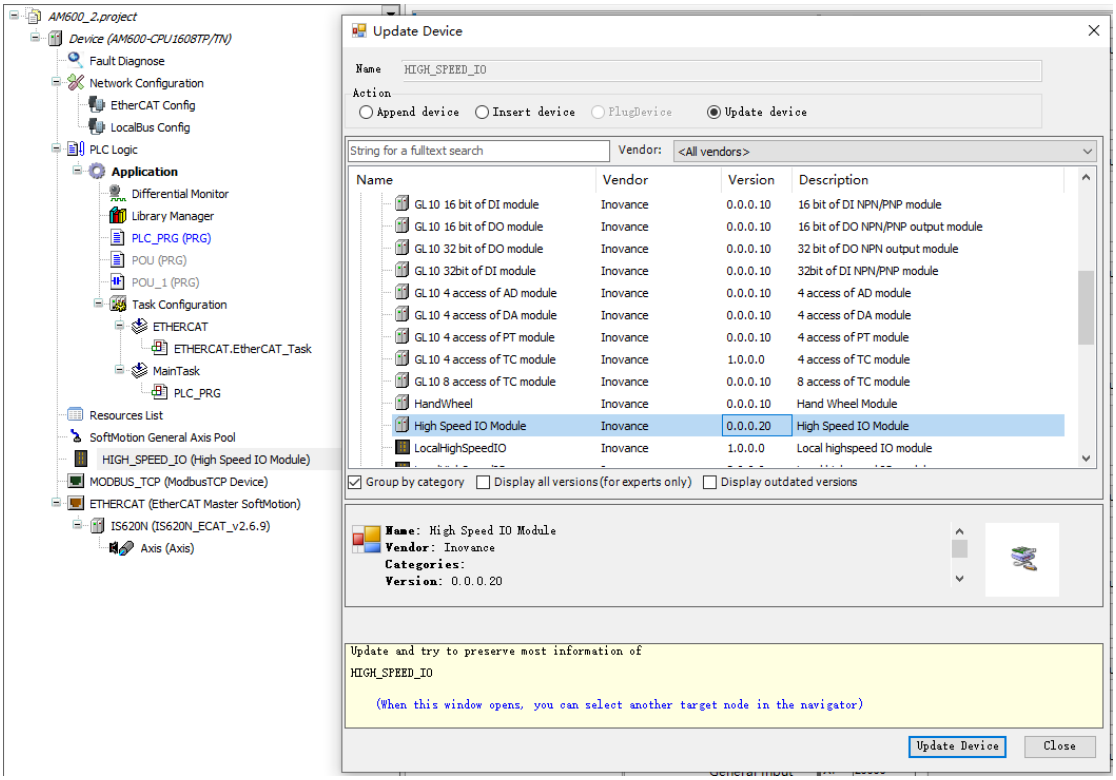


Figure 9-9 UI for high-speed I/O function block library in the latest version

- The latest version of high-speed I/O device can only be used with the latest version of high-speed I/O library. The earlier version of high-speed I/O device can only be used with the earlier version of high-speed I/O library.

- To use functions of the latest version of high-speed I/O device, you need to upgrade both PLC firmware and FPGA.
- The latest and earlier high-speed I/O functions are compatible (except the homing function). If the latest high-speed I/O device is mixed with the earlier high-speed I/O device (the latest high-speed I/O project with an earlier PLC high-speed I/O device or an earlier I/O project with the latest PLC high-speed I/O device), a message is displayed reminding you to switch the PLC, as shown in the following figure. You need to switch the version of high-speed I/O device and restart the PLC.
- If you do not want to switch the PLC, you can switch the version of high-speed I/O device for the project on the "Update Device" page.
  As shown in the following figure, the earlier high-speed I/O device is V0.0.0.10, and the latest I/O device is V0.0.0.20.



- If you use earlier software tool (for example, V1.1.0 or V0.0.9.10) to download an earlier high-speed I/O project to the PLC with the latest version of firmware (later than V1.19.70.0), the error indicating version not matched is reported.
  Solution: Install the latest version of software tool, and use it to open and download an earlier project (the high-speed I/O function not used) to the PLC with the latest version of firmware. In this case, no error is reported.

9.7.2 High-Speed I/O Diagnosis

High-speed I/O device of the latest version

Basic format:

Library + function block + error code

3 3-bit

- Library: The default library of the high-speed I/O device is 0.
- Function block: Function blocks are numbered from 01.
- Error code: The error code starts from 01. Common error codes are listed in the following tables. An error code less than 500 indicates a major error, whereas an error code greater than 500 indicates a function block error. For example, in the error code 14506, 14 indicates HC_WriteParameter, and 506 indicates a parameter error. In the error code 31520, 31 indicates MC_WriteParameter_P, and 520 indicates a parameter error.

Table 9–1 List of counter error codes

Error Code	Definition	Description
001	ERR_COUNTERID_INVALID	The entered channel number is invalid. The value range is from 0 to 7.
003	ERR_CNT_OVERFLOW	A counter overflow or underflow error occurs.
004	ERR_COUNTER_NOT_CHOSEN	No high-speed function is selected. Select one in the programming software.
007	ERR_COUNTER_NOT_ENABLED	HC_Counter is disabled.
101	ERR_WRITEINTERRUPTPARAMETER_ UNVALIAD	The write interrupt parameter is invalid.
102	ERR_INTERRUPT_NOT_CHOSE	Interrupt Input is not selected in the programming software.
501	ERR_SETCOMPARE_IMREFRESHCYCLE_ OVERFLOW	The comparison value ImRefreshCycle exceeds 30000. The value range is from 0 to 30000.
502	ERR_SETCOMPAREM_NUMBERS_ OVERFLOW	The value range of HC_SetCompareM number is from 1 to 100.
503	ERR_PREWR_VALUE_OVERFLOW	The preset value is out of range.
504	ERR_AVERAGE_PARA_UNVALIAD	The set average frequency and average rotational velocity are invalid.
505	ERR_ROTATION_PULSES_UNIT_UNVALIAD	The set number of pulses per rotation is invalid.
506	ERR_WRITEBOOlPARAMETER_UNVALIAD	The set parameter HC_WriteBoolParameter is invalid.
507	ERR_READBOOLPARAMETER_UNVALIAD	The obtained parameter HC_ReadBoolParameter is invalid.
508	ERR_MEASURE_WIDTH_OVERFLOW	The measured width is invalid.
509	ERR_SETCOMPAREM_IMREFRESHCYCLE_ OVERFLOW	The comparison value ImRefreshCycle exceeds 30000. The value range is from 0 to 30000.
510	ERR_PRESET_TRIGGERTYPE_OVERFLOW	The preset parameter is invalid.
511	ERR_WRITEPARAMETER_UNVALIAD	The set parameter HC_WriteParameter is invalid.
513	ERR_FUNC_COUNTERID_INVALID	The special function channel number is invalid. The value range is from 0 to 3.
514	ERR_COUNTER_NOT_CHOSE_EXETERNAL_ X	External Trigger is not selected in the programming software.
515	ERR_CNT_FORMAT_NOT_RING	The ring counting type is incorrect. Select a correct type in the programming software.
516	ERR_RING_DOWNVAL_BEYOND_UPVAL	The lower limit for ring counting is equal to or greater than the upper limit.
517	ERR_SAMPLE_VALUE_LESS	The sampling time is too short. The value range is from 10 to 65535, in ms.
518	ERR_RING_VALUE_OVERFLOW	The ring counting is out of range.

Table 9–2 List of high-speed axis error codes

Error Code	Definition	Description
001	ERR_NOT_POWER	MC_Power is disabled.
002	ERR_UP_SOFTWARE_LIMIT	The current position is beyond the software stroke limit (Up).
003	ERR_DOWN_SOFTWARE_LIMIT	The current position is beyond the software stroke limit (Down).
004	ERR_AXIS_FUNC_UNUSED	The high-speed axis is disabled. Enable the axis in the programming software.
005	ERR_INPUT_CHANNAL_NUM_INVALID	The axis number is invalid. The value range is from 0 to 3.
006	ERR_DEST_POS_OVER_SOFT_UP_LIMIT	The target position is beyond the upper software limit.
007	ERR_DEST_POS_OVER_SOFT_DOWN_ LIMIT	The target position is beyond the lower software limit.
010	ERR_POS_DECPOINT_OVERLOW	Invalid deceleration point: In position mode, when the device is repositioned, the deceleration length is greater than the actual distance.
011	ERR_VEL_DECPOINT_OVERLOW	Invalid deceleration point: When you switch from the velocity mode to the position mode, the deceleration length is greater than the actual distance.
012	ERR_POS_PLSNUM_OVERLOW	The maximum PLSNUM positioning length 2147483647 is exceeded.
013	ERR_POS_DECPOINT2_OVERLOW	An error occurs while recomputing the deceleration point.
501	ERR_ACC_SET_OVERFLOW	The acceleration exceeds the maximum value set by MC_WriteParameter_P.
502	ERR_ACC_SET_LOW	The acceleration is below the minimum value set by MC_WriteParameter_P.
503	ERR_DEC_SET_OVERFLOW	The deceleration exceeds the maximum value set by MC_WriteParameter_P.
504	ERR_DEC_SET_LOW	The deceleration is below the minimum value set by MC_WriteParameter_P.
505	ERR_VEL_SET_OVERFLOW	The set velocity is out of range. Set the velocity in the programming software or through MC_ WriteParameter_P.
506	ERR_VEL_SET_LOW	The set velocity is too low.
508	ERR_VEL_LESS_THAN_STARTVEL	The velocity is lower than the startup offset velocity. Set the startup offset velocity in the programming software.
509	ERR_STARTVEL_SET_LOW	The starting velocity is too small.
510	ERR_FBD_MOVEMODE_INVALIAD	The motion mode of the function block is invalid.
511	ERR_WASNT_STANDSTILL	The axis is not in Standstill state.
512	ERR_WASNT_DISABLED	The axis is not in Disabled state.
513	ERR_IN_ERRORSTOP	The axis is not in ErrorStop state.
514	ERR_NOT_READY_FOR_MOTION	The axis is not ready to run.
515	ERR_INVLALID_VELOCITY_MODE	The velocity mode is invalid.
516	ERR_INVLALID_POSTION_MODE	The position mode is invalid.
520	ERR_AXIS_WRITEPARAMETER_UNVALIAD	The MC_WriteParameter_P parameter is invalid.
521	ERR_AXIS_READPARAMETER_UNVALIAD	The MC_ReadParameter_P parameter is invalid.

Error Code	Definition	Description
522	ERR_HOME_MODE_UNVALIAD	The homing mode is invalid. Select a valid mode in the programming software.
523	ERR_AXIS_WRITEPARAMETER_HOME_MODE_UNVALIAD	The homing mode is invalid.

Errors are classified into axis errors and function block errors.

Conditions for setting the axis to ErrorStop state:

- An axis error occurs.
- A function block occurs when the axis is in DiscreteMotion, ContinuousMotion, or Homing state.

High-speed I/O device of earlier versions

High-speed I/O diagnosis information is displayed on the high-speed I/O self-diagnosis page. For descriptions of the self-diagnosis page, see the overview of the list of device self-diagnosis information.

You can obtain high-speed I/O diagnosis information through high-speed I/O soft elements. High-speed I/O diagnosis results include channel errors, channel alarms, axis errors, axis alarms, and other faults. The diagnosis states and diagnosis codes of channel errors, channel alarms, axis errors, and axis alarms are indicated by soft elements. The diagnosis state indicates whether diagnosis information exists, and the diagnosis code indicates the error code. The following table shows soft elements, diagnosis codes, and diagnosis information corresponding to each type.

- Channel error
  The following table lists the relationship among the channel number, error flag soft element, and error diagnosis code soft element.

Channel Number	Error Flag Soft Element	Error Diagnosis Code Soft Element
0	SM9030	SD9007
1	SM9080	SD9017
2	SM9130	SD9027
3	SM9180	SD9037
4	SM9230	SD9047
5	SM9380	SD9057
6	SM9330	SD9067
7	SM9380	SD9077

  The following table lists the relationship between the diagnosis code and diagnosis information.

Diagnosis Code	Diagnosis Information
1001	The channel type does not match.
1002	A counter overflow occurs.
1003	A pulse width measurement overflow occurs.
1011	The lower limit of the ring counter exceeds the upper limit.
1012	The counter type does not match.
1013	The high-speed counting function is not used.
1014	The high-speed counter function does not match.
1015	The preset value is out of range.
1016	The average parameter is invalid.
1017	The set number of pulses per rotation is invalid.

- Channel alarm

The following table lists the relationship among the channel number, alarm flag soft element, and alarm diagnosis code soft element.

Channel Number	Alarm Flag Soft Element	Alarm Diagnosis Code Soft Element
0	SM9031	SD9008
1	SM9081	SD9018
2	SM9131	SD9028
3	SM9181	SD9038
4	SM9231	SD9048
5	SM9381	SD9058
6	SM9331	SD9068
7	SM9381	SD9078

The following table lists the relationship between the diagnosis code and diagnosis information.

Diagnosis Code	Diagnosis Information
1501	A sampling value overflow occurs.

- Axis error

  The following table lists the relationship among the axis number, error flag soft element, and error diagnosis code soft element.

Axis Number	Error Flag Soft Element	Error Diagnosis Code Soft Element
0	SM9405	SD9105
1	SM9425	SD9125
2	SM9445	SD9145
3	SM9465	SD9165

The following table lists the relationship between the diagnosis code and diagnosis information.

Diagnosis Code	Diagnosis Information
2001	Indicates the hardware limit in the forward direction.
2002	Indicates the hardware limit in the reverse direction.
2003	The stop upon startup command is ON.
2004	Indicates the software limit in the forward direction.
2005	Indicates the software limit in the reverse direction.
2006	The running CPU module switches to the Stop state.
2007	Drive module ready is OFF.
2008	Zero signal is ON.
2009	Mechanical homing is not executed.
2010	A retry error occurs.
2011	ABS transmission times out.
2012	Indicates sum of ABS transmission.
2013	A speed 0 error occurs.
2014	The acceleration/deceleration setting times out.
2015	The deceleration stop setting times out.
2016	Movement during velocity/position switchover control is out of range.
2017	Velocity/position switchover is disabled.
2018	The current value is changed when the axis is not in Stop state.
2019	The acceleration/deceleration time is set to 0.
2020	The axis is not stopped upon startup.
2021	An axis stop command is received upon startup.

- Axis alarm

The following table lists the relationship among the axis number, alarm flag soft element, and alarm diagnosis code soft element.

Axis Number	Alarm Flag Soft Element	Alarm Diagnosis Code Soft Element
0	SM9406	SD9106
1	SM9426	SD9126
2	SM9446	SD9146
3	SM9466	SD9166

The following table lists the relationship between the diagnosis code and diagnosis information.

Diagnosis Code	Diagnosis Information
2501	The velocity is out of range.
2502	Target position change is disabled.
2503	Velocity change is disabled.

- Other faults
  Other faults indicate diagnosis of invalid input parameters of the high-speed I/O function block. The diagnosis data cannot be obtained through soft elements. You can check the data on the high-speed I/O self-diagnosis page and the diagnosis information list page. The following table lists diagnosis codes and diagnosis information.

Diagnosis Code	Diagnosis Information
1018	The channel number of the high-speed input function block is invalid.
1019	The input parameter of the high-speed input function block is invalid.
2022	The channel number of the high-speed output function block is invalid.
2023	The input parameter of the high-speed output function block is invalid.

9.8 Diagnosis Code and Diagnosis Information

9.8.1 Overview

Each diagnosis code has a name, which matches the type name of the corresponding diagnosis programming interface. For details, see *" Diagnosis Programming Interface" on page 453*.

9.8.2 CPU Diagnosis Code

Name	Diagnosis Code	Diagnosis Information
SDCardError	1	SD card error
FlashError	1	Flash error
SystemError	0x40	High-speed I/O interface board connection error

Name	Diagnosis Code	Diagnosis Information
InterCommError	0x11	No I/O expansion module (inter-board communication error: read check failure)
	0x12	No I/O expansion module (inter-board communication error: write check failure)
	0x13	No I/O expansion module (inter-board communication error: ACK being high level)
	0x14	No I/O expansion module (inter-board communication error: ACK being low level)
	0x21	Actual number of I/O expansion modules below configured value (inter-board communication error: read check failure)
	0x22	Actual number of I/O expansion modules below configured value (inter-board communication error: write check failure)
	0x23	Actual number of I/O expansion modules below configured value (inter-board communication error: ACK being high level)
	0x24	Actual number of I/O expansion modules below configured value (inter-board communication error: ACK being low level)
	0x31	Actual number of I/O expansion modules above configured value (inter-board communication error: read check failure)
	0x32	Actual number of I/O expansion modules above configured value (inter-board communication error: write check failure)
	0x33	Actual number of I/O expansion modules above configured value (inter-board communication error: ACK being high level)
	0x34	Actual number of I/O expansion modules above configured value (inter-board communication error: ACK being low level)
	0x41	I/O expansion module type error (inter-board communication error: read check failure)
	0x42	I/O expansion module type error (inter-board communication error: write check failure)
	0x43	I/O expansion module type error (inter-board communication error: ACK being high level)
	0x44	I/O expansion module type error (inter-board communication error: ACK being low level)

Name	Diagnosis Code	Diagnosis Information
ConformenceError (Each bit indicates one module fault.)	1	I/O module corresponding to slot 1 inconsistent with actual I/O module configuration
	2	I/O module corresponding to slot 2 inconsistent with actual I/O module configuration
	4	I/O module corresponding to slot 3 inconsistent with actual I/O module configuration
	8	I/O module corresponding to slot 4 inconsistent with actual I/O module configuration
	16	I/O module corresponding to slot 5 inconsistent with actual I/O module configuration
	32	I/O module corresponding to slot 6 inconsistent with actual I/O module configuration
	64	I/O module corresponding to slot 7 inconsistent with actual I/O module configuration
	128	I/O module corresponding to slot 8 inconsistent with actual I/O module configuration
	256	I/O module corresponding to slot 9 inconsistent with actual I/O module configuration
	512	I/O module corresponding to slot 10 inconsistent with actual I/O module configuration
	1024	I/O module corresponding to slot 11 inconsistent with actual I/O module configuration
	2048	I/O module corresponding to slot 12 inconsistent with actual I/O module configuration
	4096	I/O module corresponding to slot 13 inconsistent with actual I/O module configuration
	8192	I/O module corresponding to slot 14 inconsistent with actual I/O module configuration
	16384	I/O module corresponding to slot 15 inconsistent with actual I/O module configuration
	32768	I/O module corresponding to slot 16 inconsistent with actual I/O module configuration
IOModulePosError (Each bit indicates one module fault. As fault information is displayed on the I/O module, the diagnosis information is not displayed but flagged.)	1	I/O module corresponding to slot 1 faulty
	2	I/O module corresponding to slot 2 faulty
	4	I/O module corresponding to slot 3 faulty
	8	I/O module corresponding to slot 4 faulty
	16	I/O module corresponding to slot 5 faulty
	32	I/O module corresponding to slot 6 faulty
	64	I/O module corresponding to slot 7 faulty
	128	I/O module corresponding to slot 8 faulty
	256	I/O module corresponding to slot 9 faulty
	512	I/O module corresponding to slot 10 faulty
	1024	I/O module corresponding to slot 11 faulty
	2048	I/O module corresponding to slot 12 faulty
	4096	I/O module corresponding to slot 13 faulty
	8192	I/O module corresponding to slot 14 faulty
	16384	I/O module corresponding to slot 15 faulty
	32768	I/O module corresponding to slot 16 faulty

Name	Diagnosis Code	Diagnosis Information
FunctionErrorCode (Each bit indicates one bus fault, which is only flagged.)	0x01	DP bus faulty
	0x02	EtherCAT bus faulty
	0x04	CANopen bus faulty
	0x08	CANlink bus faulty
	0x10	Modbus TCP faulty
	0x20	Modbus serial port 0 faulty
	0x40	Modbus serial port 1 faulty
	0x80	High-speed I/O faulty

Note

As EtherCAT is implemented through CODESYS, the PLC cannot obtain EtherCAT diagnosis information directly, and EtherCAT bus flags are invalid currently.

9.8.3 I/O Module Diagnosis Code

Name	Module Type	Diagnosis Code	Diagnosis Information
BaseInfo	All	64	System shut down upon fault
ModuleError (Each bit indicates one module fault.)	AI	2	No external load voltage
		4	Analog chip connection error
	AO	2	No external load voltage
		4	Analog chip connection error
		8	Analog chip overheated
ChannelError[i] (Each array element indicates one channel diagnosis code, and each bit indicates one fault.)	AI	2	Overflow
		4	Underflow
		8	Above the upper limit
		16	Below the lower limit
		32	Disconnected
	AO	2	Overflow
		4	Underflow
		8	Current disconnected
		16	Voltage short-circuited
		32	Digital to Analog Converter (DAC) channel hardware fault

9.8.4 PROFIBUS-DP Diagnosis Code

Name	Diagnosis Code	Diagnosis Information
ExtDiagData[0] (Each bit indicates one fault.)	0x02	Unready to exchange data
	0x04	Incorrect configuration
	0x08	Expanded diagnosis information existing on the slave
	0x10	The requested function not supported by the slave
	0x20	Invalid slave response
	0x40	Incorrect parameter
	0x80	Locked by different masters
ExtDiagData[1] (Each bit indicates one fault.)	0x01	Resetting the parameter
	0x02	A static diagnosis
	0x08	Activated watchdog monitoring
	0x10	Processing slave data in latch mode
	0x20	Processing slave data in synchronous mode
	0x80	The slave not activated by the master
ExtDiagData[2] (Each bit indicates one fault.)	0x80	Diagnosis data overflow
ExtDiagData[3] (master address)	-	-
ExtDiagData[4] and ExtDiagData[5] (slave ID)	-	-

Note

The first 6 bytes correspond to the basic diagnosis, and the following diagnosis data bits correspond to the expanded diagnosis. For details, see "DP Diagnosis".

9.8.5 CANlink Diagnosis Code

Name	Diagnosis Code	Diagnosis Information
CmdFrameError (diagnosis code: remainder when divided by 100)	1	Invalid command code
	2	Abnormal command code address
	3	Value out of allowable range
	4	Invalid command code operation
	5	Invalid command code length
	6	Command code timeout
CfgFrameError (diagnosis code: remainder when divided by 100)	1	Incorrect configuration code
	2	Incorrect configuration index
	3	Incorrect configuration information
	5	Incorrect configuration data length
	6	Configuration frames failing to respond

9.8.6 Modbus Diagnosis Code

Name	Diagnosis Code	Diagnosis Information
DiagData	0x70	Incorrect Modbus slave address
	0x71	Incorrect data frame length, serial port 0 (COM0)
	0x72	Invalid data address
	0x73	CRC error
	0x74	Unsupported command code
DiagData	0x75	Reception timeout
	0x76	Invalid data value
	0x77	Buffer overflow
	0x78	Frame error
	0x79	Serial port protocol error
	0x7C	Address error
	0x7D	No data received
	0x7E	Incorrect data returned by the slave
	0x80	Incorrect Modbus slave address
	0x81	Incorrect data frame length, serial port 1 (COM1)
	0x82	Invalid data address
	0x83	CRC error
	0x84	Unsupported command code
	0x85	Reception timeout
	0x86	Invalid data value
	0x87	Buffer overflow
	0x88	Frame error
	0x89	Serial port protocol error
	0x8C	Address error
	0x8D	No data received
	0x8E	Incorrect data returned by the slave
	0x90	Incorrect Modbus slave address
	0x91	Incorrect data frame length, Ethernet (Modbus TCP)
	0x92	Invalid data address
	0x93	CRC error
	0x94	Unsupported command code
	0x95	Reception timeout
	0x96	Invalid data value
	0x97	Buffer overflow
	0x98	Frame error
	0x99	Serial port protocol error
	0x9A	Slave not connected
	0x9B	Incorrect protocol identifier
	0x9C	Address error
	0x9D	No data received
	0x9E	Incorrect data returned by the slave
	0x9F	Number of connected clients out of range
	0xA0	Invalid data value

9.8.7 EtherCAT Diagnosis Code

EtherCAT fault IDs are divided into EtherCAT bus fault IDs and EtherCAT slave fault IDs. The EtherCAT bus fault IDs describe master and slave faults with the variable "m_LastError", while the EtherCAT slave fault IDs describe the slave faults with the variable "ErrorCode".

Name	Diagnosis Code	Diagnosis Information
m_ LastError	0x1	Abnormal master communication, with loss of more than 100 consecutive frames of data
	0x2	Some slaves disconnected, number of online slaves inconsistent with configured value
	0x3	Abnormal DC clock, reference clock remaining unchanged
	0x4	Failed to open NIC
	0x5	Failed to open redundant NIC
	0x6	Failed to open redundant NIC. Same NIC configured for the redundancy function
	0x7	Slave initialization error because the slave does not exist during startup or communication cannot be established
	0x8	Configured vendor ID mismatching with the actual one
	0x9	Configured product ID mismatching with the actual one, or master failed to read the product ID from this slave
	0xA	Configured number of slaves greater than the actual number
	0xB	Failed to download SDO
	0xC	SDO download timeout
	0xD	Slave emergency event error
	0xE	Failed to download SOE
	0xF	SOE download timeout
	0x10	State machine request from master timeout
	0x20	Alias address conflicting because an alias address is assigned to multiple slaves in the network
	0x21	Slave IN/OUT connection error
	0x22	The master failed to access EEPROM of the slave during startup
	0x30	Continuous frame loss
	0x31	Slave port link disconnected
	0x32	Occasional loss frame warning
	0x64	Failed to switch communication state
	0x65	Unknown slave error
	0x66	Failed to request slave memory for the mailbox
	0x6A	Slave firmware version inconsistent with that on EEPROM
	0x6B	The slave failed to update the firmware
	0x75	State machine error
	0x76	Unknown state change request received by the slave
	0x77	State machine error because the slave does not support the boot mode
	0x78	Invalid firmware program
	0x79	Invalid mailbox configuration in the slave booth state
	0x7A	Invalid mailbox configuration in the slave pre-operational state
	0x7B	Invalid sync manager configuration detected by the slave
	0x7C	Invalid input data
	0x7D	Invalid output data
	0x7E	Synchronization error
	0x7F	Sync manager watchdog timeout
	0x80	Invalid synchronization type
	0x81	Invalid output PDO configuration
	0x82	Invalid input PDO configuration
	0x83	Invalid watchdog configuration
	0x84	Starting the slave in cold mode
	0x85	Initializing the slave
	0x86	Pre-operating the slave
	0x87	Operating the slave securely
	0x88	Invalid input mapping because the slave does not support input PDO

Name	Diagnosis Code	Diagnosis Information
m_ LastError	0x89	Invalid output mapping because the slave does not support output PDO configuration
	0x8A	Inconsistent slave settings
	0x8B	Incorrect mode because the slave does not support the free mode
	0x8C	Incorrect mode because the slave does not support the synchronous mode
	0x8E	Incorrect parameter configuration because the slave needs to run in three buffer modes in free mode
	0x8F	Invalid input and output
	0x90	DC synchronization error. The Sync0 watchdog times out in the slave DC mode.
	0x91	DC synchronization error. No Sync0 interrupt signal is detected when the slave mode is switched from safe mode to running mode.
	0x92	DC synchronization error. The slave synchronization period is too small for the slave address ({Addr}).
	0x94	Invalid DC synchronization configuration
	0x95	Invalid DC latch configuration
	0x96	PLL error
	0x97	Invalid DC
	0x98	DC timeout
	0x99	Invalid synchronization cycle period
	0x9A	Incorrect Sync0 configuration. The slave Sync0 period is out of the range.
	0x9B	Incorrect Sync1 configuration. The slave Sync1 period is out of the range.
	0xA5	Slave MBX_AOE error
	0xA6	Slave MBX_EOE error
	0xA7	Slave MBX_COE error
	0xA8	Slave MBX_FOE error
	0xA9	Slave MBX_SOE error
	0xB3	Slave MBX_VOE error
	0xB4	Failed to access the EEPROM address
	0xB5	Slave EEPROM error
	0xB6	External slave hardware unready
	0xC4	Slave already restarted locally
	0xD4	Inconsistency between the configuration of the slave coupler mounted module and the actual configuration
ErrorCode	0X08	Mismatching vendor ID
	0X09	Mismatching product
	0X20	Alias address conflict
	0X21	IN/OUT reversing fault
	0X22	EEPROM access failure
	0X30	Continuous frame loss
	0X31	Slave OUT port link disconnected
	0X32	Occasional frame loss
	0X64	Failed to switch communication state

9.8.8 Axis Diagnosis Code

Diagnosis Code	Diagnosis Information
0x1	Drive bus communication fault
0x2	Drive fault
0x3	Bus DC synchronization loss
0xA	Software limit exceeded
0xB	Hardware limit exceeded
0xC	Axis position out of the maximum allowable range in linear mode
0xD	Fast emergency stop or pause function not supported by drive
0xE	None
0xF	None
0x10	Deviation between target position and actual position out of the lag limit
0x11	Drive homing fault
0x12	None
0x14	Motion control function block executed while the axis is not enabled
0x15	Function block execution failed because an unsupported mode is detected
0x19	Logic axis operation not supported by the function block
0x1B	None
0x1E	Function block in execution not called during motion
0x1F	Incorrect input parameter axis type of the function block
0x20	Axis instance changed during function block execution
0x21	Interrupt enabled during function block execution
0x22	Axis state not meeting the PLCopen state machine requirements when the function block is triggered
0x23	Drive failed during axis moving
0x28	Virtual axis velocity exceeding the limit
0x29	Virtual axis acceleration rate exceeding the limit
0x2A	Virtual axis deceleration rate exceeding the limit
0x32	Incorrect input homing parameter of host controller
0x33	No hardware limit configured for host controller homing
0x3C	File read-write cache channel used up and handle that can be registered being empty
0x41	SDO multi-channel communication initialization failed because no activated app is obtained
0x42	Invalid IEC task handle
0x43	Too many tasks in the SDO multi-channel
0x44	SDO multi-channel underlying interface call error
0x45	None
0x46	Current control mode not supported by the function block or drive
0x47	Control mode failed to be changed because the axis is in a specific state when the function block is triggered
0x48	Control mode switchover interrupted
0x4B	Instance failed to run because the current control mode is not synchronous torque
0x50	Function block failed to reset the axis
0x51	Initialization failed when the function block resets the axis
0x55	Incorrect axis type input by the function block
0x56	Invalid input parameter of the function block

Diagnosis Code	Diagnosis Information
0x5A	Input parameter of the function block being zero
0x5B	Function block execution failed because the drive is enabled
0x5C	Invalid cycle of the rotary axis
0x5D	Rotation position cycle not an integer
0x6E	Task cycle time set to 0
0x78	No function block error that can be reset
0x79	No response to the drive reset request
0x7A	Drive fault failed to be reset
0x7B	Drive response timed out during reset
0x7C	Bus communication error failed to be reset
0x82	Unknown parameter input by the function block
0x83	An error occurred when the function block reads the drive parameters
0x84	Input parameter of the function block not in the mapping table
0x85	Internal data conversion error of the function block
0x8C	Unknown parameter input by the function block
0x8D	An error occurred when the function block writes the drive parameters
0x8E	Input parameter of the function block not in the mapping table
0x8F	Internal data conversion error of the function block
0xAA	Axis not in the standstill state
0xAB	Homing instruction failed to write parameters
0xAC	No response when the homing instruction reads parameters
0xAD	None
0xAE	Axis in the ErrorStop state during homing
0xB4	Stop function block aborted during stop
0xB5	Invalid input deceleration rate of the stop function block
0xB6	Unavailable when the direction is shortest
0xB7	Axis in the ErrorStop state
0xB8	Stop function block being executed not called in the bus cycle
0xB9	Function block execution in the Stopping state
0xC8	Task cycle time set to 0
0xC9	Invalid input velocity acceleration rate of the function block
0xCA	Invalid direction parameter of the function block
0xE2	Invalid input velocity acceleration rate of the function block
0xE3	Invalid direction parameter of the function block
0xFB	Invalid input velocity acceleration rate of the function block
0xFC	Invalid direction parameter of the function block
0x114	Invalid input velocity acceleration rate of the function block
0x115	Invalid direction parameter of the function block
0x116	Invalid execution sequence of the function block
0x12C	None
0x12D	Invalid input velocity acceleration rate of the function block
0x12E	Input direction parameter of the function block not supported
0x145	Invalid input ArraySize of the function block
0x146	Invalid input time parameter of the function block
0x15E	Invalid input ArraySize of the function block
0x15F	Invalid input time parameter of the function block

Diagnosis Code	Diagnosis Information
0x177	Invalid input ArraySize of the function block
0x178	Invalid input time parameter of the function block
0x190	Probe channel being occupied
0x191	The window probe function not supported by the drive
0x192	Probe communication error
0x19A	Probe cannot be terminated
0x1AA	Invalid input velocity acceleration rate of the function block
0x1AB	Invalid input direction parameter of the function block
0x1C3	Invalid input velocity acceleration rate of the function block
0x1C4	Invalid input direction parameter of the function block
0x1C5	Input direction parameter of the function block not supported
0x1DB	Function block execution must be in the standstill or power_off state
0x1DC	Invalid input velocity acceleration rate of the function block
0x258	No tappet in the cam table
0x259	The tappet number set to a too large value
0x25A	Over 32 activated tappet operations in one cam table
0x271	Cam table being empty
0x272	The master axis location in the cam table exceeding the cam cycle range
0x273	No velocity and acceleration rate input for cam dynamic coupling
0x274	Cam key point exceeding the range
0x275	To many tappets activated in one cycle
0x280	Input cam type of the function block not supported
0x2A3	Input gear ratio denominator being 0
0x2A4	Invalid input acceleration rate
0x2A5	Invalid input deceleration rate
0x2A6	Master axis enable state changed
0x2A7	Invalid input parameter Jerk of the function block
0x2D5	Invalid input velocity acceleration rate of the function block
0x2D6	Axis rotation cycle being zero
0x2EE	Input non-cam table structure type parameter
0x2EF	Cam table key point not in the master range
0x2F0	Master axis start value of the cam table greater than the end value
0x2F1	Invalid master axis location in the cam table
0x2F2	Invalid slave axis location in the cam table
0x307	Master axis direction changed
0x308	Reversion of the slave axis unable to be avoided
0x309	Input parameter of the function block not supporting linear axis
0x30A	The master start distance in the Buffered mode must be 0
0x30B	Motion synchronization unable to be enabled
0x320	Too large compensation clearance
0x339	Path generation internal error due to algorithm convergence failure
0x33A	Path generation internal error due to invalid parameter value
0x33B	Path generation internal error due to no results for axis calculations
0x33C	Path generation internal error due to decrease in the duration of the lower limit

Diagnosis Code	Diagnosis Information
0x33D	Path generation internal error due to no common duration detected by related axes
0x33E	Path generation internal error due to invalid result interval
0x33F	More phase parameters required to generate S-shaped velocity profile
0x340	Path generation internal error
0x352	Function block execution axis not in the standstill or power_off state
0x353	Invalid input parameters of the function block
0x354	Function block execution axis not in the standstill or power_off state
0x355	Invalid input position mode and cycle of the function block
0x356	Function block input axis not a virtual axis
0x79F	Previous motion in motion cache not supporting Blending
0x7A0	Previous motion in motion cache not supporting BufferMode
0x7A1	Previous cache instruction not activated and called in the current cycle
0x7A2	Axis motion not taken over by any function block
0x7A3	Unsupported BufferMode parameter input
0x7A4	An error occurred in the previous motion in the motion cache
0x7A5	Function block instance failed to be added again because it is already in the motion cache queue
0x4E20	Drive DC clock not synchronized
0x4E21	Drive not in OP mode
0x4E22	Failed to wait for DC synchronization during startup
0x4E23	Drive scaling ratio configured to 0
0x4E24	SMC_Basic version too low
0x4E25	No SoftMotion license
0x4E26	No activated app detected
0x4E27	Axis device disabled
0x4E28	No axis device found
0x4E29	Current axis and logical axis not in the same task
0x4E2A	Failed to get axis background configuration parameters
0x4E2B	Invalid device type
0x4E2C	Task cycle set too large
0x4E2D	Failed to get axis mapping parameters
0x5015	No response from the slave when the torque limit instruction reads/writes parameters from/to the servo
0x5016	Timeout occurred when the torque limit instruction reads/writes parameters from/to the servo
0x501F	No response from the slave when reading the digit input
0x5020	Digital input reading timed out
0x5028	Failed to change from the torque control mode to the position control mode
0x5032	Input slope or velocity exceeding the allowable range during torque instruction execution
0x5033	Runaway occurred in torque mode
0x5034	Invalid superposition in torque mode
0x5035	No maximum profile velocity PDO configured in the torque instruction
0x5036	Desired torque PDO not configured in torque instruction
0x503C	Axis state error during deviation reset instruction execution

Diagnosis Code	Diagnosis Information
0x503D	Deviation reset instruction unable to be executed in stopping state
0x503E	Deviation reset instruction execution interrupted
0x5046	Failed to change from the torque control mode to the position control mode
0x5050	Incorrect abort superposition instruction execution sequence, or no superposition instruction in the program
0x5051	Invalid input variable in the abort superposition instruction
0x505A	Insufficient probe PDO
0x505B	Exceeding the interrupt positioning instruction cache mode range
0x505C	Upper boundary of the interrupt positioning instruction greater than the lower boundary in linear mode
0x505D	Probe already occupied
0x505E	An error occurred during absolute, relative, or speed control
0x505F	An error occurred during interrupt positioning
0x5060	Probe not triggered
0x5061	Difference between upper and lower boundaries of the interrupt positioning instruction in linear mode exceeding the rotation cycle
0x5062	Invalid input feed distance
0x5063	Invalid input variable in the interrupt positioning instruction
0x5064	Master axis in the incorrect state machine
0x5065	Slave axis in the incorrect state machine
0x5066	Master axis in the incorrect control mode
0x5067	Slave axis in the incorrect control mode
0x5068	The sum of acceleration and deceleration segments cannot be greater than 1
0x5069	Acceleration/deceleration input exceeding the limit
0x506A	Unreasonable master axis displacement input
0x506B	Unreasonable slave axis displacement input
0x506C	Waiting not allowed after repeated triggering
0x506D	Master axis motion direction changed during superposition motion
0x506E	Incorrect slave axis superposition displacement direction in the synchronization superposition instruction
0x506F	The curve type must not exceed the selectable range
0x5078	Single-axis instructions not allowed for a single axis in an axis group
0x754E	SDO read/write error
0x754F	SDO read/write error

9.9 Synchronizing the Project Information

9.9.1 Overview

You can synchronize the project data with the downloaded data to ensure accurate login to the PLC.

Project information synchronization is used to solve the following unexpected download problems:

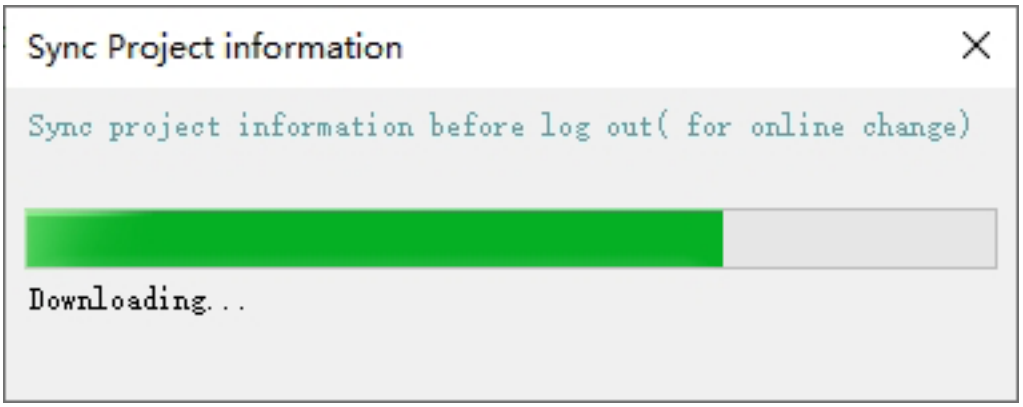
- Only the project file is copied.
- One project debugs multiple PLCs.

- The programming and debugging operations are performed by multiple people.
- The project data must be downloaded before clearing all the project data.

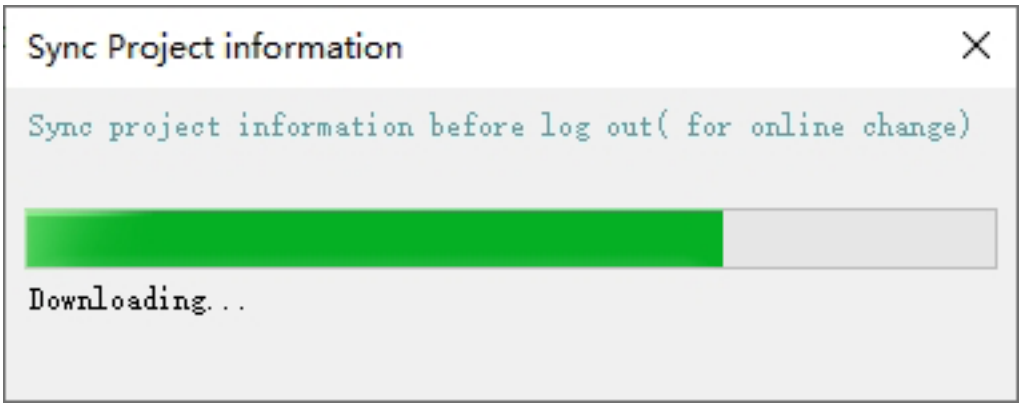
9.9.2 Synchronizing the Downloaded Project Information Automatically

Choose "Project" > "Project Settings" > "Download" > "Download project information", and download the user program. Then, the system automatically starts to synchronize the project information, during which, the status bar "Sync Project information" flashes in orange.

Double-click the status bar "Sync Project information". The current synchronization process is displayed, as shown in the following figure.



If you log out before project information synchronization is completed, the project information synchronization state box is displayed, as shown in the following figure.

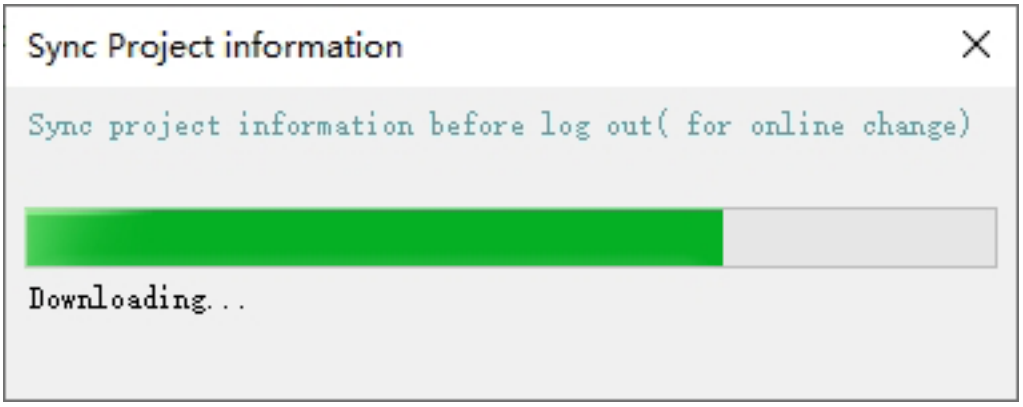


- After synchronization is completed, the "Sync Project information" page is automatically closed and login is exited.
- If you click "X", the page is closed but logout is canceled.
- If you click "Cancel sync", the synchronization project is interrupted, the state page is closed, and the system is logged out.
- If you click "Disable sync", the "Sync Project information" function of the current project is disabled, the state page is closed, and the system is logged out.

9.9.3 Synchronizing the Downloaded Project Information Manually

Choose "Online" > "Sync Project information" to synchronize the project information. This command can be executed only after you have logged into the PLC.

After this command is executed, the synchronization state is displayed, as shown in the following figure.



In the manual mode, the project information is automatically downloaded no matter whether the "Sync Project information" function is enabled for the project.

9.9.4 Special Notes on Synchronizing the Project Information

- If the CPU load ratio is greater than 85% during normal running, it may take a long time to synchronize the project information.
- If you want to debug a single PLC multiple times, disable the synchronization function first and then manually synchronize the project information after debugging.

9.10 SVN Function

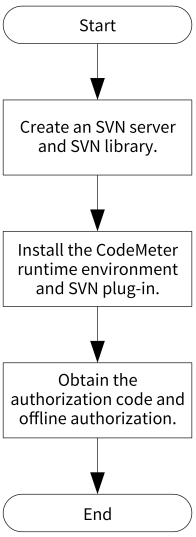
9.10.1 Overview

Caution

This function is supported by InoProShop V1.8.0.0 and later versions.

InoProShop allows you to add project files to SVN libraries for management, such as importing project files to SVN libraries or exporting project files from SVN libraries to InoProShop.

Before using the SVN function, you need to set up the SVN environment according to the following installation and configuration procedure.

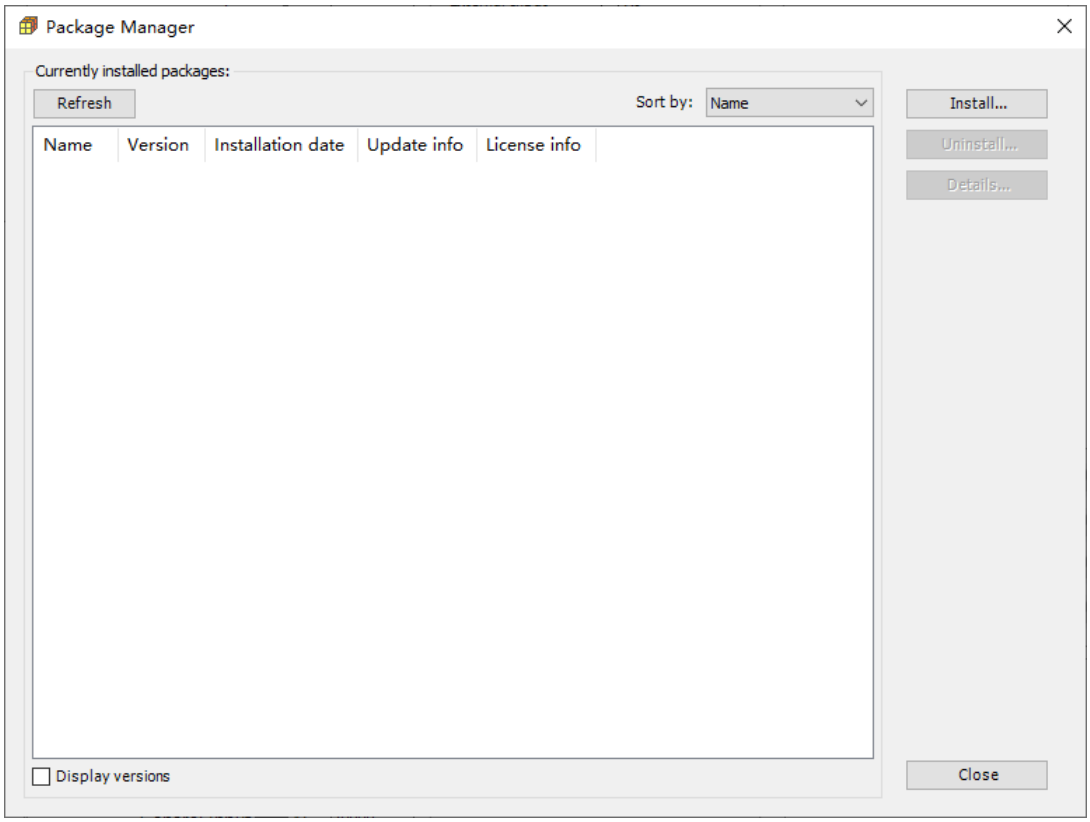


9.10.2 Creation of an SVN Server and SVN Library

For how to create an SVN server (VisualSVN Server) and SVN library (SVN Repository), access the *CODESYS official website* and download the *CODESYS_SVN* guide or contact the company's IT department for support.

9.10.3 Installation of the CodeMeter Runtime Environment and SVN Plug-in

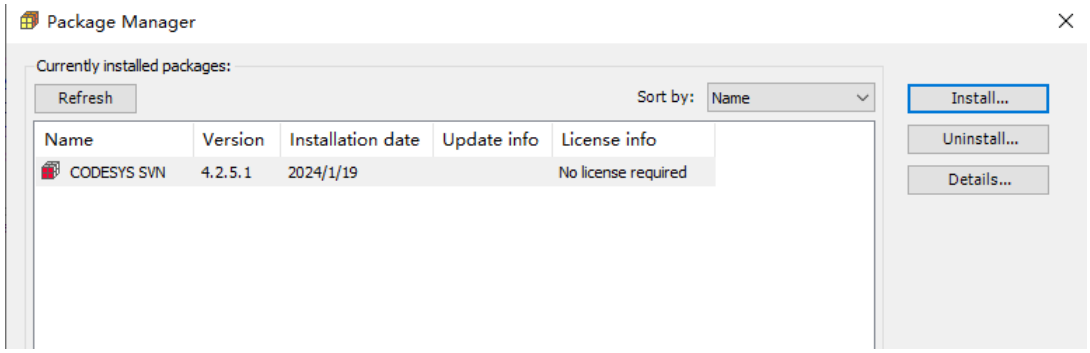
1. Install the CodeMeter runtime environment.
   a. Unzip the InoProShop installation package on your local PC, double-click the "InstallCodeMeterRuntime.bat" batch file in the "CodeMeterRuntime" folder. The program will automatically recognize the local PC system type (32-bit or 64-bit) and install it silently.
   b. Double-click the "ImportLicenses.bat" batch file in the "CodeMeter_WiBu" folder to open the "CodeMeter Control Center" page.
   c. Drag the "3S-Smart_Software_Solutions_Softlicenses.wbb", "CmFirm.wbc", "Patch_Protection_Only.wbb", and "Patch_ProtectionUpdateFile.WibuCmRaU" files from the "CodeMeter_WiBu" folder to the "License" tab page for license.
2. Install the SVN plug-in.
   a. In the menu bar, choose "Tools" > "Package Manager".



b. On the "Package Manager" page displayed, click "Install". In the dialog box displayed, select the InoProShop installation package, unzip it, select the SVN plug-in package "CODESYS SVN 4.2.5.1. package" from the folder, and then click "Open".

c. In the wizard dialog box displayed, select "Complete setup" and "CODESYS V3.5 SP11 Pathc 1" and follow the wizard prompts.

After the plug-in is installed, the plug-in package information is displayed in the "currently installed packages" list.



d. Close InoProShop.

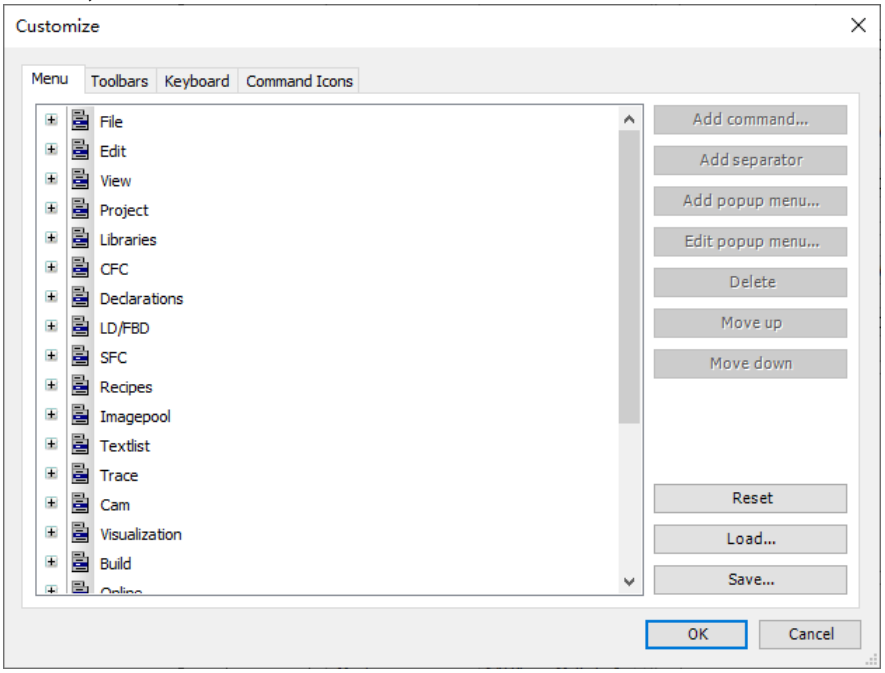
e. In the InoProShop installation directory "..\Inovance Control\InoProShop1.8.0.0\CODESYS \Common", double-click "InstallSVN.bat" batch file, and configure the SVN plug-in. After successful execution, the following figure is displayed.



f. Close the batch command window and re-open InoProShop.

g. Customize the SVN function through the menu bar.

1). In the menu bar, choose "Tools" > "Customize".



2). On the "Customize" page displayed, click "Load". In the dialog box displayed, select the InoProShop installation package, unzip it, and then select the "Menu tools bar SVN configuration.opt.menu" from the folder.

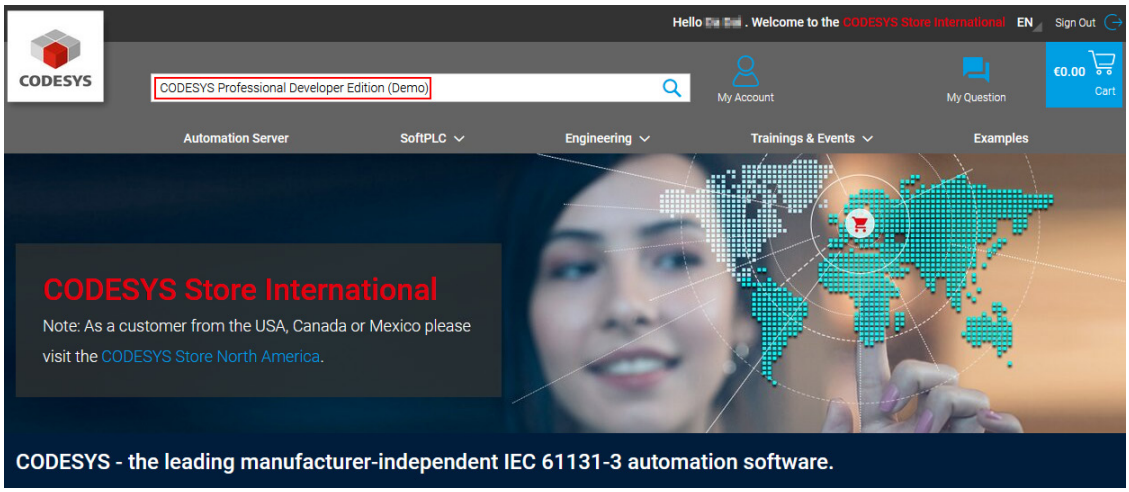
9.10.4 Acquisition of the Authorization Code and Offline Authorization

Before using the SVN function, you must get the authorization code and perform offline authorization.

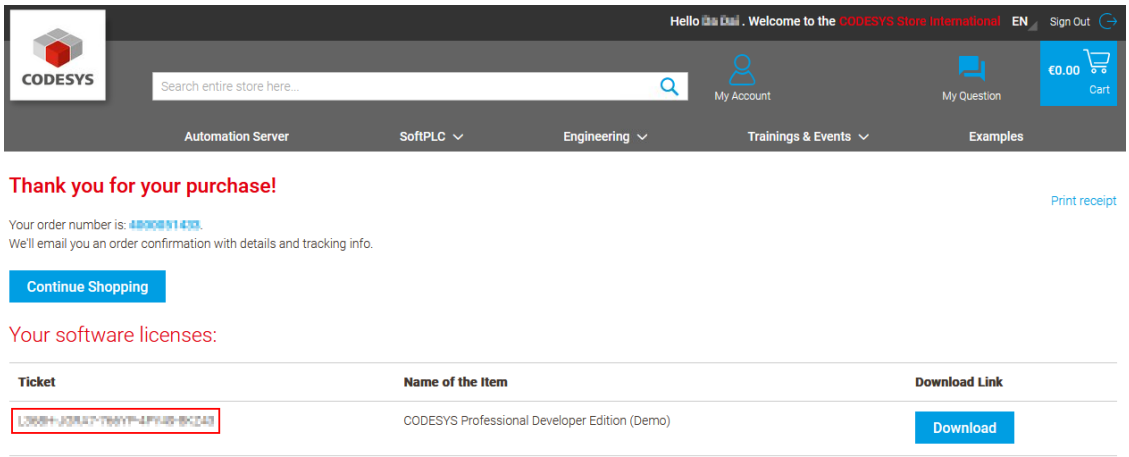
Get the authorization code

The following describes how to get the authorization code of "CODESYS Professional Developer Edition (Demo)", which is valid for 30 days. For longer license days, please purchase and get the authorization code of "CODESYS Professional Developer Edition".

1. Log in to the *CODESYS Store* and create an account.
2. On the homepage of the CODESYS Store, enter "CODESYS Professional Developer Edition (Demo)" in the search box, add it to your shopping cart, and place an order.

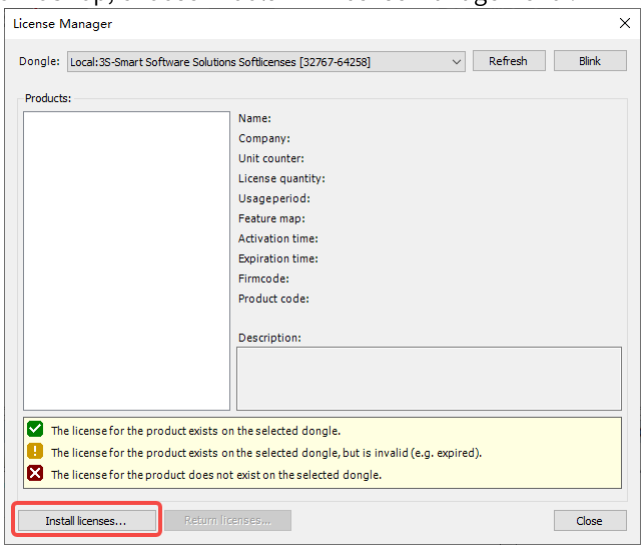


The authorization code is displayed on the page after the order is placed.

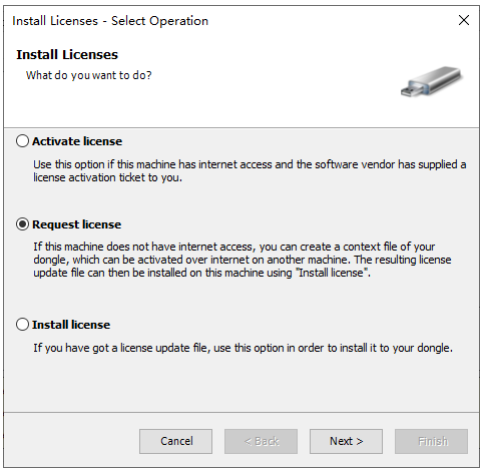


Offline authorization

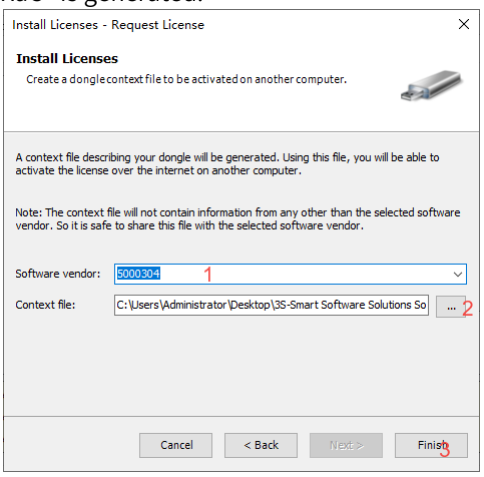
1. In the menu bar of InoProShop, choose "Tools" > "License Management".



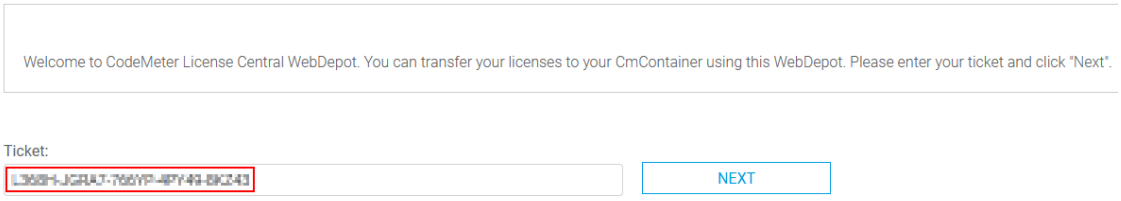
2. On the "License Manager" page displayed, click "Install Licenses". On the page displayed, select "Request license" and then click "Next".



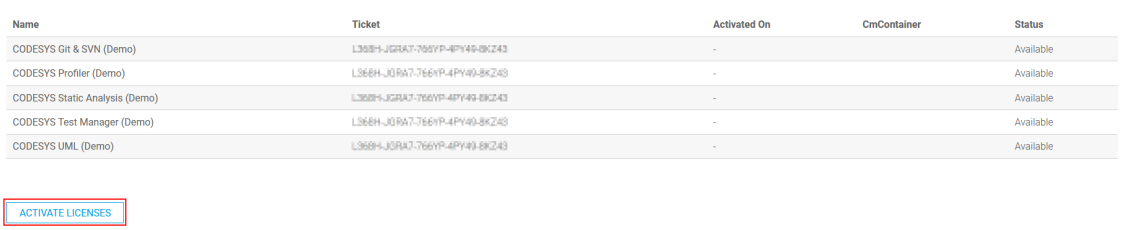
3. Select "5000304" from the drop-down list of "Software vendor". Click "..." next to "Context file", select the offline authorization file generation directory, and click "Finish". The offline request authorization file "*.WibuCmRaC" is generated.



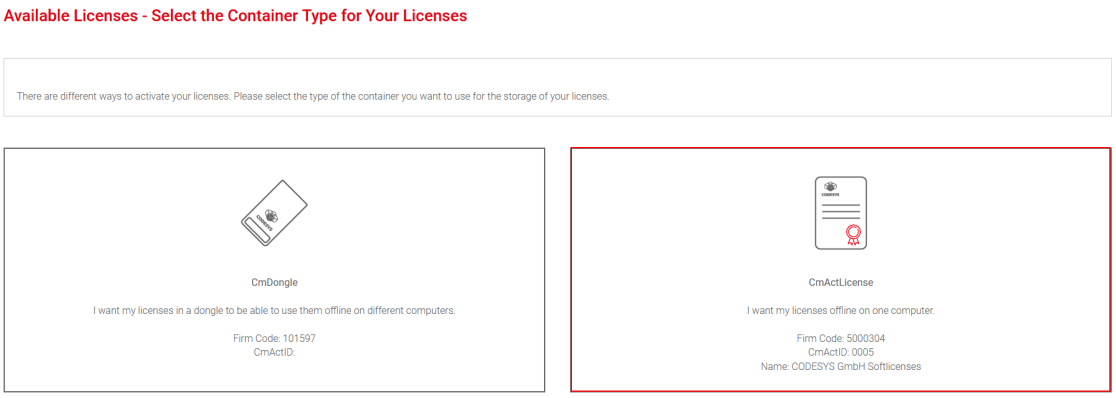
4. Log in to the *CODESYS License Center*, input the obtained authorization code, and click "NEXT".



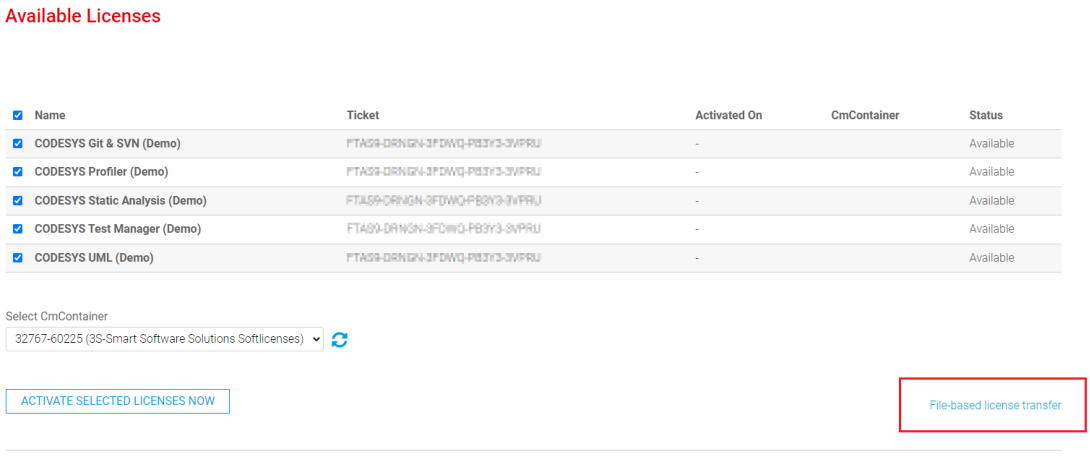
5. Click "ACTIVATE LICENSES" to activate the license.



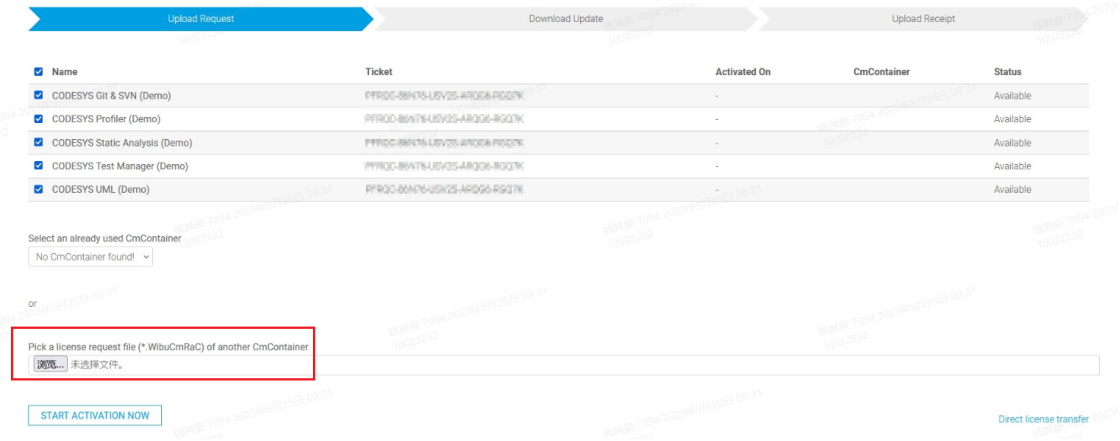
6. Select the license container type "CmActLicense" and authorize it on the same PC.



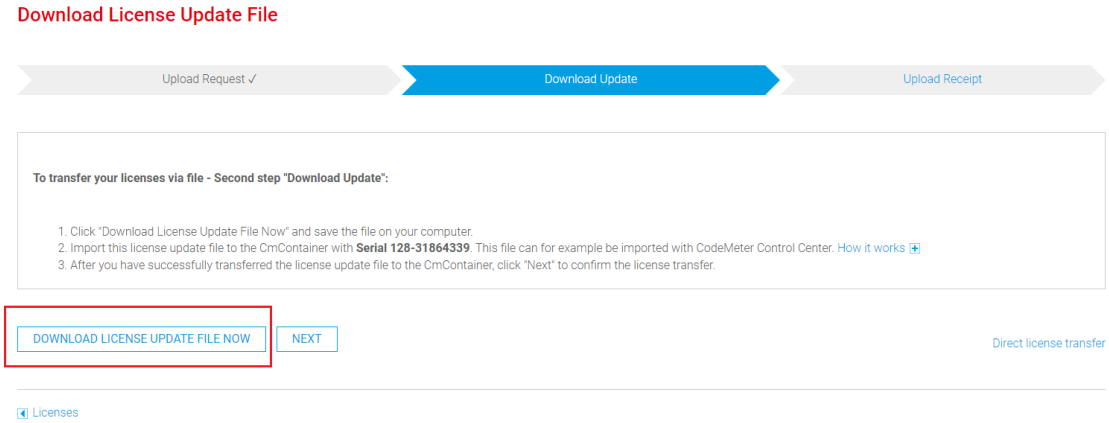
7. Select "File-based license transfer".



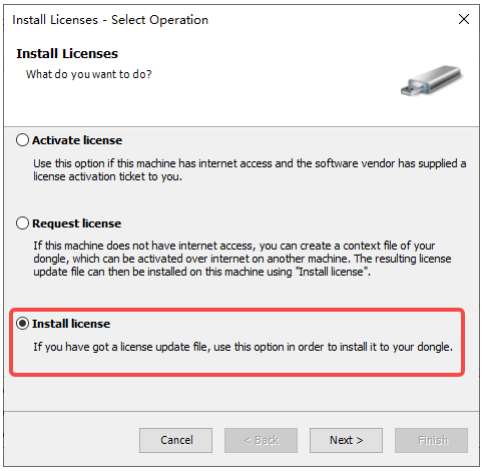
8. Click "Browse..." and select the offline request authorization file "*.WibuCmRaC" generated in step 3.



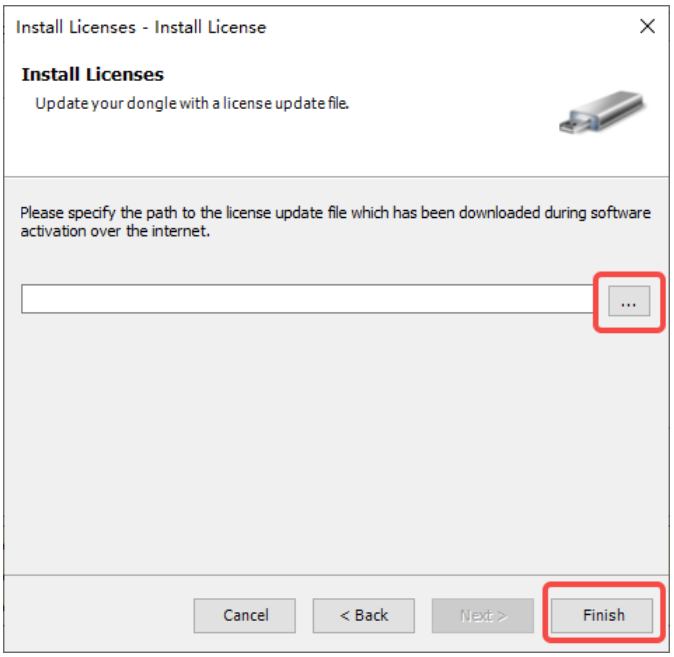
9. Click "DOWNLOAD LICENSE UPDATE FILE NOW" to download the generated offline request authorization file "*.WibuCmRaU" and save the file to a local PC.



10. Repeat step 1, on the page displayed, click "Install Licenses". On the page displayed, select "Request license" and then click "Next".



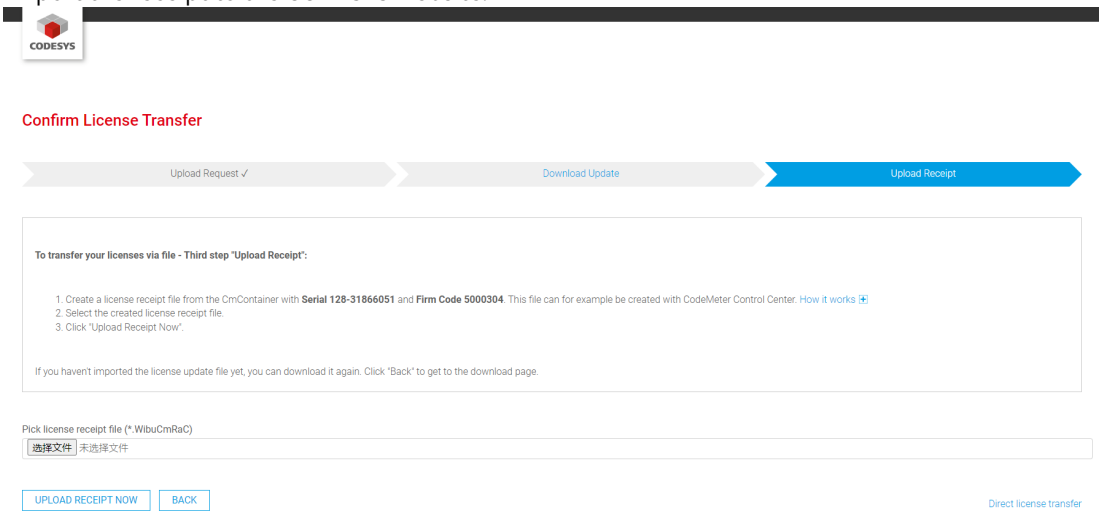
11. Click "...". In the dialog box displayed, select the offline request authorization file "*.WibuCmRaU" generated in step 9 and then click "Finish".



12. (Optional) You can create and upload receipts to the licensing company as needed.

a. On the "CodeMeter Control Center" page, click "License Updates" and select "Create Receipt".

b. Import the receipt to the CODESYS website.



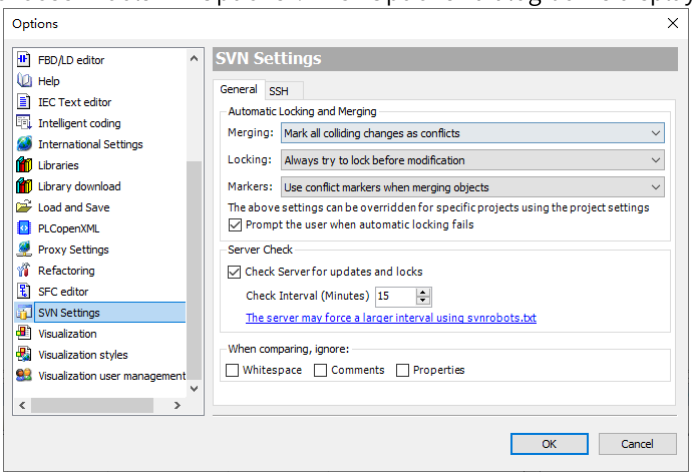
9.10.5 SVN Operator Guidance

This section describes how to operate common SVN functions. For operations of more functions, visit the *CODESYS official website* and download the *CODESYS_SVN* guide.

Set the SVN

- **Set SVN settings in "Options"**

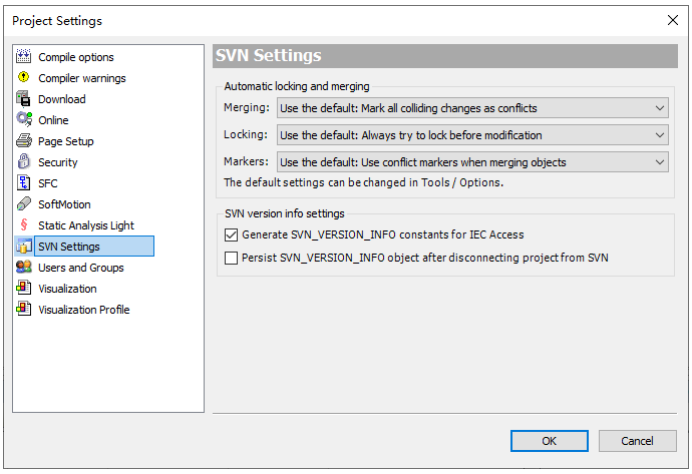
  1. In the menu bar, choose "Tools" > "Options". The "Options" dialog box is displayed.



  2. Click "SVN Settings". For detailed settings, see the *CODESYS_SVN* guide.

- **Set SVN settings in "Project Settings"**

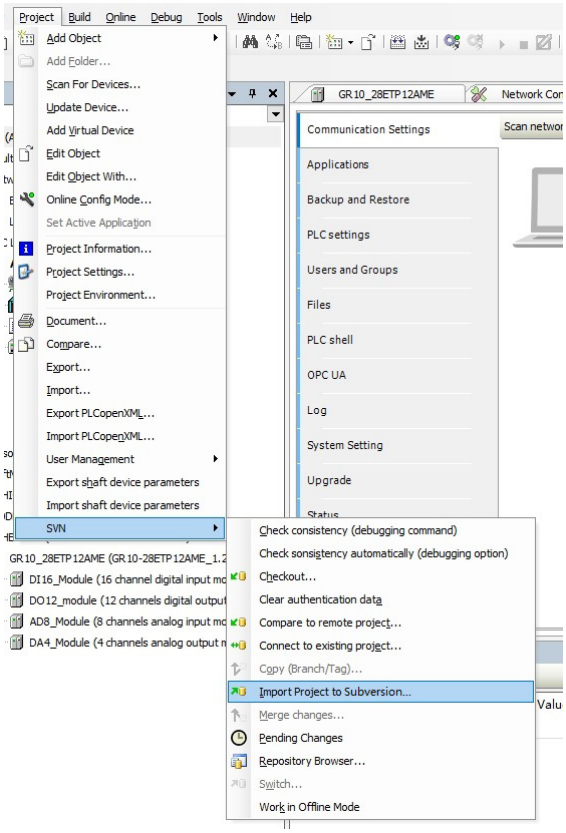
  1. In the toolbar, choose "File" > "Page Settings". The "Project Settings" dialog box is displayed.



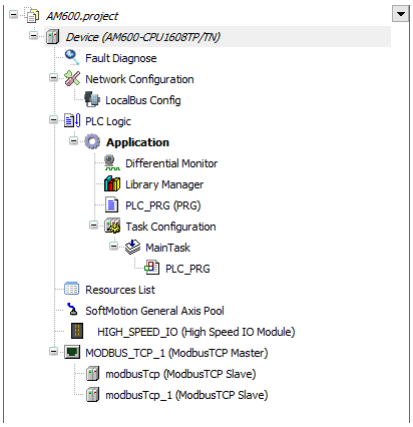
2. Click "SVN Settings". For detailed settings, see the *CODESYS_SVN* guide.

Import a project to an SVN library branch

In the menu bar, choose "Project" > "SVN" > "Import Project to Subversion" to import a project to an SVN library branch.

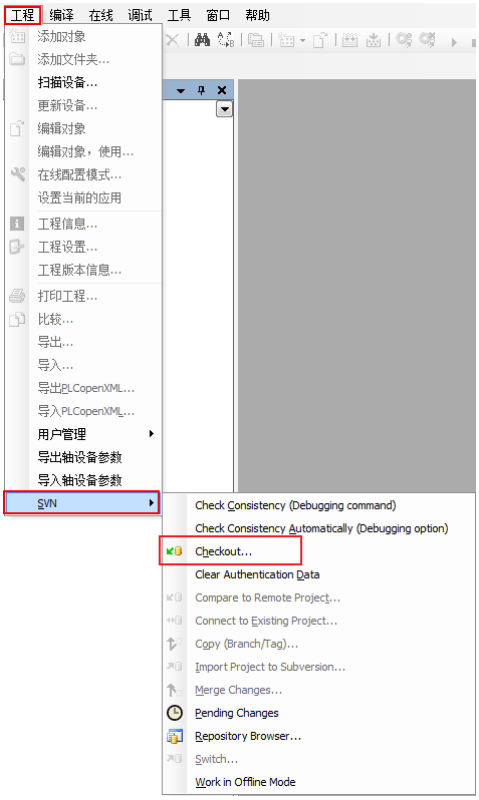


After the project is imported to the SVN library branch, the SVN tab is displayed in the device tree.

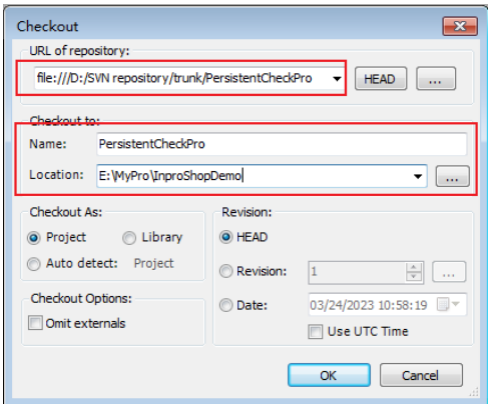


Export a project from an SVN library branch to a local PC

1. In the menu bar, choose "Project" > "SVN" > "Checkout" to export a project from an SVN library branch to a local PC.

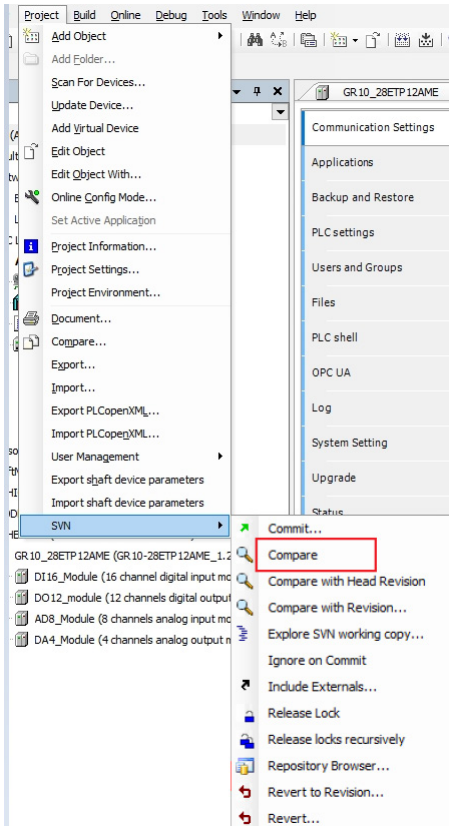


2. In the dialog box displayed, set the SVN library branch path, the name and path of the local folder in which you want to save the project information, and then click "OK".



Compare the local project file with the SVN project file

In the device tree, right-click the project file to be compared. In the shortcut menu displayed, choose "SVN" > "Compare" to compare the local project file with the SVN project file.

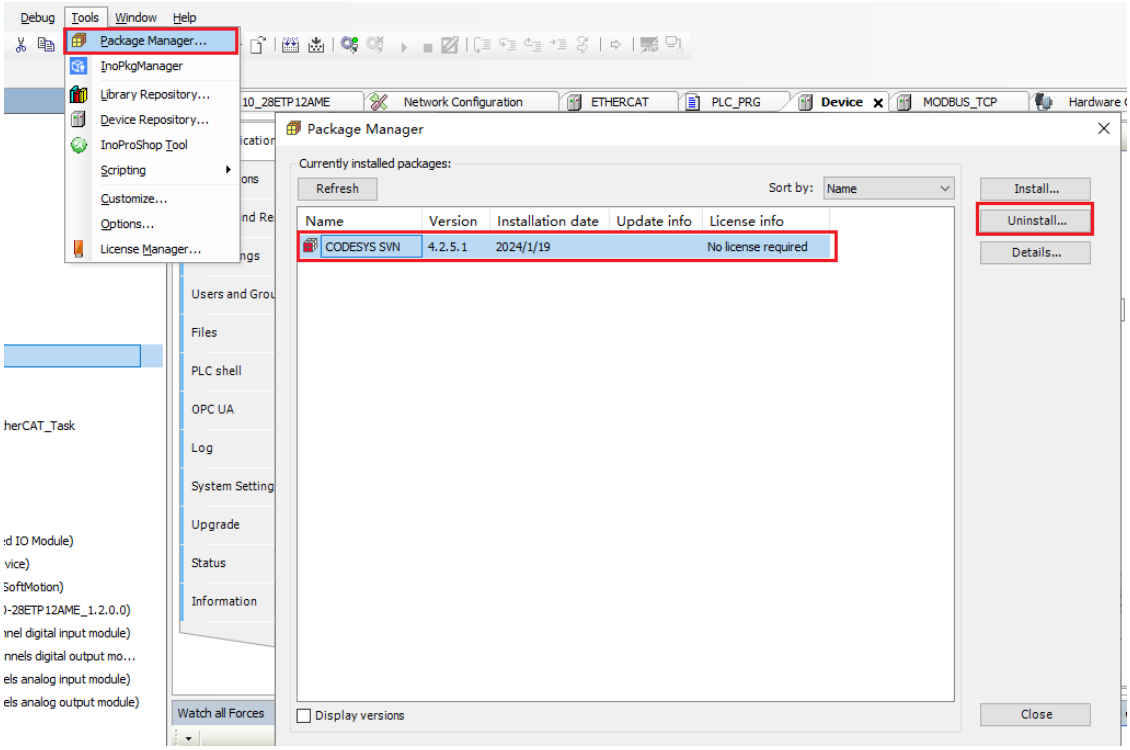


9.10.6 Uninstallation of the SVN Plug-in

1. Close InoProShop.
2. In the InoProShop installation directory "..\Inovance Control\InoProShop1.8.0.0\CODESYS \Common", double-click "UninstallSVN.bat" batch file to uninstall the SVN plug-in. After the plug-in is uninstalled, the following figure is displayed.



3. Close the batch command window and re-open InoProShop.
4. In the menu bar of InoProShop, choose "Tools" > "Package Manager". The "Package Manager" page is displayed.



5. Select the SVN plug-in, click "Uninstall", and uninstall the plug-in as prompted.
6. After the plug-in is uninstalled, close InoProShop.

Copyright © Shenzhen Inovance Technology Co., Ltd.

19010980B06

Shenzhen Inovance Technology Co., Ltd.

www.inovance.com

Add.: Inovance Headquarters Tower, High-tech Industrial Park,
Guanlan Street, Longhua New District, Shenzhen
Tel: (0755) 2979 9595 Fax: (0755) 2961 9897

Suzhou Inovance Technology Co., Ltd.

www.inovance.com

Add.: No. 16 Youxiang Road, Yuexi Town,
Wuzhong District, Suzhou 215104, P.R. China
Tel: (0512) 6637 6666 Fax: (0512) 6285 6720