Kinco

Kinco - K Series
Programmable Controller

Application Manual

Shanghai Kinco Automation Co., Ltd

Basic description

- Thank you for purchasing a Kinco-K series programmable controller.
- This manual introduces the programming software, instruction applications, etc. for the Kinco-K series programmable controllers.
- Before using the product, please read this manual carefully and program it with a full understanding of its contents.
- For a description of hardware wiring, consult the relevant hardware manual.
- Please deliver this manual to the end user.

Information for Users

- The examples listed in other technical information such as manuals are for the user's understanding and reference only.
- When using this product in combination with other products, check that it complies with the relevant specifications and principles.
- When using this product, please check for compliance and safety by yourself.
- Please set your own backup and safety functions to avoid possible machine failure or damage due to malfunction of this product.

Declaration of responsibility

- Although the contents of the manual have been carefully checked, errors are inevitable and we cannot guarantee full consistency.
- We will frequently check the contents of the manual and make corrections in subsequent editions, so we welcome your valuable comments.
- The contents described in the manual are subject to change without notice.

Preface

Shanghai Kinco Automation Co., Ltd. is a high-tech enterprise dedicated to the development, marketing, production and service of domestic high-quality PLC. The company's entrepreneurs with more than ten years of practical experience in the field of automation and a deep understanding of PLC products, the use of our PLC software/hardware core technology, the development of high-quality and international synchronization with the small PLC, 100% own intellectual property rights, excellent quality, considerate service, the creation of a small integrated PLC for domestic users of the national brand, to break the monopoly of the imported brands in the Chinese market situation, to contribute to the strength of the national automation industry. To break the monopoly of imported brands in the Chinese market and to contribute to the national automation industry.

Since 2011, Shanghai Kinco Automation Co., Ltd. has launched the Kinco-K series of small PLCs, including K5, K6, K2, KS, KW and other products. Among them, K5, K6, KS are standard products with rich functions, high performance, high reliability, good expandability, etc. K2 is an economic single-machine type product, which optimizes the hardware design to reduce the cost under the premise of guaranteeing the functions, performance, reliability, and provides functions closer to the user's needs, such as USB programming port, transistor-type DIO point (DI, DO multiplexing), etc., which is highly cost-effective. KW is a wireless product for smart factories launched by BBK. Under the premise of continuing the feature-rich, high-performance and high-reliability of K series, KW adopts a higher performance CPU, and provides functions closer to users' needs, such as wireless network interface, MicroUSB programming, higher performance high-speed input/output, compact installation, etc., so that it can satisfy the users' needs for various applications. Since its introduction to the market, the K series has been widely recognized as the best choice for the users. Since its introduction to the market, K series PLC has been applied to environmental protection machinery, woodworking machinery, textile machinery, building materials machinery, food machinery, central air conditioning and small-scale process control and other fields, with its excellent performance-price ratio has been unanimously recognized by users.

For the convenience of users, we have prepared this manual to provide a systematic introduction to

KincoBuilder, the programming software for K series PLCs. This manual is exhaustive, describing in detail the basic concepts of programming, the interface functions, operation methods and instruction sets of the KincoBuilder software, interspersed with a large number of examples. In addition, some basic concepts of IEC61131-3 are also introduced in this manual to take care of the majority of beginner PLC users.

The book has omissions and deficiencies, please correct the majority of users, thank you.

All rights reserved. Shanghai Kinco Automation Corporation reserves all rights to this book.

Address: 3F, No.1 Building, No.6, Langshan 1st Road, North District, Hi-Tech Park, Nanshan District, Shenzhen, China.

Postcode: 518057

Tel: 0755-26585555

Fax: 0755-26616372

Email: sales@kinco.cn

Website: https://en.kinco.cn/

Contents

Chapter 1 Welcome to KincoBuilder 13
1.1 KincoBuilder description 13
1.2 Comment terms used in this book 14
Chapter 2 Quick Start with KincoBuilder 17
2.1 system requirement 17
2.1.1 Hardware requirement 17
2.1.2 software requirement 17
2.2 General introduction of the windows of KincoBuilder 19
2.3 Organization of applications in KincoBuilder 20
2.3.1 Project Organizational Structure 20
2.3.2 project storage directory 21
2.3.3 Importing Projects and Exporting Projects. 22
2.4 CPU executes user program 24
2.5 How to connect computer with PLC 26
2.6 How to modify the communication parameters of the CPU? 41
2.7 CPU Status and Indicators 43
Chapter 3 PLC programming basics 46
3.1 POU, Programm Organization Unit 46
3.2 Data Type 47
3.3 identifier 49
3.3.1 Definition of Identifiers 49
3.3.2 use of identifiers 49
3.4 Introduction to Constants 49
3.5 Variable introduction 51
3.5.1 variable declaration 51
3.5.2 declare variable in KincoBuilder 52
3.5.3 Test of variables 52
3.6 Memory area and addressing mode 53
3.6.1 Types of memory regions and their characteristics 53
3.6.2 Direct addressing of memory regions 55
3.6.3 Indirect addressing of memory regions 62
3.6.4 The address range of the memory area 65
3.6.5 About function blocks and function block instances 70
3.6.6 Naming and use of FB instances 72
3.6.7 Range of FB instance memory area 73
Chapter 4 Basic functions of using KincoBuilder 75
4.1 KincoBuilder settings 75
4.2 floating window 79
4.3 PLC Hardware settings 80

4.3.1 How to enter the hardware configuration window? 81
4.3.2 Copy and paste hardware configuration information in different projects 81
4.3.3 Add and remove modules 82
4.3.4 Configure module parameters 83
4.4 Initialize data table 95
4.4.1 How to enter the initialization data table? 96
4.4.2 Enter data in table cells 96
4.4.3 Define initialization data 96
4.4.4 Edit initialization data table 97
4.5 global variable table 98
4.5.1 How to enter the global variable table? 99
4.5.2 declare global variable 99
4.6 cross-reference table 101
4.6.1 How to enter the cross-reference table? 102
4.6.2 Working with cross-reference tables 102
4.7 Variable state table 103
4.7.1 How to enter the variable state table? 105
4.7.2 Monitor the value of a variable 105
4.7.3 About the forcing function 105
4.7.4 right-click menu 106
4.7.5 Force, cancel force 107
Chapter 5 Write User Program with KincoBuilder 108
5.1 IL programming 108
5.1.1 IL's background 108
5.1.2 IL syntax 109
5.1.3 IL editor in KincoBuilder 111
5.1.4 Convert IL program to LD program 116
5.1.5 Debug and monitor programs 116
5.2 LD Programming 118
5.2.1 LD Background 118
5.2.2 Network 118
5.2.3 Standardized Graphics Objects 119
5.2.4 LD editor in KincoBuilder 121
5.2.5 Monitor and debug programs 序 131
5.2.6 View PLC errors and faults 133
5.3 How to use KincoBuilder to create main program, subprogram and interrupt program 135
Chapter 6 Basic application instruction set 141
6.1 Instruction Set Overview 141
6.2 bit instruction 142
6.2.1 standard contact 142
6.2.2 Immediate contact 145
6.2.3 Normal output 147

6.2.4 Immidiate output 148
6.2.5 Set and reset 149
6.2.6 Block set and block reset 151
6.2.7 Immediate Set and Immediate Reset 152
6.2.8 edge detection 153
6.2.9 NCR (Negate) 155
6.2.10 Bistable flip-flop 156
6.2.11 ALT (Reverse) 159
6.2.12 NOP (no-op) 160
6.2.13 bracket modifier 161
6.2.14 BCNT (Position Statistics) 162
6.3 Assignment instruction 163
6.3.1 MOVE (assign) 163
6.3.2 BLKMOVE (block transfer) 165
6.3.3 BLKMOVE (block transfer, pointer parameter) 166
6.3.4 FILL (block assignment) 168
6.3.4 SWAP (Swap) 171
6.3.5 XCH (two data exchanges) 172
6.4 compare instruction 174
6.4.1 GT (Greater) 174
6.4.2 GE (Greater or Equal) 176
6.4.3 EQ (Equal) 178
6.4.4 NE (Not equal) 180
6.4.5 LT (Less than) 181
6.4.6 LE (less than or equal to) 183
6.4.7 ZONECMP (Data zone comparison) 185
6.4.8 CMP (Comparison) 187
6.4.9 TIMECMP (Time comparison) 188
6.4.10 DATECMP (Data comparison) 190
6.5 logic operation 192
6.5.1 NOT (bitwise NOT) 192
6.5.2 AND (Bitwise AND) 193
6.5.3 ANDN (Bitwise ANDN) 194
6.5.4 OR (Bitwise OR) 196
6.5.5 ORN (Bitwise ORN) 198
6.5.6 XOR (Bitwise XOR) 199
6.6 shift instruction 201
6.6.1 SHL (Shift left) 201
6.6.2 ROL (Rotate left) 202
6.6.3 SHR (Shift right) 204
6.6.4 ROR (Rotate right) 205
6.6.5 SHL_BLK (bit string shift left) 207

6.6.6 SHR_BLK (bit string shift right) 209
6.6.7 ROL_BLK (bit string rotate left) 211
6.6.8 ROR_BLK (bit string rotate right) 213
6.6.9 WSFL (Words shift left) 215
6.6.10 WSFR (Words shift right) 216
6.7 type conversion 218
6.7.1 DI_TO_R (Double integer to real) 218
6.7.2 R_TO_DI (Real to Double Integer) 220
6.7.3 B_TO_I (Byte to Integer) 221
6.7.4 I_TO_B (Integer to byte) 222
6.7.5 DI_TO_I (Double integer to integer) 223
6.7.6 I_TO_DI (Integer to Double Integer) 225
6.7.7 BCD_TO_I (BCD code to Integer) 226
6.7.8 I_TO_BCD (Integer to BCD code) 227
6.7.9 I_TO_A (Integer to ASCII) 228
6.7.10 DI_TO_A (Double integer to ASCII) 231
6.7.11 R_TO_A (Real to ASCII) 233
6.7.12 H_TO_A (Hexadecimal to ASCII) 235
6.7.13 A_TO_H (ASCII to hexadecimal) 237
6.7.14 ENCO (Code) 239
6.7.15 DECO (Decode) 240
6.7.16 SEG (Seven-segment code) 242
6.7.17 TRUNC (Rounding) 243
6.8 computation 244
6.8.1 ADD (Addition)、SUB (subtraction) 244
6.8.2 MUL (multiplication)、DIV (division) 246
6.8.3 MOD (Find Remainder) 248
6.8.4 INC (add 1)、DEC (minus 1) 249
6.8.5 ABS (Absolute value) 250
6.8.6 SQRT (square root) 251
6.8.7 LN (natural log)、LOG (log) 252
6.8.8 EXP (exponents with base e) 253
6.8.9 SIN、COS、TAN 254
6.8.10 ASIN、ACOS、ATAN 255
6.8.11 POW (Exponentiation) 256
6.9 array instruction 258
6.9.1 Overview of Array Instructions 258
6.9.2 array instruction 261
6.9.3 Array usage example 273
6.10 stack instruction 282
6.10.1 Overview of stack instructions 282

6.10.2 stack instruction 286
6.10.3 stack usage example 291
6.11 Program control 298
6.11.1 Labels and Jump Instructions 298
6.11.2 return instruction 301
6.11.3 CAL (call subroutine) 303
6.11.4 FOR/NEXT (loop instruction) 304
6.11.5 END (End the main program) 308
6.11.6 STOP (Stop CPU) 309
6.11.7 WDR (watchdog reset) 309
6.12 interrupt instruction 311
6.12.1 Kinco-K how series handles interrupt events? 311
6.12.2 Interrupt priority and queue 311
6.12.4 List of interrupt events 312
6.12.5 ENI (enable interrupt)、DISI (Disable interrupt)instruction 314
6.12.6 ATCH (disconnect)、DTCH (interrupt separation)instruction 315
6.13 counter 317
6.13.1 CTU (count up)、CTD (count down) 318
6.13.2 CTUD (count up/down) 320
6.14 Timer 322
6.14.1 timer's time base 323
6.14.2 TON (On-delay timer) 323
6.14.3 TONS (On-delay cumulative timer) 324
6.14.4 TOF (off-delay timer) 326
6.14.5 TOFS (Off-delay cumulative timer) 328
6.14.6 TP (Pulse timer) 330
6.14.7 TP_R (Resettable Pulse Timer) 332
6.15 Additional instructions 334
6.15.1 LINCO (Linear transformation) 334
6.15.2 CRC16 (16-bit CRC check code) 336
6.15.3 ENAES (AES-128 encryption) DEAES (AES-128decryption) 337
6.15.4 Special data area read and write instructions 339
Chapter 7 Introduction to Real Time Clock 341
7.1 Real Time Clock 341
7.1.1 Adjust CPU clock 342
7.1.2 READ_RTC (read real time clock)、SET_RTC (Set real time clock) 343
7.1.3 RTC_R (read real time clock) 345
7.1.4 RTC_W (write real time clock) 347
7.1.5 ELAPSEDTIME (The interval between two moments) 348
Chapter 8 Use of High-speed Counter Function 352
8.1 Function description 352

8.2 High-speed counter operating modes and input signals 353
8.3 High-speed count value range 354
8.4 High-speed counter input terminal wiring 354
8.5 Timing diagram of high-speed counter 355
8.6 Control Register and Status Register 360
8.7 High-speed counter interrupt 363
8.8 Use of PV Interrupts 364
8.8.1 Preset value (PV value) setting 365
8.8.2 Relative and absolute values 366
8.8.3 CV=PV interrupt number 370
8.9 HDEF (High-speed counter definition)、HSC (High-speed counter)Instruction 371
8.10 SPD (pulse density) instruction 372
8.11 How to use the high-speed counter 374
8.11.1 Programming with relevant instructions 374
8.11.2 Using the HSC Wizard 382
Chapter 9 Use of high-speed pulse output function 389
9.1 Function Overview 389
9.2 Types of high-speed pulse output instructions 390
9.3 Use of position control Instructions 390
9.3.1 Model 390
9.3.2 Related variables of position control instructions 391
9.3.3 Position Control Instruction 395
9.4 Use of PLS Instructions 435
9.4.1 High-speed pulse output function (PTO/PWM) 435
9.4.2 PTO/PWM Control Register and Status Register 438
9.4.3 PLS instruction 440
9.4.4 Using the PTO function 441
9.4.5 Using the PWM function 448
Chapter 10 Use of communication functions 454
10.1 Function Overview 454
10.2 Use of the serial communication port 454
10.2.1 Free communication 455
10.2.2 Modbus RTU communication function 482
10.2.3 Dynamic modification of RS485 communication port parameters 509
10.3 Use of Ethernet port 519
10.3.1 Description 519
10.3.2 Network port parameter setting 519
10.3.3 Ethernet instructions 533
10.4 LPWAN Use of wireless communication port 588
10.4.1 Overview 592
10.4.2 Common Wireless Communication Terminology 592
10.4.3 Parameter setting of LoRa communication port 593

10.4.4 Communication using LoRa 598
10.4.5 LoRa function related instructions 602
10.5 CAN Bus usage 611
10.5.1O Overview 611
10.5.2 hardware wiring 613
10.5.3 Extended bus functionality 613
10.5.4 CANOpen functions of master stations 616
10.5.5 CANOpen Example 641
10.5.6 CANOpen slave station function 660
10.5.7 CAN free communication function 662
10.5.8 Kinco motion control function 682
Chapter 11 Use of Analog and PID 726
11.1 Function overview 726
11.2 Use of analog 726
11.2.1 Analog Introduction 726
11.2.2 Signal form, measurement range and expression form of analog quantity 727
11.2.3 Setting of analog quantity in hardware configuration 729
11.2.4 Introduction to thermocouple measurement and external compensation instructions 732
11.3 Introduction to PID instruction 736
11.4 How to use PID 741
11.4.1 Programming with PID instructions 742
11.4.2 Using the PID Wizard 748
Chapter 12 Data Retention and Data Backup 755
12.1 Data Retention and Data Backup 755
12.2 Backup battery and battery power indicator 758
Chapter 13 Troubleshooting 759
13.1 error type 759
13.1.1 fatal error 759
13.1.2 Serious error 764
13.1.3 general error 764
13.2 Error code 765
13.3 How to read the errors that have occurred in the PLC 768
13.4 Error register 769
13.5 How to restore the factory to the original state? 772
Chapter 14 Intellectual Property Protection 774
14.1 password protection 774
14.1.1 protection level 774
14.1.2 Change password and protection level 775
14.1.3 What to do if you forget your password 776
Appendix A Definition of common system memory SM area 777
1、SMB0: system status byte 777
2、error register 777

3、SMD12 and SMD16: Period of timer interrupt 779
4、Control and Status Registers for High-Speed Counters 780
5、 Control register and status register for high-speed pulse output 782
5.1 Use of PLS instruction 782
5.2 Using position control instructions 783
6、Control registers and status registers related to free communication instructions XMT/RCV 784
7、Dynamically modify the relevant registers of RS485 communication port parameters 787
8、Other commonly used function variables 789
Appendix B Updating Programs of the System 790
Appendix C Use of PLC Offline Simulator 794
2.1 startup steps 794
2.2 Instructions that do not support simulation 794
2.3 simulation tools 795
4.1 basic simulation 798
4.1.1 The running state is distinguished by the color change of the node 798
4.1.2 Introduction to basic simulation operation instructions 798
4.2 breakpoint debugging 802
4.2.1 Network interruption points and their state change indications 802
4.2.2 breakpoint operation 802
4.3 IO 804
4.4 communication simulation 805
Appendix D Introduction to the screen part of the all-in-one machine 811
1.1 Use of HMI 811
1.1.1 Engineering production 811
1.1.2 HMI manual download link 815
2.1 Use of HMI 815
2.1.1 Use of Project 815
2.1.2 HMI manual download link 819
Appendix E Introductory case (steps to create a project) 820
Appendix F Use of Expansion Modules 841
1、 Overview 841
2、 Introduction 842
Appendix G Use of Extended BD Board 845
1、Overview 845
2、Use of BD board 846
2.1 Function and parameter description of various BD boards 847
Appendix H Usage of KS expansion module as ModBus RTU slave station 853

Chapter 1 Welcome to KincoBuilder

1.1 KincoBuilder description

KincoBuilder is the upper programming software of K series PLC of Kinco Company. The programming environment conforms to IEC61131-3 standard. It is a powerful, easy-to-use and efficient development system.

IEC61131-3 is a standard developed by IEC for industrial automation programming. This standard is formulated to absorb the programming language styles of different manufacturers and to meet the requirements of future software technology development It is independent of any company and is suitable for different fields and different types of programmers. Since its release, it has been recognized by all the top PLC manufacturers, and the programming software of each manufacturer is also trying to move closer to the IEC standard.

KincoBuilder is completely independently developed by Kinco. KincoBuilder adopts a solution that conforms to the IEC61131-3 standard. Since many users have mastered most of the programming skills through various channels, this makes KincoBuilder simple, easy to learn, and easy to use.

KincoBuilder has the following characteristics:

- Comply with IEC61131-3 standard, support IL (instruction list) and LD (ladder diagram) two standard languages
- Rich instruction set, built-in standard functions, function blocks and some special application instructions defined by IEC61131-3
- Support structured programming, user programs are organized into "projects"
- Support interrupt service routine, support user-defined function block (subroutine)
- Allows the use of variable names in the program, which is convenient for the implementation and maintenance of the project
- Flexible hardware configuration, allowing users to customize various hardware parameters to the greatest extent

- Perfect online functions, including download, upload, online monitoring, force, read and write real-time clock, etc.
- Defines perfect shortcut keys and right-click menu, which is convenient for users to use
- Shield and prompt users' wrong operations as much as possible, and be considerate of users' operations

1.2 Comment terms used in this book

- Small integrated programmable controller

According to the international general classification principle, small PLC generally refers to the PLC with the actual control points less than 128 points (not the design control points). This type of PLC usually adopts an integrated structure, that is, a certain number of I/O, output power, high-speed input/output accessories, etc. are integrated on the CPU module.

- CPU

It is also called CPU module. The CPU is the heart of the control system. The application program written by the user is downloaded by the programming software and stored in the permanent memory of the CPU module. The control function of the user program is executed under the scheduling of the CPU running software. The running software is also responsible for diagnosing the working status of each module and the execution error of user program.

- Expansion module and expansion bus

Expansion modules and expansion bus expansion modules are modules used to expand the functions of the CPU body. They are divided into expansion I/O modules (increase the input/output channels of the system) and expansion function modules (expand CPU functions).

The expansion bus connects the CPU module with the data and signal channels of the expansion module. Physical medium K5 series adopts 16-core flat cable, and other series adopts ordinary direct-

connected network cable. In the expansion bus, the data bus, the address bus and the working power supply of the expansion module are integrated.

- KincoBuilder

KincoBuilder is the programming software of Kinco-K series PLC, which conforms to the IEC61131-3 standard and currently supports two standard languages, LD and IL. KincoBuilder can be used to realize various functions such as programming and debugging for Kinco-K series PLC.

- CPU running software

CPU running software, also known as CPU board-level firmware (firmware), is stored in the Flash memory of the CPU body. It starts running when the CPU module is powered on, and manages and schedules all tasks of the CPU. User-written applications are executed under the schedule of the CPU running software. The running software is also responsible for diagnosing the working status of each module and the execution error of the user program.

- User Application

User application also known as user program and user project, is a program written by the user to complete a specific control function. After the application program is downloaded, it is stored in the permanent memory of the CPU module, and is read into RAM for execution when the CPU is powered on and running.

- Main program and program scan

In the CPU, various tasks are continuously executed cyclically, and this process of cyclically executing tasks is called scanning.

The main program is the execution entry of the user application. The main program is executed once in every scan cycle. There can be only one main program in an application, but multiple subprograms can be called from the main program.

- I/O image area

The storage area of the status data of physical I/O points in the CPU is divided into input image area and output image area.

In order to ensure the consistency of data in one scan of the CPU and improve the execution speed of the program, the CPU first reads the state of the physical input point into the input image area in each scan, and only accesses the I/O image area during the execution of the application program. and then send the data in the output image area to the physical output channel at the end of the scan. The I/O addresses mentioned in this manual, such as I0.0, Q1.0, AIW0, AQW0, etc., are the addresses in the I/O image area.

- Data retention and data backup

Data retention means that after the CPU is powered off, the data in the RAM is kept in the state of the moment of power off and is available for the CPU to use when the power is next powered on. The data is kept by a one-time battery. At normal temperature, the retention time is about 3 years. Users can replace the battery by themselves.

Data backup means that the user writes data to permanent storage for preservation through instructions. Note: Persistent memory has a service life, E2PROM allows 1 million writes, and FRAM allows 10 billion writes。

Chapter 2 Quick Start with KincoBuilder

This chapter describes the KincoBuilder installation and operating environment in detail. In addition, it also describes how to connect the PLC to the computer. Finally, an example is given to illustrate the common steps in developing a project. The purpose of this chapter is to help users get started quickly. For detailed descriptions of various functions of KincoBuilder, please refer to the subsequent chapters.

2.1 system requirement

2.1.1 Hardware requirement

- CPU frequency is 1GHz or higher, hard disk space is more than 20M, memory is more than 512M
- Keyboard, mouse, serial communication port (com port) or USB port (some series require additional USB/RS232 or USB/RS485 converter)
- 256 colors or higher, resolution 1024*768 or higher

2.1.2 software requirement

Windows XP(32bit), Windows Vista(32bit), Windows7(32/64bit), Windows8(32/64bit), Windows 8.1(32/64bit), Windows10(32/64bit)。

Some users may encounter problems running KincoBuilder under Windows 7 or later operating systems. The common problems and solutions are listed below.

- 【COM port】list in 【PC communication setting】dialog is empty

KincoBuilder obtains the available COM port by reading the hardware information in the Windows registry。

In some versions of the Windows system, the user must run KincoBuilder as an administrator,

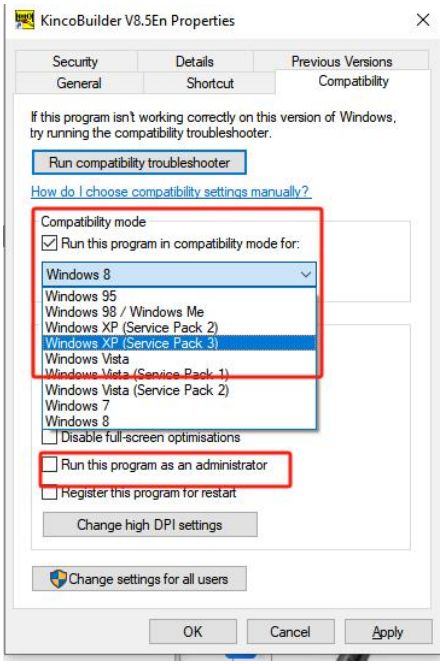
otherwise KincoBuilder will not be able to access the registry due to insufficient permissions, nor will it be able to obtain the COM port of the machine. At this point, COM1-COM255 will be displayed directly in the [COM Port] list, and the user can check the COM port of the machine from the Windows Device Manager, and directly select and use it in the list.

- KincoBuilder doesn't work on some computer

  Please try to run KincoBuilder in Windows XP compatible mode. The setting method is as follows:

  1) Right-click on the shortcut of "KincoBuilder" on the desktop or in the [Start] menu, and then execute the [Properties] instruction in the pop-up menu。

  In the [Properties] dialog box, enter the [Compatibility] page to set, as shown in Figure 2-1.



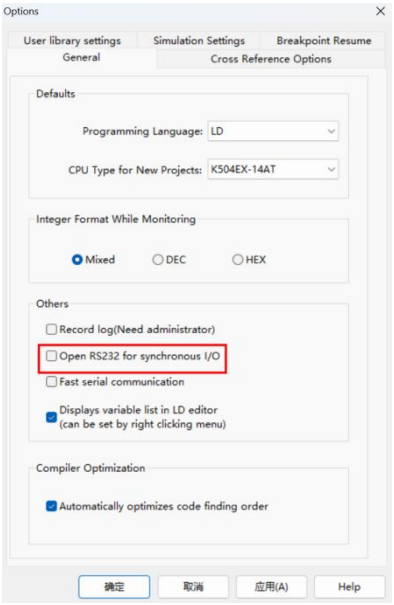


Figure 2-1Windows"Compatibility"set Figure2-2"Open serial port in synchronous mode"

- Communication with PLC often fails when using some USB to RS232 converters

  This is caused by the compatibility of this converter driver. The few failure cases found so far are

mostly the 64-bit version of Win7.

In KincoBuilder, execute the menu instruction [Tools] → [Software Settings...] to open the "Software Settings" dialog box, select "Open Serial Port in Synchronous Mode", and then click the "OK" button to exit. (Figure 2-2)

After the setting is completed, KincoBuilder will operate the serial port in a synchronous manner, which can solve the problem of inoperability in most cases.

2.2 General introduction of the windows of KincoBuilder

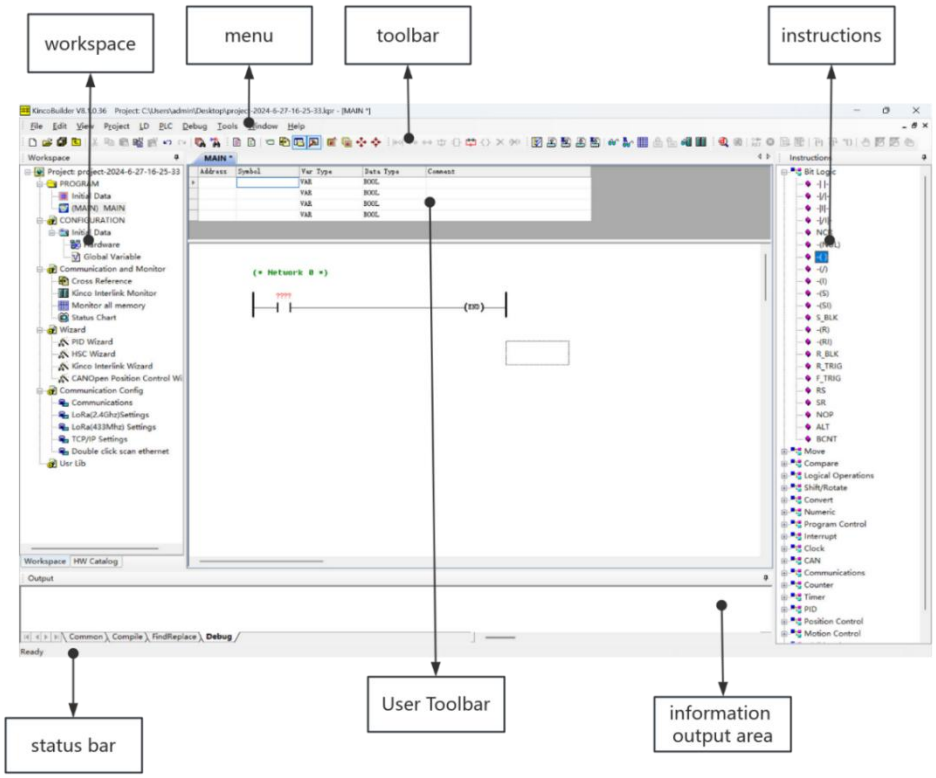


Figure 2-3 KincoBuilder Windows

- Menu: The menu contains all the operation instructions of KincoBuilder software.

- Toolbar: The toolbar contains some operating instructions that are frequently used by users.
- Status bar: The status bar provides the current status information of the software and prompt information of operation instructions.
- Project Manager: Project Manager is one of the main windows in the interface. It visually displays all the components of the current project in the form of a tree list, including programs, hardware configuration, variable status table, global variable table, etc. The user can manage, operate and maintain the currently opened project in this window. Each tree node of the project manager supports right-click, and right-clicking a node will pop up the corresponding right-click menu.
- User workspace: This is the main area used by the user, where the user can open the hardware configuration window, global variable table, editor and other windows to complete tasks such as configuring hardware, declaring global variables, and editing programs. Figure 2-3 shows the editor, including the variable definition table in the upper part of the area and the program editor in the lower part of the area (also divided into LD editor, IL editor)。
- Instruction set: Lists all instructions, functions, function blocks and subroutines written by users themselves supported by the Kinco-K series PLC in tree form. The instruction set is further divided into the LD instruction set and the IL instruction set.
- Information output area: used to display various prompt information of KincoBuilder software. Among them, the "Compile Information" window displays the last compilation information of the user, and the "General Information" window displays the prompt information of some recent operations.

2.3 Organization of applications in KincoBuilder

2.3.1 Project Organizational Structure

In KincoBuilder, the application program is organized into "Project", and the project contains all the information of the application program such as user program and hardware configuration. In this manual, the terms "user program" and "user project" have the same meaning.

The following table details the organization of the project. Items marked "optional" in the table

indicate that this item is not a necessary element of the project, and the user can ignore them in the project

Program	Initialize the data table (optional)		The user can specify the initial value for the BYTE, WORD, DWORD, INT, DINT, REAL type data in the V area here. The initialization data table is an optional part of the project, and the user can also not enter anything in the table. The initialization data table should be processed once before the CPU enters the main cycle on a cold start.
	Main program		The main program is the main loop task that runs in the CPU and is executed once in each scan cycle of the CPU. There is only one main program in the project.
	interrupt service routine (optional)		A program that is executed in response to an interrupt event is called an interrupt service routine. Interrupt service routines do not need to be called in the main program. The user only needs to connect it with a predefined interrupt event through the ATCH instruction, then when the interrupt event occurs, the CPU will automatically call the interrupt service routine for processing.
	subroutine (optional)		The subroutine must be called by the main program or the interrupt service routine before it can be executed. Subprograms are reusable code segments, and users can write common control functions into subprograms one by one. Using subroutines can better organize the structure of the application, facilitate debugging and maintenance, and effectively shorten the length of the program code
Setting	resource	Hardware Configuration	The user configures the PLC modules and their parameters used in the project. The CPU will process the hardware configuration once at a cold boot before performing other tasks.
		overall variable table (optional)	The user declares the global variables required in the project here.

Table 2-1 Organizational structure of the project

2.3.2 project storage directory

When creating a project, KincoBuilder will ask the user to input the storage path of the project, and then an empty project file (extension .kpr) will be generated and stored in this path. In addition, a subdirectory with the same name as the project will be automatically created under this path to store all program files, variable files and other temporary files of the project.

For example, if the user chooses to create a project named “project” in the c:\temp directory, the path of the project file is c:\temp\project.kpr, and other files are stored in the c:\temp\project directory.

Note: When the project is applied, the project file (extension .kpr) and the folder with the same name must exist at the same time, otherwise the project will be invalid. If you want to save the project, you need to back up both at the same time to be a complete program, and both are indispensable. There is also a shortcut method for project backup, which can directly back up the project as a compressed file, which is convenient for storage and avoids serious consequences caused by forgetting files. For the specific export method, refer to 2.3.3 Importing Projects and Exporting Projects.

2.3.3 Importing Projects and Exporting Projects.

KincoBuilder provides [Import Project...] and [Export Project...] instructions in the [File] menu to facilitate users to backup and manage projects.

- [Export Project...]

Compress all files involved in the currently open project into one file (extension .zip). In the compressed file, all file names, relative paths, and project-related settings remain unchanged. For example, if the project name is myproj1 and the project file is myproj1.kpr, the project file in the compressed file is still myproj1.kpr, and the project name is still myproj1.

After executing [Export Project...], the "Export Project..." dialog box will pop up, as shown in the figure below. After selecting the path and entering the backup file name, click the [Save] button.

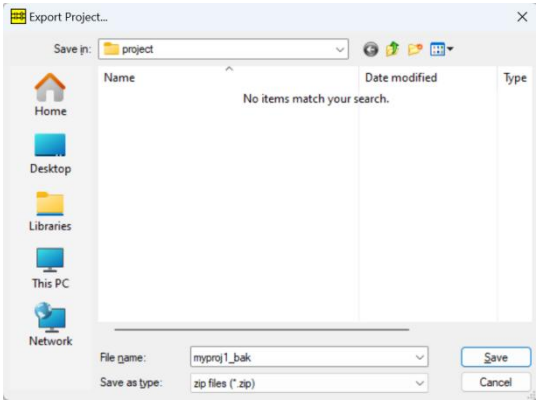


Figure 2-4Export Project

- [Import Project...]

  Import an existing project backup file (extension .zip) and open it.

  After executing this instruction, the "Import Project..." dialog box will pop up first, as shown in the figure below.

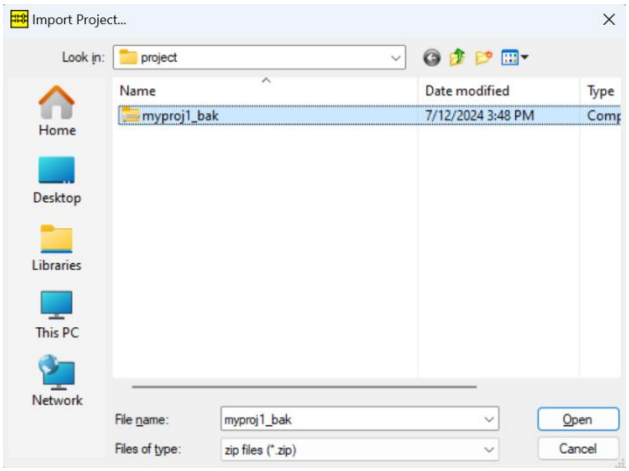


Figure 2-5 input project

After selecting the backup file and clicking the [Open] button, the following dialog box will pop up, select the storage directory of the project file after decompression. After selecting the path, click the [OK]

button to extract the project file to the selected directory and open it.

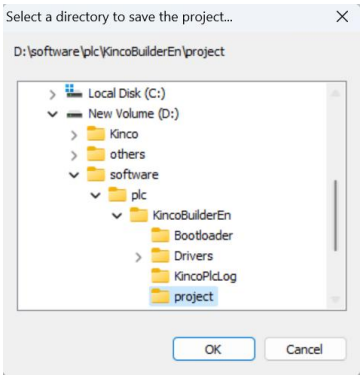


Figure 2-6 Import project: select directory

2.4 CPU executes user program

The process in which the CPU executes a series of tasks continuously in a loop is called a scan. Scanning is the main job of the CPU and is often referred to as the main loop. During one scan cycle, the CPU will perform the following tasks:

• • CPU self-diagnosis

• Read input: read the signal data of the physical input channel and write it to the input image area

• Execute user program: directly execute the main program in the user project

• Handling communication requests

• Write output: write the data in the output image area to the physical output channel

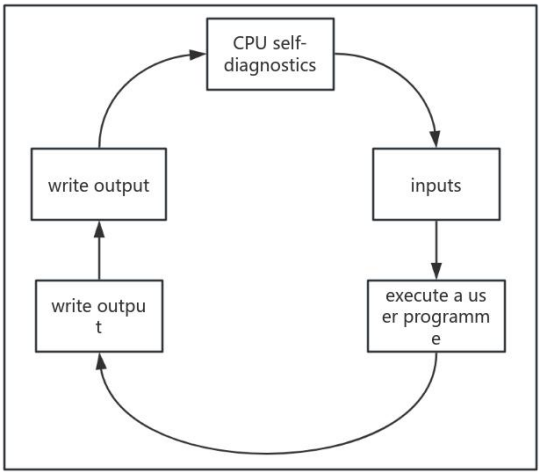


Figure 2-7 Scan cycle

Also, it should be noted that the tasks performed by the CPU during a scan cycle depend on its working state. The CPU has two working states: running (RUN) state and stop (STOP) state. The main difference between these two states is that in the RUN state the CPU will execute the user program, while in the STOP state the CPU will not execute the user program.

Kinco-K series PLC also supports interrupts, and the interrupt service routine written by the user will be executed when the interrupt event of the specified connection (ATCH) occurs. The interrupt event may occur at any time during the scanning process. At this time, the CPU will temporarily interrupt the main loop and transfer to the execution of the interrupt service routine. After the execution of the interrupt service routine is completed, it will re-enter the main loop from the previous breakpoint. As shown below.

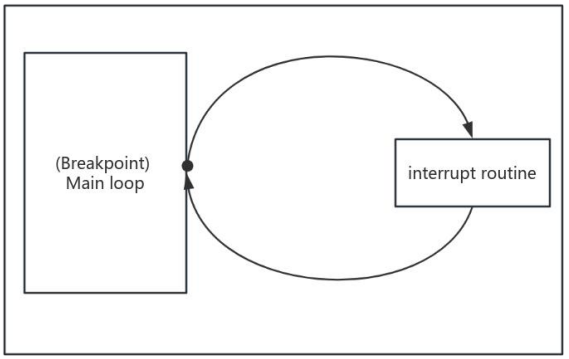


Figure 2-8 Execution of interrupt service routine

2.5 How to connect computer with PLC

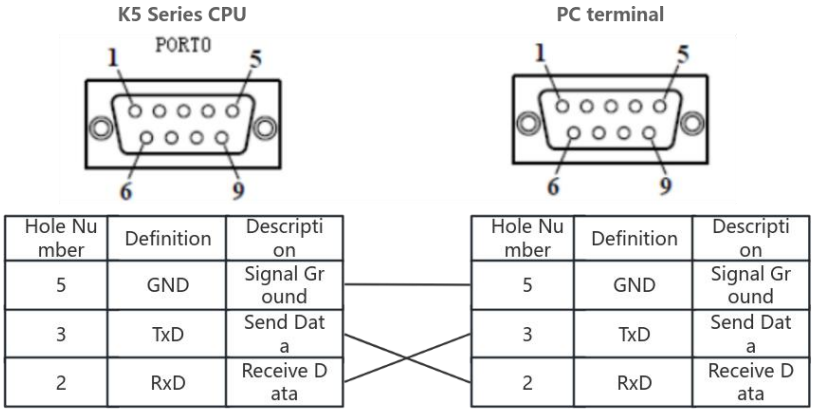
The CPU module integrates a USB port, RS232, RS485 or Ethernet communication port, and users can use these communication ports to communicate with other devices. Described here is how to start establishing communication between the computer and the CPU.

The following table lists the communication ports of the programming protocols supported by each series of CPUs。

Series	Programming port				
	USB	PORT0 (RS232)	PORT1 (RS485)	PORT2 (RS485)	Ethernet
K5 series	/	√	√	/	/
K2 series	√	CPU209EA only	√	√	K204ET only
KS series	KS101M only	√	√	/	KS101M only
KW series	√	√	√	/	/
HP series	√	/	√	/	/
MK seires	√	/	√	√	/
K6 seires	/	/	√	√	√

1) Connect the computer and CPU with the appropriate programming cable.

K5, KS, KW, K2 (CPU209EA) series PLCs can use the PORT0 (RS232) interface, and connect to the serial communication port of the computer through the following programming cable (USB/RS232 converter needs to be used if there is no RS232 port on the PC side)

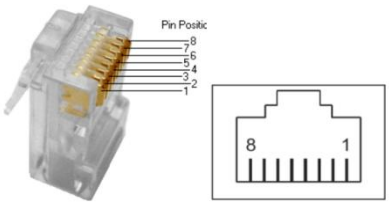


Hole Number	Definition	Description
5	GND	Signal Ground
3	TxD	Send Data
2	RxD	Receive Data

Hole Number	Definition	Description
5	GND	Signal Ground
3	TxD	Send Data
2	RxD	Receive Data

KS/KW/K209 PORT0
RS232 (on RJ45 connector)

	pipe angle	descriptions
RS232	6	RXD
	3	TXD
	4	GND

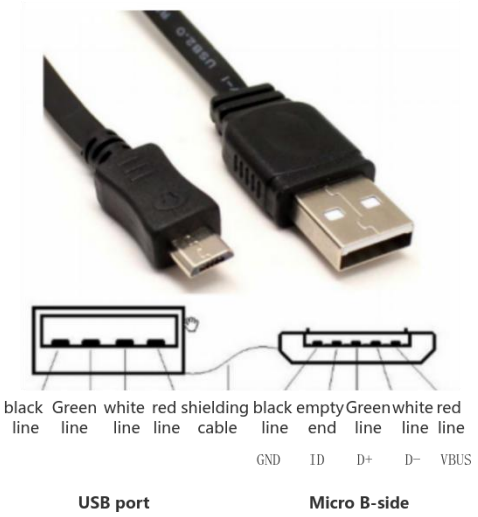


⚠ **Since the electrical standard of RS232 does not support live plugging and unplugging, the power of at least one party (CPU or computer) must be disconnected before plugging or unplugging the cable, otherwise the communication port will be easily damaged.**

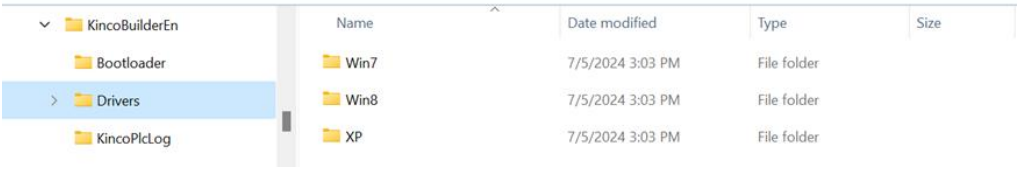
B: KS, KW, K2 series PLC and MK series PLC part use the programming cable made according to the RS485 pin on the silk screen to connect to the serial communication port of the computer (the serial

communication port of the computer needs to add a converter from RS232 to RS485) and The PORT1 port (RS485) of the CPU is connected. (If there is no serial communication port on the PC side, a USB/RS485 converter needs to be used separately)

C: K2, KW, KS (only KS101M) series PLC can use Micro USB programming cable (commonly used for Android mobile phone data cable) to connect to the computer.



On the computer, the USB programming port is used as a virtual serial port, and the driver must be installed first when using it for the first time. After installing the latest version of the KincoBuilder programming software, the drivers are stored in the \Drivers directory under the KincoBuilder programming software installation directory, and currently supports Windows XP, Windows 7, Windows 8 and Windows 10 systems. The following figure example:



When using the USB data cable to connect the PLC and the computer for the first time, the Windows system will automatically detect the new hardware and prompt to install the driver. At this time, the user can select the driver in the corresponding directory according to his Windows version.

D: MK series all-in-one machine, PLC and HMI share the same USB programming port, this interface form is the same as that used by Kinco HMI, users can directly use the same interface data cable as PLC programming cable.

On the computer, the PLC programming port is used as a virtual serial port, and the driver must be installed first when using it for the first time. After the latest version of KincoBuilder programming software is installed, the drivers for each version of Windows system are stored in the \Drivers directory under the installation directory. Currently, Windows XP, Windows 7, Windows 8 and Windows 10 are supported. When using the programming data cable to connect the integrated screen and the computer for the first time, the Windows system will automatically detect the new hardware and prompt to install the driver. At this time, the user can select the driver in the corresponding directory according to his Windows version.

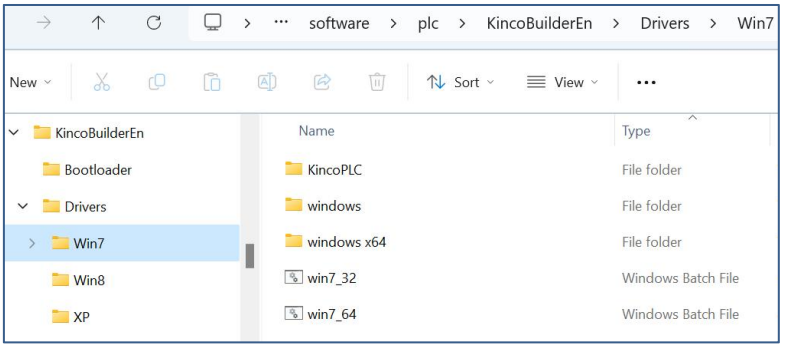
- **What should I do if the driver cannot be installed under Windows 7?**

This is because the streamlined Windows 7 system is installed on the computer, and the system lacks necessary files, which makes it impossible to install the virtual serial port. At this time, the following methods can be used to solve:

Method 1:

In the \Drivers\Win7 directory of the KincoBuilder installation directory, we have summarized the required system files and provided a batch file for the convenience of users: if the user is a 32-bit operating system, select "win7_32.bat"; If the user has a 64-bit operating system, select "win7_64.bat". The user can directly run the corresponding batch file to install the missing system files, and then the usb driver can be installed normally.

This method can solve the problems encountered by most users.

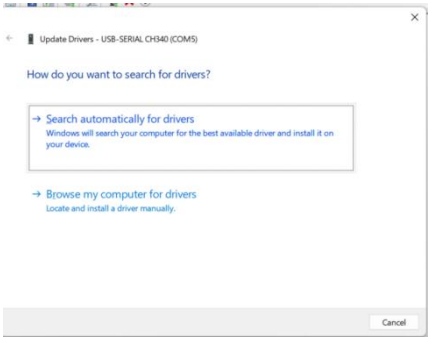


Method 2:

If the above method still fails to install the driver, the user can use the third-party driver installation software to automatically search and install.

- **How to install the driver under Windows 8 system?**

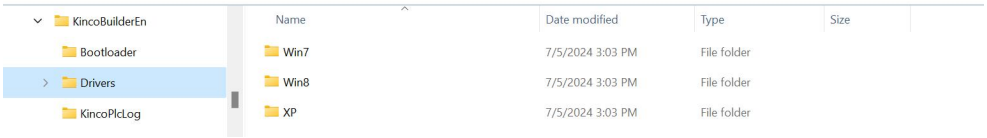
If you can connect to the Internet, you can choose the automatic search and update function of win8 and win10 to automatically install the driver.



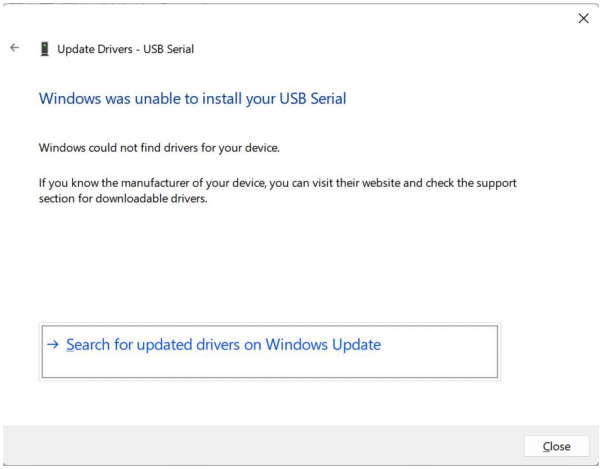
If you cannot search for updates online, please follow the graphic instructions below to install them.

The screenshot below is a screenshot of advanced startup under win8. Similar to win8, win10 also finds advanced startup from [Settings] and selects [7 Disable Driver Force Signature]。

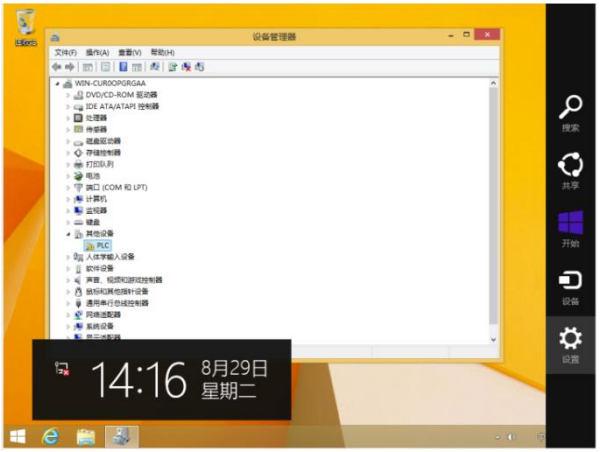
（**1**） Install the PLC driver, follow the windows prompt, and select the driver file in win8.



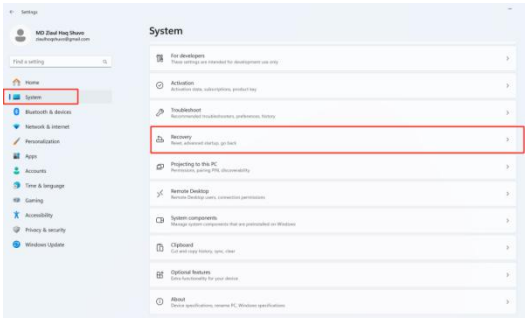
There may be an error message at this time, as shown below.



(2) Point the mouse to the lower right corner of win8, click settings

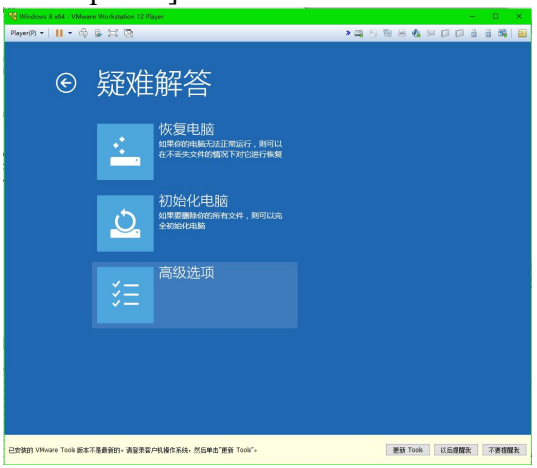


(3) In settings, click [Update and Recovery]



(4) Click [Update] in it, and then click [Start Now] under [Advanced Startup].

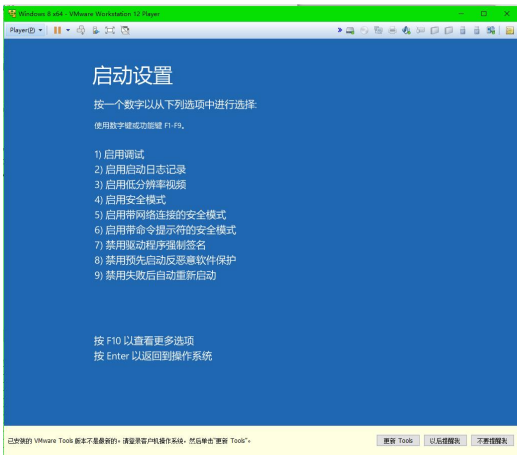
(5)Click [Advanced Options]



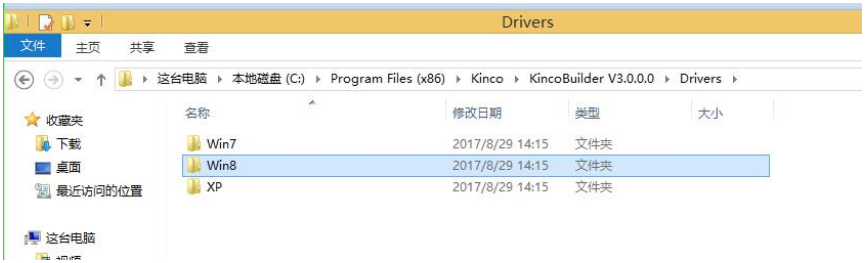
(6)Click [Startup Settings]

(7)Click [Restart]

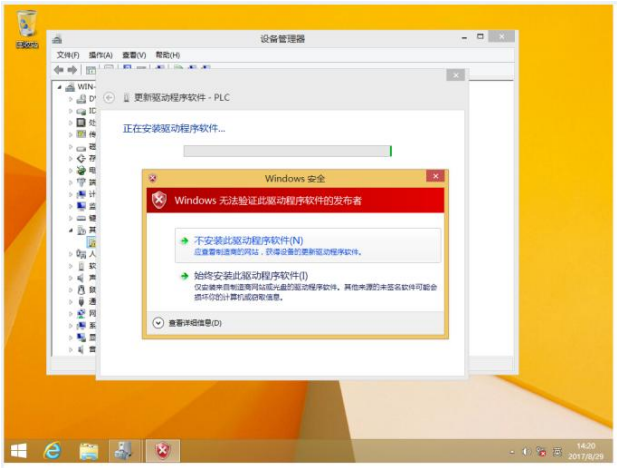
(8) This is the interface after the computer restarts. Select [7) Disable driver signature enforcement, and the computer starts to start.



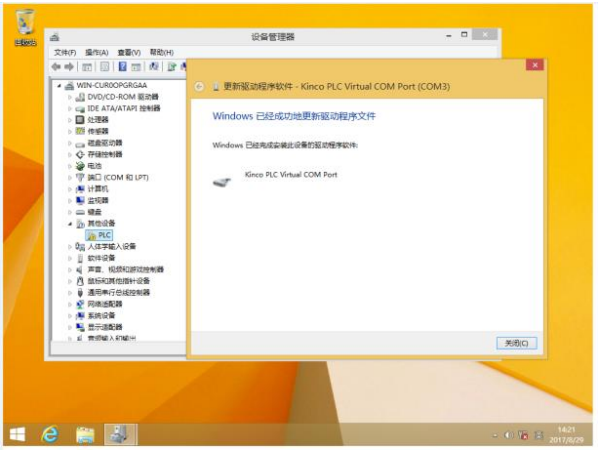
(9)Reinstall the PLC driver, follow the windows prompt, select the driver file in win8



会 The following prompt will pop up, select [Always install this driver software]。



(10)The installation is successful as shown below, and the new USB virtual COM port will be displayed in the [Ports (COM and LPT)] of the device manager.



E: K204ET-16DT, KS101M-04DX, and K6 provide an Ethernet interface that conforms to the standard IEEE802.3 specification. This interface supports programming protocol and can be used as a programming port. In addition, it also supports ModBus TCP Sever, which is commonly referred to as "slave" function.

Communication cables can be either straight-through cables (straight cables) or crossover cables (cross-over cables). The Ethernet interface of the PLC provides the "auto-negotiation" function. When the cable is inserted, the PLC will automatically negotiate with the communication partner and automatically adapt to the cable used.

When connecting the PC, make sure that the IP of the PC and the IP of the PLC are in the same network segment (the first three numbers of the IP are the same, and the last number is different).

2) Start KincoBuilder, create a new project or open an existing project.

3) PC communication settings

A: If you use a serial communication port or a USB port, you need to set the communication parameters of the computer serial communication port. These parameters must be exactly the same as the serial communication parameters of the CPU communication port for normal communication. Proceed as follows

a) In the project manager, double-click the [PC Communication Settings] node, or click the right mouse button on the [PC Communication Settings] node, and execute the [Open...] instruction in the pop-up menu, or execute [Tools] → [PC Communication Settings...] menu instruction, you can enter the "PC Communication Settings" dialog box, as shown in Figure 2-9.

Figure 2-9 "PC Communication Settings" dialog box

b) Select the station number of the target PLC to be connected in [Target PLC]; select the COM port used by the current computer in [COM Port]; set the communication parameters of the selected COM port according to the serial communication parameters of the target PLC: If the serial port of the PLC is used, then the four parameters of [Baud Rate], [Parity], [Data Bit], [Stop Bit] must be exactly the same as the corresponding parameters of the target PLC; If the USB port of the PLC is used, the above communication parameters can be set arbitrarily because the virtual serial port is used. After setting, click the [OK] button.

If the serial port of the PLC is used, and the user does not know the communication parameters of the CPU used, how to set it? There are two methods at this point:

- Method 1

First, select the computer [COM port] to be used, and then click the [Auto Detect] button, KincoBuilder will automatically detect the communication parameters of the currently connected

CPU, and the detection time may be several seconds to several minutes. If the detection is successful, KincoBuilder will automatically set the parameters of the COM port of the computer currently used according to the detected CPU communication parameters.

If the user is using a USB to RS232 converter, when the converter driver is installed and used for the first time, if the serial port parameters cannot be automatically detected, the user needs to restart the computer for the new driver to take effect.

- Method 2

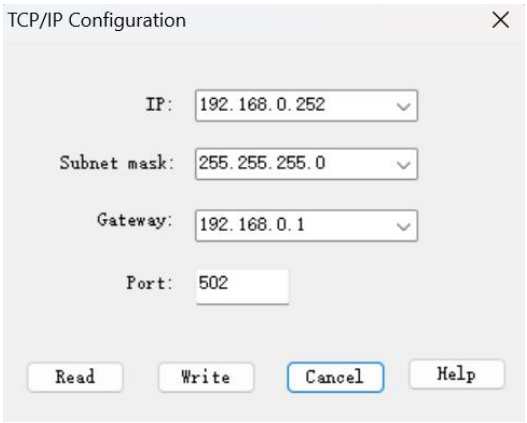
First power off the connected CPU, turn its running switch to the "STOP" position, and then power on the CPU again. At this time, the CPU will automatically set the communication parameters of some serial ports to the following default values: The station number is 1, the baud rate is 9600, no parity, 8 data bits, 1 stop bit. Users can perform various communication operations by setting the communication parameters of the COM port of the computer according to this parameter. Note: Do not change the position of the run switch until the communication parameters of the CPU are modified!

If the CPU has PORT0 (RS232) port, PORT0 will be set by default; if the CPU does not have PORT0 (RS232) port, then PORT1 (RS485) will be set by default.

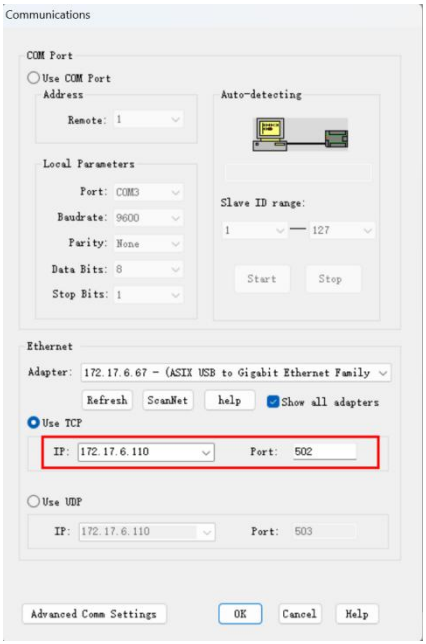
B: If using the Ethernet port, you need to set the computer's TCP/IP communication parameters (including IP address and subnet mask).

a） Before setting the TCP/IP communication parameters, first determine the interface parameters of the connected PLC, and first use serial communication to view the interface parameters. Viewing method: Software menu bar [Tools]--[TCP/IP parameter configuration]--click [Read], a prompt box "Read parameters successfully" appears, and you can get the TCP/IP interface parameters of the connected PLC.

The default parameters of the PLC network port are as follows:



b)Select [PC Communication Settings] Use TCP/IP, and set the IP address and port number under TCP/IP communication settings to be the same as the interface parameter settings read in the previous step.



Note: When downloading from the network port, the IP of the PC and the IP of the PLC must be in the same network segment. The first 3 numbers are the same, the last 1 number is

different!

4) Now that the communication parameters of the computer serial port or network port are set, the connected PLC can be programmed and debugged.。

2.6 How to modify the communication parameters of the CPU?

- **Modify the serial communication parameters of the CPU**

After connecting the computer and the CPU, the user can use KincoBuilder to check or modify the communication parameters of the CPU.

1) First, use any of the following methods to enter the hardware configuration window:
   - Double-click the [PLC Hardware Configuration] node under the [Resources] group in the project manager;
   - Right-click on the [PLC Hardware Configuration] node, and then execute the pop-up [Open…] menu instruction.

In the upper part of the hardware configuration window, the PLC modules used in the project are listed in table form, which we call the module list. The PLC module list expresses a real PLC control system: the arrangement order of each module in the table should be consistent with the connection order of the actual modules on the expansion bus.

In the lower part of the hardware configuration window, there are all configuration parameters of the selected module in the PLC module list, which we call the module parameter configuration window.

2) Click and select the CPU module in the module list, and then select the [Communication Settings] page in the module parameter configuration window at the bottom, as shown in the figure below, where the user can set its communication parameters.

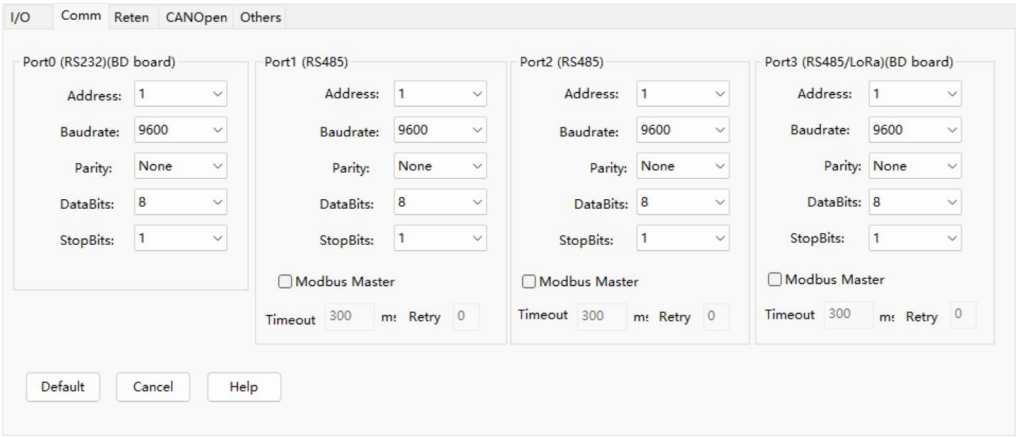


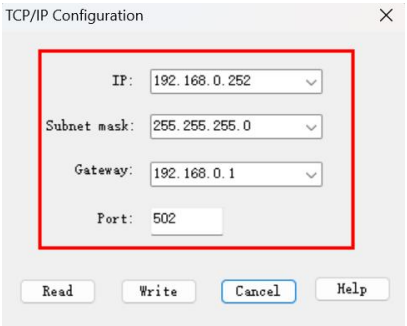
Figure 2-10 Serial port communication parameter configuration page

3) After the communication parameter configuration is completed, the current project must be downloaded to the CPU, and then the new communication parameters will take effect.。

- **Modify the communication parameters of the network port of the CPU**

Before modifying the TCP/IP communication parameters, first determine the TCP/IP interface parameters of the connected PLC, and first use the serial port, USB port, and network port (when the network segment where the IP address of the PLC is known) to communicate to view the interface parameters. Viewing method: Software menu bar [Tools]--[TCP/IP parameter configuration]--click [Read], a prompt box "Read parameters successfully" appears, and you can get the TCP/IP interface parameters of the connected PLC. Then, in the red box below, you can modify parameters such as IP address. After the modification is completed, click [Write].

The default parameters of the PLC network port are as follows:



Note: In the local area network, only PLCs and PCs located in the same network segment (that is, the first three segments of the IP address must be the same and the last segment different) can communicate with each other.

2.7 CPU Status and Indicators

The CPU module has two states: run (RUN) state and stop (STOP) state.

In the running state, the CPU module normally cyclically executes the main scan task and various interrupt tasks.

In the stop state, the CPU module only processes part of the communication requests (Including programming, debugging and other instructions from KincoBuilder programming software, as well as communication instructions responding to the master station as a Modbus RTU slave station), and at the same time outputs all output points (DO, AO) immediately to the "stop output" value defined in the [Hardware Configuration] of the user project.

- Change CPU state

The user can change the CPU state by the following methods: use the RUN/STOP switch; in the KincoBuilder software, execute the [Debug]->[Start...] or [Stop...] menu instruction.

The results of the combined operation of these two methods are as follows::

Run/Stop switch position	**After executing the KincoBuilder menu instruction once**	**PLC actual status**
RUN	Turned on	Running
	Turned off	Stopped
STOP	Turned on	Stopped
	Turned off	Stopped

In addition, in the following cases, the CPU will automatically change state regardless of the RUN/STOP switch and the [Start...], [Stop...] menu instructions.

① If the CPU detects a serious error during operation, it will immediately enter the STOP state. The CPU will detect various errors in real time during operation and divide the errors into three levels: fatal errors, serious errors, and general errors. When a fatal error is detected, the CPU automatically enters an independent safety subsystem to run, and the above methods of changing the CPU state are invalid; when a serious error is detected, the CPU will immediately enter the STOP state. See 13.1 Error Types

② When downloading a program, the CPU automatically enters the STOP state first, and then executes the entire downloading process. When the download is successful, the CPU will automatically switch to the corresponding state according to the position of the RUN/STOP switch. If the download fails or the user terminates the download process in the middle, the CPU will continue to maintain the STOP state.

③ If the "STOP" instruction is called in the user program and the instruction is executed, the CPU will immediately go to the STOP state.

- CPU indicator light

Different indicator lights are provided on the CPU module to indicate the current working condition of the CPU.

Due to the differences in CPU functions and volumes of different series, the number and types of indicators are also different, but the functions of indicators with the same logo (case-insensitive) are the same.

The indicators for all the functions provided by the KPLC are summarized below.

- [Run]: If the CPU is running, the RUN light will be on, otherwise it will be off.
- [Stop]: If the CPU is in stop state, the STOP light will be on, otherwise it will be off.
- [Comm]: When any serial communication port receives or sends data, the Comm light flashes.

- [Err]: If the PLC has an error during operation, the Err light will be on, otherwise it will be off.
- [NET]: Wireless communication indication, when the wireless communication interface receives or sends data, the NET light flashes.。

Chapter 3 PLC programming basics

This chapter describes the basic knowledge of Kinco-K series PLC programming in detail, and also introduces some basic concepts in the IEC61131-3 standard, which are very useful for users to use any kind of IEC61131-3 software. The purpose of this chapter is to help the user begin the learning and practice of programming to the point of "knowing it and knowing why".

When reading for the first time, users do not need to understand every link very thoroughly, but it is recommended that users adopt the method of "reading while experimenting", which will help to understand the content of this chapter. At the same time, it is recommended that users also use this method when reading subsequent chapters.

3.1 POU, Programm Organization Unit

IEC61131-3 introduces the concept of POU. POU contains program code and is an independent, smallest software unit, which is the basic unit of program and engineering. Traditional PLC manufacturers have defined various types of blocks for their respective PLCs in programming, and IEC61131-3 unifies these blocks into 3 types of POUs.

A user program consists of one or more POUs, which can be created by the user. POUs can call each other, but recursive calls are not allowed. It is clearly stipulated in IEC61131-3 that POUs are prohibited from calling themselves directly or indirectly.

Standard POU types are described below.

- Programm

  Keywords: PROGRAMME.

  This type of POU represents the "main program" and is used to perform a certain task.

  It can have input and output parameters, but no return value

- Function

  Keywords: FUNCTION 。

  This type of POU can have input parameters and only one return value, which is brought back by the function name. When the function is called with the same input parameters, its return value is always the same.

  It is mainly used for code reuse and can be called by other POUs.

- Function Block

  Keywords: FUNCTION_BLOCK.

  This type of POU is abbreviated as FB. It can have input and output parameters, and has static variables (that is, it can remember the previous state of the FB). The output value of the FB is passed through its output parameter. The output of an FB depends not only on its input values, but also on the state values stored in its static variables.

  FBs are also mainly used for code reuse and can be called by other POUs.

3.2 Data Type

The data type defines the bit length, value range, and initialization value of the data.

A group of the most commonly used basic data types are defined in IEC61131-3, so the meaning and usage of these data types in the PLC field are open and unified. The basic data types currently supported by K series PLCs are shown in the table below.

Data type	Description	Length (bit)	Range	default initial value
BOOL	boolean	1	true, false	false
BYTE	8 bit string	8	0 ~ (2^8-1)	0
WORD	16 bit string	16	0 ~ (2^{16}-1)	0
DWORD	32 bit string	32	0 ~ (2^{32}-1)	0
INT	Integer, signed	16	-2^{15} ~ (2^{15}-1)	0
DINT	Double integer, signed	32	-2^{31} ~ (2^{31}-1)	0
REAL	real	32	Adopt ANSI/IEEE754-1985 standard；	0.0

			$1.18*10^{-38} \sim 3.40*10^{38}$, 0, $-3.40*10^{38} \sim -1.18*10^{-38}$	

Table 3-1 Basic data types supported by Kinco-K series

The real number type in PLC follows the ANSI/IEEE754-1985 standard, which is the float (single precision) type in C language。

- **About rounding error of REAL type data**

The binary representation of real numbers cannot be very precise. Because REAL data occupies 4 bytes of space when stored, the significant number that can be represented is at most 6-7 digits, and the numbers outside the significant digits will be rounded, so rounding errors will occur.

The so-called significant digits are from the first digit on the far left of a data that is not 0 to the last digit, and all the digits in the middle are called the significant digits of the number. For example, 3.3 and 0.33 have 2 significant digits, and 3.30 and 0.330 have 3 significant digits.

According to the above description, for example, 1.23456 can be accurately represented in KPLC, but 1.2345678 cannot be accurately represented.。

- **About REAL“0.0”**

"0.0" may not be represented exactly in the PLC due to rounding errors. Therefore, according to the maximum number of significant digits that can be represented by real numbers, we make the following stipulations: Any real number within the range of [-0.000001, 0.000001] will be treated as "0.0" by the PLC.

- **Comparison of REAL**

When using comparison instructions (GT, GE, EQ, NE, LT, LE) we need to be careful: two real numbers cannot be compared exactly due to rounding errors. According to the previous description of "0.0", when the difference between the absolute values of the two real numbers is within [-0.000001, 0.000001], the PLC considers the two real numbers to be equal, otherwise they are not equal.

3.3 identifier

An identifier is a string of numbers, letters, and underscore characters, which must start with a letter or underscore. Programmers can use identifiers to define names for variables, programs, and so on.

3.3.1 Definition of Identifiers

The definition and use of identifiers must be based on the following principles:

- Must start with a letter or a single underscore character, followed by a certain number of numbers, letters or underscores.
- Identifiers are case-insensitive. For example, abc, ABC, aBC are the same identifier.
- The length of the identifier is only limited by each programming system. In KincoBuilder, the maximum length of an identifier is 16 characters.
- Keywords are not allowed for user-defined identifiers. A keyword is a standard identifier whose spelling form and purpose of use are clearly specified by IEC61131-3.。

3.3.2 use of identifiers

The language elements for which identifiers can be used in KincoBuilder are listed below:

- Program, function, function block name
- Variable
- Statement labels
- etc.

3.4 Introduction to Constants

A quantity whose value cannot be changed during the running of a program is called a constant. The properties of a constant are described by its value and data type. Depending on the data type, constants are

written in different formats. The following table lists the definitions and examples of various types of constants supported by the Kinco-K series.

Data type	Format	Range	Example
BOOL	true、false	Indicates true and false	false
BYTE	B# decimal number	B#0 ~ B#255	B#129
	B#2# binary number		B#2#10010110
	B#8# Octal number		B#8#173
	B#16#Hexadecimal number		B#16#3E
WORD	W# decimal number	W#0~ W#65535	W#39675
	2# binary number		2#100110011
	W#2#Binary number		W#2#110011
	8# octal number		8#7432
	W#8# octal number		W8#174732
	16# hexadecimal number		16#6A7D
	W#16#Hexadecimal number		W#16#9BFE
DWORD	DW# decimal number	DW#0 ~ DW#4294967295	DW#547321
	DW#2#Binary number		DW#2#10111
	DW#8# Octal number		DW#8#76543
	DW#16#hexadecimal number		DW#16#FF7D
INT	decimal number	-32768~32767	12345
	I# decimal number		I#-2345
	I#2#Binary number (2)		I#2#1111110
	I#8# Octal number (2)		I#8#16732
	I#16#hexadecimal number2		I#16#7FFF
DINT	DI# decimal number	DI#-2147483648~DI#2147483647	DI#8976540
	DI#2#Binary digit (2)		DI#2#101111
	DI#8# Octal number (2)		DI#8#126732
	DI#16#Hexadecimal number (2)		DI#16#2A7FF
REAL	decimal number with decimal point	$1.18*10^{-38} \sim 3.40*10^{38}$, 0, $-3.40*10^{38} \sim -1.18*10^{-38}$	1.0, -243.456
	Exponential form: xEy x: decimal number with decimal point y: integer.		-2.3E-23

Table 3-2 Definition of Constants

Tips:

(1)In IEC61131-3, the use of identifiers is case-insensitive. Therefore, it is legal to write format characters as uppercase or lowercase in the program, for example, W#234, dw#12345 are legal constants.

For details, see the section on the definition of identifiers.

(2) The binary, octal, and hexadecimal representations of INT and DINT constants use the standard complement representation in general-purpose computers, and the most significant bit (MSB) is the sign bit: MSB of 1 represents a negative number, and an MSB of 0 represents a positive number. For example, I#16#FFFF=-1, I#16#7FFF=32767, I#16#8000=-32768, etc.

3.5 Variable introduction

A variable whose value can change during the running of a program is called a variable.

A variable must have a name and occupy a certain storage unit in which the value of the variable is stored. In IEC61131-3, the storage location of variables can be specified by the user as a valid PLC memory address, or by the programming system.。

3.5.1 variable declaration

The use of variables must follow the "declare first, use later" principle. For the naming of variables, please refer to 3.3.1 Definition of Identifiers. When declaring a variable, you must specify a definite data type for it, and you must also specify its variable type.

Various variable types are defined in IEC61131-3. For example, a variable can be defined as a formal parameter of a POU; it can also be defined within a POU as a local variable of this POU; it can also be defined outside the POU and used as a global variable in the entire project scope. The following table describes the standard variable types supported by the Kinco-K series.

Variable type	storage permission		Description
	Outside	inside	

VAR	---	R/W	local variables. Can only be used within the POU in which it is defined, it is not visible outside this POU.
VAR_INPUT	W	R	Input variables. It is only readable as an input parameter of the POU within the POU where it is defined, and this variable is only writable when this POU is called.
VAR_OUTPUT	R	R/W	output variable. It is readable and writable as the output parameter of the POU in the POU where it is defined, and this variable is only readable when the POU is called.
VAR_IN_OUT	R/W	R/W	Input/output variables, which are a combination of VAR_INPUT and VAR_OUTPUT. This variable is readable and writable both within the POU in which it is defined and when this POU is called.
VAR_GLOBAL	R/W	R/W	global variables. Can be directly read and written by all POUs.

Table 3-3 Variable types supported by Kinco-K series

3.5.2 declare variable in KincoBuilder

In KincoBuilder, the declarations of various variables are carried out in the corresponding tables, which not only avoids the tedious input of the user, but also can strictly check the syntax of the user's input.

The declaration of global variables is done in the global variable table. The local variables and formal parameters of the POU are completed in the variable declaration table of the POU editing interface. If a global variable has the same name as a local variable, the use of the local variable in the program takes precedence.

For the specific usage method, please refer to the detailed introduction part of the interface in this manual.

3.5.3 Test of variables

When editing and compiling programs, KincoBuilder can automatically check the usage of variables, that is, determine whether the variable is processed according to its variable type and data type. This is also one of the important advantages of IEC61131-3. For example, assign a value of type BOOL to a variable of type WORD in the program, or perform assignment operation to a variable of type VAR_INPUT,

KincoBuilder will give an error warning to the user and prompt for modification.

Since the properties of a variable depend on its variable type and data type, this test can largely avoid errors due to variable usage.

3.6 Memory area and addressing mode

Each unit in the memory area has a clear, unique number, which is the physical address of the memory unit. A physical address is an abstract value that is not convenient to use in a program. Therefore, for the convenience of users, in the Kinco-K series, the memory is further divided into several areas of different types, and each area is re-addressed by bytes, and a direct address is assigned to each memory unit such as %I0.0, %VW20, etc. In this case, the direct address acts like a variable. In fact, in the description later in this book, expressions such as "direct address variable" and "variable %VW100" are often used.。

3.6.1 Types of memory regions and their characteristics

I	
Description	DI (switch input) image area At the beginning of each scan cycle, the CPU reads the status of all physical DI channels and writes these statuses into the I area for use by the user program.
interview method	Bit, byte, word, DWORD access
storage permission	read only
other	Allow forced, not power-down retention
Q	
Description	DO (switch output) image area At the end of each scan cycle, the CPU outputs all the values in the Q area to the physical DO channels.
interview method	Bit, byte, word, DWORD access
storage permission	readable and writable
other	Allow forced, not power-down retention
AI	
Description	(analog input) image area The AI expansion module samples the analog input signal and converts it to an integer value. At the beginning of each scan cycle, the CPU reads measured values from all AI expansion modules and writes them into the AI area for use by the user program.

interview method	Accessible by word (INT type)
storage permission	readable
Description	Allow forced, not power-down retention
AQ	
Description	AO (analog output) image area At the end of each scan cycle, the CPU outputs all the values in the AQ area to the physical AO channel.
interview method	Accessible by word (INT type)
storage permission	readable and writable
Description	Allow forced, not power-down retention
HC	
Description	High-speed counter area. Used to store the current count value of each high-speed counter.
interview method	Accessible as a DWORD (DINT type)
storage permission	readable
Description	Not allowed to force, can not keep power off
V	
Description	variable storage area. This area is relatively large and can be used to store large amounts of data.
interview method	Bit, byte, word, DWORD access
storage permission	readable and writable
Description	Allow force, allow power-down retention
M	
Description	Internal storage area. For some intermediate state or other data.
interview method	Compared with the V area, the access speed of the M area is faster and is more conducive to bit operations.
storage permission	Bit, byte, word, DWORD access
Description	readable and writable
SM	
Description	system storage. Used to store data. The user program can access some addresses in the SM area to obtain the current state information of the system, and can also use some addresses in the SM area to select and control some special functions of the CPU.
interview	Bit, byte, word, DWORD access

method	
storage permission	readable and writable
Description	Not allowed to force, can not keep power off
L	
Description	Description Local variable area. All local variables and input/output parameters of the POU are automatically assigned addresses in the L area. It is not recommended for users to directly operate the L area.
interview method	Bit, byte, word, DWORD access
storage permission	readable and writable
Description	Not allowed to force, can not keep power off

Table 3-4 Memory Area Allocation for Kinco-K Series PLCs

3.6.2 Direct addressing of memory regions

Users can use direct addresses to access the data in it, which is called direct addressing. It is important to note the difference between the concepts "direct address" and "data in direct address".

3.6.2.1 Direct address representation format

IEC61131-3 stipulates that all direct addresses must start with "%". When the user inputs the direct address during programming, he can not input "%", and KincoBuilder can add it automatically. For example, if the user inputs "I0.0", then KincoBuilder can automatically convert it to "%I0.0".

The direct addressing method of each memory area is shown in the following list. x and y in the table represent decimal numbers.

- I area

bit addressing	Format	**%I**x.y
	describe	x: Byte address, indicating the number (address) of the byte where the memory unit is located in the I area. y: Bit address, which indicates the digit of byte x. The range is 0~7.
	type of data	BOOL

	Example	%I0.0 %I0.7 %I5.6
byte addressing	Format	%IBx
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the I area.
	type of data	BYTE
	Example	%IB0 %IB1 %IB10
word addressing	Format	%IWx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the I area. Since the data length of the WORD type is 2 bytes, x must be an even number.
	type of data	WORD、INT
	Example	%IW0 %IW2 %IW12
DWORD addressing	Format	%IDx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the I area. Since the data length of the DWORD type is 4 bytes, x must be an even number.
	type of data	DWORD、DINT
	Example	%ID0 %ID4 %ID12

- **Q** area

bit addressing	Format	**%Q**x.y
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the Q area. y: Bit address, which indicates the digit of byte x. The range is 0~7.
	type of data	BOOL
	Example	%Q0.0 %Q0.7 %Q5.6
byte addressing	Format	%QBx
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the Q area.
	type of data	BYTE
	Example	%QB0 %QB1 %QB10
word addressing	Format	%QWx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the Q area. Since the data length of the WORD type is 2 bytes, x must be an even number.
	type of data	WORD、INT
	Example	%QW0 %QW2 %QW12
DWORD addressing	Format	%QDx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the Q area.

		Since the data length of the DWORD type is 4 bytes, x must be an even number.
	type of data	DWORD、DINT
	Example	%QD0 %QD4 %QD12

- **M** area

bit addressing	Format	**%M**x.y
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the M area. y: Bit address, which indicates the digit of byte x. The range is 0~7.
	type of data	BOOL
	Example	%M0.0 %M0.7 %M5.6
byte addressing	Format	%MBx
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the M area.
	type of data	BYTE
	Example	%MB0 %MB1 %MB10
word addressing	Format	%MWx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the M area. Since word data is 2 bytes long, x must be an even number.
	type of data	WORD、INT
	Example	%MW0 %MW2 %MW12
DWORD addressing	Format	%MDx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the M area. Since DWORD data is 4 bytes long, x must be an even number.
	type of data	DWORD、DINT
	Example	%MD0 %MD4 %MD12

- **V** area

bit addressing	Format	**%V**x.y
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the V area. y: Bit address, which indicates the digit of byte x. The range is 0~7.
	type of data	BOOL
	Example	%V0.0 %V0.7 %V5.6
byte addressing	Format	%VBx
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the V area.
	type of data	BYTE
	Example	%VB0 %VB1 %VB10
word	Format	%VWx

addressing	describe	x: Byte address, that is, the address of the first byte of the memory unit in the V area. Since word data is 2 bytes long, x must be an even number.
	type of data	WORD、INT
	Example	%VW0 %VW2 %VW12
DWORD addressing	Format	**%VD**x or **%VR**x
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the V area. Since DWORD data is 4 bytes long, x must be an even number.
	type of data	DWORD、DINT (**%VD**x)；REAL (**%VR**x)
	Example	%VD0、%VD12；REAL type: %VR0、%VR4

- **SM** area

bit addressing	Format	**%SM**x.y
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the SM area. y: Bit address, which indicates the digit of byte x. The range is 0~7.
	type of data	BOOL
	Example	%SM0.0 %SM0.7 %SM5.6
byte addressing	Format	%SMBx
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the SM area.
	type of data	BYTE
	Example	%SMB0 %SMB1 %SMB10
word addressing	Format	%SMWx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the SM area. Since word data is 2 bytes long, x must be an even number.
	type of data	WORD、INT
	Example	%SMW0 %SMW2 %SMW12
DWORD addressing	Format	%SMDx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the SM area. Since DWORD data is 4 bytes long, x must be an even number.
	type of data	DWORD、DINT
	Example	%SMD0 %SMD4 %SMD12

- **L** area**(Note: It is not recommended that users use direct addresses to access the L area)**

bit addressing	Format	**%L**x.y
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the L area. y: Bit address, which indicates the digit of byte x. The range is 0~7.
	type of data	BOOL

	Example	%L0.0 %L0.7 %L5.6
byte addressing	Format	%LBx
	describe	x: Byte address, that is, the number (address) of the byte where the memory unit is located in the L area.
	type of data	BYTE
	Example	%LB0 %LB1 %LB10
word addressing	Format	%LWx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the L area. Since word data is 2 bytes long, x must be an even number.
	type of data	WORD、INT
	Example	%LW0 %LW2 %LW12
DWORD addressing	Format	%LDx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the L area. Since DWORD data is 4 bytes long, x must be an even number.
	type of data	DWORD、DINT、REAL
	Example	%LD0 %LD4 %LD12

- **AI area**

word addressing	Format	%AIWx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the AI area. Since word data is 2 bytes long, x must be an even number.
	type of data	INT
	Example	%AIW0 %AIW2 %AIW12

- **AQ area**

word addressing	Format	%AQWx
	describe	x: Byte address, that is, the address of the first byte of the memory unit in the AQ area. Since word data is 2 bytes long, x must be an even number.
	type of data	INT
	Example	%AQW0 %AQW2 %AQW12

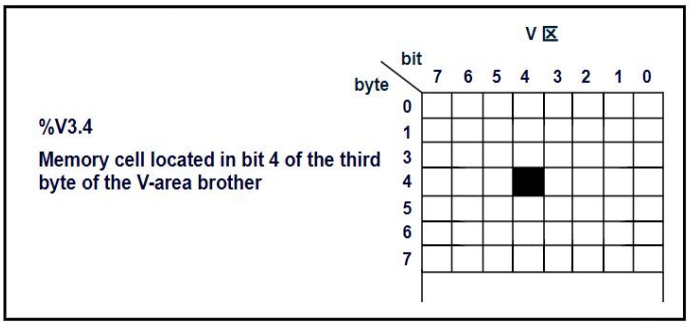
- **HC area**

word addressing	Format	%HCx
	describe	x: The number of the high-speed counter.
	type of data	DINT
	Example	%HC0 %HC1

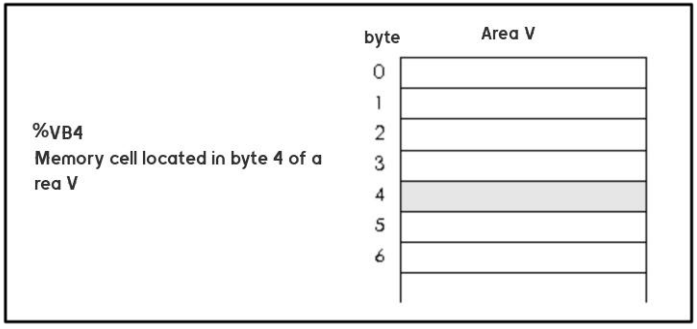
3.6.2.2 Mapping between direct addresses and memory cells

Every legal direct address corresponds to a memory location in the CPU. The operation of the direct address in the program is to operate the corresponding memory unit. The following will take the V area as an example to illustrate the mapping relationship between direct addresses and memory cells.

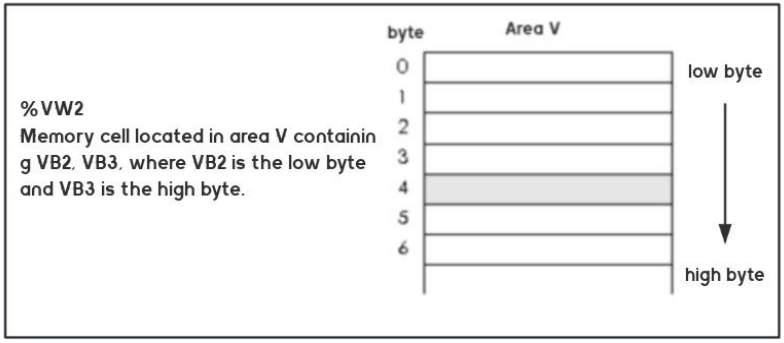
• Bit address:



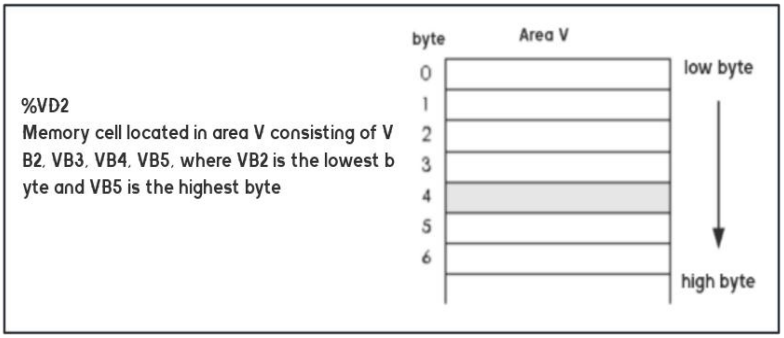
• Byte address:



• Word address:



• DWORD address:



Take the V area as an example to illustrate the relationship between bytes, words and DWORDs:

Address	Addr	Addr+1	Addr+2	Addr+3
BYTE	VB0	VB1	VB2	VB3
BOOL	0...7 V0.0...V0.7	0...7 V1.0...V1.7	0...7 V2.0...V2.7	0...7 V3.0...V3.7
WORD	VW0 0...7	8...15	VW2 0...7	8...15
DWORD	0...7	8...15	VD0 16...23	24...31
REAL			VR0	

As shown in the figure above, all directly addressed storage areas of K series PLC are stored in bytes. Every 8 BITs form a BYTE. Every 2 BYTEs form a WORD, and every two WORDs form a DWORD. The storage

method in little-endian mode is adopted, that is, the high address stores the high bits of the variable, and the low address stores the low bits of the variable.

Note that the datastores may overlap when accessed by different data types. For example, the value of %VW0 is 3, then the value of %VB0 is also 3, the value of %V0.0 is TRUE, and the value of %VX0.1 is also TRUE. For a better understanding of how storage is done, take a look at the diagram below.

	Address	Number	Format	Value 1	Value 2	Value 3	Value 4
1	%VB0	4	HEX	B#16#78	B#16#56	B#16#34	B#16#12
2							
3	%VW0	2	HEX	W#16#5678	W#16#1234		
4							
5	%VD0	1	HEX	DW#16#12345678			
6							
7	%VR0	1	DEC(signed)	5.690e-028			
8							
9	%V0.0	32	DEC(signed)	FALSE	FALSE	FALSE	TRUE
10	%V0.4			TRUE	TRUE	TRUE	FALSE
11	%V1.0			FALSE	TRUE	TRUE	FALSE
12	%V1.4			TRUE	FALSE	TRUE	FALSE
13	%V2.0			FALSE	FALSE	TRUE	FALSE
14	%V2.4			TRUE	TRUE	FALSE	FALSE
15	%V3.0			FALSE	TRUE	FALSE	FALSE
16	%V3.4			TRUE	FALSE	FALSE	FALSE

3.6.3 Indirect addressing of memory regions

A pointer is a 32-bit variable that stores the physical address of a memory unit. The user can use the pointer to access the data in the corresponding memory unit, which is called indirect addressing.

In the Kinco-K series, only direct address variables (double word length) in the V area are allowed to be used as pointers. In addition, only indirect addressing of the V area using pointers is allowed, but indirect addressing of bit variables is not supported.

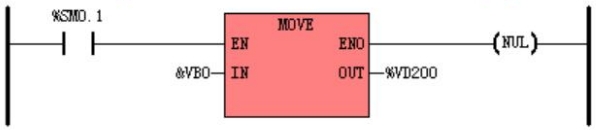
3.6.3.1 create pointer

In order to achieve indirect addressing of a memory unit, its pointer must be established first. At this

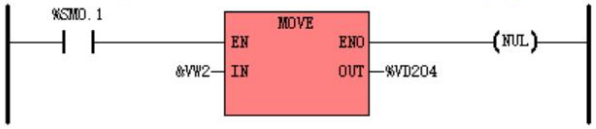
time, you need to use the address operator "&", for example, &VB100 represents the physical address of VB100。

The way to create a pointer is to use the address operator to obtain the physical address of a memory unit, and use the MOVE instruction to write it into another direct address variable as a pointer. E.g:

```
(* Network 0 *)
(* Create pointer VD200, which holds the physical address of VB0 *)
```




```
(* Network 1 *)
(* Create pointer VD204, which holds the physical address of VW2 *)
```



3.6.3.2 Access data using pointers

Prefix an operand in an instruction with "*" to indicate that the operand is a pointer. "*" is a pointer operator, adding "*" before a pointer represents the direct address variable pointed to by the pointer. When using a pointer as an operand of an instruction, you need to pay attention to the data type of the variable pointed to by the pointer. E.g:

```
(* Network 0 *)
(* Create pointer VD200, which holds the physical address of VB0
Because pointer VD200 points to the direct address variable VB0, *VD200 represents VB0
Assign the address of VB0 to VB1 *)
```

3.6.3.3 change the value of the pointer

Since a pointer is a 32-bit variable, its value can be modified using instructions such as addition, subtraction, etc. It should be noted that every time the value of the pointer changes by 1, the direct address it points to changes by 1 byte accordingly. When modifying the value of a pointer, you need to pay attention to the data type of the variable it points to:

• If it points to a variable of byte length (BYTE type), the change amount of the pointer value can be any double integer.

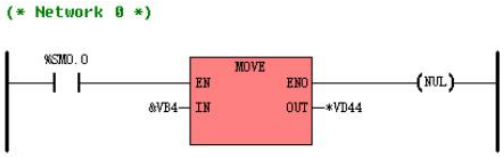
• If it points to a variable of word length (INT or WORD type), the change amount of the pointer value must be a multiple of 2.

• If it points to a variable of double word length (DINT, DWORD, or REAL type), the value of the pointer must be changed by a multiple of 4.

3.6.3.4 Notes on using pointers

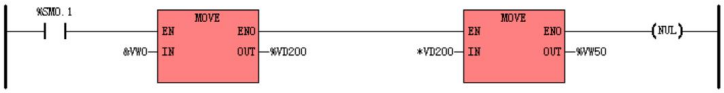
The validity of the pointer is guaranteed by the user program itself. The pointer is more flexible, so the user must be careful when using it. If the pointer points to an illegal variable address in the program, it will lead to unexpected results.

Kinco-K series only supports one pointer and address, and multiple applications are illegal. For example, the following instruction is illegal:



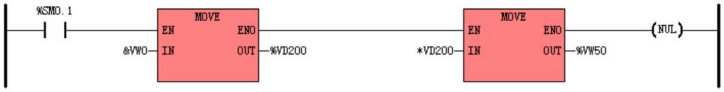
3.6.3.5 Indirect addressing example

```
(* Network 0 *)
(* Create pointer VD200, which holds the physical address of VW0
Assign the address of VW0 to VW50, because the pointer VD200 points to the direct address variable VW0, so *VD200 represents VW0 *)
```




```
(* Network 1 *)
(* The value of pointer VD200 is increased by 2, so the direct address it points to is increased by 2 bytes
This means that pointer VD200 points to VW2 and assigns the value of VW2 to VW52 *)
```

```
(* Network 0 *)
(* Create pointer VD200, which holds the physical address of VW0
Assign the address of VW0 to VW50, because the pointer VD200 points to the direct address variable VW0, so *VD200 represents VW0 *)
```




```
(* Network 1 *)
(* The value of pointer VD200 is increased by 2, so the direct address it points to is increased by 2 bytes
This means that pointer VD200 points to VW2 and assigns the value of VW2 to VW52 *)
```

3.6.4 The address range of the memory area

Kinco-K series has several different specifications of CPU. The address range of the memory area in each type of CPU is different, and the address beyond the allowable range is illegal. The following is a detailed description of each series of PLC memory areas.

		K6 series and KM series (KS101M、K209M)
I	length (bytes)	32
	bit address	%I0.0 --- %I31.7
	byte address	%IB0 --- %IB31
	word address	%IW0 ---%IW30

	DWORD address	%ID0 ---%ID28
Q	length (bytes)	32
	bit address	%Q0.0 --- %Q31.7
	byte address	%QB0 --- %QB31
	word address	%QW0 ---%QW30
	DWORD address	%QD0 ---%QD28
AI	length (bytes)	128
	word address	%AIW0 --- %AIW126
AQ	length (bytes)	128
	word address	%AQW0 --- %AQW126
HC	length (bytes)	32
	DWORD address	%HC0---%HC7
V	length (bytes)	16384
	bit address	%V0.0 ---%V16383.7
	byte address	%VB0 --- %VB16383
	word address	%VW0 --- %VW16382
	DWORD address	%VD0 --- %VD16380 %VR0 --- %VR16380
M	length (bytes)	4096
	bit address	%M0.0 --- %M4095.7
	byte address	%MB0 --- %MB4095
	word address	%MW0 --- %MW4094
	DWORD address	%MD0 --- %MD4092
SM	length (bytes)	2048
	bit address	%SM0.0 --- %SM2047.7
	byte address	%SMB0 --- %SMB2047
	word address	%SMW0 --- %SMW2046
	DWORD address	%SMD0 --- %SMD2044
L	length (bytes)	2048
	bit address	%L0.0 --- %L2047.7
	byte address	%LB0 --- %LB2047
	word address	%LW0 --- %LW2046
	DWORD address	%LD0 --- %LD2044

Table 3-3 Memory range of K6 series and KM series (KS101M, K209M)

		MK series, KS series (except KS101M), and KW series
I	length (bytes)	32

	bit address	%I0.0 --- %I31.7
	byte address	%IB0 --- %IB31
	word address	%IW0 ---%IW30
	DWORD address	%ID0 ---%ID28
Q	length (bytes)	32
	bit address	%Q0.0 --- %Q31.7
	byte address	%QB0 --- %QB31
	word address	%QW0 ---%QW30
	DWORD address	%QD0 ---%QD28
AI	length (bytes)	128
	word address	%AIW0 --- %AIW126
AQ	length (bytes)	128
	word address	%AQW0 --- %AQW126
HC	length (bytes)	16
	DWORD address	%HC0———%HC3
V	length (bytes)	4096
	bit address	%V0.0 ---%V4095.7
	byte address	%VB0 --- %VB4095
	word address	%VW0 --- %VW4094
	DWORD address	%VD0 --- %VD4092 %VR0 --- %VR4092
M	length (bytes)	1024
	bit address	%M0.0 --- %M1023.7
	byte address	%MB0 --- %MB1023
	word address	%MW0 --- %MW1022
	DWORD address	%MD0 --- %MD1020
SM	length (bytes)	300
	bit address	%SM0.0 --- %SM299.7
	byte address	%SMB0 --- %SMB299
	word address	%SMW0 --- %SMW298
	DWORD address	%SMD0 --- %SMD296
L	length (bytes)	272
	bit address	%L0.0 --- %L271.7
	byte address	%LB0 --- %LB271
	word address	%LW0 --- %LW270
	DWORD address	%LD0 --- %LD268

Table 3-4 Memory range of MK series, KS series (except KS101M), and KW series

		K5 series
I	length (bytes)	32
	bit address	%I0.0 --- %I31.7
	byte address	%IB0 --- %IB31
	word address	%IW0 ---% IW30
	DWORD address	%ID0 --- %ID28
Q	length (bytes)	32
	bit address	%Q0.0 --- %Q31.7
	byte address	%QB0 --- %QB31
	word address	%QW0 --- %QW30
	DWORD address	%QD0 --- %QD28
AI	length (bytes)	64
	word address	%AIW0 --- %AIW62
AQ	length (bytes)	64
	word address	%AQW0 -- %AQW62
HC	length (bytes)	16
	DWORD address	%HC0———%HC3
V	length (bytes)	4096
	bit address	%V0.0 ---%V4095.7
	byte address	%VB0 --- %VB4095
	word address	%VW0 --- %VW4094
	DWORD address	%VD0 --- %VD4092 %VR0 --- %VR4092
M	length (bytes)	1024
	bit address	%M0.0 --- %M1023.7
	byte address	%MB0 --- %MB1023
	word address	%MW0 --- %MW1022
	DWORD address	%MD0 --- %MD1020
SM	length (bytes)	300
	bit address	%SM0.0 --- %SM299.7
	byte address	%SMB0 --- %SMB299
	word address	%SMW0 --- %SMW298
	DWORD address	%SMD0 --- %SMD296
L	length (bytes)	272

	bit address	%L0.0 --- %L271.7
	byte address	%LB0 --- %LB271
	word address	%LW0 --- %LW270
	DWORD address	%LD0 --- %LD268

Table 3-5 Memory range of K5 series

		K2 series (except K209M)
I	length (bytes)	16
	bit address	%I0.0 --- %I15.7
	byte address	%IB0 --- %IB15
	word address	%IW0 ---%IW14
	DWORD address	%ID0 ---%ID12
Q	length (bytes)	16
	bit address	%Q0.0 --- %Q15.7
	byte address	%QB0 --- %QB15
	word address	%QW0 ---%QW14
	DWORD address	%QD0 ---%QD12
AI	length (bytes)	32
	word address	%AIW0 --- %AIW30
AQ	length (bytes)	32
	word address	%AQW0 --- %AQW30
HC	length (bytes)	16
	DWORD address	%HC0———%HC3
V	length (bytes)	4096
	bit address	%V0.0 ---%V4095.7
	byte address	%VB0 --- %VB4095
	word address	%VW0 --- %VW4094
	DWORD address	%VD0 --- %VD4092 %VR0 --- %VR4092
M	length (bytes)	1024
	bit address	%M0.0 --- %M1023.7
	byte address	%MB0 --- %MB1023
	word address	%MW0 --- %MW1022
	DWORD address	%MD0 --- %MD1020
SM	length (bytes)	300
	bit address	%SM0.0 --- %SM299.7
	byte address	%SMB0 --- %SMB299

	word address	%SMW0 --- %SMW298
	DWORD address	%SMD0 --- %SMD296
L	length (bytes)	272
	bit address	%L0.0 --- %L271.7
	byte address	%LB0 --- %LB271
	word address	%LW0 --- %LW270
	DWORD address	%LD0 --- %LD268

Table 3-6 Memory range of K2 series (except K209M)

3.6.5 About function blocks and function block instances

3.6.5.1 Standard function blocks defined in IEC61131-3

The following standard function blocks are defined in IEC61131-3:

- Timer: TP --- pulse timer; TON --- on-delay timer; TOF --- off-delay timer.
- Counter: CTU --- up counter; CTD --- down counter; CTUD --- up/down counter.
- Bistable flip-flop: SR --- SR flip-flop; RS --- RS flip-flop.
- Edge detection: R_TRIG --- rising edge detection; F_TRIG --- falling edge detection.

3.6.5.2 instantiation of FB

In IEC61131-3, the concept of "FB instantiation" is particularly important.

The so-called instantiation means that the user creates a variable by specifying the variable name and its data type in the variable declaration section.

FBs also need to be instantiated first like variables. Direct calls to FBs are not allowed in the program, but only instances of FBs. Visually speaking, a data type (such as INT type) cannot be read and written in a program, but only specific variables declared as this data type (such as INT type) can be read and written. As shown in the figure below, only T1 can be called and accessed in the program.

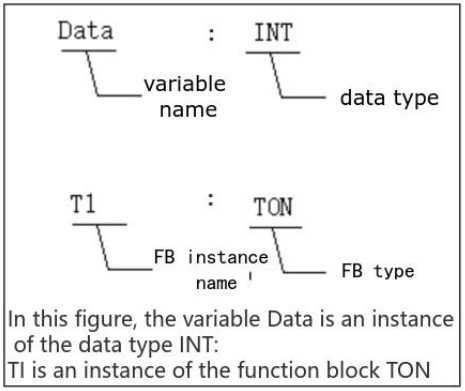


Figure 3-1 Example of instantiation

3.6.5.3 FB instance store

The Kinco-K series allocates an instance storage area in memory for each FB type. When an FB instance is defined, KincoBuilder will automatically allocate an independent storage unit for the instance in the corresponding storage area.

The following table describes the FB instance storage area of the Kinco-K series

T	
describe	Timer area, in which instances of TON, TOF, and TP can be allocated.
interview method	Used to store the state value and current timing value of all timer instances.
storage permission	Direct access to the status value and current timing value of the timer
others	readable
C	
describe	Counter area, in which instances of CTU, CTD, CTUD can be allocated.
interview method	Used to store the state value and current count value of all counter instances.
storage permission	Direct access to the status value and count value of the counter
others	readable
RS	
describe	The RS trigger area, within which instances of RS can be allocated.
interview method	Used to store state values for all RS instances.
storage	Direct access to RS state values

permission	
others	readable
SR	
describe	SR trigger area, in which instances of SR can be allocated.
interview method	Used to store state values for all SR instances.
storage permission	Direct access to SR status values
others	readable

Table 3-6 Storage areas for FB instances

3.6.6 Naming and use of FB instances

The instance of FB follows the principle of "declare first, use later".

For the convenience of users, KincoBuilder has specially made the following processing: The naming of FB instances follows the common methods of traditional PLC, such as T0, C3 and so on. The user does not need to manually input the declaration statement of the FB instance, but only needs to call the legal function block instance in the program, and the software will automatically generate the declaration statement for the instance called by the user in the global variable table.

The direct addressing method of the FB instance memory area is shown in the following list.

- T

direct addressing	Format	**T**x
	describe	x: Timer number, decimal number.
	type of data	BOOL --- Status value of the timer INT --- The current count value of the timer. Tx has both of the above two meanings, but the user only needs to use the instance name in the program, and its meaning will be automatically recognized by the software.
	Example	T0、T5、T20

- C

direct addressing	Format	**C**x
	describe	x: Counter number, decimal number.
	type of data	BOOL --- Status value of the counter INT --- The current tick value of the counter.

		Cx has both of the above two meanings, but the user only needs to use the instance name in the program, and its meaning will be automatically recognized by the software.
	example	C0、C5、C20

- RS

direct addressing	Format	RSx
	describe	x: RS trigger number, decimal number.
	type of data	BOOL --- the state value of the RS flip-flop
	Example	RS0、RS5、RS10

- SR

direct addressing	Format	SRx
	describe	x: SR trigger number, a decimal number.
	type of data	BOOL --- state value of SR flip-flop
	Example	SR0、SR5、SR10

3.6.7 Range of FB instance memory area

A storage area is allocated for each FB type in the memory of the Kinco-K series CPU. When an instance of a certain FB is defined, KincoBuilder will automatically allocate an independent storage unit for each instance in the storage area corresponding to the FB type.

The following table describes the instance memory area allocated for each FB by the Kinco-K series CPU.

T	Quantity allowed	256
	scope	T0 --- T255
	time base	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T255: 100ms
	maximum timing	32767*time base
C	Quantity allowed	256
	scope	C0 --- C255
	maximum count	32767
RS	Quantity allowed	32
	scope	RS0 --- RS31
SR	Quantity allowed	32

	scope	SR0 --- SR31

Table 3-7 FB instance memory allocation

Chapter 4 Basic functions of using KincoBuilder

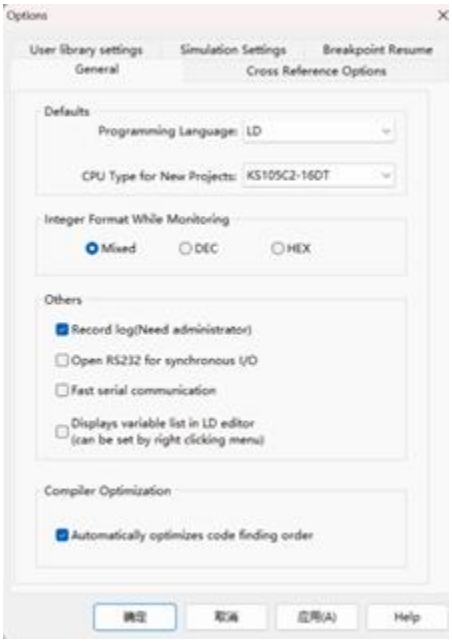
This chapter describes the functions and usage of each part of KincoBuilder in detail. On the basis of mastering the basic concepts introduced in the previous chapter, users can quickly recognize and understand the specific functions and operations of KincoBuilder by reading this chapter.

The use of the LD editor and the IL editor will involve many grammars in the IEC61131-3 standard, which are not introduced in this chapter, and the related content and grammar will be described in detail in the next chapter.

4.1 KincoBuilder settings

Before using KincoBuilder formally, users need to set some software default values, the setting results will be saved in the current computer, and KincoBuilder will automatically read and use these setting values when running. Execute the menu instruction 【Tools】→【Software Setting…】, and the "Software Setting" dialog box will pop up.

① 【General】page



➢ Project property default value

Some properties of the KincoBuilder project are allowed to be set here.

• **Default programming language**: Select the default programming language when the program is just created, IL or LD.

• **Default CPU Model:** Select the default CPU model used in the hardware configuration when the project is just created.

➢ **Integer display format during online monitoring**

Select the display format of integer data in the editor during online monitoring. There are three options as follows:

- Mixed: INT, DINT type in decimal format, BYTE, WORD, DWORD type data in hexadecimal format.
- Decimal: All integers are displayed in decimal format.
- Hex: All integer data are displayed in hexadecimal format.

➢ **Others**

- **Logging**

If this option is selected, a new KincoPlcLog subdirectory will be created in the installation directory when KincoBuilder is running. In this directory, a log file will be generated every day, and all operations of KincoBuilder will be recorded in the log, which is convenient for tracking and analysis when problems occur. If the operating system is Win7 or Win8, KincoBuilder must be run with administrator privileges to record logs. The log only records the operations of the day, and the old records will be automatically deleted.

- **Open serial port in synchronous mode**

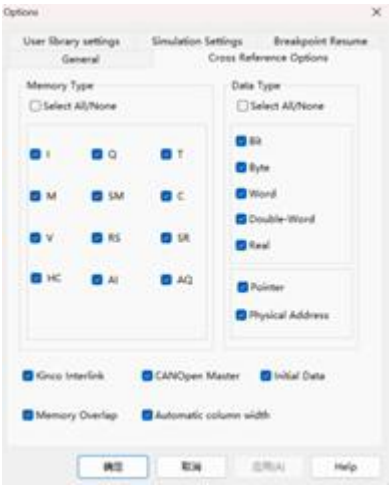
When some users use some USB to RS232 converters under Windows 7 or later, the communication with PLC may fail. Usually this is caused by the compatibility of the converter driver.

If this kind of problem occurs, select "Open serial port synchronously", and then click the "OK" button to exit. In the future, KincoBuilder will operate the serial port in a synchronous manner, which can solve this problem in most cases.

- **Speed up the serial communication process**

When this option is checked when the PC and PLC use serial communication, the time for KincoBuilder to judge that there is no communication between the PC and the PLC will be shortened.

② 【Cross Reference Options】page

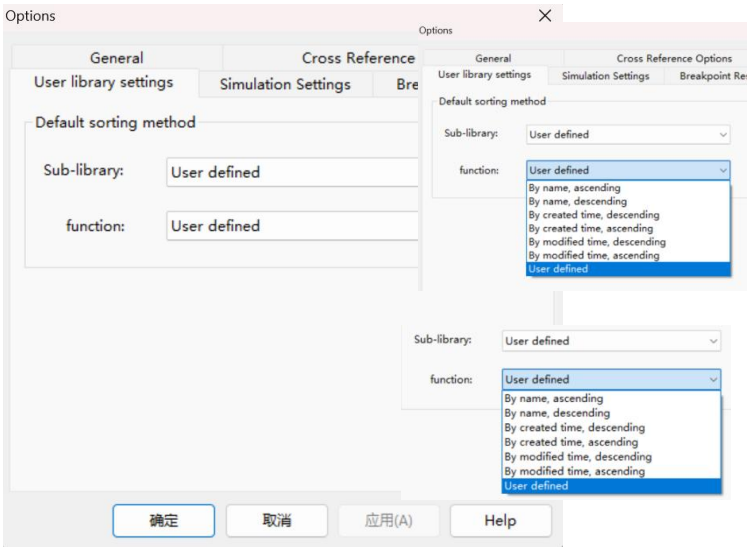


After compiling, the variables used in the user project and their used positions can be listed in the cross-reference table.

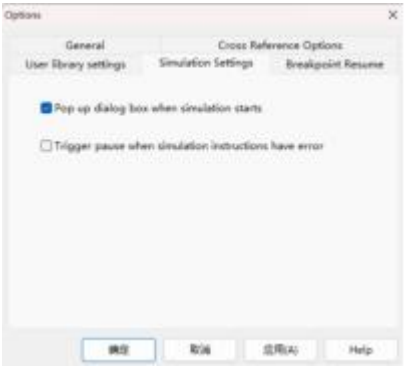
KincoBuilder displays all variables used by default.

In this option page, you can choose to display only the variables of a certain [memory area] and [data type] used in the project, which is convenient for users to query. For example, the option in the figure above specifies that only the bit variables of the M area used by the user project are displayed in the cross list.

③ 【User Library Settings】page



④ 【Offline Simulation Settings】page



4.2 floating window

The project manager, instruction set window, information output window, and PLC module list window are all designed as floating windows, which can be docked on any side of the KincoBuilder main window.

- Click the icon in the upper right corner of the floating window to automatically hide the window. In the auto-hide state, the window automatically shrinks to an icon and stays on the edge of the main

window. Now point the mouse to the icon and hold it for a moment, the window will automatically appear, and then if the mouse is placed outside the window, it will shrink to the edge of the screen.

- In the auto-hide state, click the icon in the upper right corner of the window, the window will be unhide state and re-dock to the last docking position.

4.3 PLC Hardware settings

Hardware configuration is necessary information in a project, so it is recommended that users complete the hardware configuration in the project first.

When the user creates a new project, KincoBuilder will automatically add a default CPU module whose model is the [default CPU model] specified by the user in the "Software Setting" dialog box. Users can modify all configuration information according to actual needs.

The hardware configuration information must be downloaded to the CPU to take effect. At each cold start (re-power-on), the CPU will first detect the information of the actual connected modules, and compare the detected actual information with the stored configuration information. If it is inconsistent, the PLC will enter the STOP state, otherwise it will enter the normal operation state.

The style of the hardware configuration window is as shown below. As can be seen from the figure, the window can be divided into two parts:

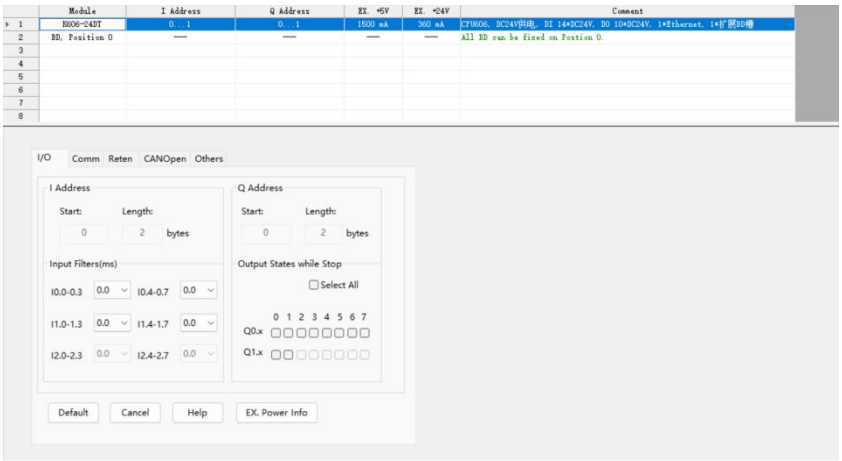


Figure 4-2 Hardware configuration window

- Module list

In the upper part of the hardware configuration window, the PLC modules used in the project are listed in table form, which we call the module list. The module list expresses a real PLC control system: the arrangement order of each module in the table should be consistent with the connection order of the actual modules on the expansion bus.

- Module parameter configuration window

In the lower part of the hardware configuration window, there are all configuration parameters of the selected module in the PLC module list, which we call the module parameter configuration window.

4.3.1 How to enter the hardware configuration window?

There are two ways to enter the hardware configuration window:

- Double-click the [PLC Hardware Configuration] node under the [Resources] group in the project manager;
- Right-click on the [PLC Hardware Configuration] node, and then execute the [Open…] instruction in the pop-up menu.

4.3.2 Copy and paste hardware configuration information in different projects

In KincoBuilder, users are allowed to copy and paste [Hardware Configuration] information in different projects. Note that this function operates various configuration information, such as CANOpen configuration, communication port configuration, etc., and will not copy or paste specific CPU models, so operations can also be performed between different CPU models. In addition, each project to be operated must be opened with KincoBuilder before operation. This function does not conflict with the copy and paste of the ladder diagram and the copy and paste of the instruction table, and can be used at the same time.。

If the user wants to transplant or inherit the CANOpen network configuration, communication port

configuration and other information in the old project, then this copy and paste [hardware configuration] function will be very useful.

There are two ways to use this function::

- Execute the instructions of [Copy Hardware Configuration] and [Paste Hardware Configuration] in the [Edit] menu.
- Right-click on the [PLC Hardware Configuration] node in the [Project Manager] tree, and then execute the [Copy Hardware Configuration] and [Paste Hardware Configuration] instructions.

4.3.3 Add and remove modules

➢ Add modules

Users can add a module as following steps:

① Click the position of the module to be added in the module list to place the focus on the row. If a module already exists in this row, the existing module must be deleted first, and then a new module can be added.

② Double-click the module to be added in the [PLC Module List] window on the left. The first row (the row with the serial number 1) can only be added to the CPU module, and the other rows can only be added to the expansion I/O and function modules. Blank lines are not allowed between modules. If there is a blank line, the software does not allow adding modules after it, and an error will be prompted when saving and compiling.

➢ Delete module

Users can delete a module using any of the following methods:

- Click the module to be deleted in the module list, and then press the Delete key to delete it.
- Right-click the module to be deleted in the module list, and execute the [Delete Module] instruction in the pop-up menu.。

4.3.4 Configure module parameters

Each module of the Kinco-K series provides a variety of parameter and option settings to suit different specific applications, including the I/O address of the module, the signal type of each channel of the analog module, etc. The KincoBuilder software allows the user to customize all these parameters and options.

In the module list, click any module with the mouse and select it, the parameter configuration window of the module will appear below, and the user can modify the parameters of the module in this module parameter configuration window. Of course, the user can also use the Up and Down arrow keys on the keyboard to change the position of the selected module in the module list

There are two public buttons on the right side of the module parameter configuration window: **[Default], [Cancel]**

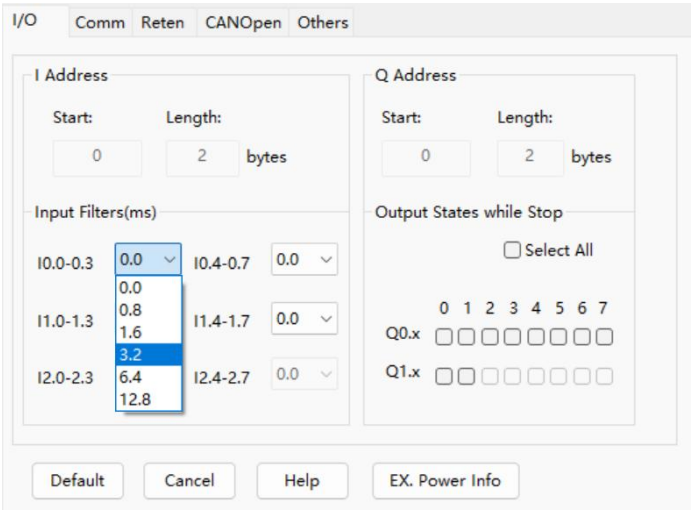
- **[Default]**: Clicking this button will cancel the user's input on the current page, and the software will automatically assign parameters to this module。
- **[Cancel]**: Clicking this button will cancel the user's input on the current page and restore the original configuration of this module

Note: The addresses of each module in the same memory area (I, Q, AI or AQ) are not allowed to overlap!

4.3.4.1 CPU parameter configuration

①【I/O setting】Page

In this page, you can complete the configuration of the I/O points on the CPU body. As shown below.



➢ Input area: used to configure the parameters of the DI point on the CPU body

- **Start address**: The start byte of the address occupied by the DI part in the I area is fixed at 0.
- **Input filtering**: Defines the filtering time required for DI signal sampling, in ms. Every 4 points are grouped into a group for definition. The DI signal from the scene must keep at least the set filter time, and then the CPU will consider the input valid and update the corresponding DI image area, otherwise the CPU will not respond to the input. This helps to filter out the input noise and enhance the anti-interference ability of the system. The shorter the filter time, the faster the CPU will respond to that input. In practical applications, the user should set the filtering time according to the specific situation of the site. All DI points are unfiltered by default.

➢ Output area: used to configure the parameters of the DO point integrated in the CPU body.

• **Start address**: Start address: the start byte of the address occupied by the DO part in the Q area, fixed at 0.

• **Output hold**: Output hold: Set the output state of each DO point when the CPU is in STOP state.

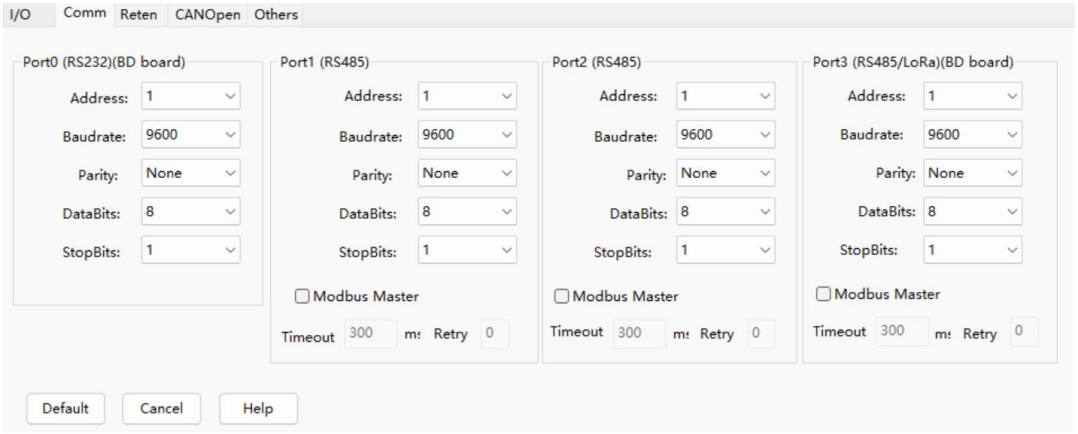
If the check box corresponding to a DO point is selected, when the CPU is in STOP, the output of this point is 1.

If the check box corresponding to a DO point is not selected, when the CPU is in STOP, the output of this point is 0.

This function is very meaningful for safety interlocks required after CPU shutdown.。

②[Communication Settings]page

Used to configure the parameters of the serial communication port on the CPU body. As shown below.



Different series and models of CPUs provide different types and quantities of serial ports. When the user adds a certain CPU, the type and quantity of serial ports provided by the CPU of this model will be automatically adjusted in the communication setting page. For example, if the user adds KS105-16DT, it has one RS232 port (PORT0) and one RS485 communication port (PORT1). Therefore, in the communication setting page, the user is allowed to configure the parameters of the two communication ports "PORT0 (RS232)" and "PORT1 (RS485)", while the parameters of other ports are all grayed out, indicating that they cannot be operated.

➢ PORT0

There is "RS232" in parentheses after the text "PORT0", indicating that PORT0 is an RS232 communication port。

• PLC **station number**: Specify a unique station number for PORT0. This station number is the station number when it is used as a Modbus RTU slave station.

• **Baud rate**: Select baud rate: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

• **Parity Check**: Select the parity check method: No Check, Odd Check, Even Check.

• **Data bits**: Select Data Bits: 8.

• **Stop bits**: Select Stop Bits: 1.

➢ PORT1

There is "RS485" in parentheses after the text "PORT1", indicating that PORT1 is an RS485 communication port.

• **PLC station number**: Specify a unique station number for PORT0. This station number is the station number when it is used as a Modbus RTU slave station.

• **Baud Rate**: Select the baud rate: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

• **Parity Check:** Select the parity check method: No Check, Odd Check, Even Check.

• **Data Bits:** Select Data Bits: 8.

• **Stop Bits:** Select Stop Bits: 1.

• **As Modbus master station:** If this item is selected, then the PORT port will be used as the master station of Modbus RTU to communicate, and at the same time

[PLC station number] on the panel will be invalid.

• **Timeout:** Set the time-out time for Modbus master communication, in ms.

• **Retry:** The number of times that the Modbus master will continue to retry communication after receiving an incorrect response from the slave.。

When the master sends an instruction, a communication error occurs in the following cases:

- If no response is received from the slave within the defined timeout period, a communication timeout error will be generated;
- If the master station receives the wrong response from the slave station, it will retry the communication and re-send the “retry” instruction at most. If the last time does not receive a correct response from the slave station, the master station will continue to wait for the timeout time and generate a communication timeout error.

③[Data Retention] page

Configure the data retention area on this page. When the CPU is powered off, the data in the data retention area will be protected by the backup battery for use when the power is turned on again. Keep at room temperature for not less than 3 months.

Note: The memory areas of "Initialize Data Table", "Data Retention" and "Data Backup" should be kept from overlapping. Because these data are restored after power-up before entering the main loop. The sequence is: restore the memory data defined in "Data Retention", assign initial values to the memory area defined in "Initialization Data Table", and restore the data permanently saved by the user using instructions.

The data retention page is as shown below.

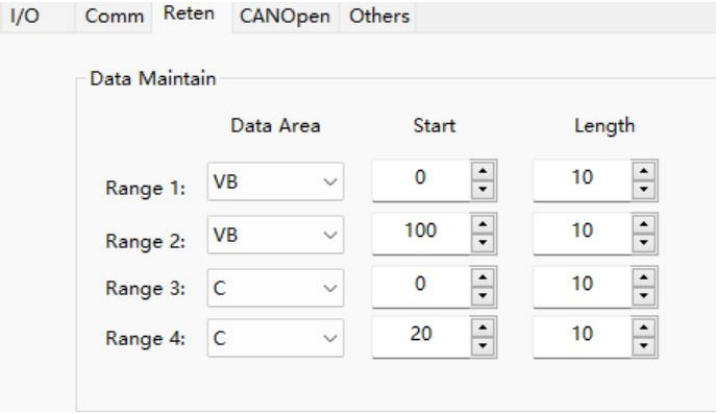


Figure 4-5 [Data retention] parameter configuration page

A total of 4 independent data holding areas can be configured.

• **Data area**: : Specify the memory area where the protected data area is located. There are two options: V area and C area.

For the counter (area C), only its current count value can be maintained.。。

• **initial address**: Specify the start byte address or counter number of this holding area.

• **Length**: Specifies the length of the holding area, in bytes or the number of counters。

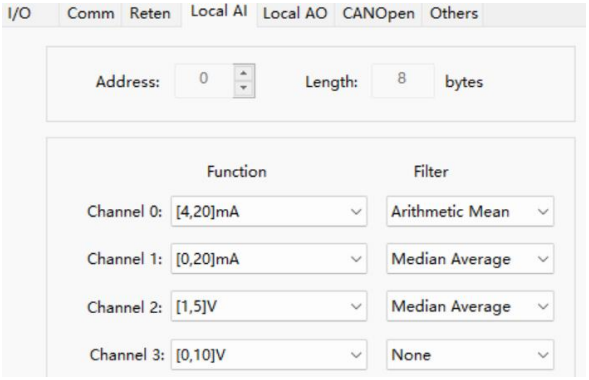
As shown in Figure 4-5, the data in Area 1 (VB0–VB9), Area 2 (VB100–VB109), Area 3 (C0–C9), and Area 4 (C20–C29) will be retained when the CPU is powered down.

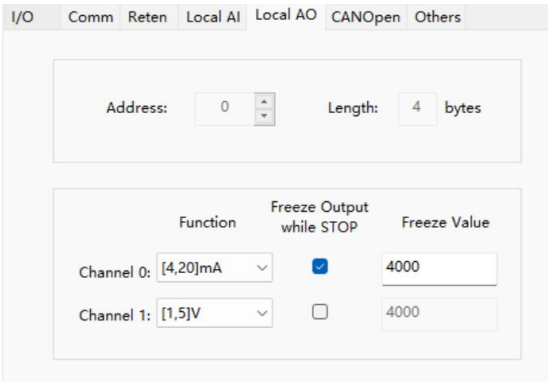
④ [Ontology AI/AO]page

Some models of CPU modules integrate AI channels, and the image area addresses of each channel are fixed as AIW0, AIW2, AIW4, etc., and so on. The sampling conversion speed of each channel is about 30 times/second.

In addition, some CPU bodies also integrate AO channels, and the image area addresses of each channel are fixed as AQW0, AQW2, etc., and so on. The conversion speed of each channel is about 30 times/second.

These integrated AI and AO points need to be selected in the [PLC Hardware Configuration] hardware configuration to select parameters such as the signal form and filtering method of each channel. The configuration interface is as follows:





For the parameter description of AI, please refer to 4.3.4.4 Parameter Configuration of AI Module.

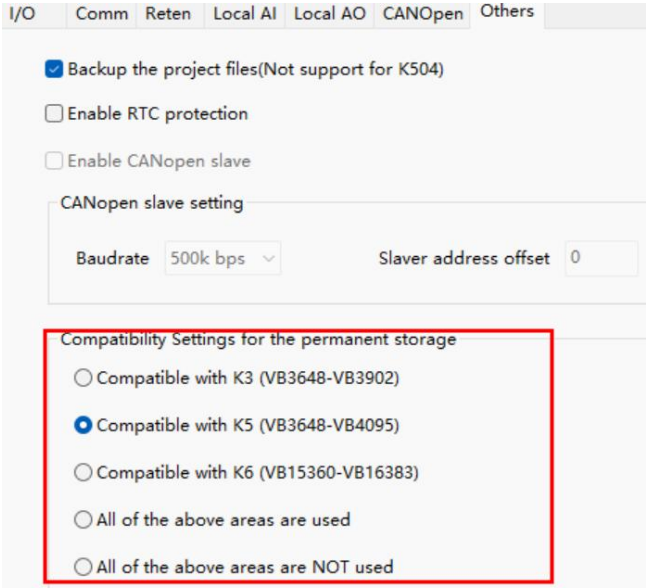
For the parameter description of AO, please refer to 4.3.4.5 Parameter Configuration of AO Module.

⑤ [CANOpen Master] page (some economical CPUs do not have this function)

For the CANOpen master function, see the introduction in chapter 10.5.4 CANOpen master function.

⑥ [**Other]** Page

1. Because the KPLC is constantly being upgraded, the data backup length of each CPU is not the same. In order to facilitate program migration and compatible with the previous K3, K5 series, etc., it is necessary to set the corresponding permanent storage area on the [Other] page of the CPU module in [PLC Hardware Configuration], as shown in the figure below.



- 【Persistent storage settings compatible with K3】-all supported

Selecting this item means that VB3648-3092 will be used as a data backup area, and the data in this area will be automatically written into permanent storage.

- 【Persistent storage settings compatible with K5】-All supported except K3

If this option is selected, it means that VB3648-4095 will be used as a data backup area, and the data in this area will be automatically written into permanent storage.

- [Persistent storage settings compatible with K6] - only supported by K6, KS101M, K209M

If this item is selected, it means that VB15360-16383 will be used as a data backup area, and the data in this area will be automatically written into permanent storage.

- [All the above areas are automatically and permanently stored] - only supported by K6

If this option is selected, it means that VB3648-4095 and VB15360-16383 will take effect at the same time as the data backup area, and the data in this area will be automatically written into the permanent storage.

- [All the above areas are not permanently stored] - only supported by K6

If this option is selected, it means that there is no permanent storage area, and all data in the V area will not be written into the permanent storage.

2. Back up the entire user project to the permanent storage of the PLC.

This option will be selected by default for new projects created by users. This function will download the complete user project (including global variable definitions, various comments, etc.) to the program memory for saving. This function will occupy part of the program memory space. If the user's program is so large that it exceeds the size of the program memory, the PLC will automatically ignore this option.

4.3.4.2 Parameter configuration of DI module

The parameter configuration of the DI module is very simple, as shown in the following figure.

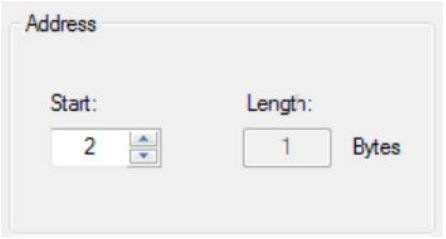


Figure 4-6 DI Module Parameter Configuration

- Address

• **Initial address** Specifies the starting byte address of the address space occupied by this module in the I area. The addresses of all channels of this module are determined by this "initial address".。

• **Length**: The length of the address space occupied by this module. This is a fixed value and depends on the number of DI channels on the module.

As shown in Figure 4-6, the module has 8 DI channels, the module address is %IB2, and its channel address is %I2.0–%I2.7.

4.3.4.3 Parameter configuration of DO module

Figure 4-7 Parameter configuration of DO module

- Address

• **Initial address**: Specifies the starting byte address of the address space occupied by this module in the Q area.

• **Length**: The length of the address space occupied by this module. This is a fixed value and depends on the number of DO channels on the module.

As shown in Figure 4-7, the module has 16 DO channels, the starting address of the module is %QB2, and the addresses of its channels are %Q2.0–%Q3.7.

- Output hold

Set the output state of each DO point of this module when the CPU is in STOP state.

If the check box corresponding to a DO point is selected, when the CPU is in STOP, the output of this point is 1; if the check box corresponding to a DO point is not selected, when the CPU is in STOP, the output of this point is 0 . The default state of the output point when the CPU is in STOP is that the output is 0.

This function is very meaningful for the safety interlock required after CPU shutdown.

4.3.4.4 Parameter configuration of AI module

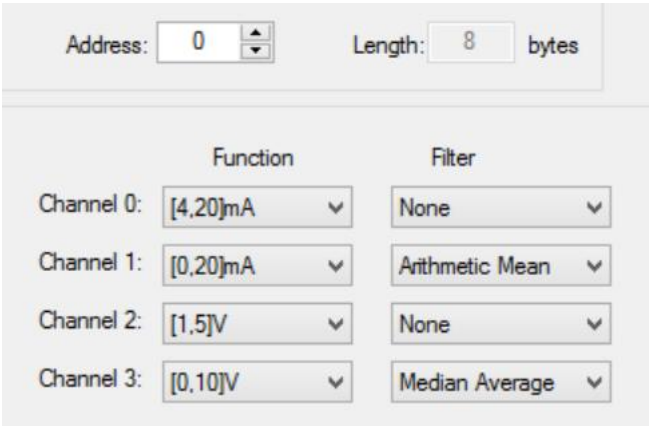


Figure 4-8 Parameter configuration of AI module

➢ Address

• **Initial address**: Specify the start byte address of the address space occupied by the module in the AI area (that is, the address of the first channel).

Each AI point occupies 2 bytes in the AI area. Therefore, the address must be an even number.

• **Length**: The length of the address space occupied by this module. This is a fixed value and depends on the number of AI channels on the module.

As shown in Figure 4-8, the starting address of the module is designated as %AIW0, and the module has 4 AI channels, so the addresses of its 4 channels are %AIW0, %AIW2, %AIW4, %AIW6 in sequence.

➢ Channel settings

- **Signal form:** select the input signal form for each channel, such as 4-20mA, 1-5V, etc. The sampled value will be automatically converted linearly in the CPU. For the data conversion format, please refer to the relevant content of "Measurement Range and Measured Value Representation Format" in the hardware manual.
- **Filter mode:** select software filter for each channel.

Using a filter for rapidly changing analog signals can make the measured value more stable.

Note: If the system needs to respond quickly to an AI signal, then the software filter at this point should not be enabled.

The input sampling of the software filter adopts the sliding method. The software filter has the following options:

- ✓ *None* --- Do not enable software filter。
- ✓ *Arithmetic Average* --- Take the arithmetic mean of a certain number of signal samples
- ✓ *Median Average* --- After removing the maximum and minimum values from a certain number of signal sample values, the remaining values are averaged.

4.3.4.5 Parameter configuration of AO module

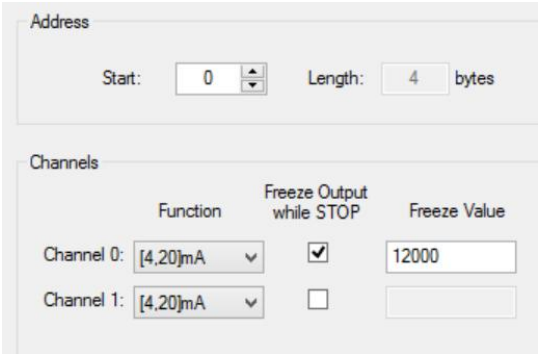


Figure 4-9 Parameter configuration of AO module

➢ Address

• **Initial address**: Specify the starting byte address of the address space occupied by the module in the AQ area (that is, the address of the first channel).

Each AO point occupies 2 bytes in the AQ area. Therefore, the address must be an even number.

• **Length**: The length of the address space occupied by this module. This is a fixed value and depends on the number of AO channels on the module.

As shown in Figure 4-9, the starting address of the module is specified as %AQW0, the module has 2 AO

channels, then the addresses of these 2 channels are %AQW0, %AQW2.

➢ Channel settings

- **Signal form:** select the output signal form for each channel, such as 4-20mA, 1-5V, etc. For the description of the output value representation format of various signals, please refer to the relevant content of "Output Range and Output Value Representation Format" in the hardware manual.
- **Stop hold:** specify whether the point will hold the original output when the CPU is in STOP state.
  If the check box corresponding to the point is selected, the point will be held in shutdown mode.
  If the check box corresponding to the point is not selected, the point does not use the shutdown hold mode.
- **Stop output value:** If this point is set to stop hold mode, set the output value of this point at stop here. The linearly converted value should be entered here instead of the actual signal value. For example, if the user chooses the signal form to be [1,5]V, and wants to output 1V when it is shut down, it should input 1000 here.

4.4 Initialize data table

In the initialization data table, the user can specify the initial value for the BYTE, WORD, DWORD, INT, DINT, REAL type data in the V area. Before entering the main loop when the CPU is powered on, the initialization data table is processed once, and the user-specified initial value is assigned to the corresponding address. The initialization data table style is as follows:

Initial Data

	Address	Value	Value	Value	Value
1	%VB0	B#1	B#11	B#11	
2	%VW12	2	22	222	
3	%VD122	DI#3	DI#33	DI#3333333	
4	%VR322	4.4	4.44	4.444	

Figure 4-10 Initialize data table

Note: The memory areas kept by the "Initialize Data Table", "Data Hold" and "Data Backup"

functions should avoid overlapping. Because these data are restored before entering the main loop after power-on, the sequence is: restore the memory data defined in "Data Retention", assign initial values to the memory area defined in "Initialization Data Table", and restore the user to use the instruction to permanently save The data.

4.4.1 How to enter the initialization data table?

There are two ways to enter the "Initialize Data Table" window:

• Double-click the [Initialize Data Table] node under the [Program] group in the project manager;

• Right-click on the [Initialize Data Table] node under the [Program] group in the project manager, and then execute the pop-up [Open...] menu instruction.

4.4.2 Enter data in table cells

When the table cell gets the focus, it will automatically enter the editing state, or click a table cell with the mouse to enter the editing state. If a table cell loses focus, its entered data will be confirmed.

Use the up, down, left, and right arrow keys to change the table cell that has focus.

Use PageUp and PageDown keys to turn pages.

If there is an error in the entered data, it will automatically turn red to indicate. Bad data is ignored when saving.

4.4.3 Define initialization data

There are 5 columns in the initialization data table: 1 [Start Address] column and 4 [Initial Value] columns.

Users can define initialization data as follows:

① Enter a direct address in the [Start Address] column;

② Enter one or more numerical values in the [Initial Value] column. If you enter multiple values, KincoBuilder implicitly assigns them to multiple consecutive direct address variables of the same type

starting from the starting address.

As shown in Figure 4-10,the meaning of the first line is to assign the initial values of B#1 and B#2 to %VB0 and %VB1 respectively; the meaning of the second line is to assign the initial values of 2, 3 and 4 to %VW10, %VW12 and %VW14 respectively; the meaning of the third line is to assign initial values DI#100, DI#200, DI#2000, and DW#2456 to %VD100, %VD104, %VD108, and %VD112, respectively.

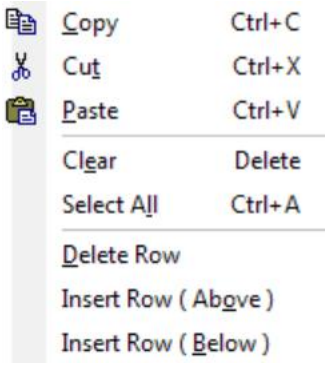
4.4.4 Edit initialization data table

- Sort

Click the column header of the [Start Address] column to sort the table in ascending or descending order of the letters in the starting address.

- Right-click menu

Right-click any cell in the table, and the following menu will pop up:



- **Delete current row:** delete the row where the focus is.
- **Insert a row (top):** Insert a new blank row at the position above the row where the focus is located.
- **Insert a row (bottom):** Insert a new blank row at the position of the row below the row where the focus is.

In addition, please note when using the paste instruction: pasting is not allowed between different types of tables, such as global variable tables and initialization data tables; pasting is not allowed between different columns.

4.5 global variable table

In the global variable table, users can easily complete the definition of global variables. In IEC61131-3, the keyword of global variables is VAR_GLOBAL. Global variables can be read and written in all POUs. The global variable table window is divided into two pages: [Global Variables] and [Function Block Instances].

- [global variable] page

Used to define global variables, that is, symbolic variables that directly access PLC memory addresses.

The global variable can be used in the program instead of the corresponding PLC direct memory address, so that the user program can be guaranteed to have good readability. Each memory address is only allowed to assign one symbolic variable name, and similarly, each symbolic variable name is only allowed to have one corresponding memory address.

For the naming rules of symbol variables, please refer to 3.3.1 Definition of Identifiers.

See 3.5 Variables for more information on global variables.

In this book, the "global variable table" usually refers to this page. The page style is shown below.

VAR_GLOBAL

	Symbol	Address	Data Type	Comment
1	MI_Guhang1	%M0.2	BOOL	1#pump failure
2	MI_Jianxiu1	%M0.5	BOOL	1#Pump Overhaul
3	MI_Yingji1	%M1.0	BOOL	1#Fire Emergency
4	MQ_JY1	%M2.0	BOOL	1#Pumps running at reduced pressure
5	MQ_QY1	%M2.4	BOOL	1#Pump full pressure operation
6	T_JQ	%VW4	INT	Delay after pressure drop
7	T_Beng	%VW6	INT	Start-up delay between pumps
8				

Global Variable | FB Instance

Figure 4-11 Global variable table page

➢ 【Function Block Instance】page

As mentioned in 3.6.6 Naming and Using FB Instances, in order to facilitate the use of users, the definition of function block instances is automatically completed by KincoBuilder, so in the [Function Block Instance] page, all table cells are not editable ,and its information is only for the user's reference. The page is as shown below

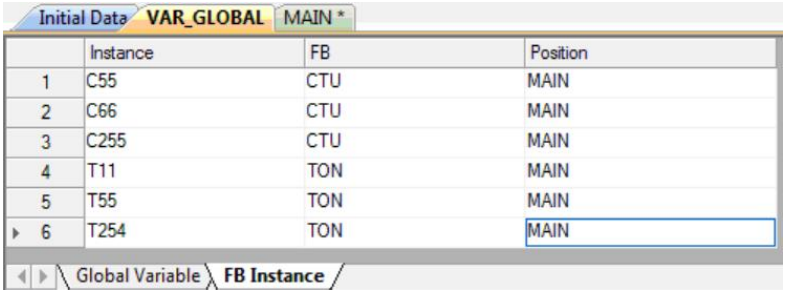


Figure 4-12 [Function Block Example] page

4.5.1 How to enter the global variable table?

There are three ways to enter the global variable table window:

• Double-click the [Global Variable Table] node under the [Configuration] - [Resources] group in the project manager;

• In the project manager, click the right mouse button on the [Global Variable Table] node under the [Configuration]-[Resources] group, and then execute the pop-up [Open…] menu instruction.

• Execute the menu instruction 【Project】→【Global Variable Table】.

4.5.2 declare global variable

There are 4 columns in the global variable table: [Global Variable Name], [Address], [Data Type] and [Comment] columns.

Users can declare a global variable as follows:

① Enter the global variable table window and switch to the [Global Variables] page;

② Enter a variable name in the [Global Variable Name] column and confirm;

③ Enter a direct memory address in the [Address] column and confirm;

④ Select a data type for the variable in the [Data Type] column and confirm;

⑤ (Optional) Enter a comment for this global variable in the [Comment] column.

The meaning of the first line in Figure 4-11 is: define a symbolic name "MI_Guzhang1" for the address "%M0.2", and its data type is "BOOL". In the program, "%M0.2" is equivalent to "MI_Guzhang1".

- **Edit global variable table**

Editing method: (eg: M0.2 is the 1# pump fault signal)

1、Open the global variable table and edit the variable name: MI_Guzhang1

VAR_GLOBAL

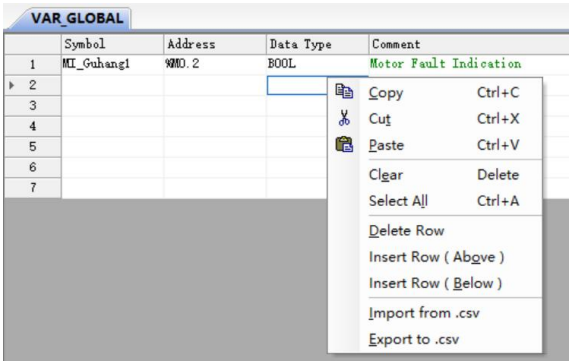
	Symbol	Address	Data Type	Comment
1	MI_Guhang1	%M0.2	BOOL	Motor Fault Indication
2				
3				
4				

2. The address is input M0.2, the data type is automatically recognized as BOOL

3. Edit the comment to make the meaning of this register clearer. Place the cursor in the corresponding input field for editing.

- **Modify the global variable name**

  - Menu bar [Project]-[Display global variable name], first switch the variable name in the program to the register state
  - Close all program windows, and then open the menu bar [Project]-[Show global variable name] after modification



The global variable table can be exported to and imported from a CSV file. Right-click to select to operate.

4.6 cross-reference table

The cross-reference table is used to analyze all address variables used in the current project and list the detailed information. Using the cross-reference table is convenient for users to perform statistics and analysis on the current project.

The information in the cross-reference table is generated after the first compilation and is automatically refreshed during subsequent compilations.

The cross-reference table is shown in the following figure:

Figure 4-13 Cross-reference table

• [Address]: Displays all memory addresses used in the current project.

• [Global variable name]: Display the global variable name corresponding to the [address] of this line.

• [POU]: Display the name of the POU where the [Address] of this line is located.

• [Location]: Indicate the specific location of this bank's [Address] in [POU].

If the POU is written in IL, the line number is displayed; if it is written in LD, the network number is displayed.

• [Read/Write]: Indicates that this line [address] has been read or written at the [location].

As shown in Figure 4-13, the first line in the table means: %I0.2 is used once in the second line of the Demo program of this project, and the read operation is performed on %I0.2 here.

4.6.1 How to enter the cross-reference table?

There are three ways to enter the cross-reference table window:

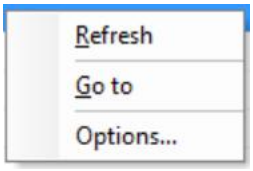
• Execute the menu instruction 【Project】→【Cross-reference Table】;

• Mouse-click the icon on the toolbar.

4.6.2 Working with cross-reference tables

➢ Right-click menu

Click the right mouse button on any row of the table, and the following right-click menu will pop up:



- **Refresh**: refresh the display cross-reference data.
- **Variable positioning**: open the [POU] of this line and locate the specific [location] where the [address] is located.
- **Filter**: The [Cross Reference Options] page in the [Software Setting] dialog box will pop up, and the user can choose to display only the variables of a certain memory area used in the project and with the same data type.

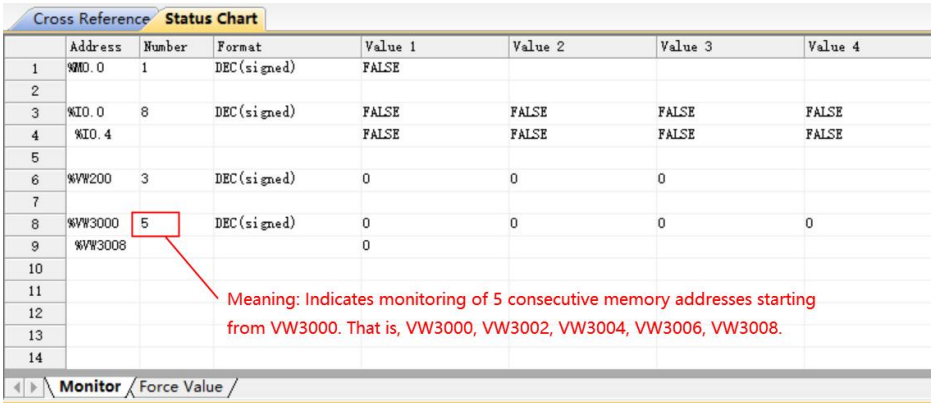
4.7 Variable state table

The user can use the variable status table to monitor and force the memory variables in the PLC online. In the variable status table, two pages of memory monitoring table and variable status table are provided.

- Memory monitoring table

Arbitrary memory address for monitoring PLC.

The memory monitoring table style is as follows:



The content of the table is shown below:

• [Memory Address]: Enter the starting address of the memory area to be monitored.。

• [Monitoring Length]: Enter the number of data to be monitored starting from "memory address", the maximum allowed number is 150.

Each row in the table displays at most 3 data. If the input length exceeds 3, the software will automatically display it in separate rows.。

• [Display format]: Select the display format of the memory value, including decimal and hexadecimal.

• [Memory Value]: Displays the monitored memory value. Each row displays up to 3 columns, where the first column is the value of the starting memory.

If the bit memory is monitored, only TRUE and FALSE are displayed, and the "display format" is invalid.

- Variable state table

The variable state table is used to monitor and force any address variable used in the current project online.

The style of the variable state table is shown in the following figure.

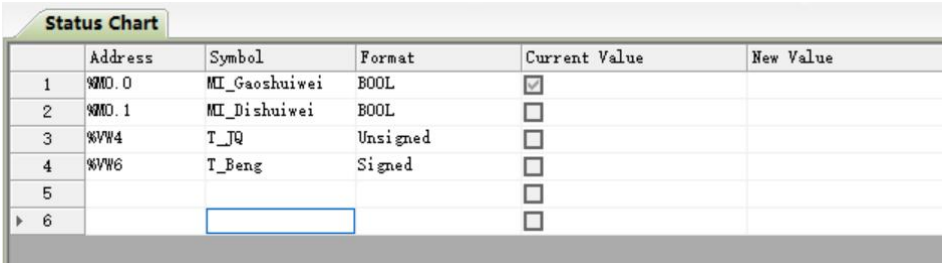


Figure 4-14 Variable state table

The table is divided into the following five columns:

• [Address]: Enter the direct address to be monitored and forced.

• [Global variable name] : Display the global variable name corresponding to the [address] of this line.

• [Display Format]: Select the data display format of the current value and forced value.

The optional formats are BOOL, REAL, signed, unsigned, binary, hexadecimal, etc.

• [Current Value]: The monitored value of the [Address] of the current line will be displayed during online monitoring.

If the check box is checked, it means that the address of this row is being enforced.

• [Forced Value]: You can enter the value to be forced to the [Address] of this line during online monitoring.

Note: In order to improve efficiency, only the variables used in the current project are allowed to be monitored and forced in the variable state table. If the user enters an unused variable, KincoBuilder will not

prompt an error, but neither the current value nor the forced value will work.

4.7.1 How to enter the variable state table?

There are three ways to enter the variable status table window:

• Double-click the [Variable Status Table] node in the project manager;

• Right-click on the [Variable Status Table] node in the project manager, and then execute the pop-up [Open…] menu instruction;

• Execute the menu instruction 【Debug】→【Variable Status Table】.

4.7.2 Monitor the value of a variable

The user can use the variable status table to monitor the value of the variables used in the user program as follows:

① Enter the direct memory address to be monitored in the [Address] column;

② Use one of the following methods to enter the online monitoring state:

- Execute the menu instruction【Debug】->【Online Monitor】;
- Click the icon on the toolbar;
- Use shortcut key F6

③ The user can change the [display format] of the monitoring value at any time.

4.7.3 About the forcing function

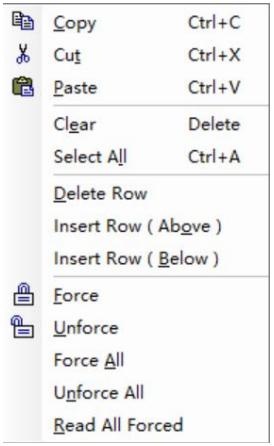
The user can use the force function to forcefully modify the variable values in the I, Q, M, V, AI, and AQ areas of the PLC. The variables in the I, Q, M and V areas can be forced by bit, byte, word or double word. Variables in AI and AQ areas can be forced by word. When the CPU is cold-started (power cycled), the forced state of all variables is canceled.

Kinco-K series PLC allows forcing up to 32 variables at the same time. Immediate directives do not allow coercion.

At a certain moment in the scan cycle, the value of a variable may be as follows: external input signal (I, AI) or user program execution result (Q, AQ, M, V), user-specified mandatory value. Therefore, the following principles of variable values are specified:

- For the variables in the M and V areas, the forced value and the program execution result are in the same priority: if the user performs forcing, the forced value will take effect at the beginning of each scan cycle, and then the program execution result will take effect.
- For the variables in the I and AI areas, the forced value takes precedence over the external signal input value. If forced by the user, the forced value will take effect during the scanning process.
- For the variables in the Q and AQ areas, the program execution result takes precedence during the scan, but the forced value takes precedence in the output task at the end of the scan cycle.

4.7.4 right-click menu



- **Force:** Write the [forced value] to the direct memory [address] in the PLC.
- **Cancel Force:** Cancel the forced state of PLC Direct [Address] in the clicked row.
- **All Force:** Write all the forced values in the [Forced Value] column into the corresponding PLC direct memory in the [Address] column.
- **Cancel all force:** Cancel the forced state for all PLC direct memory in the [Address] column.

- **Read all force:** read all forced addresses and their forced values in the connected CPU and display them in the variable status table.

4.7.5 Force, cancel force

In the variable status table, the user can perform the following steps to force or unforce a memory variable:

① Enter the direct memory address to be forced in the [Address] column.

② (Optional) Select the [Display Format] of the numerical value. Of course, the display format can also be modified at any time.

③ Enter the desired forcing value in the [Forced Value] column. Users can directly input decimal integers, and KincoBuilder will automatically adjust the format of the value according to [Display Format].

④ Force it using one of the following methods:

- Click the right mouse button in this row, and execute the [Force] instruction in the pop-up menu;
- Click anywhere in the line with the mouse, and then click the icon on the toolbar;
- Click anywhere in the line with the mouse, and then execute the menu instruction 【Debug】→【Force】.
- Use one of the following methods to cancel the forced status for PLC direct [address] in this row::

⑤ Click the right mouse button on this row, and execute the instruction 【Cancel Force】in the pop-up menu;

- Click anywhere in the line with the mouse, and then click the icon on the toolbar
- Click anywhere in the line with the mouse, and then execute the menu instruction 【Debug】→【Cancel Enforcement】.

The editing method of the variable state table is the same as that of the initialization data table, please refer to the relevant content in the initialization data table.

Chapter 5 Write User Program with KincoBuilder

This chapter describes the functions and usage of the LD and IL editors in KincoBuilder in detail, and also describes the relevant grammars and regulations of the LD and IL languages in the IEC61131-3 standard. By reading this chapter, users can easily use KincoBuilder to write applications that comply with IEC standards.

For the writing of PLC application program, 3 kinds of textual languages and 3 kinds of graphical languages are stipulated in IEC61131-3. Textual languages include: Instruction List (IL), Structured Text (ST), Sequential Function Chart (SFC, textual version); Graphical languages include: Ladder Diagram (LD), Function Block Diagram (FBD), Sequential Function Chart (SFC, graphical version).

KincoBuilder currently supports IL language and LD language programming. In a project, IL and LD languages can be mixed, but only one language is allowed in a POU. The user can execute the menu instruction 【Project】→【IL language (instruction list)】or 【Project】→【LD language (ladder diagram)】to switch the language of the current program at any time. Note: Any LD program can be converted into IL program, but only IL programs written according to certain rules can be converted into LD programs.

5.1 IL programming

5.1.1 IL's background

IL language is a low-level language, very similar to assembly language. It is a standard language formed on the basis of borrowing and absorbing the instruction list languages of famous PLC manufacturers in the world.

IL is close to machine code, so programs written in IL are more efficient and compact. IL is very suitable for experienced programmers, and sometimes using IL language can solve problems that are difficult to solve in graphical languages such as LD.。

5.1.2 IL syntax

5.1.2.1 IL Statement format

IL programs are line-oriented, and each line statement can only have one instruction or one label. Blank lines are allowed in IL programs.

The basic format of a statement in an IL program is as follows:

```
Label:
Operator/Function/Function Block Operand (Table) (* Note *)
```

- **Label (optional)**

  The purpose of using labels is to achieve program jumps. Labels are named in the same format as variables.

- **Operator/function/function block**

  PLC instruction

- **Operands number (table)**

Please refer to the detailed description of each operator, function and function block in the next chapter, which also includes the description of the corresponding operand.

There is at least one space between operator and operand.

- **Comments (optional)**

  Comments are formatted the same in all languages, delimited by (* *).

  Only one comment is allowed per line. Comments can also take a line by themselves.

  Comments are not allowed to be nested, nested comments will be considered an error by the compiler.

  An example of an IL statement is as follows::

```
(* NETWORK 0 *)
Run:          (* Marker for use when jumping *)
LD    %I1.0
TP    T2, 168   (* If I1.0 is true, start timer T2, which is declared to be of type TP *)
```

5.1.2.2 about CR

A general-purpose accumulator called "Current Result (CR)" is provided in IL, and the current execution result of the user program is stored in CR. The CR is flushed after each line of statements in the user program is executed. Depending on the subsequent statement, the CR value may be used as the execution condition of the next statement or as one of the operands of the next statement.

Different operators have different effects on the CR value after execution. The following table preliminary groups the operators in KincoBuilder according to their different effects on the CR value. For a more detailed description, please refer to the introduction of each instruction in the next chapter.

operator	Group	Effect on CR value
LD, LDN	C	CR value is re-established
Logic instructions, comparison instructions, etc.	P	The CR value is updated as the result of the operation
ST, R, S, JMP, JMPCN, JMPC etc.	U	CR value remains unchanged

Table 5-1 The effect of each operator on CR after execution

Note: The impact of various operators on CR is not well defined in IEC61131-3, so these definitions may be different in different programming systems.

5.1.2.3 Network

There is also the concept of network in the IL program. Taking the network as the basic paragraph, the code part of a POU is composed of several networks.

A typical network consists of two parts, the network label and the sentences in the network. In KincoBuilder, the format in the network is as follows:

- There can be only one statement label in a network. E.g:

```
(* NETWORK 0 *)
MRun:   (* can have only one label *)
```

- There can be only program statements in a network.

In 5.1.2.2 About CR, we divided all instructions into three groups: "C", "P", "U".

A program in the network must start with an instruction in the "C" group and end with an instruction in the "P", "U" group.

An example is as follows:

```
(* NETWORK 0 *)
LD  %M3.5  (* Start with LD instruction *)
ST  %Q2.3  (* end with an allowed instruction *)
```

- Statement labels and program statements may exist in a network.

A program in the network must start with a statement label or an instruction in the "C" group, and end with an instruction in the "P", "U" group.

An example is as follows:

```
(* NETWORK 0 *)
MRun:    (* start with statement label *)
LD %M3.5
ST %Q2.3  (* end with an allowed instruction *)
```

5.1.3 IL editor in KincoBuilder

When using IL language to create a new program, it will enter the IL editor; if a program written in IL is opened, it will also enter the IL editor. The appearance of the IL editor is as follows.

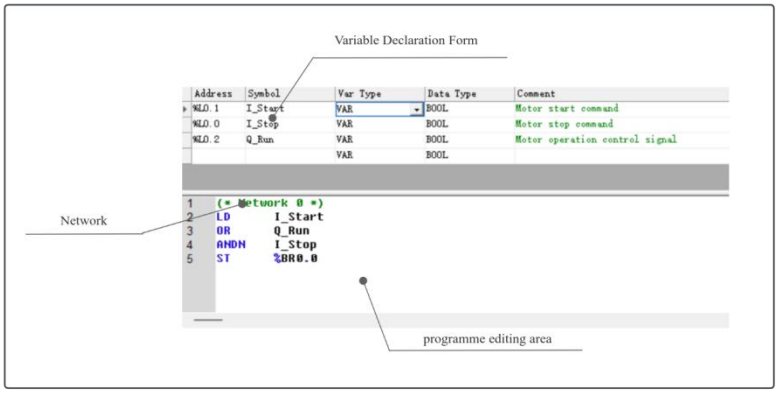


Figure 5-1 IL Editor

A POU consists of two parts: parameter, variable declaration part; code part. Therefore, the editor is correspondingly divided into two areas:

- Variable declaration form: used to declare the input/input parameters and local variables of the POU. Support right-click menu.
- Program editing area: This area is similar to a text editor, where users can directly edit their own applications.

5.1.3.1 IL Program editor

➢ Add a network

A new network can be added using any of the following methods:

- Use the Ctrl+Q shortcut.
- Click the right mouse button in the program editing area, and execute the pop-up menu instruction [Add New Network].

➢ Enter IL statement

The program editing area is similar to a text editor, where users can directly input statements and edit

their own applications. The usual keyboard operations are supported here, such as Delete, Backspace, using the up, down, left and right arrow keys to move the cursor, etc.

The IL editor can automatically format the statement entered by the user, and display the keywords in the statement in blue and the comments in green.

When editing a statement, if the cursor switches to another line, the IL editor will check the syntax of the line just left, and if there is a syntax error, a red question mark (?) will be displayed at the beginning of the line. Only the correct statement is checked for formatted display.

- Select/Delete/Copy/Cut/Paste

All editing instructions in the [Edit] menu can be used in the IL editor, including select all, copy, cut, paste, etc. In addition, clicking the right mouse button in the program editing area will pop up the right-click menu, and the user can also use the right-click menu instruction to edit.

Executing the [Select All] instruction will select all the text content in the editing area. Use the mouse to drag in the editing area, or use the up, down, left, and right arrow keys to move the cursor while pressing the SHIFT key, the text content will be selected. The selected area will use black as the background color, and the text will be highlighted.

Use the Delete key or execute the [Delete] instruction to delete the selected content.

Use the shortcut key Ctrl+C or execute the [Copy] instruction to copy the selected content to the Windows clipboard. The copied content can be pasted in any text editor, such as Windows Notepad, Word, etc.

The cut operation is equivalent to copying and deleting the selected content. Use the shortcut key Ctrl+X or execute the [Cut] instruction to cut the selected content to the Windows clipboard.

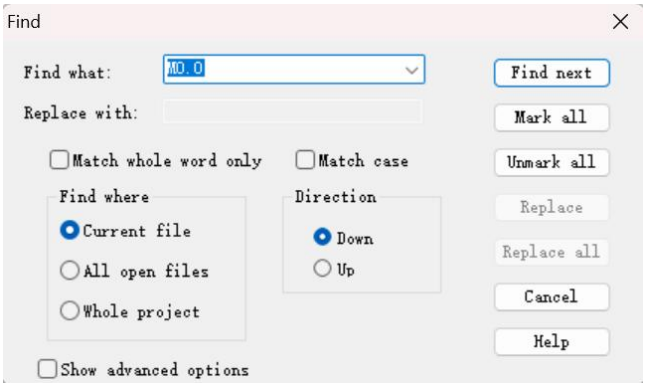
After copying or cutting, position the cursor to the position you want to paste, and then use the shortcut key Ctrl+V or execute the [Paste] instruction to paste the contents of the clipboard to the position of the cursor.

- Find/Replace

The IL editor provides standard find and replace instructions.

- Find

Use the shortcut key Ctrl+F or execute the [Edit] → [Find...] menu instruction, the following search window will pop up:

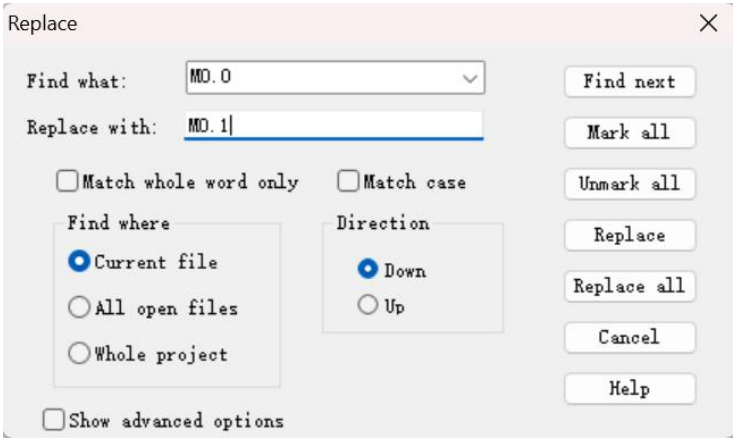


Enter the string you want to find in the [Find what] input box, and then click the [Find Next] button to start the search, and the found content will be selected and highlighted.

The other options all adopt the meanings in the Windows standard search window, and will not be repeated here.

- Replace

Use the shortcut key Ctrl+R or execute the [Edit] → [Replace...] menu instruction, the following replacement window will pop up:



Enter the string to be searched in the [Find Content] input box, enter the string to be replaced in the [Replace with] input box, and then click the [Replace] button, the IL editor will automatically find the next match to the search content. string and replace that string. Click the [Replace All] button, the IL editor will automatically find all the strings in the current program that match the search content and replace them with the specified strings.

The other options all adopt the meanings in the Windows standard search window, and will not be repeated here.

5.1.3.2 IL program example

```
(* NETWORK 0 *)
LDN    %M0.0
TON    T0, 1000      (*Dependent on the output of T1 to start T0, timed at 1000 x 1ms  *)
ST     %M0.1
(* NETWORK 1 *)
LD     %M0.1
TON    T1, 1000      (* Dependent on the output of T0 to start T1, timed at 1000 x 1ms *)
ST     %M0.0
(* NETWORK 2 *)
```

```
LD    %M0.1
ST    %Q0.0        (*Output square wave at Q0.0 *)
```

5.1.4 Convert IL program to LD program

Execute the menu instruction 【Project】→【LD Language (Ladder Diagram)】to switch the language of the current POU to LD.

Not all IL programs can be converted to LD format. To convert IL programs to LD programs, the following conditions must be met:

①The IL program itself does not have any errors;

② Each network in the IL program must strictly comply with the following rules, including:

- If the network contains statement labels, only one statement label is allowed, and there cannot be any other instructions.

- If the network does not contain statement labels, the following conditions must be met:
  - The network must start with "C" group instruction (LD class instruction);
  - The "C" group instruction (LD class instruction) as the start line of the network can only appear once in a network;
  - The program in the network must end with "P", "U" group instructions.。

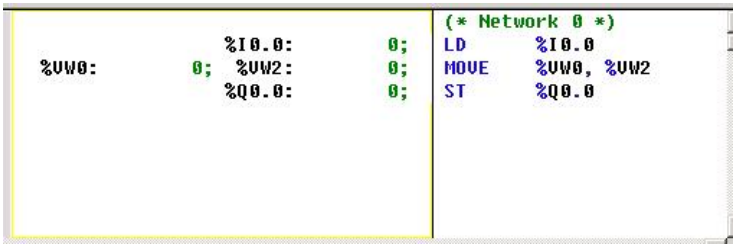
5.1.5 Debug and monitor programs

5.1.5.1 Online monitoring of IL programs

If the current window is the IL editor, you can use any of the following methods to enter the online monitoring state.

- Execute the menu instruction 【Debug】→【Online Monitor】.
- Click the icon on the toolbar.
- Use the F6 shortcut key.。

In the online monitoring state, the original program editing area will be divided into two columns, the right side displays the program, the left side displays the corresponding variable value, and the middle is separated by a vertical line. Place the cursor on the vertical line, and after its shape changes to , you can drag the line to change the size of the left and right areas.



Note: Program editing is not allowed in online monitoring state.

5.1.5.2 Force a variable to be specified

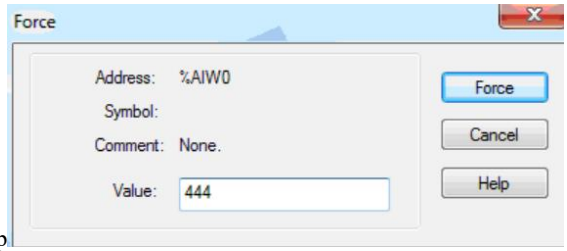
The forcing function of the KINCO-K series is described in detail in 4.7.3 About the forcing function.

When monitoring the IL program online, the user can directly force or cancel the force on the specified variable in the IL editor: right-click a variable in the program, and the following right-click menu will pop up

(Note: If the right-click is a non-switch variable, the [Force to TRUE] and [Force to FALSE] instructions in the menu will be invalid)

- Force is TRUE: Force the value of this variable (switch) to 1 (TRUE).

- Force is FALSE: Force the value of this variable (switch) to 0 (FALSE).
- Force...: After executing this instruction, the following dialog box will pop up



Enter the value to be forced (constant of the corresponding data type) in the [Force Value] input box, and then click the [Force] button.

For the representation format of constants, please refer to 3.4 Constants.

- **Unenforce:** Unenforce the variable.
- **Unenforce all:** Unenforce all variables currently in the connected CPU.

5.2 LD Programming

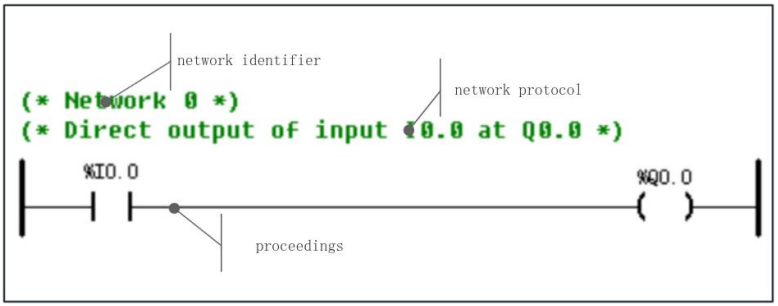
5.2.1 LD Background

LD (Ladder Diagram) language is one of the five programming languages specified in the IEC61131-3 standard, and is the most widely used graphical language in PLC programming.

LD language originates from the relay logic represented graphically in the field of mechatronics. The program written with it can correspond to the actual electrical operation schematic diagram, which is very intuitive and easy to learn and master. The strength of the LD language is the handling of Boolean logic.

5.2.2 Network

The concept of "Network" is also used in LD, with the network as the basic paragraph. A typical network consists of three parts: network labels, annotations, and programs in the network. As shown below。



The boundary of an LD network is the Power Rails located on the left and right sides. The state of the left power cord is always "1", and the state of the right power cord is undefined. We can describe an LD network as follows: electrical energy (energy flow) flows from the power line on the left along the connecting line through all the components on the line (including contacts, coils, functions, function blocks, etc.

Depending on their logic state, these elements either interrupt the flow of energy, or transfer power to subsequent elements and eventually to the power line on the right.

5.2.3 Standardized Graphics Objects

- Connect

Horizontal connection lines and vertical connection lines are used in LD, which correspond to series and parallel relationships, respectively. The value of the connection line mentioned in the table below or the value of one side of the connection line can also be understood as whether there is energy flow.

Graphic object	Name	Describe
	Horizontal connection	t can be thought of as an ideal wire. A horizontal connection transfers the value on the left side of any point on the line to the right side.
	vertical connection (with horizontal	A vertical connection first logically ORs the values of all the horizontal connections to its left, and then passes the result to all the horizontal connections to its right.

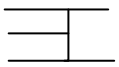
	connections)	

Table 5-2 Connections in LD

- Contacts

There are two types of contacts: normally open contacts and normally closed contacts.

Each contact must correspond to a valid boolean variable whose name is written above the graphic element.

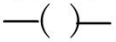
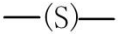
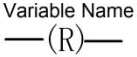
The value of this variable determines the state of the contact (closed or open), which in turn determines whether the line on which the contact is located is on or off.

Graphic object	Name	Describe
Variable Name	normally open contact	If the variable value is TRUE, the contact is closed, and the value of its left connection is passed to the right connection; otherwise, the contact is opened, and the value of its right connection is FALSE.
VariableName	normally open contact	If the value of the variable is TRUE, the contact is opened and the value of its right-hand connection is FALSE; otherwise, the contact is closed and the value of its left-hand connection is passed to the right-hand connection.

Table 5-3 Contacts in LD

- Coil

Coils are graphic elements used to assign values to Boolean variables.

graphic object	Name	Description
Variable Name 	coil	Pass the left concatenated value to the variable.
Variable Name	Negative coil	Invert the left concatenated value and pass it to a variable.
Variable Name 	set coil	If the value of the left connection is TRUE, the variable value is set to TRUE; If the value of the left join is FALSE, the variable value remains unchanged.
Variable Name 	reset coil	If the value of the left join is TRUE, the variable value is set to FALSE; If the value of the left join is FALSE, the variable value remains

		unchanged.

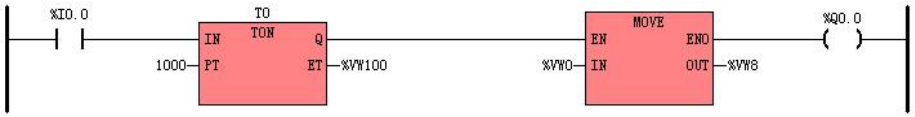
table 5-4 Coil in LD

- Calling functions and function blocks

LD supports calls to functions and function blocks. The called functions and function blocks are represented by rectangular boxes, their formal parameters are displayed in the rectangular box, and their actual parameters are displayed on the connecting lines outside the rectangular box. The parameters of the function or function block have at least one input parameter of type BOOL and one output parameter of type BOOL, which are used to connect it to the connection line.

A function must have a BOOL-type input named EN and a BOOL-type output named ENO that controls the execution of the function. If the value of EN is 1, the function is executed and ENO will also be set to 1; if the value of EN is 0, the function will not be executed and ENO will also be set to 0.

The figure below is an example of calling functions and function blocks.



5.2.4 LD editor in KincoBuilder

When using the LD language to create a new program, it will enter the LD editor; if a program written in LD is opened, it will also enter the LD editor. The appearance of the LD editor is as follows.

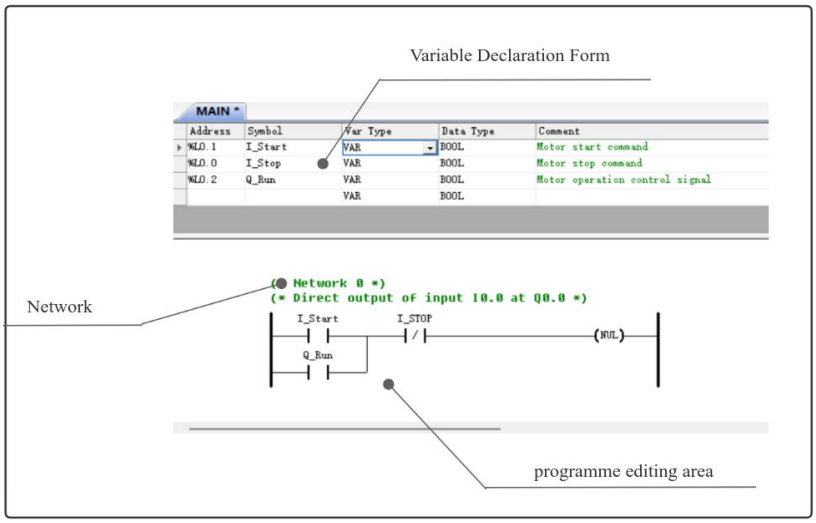


Figure 5-2 LD Editor

A POU consists of two parts: parameter, variable declaration part; code part. Therefore, the editor is correspondingly divided into two areas:

• Variable declaration form: used to declare the input/input parameters and local variables of the POU. Support right-click menu.

• Program editing area: This area is similar to a canvas, where users can input various LD graphic elements.

5.2.4.1 Limitations of the LD Program

The program editing area is divided into many cells, the user can click the mouse or use the arrow keys of the keyboard to select the cell, the selected cell will be surrounded by a dotted rectangle to indicate the current position. Various LD elements occupy one or more cells respectively according to their size. Due to the size limitation of the canvas, there are also the following limitations in each LD program:

- A maximum of 200 networks are allowed in one program (main program, subprogram or interrupt program).
- In a project, the maximum allowed number of subprograms and interrupt programs is 99 respectively,

and the total number of components used in all programs is also limited: KS101M, K209M, K6 allow a maximum of 8192 instructions, K2, K5, KS, MK , KW and other series allow a maximum of 4096 instructions.。

- Restriction on the number of components connected in series on each connecting line path: for coils and contacts, no more than 35 contacts plus 1 coil. If only the function/function is fast, it is recommended not to exceed 12 blocks plus 1 coil plus 1 contact.
- A parallel relationship is not allowed to have more than 16 branches.

5.2.4.2 LD program editor

➢ Mark components

Left-click on a component to select and mark the component. The marked component will be surrounded by a dotted rectangle, indicating that the component has the current focus. Alternatively, you can use the arrow keys on the keyboard to move the focus to change the marked component.

The marked component will be used as a reference, and users need to add new components according to this reference position.

In addition, users can perform various editing operations on the marked components, such as modifying properties, deleting, copying, cutting, and so on.

➢ Add a network

① Mark a component as a reference at the desired location.

If the mark is a net annotation, a new net will be added above the net where the marked component is located; if other components are marked, a new net will be added below the net where the marked component is located.

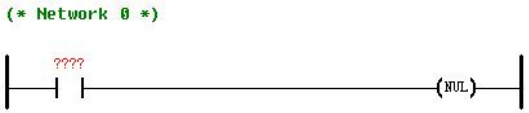
If the current program is an empty program without any components, then there is no need to mark the components, and a new network will be added at the top of the editing area.

② Use any of the following methods to add a new network to the program:

- Execute the menu instruction 【Ladder Diagram】→【New Network】.

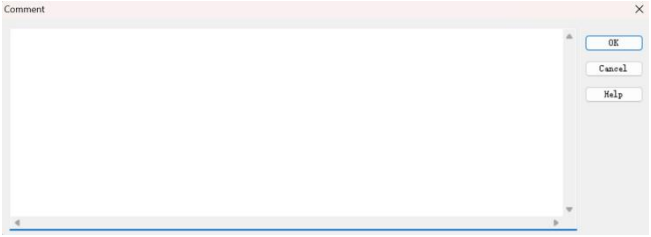
- Click the icon on the toolbar.
- Use the Ctrl+W shortcut.
- Right-click any component, and execute the [New Network] instruction in the right-click menu.

After the new network is added, it will look like the following figure:



➢ Enter comments

Double-click the network label part to pop up the "Comment" dialog box, as shown in the following figure.



The user can enter the comment of the network in the input box on the left, and then click the [Confirm] button.

➢ Add a series contact

① Mark a component as a reference at the desired location

② Use any of the following methods to add a series contact to the program:

- Execute the menu instruction 【Ladder Diagram】→【Left Contact】or 【Right Contact】.
- Click the icon (left touch) or the icon (right touch) on the toolbar.
- Use the shortcut Ctrl+L (left touch) or Ctrl+E (right touch).
- Right-click the reference component, and execute the [Left Contact] or [Right Contact] instruction in the context menu.

- Double-click the required contact instruction in "LD instruction set" on the right side of the window.。

➢ Add a parallel contact

① Mark a contact element as a reference at the desired location.

② Use any of the following methods to add a parallel contact to the program:

- Execute the menu instruction 【Ladder Diagram】→【Parallel Contact】.
- Click the icon on the toolbar.
- Use the Ctrl+D shortcut.
- Right-click the reference element and execute the instruction 【Parallel Contact】in the right-click menu.。

➢ Add a parallel coil

① Mark a coil element as a reference at the location where you want to add it.

② Use any of the following methods to add a parallel coil to the program:

- Execute the menu instruction 【Ladder Diagram】→【Parallel Coil】.
- Click the icon on the toolbar.
- Use the Ctrl+D shortcut.
- Right-click the reference element, and execute the instruction【Parallel Coil】in the right-click menu.
- Double-click the desired coil instruction in "LD instruction set" on the right side of the window.

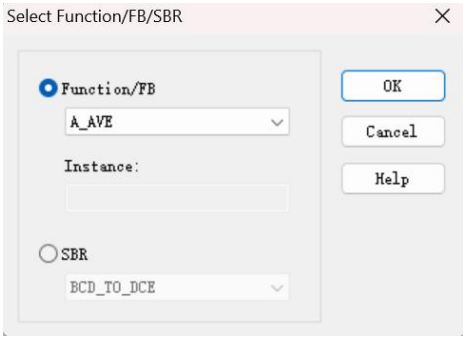
➢ Add a concatenated function/function block

① Mark a component as a reference where you want to add it.

② Add a chained function/function block to the program using any of the following methods:

- Execute the menu instruction [Ladder Diagram]→[Function/Function Block]; or click the icon on the toolbar; or use the Ctrl+B shortcut key; or right-click the reference component, and execute the

[Function/Function] in the right-click menu block] instruction. Then the following "Function/Function Block/Subroutine" window will pop up:



The user selects the desired [function/function block] or [subroutine], and then clicks the [OK] button.

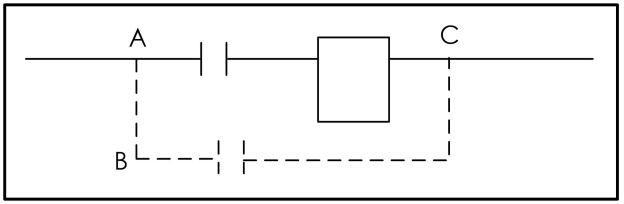
- Double-click the required function, function block instruction or subroutine in the "LD instruction set" on the right side of the window

➢ Create a parallel branch

The LD editor provides a parallel branch editing mode, allowing users to quickly create a complex parallel branch. After entering the parallel branch editing mode, the shape of the mouse pointer will change to . In addition, the user can use the ESC key to exit this mode at any time. The steps for the user to create a parallel branch are as follows:

① Execute the menu instruction [Ladder Diagram] → [Parallel Branch]; or click the icon on the toolbar; or use the Ctrl+H shortcut key; or right-click a component, and execute [Parallel Branch] in the right-click menu Order. Then it will enter the parallel branch editing mode.。

② Use the mouse to continue to follow the steps below to draw a parallel branch:



1) Click at the desired parallel start position (A in the above figure);

2) Move the mouse up or down to any position (such as B) and click;

3) Move the mouse to the desired parallel end position (such as C) and click to complete the establishment of a parallel branch.

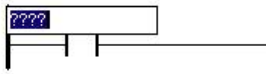
Note: The three positions mentioned above do not need to be too precise, the user can click at the "approximate" position

➢ Modify the parameters of components

After adding a component, its actual parameters are represented by red question marks, which must be modified to appropriate constants, direct addresses, global variables, or local variable names.

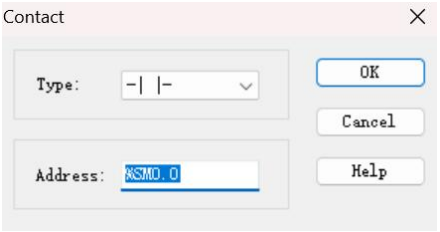
The user can use any of the following methods to modify the parameters of the component:

- Click the position where the parameter of the component to be modified is located, and an input box will appear in the parameter area, as shown in the figure below



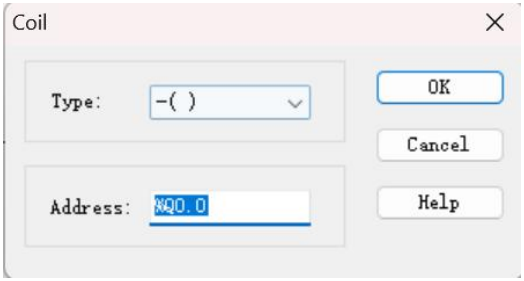
The user can directly input the desired constant, direct address, global variable or local variable name in this input box, and then press Enter to confirm. The LD editor will automatically format the direct address entered by the user, so the user can enter without the percent sign (%). During input, you can also press ESC to cancel the current modification.

- Mark the component to be modified, and then press the Enter key, an input box will appear in the area of the first parameter of the marked component, and then input according to the above method.。
- Double-click the component to be modified, and the properties dialog box of the component will pop up
  - ✓ Properties dialog of the contact



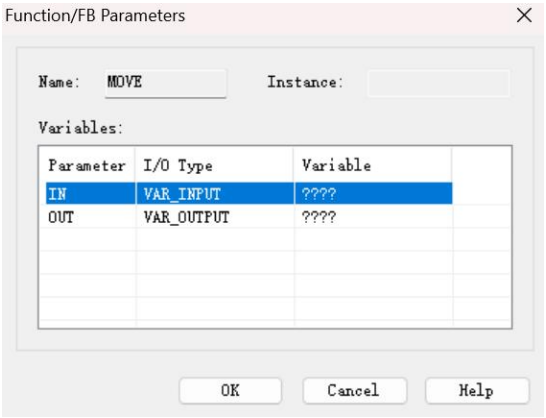
The user can modify the [Type] of the component, the [Variable] of the connection, and then click the [OK] button.

- ✓ Properties dialog of coil



The user can modify the [Type] of the component, the [Variable] of the connection, and then click the [OK] button.

- ✓ Properties dialog of function/function block



Double-click a parameter in the [Parameter] list, and an input box will appear at the [Variable Connection] of the parameter. The user can directly input the desired constant, direct address, global variable or local variable name in this box, and then press Enter to confirm.

The user can also click the mouse or use the up and down arrow keys to select a parameter, and then press the Enter key to enter the input box to modify the variable.

KincoBuilder will perform strict syntax checking on the user's input, and will not accept incorrect input, including illegal variables, data type errors, memory area type errors, etc.

After all input is completed, click the [OK] button.

➢ Delete an component

① Mark the components you want to delete.

② Use any of the following methods to delete the marked components:

- Execute the menu instruction 【Ladder Diagram】→【Delete Components】.
- Click the icon on the toolbar.
- Click the right mouse button to execute the instruction 【Delete Component】in the right-click menu.
- Execute the menu instruction 【Edit】→【Delete】.
- Use the Delete key.

➢ Delete a network

① Mark any element in the net to be deleted.

② Use any of the following methods to delete the network where the marked component is located:

- Execute the menu instruction 【Ladder Diagram】→【Delete Network】.
- Click the icon on the toolbar.
- Click the right mouse button and execute the instruction 【Delete Network】in the right-click menu.
- Use the Ctrl+Del shortcut.。

➢ Select/Delete/Copy/Cut/Paste

In the LD editor, you can use the editing instructions in the [Edit] menu, including select all, copy, cut, paste, etc. In addition, clicking the right mouse button in the program editing area will pop up the right-click menu, and the user can also use the right-click menu instruction to edit.

Executing the [Select All] instruction will select all the content in the editing area. Use the mouse to drag in the editing area, or use the up, down, left and right arrow keys to move the cursor while pressing the SHIFT key, the content will be selected. The selected area will use black as the background color, and the graphic elements and text will be highlighted.

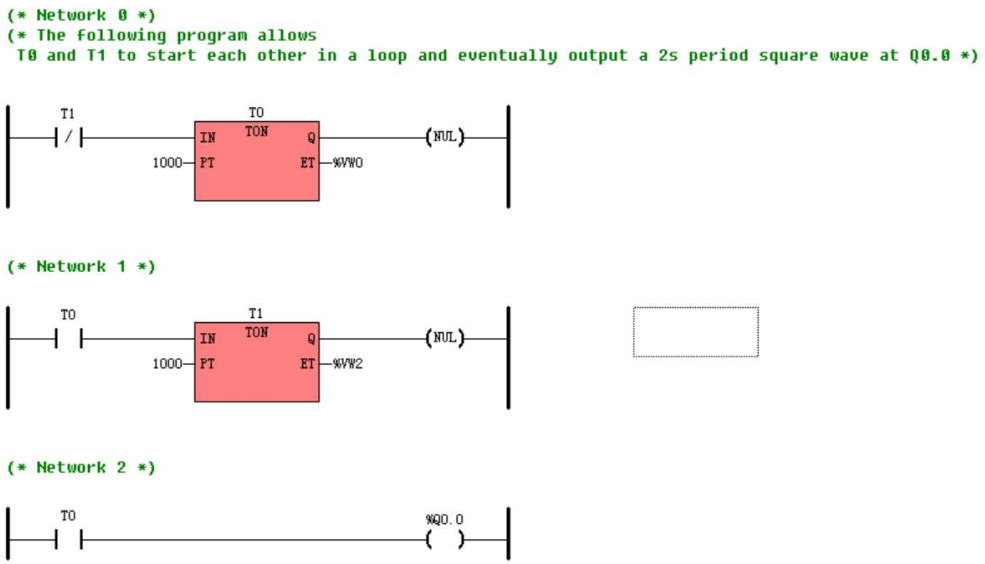
Use the Delete key or execute the [Delete] instruction to delete the selected content.

Use the shortcut key Ctrl+C or execute the [Copy] instruction to copy the selected content. The copied content can be pasted in the LD editor.。

The cut operation is equivalent to copying and deleting the selected content. Use the shortcut key Ctrl+X or execute the [Cut] instruction to cut the selected content.

After copying or cutting, mark a component at the position you want to paste, and then use the shortcut key Ctrl+V or execute the [Paste] instruction to paste the copied content to the corresponding position: if the mark is a network comment, Then it will be pasted above the net where the marked component is located; if another component is marked, it will be pasted below the net where the marked component is located.

5.2.4.3 LD program example



5.2.5 Monitor and debug programs 序

5.2.5.1 Online monitoring of LD programs

Note: Program editing is not allowed in online monitoring state.

If the current window is the LD editor, you can use any of the following methods to enter the online monitoring state:

- Execute the menu instruction 【Debug】→【Online Monitor】.
- Click the icon on the toolbar.
- Use the F6 shortcut.。

In the online monitoring state, the variable status display mode in the program is as follows::

- %I0.0 Indicates that the contact is open, %I0.0 Indicates that the contact is closed.
- %M1.0 Indicates that the coil is disconnected, %M1.0 Indicates that the coil is closed.
- The value of a non-switch variable in the program will be displayed directly to the right of the variable.

The following figure is an example of an online monitoring program:

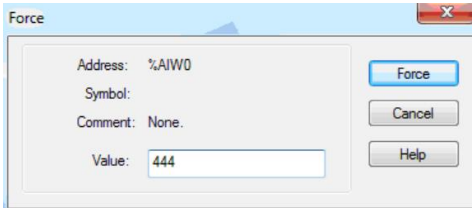
Note: Yellow locks represent mandatory values

5.2.5.2 Force a variable to be specified

The forcing function of KINCO-K series is described in detail in the section 4.7.3 about the forcing function.

When monitoring the LD program online, click the right mouse button on a variable in the program, and the following right-click menu will pop up. (Note: If the right-clicked variable is a non-switch variable, the instructions [Force to TRUE] and [Force to FALSE] in the menu will be invalid).

- **Force is TRUE:** Force the value of this variable (switch) to 1 (TRUE).
- **Force is FALSE:** Force the value of this variable (switch) to 0 (FALSE).
- **Force...:** After executing this instruction, the following dialog box will pop up::



Enter the value to be forced (constant of the corresponding data type) in the [Force Value] input box, and then click the [Force] button.

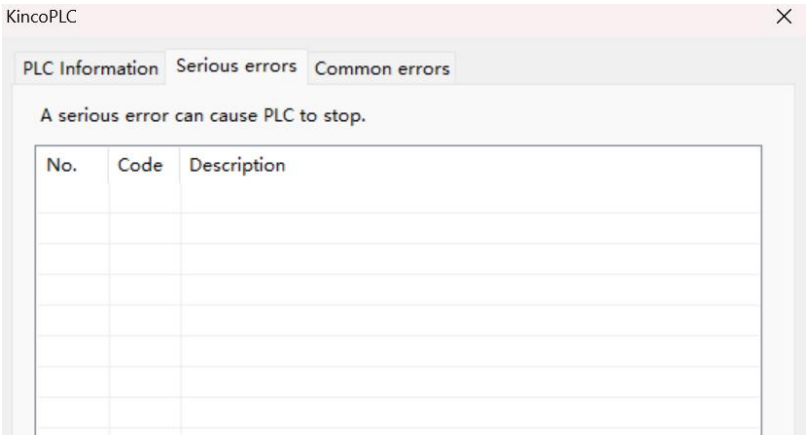
For the representation format of constants, please refer to 3.4 Constants

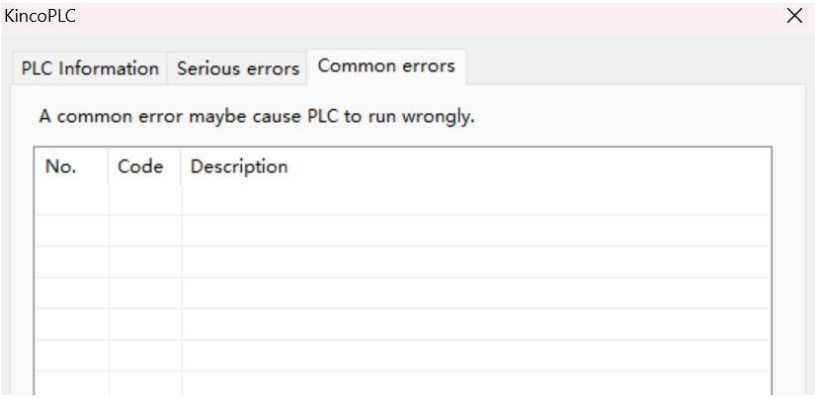
- **Unenforce:** Unenforce the variable.
- **Unenforce all:** Unenforce all variables currently in the connected CPU.

5.2.6 View PLC errors and faults

Click [PLC Information] under the [PLC] menu of the KincoBuilder programming software, and the following window will appear:

KincoPLC
PLC Information
Serious errors
Common errors
CPU
Type:
K606
Operating Mode:
RUN
Firmware:
8.7.2.0
Project:
Standard, V1.6
IP:
172.17.6.110
Subnet mask:
255.255.255.0
Gateway:
192.168.6.1
MAC:
54:EE:75:38:D6:6A
Modules
Positi...
Module
FW version
Remark
1
K606-24DT
8.7.2.0
Refresh
Help


KincoPLC
PLC Information
Serious errors
Common errors
A serious error can cause PLC to stop.
No.
Code
Description



As shown in the figure above, it contains 3 pieces of content: PLC information, serious errors, and common errors.

PLC information: including the type of CPU, current running status, firmware version, IP address, etc.

Critical errors: Error codes and error descriptions of critical errors that have occurred.

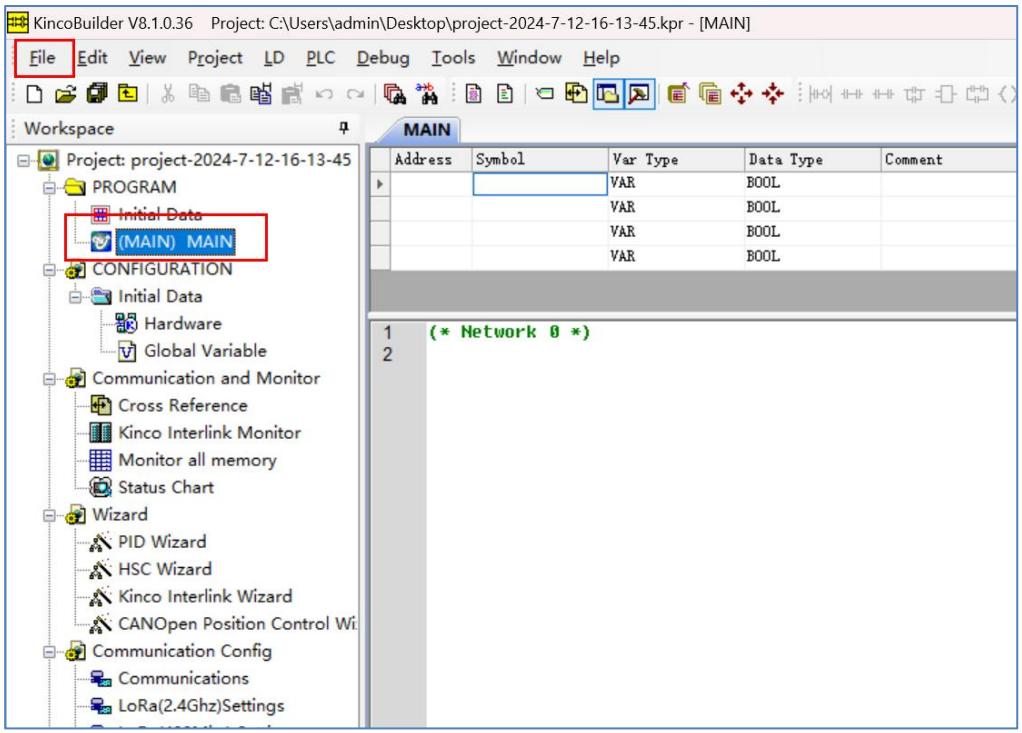
Common errors: Error codes and error descriptions of common errors that have occurred.

For the introduction of serious errors and common errors, please refer to the introduction of Chapter 13 Troubleshooting.

5.3 How to use KincoBuilder to create main program, subprogram and interrupt program

- **Build the main program**

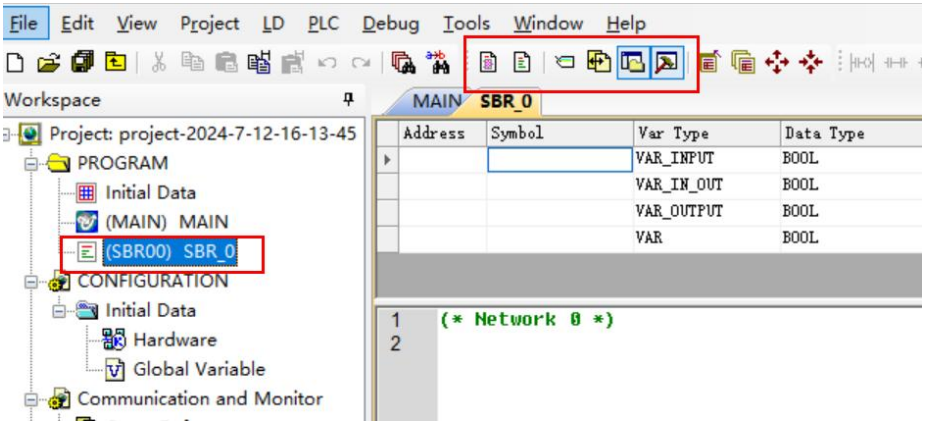
Open KincoBuilder software, click [New Project] under the [File] menu, the software will automatically generate a main program called MAIN, the execution of all programs in the project can only be called by MAIN, the user can program in the main program of MAIN. If you want the program structure to be more organized, you can create different subroutines to be called by the main program MAIN



- **Create subroutines**

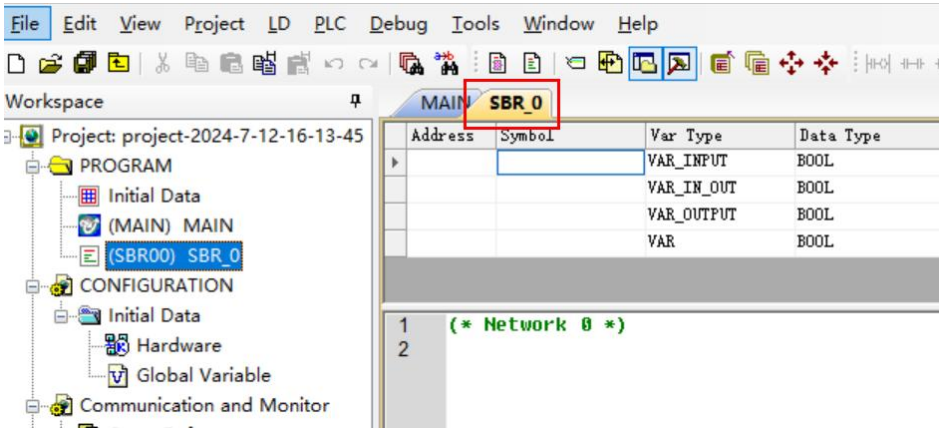
1)new subroutine

Click the icon in the figure below, and a prompt "Create a new subroutine" will appear. After clicking, a new subroutine named SBR_0 is automatically created.

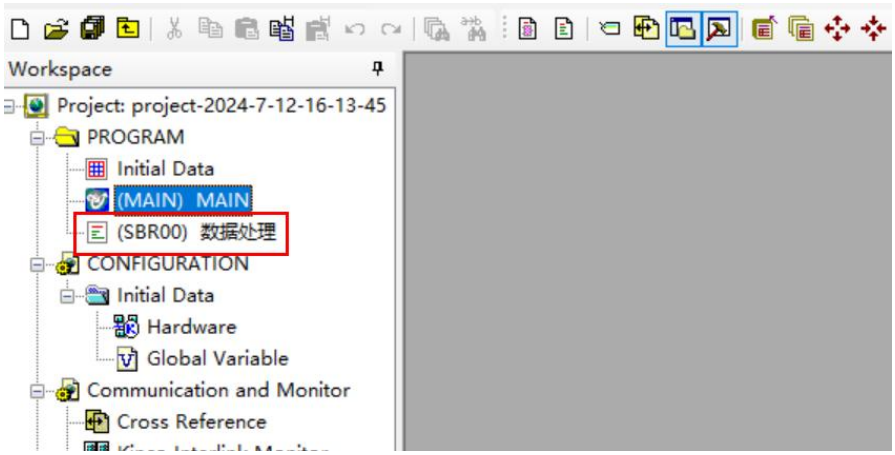


2)Change the name of the subroutine

Right-click on the red box in the image below and click to close the window.。

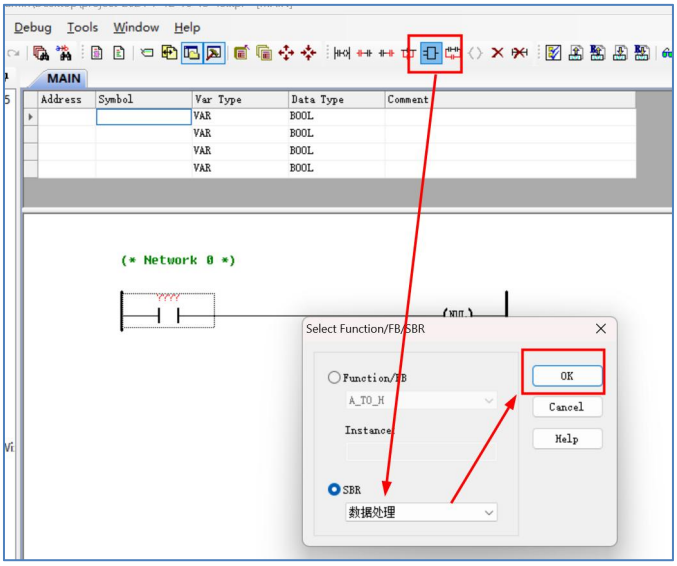


Right-click the red box in the picture below, click Rename, and enter the name you want to name.

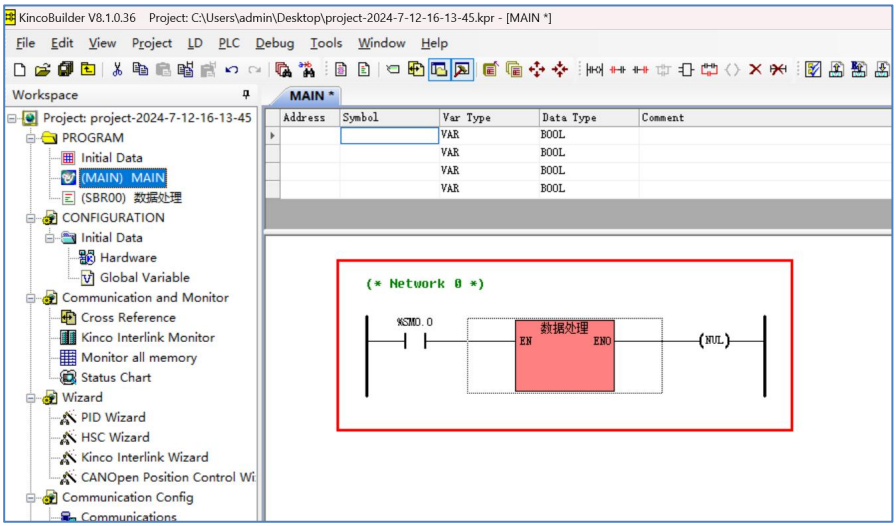


3)call subroutine

The subprogram needs to be called in the main program MAIN, otherwise the subprogram will not be executed. First, point the mouse to SM0.0 in the network of the program, and then operate in the order of the arrows in the figure.。



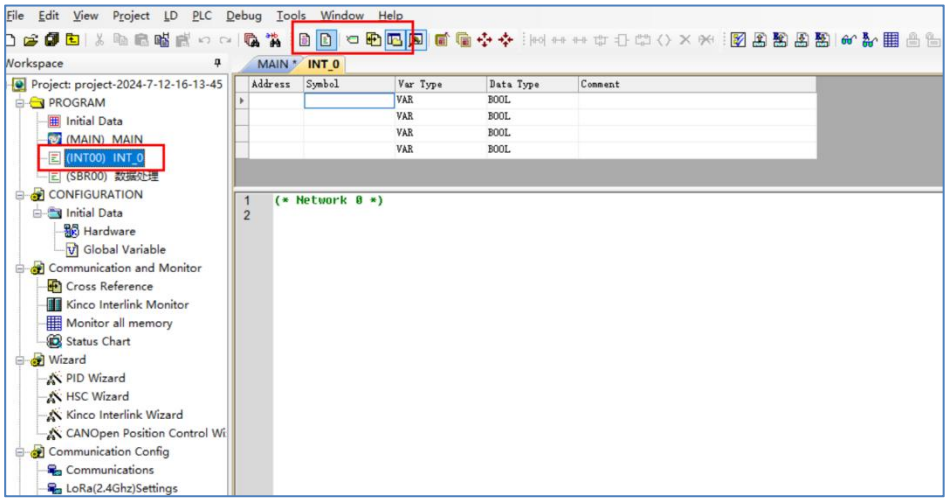
The following figure shows the network of the main program calling subroutines.



- **Create an interrupt routine**

1)Create a new interrupt program

Click the icon in the figure below, and the prompt "Create a new interrupt program" will appear. After clicking, a new subroutine named INT_0 will be created automatically.。

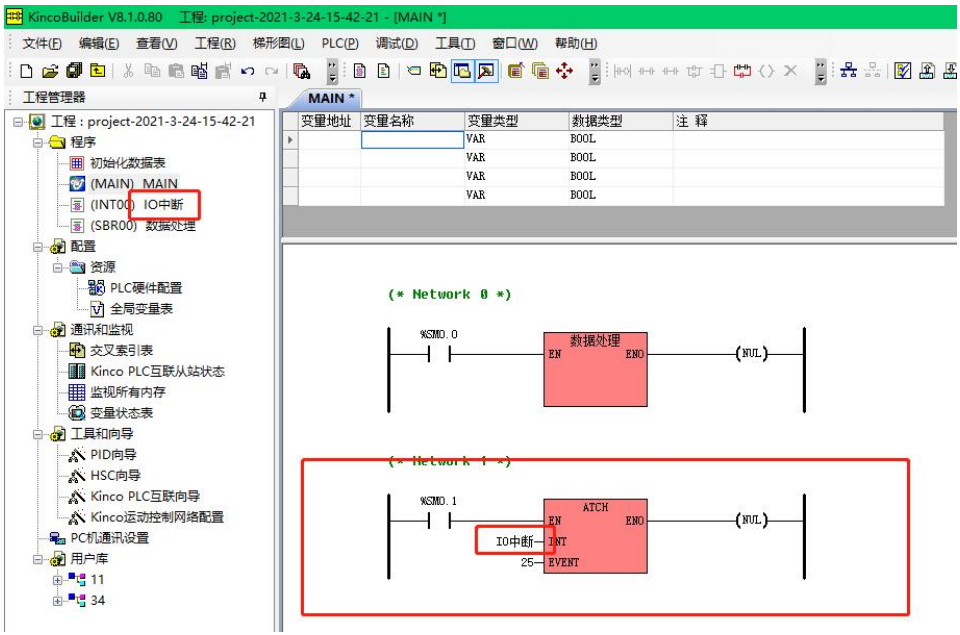


2)Change the name of the interrupt program

The process is the same as modifying the subroutine name, so it will not be repeated here.。

3)disconnection program

To use the disconnection program, it is necessary to connect the interrupt through the relevant instruction in the main program MAIN.



For the introduction to the use of interrupts, please refer to the chapter 6.12 Interrupt Instructions.

Chapter 6 Basic application instruction set

K series PLC supports the basic instructions of the IEC 61131-3 standard and most of its functions/function blocks, and the programming style meets the requirements of the IEC 61131-3 standard. At the same time, according to the actual needs, the standard instructions are appropriately expanded, which can meet the actual needs of different users and multi-application fields.

6.1 Instruction Set Overview

This chapter gives a detailed description of all the instructions in the instruction set, and also provides specific usage examples for most of the instructions. The instruction description includes two formats of LD and IL.

In the LD format, the following description does not specifically mention energy flow, the meaning of energy flow is fixed, and the state of each instruction on a network controls the flow of energy flow on this network. We can think of it as virtual input for some instructions (instructions without EN, ENO operands), and its type is BOOL. For the convenience of description, in the following we will simply refer to whether the energy flow reaches a certain point as the value of the energy flow at this point is 1 or 0.

In addition, in the LD format, the EN, ENO operands and their data types are not described below, because they are all BOOL type and their meanings are also fixed. EN means "enable". If the EN value of a function/function block is 1, the function/function block is executed, the energy flow continues to flow forward, and the output at the ENO terminal is 1. If the value of EN is 0, the function/function block is not executed, and the output is 0 at the ENO terminal.

As mentioned in 5.1.2.2 about CR, in the IL program, each instruction will have a corresponding impact on the CR value after execution. This chapter describes the effect of each IL instruction on the CR value, using the "group abbreviation" notation specified in 5.1.2.2.

6.2 bit instruction

6.2.1 standard contact

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	normally open contact	*bit* ⊣ ⊢		☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	normally closed contact	*bit* ⊣/⊢		
IL	LD	LD *bit*	C	
	AND	AND *bit*	P	
	OR	OR *bit*		
	LDN	LDN *bit*	C	
	ANDN	ANDN *bit*	P	
	ORN	ORN *bit*		

operand	input Output	type of data	allowed memory area
bit	Input	BOOL	I、Q、V、M、SM、L、T、C、RS、SR、constant

- LD

The function of the normally open contact is: if the bit value is 1, the contact is closed, and the energy flow continues to pass backward; if the bit value is 0, the contact is disconnected, and the energy flow is also cut off.

The function of the normally closed contact is: if the bit value is 0, the contact is closed, and the energy flow continues to pass backward; if the bit value is 1, the contact is disconnected, and the energy flow is also cut off.

- IL

Normally open contacts are provided by LD, AND, OR instructions.

The LD instruction loads the bit value into CR as the new CR value;

The AND instruction performs an AND operation on the CR value and the bit value, and uses the operation result as a new CR value;

The OR instruction performs an OR operation on the CR value and the bit value, and uses the operation result as a new CR value.

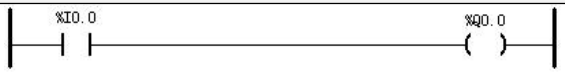
Normally closed contacts are provided by LDN, ANDN, ORN instructions.

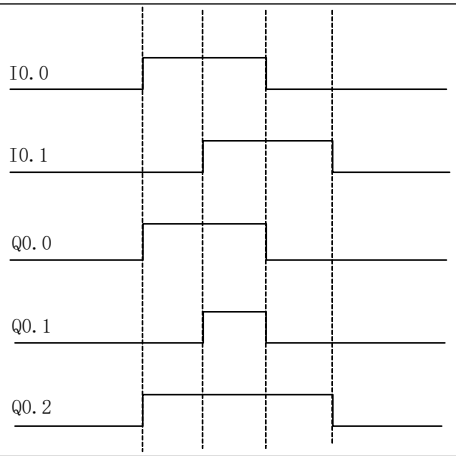
The LDN instruction inverts the bit value, and then loads the result into CR as the new CR value;

The ANDN instruction inverts the bit value, then performs an AND operation on the result and the CR value, and uses the operation result as the new CR value;

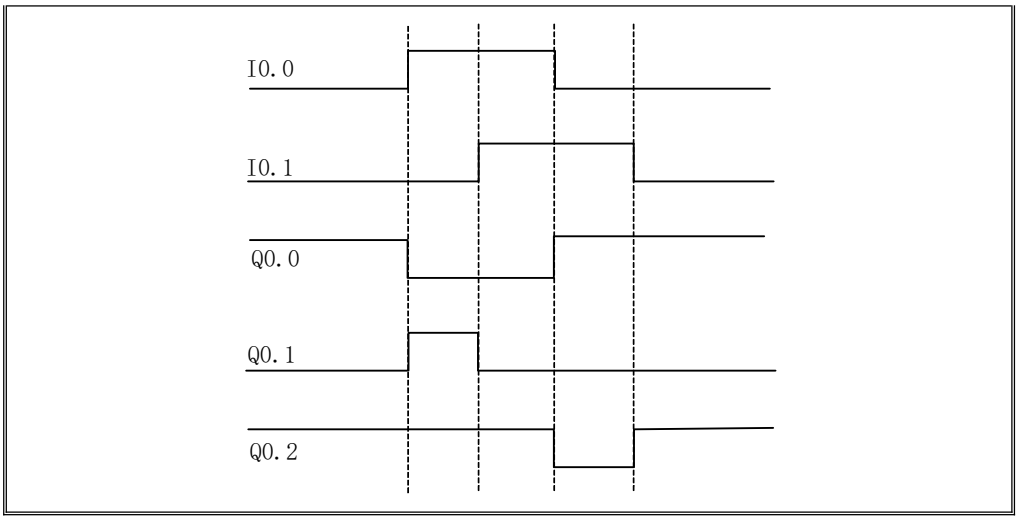
The ORN instruction inverts the bit value, then "ORs" the result with the CR value, and uses the operation result as the new CR value.

- Instruction usage example

LD	IL
	LD %I0.0 ST %Q0.0
	LD %I0.0 AND %I0.1 ST %Q0.1
	LD %I0.0 OR %I0.1 ST %Q0.2
Timing diagram	



LD	IL
	LDN %I0.0 ST %Q0.0
	LD %I0.0 ANDN %I0.1 ST %Q0.1
	LD %I0.0 ORN %I0.1 ST %Q0.2
Timing diagram	



6.2.2 Immediate contact

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	Immediately normally open contacts	*bit* —| I |—		☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	Immediately normally closed contacts	*bit* —|/I|—		
IL	LDI	LDI *bit*	C	
	ANDI	ANDI *bit*	P	
	ORI	ORI *bit*		
	LDNI	LDNI *bit*	C	
	ANDNI	ANDNI *bit*	P	
	ORNI	ORNI *bit*		

operand	Input/Output	Data type	Allowed memory area
bit	Input	BOOL	I (CPU)

When the instruction is executed, the immediate contact directly reads the actual state of the physical input channel of the bit, but does not update the input image area. Compared with the ordinary contact instruction, the immediate contact instruction does not read the data in the input image area, so it will not be affected by the CPU scan cycle, and can quickly respond to the input signal.

Immediate contact instructions can only be used for the DI points of the CPU body. In the [PLC Hardware Configuration] of the user project, the setting of [I/O Settings] > [Input Filter] of the CPU module has no effect on the immediate contact instruction.

- LD

The function of the immediately normally open contact is: if the actual state value of the physical input channel of the bit is 1, the contact is closed, and the energy flow continues to be transmitted backward; otherwise, the contact is disconnected and the energy flow is also cut off.。

The function of the immediately normally closed contact is: if the actual state value of the physical input channel of the bit is 0, the contact is closed, and the energy flow continues to pass backward, otherwise the contact is disconnected and the energy flow is also cut off。

- IL

Immediately normally open contacts are provided by the LDI, ANDI, ORI instructions.

The LDI instruction loads the actual state value of the bit's physical input channel into CR and uses it as the new CR value;

The ANDI instruction performs an "AND" operation between the CR value and the actual state value of the physical input channel of the bit, and uses the operation result as a new CR value;

The ORI instruction performs an OR operation on the CR value and the actual state value of the physical input channel of the bit, and uses the operation result as a new CR value.

Immediate normally closed contacts are provided by the LDNI, ANDNI, ORNI instructions.

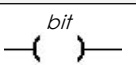
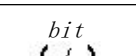
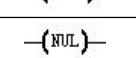
The LDNI instruction inverts the actual state value of the physical input channel of the bit, and then loads the result into CR as the new CR value;

The ANDNI order inverts the actual state value of the physical input channel of the bit, then performs an "AND" operation on the result and the CR value, and uses the operation result as the new CR value;

The ORNI instruction inverts the actual state value of the physical input channel of the bit, then performs an OR operation on the result and the CR value, and uses the operation result as the new CR value.

6.2.3 Normal output

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	coil			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	Negative coil			
	empty coil			
IL	ST	ST *bit*	U	
	STN	STN *bit*		

Operand	Input/output	Data type	Allowed memory area
bit	Output	BOOL	Q、V、M、SM、L

- LD

The role of the operating coil: assign the value of the energy flow on the left side of the coil to the bit.

The function of inverting the coil: invert the value of the energy flow on the left side of the coil and assign it to the bit.

The role of the empty coil: Indicates the end of a network in the LD program. This instruction is only for the convenience of user programming and does not perform specific operations.

- IL

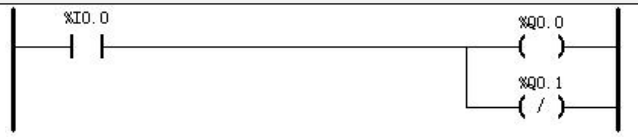
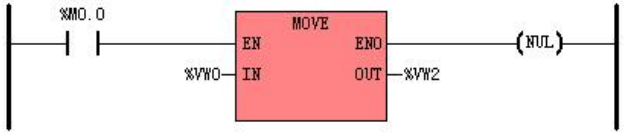
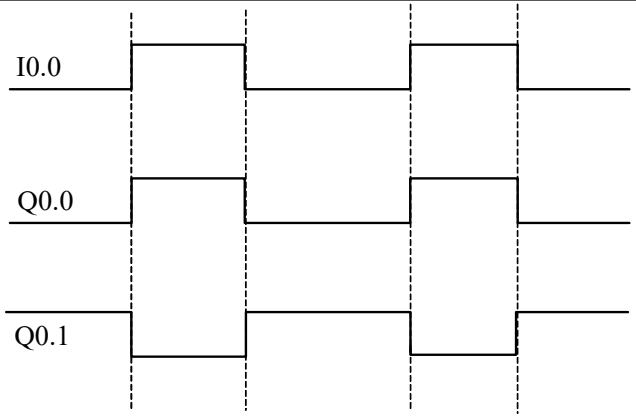
The coil is provided by the ST instruction, and the inverted coil is provided by the STN instruction.

The ST instruction is used to assign a CR value to a bit.

The STN instruction is used to invert the CR value and assign it to the bit.

The execution of ST and STN instructions has no effect on the CR value.

➢ Instruction usage example

<table>
<tr><td>LD</td><td>IL</td></tr>
<tr><td> 
</td><td>LD %I0.0
ST %Q0.0
STN %Q0.1</td></tr>
<tr><td> 
</td><td>LD %M0.0
MOVE %VW0, %VW2</td></tr>
<tr><td colspan="2">Timing diagram</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

6.2.4 Immidiate output

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
LD	Immediate coil	*bit* —(I)—		
IL	STI	STI*bit*	U	

Operand	Input/Output	Data type	Allowed memory area
bit	Output	BOOL	Q (CPU)

The immediate output instruction can only be used for the DO point of the CPU body.

- LD

When the instruction is executed, the value of the energy flow on the left side of the immediate coil is directly written into the physical output point of the bit for output, and the corresponding output image area is updated at the same time.

- IL

Immediate output is provided by the STI instruction.

The STI instruction writes the CR value directly into the physical output point of the bit for output, and updates the corresponding output image area at the same time.

The execution of the STI instruction has no effect on the CR value.

6.2.5 Set and reset

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	☑ K5

LD	Reset	*bit* —(R)—		☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	Set	*bit* —(S)—		
IL	Reset	R *bit*	U	
	Set	S *bit*		

Operand	Input/Output	Data type	Allowed memory area
bit	Output	BOOL	Q、V、M、SM、L

- LD

The function of the reset coil: if the value of the energy flow on the left side of the coil is 1, the bit value is set to 0, otherwise the bit value remains unchanged.

The function of setting the coil: if the value of the energy flow on the left side of the coil is 1, the bit value is set to 1, otherwise the bit value remains unchanged.

- IL

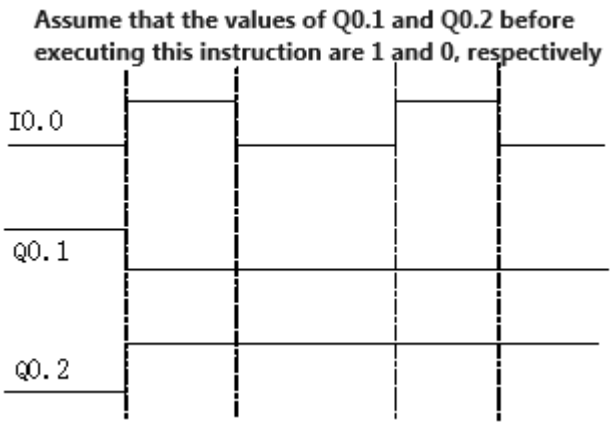
The function of the reset instruction is: if the CR value is 1, the bit value is set to 0, otherwise the bit value remains unchanged.

The function of the set instruction is: if the CR value is 1, the bit value is set to 1, otherwise the bit value remains unchanged.

The execution of R and S instructions does not affect the CR value.

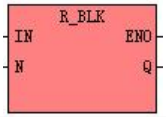
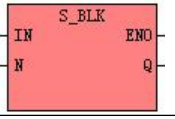
➢ Instruction usage example

LD	IL
	LD %I0.0 R %Q0.1 S %Q0.2

Timing diagram


6.2.6 Block set and block reset

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	Block reset			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	Block set			
IL	Block reset	R_BLK *N,Q*	U	
	Block set	S_BLK *N,Q*		

Operand	Input/Output	Data type	Allowed memory area
IN	Input	BOOL	energy flow
N	Input	INT	L、M、V、constant
Q	Output	BOOL	Q、V、M、SM、L

The maximum number of parameters N is 1024. The parameter Q is the starting address of a memory block, and the length of the memory block is N. Note that the entire memory block must be located in a legal

address range, otherwise the result will be unpredictable.

- LD

The role of R_BLK: If the IN value is 1, then the N consecutive bits starting from the specified bit address Q are set to 0, otherwise these bits remain unchanged.

The role of S_BLK: if the value of IN is 1, the consecutive N bits starting from the specified bit address Q are set to 1, otherwise these bits remain unchanged。

- IL

The role of R_BLK: If the CR value is 1, the consecutive N bits starting from the specified bit address Q are set to 0, otherwise these bits remain unchanged.

The role of S_BLK: If the CR value is 1, the consecutive N bits starting from the specified bit address Q are set to 1, otherwise these bits remain unchanged.

The execution of the R_BLK and S_BLK instructions does not affect the CR value.

6.2.7 Immediate Set and Immediate Reset

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	Immediate reset	*bit* —(RI)—		☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	Immediate set	*bit* —(SI)—		
IL	Immediate reset	RI*bit*	U	
	Immediate set	SI*bit*		

Operand	Input/Output	Data type	Allowed memory area
bit	Output	BOOL	Q (CPU)

Immediate set and immediate reset instructions can only be used for the DO point of the CPU body.

- LD

The function of resetting the coil immediately is: if the value of the energy flow on the left side of the coil is 1, the output image register and the physical output point corresponding to the bit are both set to 0, otherwise the values of the two remain unchanged.

The function of setting the coil immediately is: if the value of the energy flow on the left side of the coil is 1, the output image register corresponding to the bit and the physical output point are both set to 1, otherwise the values of the two remain unchanged.

- IL

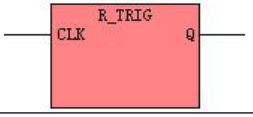
The function of the immediate reset instruction is: if the CR value is 1, the output image register corresponding to the bit and the physical output point are both set to 0, otherwise the values of the two remain unchanged.

The function of the immediate set instruction is: if the CR value is 1, the output image register corresponding to the bit and the physical output point are both set to 1, otherwise the values of the two remain unchanged.

The execution of RI and SI instructions does not affect the CR value.

6.2.8 edge detection

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	Rising edge detection			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK

	Falling edge detection			☑ K6
IL	Rising edge detection	R_TRIG	P	
	Falling edge detection	F_TRIG		

Operand	Input/Output	Data type	Allowed memory area
CLK (LD)	Input	BOOL	energy flow
Q (LD)	Output	BOOL	energy flow

- LD

  R_TRIG is used to detect the rising edge transition of the CLK input: if the CLK value produces a transition from 0 to 1, the Q output is 1 and remains for one scan period, and then the Q output is 0.

  F_TRIG is used to detect the falling edge transition of the CLK input: if the CLK value produces a transition from 1 to 0, the Q output is 1 and remains for one scan period, and then the Q output is 0.

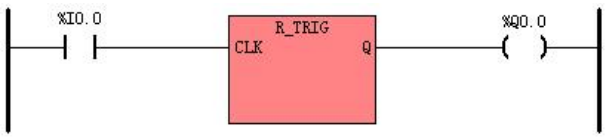
- IL

  R_TRIG is used to detect the rising edge transition of the CR value of the current point: if the CR value of the current point produces a transition from 0 to 1, the CR value is set to 1 immediately, otherwise the CR value is set to 0.

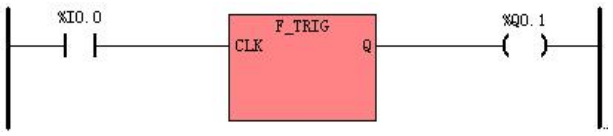
  F_TRIG is used to detect the falling edge transition of the CR value of the current point: if the CR value of the current point produces a transition from 1 to 0, the CR value is set to 1 immediately, otherwise the CR value is set to 0.

➢ Instruction usage example

LD	IL

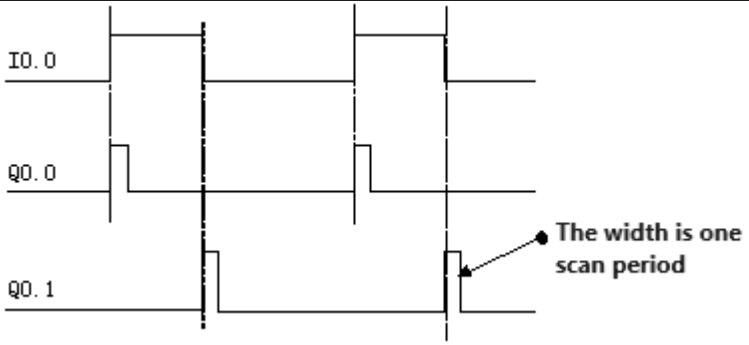



```
    LD    %I0.0
R_TRIG
    ST    %Q0.0
```



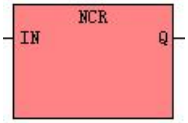

```
    LD    %I0.0
    F_TRIG
    ST    %Q0.1
```

Timing diagram



6.2.9 NCR (Negate)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	取反			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	取反	NCR	P	

Operand	Input/Output	Data type	Allowed memory area
IN	Input	BOOL	Energy flow
Q	Output	BOOL	Energy flow

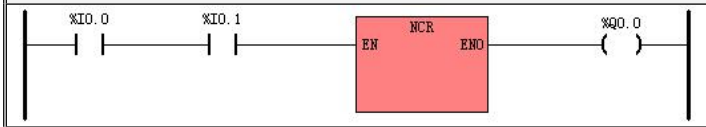
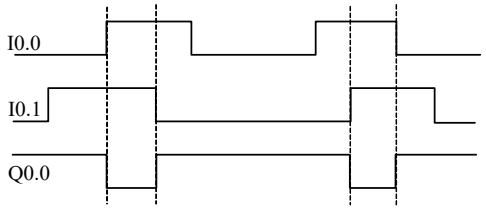
- LD

  This instruction is used to change the state of the energy flow: invert the value of the energy flow at the input and output.

- IL

  This instruction is used to change the CR value: invert the CR value and use the result as the new CR value.

➢ Instruction usage example

<table>
<tr><td>LD</td><td>IL</td></tr>
<tr><td> 
</td><td>LD %I0.0
AND %I0.1
NCR
ST %Q0.0</td></tr>
<tr><td colspan="2">Timing diagram</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

6.2.10 Bistable flip-flop

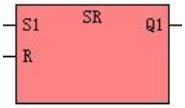
Bistable flip-flop is one of the functional blocks defined in the IEC61131-3 standard. There are two types of SR flip-flops (set priority flip-flops) and RS flip-flops (reset priority flip-flops).

For the use of function blocks and their instances, please refer to the description in 3.6.5.

6.2.10.1 SR (set priority flip-flop)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	☑ K5

LD	set priority flip-flop	*SRx*  		☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	set priority flip-flop	LD *S1* SR *SRx*, *R*	P	

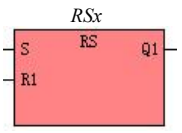
Operand	Input/Output	Data type	Allowed memory area
SRx	-	SR Trigger instance	SR
S1	Input	BOOL	Energy flow
R	Input	BOOL	I、Q、M、V、L、SM、T、C、RS、SR
OUT	Output	BOOL	Energy flow

The input of the set end of the SR flip-flop is S1, and the input of the reset end is R. Its truth table is as follows:

S1	*R*	Output Q1 and SRx status value
0	0	keep the previous state
0	1	0
1	0	1
1	1	1

6.2.10.2 RS (Reset priority flip-flop)

➢ Instruction and its operand description

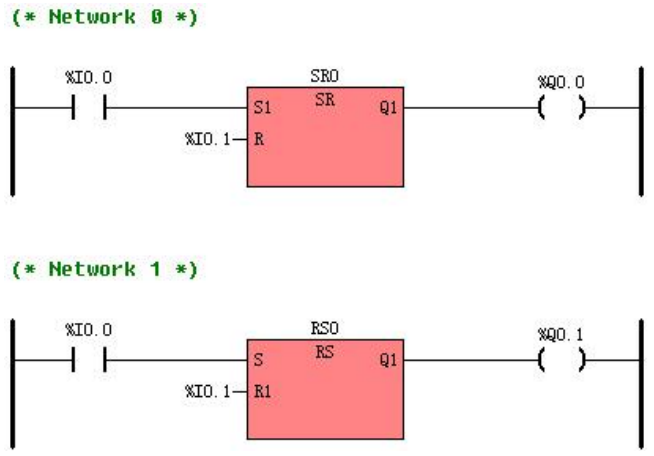
	Name	Instraction format	Affect CR value	
LD	Reset priority flip-flop	*RSx*  		☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	Reset priority flip-flop	LD *S* RS *RSx*, *R1*	P	

Operand	Input/Output	Data type	Allowed memory area
RSx	-	RS Trigger instance	RS
S	Input	BOOL	Energy flow
R1	Input	BOOL	I、Q、M、V、L、SM、T、C、RS、SR
OUT	Output	BOOL	Energy flow

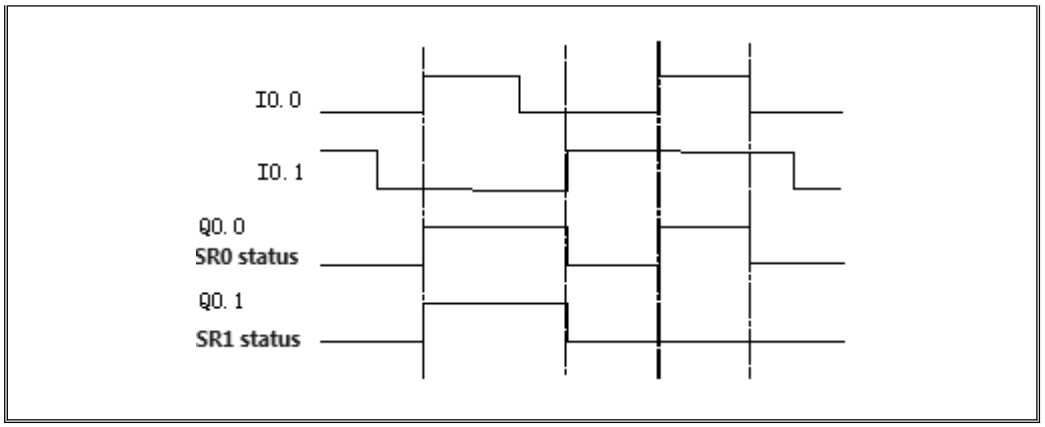
The input of the set end of the RS flip-flop is S, and the input of the reset end is R1. The truth table of the RS flip-flop is as follows:

R1	S	OUT and RSx status values
0	0	keep the previous state
0	1	1
1	0	0
1	1	0

6.2.10.3 RS、SR usage example

<table>
<tr><th>LD</th><th>IL</th></tr>
<tr><td> 
</td><td>(* Network 0 *)
LD %I0.0
SR SR0, %I0.1
ST %Q0.0

(* Network 1 *)
LD %I0.0
RS RS0, %I0.1
ST %Q0.1</td></tr>
<tr><td colspan="2">Timing diagram</td></tr>
</table>



6.2.11 ALT (Reverse)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	Reverse			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	Reverse	ALT *Q*	U	

Operand	Input/Output	Data type	Allowed memory area
IN (LD)	Input	BOOL	Energy flow
Q	Output	BOOL	Q、V、M、SM、L

- LD

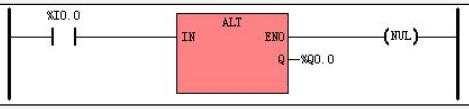
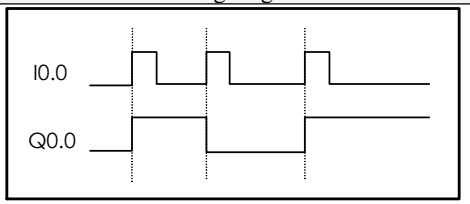
  If the input IN has a rising edge transition, Q is inverted immediately, otherwise Q remains unchanged.

- IL

  If the CR value of the current point has a rising edge transition, Q is immediately inverted, otherwise Q remains unchanged.

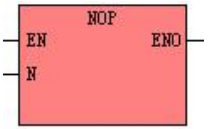
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD	IL
	LD %I0.0 ALT %Q0.0
Timing diagram	
	

6.2.12 NOP (no-op)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
LD	No-op			
IL	No-op	NOP *N*	U	

Operand	Input/Output	Data type	Allowed memory area
N	Input	INT	Constant (positive)

The NOP instruction only allows the CPU to generate N no-ops, and will not affect the execution result of the user program. The parameter N specifies the number of times to perform the null operation, and it must be an INT type constant greater than 0.

The user can call the NOP instruction in the program to perform imprecise delay. The specific delay effect needs to be determined by the user by himself.。

6.2.13 bracket modifier

- Instruction and its operand description

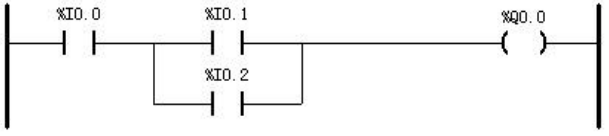
	Name	Instraction format	Affect CR value	
IL	AND(	AND(	U	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	OR(	OR(		
	)	)	P	

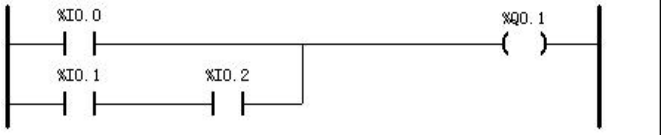
Bracket modifiers are only available in IL. There are only simple expressions in IL language, and it is impossible to use complex expressions as operands as in LD, ST and other languages, so brackets are defined in the IEC61131-3 standard to deal with some complex expressions. "AND(" or "OR(" are used in conjunction with ")".

In the IL program, before executing the instructions between "AND(" and ")", the CR value is temporarily stored, and then the instructions in parentheses are executed. After the execution, perform an AND operation on the result with the temporarily stored CR value, and use the operation result as a new CR value.

Similarly, before executing the instruction between "OR(" and ")", first temporarily store the CR value, and then execute the instruction in parentheses. and use the result of the operation as the new CR value.

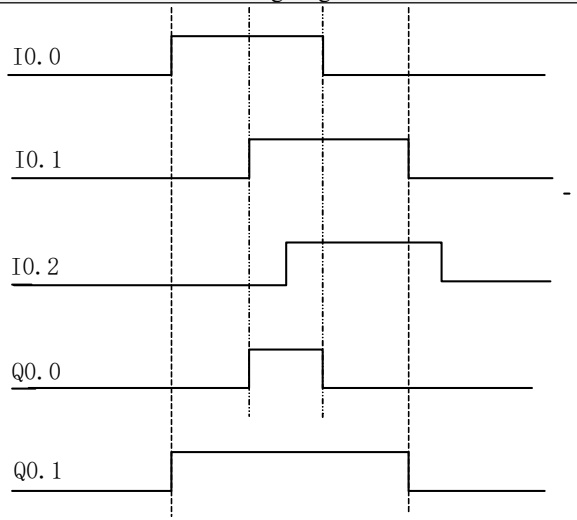
- Instruction usage example

LD	IL
	LD %I0.0 AND(LD %I0.1 OR %I0.2) ST %Q0.0



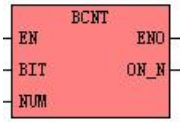

```
LD    %I0.0
OR(
LD    %I0.1
AND   %I0.2
)
ST    %Q0.1
```

Timing diagram



6.2.14 BCNT (Position Statistics)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	BCNT			☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	BCNT	BCNT BIT, ON_N, NUM	U	

Operand	Input/Output	Data type	Allowed memory area
BIT	Input	BIT	I、Q、M、V、L
NUM	Input	WORD	I、Q、M、V、L、constant
ON_N	Output	WORD	I、Q、M、V、L

This instruction is used to count the number of consecutive NUM bits starting from the BIT address, the number of bits whose value is equal to TRUE, and output the statistics to ON_N.**Note: 1 <= *NUM* <=256。**

- LD

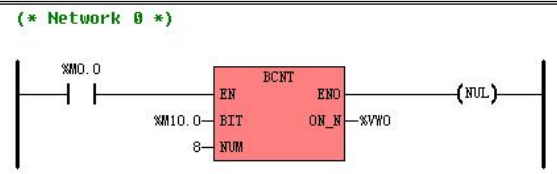
  If EN is 1, the instruction will set statistics and output the result to ON_N.

- IL

  If the CR value is 1, the instruction will set the statistics and output the result to ON_N.

  The execution of this instruction has no effect on the CR value.

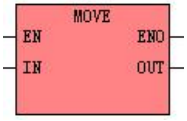
➢ Instruction usage example

LD		If M0.0 is TRUE, execute the BCNT instruction: count how many of the 8 bits starting from M10.0 are equal to TRUE, and store the result in VW0xian
IL	LD %M0.0 BCNT %M10.0, 8, %VW0	

6.3 Assignment instruction

6.3.1 MOVE (assign)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK
LD	MOVE			

IL	MOVE	MOVE IN, OUT	U	☑ K6

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer
OUT	Output	BYTE、WORD、DWORD、INT、DINT、REAL	Q、M、V、L、SM、AQ、Pointer

This instruction performs assignment operation, and the data types of its parameters IN and OUT must be the same.

- LD

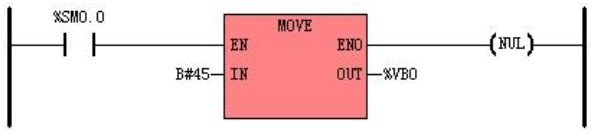
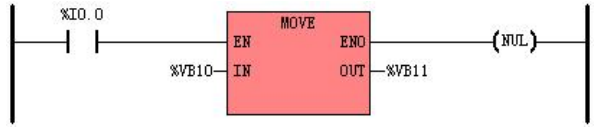
  If EN is 1, the instruction assigns the value of the input variable IN to the output variable OUT.

- IL

  If the value of CR is 1, the instruction assigns the value of the input variable IN to the output variable OUT.

  The execution of this instruction has no effect on the CR value.

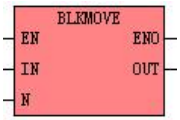
➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>SM0.0 is fixed at 1, so the MOVE instruction is always executed: the BYTE type constant B#45 is assigned to VB0.</td></tr>
<tr><td> 
</td><td>I0.0 is 0: MOVE instruction is not executed.
I0.0 is 1: The value of VB10 is assigned to VB11.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* Create CR with a value of 1 *)
MOVE B#45, %VB0 (* VB0 is assigned the value B#45 *)</td></tr>
</table>

```
LD     %I0.0          (*Creat CR with the value of I0.0  *)
MOVE   %VB10, %VB11   (* If CR is 1: the value of VB10 is assigned to VB11  *)
   (*If CR is 0: MOVE instruction is not executed, the value of VB11 remains unchanged *)
```

6.3.2 BLKMOVE (block transfer)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	BLKMOVE			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	BLKMOVE	BLKMOVE *IN*, *OUT*,*N*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、AI、AQ、T、C、HC
N	Input	BYTE	I、Q、M、V、L、SM、constant
OUT	Output	BYTE、WORD、DWORD、INT、DINT、REAL	Q、M、V、L、AQ

IN and OUT are the starting addresses of variable-length memory blocks, and the entire memory block is not allowed to fall into an illegal memory area, otherwise the result is unpredictable

The data types of parameters IN and OUT must be the same. This instruction is used to transfer the values of consecutive N variables starting from the address IN to the consecutive N variables starting from the address OUT. The data types of these variables are consistent with IN and OUT.

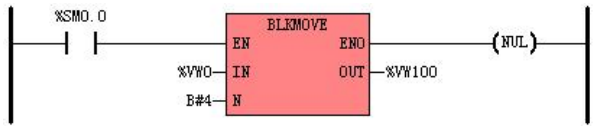
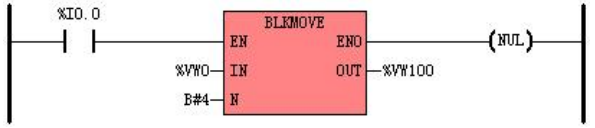
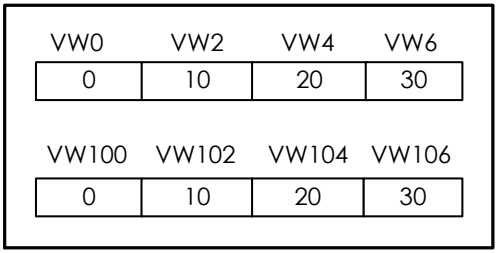
- LD

  If EN is 1, the instruction is executed。

- IL

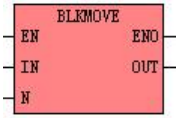
If the CR value is 1, the instruction is executed. The execution of this instruction has no effect on the CR value.

➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>SM0.0 is always 1, so the BLKMOVE instruction is always executed: the data in VW0 to VW6 are sequentially transferred to VW100 to VW106.</td></tr>
<tr><td> 
</td><td>I0.0 is 0: BLKMOVE is not performed.
I0.0 is 1: The data in VW0 to VW6 are transferred to VW100 to VW106 in sequence.</td></tr>
<tr><td rowspan="2">IL</td><td colspan="2">LD %SM0.0 (* Create CR, Value is 1 *)
BLKMOVE %VW0, %VW100, B#4 (*The data in VW0 to VW6 is transferred to VW100 to VW106 *)</td></tr>
<tr><td colspan="2">LD %I0.0 (*Create CR with the value of I0.0 *)
BLKMOVE %VW0, %VW100, B#4(*If CR is 1: Data in VW0 to VW6 is transferred to VW100 to VW106 *)
(*If CR is 0: do not execute BLKMOVE instruction *)</td></tr>
<tr><td>Result</td><td colspan="2">If BLKMOVE is executed, the result is as follows:

</td></tr>
</table>

6.3.3 BLKMOVE (block transfer, pointer parameter)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	BLKMOVE			☑ K6S ☑ MK ☑ K6
IL	BLKMOVE	BLKMOVE *IN, OUT, N*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、POINTER	V、L
N	Input	INT	V、L、Constant
OUT	Output	BYTE、POINTER	V、L

IN and OUT are the starting addresses of memory blocks of variable length. The entire memory block is not allowed to fall into the illegal memory area; otherwise, the result is unpredictable.

If the parameter uses a pointer, it must be a pointer to a BYTE type variable in the V zone!

This instruction is used to pass the values of N consecutive variables starting at address IN to N consecutive variables starting at address OUT.

- LD
- If EN is 1, the instruction is executed.
- IL
- If the CR value is 1, the instruction is executed. The execution of this command has no effect on the CR value.

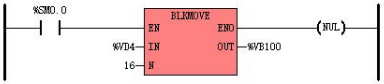
➢ Instruction usage example

LD	

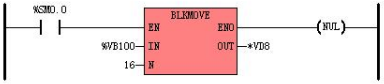
6.3.4 FILL (block assignment)

- Instruction and its operand description

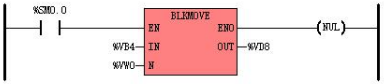
```
(* Network 0 *)
(* Passes 16 consecutive variable values from the address represented by
the pointer VD4 (i.e., VB400) to a variable from VB100. *)
```

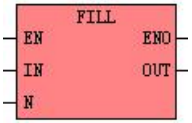



```
(* Network 1 *)
(* The values of 16 consecutive variables, starting with VB100,
 are passed to the variable starting at the address represented by the pointer VD8 (i.e. VB800) *)
```




```
(* Network 2 *)
(* The values of consecutive VW0 variables starting at the address represented
by the pointer VD4 (i.e., VB400) are passed to the variable starting at the address
represented by the pointer VD8 (i.e., VB800) *)
```



	Name	Instraction format	Affect CR value	
LD	FILL			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	FILL	FILL*IN*, *OUT*,*N*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、WORD、DWORD	Constant
N	Output	BYTE	Constant
OUT	Output	BYTE、WORD、DWORD	M、V、L

The OUT parameter is the starting address of a variable-length memory block. The entire block of memory cannot fall into the illegal memory area, otherwise the result will be unpredictable.。

This instruction is used to assign N consecutive byte variables starting from address OUT to constant IN.

Note that OUT is the starting address of a variable-length memory block, and the entire memory block cannot fall into an illegal memory area, otherwise the result will be unpredictable.

- LD

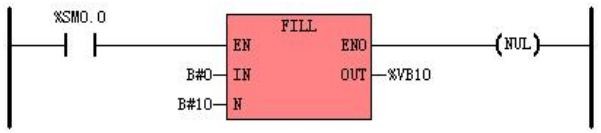
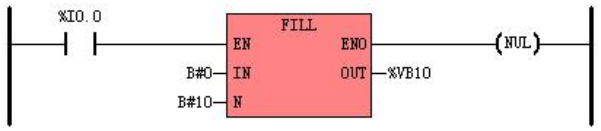
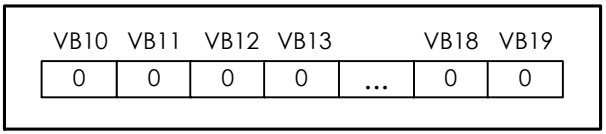
  If EN is 1, the instruction is executed。

- IL

  If the CR value is 1, the instruction is executed.

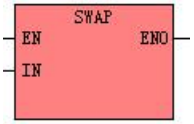
  The execution of this instruction has no effect on the CR value。

➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>SM0.0 is always 1, so the FILL instruction is always executed: a total of 10 variables from VB10 to VB19 are assigned as B#0.</td></tr>
<tr><td> 
</td><td>I0.0 is 0: FILL instruction is not executed.
I0.0 is 1: A total of 10 variables from VB10 to VB19 are assigned to B#0。</td></tr>
<tr><td rowspan="2">IL</td><td colspan="2">LD %SM0.0 (*Create CR with a value of 1 *)
FILL B#0, %VB10, B#10 (* A total of 10 variables from VB10 to VB19 are assigned the value B#0 *)</td></tr>
<tr><td colspan="2">LD %I0.0 (* Create CR with the value of I0.0 *)
FILL B#0, %VB10, B#10
(*CR is 1, a total of 10 variables from VB10 to VB19 are assigned the value B#0 *)
(* If CR is 0: the FILL command is not executed *)</td></tr>
<tr><td>Result</td><td colspan="2">If FILL is executed, the result is as follows:
 
</td></tr>
</table>

6.3.4 SWAP (Swap)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	SWAP			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	SWAP	SWAP*IN*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input/Output	WORD、DWORD	Q、M、V、L、SM

If the parameter IN is of WORD type, this instruction is used to exchange the high byte and low byte of IN;

If the parameter IN is of DWORD type, this instruction is used to exchange the high word and low word of IN.

- LD

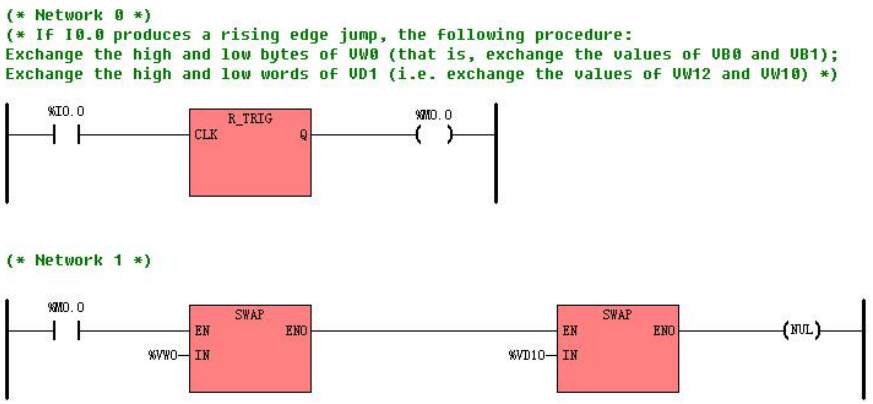
  If EN is 1, the instruction is executed。

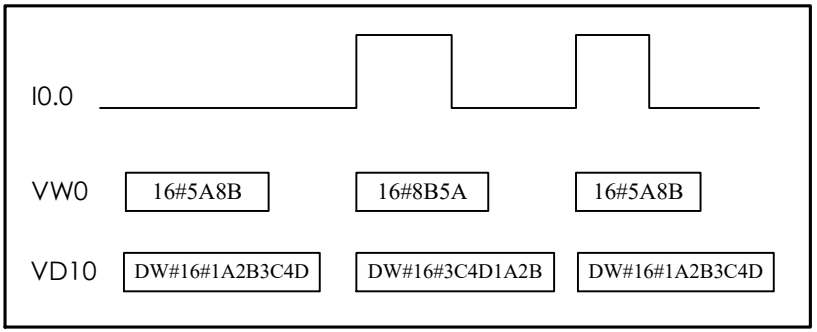
- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction has no effect on the CR value.

➢ Instruction usage example

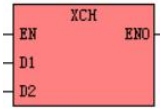
<table>
<tr><td>LD</td><td>


</td></tr>
<tr><td>IL</td><td>(* Network 0 *)
LD %I0.0
R_TRIG (*If I0.0 generates a rising edge transition *)
SWAP %VW0 (*then exchange the high and low bytes of VW0 *)
SWAP %VD10 (*Then exchange the high and low words of VD10 *)</td></tr>
<tr><td>Result</td><td>Assume that the initial value of VW0 is W#16#5A8B, and the initial value of VD10 is DW#16#1A2B3C4D。


</td></tr>
</table>

6.3.5 XCH (two data exchanges)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	XCH			☑ K2 ☑ KS ☑ MK ☑ K6
IL	XCH	XCH *D1,D2*	U	

Parameter	Input/Output	Data type	Allowed memory area
D1	Input/Output	BYTE、WORD、DWORD、INT、DINT、REAL	Q、M、V、L
D2	Input/Output	BYTE、WORD、DWORD、INT、DINT、REAL	Q、M、V、L

This instruction is used to exchange the values in D1 and D2 with each other.

Note: The data types of D1 and D2 must be the same!

- LD

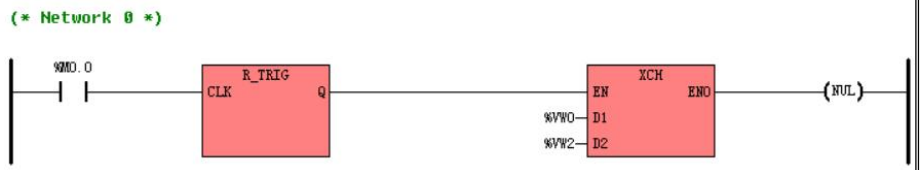
  If EN is 1, the instruction is executed.。

- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction has no effect on the CR value.

➢ Instruction usage example

LD	

IL	(* Network 0 *) LD %M0.0 R_TRIG (*If M0.0 generates a rising edge transition *) XCH %VW0, %VW2 (*then swap the values of VW0 and VW2 with each other *)
Result	Assume that the value of VW0 is 1234 and the value of VW2 is 5678 before execution. The result after one execution is: the value of VW0 is 5678, and the value of VW2 is 1234.

6.4 compare instruction

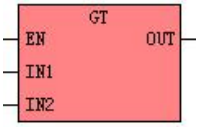
For all comparison instructions, BYTE, WORD, DWORD types use unsigned data for comparison, and other data types use signed data for comparison.

For all comparison instructions, BYTE, WORD, DWORD types use unsigned data for comparison, and other data types use signed data for comparison.

If the input parameter is a real number (REAL type), it is impossible to compare the two real numbers exactly because of rounding errors. When the difference between the absolute values of the two real numbers is within [-0.000001, 0.000001], the PLC considers the two real numbers to be equal, otherwise they are not equal. For the error, please refer to the relevant description in 3.2 Data Types.

6.4.1 GT (Greater)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	GT			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	GT	GT *IN1*, *IN2*	P	

Parameter	Input/Output	Data type	Allowed memory area

IN1	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、Constant、Pointer
IN2	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、Constant、Pointer
OUT (LD)	Output	BOOL	energy flow

The data types of parameters IN1 and IN2 must be the same.

- LD

If EN is 1, the instruction is executed: if IN1 is greater than IN2, the output is 1 at OUT, otherwise the output is 0;

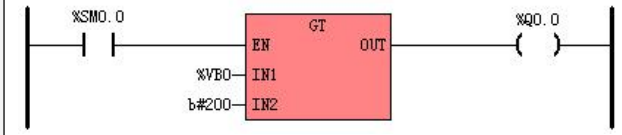
If EN is 0, the instruction is not executed, and the output is 0 at the OUT terminal.

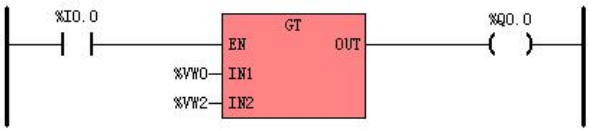
- IL

If the CR value is 1, the instruction is executed: if IN1 is greater than IN2, the CR is set to 1, otherwise the CR is set to 0;

If the CR value is 0, the instruction is not executed, and the CR value remains 0.

➢ Instruction usage example

LD		Since SM0.0 is fixed at 1, GT is always executed: if the value of VB0 is greater than B#200, Q0.0 is set to 1, otherwise Q0.0 is set to 0.

		If I0.0 is 0: GT is not executed, and Q0.0 is set to 0. If I0.0 is 1: If the value of VW0 is greater than the value of VW2, Q0.0 is set to 1, otherwise Q0.0 is set to 0。
IL	LD %SM0.0 (*Create CR with a value of 1 *) GT %VB0, b#200 (*If VB0 is greater than B#200, CR is set to 1, otherwise CR is set to 0 *) ST %Q0.0 (*Assign CR value to Q0.0 *)	
	LD %I0.0 (*Create CR with the value of I0.0 *) GT %VW0, %VW2 (*If CR is 1: If VW0 is greater than VW2, then CR is set to 1, otherwise CR is set to 0 *) (*If CR is 0: GT comparison is not performed, CR remains 0 *) ST %Q0.0 (*Assign CR value to Q0.0 *)	

6.4.2 GE (Greater or Equal)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	GE			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	GE	GE *IN1*, *IN2*	P	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer
IN2	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer

OUT (LD)	Output	BOOL	energy flow

The data types of parameters IN1 and IN2 must be the same.

- LD

If EN is 1, the instruction is executed: if IN1 is greater than or equal to IN2, the output is 1 at OUT, otherwise the output is 0;

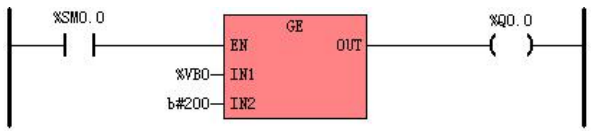
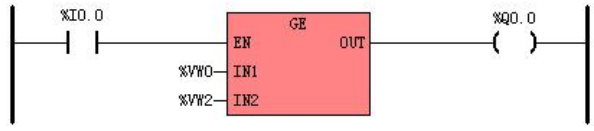
If EN is 0, the instruction is not executed, and the output is 0 at the OUT terminal。

- IL

If the CR value is 1, the instruction is executed: if IN1 is greater than or equal to IN2, the CR value is set to 1, otherwise the CR value is set to 0.

If the CR value is 0, the instruction is not executed, and the CR value remains 0。

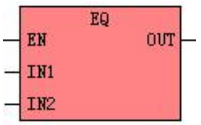
➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Since SM0.0 is fixed at 1, GE always executes: if the value of VB0 is greater than or equal to B#200, Q0.0 is set to 1, otherwise Q0.0 is set to 0.</td></tr>
<tr><td> 
</td><td>If I0.0 is 0: do not execute GE, Q0.0 is set to 0.
If I0.0 is 1: If the value of VW0 is greater than or equal to the value of VW2, then Q0.0 is set to 1, otherwise Q0.0 is set to 0.</td></tr>
</table>

IL	LD %SM0.0 (*Create CR with a value of 1 *) GE %VB0, b#200 (*If VB0 is greater than or equal to B#200, CR is set to 1, otherwise CR is set to 0 *) ST %Q0.0 (* Assign CR value to Q0.0 *)
	LD %I0.0 (* Create CR with the value of I0.0 *) GE %VW0, %VW2 (* If CR is 1: then if VW0 is greater than or equal to VW2, CR is set to 1, Otherwise CR is set to 0 *) (*If CR is 0, GE comparison is not performed and CR remains 0 *) ST %Q0.0 (* Assign CR value to Q0.0 *)

6.4.3 EQ (Equal)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	EQ			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	EQ	EQ IN1, IN2	P	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、Constant、Pointer
IN2	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、Constant、Pointer
OUT (LD)	Output	BOOL	Energy flow

The data types of parameters IN1 and IN2 must be the same.

- LD

If EN is 1, the instruction is executed: if IN1 is equal to IN2, the output is 1 at OUT, otherwise the

output is 0;

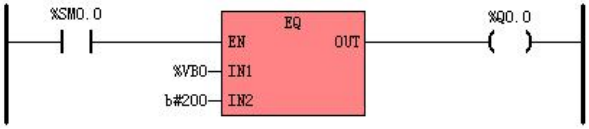
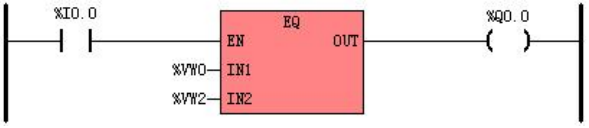
If EN is 0, the instruction is not executed, and the output is 0 at the OUT terminal.

- IL

If the CR value is 1, the instruction is executed: if IN1 is equal to IN2, CR is set to 1, otherwise CR is set to 0;

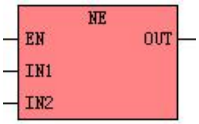
If the CR value is 0, the instruction is not executed and the CR value remains 0.

➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Since SM0.0 is fixed at 1, EQ is always executed: if the value of VB0 is equal to B#200, then Q0.0 is set to 1, otherwise Q0.0 is set to 0.</td></tr>
<tr><td> 
</td><td>If I0.0 is 0: EQ is not performed, and Q0.0 is set to 0.
If I0.0 is 1: If the value of VW0 is equal to the value of VW2, then Q0.0 is set to 1, otherwise Q0.0 is set to 0.</td></tr>
<tr><td rowspan="2">IL</td><td colspan="2">LD %SM0.0 (* Create CR with a value of 1 *)
EQ %VB0, b#200 (* CR is set to 1 if VB0 is equal to B#200, otherwise CR is set to 0*)
ST %Q0.0 (* Assign CR value to Q0.0 *)</td></tr>
<tr><td colspan="2">LD %I0.0 (* Create CR with the value of I0.0 *)
EQ %VW0, %VW2 (* If CR is 1: then if VW0 is equal to VW2, then CR is set to 1, otherwise CR is set to 0 *)
(* If CR is If CR is 0: no EQ comparison is performed, and CR remains 0 *)
ST %Q0.0 (* Assign CR value to Q0.0 *)</td></tr>
</table>

6.4.4 NE (Not equal)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	NE			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	NE	NE IN1, IN2	P	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer
IN2	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer
OUT (LD)	Output	BOOL	Energy Flow

The data types of parameters IN1 and IN2 must be the same.

- LD

If EN is 1, the instruction is executed: if IN1 is not equal to IN2, the output is 1 at OUT, otherwise the output is 0;

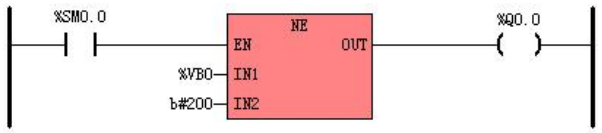
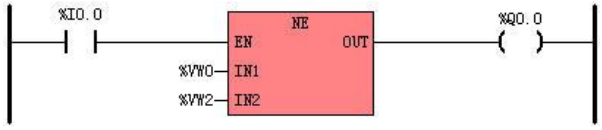
If EN is 0, the instruction is not executed, and the output is 0 at the OUT terminal.

- IL

If the CR value is 1, the instruction is executed: if IN1 is not equal to IN2, set CR to 1, otherwise set CR to 0;

If the CR value is 0, the instruction is not executed, and the CR value remains 0.

➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Since SM0.0 is fixed at 1, NE always executes: if the value of VB0 is not equal to B#200, Q0.0 is set to 1, otherwise Q0.0 is set to 0.</td></tr>
<tr><td> 
</td><td>If I0.0 is 0: NE is not executed, and Q0.0 is set to 0.
If I0.0 is 1: If the value of VW0 is not equal to the value of VW2, then Q0.0 is set to 1, otherwise Q0.0 is set to 0.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* Create CR with a value of 1 *)
NE %VB0, b#200 (*CR is set to 1 if VB0 is not equal to B#200, otherwise CR is set to 0 *)
ST %Q0.0 (* Assign CR value to Q0.0 *)</td></tr>
<tr><td></td><td colspan="2">LD %I0.0 (* Create CR with the value of I0.0 *)
NE %VW0, %VW2 (*If CR is 1: CR is set to 1 if VW0 is not equal to VW2 *)
(* Otherwise CR is set to 0 *)
(* If CR is 0: NE comparison is not performed, CR remains 0 *)
ST %Q0.0 (* Assign CR value to Q0.0 *)</td></tr>
</table>

6.4.5 LT (Less than)

➢ Instruction and its operand description

<table>
<tr><td></td><td>Name</td><td>Instraction format</td><td>Affect CR value</td><td rowspan="3">☑ K5
☑ K2
☑ KS
☑ KW
☑ MK
☑ K6</td></tr>
<tr><td>LD</td><td>LT</td><td>
</td><td></td></tr>
<tr><td>IL</td><td>LT</td><td>LT IN1, IN2</td><td>P</td></tr>
</table>

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer
IN2	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer
OUT (LD)	Output	BOOL	Energy flow

The data types of parameters IN1 and IN2 must be the same.

- LD

If EN is 1, the instruction is executed: if IN1 is less than IN2, the output is 1 at OUT, otherwise the output is 0;

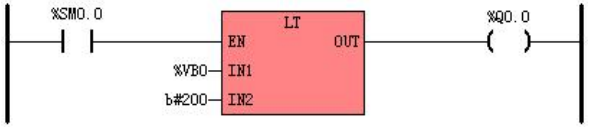
If EN is 0, the instruction is not executed, and the output is 0 at the OUT terminal。

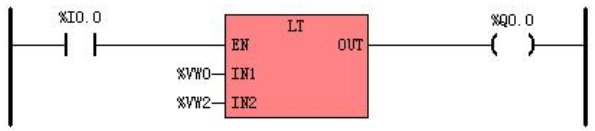
- IL

If the CR value is 1, the instruction is executed: if IN1 is less than IN2, set CR to 1, otherwise set CR to 0;

If the CR value is 0, the instruction is not executed, and the CR value remains 0.

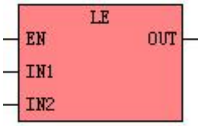
➢ Instruction usage example

LD		Since SM0.0 is fixed at 1, LT is always executed: if the value of VB0 is less than B#200, Q0.0 is set to 1, otherwise Q0.0 is set to 0.

		If I0.0 is 0: LT is not executed, and Q0.0 is set to 0. If I0.0 is 1: If the value of VW0 is less than the value of VW2, then Q0.0 is set to 1, otherwise Q0.0 is set to 0.
IL	LD %SM0.0 (* Create CR with a value of 1 *) LT %VB0, b#200 (*If VB0 is less than B#200, CR is set to 1, otherwise CR is set to 0 *) ST %Q0.0 (* Assign CR value to Q0.0 *)	
	LD %I0.0 (* Create CR with the value of I0.0 *) LT %VW0, %VW2 (* If CR is 1: CR is set to 1 if VW0 is less than VW2 *) (* Otherwise CR is set to 0 *) (*If CR is 0: LT is not executed, CR remains 0 *) ST %Q0.0 (* Assign CR value to Q0.0 *)	

6.4.6 LE (less than or equal to)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
LD	LE			
IL	LE	LE IN1, IN2	P	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer
IN2	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、T、C、HC、constant、pointer

OUT (LD)	Output	BOOL	Energy flow

The data types of parameters IN1 and IN2 must be the same.

- LD

If EN is 1, the instruction is executed: if IN1 is less than or equal to IN2, the output is 1 at OUT, otherwise the output is 0;

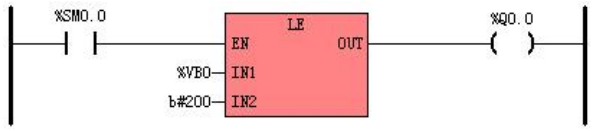
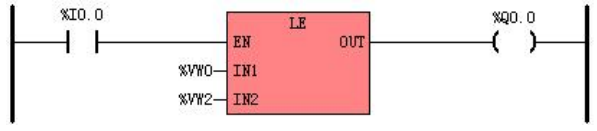
If EN is 0, the instruction is not executed, and the output is 0 at the OUT terminal.

- IL

If the CR value is 1, the instruction is executed: if IN1 is less than or equal to IN2, the CR value is set to 1, otherwise the CR value is set to 0;

If the CR value is 0, the instruction is not executed, and the CR value remains 0.

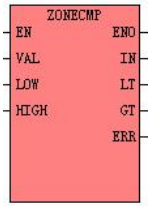
➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>Since SM0.0 is fixed at 1, LE is always executed: if the value of VB0 is less than or equal to B#200, then Q0.0 is set to 1, otherwise Q0.0 is set to 0.</td></tr>
<tr><td> 
</td><td>If I0.0 is 0: LE is not executed, and Q0.0 is set to 0.
If I0.0 is 1: If the value of VW0 is less than or equal to the value of VW2, then Q0.0 is set to 1, otherwise Q0.0 is set to 0.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* Create CR with a value of 1 *)
LE %VB0, b#200 (* If VB0 is less than or equal to B#200, CR is set to 1, otherwise CR is set to 0 *)
ST %Q0.0 (* Assign CR value to Q0.0 *)</td></tr>
</table>

```
LD    %I0.0        (* Create CR with the value of I0.0 *)
LE    %VW0, %VW2   (*If CR is 1: If VW0 is less than or equal to VW2, then CR is set to 1 *)
                        (* Otherwise CR is set to 0 *)
                        (*If CR is 0: LE is not executed, CR remains 0 *)
ST    %Q0.0        (* Assign CR value to Q0.0  *)
```

6.4.7 ZONECMP (Data zone comparison)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	ZONECMP			☑ K2 ☑ KS ☑ MK ☑ K6
IL	ZONECMP	ZONECMP VAL, LOW, HIGH ,IN , LT ,GT, ERR	U	

Parameter	Input/Output	Data type	Allowed memory area
VAL	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、HC
LOW	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、HC、constant
HIGH	Input	BYTE、WORD、DWORD、INT、DINT、REAL	I、Q、M、V、L、SM、AI、AQ、HC、Constant
IN	Output	BOOL	Q、M、V、L、SM
LT	Output	BOOL	Q、M、V、L、SM
GT	Output	BOOL	Q、M、V、L、SM
ERR	Output	BOOL	Q、M、V、L、SM

Note that the data types of parameters VAL, LOW, HIGH must be consistent, and LOW <= HIGH.

This instruction is used to compare VAL with LOW, HIGH and output the result to the corresponding output parameter:

- If LOW > HIGH, ERR output is 1, other parameters output is 0;
- If LOW <= VAL <= HIGH, the output of IN is 1, and the output of other parameters is 0;
- If VAL < LOW, the output of LT is 1, and the output of other parameters is 0;
- If VAL > HIGH, the output of GT is 1, and the output of other parameters is 0.

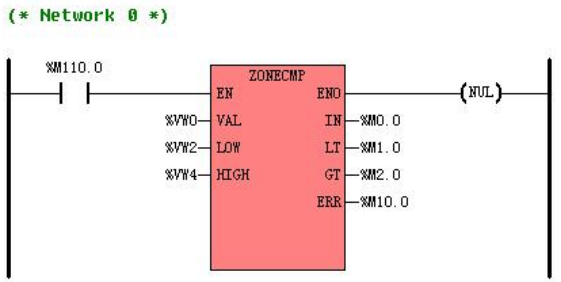
- LD

  If EN is 1, the instruction is executed.

- IL

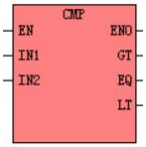
  If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

➢ Instruction usage example

<table>
<tr><td>LD</td><td> 
</td><td>If M110.0 is 0: ZONECMP is not executed, and the output terminal keeps the original state.
If M110.0 is 1: M0.0=TRUE if VW2<=VW0<=VW4;
M1.0=TRUE if VW0<VW2;
M2.0=TRUE if VW0>VW4;</td></tr>
<tr><td>IL</td><td colspan="2">LD %M110.0 (* create CR with the value of M110.0 *)
ZONECMP %VW0, %VW2, %VW4, %M0.0, %M1.0, %M2.0, %M10.0
(*If M110.0 is 0, the instruction is not executed, and the output terminal keeps the original state.
If M110.0 is 1, the instruction is executed: if VW2<=VW0<=VW4, then M0.0=TRUE; if VW0<VW2, then M1.0=TRUE; if VW0>VW4, then M2.0=TRUE *)</td></tr>
</table>

6.4.8 CMP (Comparison)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	CMP			☑ K2 ☑ KS ☑ MK ☑ K6
IL	CMP	CMP IN1, IN2, GT, EQ, LT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD、 INT、DINT、REAL	I、Q、M、V、L、AI、AQ、HC
IN2	Input	BYTE、WORD、DWORD、 INT、DINT、REAL	I、Q、M、V、L、AI、AQ、HC、Constant
GT	Output	BOOL	Q、M、V、L
EQ	Output	BOOL	Q、M、V、L
LT	Output	BOOL	Q、M、V、L

Note that the data types of parameters IN1, IN2, and HIGH must be the same.

This instruction compares IN1 with IN2 and outputs the result to the corresponding output parameter:

- If IN1 is greater than IN2, the output of GT is 1, and the output of other parameters is 0.
- If IN1 is equal to IN2, the output of EQ is 1, and the output of other parameters is 0.
- If VAL is less than LOW, the output of LT is 1, and the output of other parameters is 0;

- LD

  If EN is 1, the instruction is executed.

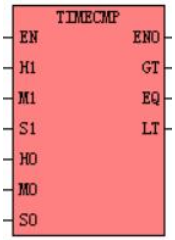
- IL

  If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR

value.

6.4.9 TIMECMP (Time comparison)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	TIMECMP			☑ K2 ☑ KS ☑ MK ☑ K6
IL	TIMECMP	TIMECMP H1,M1,S1,H0,M0,S0,GT,EQ,LT	U	

Parameter	Input/Output	Data type	Allowed memory area
H1	Input	BYTE	M、V、L
M1	Input	BYTE	M、V、L
S1	Input	BYTE	M、V、L
H0	Input	BYTE	M、V、L、Constant
M0	Input	BYTE	M、V、L、Constant
S0	Input	BYTE	M、V、L、Constant
GT	Output	BOOL	Q、M、V、L
EQ	Output	BOOL	Q、M、V、L
LT	Output	BOOL	Q、M、V、L

Note: Parameters H0, M0, and S0 must use variables or constants at the same time.

This instruction combines H1 (hours), M1 (minutes), and S1 (seconds) into a time value (assuming Time1) in 24-hour format. Combine H0 (hours), M0 (minutes), and S0 (seconds) into another time value (assumed to be Time0), then compare Time1 and Time0, and output the result to the corresponding output parameter. **The**

user must input a valid time value, the immediate value range is [0,23], the minute value range is [0,59], and the second value range is [0,59].

- If the value of any input parameter is invalid, the output of all parameters is 0;
- If Time1 is greater than Time0, the output of GT is 1, and the output of other parameters is 0;
- If Time1 is equal to Time0, the output of EQ is 1, and the output of other parameters is 0;
- If Time1 is less than Time0, the LT output is 1, and other output parameters are 0.。

- LD

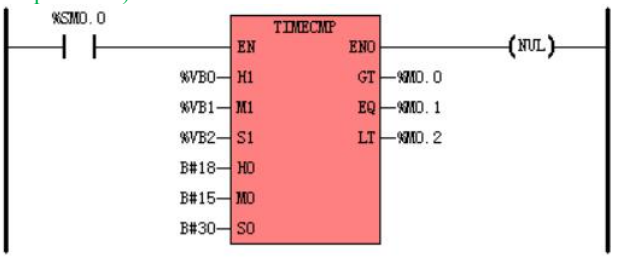
  If EN is 1, the instruction is executed, otherwise it is not executed.

- IL

  If the CR value is 1, the instruction is executed, otherwise it is not executed.

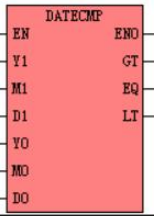
  The execution of this instruction does not affect the CR value。

➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td>(* Network 0 *)
(*The user can use the RTC_R instruction to read the current time value and use it in the following comparison.*)
 
</td></tr>
<tr><td>SM0.0 is always TRUE, so the TIMECMP instruction is always executed: the time value composed of VB0 (hour), VB1 (minute), and VB2 (second) is compared with the time 18:15:30. Assuming VB0=B#8,VB1=B#23,VB2=B#34, then the composition time value is 8:23:34, which is less than 18:15:30, so that the output of M0.2 is 1, the output of M0.0 and M0.1 are 0.</td></tr>
</table>

6.4.10 DATECMP (Data comparison)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	DATECMP			☑ K2 ☑ KS ☑ MK ☑ K6
IL	DATECMP	DATECMP Y1, M1,D1,Y0,M0,D0,GT,EQ,LT	U	

Parameter	Input/Output	Data type	Allowed memory area
Y1	Input	BYTE	M、V、L
M1	Input	BYTE	M、V、L
D1	Input	BYTE	M、V、L
Y0	Input	BYTE	M、V、L、Constant
M0	Input	BYTE	M、V、L、Constant
D0	Input	BYTE	M、V、L、Constant
GT	Output	BOOL	Q、M、V、L
EQ	Output	BOOL	Q、M、V、L
LT	Output	BOOL	Q、M、V、L

Note: Parameters Y0, M0 and D0 must be variables or constants at the same time.

The input parameters Y1 and Y0 both represent the year value based on 2000, that is: the real year value = 2000 + parameter value. For example, suppose the Y1 value is 26, which means 2026; if the Y0 value is 16, it means 2016.

This instruction combines Y1 (year), M1 (month), D1 (day) into a date value (assumed to be Date1). Combine Y0 (year), M0 (month), D0 (day) into another time value (assumed to be Date0), then compare

Date1 and Date0, and output the result to the corresponding output parameter. **The user must enter a valid date value, that is, the value range of the month is [1,12], and the value range of the day is [1,31]. However, this instruction only judges whether each single value is legal, and does not judge whether a complete date is legal. For example, although the date "February 30, 2021" does not exist, this Directive will not judge wrong.**

- If the value of any input parameter is invalid, the output of all parameters is 0;
- If Date1 is greater than Date0, the output of GT is 1, and the output of other parameters is 0;
- If Date1 is equal to Date0, the output of EQ is 1, and the output of other parameters is 0;
- If Date1 is less than Date0, the LT output is 1, and other output parameters are 0.

- LD

  If EN is 1, the instruction is executed, otherwise it is not executed.

- IL

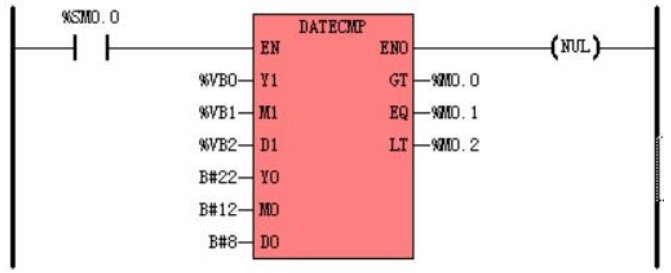
  If the CR value is 1, the instruction is executed, otherwise it is not executed.

  The execution of this instruction does not affect the CR value。

➢ Instruction usage example

LD

```
(* Network 0 *)
(*The user can use the RTC_R instruction to read the current date value and use it in the comparison below. *)
```

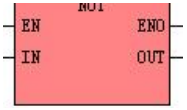


	SM0.0 is always TRUE, so the DATECMP instruction is always executed: the date value composed of VB0 (year), VB1 (month), and VB2 (day) is compared with the date "December 8, 2022". Assuming VB0=B#8,VB1=B#11,VB2=B#21,so that the date is “2008.11.21”, the date is earlier than“2022.12.8”, so the output of M0.2 is 1, the outputs of M0.0 and M0.1 are 0.

6.5 logic operation

6.5.1 NOT (bitwise NOT)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	NOT			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	NOT	NOT *OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、pointer

- LD

  The data types of parameters IN and OUT must be the same.

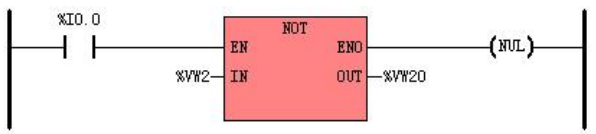
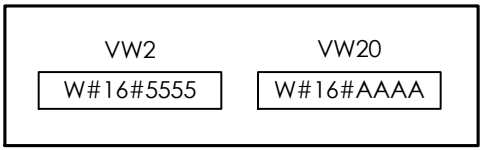
  If EN is 1, the instruction is executed: invert each bit of IN and assign the result to OUT.。

- IL

  If the CR value is 1, the instruction is executed: each bit of OUT is inverted, and the result is still assigned to OUT.

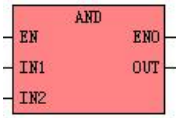
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD		If I0.0 is 0: NOT execute the instruction. If I0.0 is 1: Invert the value of VW2 bit by bit, and assign the result to VW20.
IL	LD %I0.0 (* Create CR with the value of I0.0 *) NOT %VW20 (*If CR is 1: Invert VW20 bitwise, the result is still placed in VW20 *) (* If CR is 0: do not execute NOT instruction *)	
result	Referring to the LD example above, if the NOT instruction is executed, the result is an example as follows: 	

6.5.2 AND (Bitwise AND)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	AND			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	AND	AND IN1, OUT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
IN2	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、pointer

- LD

The data types of parameters IN1, IN2, and OUT must be the same.

If EN is 1, then the instruction is executed: after the "AND" operation is performed on IN1 and IN2 according to binary bits, the result is assigned to OUT.。

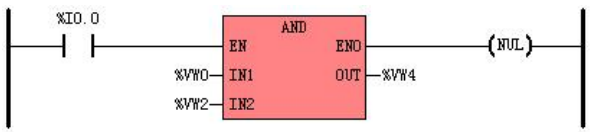
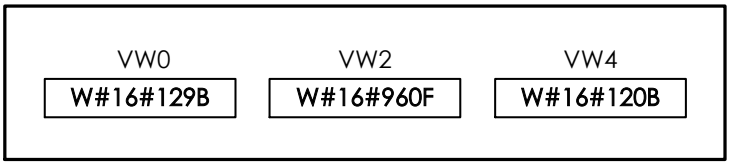
- IL

The data types of parameters IN1 and OUT must be the same.

If the value of CR is 1, the instruction is executed: after performing the "AND" operation between IN1 and OUT according to binary bits, assign the result to OUT.

The execution of this instruction does not affect the CR value。

➢ Instruction usage example

<table>
<tr><td>LD</td><td> 
</td><td>If I0.0 is 0, the "AND" command is not executed.
If I0.0 is 1: "AND" the values of VW0 and VW2 in bits, and assign the result to VW4.</td></tr>
<tr><td>IL</td><td colspan="2">LD %I0.0 (* Create CR with the value of I0.0 *)
AND %VW0, %VW2 (* If CR is 1: the values of VW0 and VW2 are "AND" in phase, and the result is still placed in VW2 *)
(*If CR is 0, the"AND"command is not executed *)</td></tr>
<tr><td>Result</td><td colspan="2">Referring to the LD example above, if the "AND" command is executed, the result is as follows:
 
</td></tr>
</table>

6.5.3 ANDN (Bitwise ANDN)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	☑ K5

LD	ANDN			☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	ANDN	ANDN *IN1, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
IN2	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、pointer

- LD

The data types of parameters IN1, IN2, and OUT must be the same.

If EN is 1, the instruction is executed: AND IN1 and IN2 in binary bits and negate them, and then assign the result to OUT.

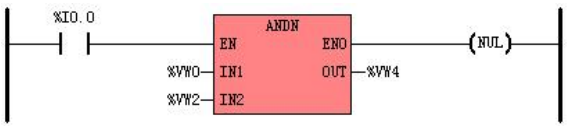
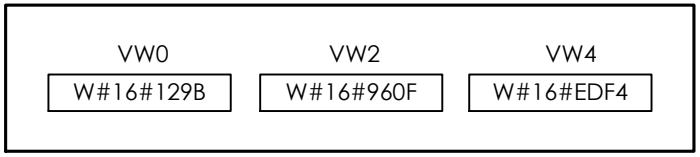
- IL

The data types of parameters IN1 and OUT must be the same.

If the value of CR is 1, the instruction is executed: "AND" IN1 and OUT by binary bits and negate them, and then assign the result to OUT。

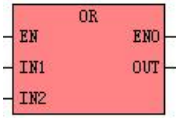
The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD		If I0.0 is 0: do not execute ANDN instruction. If I0.0 is 1: "AND" the values of VW0 and VW2 by bit and negate them, and assign the final result to VW4.
IL	LD %I0.0 (* Create CR with the value of I0.0 *) ANDN %VW0, %VW2 (*If CR is 1: the values of VW0 and VW2 are phased and inverted, and the result is still placed in VW2 *) (*If CR is 0: do not execute ANDN instruction *)	
Result	Referring to the LD example above, if the ANDN instruction is executed, the result is an example as follows 	

6.5.4 OR (Bitwise OR)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	OR			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	OR	OR IN1, OUT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
IN2	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer

OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、Pointer

- LD
  - The data types of parameters IN1, IN2, and OUT must be consistent.
  - If EN is 1, the instruction is executed: After the "OR" operation of IN1 and IN2 in binary bits, the result is assigned to OUT.

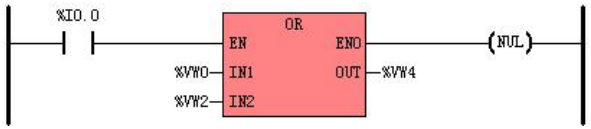
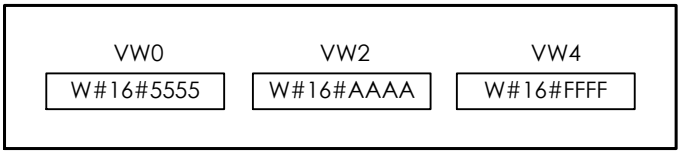
- IL

  The data types of parameters IN1 and OUT must be the same.

  If the value of CR is 1, the instruction is executed: after the "OR" operation of IN1 and OUT is performed according to binary bits, the result is assigned to OUT.

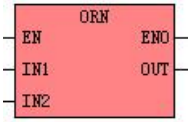
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

<table>
<tr><td>LD</td><td> 
</td><td>If I0.0 is 0: the OR instruction is not executed.
If I0.0 is 1: "OR" the values of VW0 and VW2 by bit, and assign the result to VW4.</td></tr>
<tr><td>IL</td><td colspan="2">LD %I0.0 (* Create CR with the value of I0.0 *)
OR %VW0, %VW2 (*If CR is 1: phase OR the values of VW0 and VW2, the result is still placed in VW2 *)
(*If CR is 0: do not execute OR instruction *)</td></tr>
<tr><td>Result</td><td colspan="2">Referring to the above LD example, if the OR instruction is executed, the result is as follows:
 
</td></tr>
</table>

6.5.5 ORN (Bitwise ORN)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	ORN			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	ORN	ORN I*N1, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
IN2	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、Pointer

- LD

  The data types of parameters IN1, IN2, and OUT must be the same。

  If EN is 1, the instruction is executed: OR IN1 and IN2 by binary bit and negate, and then assign the result to OUT.

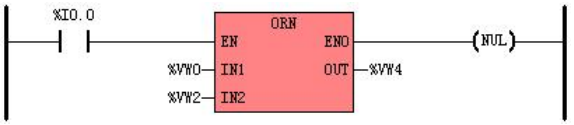
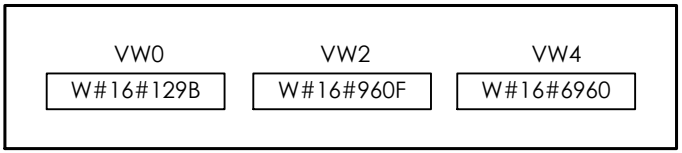
- IL

  The data types of parameters IN and OUT must be the same.

  If the CR value is 1, the instruction is executed: OR IN1 and OUT in binary bits and invert, and assign the result to OUT.

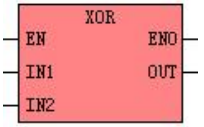
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD		If I0.0 is 0: the ORN instruction is not executed. If I0.0 is 1: "OR" the values of VW0 and VW2 by bit and negate them, and assign the final result to VW4.
IL	LD %I0.0 (* Create CR with the value of I0.0 *) ORN %VW0, %VW2 (*If CR is 1: the values of VW0 and VW2 are phase-wise ORed and inverted, and the result is still placed in VW2*) (*If CR is 0: do not execute ORN instruction *)	
Result	Referring to the LD example above, if the ORN instruction is executed, the result is an example as follows: 	

6.5.6 XOR (Bitwise XOR)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	XOR			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	XOR	XOR IN1, OUT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
IN2	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer

OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、Pointer

- LD

The data types of parameters IN1, IN2, and OUT must be the same.

If EN is 1, then this instruction is executed: After performing the "XOR" operation on IN1 and IN2 according to binary bits, assign the result to OUT.

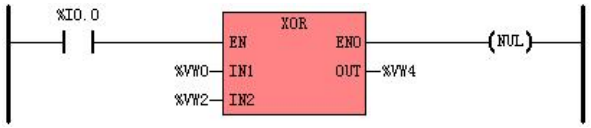
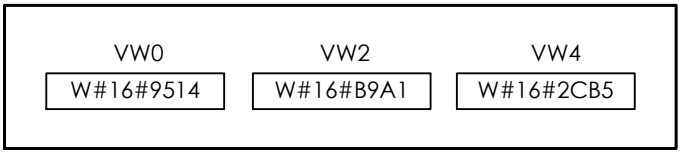
- IL

The data types of parameters IN1 and OUT must be the same.

If the value of CR is 1, the instruction is executed: after performing the "XOR" operation on IN1 and OUT according to binary bits, assign the result to OUT.

The execution of this instruction does not affect the CR value。

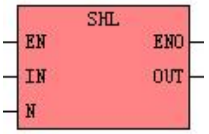
➢ Instruction usage example

LD		If I0.0 is 0: XOR instruction is not executed. If I0.0 is 1: XOR the values of VW0 and VW2 by bit, and assign the result to VW4.
IL	LD %I0.0 (* Create CR with the value of I0.0 *) XOR %VW0, %VW2 (*If CR is 1: XOR the values of VW0 and VW2 by bit, and the result is still placed in VW2 *) (*If CR is 0: do not execute XOR instruction*)	
Result	Referring to the LD example above, if the XOR instruction is executed, the result is an example as follows: 	

6.6 shift instruction

6.6.1 SHL (Shift left)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	SHL			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	SHL	SHL *OUT, N*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
N	Input	BYTE	I、Q、M、V、L、SM、constant、pointer
OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、Pointer

- LD

The data types of parameters IN and OUT must be the same.

If EN is 1, the instruction is executed: move all the binary bits of IN to the left by N bits, the shifted high bits are discarded and the low bits are filled with 0, and the final result is assigned to OUT.

- IL

If the CR value is 1, the instruction is executed: move all the binary bits of OUT to the left by N bits, the shifted high bits are discarded and the low bits are filled with 0, and the final result is still assigned to OUT.

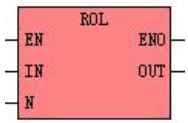
The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD	 On each rising edge of M0.0, a SHL instruction is executed to shift all the binary bits of the QB0 value to the left by 1 bit, and the result is still assigned to QB0.
IL	LD %M0.0 R_TRIG (*Take the rising edge of M0.0 and use the result as the CR value *) SHL %QB0, B#1 (*If CR is 1: Shift all the binary bits of the QB0 value to the left by 1 bit, and the result is still placed in QB0 *) (*If CR is 0: do not execute SHL instruction *)
Result	If the SHL instruction is executed, the result is as follows: QB0 Initial value: B2#10000001 After the 1st shift / After the 2nd shift / After the 3rd shift / After the 4th shift QB0 value: B#2#00000010 / B#2100000100 / B#2100001000 / B#2#00010000

6.6.2 ROL (Rotate left)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	ROL			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	ROL	ROL *OUT, N*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、

			constant、pointer
N	Input	BYTE	I、Q、M、V、L、SM、 constant、pointer
OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、Pointer

- LD

The data types of parameters IN and OUT must be the same.

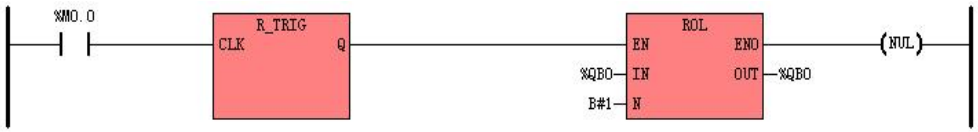
If EN is 1, the instruction is executed: move all the binary bits of IN to the left by N bits, the shifted high bits are shifted into the low bits in turn, and the final result is assigned to OUT.

- IL

If the value of CR is 1, the instruction is executed: move all the binary bits of OUT to the left by N bits, the shifted high bits are shifted into the low bits in turn, and the final result is still assigned to OUT.

The execution of this instruction does not affect the CR value。

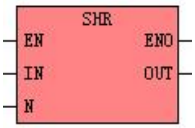
➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td></tr>
<tr><td>On each rising edge of M0.0, the ROL instruction is executed once, and all the binary bits of the QB0 value are shifted to the left by 1 bit at a time, and the shifted high bits are supplemented to the low bits in turn, and the result is still assigned to QB0.</td></tr>
<tr><td>IL</td><td>LD %M0.1
R_TRIG (*Take the rising edge of M0.0 and use the result as the CR value *)
ROL %QB0, B#1 (*If CR is 1: Rotate all the binary bits of the QB0 value to the left by 1 bit, and the result is still placed in QB0 *)
(*If CR is 0: do not execute SHL instruction *)</td></tr>
</table>

Result	If the ROL instruction is executed, the result is as follows: QB0 Initial value: B#2#10100001 After the 1st shift: QB0 value: B#2#01000011 After the 2nd shift: B#2#10000110 After the 3rd shift: B#2#00001101 After the 4th shift: B#2#00011010

6.6.3 SHR (Shift right)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	SHR			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	SHR	SHR *OUT, N*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
N	Input	BYTE	I、Q、M、V、L、SM、constant、pointer
OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、Pointer

- LD

The data types of parameters IN and OUT must be the same.

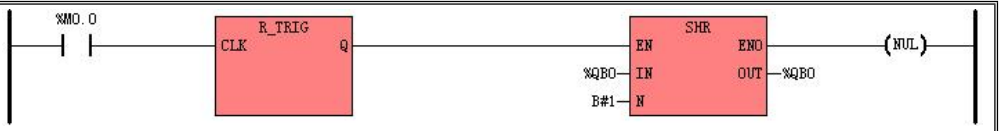
If EN is 1, the instruction is executed: move all the binary bits of IN to the right by N bits, the shifted low bits are discarded and the high bits are filled with 0, and the final result is assigned to OUT.

- IL

If the CR value is 1, the instruction is executed: move all the binary bits of OUT to the right by N bits, the shifted low bits are discarded and the high bits are filled with 0, and the final result is still assigned to OUT.

The execution of this instruction does not affect the CR value.

➢ Instruction usage example

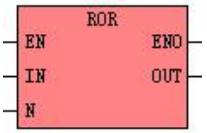
<table>
<tr><td rowspan="2">LD</td><td> 
</td></tr>
<tr><td>On each rising edge of M0.0, the SHR instruction is executed once, all binary bits of the QB0 value are shifted to the right by 1, and the result is still assigned to QB0.</td></tr>
<tr><td>IL</td><td>LD %M0.0
R_TRIG (*Take the rising edge of M0.0 and use the result as the CR value *)
SHR %QB0, B#1 (*If CR is 1: Shift all the binary bits of the QB0 value to the right by 1 bit, and the result is still placed in QB0 *)
(*If CR is 0: do not execute SHL instruction *)</td></tr>
<tr><td>Result</td><td>If the SHR instruction is executed, the result is as follows:

QB0 Initial value: B#2#10000001
After the 1st shift After the 2nd shift After the 3rd shift After the 4th shift
QB0 value: B#2#01000000 B#2#00100000 B#2#00010000 B#2#00001000</td></tr>
</table>

6.6.4 ROR (Rotate right)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5

LD	ROR			☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	ROR	ROR *OUT, N*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE、WORD、DWORD	I、Q、M、V、L、SM、constant、pointer
N	Input	BYTE	I、Q、M、V、L、SM、constant、pointer
OUT	Output	BYTE、WORD、DWORD	Q、M、V、L、SM、Pointer

- LD

The data types of parameters IN and OUT must be the same.

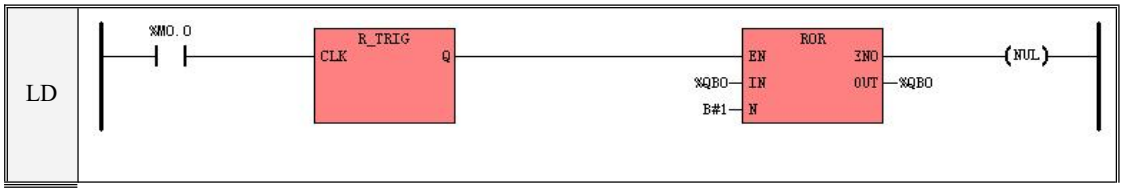
If EN is 1, the instruction is executed: move all the binary bits of IN to the right by N bits, the shifted low bits are shifted into the high bits in turn, and the final result is assigned to OUT.

- IL

If the CR value is 1, the instruction is executed: move all the binary bits of OUT to the right by N bits, the shifted low bits are sequentially moved into the high bits, and the final result is still assigned to OUT.

The execution of this instruction does not affect the CR value.

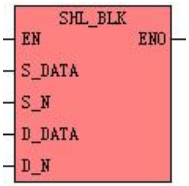
➢ Instruction usage example



	On each rising edge of M0.0, the ROR instruction is executed once, and all the binary bits of the QB0 value are shifted to the right by 1 bit at a time.
IL	LD %M0.1 R_TRIG (*Take the rising edge of M0.0 and use the result as the CR value *) ROR %QB0, B#1 (*If CR is 1: Rotate all the binary bits of the QB0 value by 1 bit to the right, and the result is still placed in QB0 *) (*If CR is 0: do not execute SHL instruction *)
Result	If the ROR instruction is executed, the result is as follows: QB0 Initial value: B#2#10100001 After the 1st shift / After the 2nd shift / After the 3rd shift / After the 4th shift QB0 value: B#2#11010000 / B#2#01101000 / B#2#00110100 / B#2#00011010

6.6.5 SHL_BLK (bit string shift left)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	SHL_BLK			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	SHL_BLK	SHL_BLK S_DATA, S_N, D_DATA, D_N	U	

Parameter	Input/Output	Data type	Allowed memory area
S_DATA	Input	BOOL	I、Q、M、V、L
S_N	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、constant、pointer
D_DATA	Input/Output	BOOL	Q、M、V、L

D_N	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、constant、pointer

The maximum value of S_N, D_N parameters is 1024, when the input is greater than 1024, take 1024. When S_N is greater than D_N, take S_N = D_N. S_N, D_N must be greater than 0.

Note that the S_DATA and D_DATA parameters are variable-length block memory parameters, and the entire block memory cannot fall into the illegal memory area, otherwise the result will be unpredictable.

When the instruction is executed, the consecutive D_N binary bits from D_DATA are shifted to the left by S_N bits, the shifted high bits are discarded, and the consecutive S_N binary bits from S_DATA are added to the rightmost end of D_DATA.

- LD

  If EN is 1, the instruction is executed.。

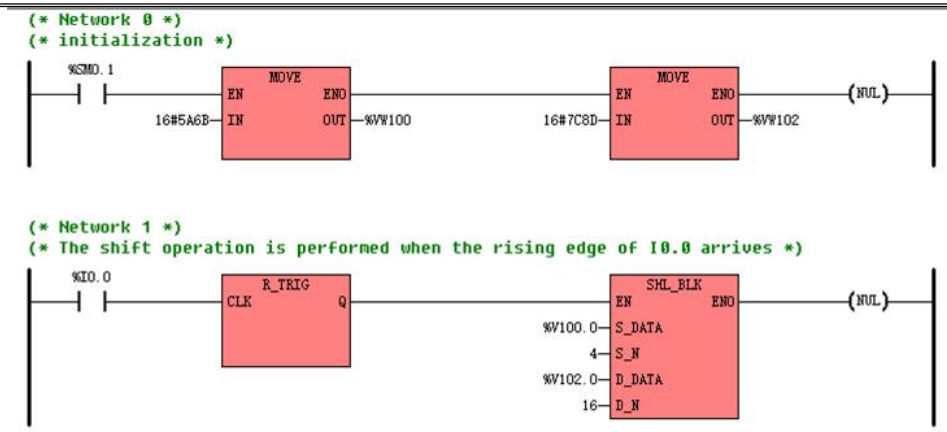
- IL

  If the CR value is 1, the instruction is executed.

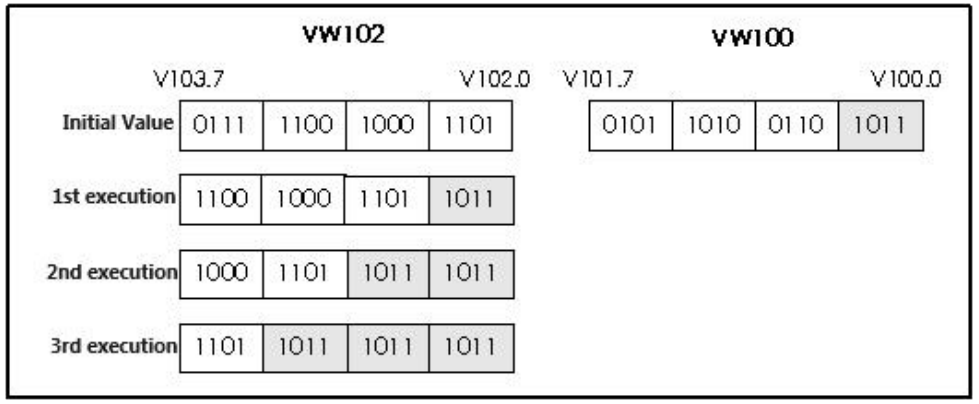
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD

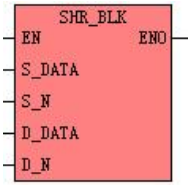


Result



6.6.6 SHR_BLK (bit string shift right)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	SHR_BLK			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	SHR_BLK	SHR_BLK S_DATA, S_N, D_DATA,	U	

		D_N		

Parameter	Input/Output	Data type	Allowed memory area
S_DATA	Input	BOOL	I、Q、M、V、L
S_N	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、constant、pointer
D_DATA	Input/Output	BOOL	Q、M、V、L
D_N	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、constant、pointer

The maximum value of S_N, D_N parameters is 1024, when the input is greater than 1024, take 1024. When S_N is greater than D_N, take S_N = D_N. Both S_N, D_N must be greater than 0.

Note that the S_DATA and D_DATA parameters are variable-length block memory parameters, and the entire block memory cannot fall into the illegal memory area, otherwise the result will be unpredictable.

When this instruction is executed, the consecutive D_N binary bits from D_DATA are shifted to the right by S_N bits, the shifted low bits are discarded, and the consecutive S_N binary bits from S_DATA are added to the leftmost end of D_DATA.

- LD

  If EN is 1, the instruction is executed.

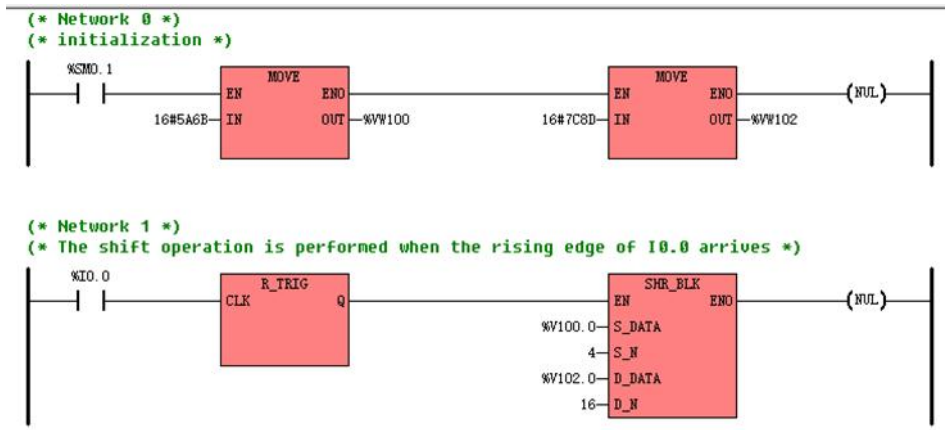
- IL

  If the CR value is 1, the instruction is executed.

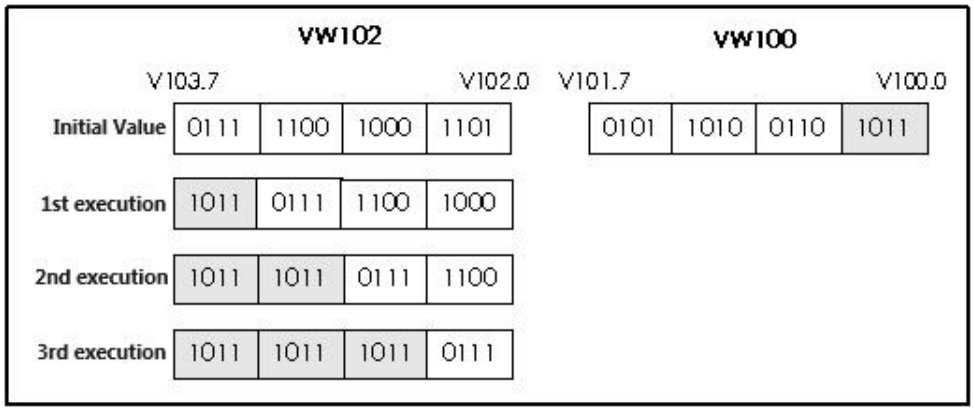
  The execution of this instruction does not affect the CR value.

- Instruction usage example

LD



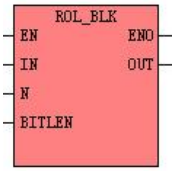
Result



6.6.7 ROL_BLK (bit string rotate left)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5

LD	ROL_BLK			☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	ROL_BLK	ROL_BLK *IN, N, BITLEN ,OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BOOL	I、Q、M、V、L、SM
N	Input	WORD	I、Q、M、V、L、SM、constant、pointer
BITLEN	Input	WORD	I、Q、M、V、L、SM、constant、pointer
OUT	Output	BOOL	Q、M、V、L、SM

- LD

The data types of parameters IN and OUT must be the same.

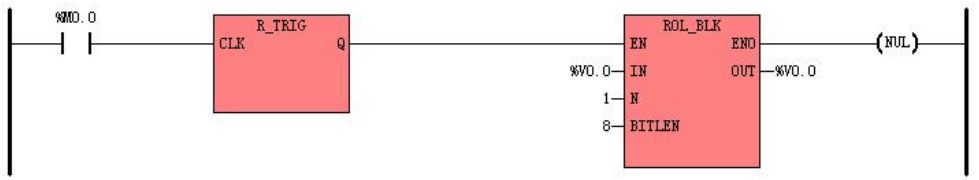
If EN is 1, the instruction is executed: the BITLEN bits starting from IN are shifted to the left by N bits, the shifted high bits are sequentially moved into the low-order positions, and the final result is assigned to the BITLEN bits starting from OUT in turn.

- IL

If the CR value is 1, the instruction is executed: the BITLEN bits starting from IN are shifted to the left by N bits, the shifted high bits are sequentially moved into the low-order positions, and the final result is assigned to the BITLEN bits starting from OUT in turn.

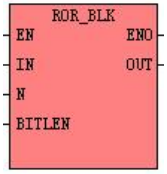
The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD	
	On each rising edge of M0.0, the ROL_BLK instruction is executed once: the 8-bit (ie V0.0 to V0.7) data from V0.0 is shifted to the left by 1 bit, and the shifted high bits are sequentially filled to the low bits position and then assign the result to the 8 bits starting at V0.0 (ie V0.0 to V0.7).
Result	Assume the initial value of 8 bit data starting from V0.0: 2#10010001 After first shift after second shift after third shift Result 8-bit data value starting at V0.0: 2#00100011 2#01000110 2#10001100

6.6.8 ROR_BLK (bit string rotate right)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	ROR_BLK			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	ROR_BLK	ROR_BLK *IN, N, BITLEN ,OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BOOL	I、Q、M、V、L、SM
N	Input	WORD	I、Q、M、V、L、SM、constant、pointer
BITLEN	Input	WORD	I、Q、M、V、L、SM、constant、

			pointer
OUT	Output	BOOL	Q、M、V、L、SM

- LD

The data types of parameters IN and OUT must be the same.

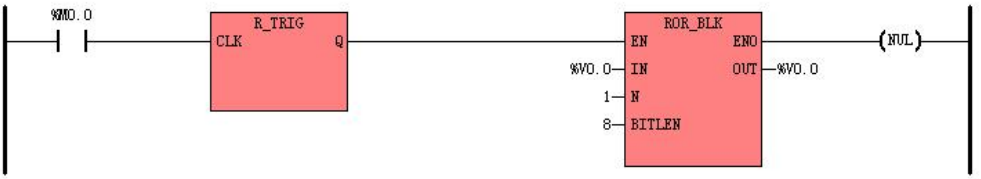
If EN is 1, the instruction is executed: the BITLEN bits starting from IN are shifted to the right by N bits, the shifted low bits are sequentially moved into the high-order position, and the final result is assigned to the BITLEN bits starting from OUT in turn.

- IL

If the CR value is 1, the instruction is executed: the BITLEN bits starting from IN are shifted to the right by N bits, the shifted low bits are sequentially moved into the high-order position, and the final result is assigned to the BITLEN bits starting from OUT in turn.

The execution of this instruction does not affect the CR value.

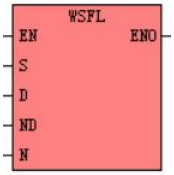
➢ Instruction usage example

LD	
	On each rising edge of M0.0, the ROR_BLK instruction is executed once: The 8-bit data (ie V0.0 to V0.7) starting from V0.0 is shifted to the right by 1 bit, and the shifted low bits are sequentially filled to the high-order position, and then the result is still assigned to the 8 bits (ie V0.0 to V0.7) starting from V0.0.

Result	Assume the initial value of 8 bit data starting from V0.0: 2#10010001 After first shift after second shift after third shift Result 8-bit data value starting at V0.0: 2#11001000 2#01100100 2#10110010

6.6.9 WSFL (Words shift left)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	WSFL			☑ K2 ☑ KS ☑ MK ☑ K6
IL	WSFL	WSFL *S, D, ND,N*	U	

Parameter	Input/Output	Data type	Allowed memory area
S	Input	WORD、INT	V、M、L
D	Input/Output	WORD、INT	V、M、L
ND	Input	INT	V、M、L、constant
N	Input	INT	V、M、L、constant

The data types of S and D must be the same; ND and N must both be constants or variables.

The values of parameters ND and N must satisfy the following conditions: 0 < N <= ND <= 512.

This instruction is used to move N consecutive ND words starting from D and N consecutive words starting from S to the left (ie high address direction) N times (1 bit at a time), and the words shifted from the high address will be discarded.。 The consecutive ND words with D as the starting address are shifted to the left N times, the N words of the high address will be shifted out and discarded, and the N words of the low address will be supplemented with the consecutive N words of S as the starting address. **Note: All memory variables involved in the move must be in a legal address range!**

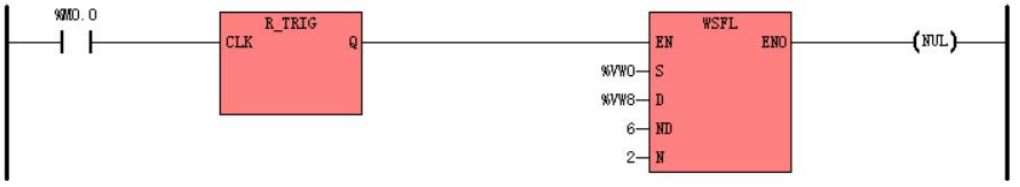
- LD

  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

➢ Instruction usage example

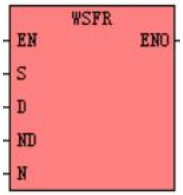
LD	 When the rising edge of M0.0 is detected, WSFL is executed once: Starting from VW8, 6 consecutive words are shifted to the left (high address direction) twice, the values of the highest 2 words (VW18 and VW16) will be discarded, and the values of the lowest 2 words (VW8 and VW10) will be replaced with the values of VW0 and VW2 respectively.

An example of the execution result is as follows. The numbers in the table represent the values in the corresponding memory addresses. The result of the first execution of the instruction uses different colors to illustrate the source of the value.

Instruction execution times	D						S	
	VW18	VW16	VW14	VW12	VW10	VW8	VW2	VW0
Initial value	18	16	14	12	10	8	2	1
First	14	12	10	8	2	1	2	1
Second	10	8	2	1	2	1	2	1

6.6.10 WSFR (Words shift right)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	WSFR			☑ K2 ☑ KS ☑ MK ☑ K6
IL	WSFR	WSFR *S, D, ND,N*	U	

Parameter	Input/Output	Data type	Allowed memory area
S	Input	WORD、INT	V、M、L
D	Input/Output	WORD、INT	V、M、L
ND	Input	INT	V、M、L、constant
N	Input	INT	V、M、L、constant

The data types of S and D must be the same; ND and N must both be constants or variables.

The values of parameters ND and N must satisfy the following conditions: 0 < N <= ND <= 512.

This instruction is used to move N consecutive ND words starting from D and N consecutive words starting from S to the right (ie high address direction) N times (1 bit at a time), and the words shifted from the high address will be discarded.。The consecutive ND words with D as the starting address are shifted to the right N times, the N words of the high address will be shifted out and discarded, and the N words of the low address will be supplemented with the consecutive N words of S as the starting address. **Note: All memory variables involved in the move must be in a legal address range!**

- LD

  If EN is 1, the instruction is executed.

- IL

If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD	
	When the rising edge of M0.0 is detected, WSFL is executed once: Starting from VW8, 6 consecutive words are shifted to the right (high address direction) twice, the values of the highest 2 words (VW18 and VW16) will be discarded, and the values of the lowest 2 words (VW8 and VW10) will be replaced with the values of VW0 and VW2 respectively.

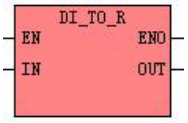
An example of the execution result is as follows. The numbers in the table represent the values in the corresponding memory addresses. The result of the first execution of the instruction uses different colors to illustrate the source of the value.

Instruction execution times	D						S	
	VW18	VW16	VW14	VW12	VW10	VW8	VW2	VW0
Initial value	18	16	14	12	10	8	2	1
First	2	1	18	16	14	12	2	1
Second	2	1	2	1	18	16	2	1

6.7 type conversion

6.7.1 DI_TO_R (Double integer to real)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	DI_TO_R			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	DI_TO_R	DI_TO_R *IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	DINT	I、Q、M、V、L、SM、HC、constant
OUT	Output	REAL	V、L

This instruction converts the input double integer IN to a real number and assigns it to OUT.。

- LD

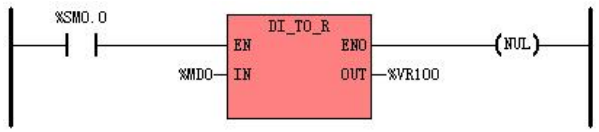
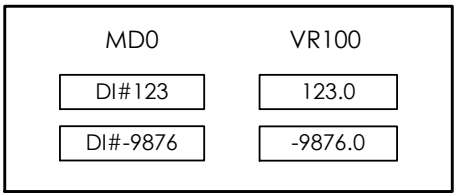
  If EN is 1, the instruction is executed。

- IL

  If the CR value is 1, the instruction is executed.

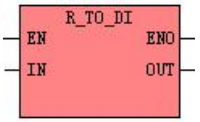
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

<table>
<tr><td>LD</td><td> 
</td><td>SM0.0 is always 1, so the DI_TO_R instruction is always executed: convert the value of MD0 to a real number and assign it to VR100.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0 (* Create CR with a value of 1 *)
DI_TO_R %MD0, %VR100 (*Convert the value of MD0 to a real number and assign it to VR100 *)</td></tr>
<tr><td>result</td><td colspan="2">Examples of results are as follows:
 
</td></tr>
</table>

6.7.2 R_TO_DI (Real to Double Integer)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
LD	R_TO_DI			
IL	R_TO_DI	R_TO_DI *IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	REAL	V、L 、constant
OUT	Output	DINT	M、V、L、SM

This instruction converts the input real number IN to a double integer (the fractional part is rounded off) and assigns it to OUT.

- LD

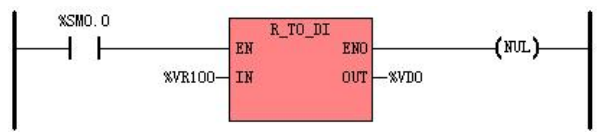
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

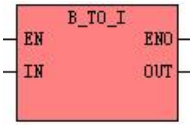
- Instruction usage example

LD		SM0.0 is always 1, so the R_TO_DI instruction is always executed: convert the real number VR100 to a double integer and assign it to VD0.

IL	LD %SM0.0 (* Create CR with a value of 1 *) R_TO_DI %VR100, %VD0 (*Convert the real number VR100 to a double integer and assign it to VD0 *)
Result	Examples of results are as follows: VR100 VD0 123.4 DI#123 5213.6 DI#5214

6.7.3 B_TO_I (Byte to Integer)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	B_TO_I			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	B_TO_I	B_TO_I *IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE	I、Q、M、V、L、SM、constant
OUT	Output	INT	Q、M、V、L、SM、AQ

This instruction converts the input byte data IN to an integer and assigns it to OUT.

- LD

If EN is 1, the instruction is executed.

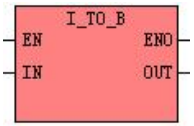
- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

6.7.4 I_TO_B (Integer to byte)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	I_TO_B			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	I_TO_B	I_TO_B*IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	INT	I、Q、M、V、L、SM、AI、AQ、T、C、constant
OUT	Output	BYTE	Q、M、V、L、SM

This instruction converts the integer IN to a byte value and assigns it to OUT.

The value range of the byte type is [0,255]. If the value of parameter IN exceeds this range, the conversion result will be the low byte value, and the PLC will record an overflow error.

- LD

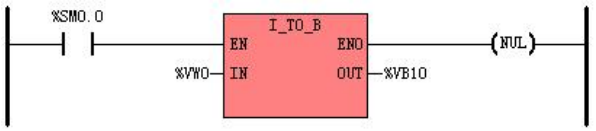
  If EN is 1, the instruction is executed。

- IL

  If the CR value is 1, the instruction is executed.

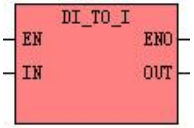
The execution of this instruction does not affect the CR value.

- Instruction usage example

LD		SM0.0 is always 1, so the I_TO_B instruction is always executed: assign the low byte of VW0 to VB10.
IL	LD %SM0.0 (* Create CR with a value of 1 *) I_TO_B %VW0, %VB10 (*Convert VW0 integer value to byte value and assign to VB10 *)	
Result	Examples of results are as follows: VW0 / VB10 24 / B#24 255 / B#255 I#16#FFFD / B#16#FD	

6.7.5 DI_TO_I (Double integer to integer)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	DI_TO_I			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	DI_TO_I	DI_TO_I *IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	DINT	I、Q、M、V、L、SM、HC、constant
OUT	Output	INT	Q、M、V、L、SM、AQ

This instruction converts the double integer IN to an integer and assigns it to OUT.

The value range of the integer is [-32768,32767]. If the value of parameter IN exceeds this range, the conversion result will be the low word value, and the PLC will record an overflow error.

- LD

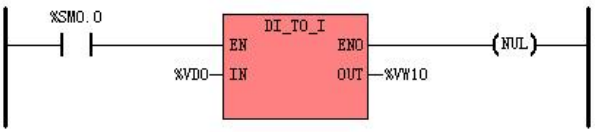
  If EN is 1, the instruction is executed.

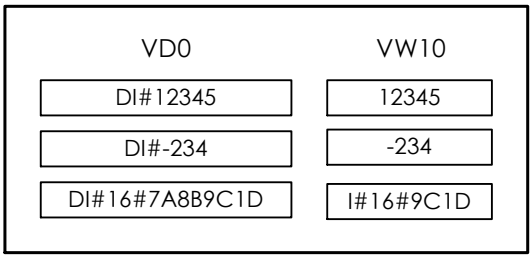
- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

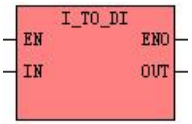
➢ Instruction usage example

LD		SM0.0 is always 1, so the DI_TO_I instruction is always executed: assign the low word of VD0 to VW10.
IL	LD %SM0.0 (* Create CR with a value of 1 *) DI_TO_I %VD0, %VW10 (*Convert VD0 double integer value to integer and assign to VW10 *)	

Result	Examples of results are as follows: 

6.7.6 I_TO_DI (Integer to Double Integer)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
LD	I_TO_DI			
IL	I_TO_DI	I_TO_DI *IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	INT	I、Q、M、V、L、SM、AI、AQ、T、C、constant
OUT	Output	DINT	Q、M、V、L、SM

This instruction converts the input integer IN to a double integer and assigns it to OUT.

- LD

  If EN is 1, the instruction is executed.

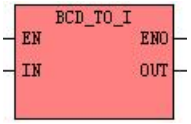
- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

6.7.7 BCD_TO_I (BCD code to Integer)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
LD	BCD_TO_I			
IL	BCD _TO_I	BCD _TO_I *IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	WORD	I、Q、M、V、L、SM、constant
OUT	Output	INT	Q、M、V、L、SM、AQ

This instruction converts the input BCD code IN to an integer and assigns it to OUT.

Note: BCD code adopts 8421 code. The valid range for IN is 0∽9999 BCD.

- LD

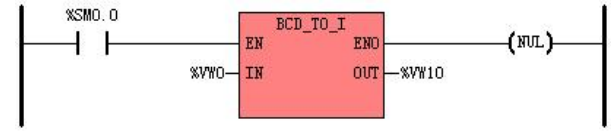
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

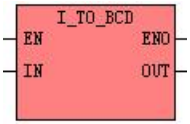
➢ Instruction usage example

LD		SM0.0 is always 1, so the BCD_TO_I instruction is always executed: convert the value of VW0 from BCD to an integer and assign it to VW10.

IL	LD %SM0.0 (* Create CR with a value of 1 *) BCD_TO_I %VW0, %VW10 (*Convert the value of VW0 from BCD to integer and assign it to VW10 *)
Result	Examples of results are as follows: VW0: 16#99 → VW10: 99 VW0: 16#4567 → VW10: 4567 VW0: 16#9999 → VW10: 9999

6.7.8 I_TO_BCD (Integer to BCD code)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	I_TO_BCD			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	I_TO_BCD	I _TO_BCD*IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	INT	I、Q、M、V、L、SM、AI、AQ、T、C、constant
OUT	Output	WORD	Q、M、V、L、SM

This instruction converts the input integer IN into BCD code and assigns it to OUT.

Note: The BCD code adopts 8421 code, so the valid range of IN is 0∽9999. If it exceeds this range, the PLC will record an overflow error and handle it as follows: If IN ≤ 0, the conversion result remains 0, and if IN ≥ 9999, the conversion result remains 9999 BCD.

- LD

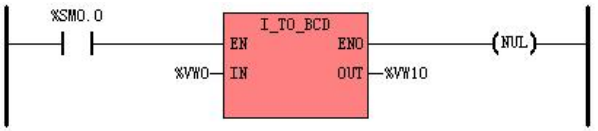
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

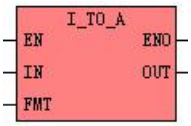
➢ Instruction usage example

LD		SM0.0 is always 1, so the I_TO_BCD instruction is always executed: convert the value of VW0 from an integer to BCD and assign it to VW10.
IL	LD %SM0.0 (* Create CR with a value of 1 *) I_TO_BCD %VW0, %VW10 (*Convert the value of VW0 from integer to BCD and assign it to VW10 *)	
Result	Examples of results are as follows: VW0 / VW10 99 / 16#99 4567 / 16#4567 9999 / 16#9999	

6.7.9 I_TO_A (Integer to ASCII)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5

LD	I_TO_A			☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	I_TO_A	I_TO_AIN, OUT, FMT	U	

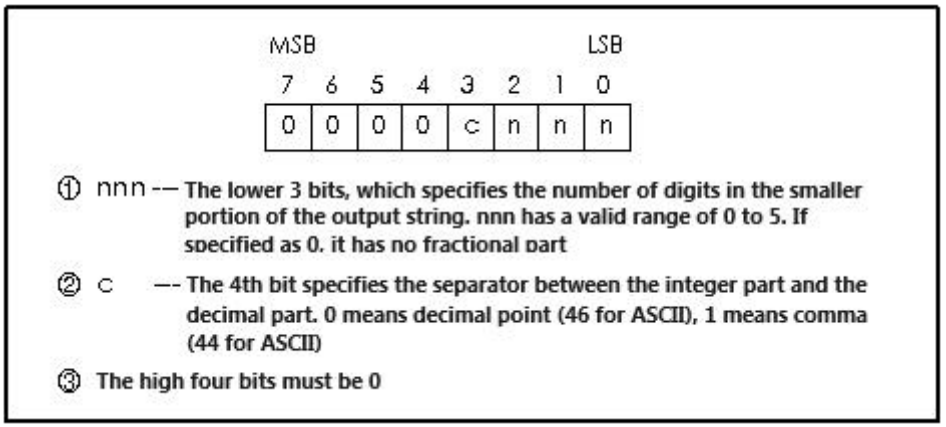
Parameter	Input/Output	Data type	Allowed memory area
IN	Input	INT	I、Q、M、V、L、SM、AI、AQ、T、C、constant
FMT	Input	BYTE	I、Q、M、V、L、SM
OUT	Output	BYTE	Q、M、V、L、SM

Note: The OUT parameter is the starting address of a variable-length memory block. The user needs to ensure that all the memory blocks are in the legal address range, otherwise the result is unpredictable.

This instruction converts the input integer IN into an ASCII string, and formats the conversion result into the output buffer. Positive numbers are converted without a sign, and negative numbers are converted with a negative sign.

The parameter OUT defines the starting address of the output buffer, which occupies a continuous 8-byte memory space. Strings are right-aligned in the buffer, and unused addresses in the buffer are assigned spaces (ASCII value 32).

The parameter FMT defines the representation of the output string, which is composed as shown below:



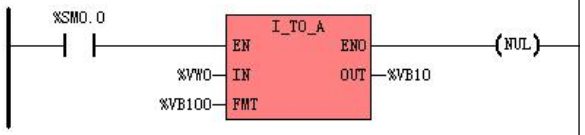
- LD

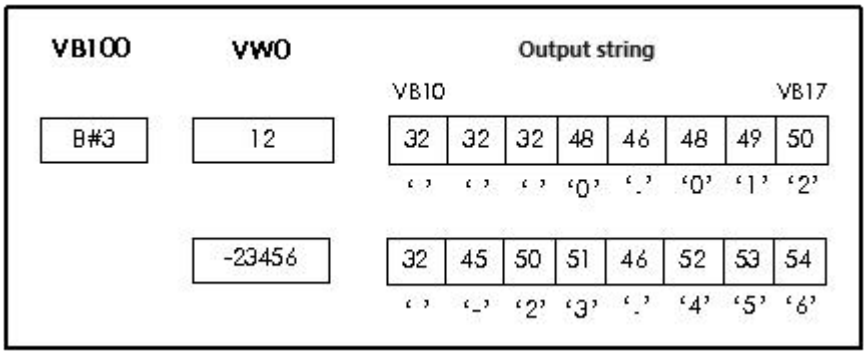
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

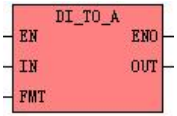
➢ Instruction usage example

LD		SM0.0 is always 1, so the I_TO_A instruction is always executed: the value of VW0 is converted into a string and formatted and output to 8 consecutive bytes starting from VB10.
IL	LD %SM0.0 I_TO_A %VW0, %VB10, %VB100	

Result	Examples of results are as follows: 

6.7.10 DI_TO_A (Double integer to ASCII)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	DI_TO_A	 		☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	DI_TO_A	DI_TO_AIN, OUT, FMT	U	

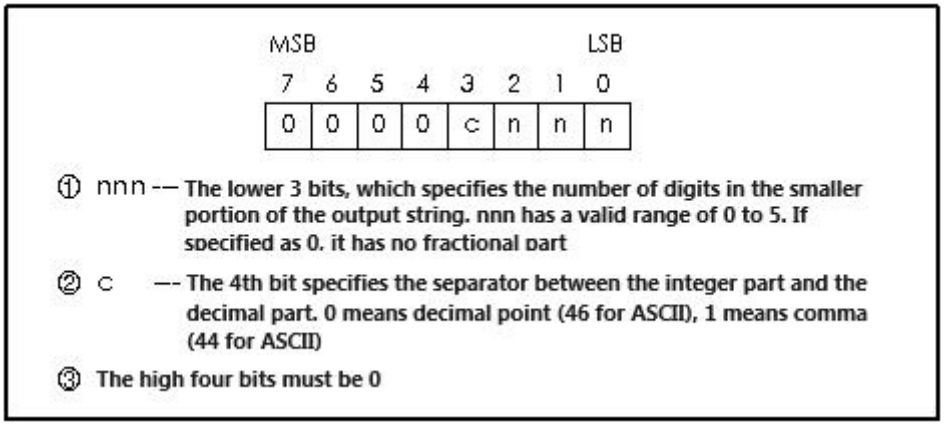
Parameter	Input/Output	Data type	Allowed memory area
IN	Input	DINT	I、Q、M、V、L、SM、HC、constant
FMT	Input	BYTE	I、Q、M、V、L、SM
OUT	Output	BYTE	Q、M、V、L、SM

Note: The OUT parameter is the starting address of a variable-length memory block. The user needs to ensure that all the memory blocks are in the legal address range, otherwise the result is unpredictable.

This instruction converts the input double integer IN into an ASCII string and formats the conversion result into the output buffer. Positive numbers are converted without a negative sign, and negative numbers are converted with a negative sign..

The parameter OUT defines the starting address of the output buffer, which occupies a continuous 12-byte memory space. Strings are right-aligned in the buffer, and unused addresses in the buffer are assigned the space character (ASCII 32).

The parameter FMT defines the representation of the output string, and its composition is as follows:



- LD

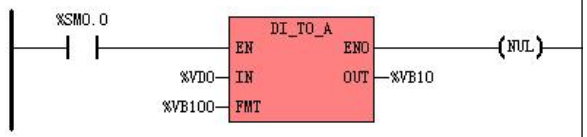
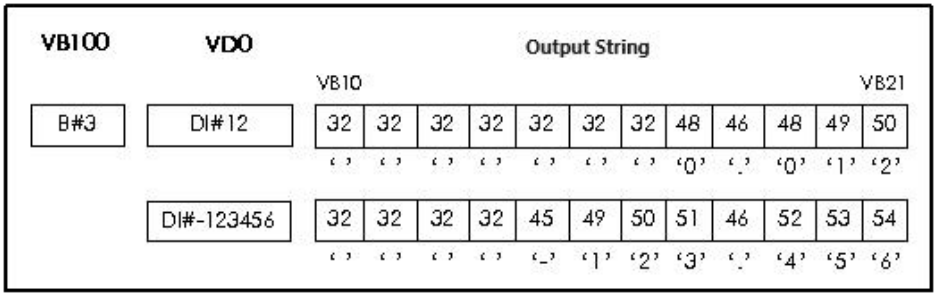
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

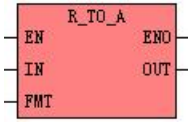
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD		SM0.0 is always 1, so the DI_TO_A instruction always executes: convert the value of VD0 to a string and format the output into 12 consecutive bytes at the beginning of VB10.
IL	LD %SM0.0 DI_TO_A %VD0, %VB10, %VB100	
Result	Examples of results are as follows: 	

6.7.11 R_TO_A (Real to ASCII)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	R_TO_A			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	R_TO_A	R_TO_AIN, OUT, FMT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	REAL	V、L、constant

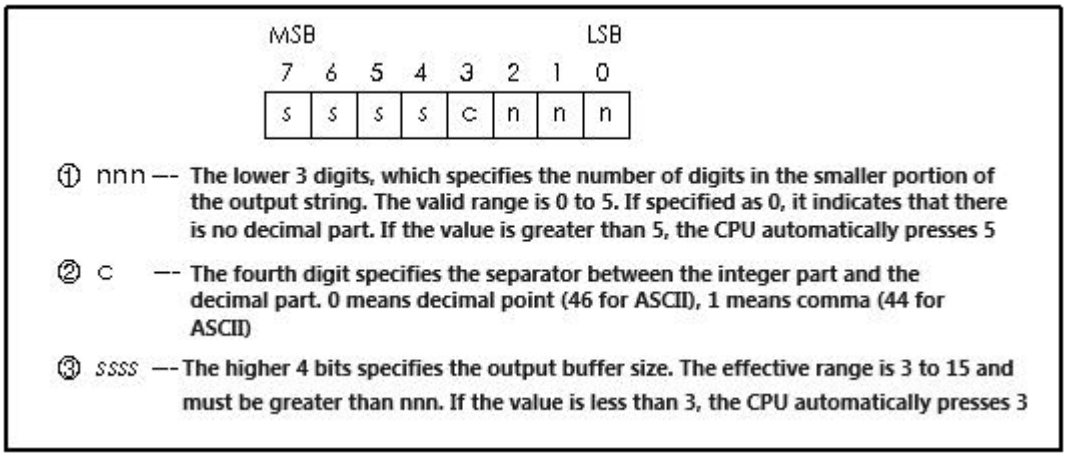
FMT	Input	BYTE	I、Q、M、V、L、SM
OUT	Output	BYTE	Q、M、V、L、SM

Note: The OUT parameter is the starting address of a variable-length memory block. The user needs to ensure that all the memory blocks are in the legal address range, otherwise the result is unpredictable.

This instruction converts the input real number IN into an ASCII string and formats the conversion result into the output buffer. Positive numbers are converted without a sign, and negative numbers are converted with a negative sign. If the number of decimal places of IN is greater than the number of decimal places specified by nnn in the parameter FMT, the IN will be rounded first before conversion, and if it is less than the number of decimal places of IN will be filled with 0.

The parameter OUT defines the starting address of the output buffer, the size of which is specified in FMT. Strings are right-aligned in the buffer, and unused addresses in the buffer are assigned spaces (ASCII 32).

The parameter FMT defines the representation of the output string, which is composed as shown below:



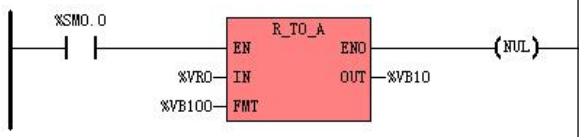
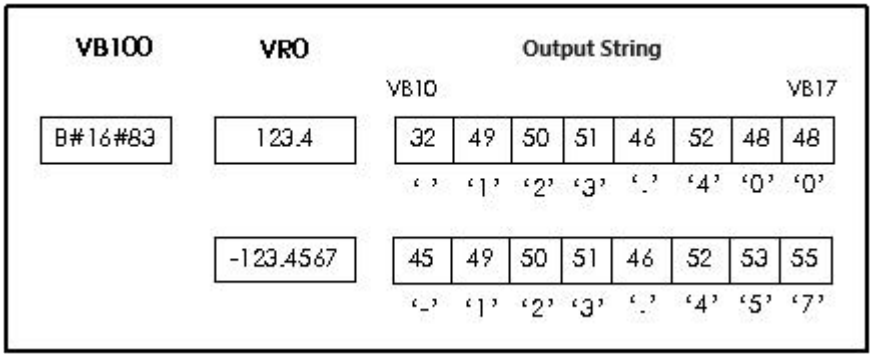
Note when using this instruction: the allowable range of input IN is [-2147480000.0, 4294960000.0]. If IN exceeds this range or the length of the conversion result exceeds the length of the output buffer, the output buffer will be filled with spaces (ASCII value is 32).

- LD
  If EN is 1, the instruction is executed.

- IL

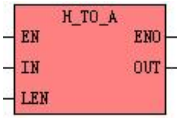
  If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

➢ Instruction usage example

<table>
<tr><td>LD</td><td> 
</td><td>SM0.0 is always 1, so the R_TO_A instruction is always executed: convert the value of VR0 to a string and format it and output it to the buffer starting from VB10.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0
R_TO_A %VR0, %VB10, %VB100</td></tr>
<tr><td>Result</td><td colspan="2">Examples of results are as follows:
 
</td></tr>
</table>

6.7.12 H_TO_A (Hexadecimal to ASCII)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	H_TO_A			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	H_TO_A	H_TO_AIN, OUT, LEN	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE	I、Q、M、V、L、SM
LEN	Input	BYTE	I、Q、M、V、L、SM、constant
OUT	Output	BYTE	Q、M、V、L、SM

Note: The IN and OUT parameters are both expressed as the starting address of a variable-length memory block. The user needs to ensure that all the memory blocks are in the legal address range, otherwise the result is unpredictable.

This instruction converts the hexadecimal number of consecutive LEN bytes starting at address IN into an ASCII string and outputs it to the output buffer starting at address OUT. **Note: Since every 4 binary bits represents a hexadecimal number, each input byte contains 2 hexadecimal numbers, and the output buffer occupies a total of LEN×2 bytes of space**.

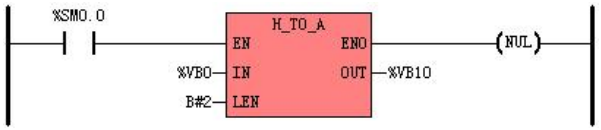
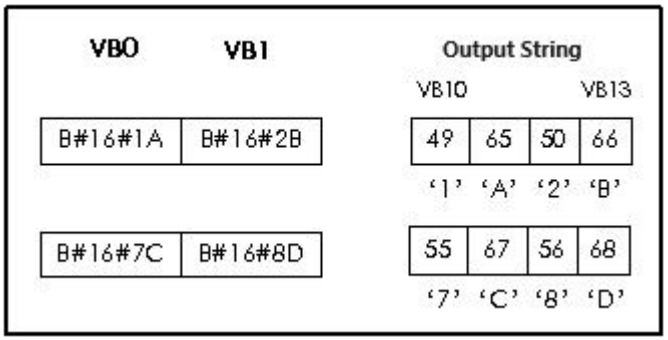
- LD

  If EN is 1, the instruction is executed.

- IL

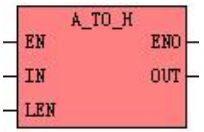
  If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD		SM0.0 is always 1, so the H_TO_A instruction is always executed: convert the hexadecimal number of 2 consecutive bytes starting from VB0 into a string and output it to 4 consecutive bytes starting from VB10.
IL	LD %SM0.0 H_TO_A %VB0, %VB10, B#2	
Result	Examples of results are as follows: 	

6.7.13 A_TO_H (ASCII to hexadecimal)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	A_TO_H			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	A_TO_H	A_TO_HIN, OUT, LEN	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE	I、Q、M、V、L、SM

LEN	Input	BYTE	I、Q、M、V、L、SM、constant
OUT	Output	BYTE	Q、M、V、L、SM

Note: The IN and OUT parameters are both expressed as the starting address of a variable-length memory block. The user needs to ensure that all the memory blocks are in the legal address range, otherwise the result is unpredictable.

This instruction converts consecutive LEN ASCII characters starting at address IN into hexadecimal numbers and outputs them to the output buffer starting at address OUT. **Note: Since a hexadecimal number is represented by 4 binary bits, each input byte (representing an ASCII character) after conversion occupies 4 binary bits (half byte) of space in the output buffer.**

The valid ASCII input range is: B#16#30∽B#16#39 (0∽9), B#16#41∽B#16#46 (A∽F), B#16#61∽B#16#66 (a∽f)。

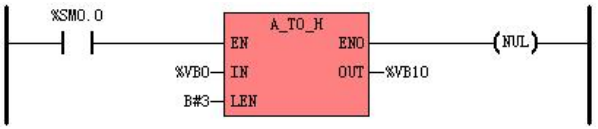
- LD

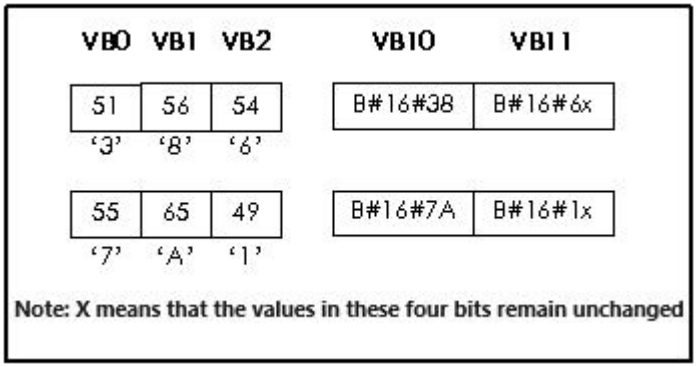
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.The execution of this instruction does not affect the CR value.

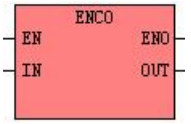
➢ Instruction usage example

LD		SM0.0 is always 1, so the A_TO_H instruction is always executed: convert the ASCII characters of 3 consecutive bytes starting from VB0 into hexadecimal numbers and output them to the output buffer starting from VB100.
IL	LD %SM0.0 A_TO_H%VB0, %VB10, B#3	

Result	Examples of results are as follows: 

6.7.14 ENCO (Code)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	ENCO	 		☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	ENCO	ENCOIN, OUT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	WORD	I、Q、M、V、L、SM、constant
OUT	Output	BYTE	Q、M、V、L、SM

This instruction starts counting from the lowest bit of the input word IN, and writes the bit number of the first bit that is 1 into the output byte OUT. Note: If the value of IN is equal to 0, the calculation result is meaningless.

- LD

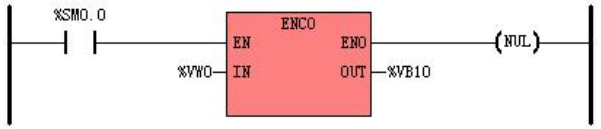
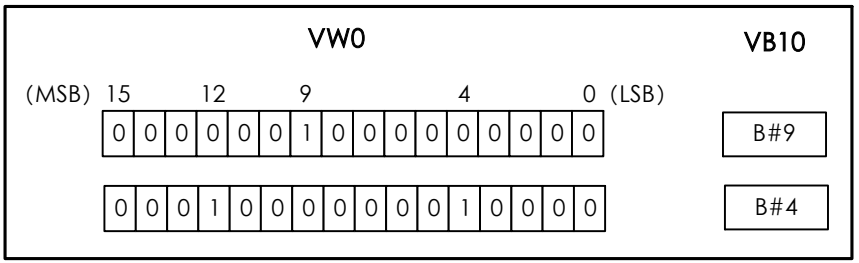
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

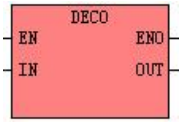
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD		SM0.0 is always 1, so the ENCO instruction is always executed: start from the lowest bit in VW0, and write the bit number of the first bit equal to 1 into VB10.
IL	LD %SM0.0 ENCO %VW0, %VB10	
Result	Examples of results are as follows , The value of VW0 in the figure is expressed in binary.。 	

6.7.15 DECO (Decode)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	DECO			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	DECO	DECOIN, OUT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE	I、Q、M、V、L、SM、constant
OUT	Output	WORD	Q、M、V、L、SM

This instruction uses the value represented by the lower four bits of the input byte IN to specify a bit number, and then assigns the corresponding bit in the output word OUT to 1 according to the bit number, and assigns all other bits to 0.

- LD

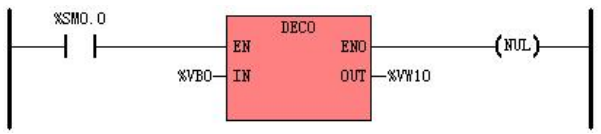
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

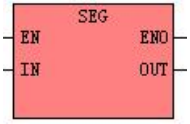
  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

<table>
<tr><td>LD</td><td> 
</td><td>SM0.0 is always 1, so the DECO instruction is always executed: the value represented by the lower four bits of VB0 represents the bit number. According to this bit number, the corresponding bit in VW0 is assigned to 1, and the others are all assigned to 0.</td></tr>
<tr><td>IL</td><td colspan="2">LD %SM0.0
DECO %VB0, %VW10</td></tr>
<tr><td>Result</td><td colspan="2">Examples of results are as follows, The value of VW10 in the figure is represented by binary。
</td></tr>
</table>

6.7.16 SEG (Seven-segment code)

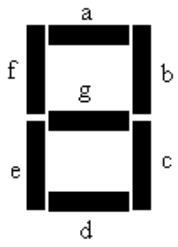
➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	SEG			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	SEG	SEGIN, OUT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE	I、Q、M、V、L、SM、constant
OUT	Output	BYTE	Q、M、V、L、SM

The lower four bits of the input byte IN represent the value to be displayed, and the SEG instruction generates the segment code value of the seven-segment code according to the value and outputs it to OUT.

IN Low 4 bits	Display	*OUT* (- g f e d c b a)
0	0	0 0 1 1 1 1 1 1
1	1	0 0 0 0 0 1 1 0
2	2	0 1 0 1 1 0 1 1
3	3	0 1 0 0 1 1 1 1
4	4	0 1 1 0 0 1 1 0
5	5	0 1 1 0 1 1 0 1
6	6	0 1 1 1 1 1 0 1
7	7	0 0 0 0 0 1 1 1



IN Low 4 bits	Display	*OUT* (- g f e d c b a)
8	8	0 1 1 1 1 1 1 1
9	9	0 1 1 0 0 1 1 1
A	A	0 1 1 1 0 1 1 1
B	B	0 1 1 1 1 1 0 0
C	C	0 0 1 1 1 0 0 1
D	D	0 1 0 1 1 1 1 0
E	E	0 1 1 1 1 0 0 1
F	F	0 1 1 1 0 0 0 1

- LD

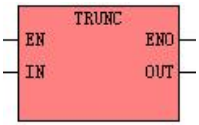
  If EN is 1, the instruction is executed.

- IL

If the CR value is 1, the instruction is executed.The execution of this instruction does not affect the CR value.

6.7.17 TRUNC (Rounding)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	TRUNC			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	TRUNC	TRUNC*IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	REAL	V、L 、constant
OUT	Output	DINT	M、V、L、SM

This instruction converts the input real number IN to a double integer (the fractional part is discarded) and assigns it to OUT.

- LD

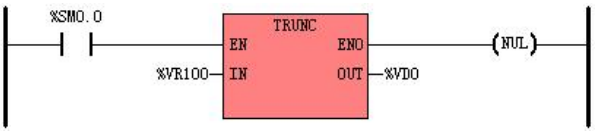
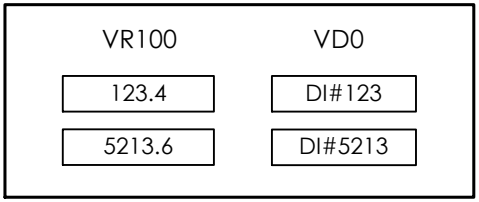
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

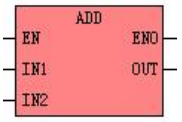
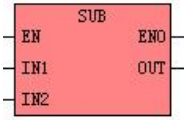
- Instruction usage example

LD		SM0.0 is always 1, so TRUNC is always executed: the real number VR100 is converted to a double integer (the fractional part is discarded) and assigned to VD0.
IL	LD %SM0.0 (* Create CR with a value of 1 *) TRUNC %VR100, %VD0 (*Convert the real number VR100 to a double integer after discarding the fractional part and assign it to VD0 *)	
Result	Examples of results are as follows: 	

6.8 computation

6.8.1 ADD (Addition)、SUB (subtraction)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	ADD			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	SUB			

IL	ADD	ADD *IN1, OUT*	U	
	SUB	SUB *IN1, OUT*		

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	INT、DINT、REAL、BYTE、WORD、DWROD	I、Q、AI、AQ、M、V、L、SM、T、C、HC、constant、pointer
IN2	Input	INT、DINT、REAL、BYTE、WORD、DWROD	I、Q、AI、AQ、M、V、L、SM、T、C、HC、constant、pointer
OUT	Output	INT、DINT、REAL、BYTE、WORD、DWROD	Q、AQ、M、V、L、SM、pointer

The data types of parameters IN1, IN2, and OUT must be the same.

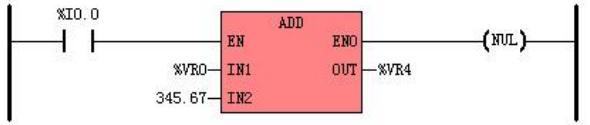
- LD

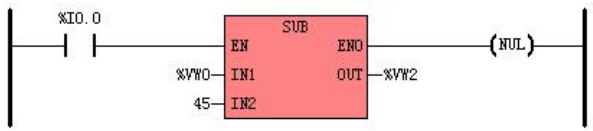
If the value of EN is 1, the instruction is executed. Among them, the function of the ADD instruction is: *OUT*=*IN1*+*IN2*; The function of the SUB instruction is: *OUT*=*IN1*−*IN2*。

- IL

If the value of CR is 1, the instruction is executed. Among them, the function of the ADD instruction is: *OUT*= *OUT* + *IN1*; The function of the SUB instruction is: *OUT*= *OUT*−*IN1*。 The execution of ADD and SUB instructions does not affect the CR value.

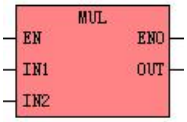
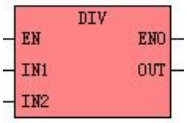
➢ Instruction usage example

LD		If I0.0 is 0: The ADD instruction is not executed. If I0.0 is 1: VR4 = VR4 + 345.67.

		If I0.0 is 0: The SUB instruction is not executed. If I0.0 is 1: VW2 = VW2 – 45.
IL	LD %I0.0 (* Create CR with the value of I0.0 *) ADD 345.67, %VR4(* if CR is 1: VR4 = VR4 + 345.67 *) (* if CR is 0: ADD instruction is not executed *)	
	LD %I0.0 (* Create CR with the value of I0.0 *) SUB 45, %VW2(* if CR is 1: VW2 = VW2－45 *) (* if CR is 0: SUB instruction is not executed *)	

6.8.2 MUL (multiplication)、DIV (division)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	MUL			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	DIV			
IL	MUL	MUL *IN1, OUT*	U	
	DIV	DIV *IN1, OUT*		

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	INT、DINT、REAL、BYTE、WORD、DWROD	I、Q、AI、AQ、M、V、L、SM、T、C、HC、constant、pointer
IN2	Input	INT、DINT、REAL、BYTE、WORD、DWROD	I、Q、AI、AQ、M、V、L、SM、T、C、HC、constant、pointer
OUT	Output	INT、DINT、REAL、BYTE、WORD、DWROD	Q、AQ、M、V、L、SM、pointer

The data types of parameters IN1, IN2, and OUT must be the same.

If the divisor of DIV is 0, the output value of the instruction maintains the last result, and the PLC records the "divide by 0" error.

- LD

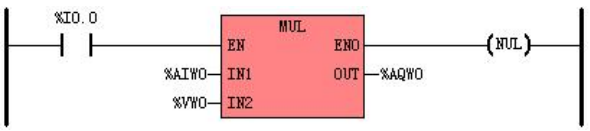
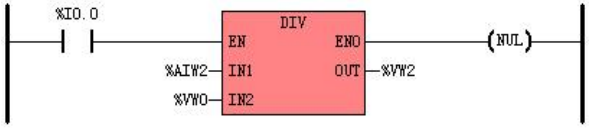
If the value of EN is 1, the instruction is executed. Among them, the function of the MUL instruction is: OUT=IN1×IN2; the function of the DIV instruction is: OUT=IN1÷IN2.

- IL

If the CR value is 1, the instruction is executed. Among them, the function of the MUL instruction is: OUT=OUT×IN1; the function of the DIV instruction is: OUT=OUT÷IN1.

The execution of MUL and DIV instructions does not affect the CR value.

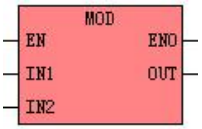
➢ Instruction usage example

<table>
<tr><td rowspan="2">LD</td><td> 
</td><td>If I0.0 is 0: MUL instruction is not executed.
If I0.0 is 1: Multiply AIW0 by VW0 and assign the result to AQW0.</td></tr>
<tr><td> 
</td><td>If I0.0 is 0: DIV instruction is not executed.
If I0.0 is 1: Divide AIW2 by VW0 and assign the result to VW2.</td></tr>
<tr><td>IL</td><td colspan="2">LD %I0.0 (* Create CR with the value of I0.0 *)
MUL %AIW0, %VW0 (* if CR is 1: VW0 = VW0×AIW0 *)
(*If CR is 0: MUL instruction is not executed *)</td></tr>
</table>

```
LD   %I0.0                (* Create CR with the value of I0.0  *)
DIV  %AIW2, %VW0          (* if CR is 1: VW0 = VW0÷AIW2  *)
                          (* if CR is 0: DIV instruction is not executed *)
```

6.8.3 MOD (Find Remainder)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	MOD			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	MOD	MOD *IN1*, *OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	INT、DINT、BYTE、WORD、DWROD	I、Q、AI、AQ、M、V、L、SM、T、C、HC、constant、pointer
IN2	Input	INT、DINT、BYTE、WORD、DWROD	I、Q、AI、AQ、M、V、L、SM、T、C、HC、constant、pointer
OUT	Output	INT、DINT、BYTE、WORD、DWROD	Q、AQ、M、V、L、SM、pointer

The data types of parameters IN1, IN2, and OUT must be the same.

If the divisor (IN2) is 0, the output value of the instruction maintains the last result, and the PLC records the "divide by 0" error.

- LD

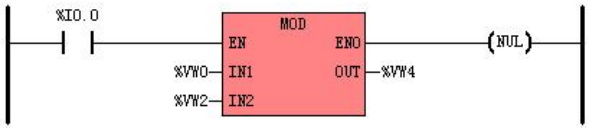
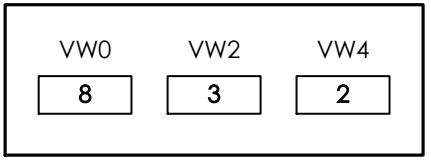
  If the value of EN is 1, the instruction is executed: divide IN1 by IN2 and assign the remainder to OUT.

- IL

  If the value of CR is 1, the instruction is executed: divide OUT by IN1 and assign the remainder to OUT.

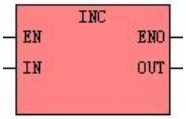
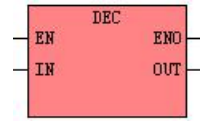
  The execution of the MOD instruction does not affect the CR value.

➢ Instruction usage example

LD		If I0.0 is 0: do not execute the MOD instruction. If I0.0 is 1: divide VW0 by VW2, and assign the remainder to VW4.
IL	LD %I0.0 (* Create CR with the value of I0.0 *) MOD %VW0, %VW4 (*If I0.0 is 0: do not execute the MOD instruction. If I0.0 is 1: divide VW0 by VW2, and assign the remainder to VW4.*) (*If CR is 0: do not execute MOD instruction*)	
Result	If the MOD command in the LD example above is executed, then examples of results are as follows: 	

6.8.4 INC (add 1)、DEC (minus 1)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	INC			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	DEC			
IL	INC	INC *OUT*	U	
	DEC	DEC *OUT*		

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	INT、DINT、BYTE、WORD、DWROD	I、Q、AI、AQ、M、V、L、SM、T、C、HC、constant、pointer
OUT	Output	INT、DINT、BYTE、WORD、DWROD	Q、AQ、M、V、L、SM、pointer

The data types of parameters IN and OUT must be the same.

- LD

If the value of EN is 1, the instruction is executed. Among them, the function of the INC instruction is: OUT=IN+1; the function of the DEC instruction is: OUT=IN-1.

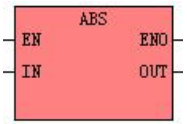
- IL

If the CR value is 1, the instruction is executed. Among them, the function of the INC instruction is: OUT=OUT+1; the function of the DEC instruction is: OUT=OUT-1.

The execution of INC and DEC instructions does not affect the CR value.

6.8.5 ABS (Absolute value)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	ABS			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	ABS	ABS *IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	INT、DINT、REAL	I、Q、V、M、L、SM、T、C、AI、AQ、HC、constant、pointer
OUT	Output	INT、DINT、REAL	Q、V、M、L、SM、AQ、pointer

The data types of parameters IN and OUT must be the same.

This instruction calculates the absolute value of the input IN and assigns the result to OUT, as shown in the following formula: OUT=|IN|.

- LD

  If the value of EN is 1, the instruction is executed.

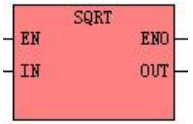
- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

6.8.6 SQRT (square root)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	SQRT			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	SQRT	SQRT *IN*, *OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	REAL	V、L、constant、pointer
OUT	Output	REAL	V、L、pointer

This instruction squares the input IN and assigns the result to OUT, as shown in the following formula:

OUT= $\sqrt{IN}$ 。

Note: If the input parameter IN is a negative number, the output value of the instruction will maintain the previous result, and the PLC will record the error at the same time.

- LD

  If the value of EN is 1, the instruction is executed.

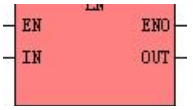
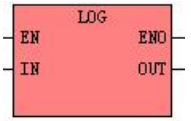
- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

6.8.7 LN (natural log)、LOG (log)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	LN			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	LOG			
IL	LN	LN *IN*, *OUT*	U	
	LOG	LOG *IN*, *OUT*		

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	REAL	V、L、constant、pointer
OUT	Output	REAL	V、L、pointer

The LN instruction takes the natural logarithm of Input IN and assigns the result to OUT, as shown in the following formula: OUT=loge(IN).

The LOG instruction calculates the common logarithm of the input IN and assigns the result to OUT, as shown in the following formula: OUT=log10(IN).

Note: If the Input parameter IN is 0 or a negative number, the output value of the instruction will maintain the last result, and the PLC will record the error.

- LD

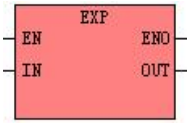
  If the value of EN is 1, the instruction is executed.

- IL
  - If the CR value is 1, the instruction is executed.

  Running the LN and LOG commands does not affect the CR value.

6.8.8 EXP (exponents with base e)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	EXP			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	EXP	EXP *IN, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	REAL	V、L、constant、pointer
OUT	Output	REAL	V、L、pointer

This instruction takes the input IN to the base e and assigns the result to OUT, as shown in the following formula: OUT＝e^{IN}。

- LD

  If the value of EN is 1, the instruction is executed.

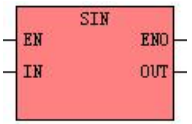
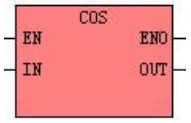
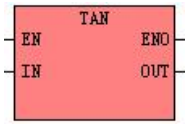
- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

6.8.9 SIN、COS、TAN

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	SIN			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	COS			
	TAN			
IL	SIN	SIN *IN, OUT*	U	
	COS	COS *IN, OUT*		
	TAN	TAN *IN, OUT*		

Parameter	Input/Output	Data type	Allowed memory area

IN	Input	REAL	V、L、constant、pointer
OUT	Output	REAL	V、L、pointer

The input IN represents the value in radians.

The SIN instruction sines IN and assigns the result to OUT, as shown in the following formula: OUT=SIN(IN).

The COS instruction cosines IN and assigns the result to OUT, as shown in the following formula: OUT=COS(IN).

The TAN instruction tangents IN and assigns the result to OUT, as shown in the following formula: OUT=TAN(IN).

- LD

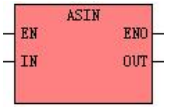
  If the value of EN is 1, the instruction is executed.

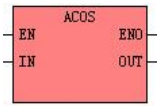
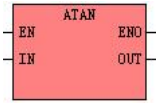
- IL

  If the CR value is 1, the instruction is executed.The execution of the instruction does not affect the CR value.

6.8.10 ASIN、ACOS、ATAN

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW
LD	ASIN			

	ACOS			☑ MK ☑ K6
	ATAN			
IL	ASIN	ASIN *IN, OUT*	U	
	ACOS	ACOS *IN, OUT*		
	ATAN	ATAN *IN, OUT*		

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	REAL	V、L、constant、pointer
OUT	Output	REAL	V、L、pointer

The ASIN instruction takes the arc sine of IN and assigns the result to OUT, as shown in the following formula: OUT=ARCSIN(IN).

The ACOS instruction cosines IN and assigns the result to OUT as follows: OUT=ARCCOS(IN).

The ATAN instruction tangents IN and assigns the result to OUT, as shown in the following formula: OUT=ARCTAN(IN).

Note: The result is a radian value.

- LD

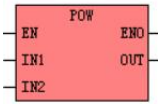
  If the value of EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.The execution of the instruction does not affect the CR value.

6.8.11 POW (Exponentiation)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	POW			☑ K2 ☑ KS ☑ MK ☑ K6
IL	POW	*POW IN1,IN2, OUT*	U	

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	REAL	V、L
IN2	Input	REAL	V、L、constant
OUT	Output	REAL	V、L

This instruction is used to find the value of the power of IN2 with the base IN1 and assign the result to OUT, as shown in the following formula:$OUT = IN1^{IN2}$。

Note: If the base IN1 is negative and the exponent IN2 is not an integer, the calculation cannot be performed, and an incorrect result will be output; if both IN1 and IN2 are 0, the output result of this instruction is 1. This result has only mathematical limit significance but has no meaning in practical application.

- LD

  If the value of EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

6.9 array instruction

The new KPLC provides a set of array instructions for the convenience of users. The specific models and software versions that support array instructions are shown in the table below.

CPU type	Firmware version	Programming software version
K209M	Any version is supported	KincoBuilder V8.1.0.0 and later versions
KS101M		
K6 series		

6.9.1 Overview of Array Instructions

In practical application programming, we will encounter requirements such as mathematical processing of large amounts of data or repeated calls to some single logic. If basic logic instructions and mathematical operation instructions are used for processing, a large number of programs need to be written, which is inconvenient to use and occupies program space. Using array instructions can effectively solve such problems and make the program simple and clear.

Array refers to a collection used to store multiple data of the **same data type**. It is a form in which several elements of the same type are organized together for the convenience of processing in program design.

KPLC supports multiple arrays, each array is assigned a unique number, and the user operates the corresponding array through the number in the program. In the following, we usually refer to the array numbered N by "array N". For example, K6 provides 16 arrays, and their numbers are integers from 0 to 15. In the user program, all array instructions whose input parameter "ARRAY" is "1" are to operate "array 1" (i.e. the array numbered "1").

Each data stored in the array is called the element of the array, and each array is allowed to store up to 1024 elements. In an array, in order to distinguish each element, each element position is assigned a unique number, which is called a subscript, and the user operates the corresponding element in the array through the subscript in the program. For example, the maximum number of elements in a single KPLC array is 1024, and the subscripts of each element are from 0 to 1023 in sequence.

Arrays have data types. When a data type is specified for an array, it means that all elements in the array

have the same data type. For example, the data type of "array 0" is INT, then the data type of all elements in this array is INT.

The following table shows an example of the composition of an array.

array number	Subscript 0	subscript1	subscript2	subscript3	Description
8	DI#252	DI#-3	DI#16	DI#678	The array numbered 8 (array 8), its data type is DINT, contains a total of 4 elements of subscript 0-3. In this example, the element value of the subscript 2 in array 8 is DI#16.

In order to facilitate the use of arrays, KincoBuilder provides the following array-related instructions, which are located in the [Array Instructions] group of the instruction set.

Name	Description
A_READ	read element in array
A_WRITE	write element in array
A_FILL	fill array
A_GETSIZE	Get the length (size) of an array
A_SETSIZE	Set the length (size) of the array
A_GETTYPE	Get the data type of an array element
A_SETTYPE	Set the data type of an array element
A_MIN	Find the minimum value of the data in the specified range in the array
A_MAX	Find the maximum value of data within a specified range in an array
A_AVE	Find the average of data within a specified range in an array
A_SUM	Find the sum value of the data in the specified range in the array
A_SORT	Sorts the data in the specified range in the array

1) Precautions

When using these instructions, users should pay attention to the following points:

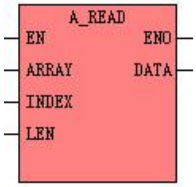
- Only one-dimensional arrays are supported, two-dimensional arrays, three-dimensional arrays, etc. are not supported.
- In a user project, a total of 16 array numbers can be used, and the numbers are 0-15 in sequence. The maximum number of elements in each array is 1024, and the subscript numbers of the elements are 0-1023.

- The number of elements of an array can be specified with the A_SETSIZE directive. If not specified, a maximum of 1024 elements is supported by default. However, it is recommended that the user first specify the number of array elements when using it.
- Array supports all data types (BOOL, BYTE, WORD, DWORD, INT, DINT, REAL). Users can call the A_SETTYPE instruction to specify the data type of the array. If no data type is specified, the array allows elements of all data types to be stored by default, but it is recommended that users specify a data type for the array first. Before using A_MIN, A_MAX, A_AVE, A_SUM and other operation instructions, you must use the A_SETTYPE instruction to specify the data type of the array.
- If the elements in the array are not initialized and assigned (the A_FILL instruction can be used), each element will use the default initial value according to the data type, for example, the initial value of BOOL type is FALSE, and the initial value of REAL type is 0.0.
- The memory space occupied by the array is independent and does not occupy basic memory space such as V and M. **The array does not support power-down save!**
  - The array instruction provides information output such as execution error (as shown in the figure below), but does not provide parameters such as instruction execution error output. It is recommended that the user reasonably plan the data type, element, array number, length, etc. of the array before use to avoid errors.
- All input parameters using array instructions need to be careful not to exceed the allowable range, such as array number, element subscript, etc. After the execution of the array instruction fails, an error code will be generated, which can be read with KincoBuilder.

Error code	Description
600	The array number of the array instruction is wrong
601	Subscript error for array instruction, or wrong length for array operation
602	The array instruction operates on an unsupported data type or is inconsistent with the array data type explicitly specified by the user
603	The array operation has the wrong length, or exceeds a user-specified array length limit
606	The memory area of the input and output of the array instruction is insufficient, for example, the back boundary of the output memory area is exceeded
607	Unsupported sort order

6.9.2 array instruction

6.9.2.1 A_READ (read array element)

	Name	Instruction format	Applicable products
LD	A_READ		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
INDEX	Input	INT	0-1023	I、Q、V、M、L、constant
LEN	Input	INT	1-1024	I、Q、V、M、L、constant
DATA	Output	BOOL,BYTE, WORD,DWORD INT,DINT, REAL	----	V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
INDEX	The starting index address of the object to be accessed in the array, range 0-1023
LEN	Read the number of array elements, range 1-1024
DATA	The starting address where the read data is stored

Note: ARRAY, INDEX, LEN must be constant or variable at the same time, and these parameters form a variable-length memory block. This memory block must all be located in a legal memory area, otherwise the array instruction will be executed incorrectly.

This instruction is used to copy and transfer the consecutive LEN elements from the starting element

of the INDEX parameter in the array numbered ARRAY to the consecutive LEN variables starting from the address DATA. The transmitted data type remains the same, and the maximum length of the transmission should not exceed the maximum valid element range of 1024, nor should it exceed the valid element range of the array specified by the A_SETSIZE instruction.。

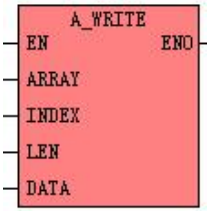
· LD Format instruction description

If EN is 1, the instruction is executed.

If EN is 0, the instruction is not scanned and will not be executed.

· Instruction usage, see 6.9.3 Array usage example for details.

6.9.2.2 A_WRITE (write array element)

	Name	Instruction format	Applicable products
LD	A_WRITE		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
INDEX	Input	INT	0-1023	I、Q、V、M、L、constant
LEN	Input	INT	1-1024	I、Q、V、M、L、constant
DATA	Input	BOOL,BYTE, WORD,DWORD INT,DINT, REAL	----	I、Q、V、M、L

Note: ARRAY, INDEX, LEN must be constants or variables at the same time, and these parameters form a variable-length memory block. This memory block must all be located in a legal

memory area, otherwise the array instruction will be executed incorrectly.

Note: VWxxx is recorded in WORD type by default, and VDxxx is recorded in DWORD type by default. If you want to record as INT or DINT and then perform various operations, you need to declare INT or DINT in the global variable table first, and then enter it in the DATA parameter of A_WRITE.

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
INDEX	The starting index address of the object to be accessed in the array, range 0-1023
LEN	Number of elements to write to the array, range 1-1024
DATA	The starting address where the write data is stored

This instruction is used to write the consecutive LEN variables starting from the address DATA into the consecutive LEN elements starting from the starting element of the INDEX parameter in the array numbered ARRAY. The written data type must be consistent with the data type of the array specified by the A_SETTYPE instruction. . The maximum length transferred should not exceed the maximum valid element range of 1024, nor should it exceed the valid element range for this array specified by the A_SETSIZE directive.

· LD Format instruction description

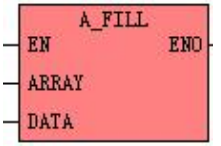
If EN is 1, the instruction is executed.

If EN is 0, the instruction is not scanned and will not be executed.

·For instruction usage, see 6.9.3 Array usage example for details.

6.9.2.3 A_FILL (fill array elements)

	Name	Instruction format	Applicable products

LD	A_FILL		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
DATA	Input	BOOL,BYTE, WORD,DWORD INT,DINT, REAL	----	I、Q、V、M、L、constant

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
DATA	Fill array elements with numeric values

This instruction is used to assign all valid elements in the array numbered ARRAY to the value of DATA. Note that the data type of DATA must be consistent with the data type of the array.

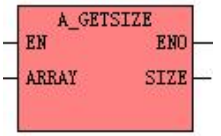
· LD Format instruction description

If EN is 1, the instruction is executed.If EN is 0, the instruction is not scanned and will not be executed.

· For instruction usage, see 6.9.3 Array usage example for details.

6.9.2.4 A_GETSIZE (Get the valid range of array elements)

	Name	Instruction format	Applicable products

LD	A_GETSIZE		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
SIZE	Output	INT	1-1024	V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
SIZE	Valid length of array elements, in the range 1-1024

This instruction is used to obtain the size (ie the number of elements) of the array numbered ARRAY, and write the result to SIZE.

· LD Format instruction description

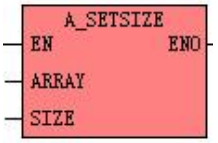
If EN is 1, the instruction is executed.

If EN is 0, the instruction is not scanned and will not be executed.

· For instruction usage, see 6.9.3 Array usage example for details.

6.9.2.5 A_SETSIZE (Set the valid range of array elements)

	Name	Instruction format	Applicable products

LD	A_SETSIZE		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
SIZE	Input	INT	1-1024	I、Q、V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	Enable. If EN is 1, the instruction is enabled, allowing execution.
ARRAY	The array number used ranges from 0 to 15
SIZE	The effective length of an array element, ranging from 1 to 1024

This instruction sets the maximum number of elements allowed in an ARRAY numbered array to SIZE.

· LD Format instruction description

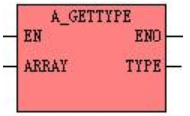
If EN is 1, the instruction is executed.

If EN is 0, the instruction is not scanned and will not be executed.

- Instruction usage. For details, see 6.9.3 Array Usage Examples.

6.9.2.6 A_GETTYPE (Get the data type of an array element)

	Name	Instruction format	Applicable products

LD	A_GETTYPE		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、Constant
TYPE	Output	INT		V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
TYPE	The data type of the array elements, the specific values are as follows: Default type 0 BOOL_TYPE 1 BYTE_TYPE 2 WORD_TYPE 3 INT_TYPE 4 DWORD_TYPE 5 DINT_TYPE 6 REAL_TYPE 7

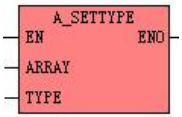
This instruction is used to obtain the data type of the array numbered ARRAY, and write the result to TYPE.

· LD Format instruction description

If EN is 1, the instruction is executed.If EN is 0, the instruction is not scanned and will not be executed.

6.9.2.7 A_SETTYPE (Set the data type of an array element)

	Name	Instruction format	Applicable products

LD	A_SETTYPE		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
TYPE	Input	INT	0-7	I、Q、V、M、L

For the specific usage instructions of each parameter, see the following table:

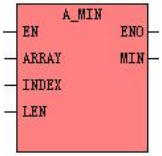
Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
TYPE	Data type of array elements: default type 0 BOOL_TYPE 1 BYTE_TYPE 2 WORD_TYPE 3 INT_TYPE 4 DWORD_TYPE 5 DINT_TYPE 6 REAL_TYPE 7

This instruction is used to set the data type supported by the array numbered ARRAY to TYPE.

· LD Format instruction description

If EN is 1, the instruction executes.If EN is 0, the instruction is not scanned and will not be executed.

6.9.2.8 A_MIN (The minimum value of the data in the specified range in the array)

	Name	Instruction format	Applicable products
LD	A_MIN		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
INDEX	Input	INT	0-1023	I、Q、V、M、L、constant
LEN	Input	INT	1-1024	I、Q、V、M、L、constant
MIN	Output	BYTE, WORD,DWORD, INT,DINT, REAL	----	V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
INDEX	The starting index address of the object to be accessed in the array, range 0-1023
LEN	Number of array elements for data operations, range 1-1024
MIN	The memory address where the minimum value obtained by the operation is stored

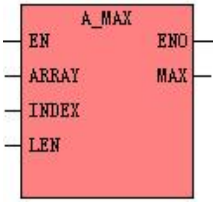
Note: ARRAY, INDEX, LEN must be both constants or variables at the same time! Before using this instruction, you must set the data type of the array with the A_SETTYPE instruction, and this instruction does not support the operation of BOOL type arrays.

This instruction is used to take the minimum value of consecutive LEN elements starting from INDEX in the array numbered ARRAY, and the obtained minimum value is stored in the memory address specified by MIN.

· LD Format instruction description

If EN is 1, the instruction executes.If EN is 0, the instruction is not scanned and will not be executed.

6.9.2.9 A_MAX (The maximum value of data within a specified range in an array)

	Name	Instruction format	Applicable products
LD	A_MAX		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
INDEX	Input	INT	0-1023	I、Q、V、M、L、constant
LEN	Input	INT	1-1024	I、Q、V、M、L、constant
MAX	Output	BYTE, WORD,DWORD INT,DINT, REAL	----	V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
INDEX	The starting index address of the object to be accessed in the array, range 0-1023
LEN	Number of array elements for data operations, range 1-1024
MAX	The memory address where the maximum value obtained by the operation is stored

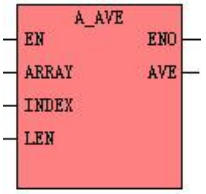
Note: ARRAY, INDEX, LEN must be both constants or variables at the same time! Before using this instruction, you must set the data type of the array with the A_SETTYPE instruction, and this instruction does not support the operation of BOOL type arrays.

This instruction is used to take the maximum value of consecutive LEN elements starting from INDEX in the array numbered ARRAY, and the obtained minimum value is stored in the memory address specified by MAX.

· LD Format instruction description

If EN is 1, the instruction executes.If EN is 0, the instruction is not scanned and will not be executed.

6.9.2.10 A_AVE (The average of data within a specified range in an array)

	Name	Instruction format	Applicable products
LD	A_AVE		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed
ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
INDEX	Input	INT	0-1023	I、Q、V、M、L、constant
LEN	Input	INT	1-1024	I、Q、V、M、L、constant
AVE	Output	BYTE, WORD,DWORD INT,DINT, REAL	----	V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
INDEX	The starting index address of the object to be accessed in the array, range 0-1023
LEN	Number of array elements for data operations, range 1-1024
AVE	The memory address where the average value obtained by the operation is stored

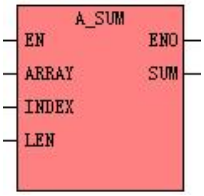
Note: ARRAY, INDEX, LEN must be both constants or variables at the same time! Before using this instruction, you must set the data type of the array with the A_SETTYPE instruction, and this instruction does not support the operation of BOOL type arrays.

This instruction is used to average the consecutive LEN elements starting from INDEX in the array numbered ARRAY, and store the average value in the memory address specified by AVE.

· LD Format instruction description

If EN is 1, the instruction executes.If EN is 0, the instruction is not scanned and will not be executed.

6.9.2.11 A_SUM(Sums the value of a specified range of data in an array)

	Name	Instruction format	Applicable products
LD	A_SUM		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed

ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
INDEX	Input	INT	0-1023	I、Q、V、M、L、constant
LEN	Input	INT	1-1024	I、Q、V、M、L、constant
SUM	Output	BOOL, BYTE, WORD,DWORD, INT,DINT, REAL	----	V、M、L

Note: ARRAY, INDEX, LEN must be both constants or variables at the same time! Before using this instruction, you must set the data type of the array with the A_SETTYPE instruction, and this instruction does not support the operation of BOOL type arrays.

For the specific usage instructions of each parameter, see the following table:

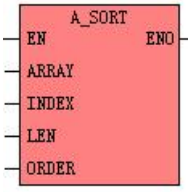
Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
INDEX	The starting index address of the object to be accessed in the array, range 0-1023
LEN	Number of array elements for data operations, range 1-1024
SUM	The memory address where the sum value obtained by the operation is stored

This instruction is used to sum the consecutive LEN elements starting from INDEX in the array numbered ARRAY, and store the calculation result in the memory address specified by SUM

· LD Format instruction description

If EN is 1, the instruction executes.If EN is 0, the instruction is not scanned and will not be executed.

6.9.2.12 A_SORT (Sorts the data in the specified range in the array)

	Name	Instruction format	Applicable products
LD	A_SORT		☑ K209M ☑ KS101M ☑ K6

Parameter	Input/Output	Datatype	Value range	Memory area allowed

ARRAY	Input	INT	0-15	I、Q、V、M、L、constant
INDEX	Input	INT	0-1023	I、Q、V、M、L、constant
LEN	Input	INT	1-1024	I、Q、V、M、L、constant
ORDER	Input	INT	ascending order :1 descending order:2	I、Q、V、M、L、constant

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
ARRAY	Array number to use, in the range 0-15
INDEX	The starting index address of the object to be accessed in the array, range 0-1023
LEN	Number of array elements for data operations, range 1-1024
ORDER	The order in which the data is sorted, 1 is ascending, 2 is descending

Note: ARRAY, INDEX, LEN must be both constants or variables at the same time! Before using this instruction, you must set the data type of the array with the A_SETTYPE instruction!

This instruction is used to sort the consecutive LEN elements starting from INDEX in the array numbered ARRAY. The ORDER parameter value indicates ascending order (from small to large) or descending order (from large to small).

· LD Format instruction description

If EN is 1, the instruction executes.If EN is 0, the instruction is not scanned and will not be executed.

6.9.3 Array usage example

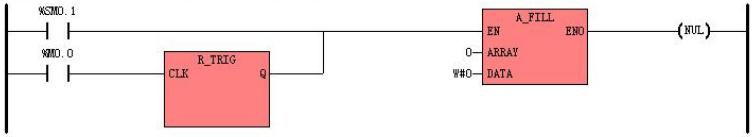
6.9.3.1 Demonstration of the basic instructions of each instruction

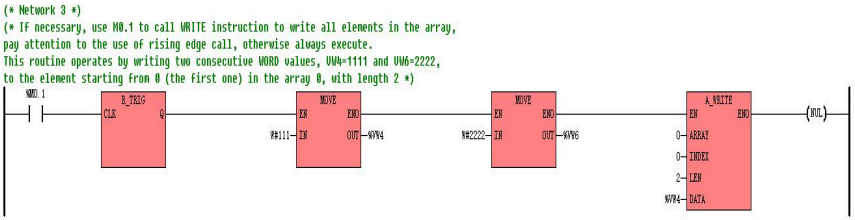
(一)LD format (ladder diagram) program (program example can be downloaded from www.kinco.cn website)

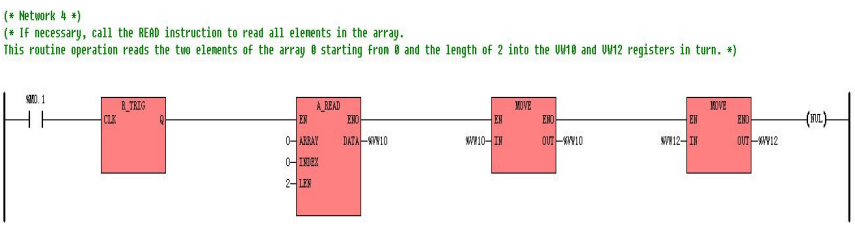
```
(* Network 0 *)
(* For the first time, SETSIZE is used to specify the effective length of array 0 to be 100.
GETSIZE has no practical use here. It is only used for comparison *)
```

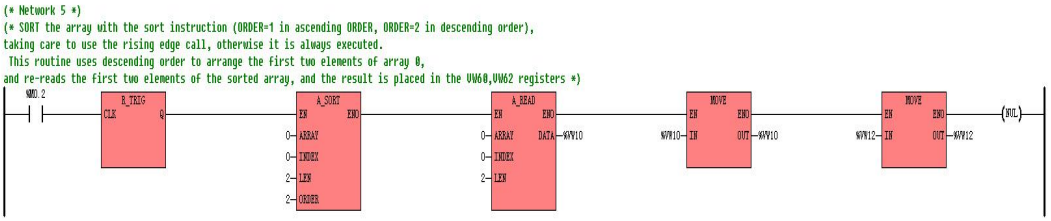
```
(* Network 1 *)
(* The first time SETTYPE is used to set the data type of array 0 to 3(WORD),
 GETTYPE has no practical use here, and can help the program rigor in practical applications.
The meanings of the type are as follows:
Default    0
BOOL_TYPE  1
BYTE_TYPE  2
WORD_TYPE  3
INT_TYPE   4
DWORD_TYPE 5
DINT_TYPE  6 *)
```

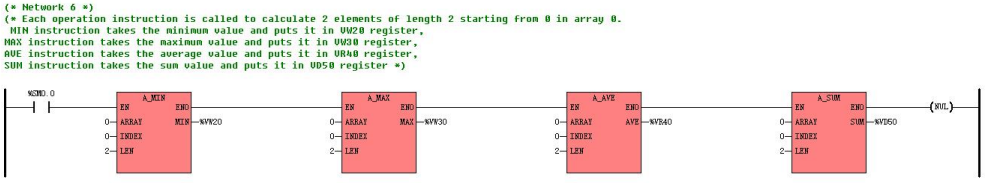
```
(* Network 2 *)
(* When powering on for the first time or if necessary,
call the FILL command with M0.0 to initialize all elements in the array to 0,
pay attention to using the rising edge call, otherwise keep running *)
```







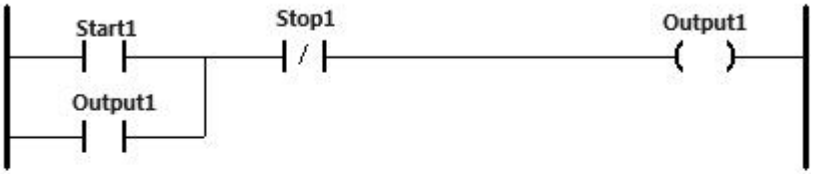




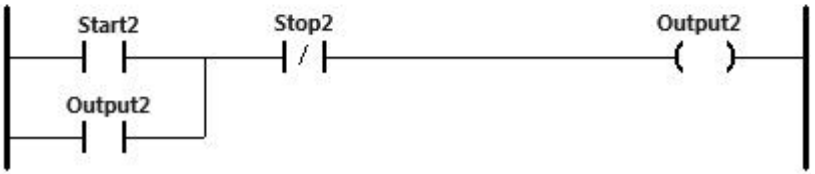
6.9.3.2 Array Application Demonstration 1: Logic Processing

In actual programming, some action logic needs to be processed in the same way, such as the common manual start-stop circuit and marquee output as shown in the ladder diagram below.

```
(* Network 10 *)
```




```
(* Network 11 *)
```



These actions are all repeated in a single way. If they are processed with basic instructions, the exact same program will be copied for as many actions as needed, so that the program looks complicated, and the same is true for the marquee output. Regular single rotation output is easier to handle, but if you encounter situations such as every 4 outputs, or outputs with different rules each time, it is difficult to handle with traditional instructions, but it can be effectively solved by using arrays. The following is a demonstration example of using arrays combined with loop instructions to solve 32 groups of manual start and stop outputs. Only a few lines of program are needed to complete the original 32-line program that needs to be copied, and the program is theoretically increased to a maximum of 1024 groups of the same statement, no need to add more statements.

This example only shows the most common way of combining arrays with loops. There are many other factors to be considered in practical application scenarios, and other methods such as subroutines, pointers, etc. can be used to solve such problems. This example is for reference only and cannot be used in practical applications.

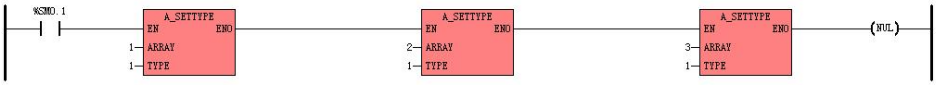
Requirements in this example: There are 32 sets of Input start and stop signals to complete the output.

Program idea:

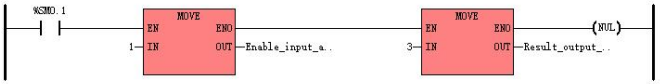
- Assign 32 consecutive bits of ID0 to 32 groups of input start signals, and similarly assign ID4 to stop signals.
- Allocate three arrays 1, 2, 3, the data type In this example, the BOOL action is processed, so it is the BOOL type. Array 1 is used for start signal, array 2 is used for stop signal, and array 3 is used for result output.
- The programming idea is: use the A_WRITE instruction to store the input signals in the input array uniformly, so that the A_READ instruction can be used to read them and decompose them into individual elements, so that the FOR loop can be used to process the 32 groups at one time, and the processing is completed. The result also needs to be stored in the output array in a unified order with the A_WRITE instruction. Finally, the A_READ instruction can also be used to read the value of each element in the output array for use in other parts of the program.

(1): LD format (ladder diagram) program (Examples can be downloaded at www.kinco.cn)

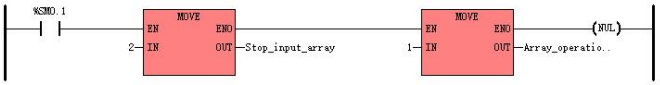
```
(* Network 0 *)
(* Initialize, set all three arrays to BOOL array types (TYPE=1)■Array 1 is the start input array,
array 2 is the stop input array, and array 3 is the result output array *)
```




```
(* Network 1 *)
(* Set some fixed values of the array, mainly because the READ and WIRTE directives
restrict the instruction input parameters to be variable types at the same time *)
```

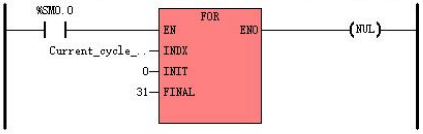



```
(* Network 2 *)
(* Set some fixed values of the array, mainly because the READ and WIRTE directives restrict
 the instruction input parameters to be variable types at the same time *)
```



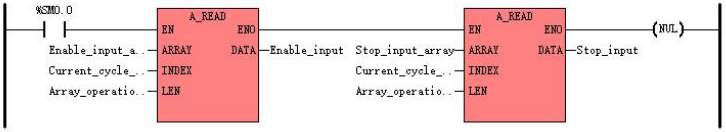
```
(* Network 3 *)
(* Send all 32 input points of the startup input signal ID0 to array 1 by WIRTE instruction as startup input array.
Send all 32 input points of the stop input signal ID4 to array 2 by WIRTE instruction as the stop input array. *)
```

```
(* Network 4 *)
(* Execute a FOR loop that loops 32 times (from INIT=0 to FINAL=31),INDX is the actual number of loops *)
```



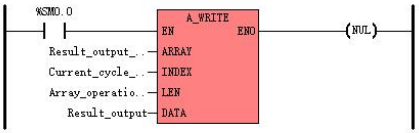

```
(* Network 5 *)
(* Read the current element values of the start input array (storing ID0) and stop input array
(storing ID4) (a total of 32 elements from 0 to 31) to facilitate subsequent logical operations *)
```

```
(* Network 5 *)
(* Read the current element values of the start input array (storing ID0) and stop input array
(storing ID4) (a total of 32 elements from 0 to 31) to facilitate subsequent logical operations *)
```



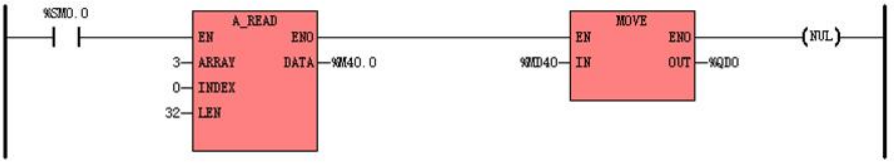

```
(* Network 6 *)
(* Performs logical operations and outputs when started and not stopped *)
```

```
(* Network 7 *)
(* Store the results of each operation into the output array one by one *)
```



```
(* Network 8 *)
(* Closing loop operation *)
```

```
(* Network 9 *)
(* The BOOL value of the individual elements of the output array is the result of the final operation,
which is extracted for use elsewhere in the program. For example, give the output point directly *)
```



6.9.3.3 Array Application Demonstration 2: worksheet output

In actual programming, continuous data needs to be recorded, and it is necessary to perform mathematical processing on the recorded data or find data that meets a certain condition. This kind of worksheet application is more troublesome to use the traditional instruction processing even with the pointer program, and the array is the most suitable for this kind of application.

This example only shows the most common way of combining arrays with loops. There are many other factors to be considered in practical application scenarios, and other methods such as subroutines, pointers, etc. can be used to solve such problems. This example is for reference only and cannot be used in practical applications.

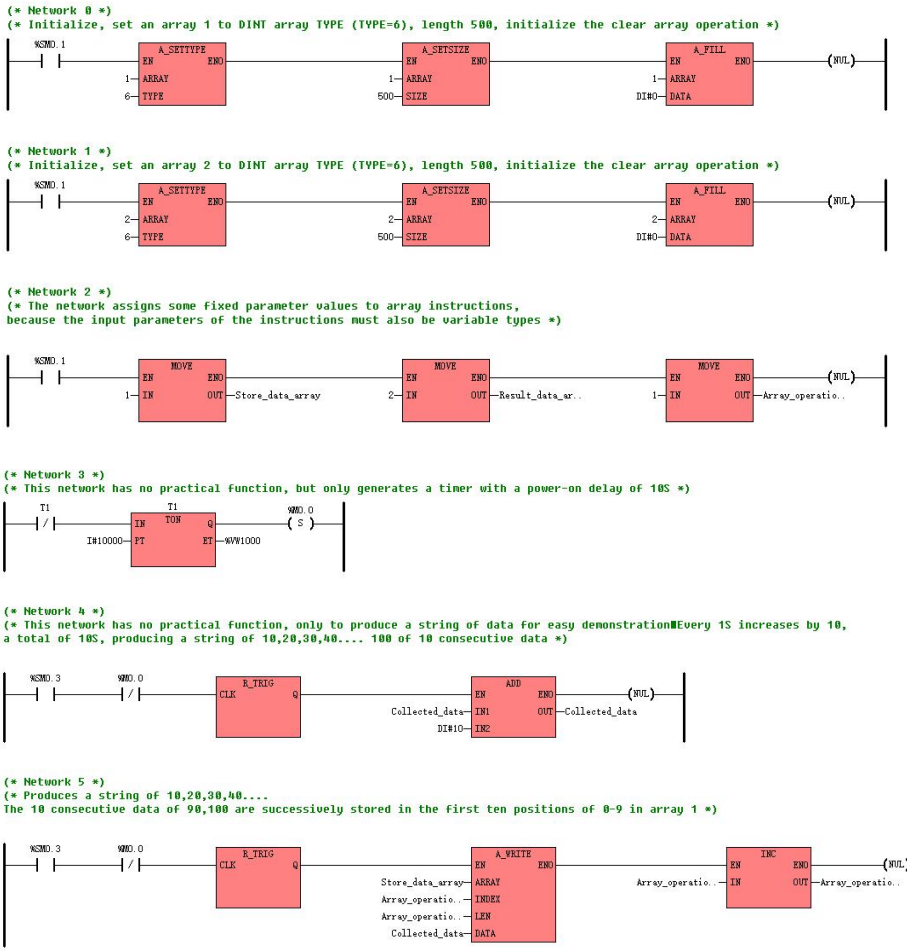
Requirements in this example: collect a set of parameters, remove the values less than 10 and greater than 80, and calculate the average value

Programming idea:

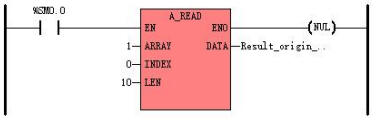
- Allocate two arrays 1 and 2. In this example, the data types are all DINT types. Array 1 is used to store the source data, and array 2 is used to store the processed data that meets the conditions.
- Programming idea: store the source data into the input array in turn with the A_WRITE instruction. In this way, the A_READ instruction can be read out and can be decomposed into

individual elements, so that each element is taken out in turn by the FOR loop and compared with the conditions (greater than 10 and less than 80). The processed results also need to be uniformly stored in the output array with the A_WRITE instruction. Finally, the A_READ instruction can also be used to read the value of each element in the output array to take the average value and use it elsewhere in the program.

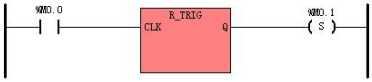
(1): LD format (ladder diagram) program (Examples can be downloaded at www.kinco.cn)



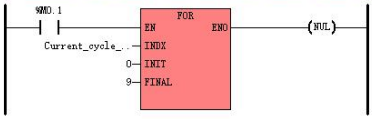
```
(* Network 6 *)
(* Take the continuous DATA stored in the array,
in the continuous area beginning with the output parameter data of the READ instruction, for verification and other programs to use *)
```



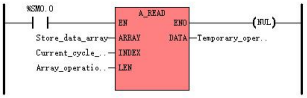

```
(* Network 7 *)
(* After the resulting data is stored in the array,
M0.0, M0.1 is started immediately to perform a FOR loop to process the filtered data *)
```



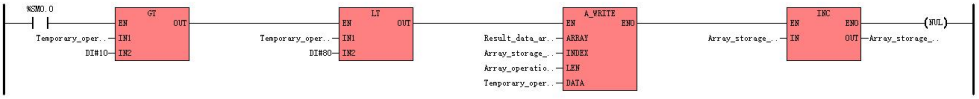

```
(* Network 8 *)
(* Execute a FOR loop that loops 10 times (from INIT=0 to FINAL=9),INDX is the actual number of loops *)
```



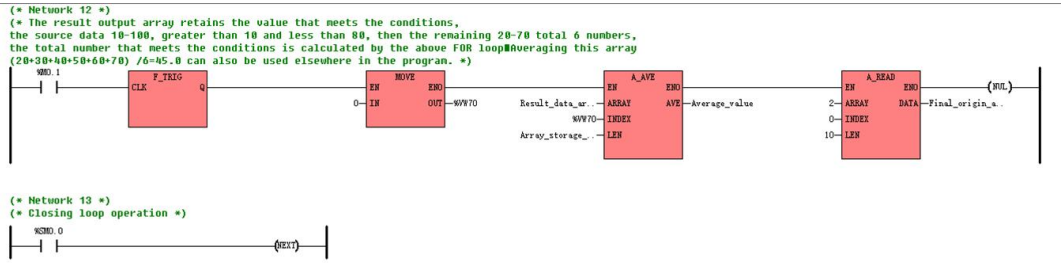
```
(* Network 9 *)
(* Read the current element value of the array containing the data (a total of 10 elements from 0 to 9),
so as to facilitate subsequent filtering of qualified operations *)
```




```
(* Network 10 *)
(* Determine whether the source data is greater than 10, less than 80, and store it in a new array
(that is, discard the data that does not meet the conditions). *)
```




```
(* Network 11 *)
(* Reset M0.1 and the loop is executed only once *)
```



6.10 stack instruction

The new KPLC provides a set of stack instructions for the convenience of users. The specific supported instruction models and related software versions are shown in the table below.

below.	below.	below.
K209M	Any version is supported	KincoBuilder V8.1.0.0 and later versions
KS101M		
K6 series		

6.10.1 Overview of stack instructions

The stack is a commonly used data structure, generally used to store data. The so-called "stack" can be understood as a row of shelves with a specified maximum number of layers. In programming, you can use the push stack instruction (also known as loading the stack, PUSH) to push data into each layer in the "stack" (that is, to store goods). The storage order can be specified first in first out (FIFO) or last in first out (LIFO), and the system background automatically stores the data in each layer according to the rules. When you need to use the data in the "stack", you can use the pop-up instruction (also known as pop, POP) to take out the data (that is, take out the goods), and the order of taking out is the order set when storing. For a clearer view of the stack structure, please refer to the description of the example in the following figure.

FIFO queue	Initial status	Load data 1, after the 1st PUSH instruction	Load data 2, after the 2nd PUSH instruction	Load data 3, after the 3rd PUSH instruction	Load data n, after the 'n'th PUSH instruction	Load data 1024, after the 1024th PUSH instruction
Layer S1 (top of stack)	blank	1	1	1	1	1
Layer S2 (in stack)	blank	blank	2	2	2	2
Layer S3 (in stack)	blank	blank	blank	3	3	3
Layer S4 (in stack)	blank	blank	blank	blank	4	4
Layer SN (in stack)	blank	blank	blank	blank	5 . . . N	5 . . . N
S1024 layer (bottom of stack)	blank	blank	blank	blank	blank	1024
Number of valid layers read by FIFO SIZE	0	1	2	3	N	1024
In summary, data pushed into the stack always enters from the bottom of the stack						

FIFO queue	Initial status	RLEN=1 After the POP command is executed for the 1st time, the DATA read by the command is 1	RLEN=1 After the POP command is executed for the 2nd time, the DATA read by the command is 2	RLEN=1 After the POP command is executed for the 3rd time, the DATA read by the command is 3	RLEN=1 After the POP command is executed for the 'n'th time, the DATA read by the command is n	RLEN=1 After the POP command is executed for the 1024th time, the DATA read by the command is 1024
Layer S1 (top of stack)	1	2	3	4	5	blank
Layer S2 (in stack)	2	3	4	5	6	blank
Layer S3 (in stack)	3	4	5	6	7	blank
Layer S4 (in stack)	4	5	6	7	8	blank
Layer SN (in stack)	5 . . . 1023	6 . . . 1024	7 . . . 1024	8 . . . 1024	9 . . . N	blank
			blank	blank	blank	blank
				blank	blank	blank
					blank	blank
S1024 layer (bottom of stack)	1024	blank	blank	blank	blank	blank
Number of valid layers read by FIFO SIZE	0	1	2	3	N	1024
In summary, the stack data is always read from the top of the stack						

Fig1: FIFO stack operation and background data storage

LIFO queue	Initial status	Load data 1, after the 1st PUSH instruction	Load data 2, after the 2nd PUSH instruction	Load data 3, after the 3rd PUSH instruction	Load data n, after the 'n'th PUSH instruction	Load data 1024, after the 1024th PUSH instruction
Layer S1 (top of stack)	blank	1	2	3	N	1024
Layer S2 (in stack)	blank	blank	1	2	N-1	1023
Layer S3 (in stack)	blank	blank	blank	1	N-2	1022
Layer S4 (in stack)	blank	blank	blank	blank	N-3	1021
Layer SN (in stack)	blank	blank	blank	blank	N-4 . . . 1	1020 . . . 2
S1024 layer (bottom of stack)	blank	blank	blank	blank	blank	1
Number of valid layers read by LIFO SIZE	0	1	2	3	N	1024
In summary, data pushed into the stack always enters from the top of the stack						

TIFO queue	Initial status	RLEN=1 After the POP command is executed for the 1st time, the DATA read by the command is 1024	RLEN=1 After the POP command is executed for the 2nd time, the DATA read by the command is 1023	RLEN=1 After the POP command is executed for the 3rd time, the DATA read by the command is 1022	RLEN=1 After the POP command is executed for the 'n'th time, the DATA read by the command is 1025-n	RLEN=1 After the POP command is executed for the 1024th time, the DATA read by the command is 1
Layer S1 (top of stack)	1024	1023	1022	1021	1024-n	blank
Layer S2 (in stack)	1023	1022	1021	1020	1023-n	blank
Layer S3 (in stack)	1022	1021	1020	1019	1022-n	blank
Layer S4 (in stack)	1021	1020	1019	1018	1021-n	blank
Layer SN (in stack)	1020 . . . 2	1019 . . . 1	1018 . . . 1	1017 . . . 1	1020-n . . . 1	blank
						blank
				blank	blank	blank
			blank	blank	blank	blank
S1024 layer (bottom of stack)	1	blank	blank	blank	blank	blank
Number of valid layers read by TIFO SIZE	0	1	2	3	N	1024
In summary, the stack data is always read from the top of the stack						

Fig2: LIFO stack operation and background data storage

In order to facilitate the use of the stack, KincoBuilder provides the following stack-related instructions, all of which are located in the [Stack Instructions] group of the instruction set.

Name	Function description

FIFO_INIT	Initialize (empty) a FIFO queue
FIFO_PUSH	Load data into FIFO queue
FIFO_POP	Fetch data from FIFO column
FIFO_SIZE	Get the effective height (size) of a FIFO column
LIFO_INIT	Initialize (empty) LIFO queue
LIFO_PUSH	Load data into LIFO queue
LIFO_POP	Fetch data from LIFO column
LIFO_SIZE	Get the effective height (size) of the LIFO column

2) Precautions

When using these instructions, pay attention to the following points::

- In a user project, there are 2 stacks that can be used. They are the FIFO stack that follows the FIFO order and the LIFO stack that follows the LIFO order.
- Both stack columns support a maximum of 1024 data by default, and also support loading any data type (BOOL, BYTE, WORD, DWORD, INT, DINT, REAL) supported by K series PLC. There is no need to specify the data type with special instructions, and the system is automatically compatible, that is, in principle, the user can load stack columns of mixed data types, but the parsing should also be performed with the corresponding data type after the data is retrieved.
- FIFO and LIFO are two independent stacks, and their storage order is different, so their related instructions should be used together. In addition, the PUSH loading and POP popping instructions should also be used together, that is, the status of the stack column initialization (the INIT instruction is available) is empty, and the PUSH instruction must be used to load the data to make sense (the actual effective layer number of the current stack column data can be read out with the SIZE instruction) . It is important to note that 0 and 0.0 are also valid data.
- The stack column instruction occupies the SRAM memory area alone, that is, it does not occupy the space of the basic storage area such as the V area, and **does not support power-down saving.**
- The stack column instruction provides information output such as execution errors (as shown in the figure below), but it does not provide parameters such as instruction execution error output. It is recommended that users plan the stack column reasonably before use to avoid errors! If the stack

column does not execute the PUSH instruction to load the pushed data, it is naturally wrong to execute the POP pop-up instruction.。

The stack column occupies a fixed-length 1024 memory block. All operations must be located in this legal memory area and the PUSH instruction should be used to load valid data first, otherwise the stack instruction will be executed incorrectly.

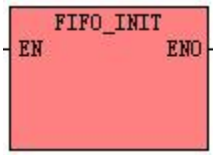
For example, data that exceeds the length of 1024 is loaded, the read is performed without loading the data, or the read range exceeds the effective level of the actual load. The inconsistency between the pop data type and the load type will cause errors in the user's meaning, such as popping an INT to a BYTE, which will cause the high byte value to be lost.

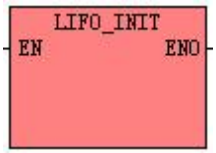
After an error occurs in the execution of the stack instruction, an error code will be generated, which can be read using KincoBuilder.。

Error code	Description
604	The FIFO queue is empty, or the FIFO queue is full
605	The LIFO queue is empty, or the LIFO queue is full

6.10.2 stack instruction

6.10.2.1 FIFO_INIT and LIFO_INIT (initialize stack)

	Name	Instruction format	Applicable products
LD	FIFO_INIT		☑ K209M ☑ KS101M ☑ K6

	LIFO_INIT		

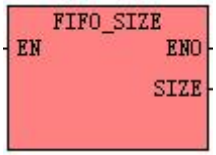
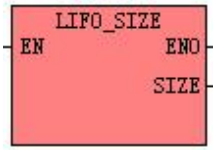
· LDFormat instruction description

If EN is 1, the instruction is executed once per scan cycle, and the FIFO stack (or LIFO stack) is completely emptied each time the instruction is executed.

If EN is 0, the instruction is not scanned and will not be executed.

·For instruction usage, see 6.10.3 Stack Usage Example for details.

6.10.2.2 FIFO_SIZE and LIFO_SIZE (Get the valid load range of the stack column)

	Name	Instruction format	Applicable products
LD	FIFO_SIZE		☑ K209M ☑ KS101M ☑ K6
	LIFO_SIZE		

Parameter	Input/Output	Datatype	Value range	Memory area allowed
SIZE	Output	INT	0-1024	V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
SIZE	Get the storage address of the effective length of the stack column

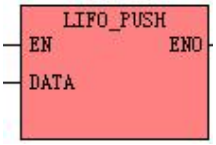
·LD Format instruction description

If EN is 1, the instruction is executed once per scan cycle, and the current actual effective length of the FIFO stack (or LIFO stack) is written to SIZE each time the instruction is executed. Note that it is the effective length. When the stack column is initialized, it is all empty, and its effective length is 0. Each time a PUSH loading data operation is performed, the length is increased by 1 (maximum 1024), and each time a POP operation is performed, the length of the current popping (minimum is 0) of the response is subtracted.。

If EN is 0, the instruction is not scanned and will not be executed.

·Instruction usage, see 6.10.3 Stack usage example for details.

6.10.2.3 FIFO_PUSH and LIFO_PUSH (load data to stack)

	Name	Instruction format	Applicable products
LD	FIFO_PUSH		☑ K209M ☑ KS101M ☑ K6
	LIFO_PUSH		

Parameter	Input/Output	Datatype	Value range	Memory area allowed
DATA	Input	BOOL,BYTE WORD,DWORD INT,DINT	----	I、V、M、L、constant

		REAL		

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
DATA	The address of the data stored in the stack column

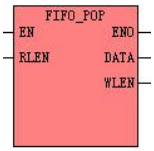
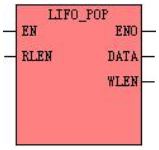
· LDFormat instruction description

If EN is 1, the instruction is executed once per scan cycle, and the data of DATA is stored in the FIFO stack (or LIFO stack) every time the instruction is executed. At the same time, the system background automatically increases the length of the stack column by 1, and determines the storage location according to the FIFO (or LIFO) rule. See 6.10.1 Overview of Stack Instructions for a detailed introduction to stack columns.

If EN is 0, the instruction is not scanned and will not be executed.

·For instruction usage, see 6.10.3 Stack usage example for details。

6.10.2.4 FIFO_POP and LIFO_POP (Pop data from the data stack column)

	Name	Instruction format	Applicable products
LD	FIFO_POP		☑ K209M ☑ KS101M ☑ K6
	LIFO_POP		

Parameter	Input/Output	Datatype	Value range	Memory area allowed
RLEN	Input	INT	1-1024	I、V、M、L、constant

WLEN	Output	INT	0-1024	V、M、L
DATA	Output	BOOL,BYTE WORD,DWORD INT,DINT REAL	----	V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameter	Function
EN	enable terminal. If EN is 1, the instruction is enabled and execution is allowed.
RLEN	Want to read the length of the stack column data
WLEN	The actual length of the successfully read data
DATA	Read the data storage start address in the stack column

· LDFormat instruction description

If EN is 1, the instruction starts to execute once per scan cycle. Each execution of the instruction will copy and transfer RLEN consecutive valid data from the top of the stack in the FIFO stack column (or LIFO stack column) to WLEN variables of DATA . The data type to be transferred remains unchanged, and the maximum length of the transfer should not exceed the maximum valid element range of 1024, nor should it exceed the valid data range of the current stack column obtained by the FIFO_SIZE LIFO_SIZE instruction. After the pop operation is completed, the system background automatically subtracts WLEN from the length of the stack column, and automatically increases the WLEN bits to the top of the stack in order to ensure the uniform rules of data entry and exit in the stack (FIFO or LIFO). See 6.10.1 Overview of Stack Instructions for a detailed introduction to stack columns.

Note that the valid data is operated here. If the current valid data length in the stack is less than the RLEN length, only the actual remaining valid data can be taken out, that is, the actual number of successful WLEN reads will be less than the number of reads set by RLEN.

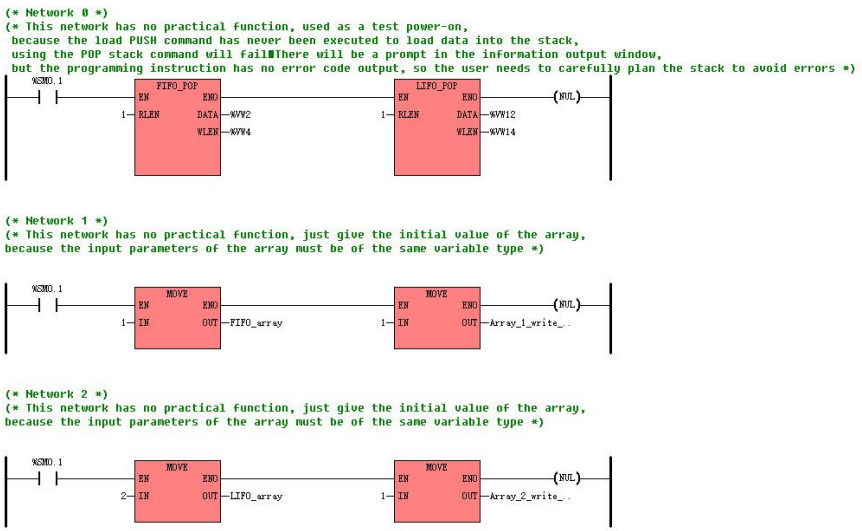
If EN is 0, the instruction is not scanned and will not be executed.

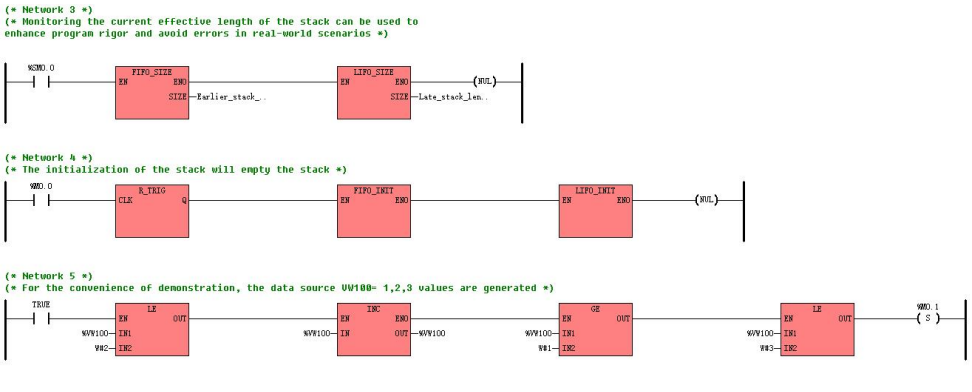
· For instruction usage, see 6.10.1 Stack usage example for details。

6.10.3 stack usage example

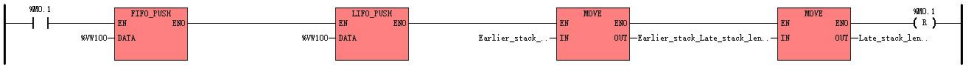
6.10.3.1 Demonstration of the basic instructions of each instruction

(1): LD format (ladder diagram) program (Example can be downloaded from www.kinco.cn)

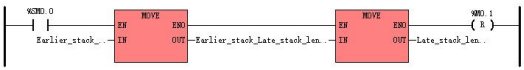




(* Network 6 *)
(* The three values of data source VW100, 1,2, and 3, are stored in two stacks in three times (M0.1 triggers three times).
■After the data is generated and stored in the stack, reset M0.1 *)

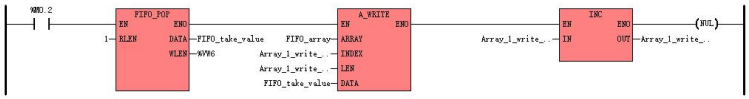


(* Network 7 *)
(* Monitor the actual effective length of the stack, when storing data is 3
(in fact, it is also 123 per load a number of changes once,
but the real PLC is executed per scan cycle, soon can not be monitored,
you can use offline simulation to adjust the scan cycle monitoring)
■Then the value is reduced by 1 every 1S, that is, 3210 *)

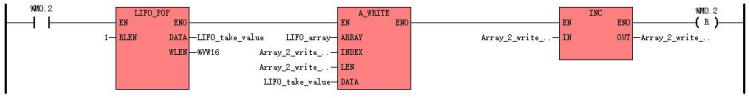


(* Network 8 *)
(* Every second, an M0.2 edge trigger is generated for a total of 3 times *)

(* Network 9 *)
(* Retrieves the number of the stack columns and stores them in the first 3 elements of the first-in first-out array,
three times (M0.2 triggers three times), one at a time (RLEN=1) *)



(* Network 10 *)
(* Take the number of LIFO stack columns and store them in the first 3 elements of the LIFO array in three times (M0.2 will trigger three times),
one at a time (RLEN=1).■After reading the stack column three times and storing the array, reset M0.2 to end the operation *)



(* Network 11 *)
(* Take the first three element values of the first-in first-out array and monitor them separately, their order is 123, *)

(* Network 12 *)
(* The values of the first three elements of the LIFO array are extracted and monitored separately,
and their order is 321, following LIFO *)

6.10.3.2 Stack Application Demo 1: Phone Arrangement

The stack is also a storage for a group of consecutive arrays. It differs from the array in that the order can be specified, that is, the concept of queuing. Therefore, in actual programming, the stack is often used to deal with queuing time, such as customer service phone access, bank call queuing, elevator scheduling, AGV scheduling, etc. Take all total requirements as a queue, and then follow the first-in, first-out order to reflect the role of stack instructions.

This example is only to demonstrate the most common use of stacks combined with loops. There are many other factors to be considered in actual application scenarios. This example is for reference only and cannot be used in practical applications.

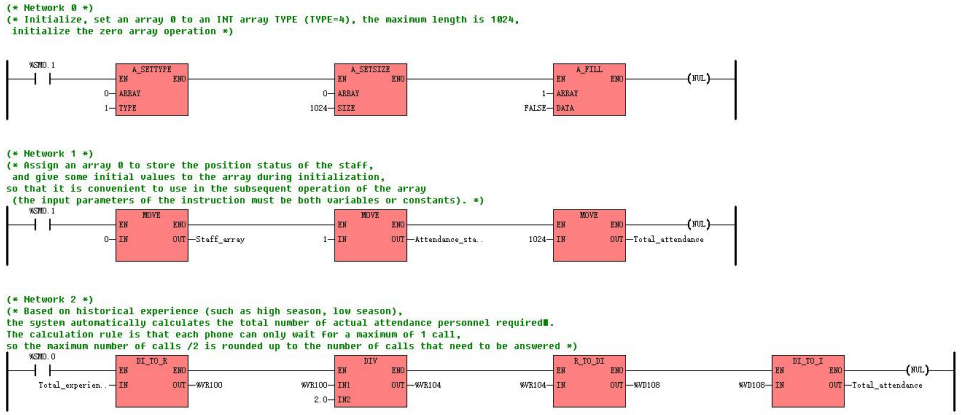
Requirements in this example: There is a customer service call center, which is responsible for answering calls. There are a total of 32 telephones handled by 32 operators. The design program fulfills the following requirements:

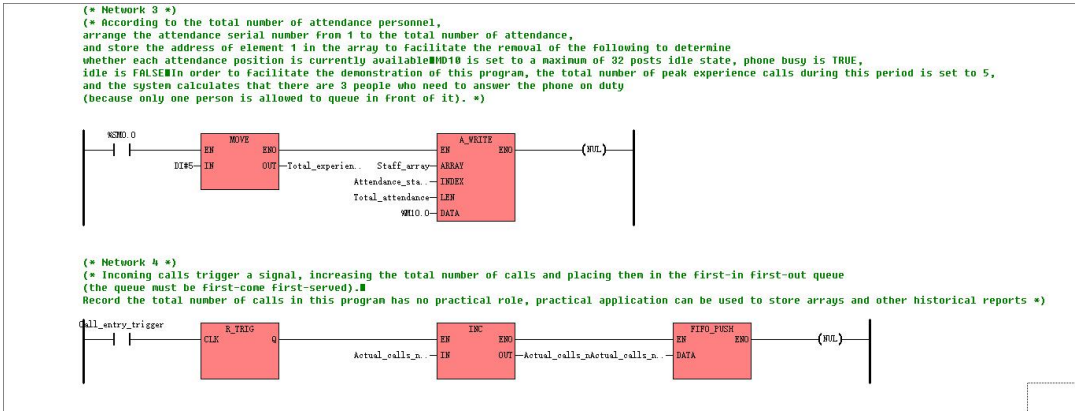
- According to the maximum possible number of peak calls per day obtained from historical experience, and only one call is allowed to be queued at most, the actual number of telephone calls that need to be arranged is automatically calculated.
- After the call comes, it will be assigned to each idle phone in the first-in, first-answer queue order, that is, each phone will be assigned to the corresponding idle post number after it is connected.

Program ideas:

- Allocate an array 0, the data type is BOOL in this example, and the length of the array is 32, that is, the array 0 is used to store the status of whether the current position is free.
- The programming idea: store the incoming calls into the FIFO first-in-first-out queue with the PUSH instruction according to the serial number, and determine whether the length of the stack column is 0 in each scan cycle. The loop takes out each element in the post status array in turn (ie the post status) and compares it with the conditions (whether it is free or not). If it is free, it can be arranged. At the same time, the POP instruction is used to remove the currently processed phone serial number from the stack.

(1): LD format (ladder diagram) program (Example can be downloaded from www.kinco.cn)

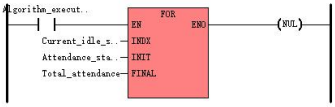




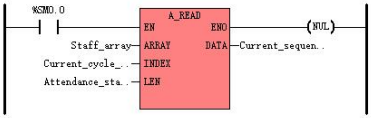
```
(* Network 5 *)
(* The total length of the stack column is the number of currently unscheduled calls,
because scheduled calls have already been removed from the stack by POP *)
```

```
(* Network 6 *)
(* If there are unscheduled calls, the algorithm calculates the number of phone positions that can be accessed until there are no waiting calls *)
```

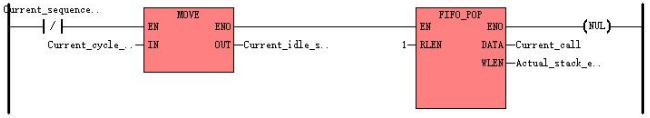
```
(* Network 7 *)
(* Execute the FOR loop, the idea is: there are unscheduled calls waiting,
then execute the loop to see the current vacant position that is available, *)
```



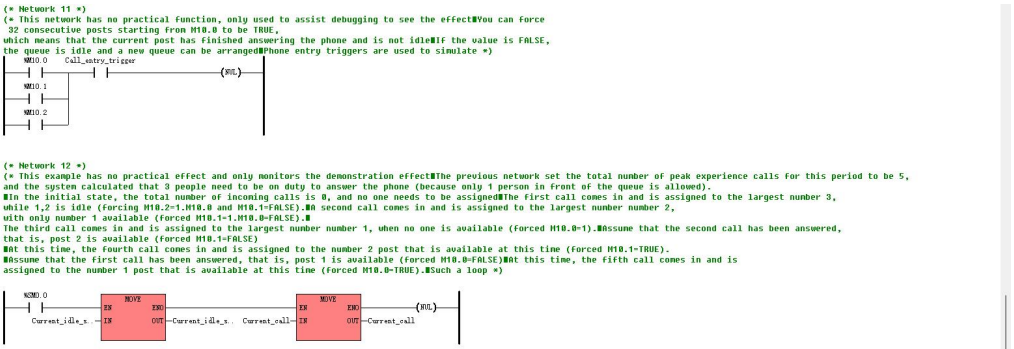

```
(* Network 8 *)
(* Read the idle status of each post *)
```




```
(* Network 9 *)
(* If the current post is idle, the number of the idle post is taken out,
and the next queuer is arranged to this idle post by executing the stack command■At this time,
the position should be set to TRUE, this example does not do special processing, equivalent to manual control
(pick up the phone TRUE, hang up the phone FALSE),
the actual programming can also be processed by the array■If multiple positions are available,
no special processing is performed in this example.
That is, the position with the largest serial number is assigned first by default *)
```




```
(* Network 10 *)
(* Ends the FOR loop statement *)
```



6.10.3.3 Stack application demo 2: reverse order output

The common scenario of stack is to output in reverse order (note that it is distinguished from the sorting instruction of numbers. The sorting instruction of array refers to sorting according to the size of the value, and the reverse order of the queue is the reverse order, which has nothing to do with the specific value. For example, the original order of inputting a string of characters is ABCD. , the reverse order is DCBA).

In practical applications, the reverse sequence output of the logic circuit is often encountered. For example, the output sequence of starting the motor is positive, and the reverse sequence output is required when stopping. If it is output next to each other, such as the standard order of Q0.0-Q0.1-Q0.2-Q0.7, it can also be processed by the shift instruction. But if it is an irregular sequence such as Q0.0-Q0.2-Q0.1, it should be handled with the stack LIFO instruction. The following example demonstrates this processing, in fact it is very common in communication to reverse the order of the received results.见。

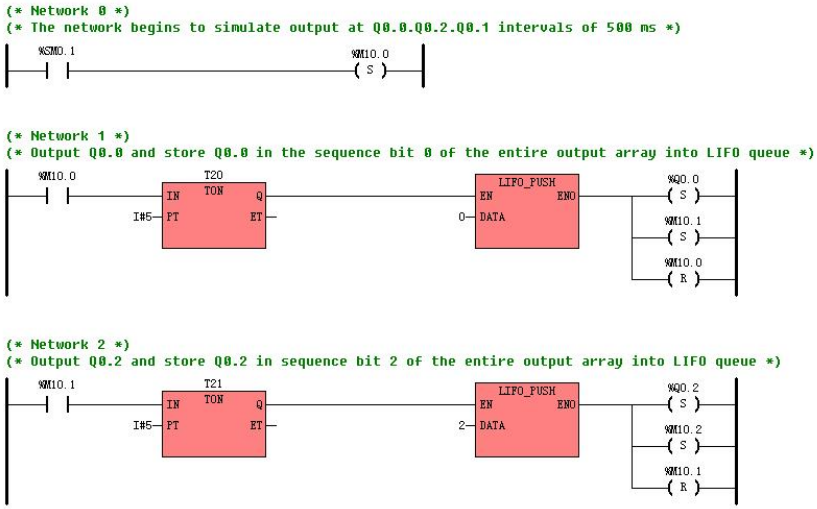
This example is only to demonstrate the most common way of stack output in reverse order. There are many other factors to be considered in actual application scenarios. This example is for reference only and cannot be used in practical applications.

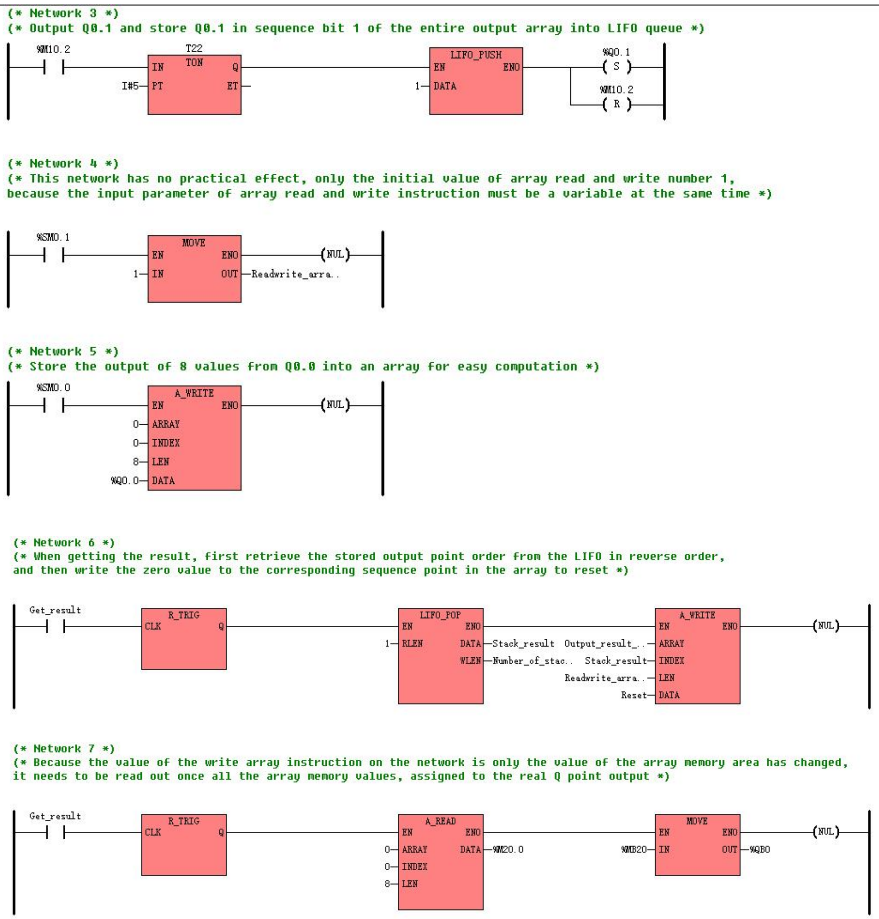
The requirements of this example: after starting, output in the order of Q0.0-Q0.2-Q0.1, and when it stops, output in reverse order of Q0.1-Q0.2-Q0.0

Program idea:

- Allocate an array 0, the data type is BOOL in this example, the array length is 8, that is, the array 0 is used to store the status of the current output point.
- The programming idea is: every time the output is output, the sequence number of the current output point in the entire output queue is stored in the LIFO last-in-first-out queue with the PUSH instruction, and the total state of the output points is stored in the array in the normal order next to each other. . When output in reverse order is required, first execute the POP pop-out instruction to obtain the sequence number of the output point of this pop-up, and then write to the array to reset the corresponding sequence number, but it should be noted that the result of reading all elements of the array once is given to The output point can only be reset in the memory (the previous execution just reset the result in the array), please refer to the program comment for details.

(1): LD format (ladder diagram) program (Example can be downloaded from www.kinco.cn)



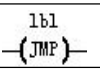
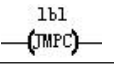
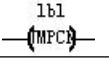


6.11 Program control

6.11.1 Labels and Jump Instructions

- Instruction and its operand description

	Name	Order format	Influence CR value	☑ K5 ☑ K2 ☑ KS
LD	LBL	lbl —(LBL)—		

	JMP			☑ KW ☑ HP ☑ MK ☑ K6
	JMPC			
	JMPCN			
IL	Label	lbl:	U	
	JMP	JMP *lbl*		
	JMPC	JMPC *lbl*		
	JMPCN	JMPCN *lbl*		

Parameter	Description
lbl	legal identifier

Note that the lbl label specified in the jump instruction must exist and must be in the same program as the instruction.

Note that this instruction may take a long time and will trigger the watchdog. The WDR instruction can be added to prolong the triggering time of the watchdog, and pay attention to the JMP condition not to cause an infinite loop that cannot be jumped out.

- LD

The LBL instruction is used to define a label at the current location that will be the destination of the jump instruction. Labels are not allowed to be defined repeatedly. The LBL instruction is unconditionally executed, and it is not recommended that users add any components to the left end of the LBL instruction. In fact, the compiler ignores all elements on the left end of the LBL instruction at compile time.

The JMP instruction is used to unconditionally jump to the lbl label to continue execution.

The function of the JMPC instruction is: if the value of the energy flow on the left side of the instruction is 1, the program will jump to the lbl label and continue to execute, otherwise the instruction will not work, and the program will continue to be executed sequentially.

The function of the JMPCN instruction is: if the value of the energy flow on the left side of the instruction is 0, the program will jump to the lbl label to continue execution, otherwise the instruction will not work, and the program will continue to be executed sequentially.

- IL

The definition format of the label is: lbl: The definition of the label occupies a separate line. Labels are not allowed to be defined repeatedly.

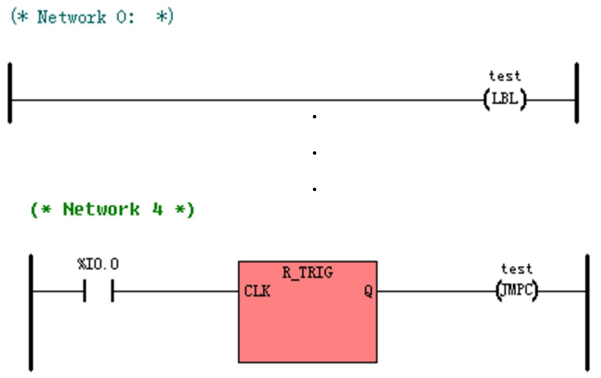
The JMP instruction is used to unconditionally jump the program to the lbl label to continue execution。

The function of the JMPC instruction is: if the CR value is 1, the program will jump to the lbl label to continue execution, otherwise the instruction will not work, and the program will continue to be executed in sequence.

The function of the JMPCN instruction is: if the CR value is 0, the program will jump to the lbl label and continue to execute, otherwise the instruction will not work, and the program will continue to be executed in sequence.。

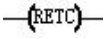
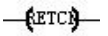
Note: The execution of the jump instruction does not affect the CR value, so if necessary, the user should re-establish a new CR value after the target label of the jump to avoid errors in program execution.

➢ Instruction usage example

LD	IL
	(* NETWORK 0 *) test: ... (* NETWORK 4 *) LD %I0.0 R_TRIG JMPC test

6.11.2 return instruction

- Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
LD	RETC			
	RETCN			
IL	RETC	RETC	U	
	RETCN	RETCN		

The return instruction can only be used in subroutines and interrupt service routines to terminate the program and return to the calling point of the program to continue execution.

At the end of each subroutine and interrupt service routine, KincoBuilder software automatically implicitly calls the RETC instruction.

- LD

RETC instruction: If the value of the energy flow on the left side of the instruction is 1, the instruction is executed, otherwise the instruction does not work, and the program continues to be executed in sequence.

RETCN instruction: If the value of the energy flow on the left side of the instruction is 0, the instruction is executed, otherwise the instruction does not work, and the program continues to be executed in sequence.

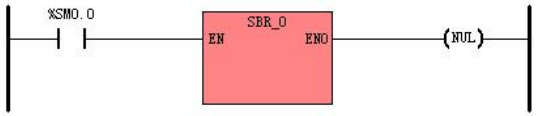
- IL

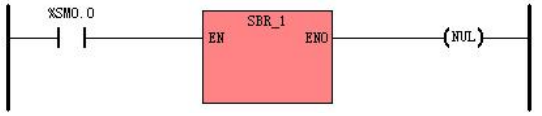
RETC instruction: If the CR value is 1, the instruction is executed, otherwise the instruction does not work, and the program continues to be executed in sequence.

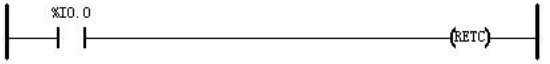
RETCN instruction: If the CR value is 0, the instruction is executed, otherwise the instruction does not work, and the program continues to be executed in sequence.

Note: In addition, the execution of the return instruction does not affect the CR value, so if necessary, the user should re-establish a new CR value after the return point of the program to avoid errors in the program execution.

➢ Instruction usage example

<table>
<tr>
<td>LD</td>
<td>Main program:
(* Network 0 *)
 

(* Network 1 *)
 

Subroutine SBR_0:
(* Network 0 *)
 

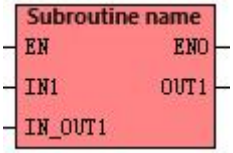
(* Network 1 *)

</td>
<td>In subroutine SBR_0:
If I0.0 is 0, continue to execute the following instructions in sequence.
If I0.0 is 1, return to the calling point of SBR_0 in the main program and continue to execute the instructions in Network 1 downward.</td>
</tr>
<tr>
<td>IL</td>
<td colspan="2">Main program:
LD %SM0.0 (* Create CR, the value is always I *)
CAL SBR_0 (* Call subroutine SBR_0 *)
CAL SBR_1 (*Call subroutine SBR_1 *)

Subroutine SBR_0:
LD %I0.0 (* Create CR with the value of I0.0 *)
RETC (* If CR is 1: return to the main program and continue to execute the CAL SBR_1 instruction *)
LD %I0.1 (* If the RETC instruction does not work, the instruction and subsequent instructions are executed *)
ANDN %I0.2
ST %Q0.0</td>
</tr>
</table>

6.11.3 CAL (call subroutine)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	CAL			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
IL	CAL	CACAL subroutine name, subroutine argument 1, subroutine argument 2...	U	

his instruction is used to call and execute the subroutine with the specified name. After the subroutine is executed, it will return to the instruction after CAL to continue execution. The subroutine called by the CAL instruction must already exist in the user project.。

The actual parameters entered in the CAL instruction must match the data types and variable types of the called subroutine's formal parameters (defined in the subroutine's local variable table). The user must enter the actual parameters of the subroutine in the following order: all input parameters, all input/output parameters, all output parameters.

- LD

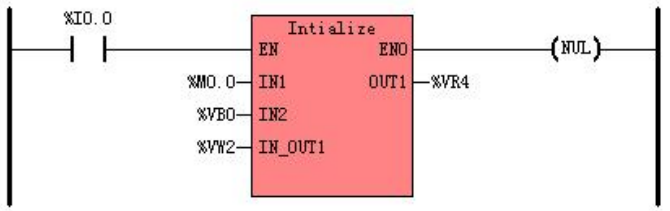
If the user has written a subprogram in the project, the name of the subprogram will appear in the [Subprogram] group of the instruction tree. Double-click the name to add the calling instruction of the subprogram at the corresponding position in the program. If the value of the energy flow on the left side of the call instruction is 1, the subroutine is called and executed.

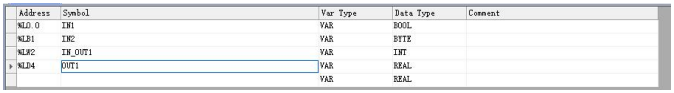
- IL

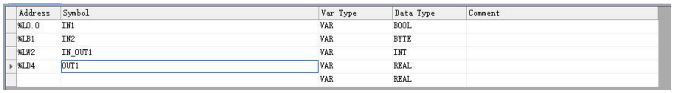
If the CR value is 1, the specified subroutine will be called and executed, otherwise the instruction will not work, and the program will continue to be executed in sequence.

The execution of the CAL instruction does not affect the CR value, but it should be noted that the CR value may have changed in the subroutine.

➢ Instruction usage example

<table>
<tr><td>LD</td><td>Main program :
(* Network 0 *)
 

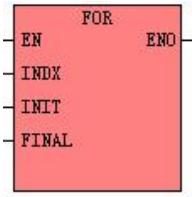
Local variable table in subroutine Initialize:
 
</td><td>In the main program:
If I0.0 is 0, continue to execute the following instructions in sequence.
If I0.0 is 1, the subroutine Initialize is called and executed.</td></tr>
<tr><td>IL</td><td colspan="2">Main program:
(* Network 0 *)
LD %I0.0(* Create CR with the value of I0.0 *)
CAL Intialize %M0.0, %VB0, %VW2, %VR10 (* If CR is 1, call and execute Initialize *)

Local variable table in subroutine Initialize:
 
</td></tr>
</table>

6.11.4 FOR/NEXT (loop instruction)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5

LD	FOR			☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
	NEXT			
IL	FOR	FOR INDX, INIT, FINAL	U	
	NEXT	NEXT		

Parameter	Input/Output	Data type	Memory area allowed
INDX	Input	INT	M、V、L、SM
INIT	Input	INT	M、V、L、SM、T、C、constant
FINAL	Output	INT	M、V、L、SM、T、C、constant

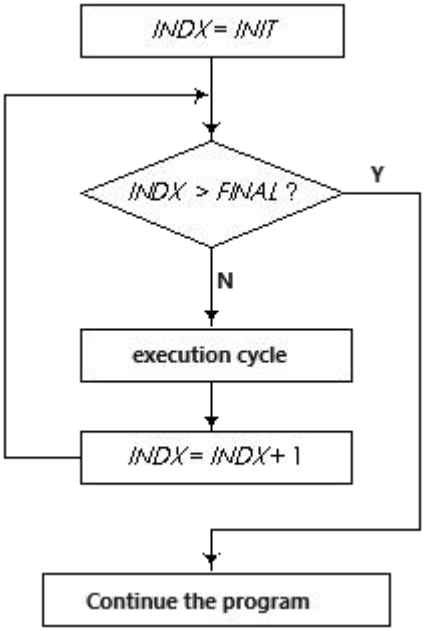
FOR/NEXT is used to implement loops, and is suitable for situations where the number of loops has been determined.

The FOR instruction and the NEXT instruction must be used in pairs, where FOR marks the start of the loop and NEXT marks the end of the loop. The statement between FOR and NEXT is called the loop body. The FOR instruction, its matching NEXT instruction and the loop body between them form a complete loop structure.

The FOR/NEXT instruction repeatedly executes the statements in the loop body until the specified number of loops is reached. The count value of the number of loops is stored in the parameter INDX, while INIT specifies the initial value of INDX, and FINAL specifies the final value of INDX.

A loop body contains another complete loop structure, called the nesting of loops. FOR/NEXT loops support nesting, and the nesting depth is up to 8 levels.

The execution process of the FOR/NEXT loop is as follows:



When using FOR/NEXT loops, users need to pay attention to the following aspects:

✧ The FOR instruction must be the second instruction in a network;

The NEXT instruction must occupy a separate network (Network).。

✧ In the body of the loop, the user can change the final value FINAL, thereby changing the end condition of the loop.

✧ **If there are too many loops, the execution time of the program may exceed the timing value of the system watchdog,** resulting in the failure of CPU scan timeout. The user can use the WDR instruction in the loop body to prolong the trigger time of the watchdog.

✧ **parameter cannot be equal to 32767, INIT must be less than or equal to FINAL, otherwise PLC will not execute the loop body.**

- LD

If the value of the energy flow on the left side of the FOR instruction is 1, the FOR/NEXT loop is

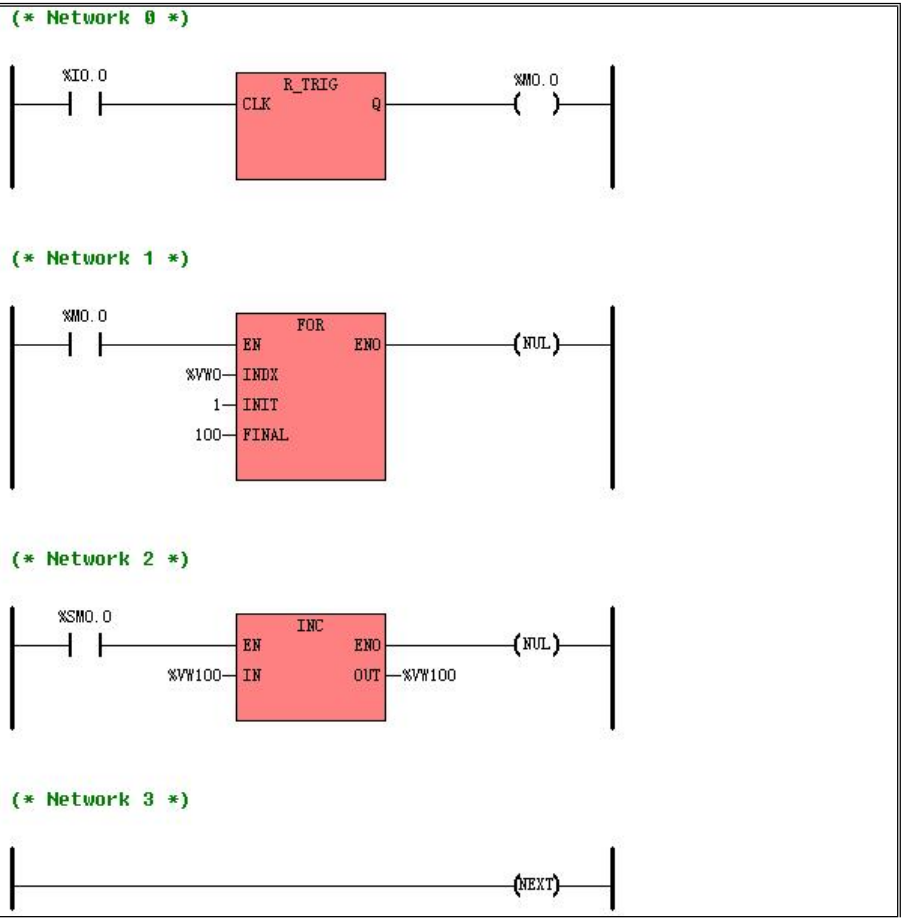
executed; otherwise, the loop will be skipped and the program after the loop structure will continue to be executed downward.

- IL

If the CR value at the FOR instruction is 1, the FOR/NEXT loop is executed, otherwise the loop will be skipped and the program after the loop structure will continue to be executed downward.

➢ Instruction usage example

LD



IL	(* Network 0 *) (*If the rising edge of I0.0 is detected, the following loop body is executed 100 times *) LD %I0.0 R_TRIG ST %M0.0 (* Network 1 *) LD %M0.0 FOR %VW0, 1, 100 (* Network 2 *) LD %SM0.0 INC %VW100 (* Network 3 *) LD TRUE NEXT

6.11.5 END (End the main program)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
LD	END	—(END)—		
IL	END	END	U	

This instruction can only be used in the main program to terminate the execution of the main program.

At the end of the main program, KincoBuilder software automatically calls the END instruction implicitly.

- LD

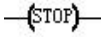
If the value of the energy flow on the left side of the instruction is 1, the instruction is executed, otherwise the instruction will not work, and the program will continue to be executed in sequence.

- IL

If the CR value is 1, the instruction is executed, otherwise the instruction has no effect, and the program continues to be executed in sequence.

6.11.6 STOP (Stop CPU)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
LD	STOP			
IL	STOP	STOP	U	

This instruction immediately transitions the CPU from the run (RUT) state to the stop (STOP) state.

- LD

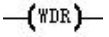
If the value of the energy flow on the left side of the instruction is 1, the instruction is executed, and the CPU is immediately transferred to the STOP state, otherwise the instruction will not work, and the program will continue to be executed in sequence.

- IL

If the CR value is 1, the instruction is executed, and the CPU is immediately transferred to the STOP state, otherwise the instruction does not work, and the program continues to be executed in sequence.

6.11.7 WDR (watchdog reset)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
LD	WDR			
IL	WDR	WDR	U	

This instruction resets the system watchdog and restarts the watchdog timer。

Using the WDR instruction can increase the time allowed for one scan, so that the program that takes a long time can be executed smoothly. However, users need to pay attention that if the program takes too long, the following tasks may not be completed in time.

✧ CPU self-diagnosis

✧ Read input (read the signal of the physical input channel and refresh the input image area)

✧ communication

✧ Write output (here refers to writing the data in the output image area to the physical output channel, excluding the immediate output instruction)

✧ Timing of 10ms and 100ms timers

- LD

If the value of the energy flow on the left side of the instruction is 1, the instruction is executed, otherwise the instruction will not work, and the program will continue to be executed in sequence.

- IL

If the CR value is 1, the instruction is executed, otherwise the instruction has no effect, and the program continues to be executed in sequence.

6.12 interrupt instruction

The purpose of using the interrupt technology is to improve the execution efficiency of the CPU, and to achieve fast response to various internal or external predefined interrupt events. Kinco-K series PLC defines dozens of interrupt events, and each interrupt is assigned a unique event number for CPU identification.

6.12.1 Kinco-K how series handles interrupt events?

The user can call the ENI and DISI instructions in the program to globally enable or disable the CPU to process interrupt events. Kinco-K series allows processing interrupt events globally by default.

If the user needs the CPU to respond to an interrupt event, the ATCH (interrupt connection) instruction must be used in the program to connect the interrupt event (identified with an event number) to an interrupt service routine. In this way, when the interrupt event occurs, the CPU will automatically call the interrupt service routine it is connected to for processing. Once the last instruction in the interrupt service routine is executed, the CPU will return to the breakpoint of the main loop to continue execution. The user can also call the RETC or RETCN instruction in the interrupt service routine to exit the program.

An interrupt event can only correspond to one interrupt service routine, but one interrupt service routine can correspond to multiple interrupt events. Interrupt service routines have no parameters, but allow the use of local variables.

Due to the requirement of fast response, users should optimize the interrupt service routine as much as possible to make it short and lean.

6.12.2 Interrupt priority and queue

Interrupt events have different priorities. When the interrupt event occurs, it will be queued according to the priority and the sequence of occurrence. The interrupt events with the higher priority in the queue are processed first; the interrupts with the same priority are processed according to the sequence of occurrence, and the one that occurs first is processed first.

Only one interrupt service routine is allowed to be executing at any time.

Once an interrupt service routine starts to execute, it will continue to complete without being interrupted by any other interrupt service routine, and any interrupt events that occur during its execution will be queued for processing. 6.12.3 Interrupt event classification

- Communication port interruption

For free protocol communication。

The send complete interrupt occurs when the user-specified data has been sent.

A receive complete interrupt occurs when a user-specified amount of data is received.

- I/O interruption

Including rising edge or falling edge interrupt, high-speed counter interrupt.

The rising edge and falling edge interrupts can only be captured by the 0th to 3rd channels (addresses I0.0---I0.3) of the integrated DI of the CPU body.

High-speed counter interrupts occur when the counter is reset, when the counting direction is changed, or when the count value is equal to a preset value.

- Time interruption

Contains timed interrupts and timer interrupts.

Timing interrupts will be generated periodically, and the cycle unit is 0.1ms, which can be used to complete periodic tasks. Timed interrupts are not affected by the PLC scan cycle and can be used for precise timing. **(This type of interrupt can generate timing that is not affected by the PLC scan cycle)**

The timer interrupt occurs when the current value of T2 and T3 is equal to the preset value, which can be used to respond to the timing event in time. Timer interrupts are affected by the PLC scan cycle. **(This type of interrupt is affected by the PLC scan cycle)**

6.12.4 List of interrupt events

The table below is a complete list of interrupt events for the Kinco-K series. It should be noted that for a specific CPU model, because it does not have certain functions, the corresponding interrupt events are not supported. For example, the CPU504 has only one communication port of PORT0, so the interrupt event

about PORT1 is not supported. Another example is the K5 series CPU only supports HSC0, HSC1, then the interrupt events of HSC2 and HSC3 are not supported.

Event number	Interrupt description	classification
191	The 32nd "CV=PV" when HSC3 uses multi-segment PV values	I/O Interrupt
…	... (add 1 in turn)	
161	The second "CV=PV" when HSC3 uses multi-segment PV values	
160	The first "CV=PV" when HSC3 uses multi-segment PV values	
159	The 32nd "CV=PV" when HSC2 uses multi-segment PV values	
…	... (add 1 in turn)	
129	The second "CV=PV" when HSC2 uses multi-segment PV values	
128	The first "CV=PV" when HSC2 uses multi-segment PV values	
127	The 32nd "CV=PV" when HSC1 uses multi-segment PV values	
…	... (add 1 in turn)	
97	The second "CV=PV" when HSC1 uses multi-segment PV values	
96	The first "CV=PV" when HSC1 uses multi-segment PV values	
95	The 32nd "CV=PV" when HSC0 uses multi-segment PV values	
…	... (add 1 in turn)	
65	The second "CV=PV" when HSC0 uses multi-segment PV values	
64	The first "CV=PV" when HSC0 uses multi-segment PV values	
63-33	reserve	
34	PORT 2: send data completed	Communication interruption
33	PORT 2: Receive data completed	
32	PORT 1: send data completed	
31	PORT 1: Receive data completed	
30	PORT 0: send data completed	
29	PORT 0: Receive data completed	
28--27	reserve	I/O interrupt
26	I0.0 detects falling edge	
25	I0.0 detects rising edge	
24	I0.1 detects falling edge	
23	I0.1 detects rising edge	
22	I0.2 detects falling edge	
21	I0.2 detects rising edge	
20	I0.3 detects falling edge	
19	I0.3 detects rising edge	
18	When HSC0 uses a single PV value, CV=PV (current value=preset value)	
17	HSC0 input direction change	
16	HSC0 external reset	

15	When HSC1 uses a single PV value, CV=PV (current value=preset value)	
14	HSC1 input direction change	
13	HSC1 external reset	
12	When HSC2 uses a single PV value, CV=PV (current value=preset value)	
11—10	reserve	
9	When HSC3 uses a single PV value CV=PV (current value=preset value)	
8—5	reserve	
4	Timed interrupt 1. The period is specified in SMD16, the unit is 0.1ms. **(Not affected by scan cycle)**	Time interrupt
3	Timed interrupt 0. The period is specified in SMD12, the unit is 0.1ms. **(Not affected by scan cycle)**	
2	Timer T3 ET=PT (current value = preset value) **(affected by scan cycle)**	
1	Timer T2 ET=PT (current value=preset value) **(affected by scan cycle)**	

Table 6-1 List of interrupt events of K series PLC

Timing interrupt 1 and timing interrupt 2 with event numbers 3 and 4 are not affected by PLC scan. However, they are affected by hardware interrupts with higher priority, and the timings are mostly long. Users can make compensations according to their own application programs when measuring the speed and flow.

Affected by PLC scan, the timers with event numbers 1 and 2 are interrupted, and 3 and 4 are recommended for precise timing.

6.12.5 ENI (enable interrupt)、DISI (Disable interrupt)instruction

- Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK
LD	ENI	—(ENI)—		
	DISI	—(DISI)—		
IL	ENI	ENI	U	☑ K6

	DISI	DISI		

The ENI and DISI instructions can be executed once in the program. By default, the PLC allows programs to handle interrupt events.

- LD

The role of the ENI instruction is: if the value of the energy flow at the left end of the instruction is 1, the interrupt event is globally allowed to be processed (that is, the CPU is allowed to call the interrupt service routine when an interrupt occurs), otherwise the instruction is not executed.

The function of the DISI instruction is: if the value of the energy flow at the left end of the instruction is 1, the processing of the interrupt event is globally prohibited, otherwise the instruction is not executed.

- IL

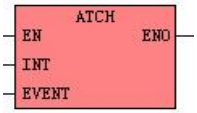
The role of the ENI instruction is: if the CR value is 1, the interrupt event is globally allowed to be processed, otherwise the instruction is not executed.

The function of the DISI instruction is: if the CR value is 1, the processing of the interrupt event is globally prohibited, otherwise the instruction is not executed.

The execution of ENI and DISI instructions does not affect the CR value.

6.12.6 ATCH (disconnect)、DTCH (interrupt separation)instruction

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	ATCH			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
	DTCH			
IL	ATCH	ATCH INT, EVENT	U	

	DTCH	DTCH EVENT		

Parameter	Input/Output	Datatype	Description
INT	Input	identifier	The name of the interrupt service routine that already exists in the user project
EVENT	Input	INT type constant	interrupt event number

The function of the TCH instruction: Connect the interrupt event specified by EVENT with the interrupt service routine specified by INT. In this way, if the CPU allows to process the interrupt event, the interrupt service routine will be automatically called when the interrupt event occurs. Multiple interrupt events can be connected to one interrupt service routine, but one interrupt event is not allowed to connect to multiple interrupt service routines.

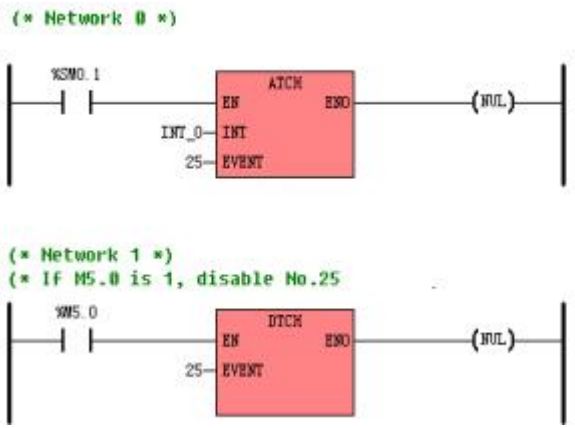
The function implemented by the DTCH instruction is to cancel the connection between the interrupt event specified by the parameter EVENT and all interrupt service routines.

- LD

  If the value of EN is 1, the ATCH and DTCH instructions are executed, otherwise they are not executed.
- IL

If the CR value is 1, the ATCH and DTCH instructions are executed, otherwise they are not executed. Their execution does not affect the CR value.

- Instruction usage example

<table>
<tr><td>LD</td><td colspan="2"> 
</td></tr>
<tr><td>IL</td><td>(* NETWORK 0 *)
LD %M5.0
ENI

LDN %M5.0
DISI

LD %SM0.1
ATCH INT_0, 25

LD %I1.0
DTCH 25</td><td>(* Create CR with a value of M5.0 *)
(*If the CR value is 1, interrupt processing is allowed *)

(* Create CR with a value of M5.0 reversed *)
(*If the CR value is 0, interrupt processing is disabled *)

(*On the first scan, connect interrupt 25 with the INT_0 interrupt service routine *)

(*Create CR with value I1.0*)
(*If the CR value is 1, cancel the connection between the 25th interrupt and all interrupt service routines *)</td></tr>
</table>

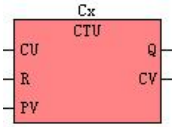
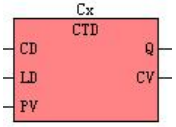
6.13 counter

The counter is one of the function blocks defined in the IEC61131-3 standard, and there are three types: CTU, CTD and CTUD.

For the use of function blocks and their instances, please refer to the description in 3.6.5 About Function Blocks and Function Block Instances. Note that the counter instance is not allowed to be reused, and it is not allowed to assign the same counter instance to multiple counter instructions in the user project.

6.13.1 CTU (count up)、CTD (count down)

- Instruction and its operand description

	Name	Instracition format	Affect CR value	
LD	CTU			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
	CTD			
IL	CTU	CTU Cx, R, PV	P	
	CTD	CTD Cx, LD, PV		

Parameter	Input/Output	Data type	Allowed memory area
Cx	-	Counter instance (x represents the number)	C
CU	Input	BOOL	energy flow
R	Input	BOOL	I、Q、M、V、L、SM、T、C
CD	Input	BOOL	energy flow
LD	Input	BOOL	I、Q、M、V、L、SM、T、C
Q	Output	BOOL	energy flow
CV	Output	INT	Q、M、V、L、SM、AQ

- LD

The CTU instruction increments the rising edge of the CU input (by 1 each time) and assigns the count value of Cx to CV. Cx will count until it reaches and stays at the maximum value of 32767. When CV is greater than or equal to the preset value PV, the state values of Q and Cx are both set to 1, otherwise they are both set to 0. If R input is 1, Cx is reset, CV and its count value are all cleared.

The CTD instruction counts down the falling edges of the CD input (by 1 each time) and assigns the count value of Cx to CV. When CV is equal to 0, Cx stops counting, and the state values of Q and Cx are both set to 1, otherwise they are all set to 0. If the LD input is 1, Cx is reset, and PV is reassigned to the count value and CV of Cx.

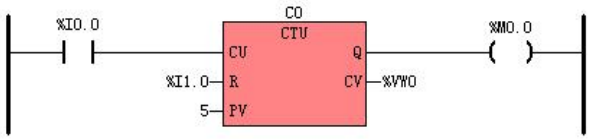
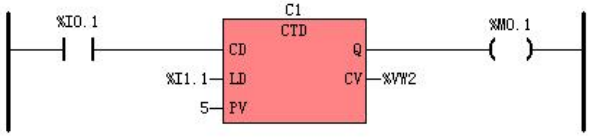
- IL

The CTU instruction increments the rising edge of the CR value (by 1 each time). Cx will count until it reaches and stays at the maximum value of 32767. When the count value is greater than or equal to the preset value PV, the state value of Cx is set to 1, otherwise it is set to 0. If the R input is 1, Cx is reset and its count value is cleared.

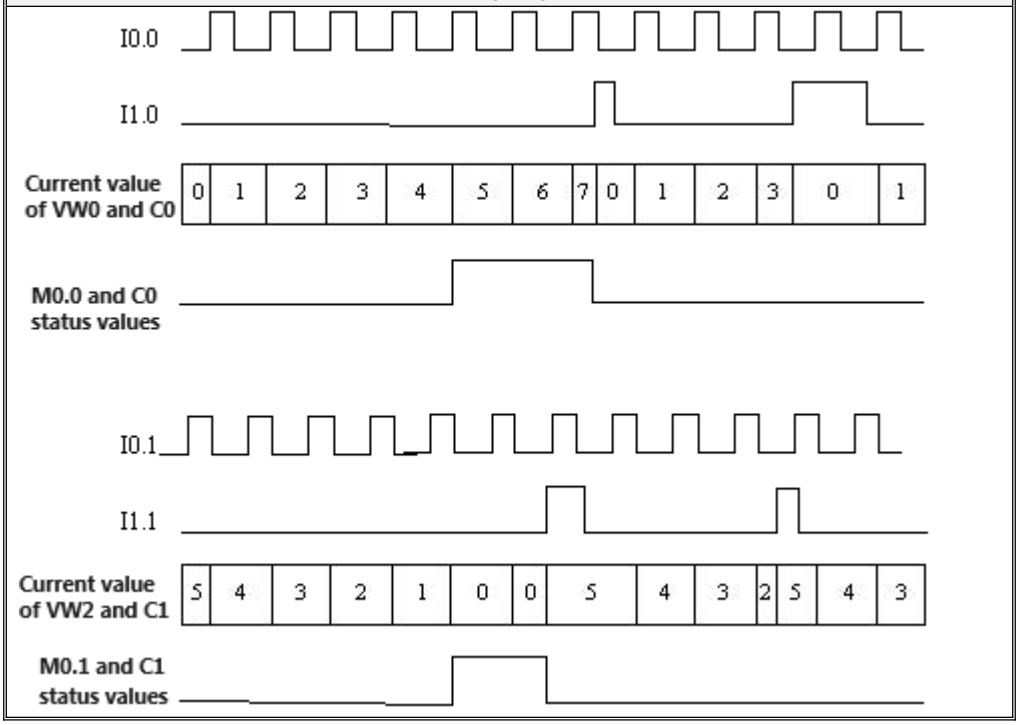
The CTD instruction counts down the falling edge of the CR value (decrement by 1 each time). When the count value is equal to 0, Cx stops counting and its status value is set to 1, otherwise it is set to 0. If the LD input is 1, Cx is reset and the PV is reassigned to the count value of Cx.

After each scan of CTU and CTD instructions, the CR value is updated to the state value of Cx.

➢ CTU、CTD usage example

LD	IL
(* Network 0: *)  (* Network 1: *)  	(* NETWORK 0 *) LD %I0.0 CTU C0, %I1.0, 5 ST %M0.0 (* NETWORK 1 *) LD %I0.1 CTD C1, %I1.1, 5 ST %M0.1

Timing diagram



6.13.2 CTUD (count up/down)

- Instruction and its operand description

	Name	Instracition format	Affect CR value	
LD	CTUD	Cx CTUD CU QU CD QD R CV LD PV		☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
IL	CTUD	CTUDCx, CD, R, LD, PV, QD	P	

Parameter	Input/Output	Data type	Allowed memory area

Cx	-	Counter instance (x represents the number)	C
CU	Input	BOOL	energy flow
CD	Input	BOOL	I、Q、M、V、L、SM、T、C、RS、SR
R	Input	BOOL	I、Q、M、V、L、SM、T、C、RS、SR
LD	Input	BOOL	I、Q、M、V、L、SM、T、C、RS、SR
PV	Input	INT	I、Q、M、V、L、SM、AI、AQ、constant
QU	Output	BOOL	energy flow
QD	Output	BOOL	Q、M、V、L、SM
CV	Output	INT	Q、M、V、L、SM、AQ

- LD

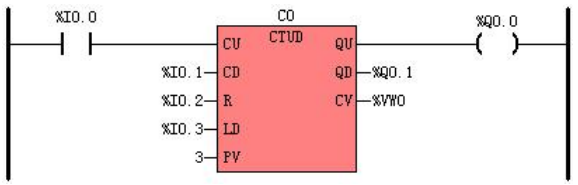
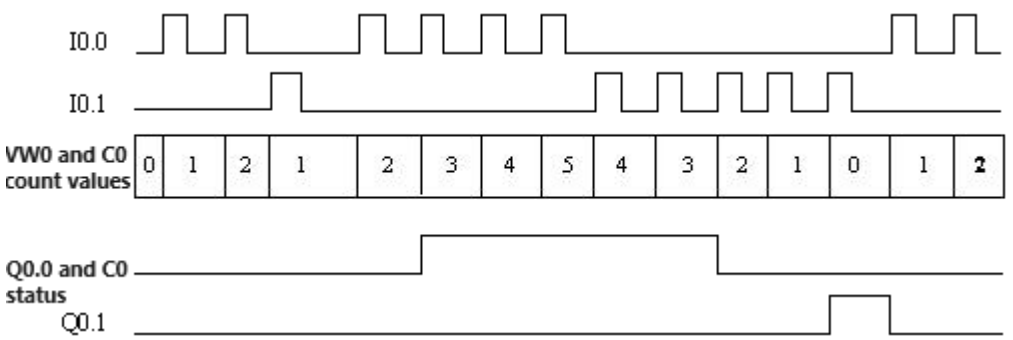
The CTUD instruction counts up the rising edge of the CU input (by 1 each time) and counts down (by 1 each time) on the rising edge of the CD input, and the count value of Cx is assigned to CV. When CV is greater than or equal to the preset value PV, the state values of QU and Cx are both set to 1, otherwise they are both set to 0. When CV is equal to 0, QD is set to 1, otherwise it is set to 0. If the R input is 1, Cx is reset, its count value and CV are cleared. If the LD input is 1, reassign PV to the count value of Cx and CV. When both R and LD are input as 1, R takes precedence.

- IL

The CTUD instruction counts up (by 1 each time) rising edges of the CR value and decrements (by 1 each time) on the rising edges of the CD input. When the count value is greater than or equal to the preset value PV, the state value of Cx is set to 1, otherwise it is set to 0. When the count value is equal to 0, the QD is set to 1, otherwise it is set to 0. If the R input is 1, Cx is reset and its count value is cleared. If the LD input is 1, the PV is reassigned to the count value of Cx. When both R and LD are input as 1, R takes precedence.

After each scan of the CTUD instruction, the CR value is updated to the state value of Cx.。

➢ CTUD usage example

LD	IL
	LD %I0.0 CTUD C1,%I0.1,%I0.2,%I0.3, 4 ,%Q0.1 ST %Q0.0
Timing diagram	
	

6.14 Timer

The timer is one of the functional blocks defined in the IEC61131-3 standard, and there are three types: TON, TOF and TP.

For the use of function blocks and their instances, please refer to section 3.6.5 About Function Blocks and Function Block Instances.

6.14.1 timer's time base

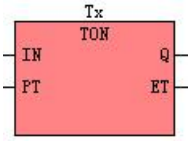
KPLC provides timers with three time bases. The timer number determines its time base. The time base of T0-T3 is 1ms, the time base of T4-T19 is 10ms, and the time of T20-T255 is 100ms.

The maximum timing time of the timer is 32767×time base. The preset value and timing value of the timer are multiples of its time base. For example, for a timer with a 10ms time base, 100 means 1000ms.

The PLC updates the ET value of the timer only when the timer instruction is being executed, so it will be affected by the scan cycle. For precise timing, use a timed interrupt.

6.14.2 TON (On-delay timer)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	TON			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	TON	TON Tx, PT	P	

Parameter	Input/Output	Data type	Allowed memory area
Tx	-	Timer instance	T
IN	Input	BOOL	energy flow
PT	Input	INT	I、AI、AQ、M、V、L、SM、constant
Q	Output	BOOL	energy flow
ET	Output	INT	Q、M、V、L、SM、AQ

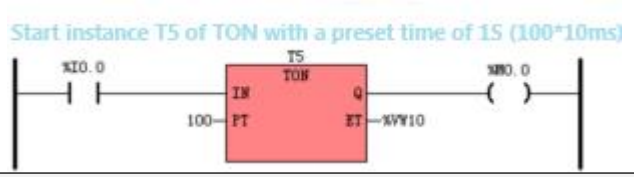
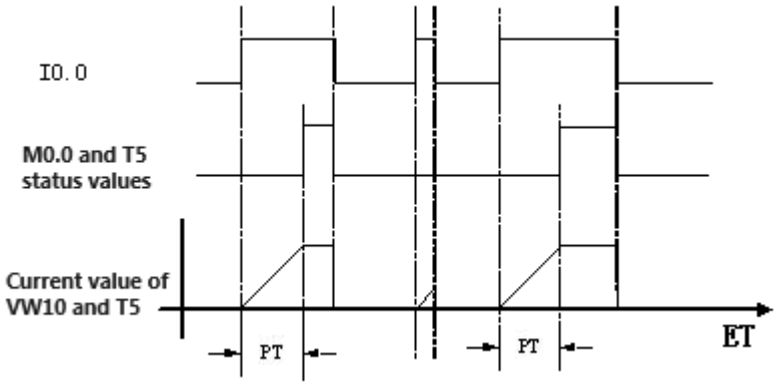
- LD

If the rising edge of the input terminal IN is detected, the Tx starts to start the timing. When the timing value ET is greater than or equal to the preset value PT, the Tx stops, and its output Q and its state value are both set to 1. If the input IN becomes 0, Tx is reset, its output Q and state value are both set to 0, and the timing value ET is also cleared.

- IL

If the rising edge of the CR value is detected, the Tx starts to start the timing. When the timing value is greater than or equal to the preset value PT, the Tx stops, and its state value is set to 1. If the CR value becomes 0, Tx is reset, and its state value and timing value are cleared to zero. After each scan of TON, the CR value is set to the state value of Tx.

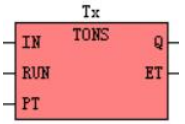
- TON usage example

LD	IL
	(* NETWORK 0 *) LD %I0.0 TON T5, 100 ST %M0.0
Timing diagram	
	

6.14.3 TONS (On-delay cumulative timer)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	☑ K2

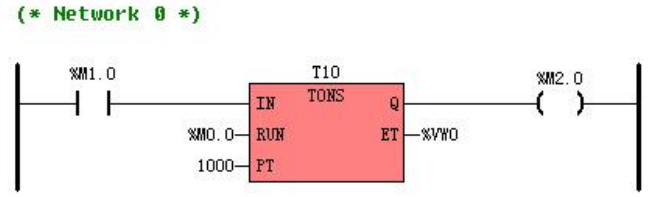
LD	TONS			☑ KS ☑ KW ☑ MK ☑ K6
IL	TONS	TONS Tx, RUN, PT	P	

Parameter	Input/Output	Data type	Allowed memory area
Tx	-	Timer instance	T
IN	Input	BOOL	energy flow
RUN	Input	BOOL	I、Q、M、V、L、SM
PT	Input	INT	I、AI、AQ、M、V、L、SM、constant
Q	Output	BOOL	energy flow
ET	Output	INT	Q、M、V、L、SM、AQ

- LD

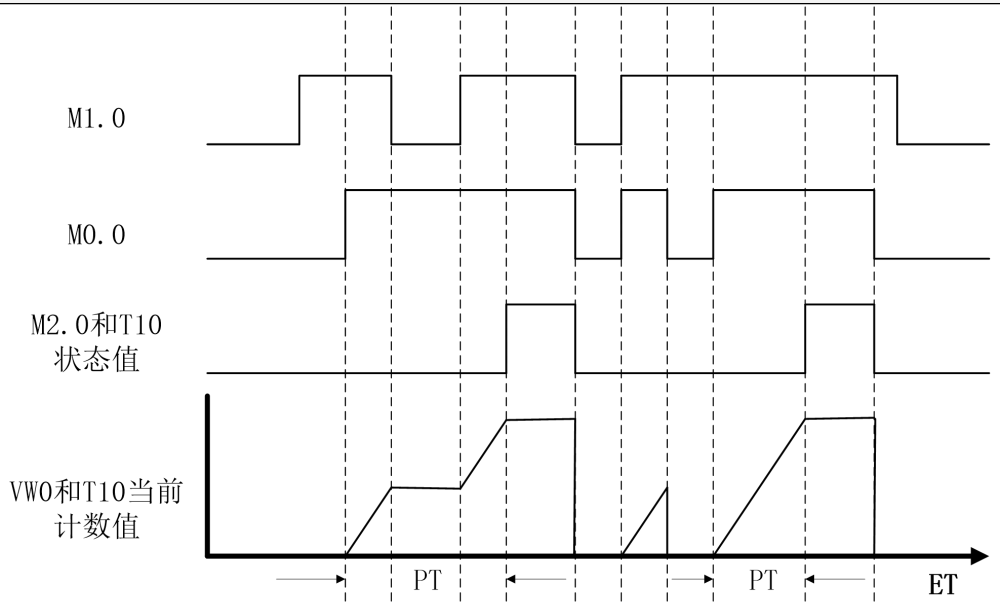
If it is detected that both the input terminals IN and RUN become 1, Tx starts to start the timing. When the timing value ET is greater than or equal to the preset value PT, Tx stops, and its output Q and its state value are both set to 1. If the input IN becomes 0, then Tx is suspended for timing, and its output Q, state value, and timing value ET will not change. If the input RUN becomes 0, its output Q and state value are both set to 0, and the timing value ET is also cleared to zero.

➢ TONS usage example

LD	Description
	When M1.0 and M0.0 are 1 at the same time, T10 starts timing, and the preset time is 10s. When M1.0 becomes 0, the timer suspends timing, and the ET value remains unchanged; when M1.0 becomes 1 again, the timer continues to count, and ET continues to accumulate. When the count value is greater than

	or equal to the preset time, both output Q and T10 are set to 1. When M0.0 is 0, its output Q and state value are both set to 0, and the timing value ET is also cleared.

Timing diagram



6.14.4 TOF (off-delay timer)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	☑ K5

LD	TOF			☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	TOF	TOF Tx, PT	P	

Parameter	Input/Output	Data type	Allowed memory area
Tx	-	Timer instance	T
IN	Input	BOOL	energy flow
PT	Input	INT	I、AI、AQ、M、V、L、SM、constant
Q	Output	BOOL	energy flow
ET	Output	INT	Q、M、V、L、SM、AQ

- LD

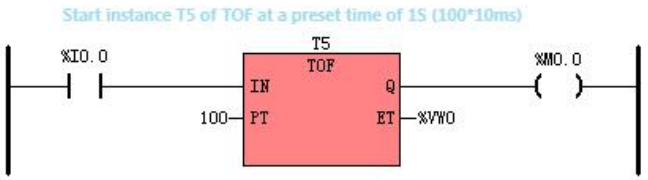
If the falling edge of the input terminal IN is detected, Tx starts to start timing, when the timing value ET is greater than or equal to the preset value PT, Tx stops, and its output Q and its state value are both set to 0. If the input IN becomes 1, Tx is reset, its output Q and state value are both set to 1, and the timing value ET is cleared at the same time.

- IL

If the falling edge of the CR value is detected, the Tx starts to start the timing. When the timing value is greater than or equal to the preset value PT, the Tx stops, and its state value is set to 0. If the CR value becomes 1, the Tx is reset, its state value is set to 1, and the timer value is cleared. After each scan of TOF, the CR value is set to the state value of Tx.

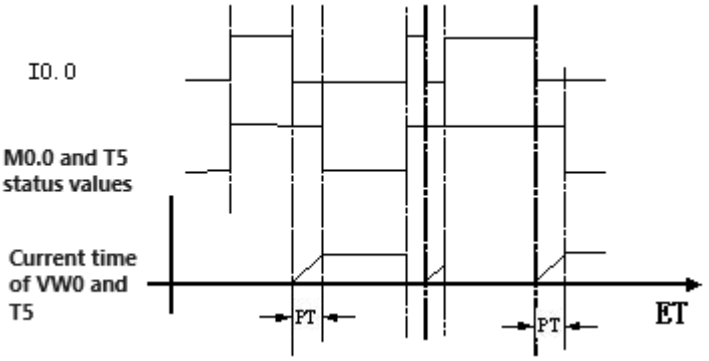
➢ TOF usage example

LD	IL



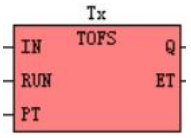

```
(* NETWORK 0 *)
LD    %I0.0
TOF   T5, 100
ST    %M0.0
```

Timing diagram



6.14.5 TOFS (Off-delay cumulative timer)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	TOFS			☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	TOFS	TOFS Tx, RUN, PT	P	

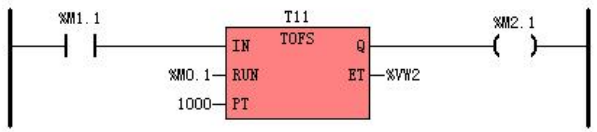
Parameter	Input/Output	Data type	Allowed memory area

Tx	-	Timer instance	T
IN	Input	BOOL	energy flow
RUN	Input	BOOL	I、Q、M、V、L、SM
PT	Input	INT	I、AI、AQ、M、V、L、SM、constant
Q	Output	BOOL	energy flow
ET	Output	INT	Q、M、V、L、SM、AQ

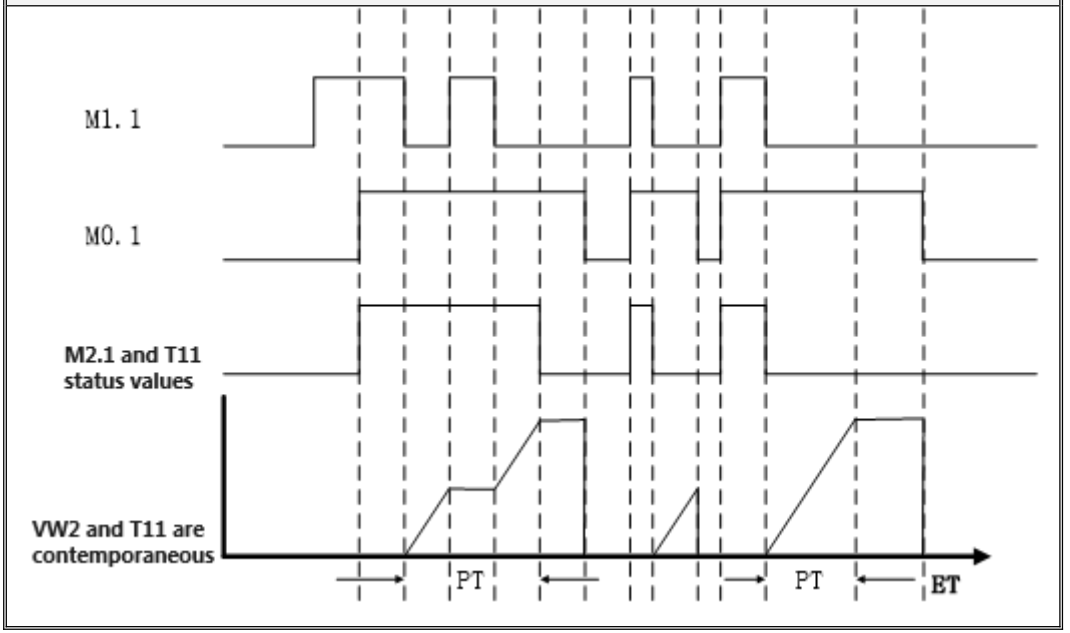
- LD

After detecting that the input terminals IN and RUN are both changed to 1, the output Q and its state value are both set to 1. When the input terminal IN becomes 0 again, Tx starts to start the timing. When the timing value ET is greater than or equal to the preset value PT, the Tx stops, and its output Q and its state value are both set to 0. If the input IN counts to 1 in the middle of the count, the Tx is suspended from timing, and the output Q, state value, and timing value ET will not change. If the input RUN becomes 0, its output Q and state values are both set to 0, and the timing value ET is also cleared to zero.

➢ TOFS usage example

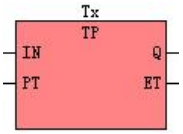
LD	Description
	When M1.1 and M0.1 are 1 at the same time, the output Q and T11 are both set to 1. When M1.1 becomes 0, the timing starts, and the preset time is 10s. When M1.1 becomes 1, the timer stops timing. When M1.1 becomes 1 again, it continues to count the last accumulated time. When the count value is greater than or equal to the preset time, the output Q and T11 are both set to 0. When M0.1 is 0, its output Q and state value All are set to 0, and the timing value ET is also cleared to zero.

Timing diagram



6.14.6 TP (Pulse timer)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	TP			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	TP	TP Tx, PT	P	

Parameter	Input/Output	Data type	Allowed memory area
Tx	-	Timer instance	T

IN	Input	BOOL	energy flow
PT	Input	INT	I、AI、AQ、M、V、L、SM、constant
Q	Output	BOOL	energy flow
ET	Output	INT	Q、M、V、L、SM、AQ

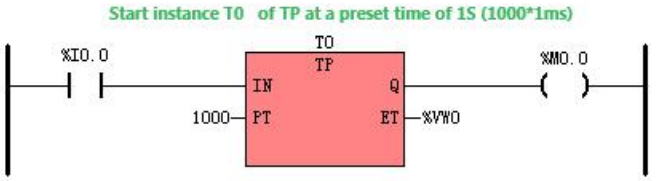
- LD

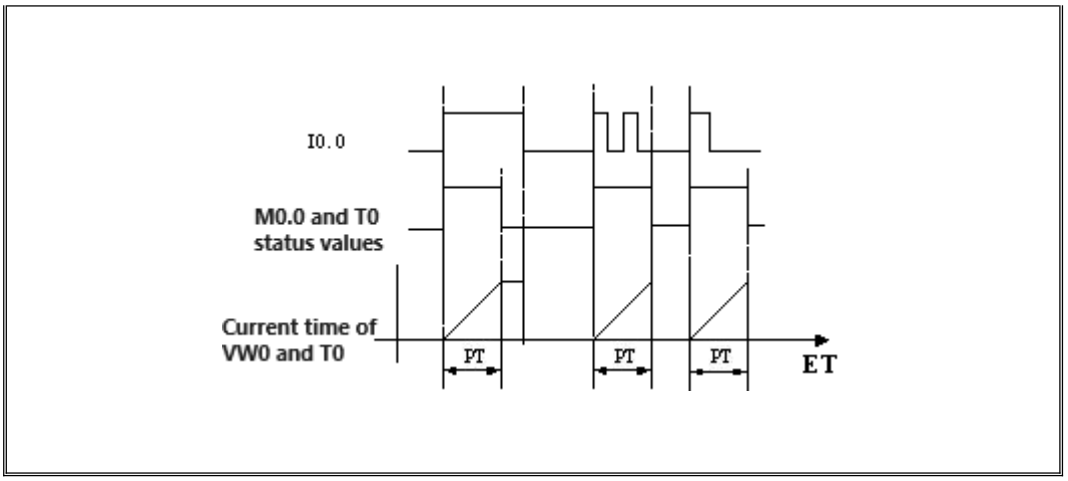
If the rising edge at the input terminal IN is detected, Tx starts to start timing, and outputs a pulse of constant width at its output terminal Q and its state value, and the pulse width value is the preset time PT. The timing value of Tx is stored in the parameter ET.

- IL

If the rising edge of the CR value is detected, the Tx starts to start timing, and its state value outputs a pulse with a constant width, and the pulse width value is the preset time PT. After each scan of TP, the CR value is set to the state value of Tx.

➢ TP usage example

LD	IL
	(* NETWORK 0 *) LD %I0.0 TP T0, 1000 ST %M0.0
Timing diagram	



6.14.7 TP_R (Resettable Pulse Timer)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	TP_R	Tx TP_R IN Q R ET PT		☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	TP_R	TP_R Tx, R, PT	P	

Parameter	Input/Output	Data type	Allowed memory area
Tx	-	Timer instance	T
IN	Input	BOOL	energy flow
R	Input	BOOL	I、Q、M、V、L、SM
PT	Input	INT	I、AI、AQ、M、V、L、SM、constant
Q	Output	BOOL	energy flow
ET	Output	INT	Q、M、V、L、SM、AQ

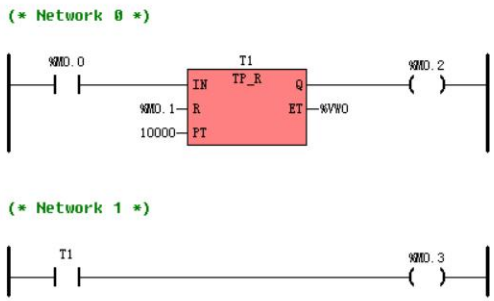
- LD

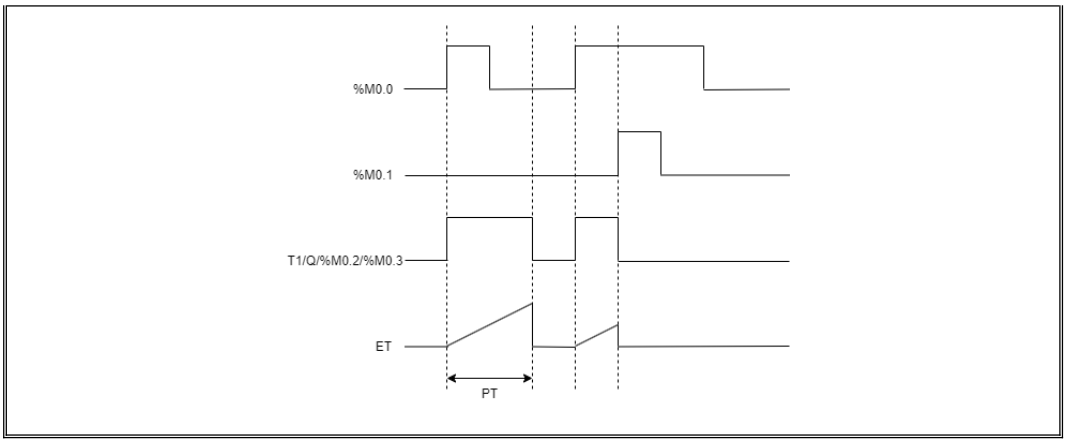
Input R is used to reset the timer. If R is 0, when the rising edge at the input terminal IN is detected, Tx starts to start timing, and the output terminal Q and its state value will output a pulse of constant width, and the pulse width value is the preset time PT. The timing value of Tx is stored in the parameter ET. If R is 1, Tx is reset, the output terminal Q and its state value will become 0, and the timing value ET will also be cleared.

- IL

Input R is used to reset the timer. If R is 0, when the rising edge of the CR value is detected, Tx starts to start timing, and its state value will output a pulse of constant width, and the pulse width value is the preset time PT. If R is 1, Tx is reset, its state value will become 0, and the timing value will also be cleared. After the PLC scans TP each time, the CR value is set as the state value of Tx.

➢ TP_R usage example

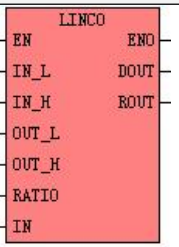
<table>
<tr><th>LD</th><th>IL</th></tr>
<tr><td> 
</td><td>(* Network 0 *)
LD %M0.0
TP_R T1, %M0.1, 10000
__CR_EQ_1
MOVE T1, %VW0
__CR_RESTORE
ST %M0.2
(* Network 1 *)
LD T1
ST %M0.3</td></tr>
<tr><td colspan="2">Timing diagram</td></tr>
</table>



6.15 Additional instructions

6.15.1 LINCO (Linear transformation)

- Instruction and its operand description

	Name	Instraction format	Affect CR value	
LD	LINCO			☑ K5 ☑ K2 ☑ KS ☐ KW ☑ MK ☑ K6
IL	LINCO	LINCO IN_L, IN_H, OUT_L, OUT_H, RATIO, IN, DOUT, ROUT	U	

Parameter	Input/Output	Data type	Allowed memory area
IN_L	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、constant
IN_H	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、constant

OUT_L	Input	REAL	V、L、constant
OUT_H	Input	REAL	V、L、constant
RATIO	Input	REAL	constant
IN	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ
DOUT	Output	DINT	Q、M、V、L、SM
ROUT	Input	REAL	V、L

Note: Parameters IN_L, IN_H, OUT_L and OUT_H must be all constants or all variables.

The LINCO instruction calculates the input IN according to the specified linear relationship, multiplies the calculation result by the coefficient RATIO and assigns it to ROUT, and then rounds ROUT (discards decimals) and assigns it to DOUT. The linear relationship used is as follows: input lower limit IN_L and output lower limit OUT_L, input upper limit IN_H and output upper limit OUT_H, these parameters are calculated according to the method of "two points set a straight line" to obtain the linear relationship.

The role of the Linco instruction can be described by the following formula::

ROUT ＝ RATIO×(k×IN + b)

DOUT ＝ TRUNC(ROUT)

Where $k = \frac{OUT\_H - OUT\_L}{IN\_H - IN\_L}$ ； $b = OUT\_L - k \times IN\_L$ 。

- LD

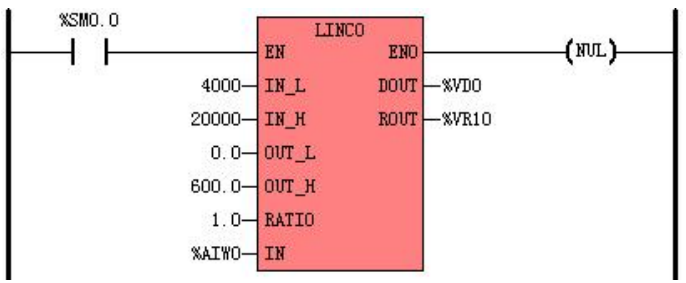
  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

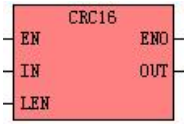
➢ Instruction usage example

Assume that the measurement range of a temperature transmitter on site is 0∽600℃, and the signal output range is 4∽20mA. The output signal of the transmitter is connected to the AIW0 channel of the PLC. The PLC is required to calculate the actual temperature value.

LD	
IL	LD %SM0.0 LINCO 4000, 20000, 0.0, 600.0, 1.0, %AIW0, %VD0, %VR10

6.15.2 CRC16 (16-bit CRC check code)

➢ Instruction and its operand description

	Name	Instraction format	Affect CR value	Suitable for
LD	CRC16			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	CRC16	CRC16 IN, OUT, LEN	U	

Parameter	Input/Output	Data type	Allowed memory area
IN	Input	BYTE	I、Q、M、V、L、SM
LEN	Input	BYTE	I、Q、M、V、L、SM、constant
OUT	Output	BYTE	Q、M、V、L、SM

This instruction is used to calculate the 16-bit CRC check code of the specified data. The data to be checked is stored in an area of consecutive LEN bytes from IN. The calculation result is stored in a

continuous 2-byte area from OUT, where OUT stores the high byte of the CRC code, and the next byte of OUT stores the low byte of the CRC code.

Note that the memory area used cannot exceed the valid memory range, otherwise the execution result will be unpredictable.

- LD

  If the EN value is 1, the instruction is executed, otherwise it is not executed.

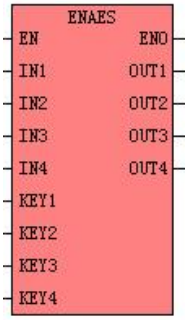
- IL

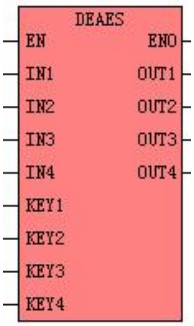
  If the CR value is 1, the instruction is executed, otherwise it is not executed.

  The execution of this instruction does not affect the CR value.

6.15.3 ENAES (AES-128 encryption) DEAES (AES-128decryption)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	Suitable for
LD	ENAES			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6

	DEAES			
	ENAES	ENAES IN1,IN2,IN3,IN4,KEY1,KEY2,KEY3, KEY4,OUT1,OUT2,OUT3,OUT4	U	
	DEAES	DEAES IN1,IN2,IN3,IN4,KEY1,KEY2,KEY3, KEY4,OUT1,OUT2,OUT3,OUT4		

Parameter	Input/Output	Data type	Allowed memory area
IN1	Input	DWORD	I、Q、L、M、V、SM、constant
IN2	Input	DWORD	I、Q、L、M、V、SM、constant
IN3	Input	DWORD	I、Q、L、M、V、SM、constant
IN4	Input	DWORD	I、Q、L、M、V、SM、constant
KEY1	Input	DWORD	I、Q、L、M、V、SM、constant
KEY2	Input	DWORD	I、Q、L、M、V、SM、constant
KEY3	Input	DWORD	I、Q、L、M、V、SM、constant
KEY4	Input	DWORD	I、Q、L、M、V、SM、constant
OUT1	Output	DWORD	Q、SM、L、M、V
OUT2	Output	DWORD	Q、SM、L、M、V
OUT3	Output	DWORD	Q、SM、L、M、V
OUT4	Output	DWORD	Q、SM、L、M、V

IN1, IN2, IN3, IN4, KEY1, KEY2, KEY3, KEY4 must be both constant types or both memory types。

ENAES and DEAES instructions use AES-128 encryption and decryption instructions.。

The parameters IN1, IN2, IN3, and IN4 are the data to be encrypted/decrypt, KEY1, KEY2, KEY3, and KEY4 are the keys specified by the user, and OUT1, OUT2, OUT3, and OUT4 are the encrypted/decrypt data.

- LD

  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

6.15.4 Special data area read and write instructions

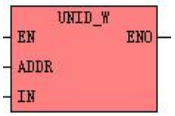
K series PLC provides a special data area with a length of 128 bytes in the permanent memory, and is divided into 32 independent blocks in units of 4 bytes, and users can read and write any block.

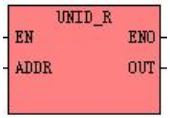
The data in this area has been cleared when the PLC leaves the factory, **and the user can only write non-0 data to each block once. The PLC will lock the block with non-zero data, not allow to write again, but can read arbitrarily**. Note: When writing, the 4-byte data to be written cannot be all 0, otherwise the writing will be ignored.

Execute the menu instruction【PLC】->【Clear...】in KincoBuilder to clear all data in PLC, including user project, data in special area, etc. **This special data area can only be cleared with the clear instruction and cannot be cleared individually.**

Device manufacturers can use this area to write data such as S/N in the PLC.

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	Suitable for
LD	UNID_W			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK

	UNID_R			☑ K6
	UNID_W	UNID_W ADDR, IN	U	
	UNID_R	UNID_W ADDR, OUT		

Parameter	Input/Output	Data type	Allowed memory area
ADDR	Input	INT	I、Q、V、L、M、T、C、SM、AI、AQ、constant
OUT	Output	DWORD	Q、L、M、V、SM
IN	Input	DWORD	I、Q、L、M、V、SM、constant

UNID_W is used to write the data specified by parameter IN into a block in the special data area. The parameter ADDR specifies the block number to be written, and the range is 0-31.

UNID_R is used to read the data of a block in the special data area and store it in the parameter OUT. The parameter ADDR specifies the block number to be read, the range is 0-31.

- LD

  If EN is 1, the instruction is executed.

- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

Chapter 7 Introduction to Real Time Clock

7.1 Real Time Clock

Kinco-K series PLC CPUs are integrated with real-time clock (RTC), which can provide real-time time/calendar representation (CPU504 does not support real-time clock). Real-time clock/calendar seconds to years are encoded in BCD format with automatic leap year adjustments. In the event of a power failure, the real-time clock is powered by a backup capacitor. Under normal temperature, the backup time is not less than 3 years.

Kinco-K series PLC allows the use of lithium batteries of specific specifications as backup batteries. When power is lost, the backup battery is used to power the real-time clock to keep the clock running, and also power the RAM for data retention. At room temperature, the typical life of the battery is 5 years, and the accumulative power retention time is not less than 3 years.

he backup battery can be removed. The upper part of the side of the module is the position of the battery box, and the battery is installed in the battery box. When the backup battery is exhausted, the user can open the battery box to replace a new battery by himself. The battery is a CR2032 3V lithium battery with a connector, the shape is as shown in the figure below, users can order the battery separately.

If the battery is depleted, the real-time clock cannot be maintained after the PLC is powered off. KPLC provides the following alarm registers for battery power, which the user can use in the program.

SM0.7	Read only. SM0.7 is TRUE, the battery voltage is low, SM0.7 is FALSE, the battery voltage is normal.
SMW10	Read only. Stores the voltage value of the backup battery, unit: 0.01V. (for K5 only) If the power supply of the backup battery continues to be lower than 2.6V, the PLC will generate an alarm of "low backup battery charge".

Note: The backup battery of the MK series all-in-one computer supplies power to the RTC in the HMI and PLC parts to maintain the clock operation, and also supplies power to the RAM for data

retention. The real-time clock (RTC) on the screen can also provide real-time time/calendar representation. The user can modify the time in the system setting screen or modify the time through special registers LW10000-LW10006. HMI clock setting method refer to Kinco DTOOLS manual chapter 2.7 clock setting (2.7.3 system time and PLC CPU time synchronization).

The running error of the real-time clock is about 3-5 minutes/month, so if users need to use accurate time, they need to set the clock to the standard time regularly.

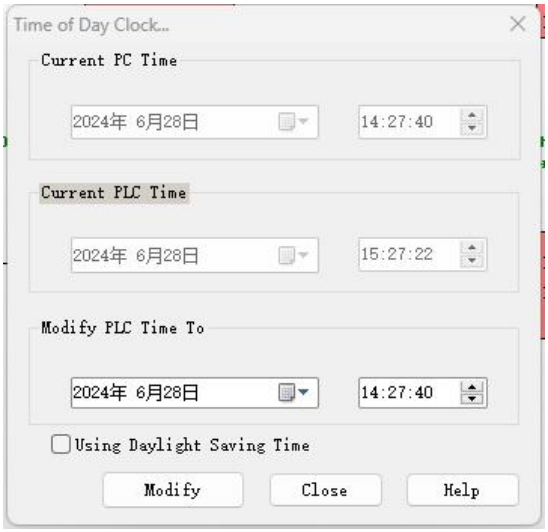
There are two ways for the PLC to adjust the real-time clock:

1. Adjust the real-time clock of the CPU through KincoBuilder software. 2. Use the instructions to read and write real-time clock (READ_RTC, SET_RTC, RTC_W, RTC_R) to realize real-time clock adjustment and related control applications.

7.1.1 Adjust CPU clock

The real-time clock must be adjusted at least once before it is officially used to adjust its time to the current actual time. The time value within the PLC may be a random value before adjustment.

Execute the menu instruction 【PLC】→【Adjust CPU Clock...】to enter the "Adjust CPU Clock" dialog box, as shown in the figure below.

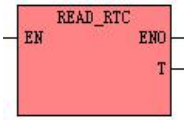
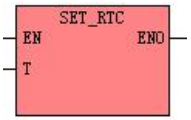


- **System current time**: Display the current date and time of the PC used for programming.
- **PLC current time**: display the current time of the real-time clock in the connected CPU. If the background color is green, it means that the real-time clock has been successfully read from the CPU. If the background color is yellow, it means that reading the real-time clock from the CPU failed.
- **Adjust the PLC time to**: the user can input the new time value of the CPU real-time clock to be set here. The user can either directly input the keyboard in the input box, or use the left mouse button to click the arrow on the right end of the input box to select and adjust.
- **Use daylight saving time**: In countries or regions that use daylight saving time, you need to select this option.
- Click the【**Modify PLC Clock**】button,and the new time value input by the user will be written into the real-time clock in the CPU.

  Note: After changing the time zone or daylight saving time, kincoBuilder must be restarted (Windows system required).

7.1.2 READ_RTC (read real time clock)、SET_RTC (Set real time clock)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	READ_RTC			☑ K5 (CPU504 except) ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
	SET_RTC			
IL	READ_RTC	READ_RTC *T*	U	
	SET_RTC	SET_RTC *T*		

Parameter	Input/Output	Data type	Allowed memory area
T	Input (SET_RTC)	BYTE	V
	Output (READ_RTC)		

Note that the T parameter is a variable-length block memory parameter, and the entire block memory cannot fall into the illegal memory area, otherwise the result will be unpredictable.

The READ_RTC instruction is used to read the current date and time from the hardware clock and put it into the time buffer.

The SET_RTC instruction is used to write the date and time specified by the time buffer to the hardware clock.

The parameter T defines the start address of the time buffer, which occupies a continuous 8-byte memory space. All values in the buffer are encoded in BCD format. Note that the buffer cannot exceed the valid memory range, ot

The data storage form in the time buffer is shown in the following table.

Memory bytes	Meaning of data	Remarks
T	Week	Range: 1~7, where 1 represents Monday and 7 represents Sunday.
T+1	second	Range: 0~59
T+2	Minute	Range: 0~59
T+3	Time	Range: 0~23
T+4	day	Range: 1~31
T+5	moon	Range: 1~12
T+6	year	Range: 0~99
T+7	century	fixed at 20

Table 6-2 Time buffers for real-time clock instructions

Attention:

(1) You can also read and write the CPU clock by executing the menu instruction 【PLC】→【Adjust CPU Clock...】in KincoBuilder. It is recommended that the user use the menu instruction to set the correct clock for the CPU first before using the real-time clock.

(2) The CPU does not check whether the date and time entered by the user are valid, and invalid dates (such

as February 30) will also be accepted by the CPU, so the user must ensure that the valid date and time are entered when setting the real-time clock.

- LD

  If the EN value is 1, execute the READ_RTC, SET_RTC instructions, otherwise do not execute.

- IL

  If the CR value is 1, the READ_RTC and SET_RTC instructions are executed, otherwise it is not executed.

  READ_RTC、SET_RTC The execution of the instruction does not affect the CR value.

7.1.3 RTC_R (read real time clock)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	RTC_R			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	RTC_R	RTC_R*FMT, WEEK, SECOND, MINUTE, HOUR, DAY, MONTH, YEAR, CENTURY*	U	

Parameter	Input/Output	Data type	Allowed memory area
FMT	Input	BYTE	L、M、V、constant

WEEK	Output	BYTE	L、M、V
SECOND	Output	BYTE	L、M、V
MINUTE	Output	BYTE	L、M、V
HOUR	Output	BYTE	L、M、V
DAY	Output	BYTE	L、M、V
MONTH	Output	BYTE	L、M、V
YEAR	Output	BYTE	L、M、V
CENTURY	Output	BYTE	L、M、V

The following table details the effect of each parameter.

Parameter	Description
EN	enable terminal.
FMT	Output format, 0 means decimal, 1 means BCD code.
WEEK	Week, range: 1~7, where 1 represents Monday and 7 represents Sunday.
SECOND	seconds, range: 0~59
MINUTE	minutes, range: 0~59
HOUR	Hour, range: 0~23
DAY	day, range: 1~31
MONTH	month, range: 1~12
YEAR	Year, range: 0~99
CENTURY	century, fixed at 20

RTC_R is used to read the current date and time value from the real time clock and write the various output parameters.

FMT specifies the format of each parameter. If FMT is 0, it is decimal; if FMT is 1, it is BCD code.

- LD

  If the EN value is 1, the RTC_R instruction is executed, otherwise it is not executed.

- IL

  If the CR value is 1, the RTC_R instruction is executed, otherwise it is not executed.

  RTC_R The execution of the instruction does not affect the CR value.

7.1.4 RTC_W (write real time clock)

- Instruction and its operand description

	Name	Instruction format	Affect CR value	
LD	RTC_W			☑ K5 (Excluding CPU504) ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
IL	RTC_W	RTC_W *FMT, WEEK, SECOND, MINUTE, HOUR, DAY, MONTH, YEAR, CENTURY*	U	

Parameter	Input/Output	Data type	Allowed memory area
FMT	Input	BYTE	L、M、V、constant
WEEK	Input	BYTE	L、M、V 、constant
SECOND	Input	BYTE	L、M、V、constant
MINUTE	Input	BYTE	L、M、V、constant
HOUR	Input	BYTE	L、M、V、constant
DAY	Input	BYTE	L、M、V、constant
MONTH	Input	BYTE	L、M、V、constant
YEAR	Input	BYTE	L、M、V、constant
CENTURY	Input	BYTE	L、M、V、constant

The following table details the effect of each parameter. FMT, WEEK, SECOND, MINUTE, HOUR, DAY, MONTH, YEAR, CENTURY parameters must be both constants or variables.

Parameter	Description
EN	enable terminal.
FMT	Output format, 0 means decimal, 1 means BCD code.

WEEK	Week, range: 1~7, where 1 represents Monday and 7 represents Sunday.
SECOND	seconds, range: 0~59
MINUTE	minutes, range: 0~59
HOUR	Hour, range: 0~23
DAY	day, range: 1~31
MONTH	month, range: 1~12
YEAR	Year, range: 0~99
CENTURY	century, fixed at 20

R RTC_W is used to modify the real-time clock according to the time specified by each parameter.

FMT specifies the format of each parameter. If FMT is 0, it is decimal; if FMT is 1, it is BCD code.

- LD

  If the EN value is 1, the RTC_R instruction is executed, otherwise it is not executed.

- IL

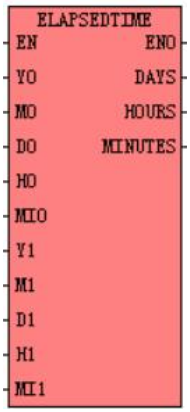
  If the CR value is 1, the RTC_R instruction is executed, otherwise it is not executed.

  RTC_R The execution of the instruction does not affect the CR value.

7.1.5 ELAPSEDTIME (The interval between two moments)

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	☑ K6S

LD	ELAPSEDTIME		U	☑ MK ☑ K6

Parameter	Input/Output	Data type	Allowed memory area	Parameter
Y0	Input	BYTE	L、M、V	The year value of the start time, since the year 2000
M0	Input	BYTE	L、M、V	Value of month at start time. The value ranges from 1 to 12
D0	Input	BYTE	L、M、V	Day value at the start time. The value ranges from 1 to 31
H0	Input	BYTE	L、M、V	Start time Specifies the hour value. The value ranges from 0 to 23
MI0	Input	BYTE	L、M、V	Value of minute at start time. The value ranges from 0 to 59
Y1	Input	BYTE	L、M、V	The "year" value of the end time, since 2000
M1	Input	BYTE	L、M、V	Value of month at end time. The value ranges from 1 to 12
D1	Input	BYTE	L、M、V	Day value at the end time. The value ranges from 1 to 31
H1	Input	BYTE	L、M、V	End time Specifies the hour value. The value ranges from 0 to 23
MI1	Input	BYTE	L、M、V	Value of minute at end time. The value ranges from 0 to 59
DAYS	Output	BYTE	L、M、V	Elapsed Days from Start time to End Time Indicates the elapsed days from start time to end time
HOURS	Output	BYTE	L、M、V	The number of hours elapsed from the start time to the end time

MINUTES	Output	BYTE	L、M、V	Minutes elapsed from the start time to the end time

Y0 (Y1) represents the year 2000+Y0 (Y1), for example, if Y0 is 19, then it represents 2019.

The ELAPSEDTIME command is used to calculate the total amount of time (represented by the combination of DAYS, HOURS, and MINUTES) that has elapsed between the start time (composed of Y0, M0, D0, H0, and MI0) and the end time (composed of Y1, M1, D1, H1, and MI1).

LD

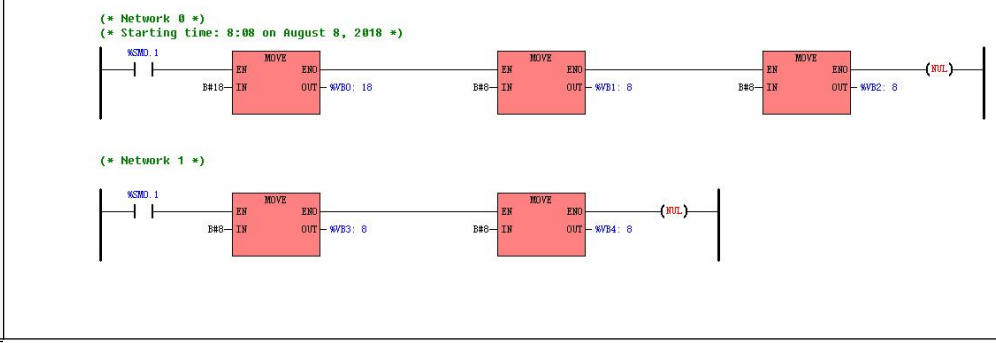
If the EN value is 1, the instruction is executed, otherwise it is not executed.

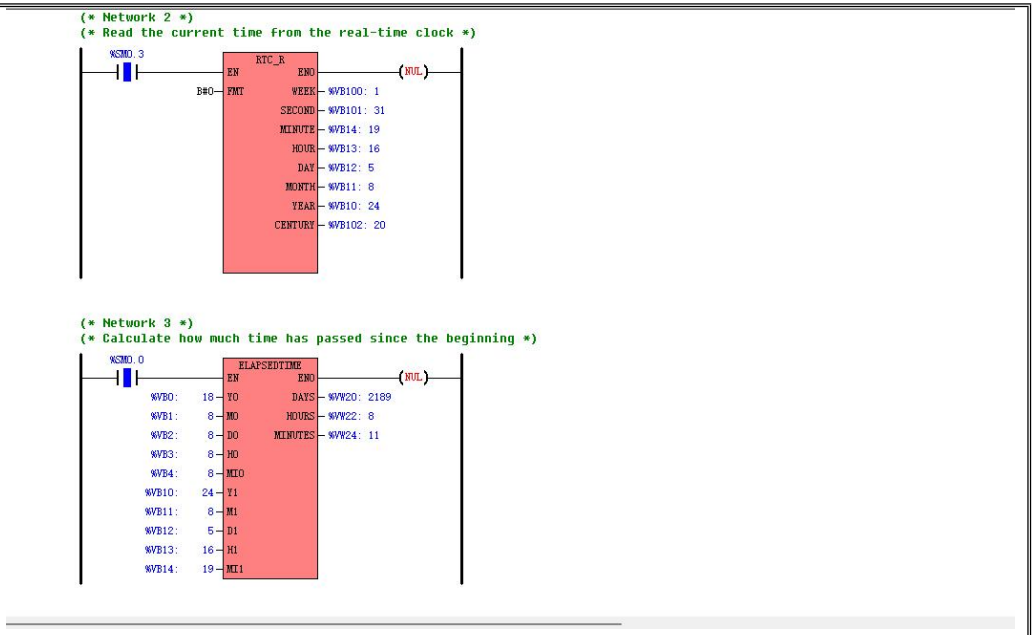
IL

If the CR value is 1, the command is executed, otherwise it is not executed. The execution of this instruction does not affect the CR value.

- Instruction usage example

LD





From the above execution results, it can be seen that from 8:8 on August 8, 2018 to 15:16 on August 5, 2024, a total of 2189 days, 8 hours and 11 minutes have passed.

Chapter 8 Use of High-speed Counter Function

This chapter mainly introduces the high-speed counting function of Kinco-K series PLC, including the high-speed counting working mode, input signal, status register, control register, PV interrupt, etc.

8.1 Function description

CPU model	Single Phase		AB Phase/Dual Phase	
	HSC0 and HSC1	HSC2 and HSC3	HSC0 and HSC1	HSC2 and HSC3
K5 series	60K	/	20K	/
K2 series (K205)	50K	20K	50K	10K
K2 series (K204ET)	200K	200K	200K	200K
K2 series (K209EA)	200K	20K	100K	10K
K2 series (K209M)	200K	/	200K	/
KS series (KS105)	200K	200K	200K	200K
KS series (KS101M)	200K	/	200K	/
KW series	200K	200K	200K	200K
MK series	50K	50K	50K	50K
K6 series	200K	6K	200K	2K

Kinco-K series PLC provides up to 4 high-speed counters, numbered from HSC0 to HSC3, and the highest counting frequency depends on the specific model.

The high-speed counter has a variety of working modes, and can count single-phase, double-phase (Up/Down), AB-phase (1 frequency and 4 frequency). The measurement of high-speed input signals such as measuring sensors, rotary encoders, grating rulers, etc. can be realized by selecting different working modes. Before using the high-speed counter, you need to use the HDEF instruction or use the HSC wizard (K5 does not support) to specify a working mode for it. All high-speed counters have the same function in the same operating mode.。

All high-speed counters (except K5 series) allow specifying a maximum of 32 preset values (PV), and

each PV value supports "count value (CV) = preset value (PV)" interrupt. The PV value can be specified as a relative value or an absolute value. If the relative value is selected, the "count value = preset value" interrupt allows the selection cycle to occur.

8.2 High-speed counter operating modes and input signals

The input signals of the high-speed counter include the following: clock (ie input pulse), direction, start and reset signals.

In different working modes, the required input signal is also different. The following tables describe in detail the operating modes supported by each high-speed counter and the assignment of input signals.

HSC 0				
Mode	**Description**	**I0.1**	**I0.0**	**I0.5**
0	Single-phase up/down counter with internal direction control Direction control bit: SM37.3	Clock		
1			Reset	
2			Reset	Turn on
3	Single-phase up/down counter with external direction control	Clock		Direct
4			Reset	Direct
6	Two-phase counter with up/down count clock input	Clock (minus)	Clock (plus)	
9	A/B Phase Quadrature Counter	Clock Phase A	Clock Phase B	

HSC1					
Mode	**Description**	**I0.4**	**I0.6**	**I0.3**	**I0.2**
0	Single Phase Plus/minus Counter with Internal Direct Control Direct Control Bit: SM47.3			Clock	
1		Reset			
2		Reset	Start		
3	Single Phase Plus/minus Counter with External Direct Control			Clock	Direct
4		Reset			Direct
6	Dual Phase Counter with Plus/minus Counter Clock Input			Clock (minus)	Clock (Plus)
7		Reset			
9	A/B Phase Quadrature Counter			Clock A	ClockB

10		Reset			

HSC 2			
Mode	**Description**	**I0.4**	**I0.5**
0	Single Phase Plus/minus counter with internal Direct control. Direct Control Bit: SM57.3		Clock
9	A/B Phase Quadrature Counter	Clock Phase B	Clock Phase A

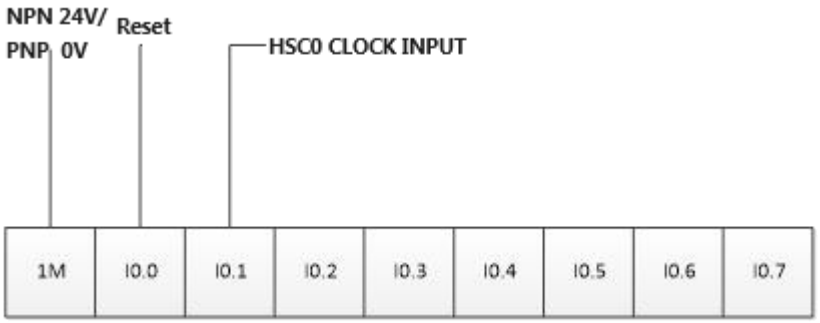
HSC 3			
Mode	**Description**	**I0.6**	**I0.7**
0	Single Phase Plus/minus counter with internal Direct control. Direct Control Bit: SM127.3		Clock
9	A/B Phase Quadrature Counter	Clock Phase B	Clock Phase A

8.3 High-speed count value range

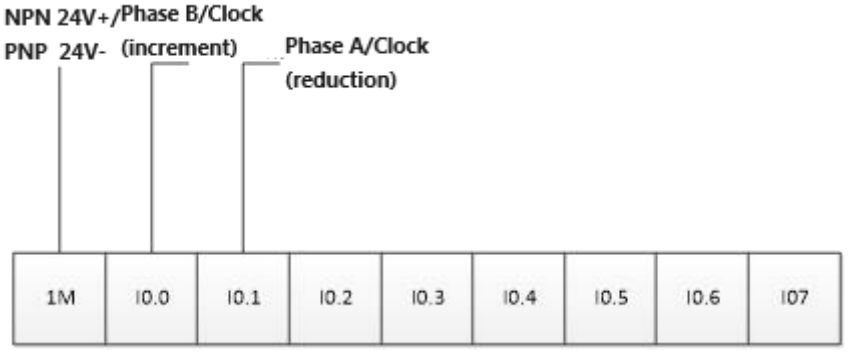
The count value of the high-speed counter is DINT type double integer, and its range is: -2,147,483,648 ~ +2,147,483,647. When the count value exceeds this range, overflow or underflow occurs. The so-called overflow occurs, that is, the count value jumps from +2,147,483,647 to -2,147,483,648, and continues to count; and when an underflow occurs, the count value jumps from -2,147,483,648 to +2,147,483,647, and continues to count.

8.4 High-speed counter input terminal wiring

The input terminal wiring of the high-speed counter is related to the counter and working mode it uses. The following two figures take HSC0 as an example to introduce.



Counter HSC0 single-phase pulse input, working mode 1



Counter HSC0 dual-phase/AB-phase pulse input, working mode 6/9

8.5 Timing diagram of high-speed counter

In order for users to better understand the high-speed counter, the following figures describe the working sequence of the high-speed counter in detail.

Figure 8-1 and Figure 8-2 apply to all operating modes with reset input signal and start input signal. Figure 8-3 to Figure 8-6 describe the Timing diagrams of each operating mode of the high-speed counter.

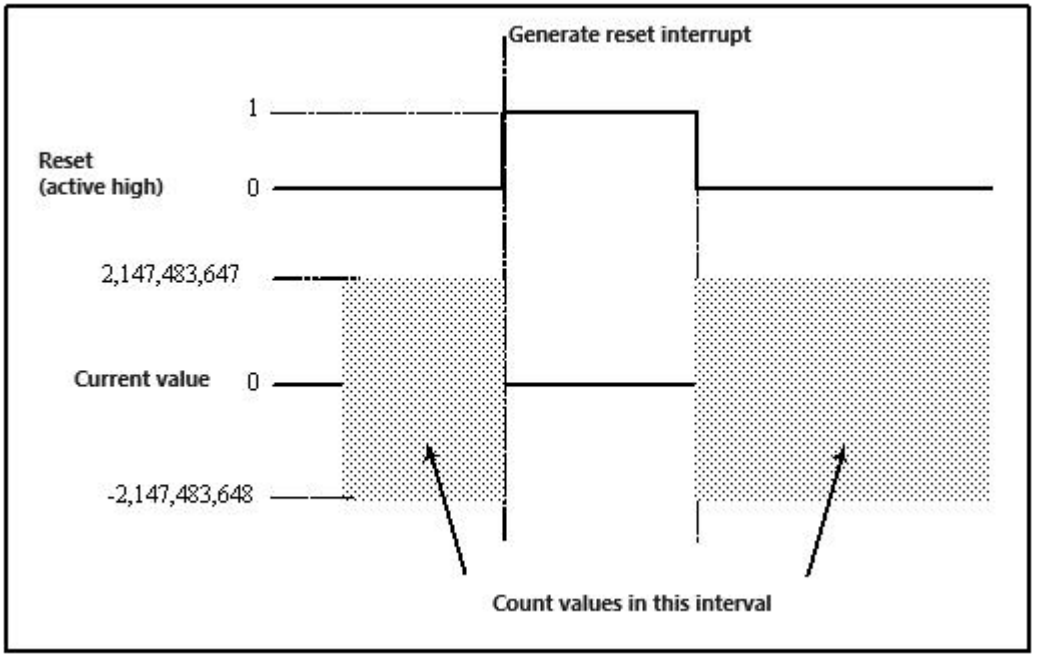


Figure 8-1 Timing diagram of high-speed counter with reset signal and no start signal

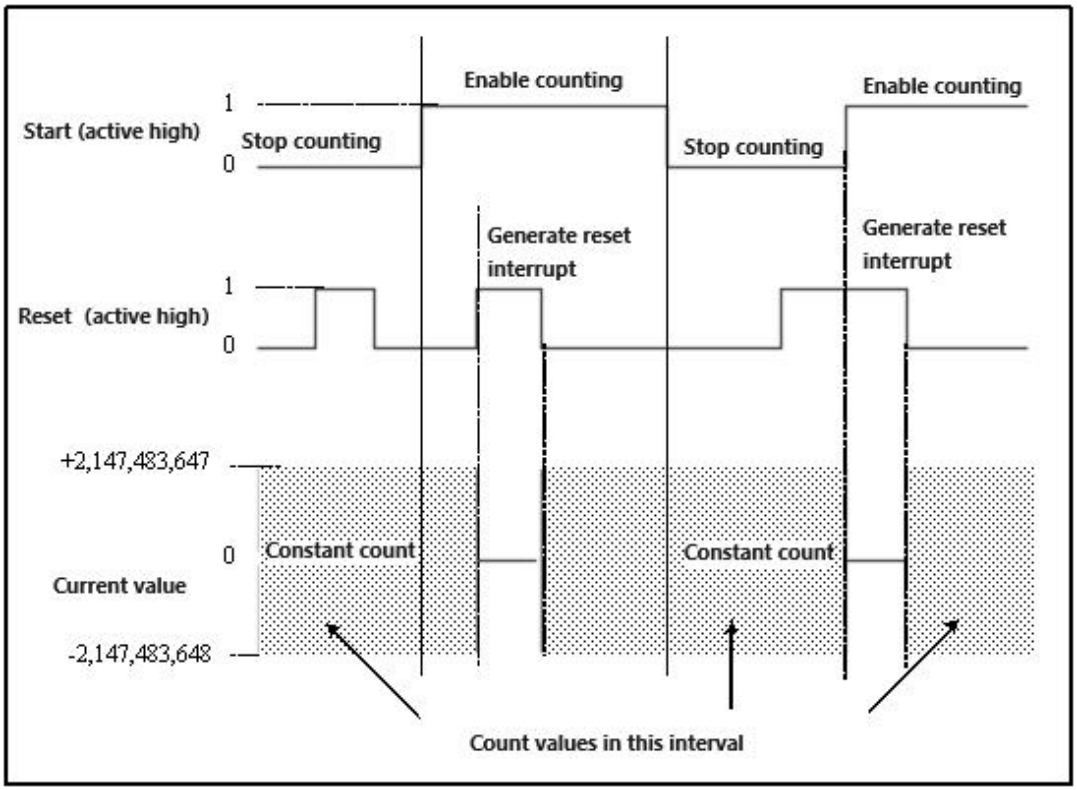


Figure 8-2 Timing diagram of high-speed counter with reset signal and start signal

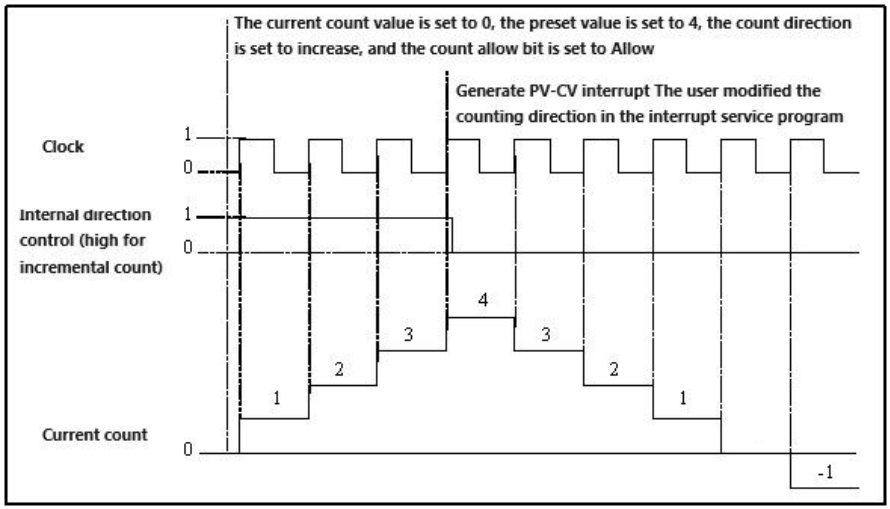


Figure 8-3 Timing diagram for modes 0, 1, and 2

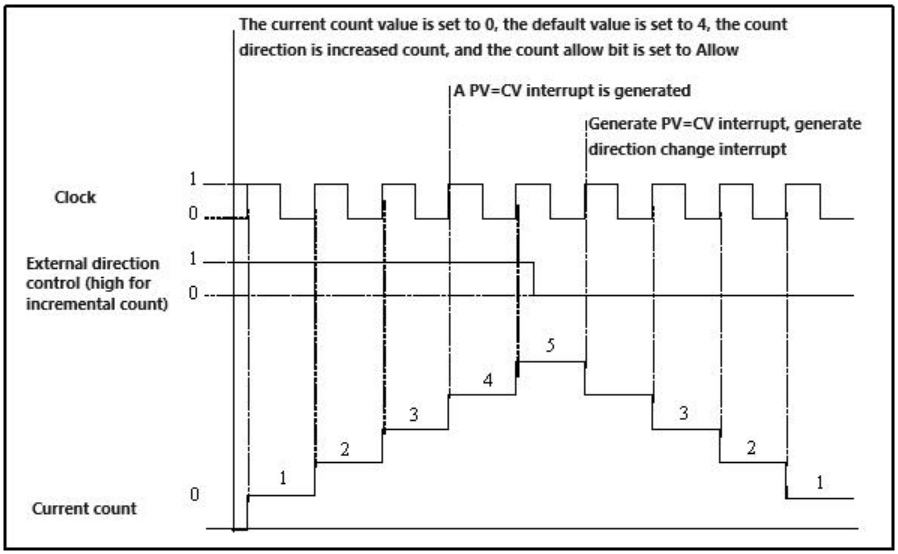


Figure 8-4 Timing diagram for modes 3 and 4

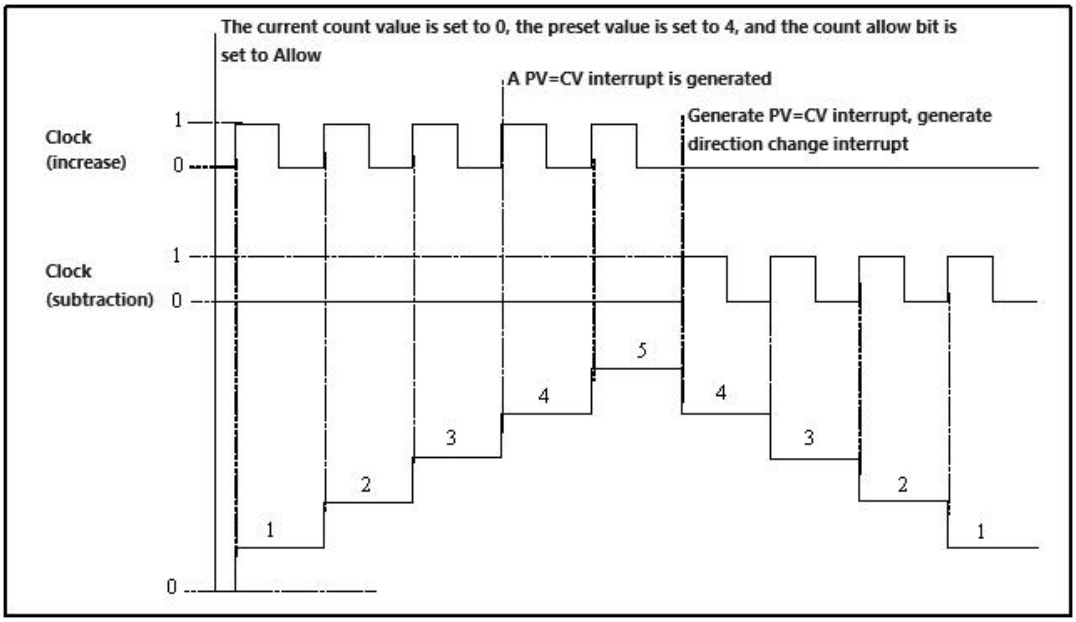


Figure 8-5 Timing diagram for modes 6 and 7

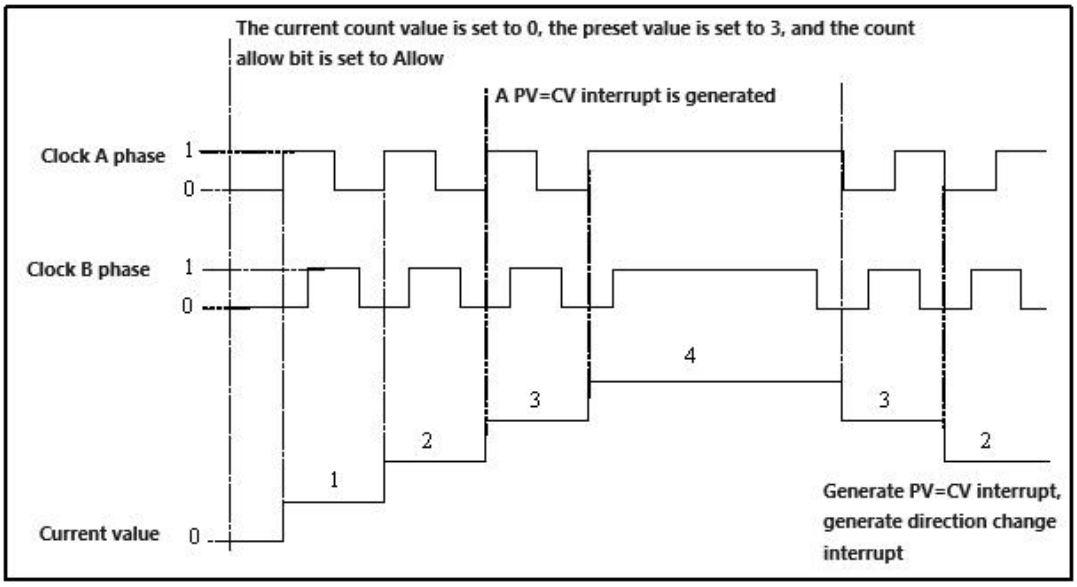


Figure 8-6 Timing diagram for modes 9 and 10 (orthogonal, 1x)

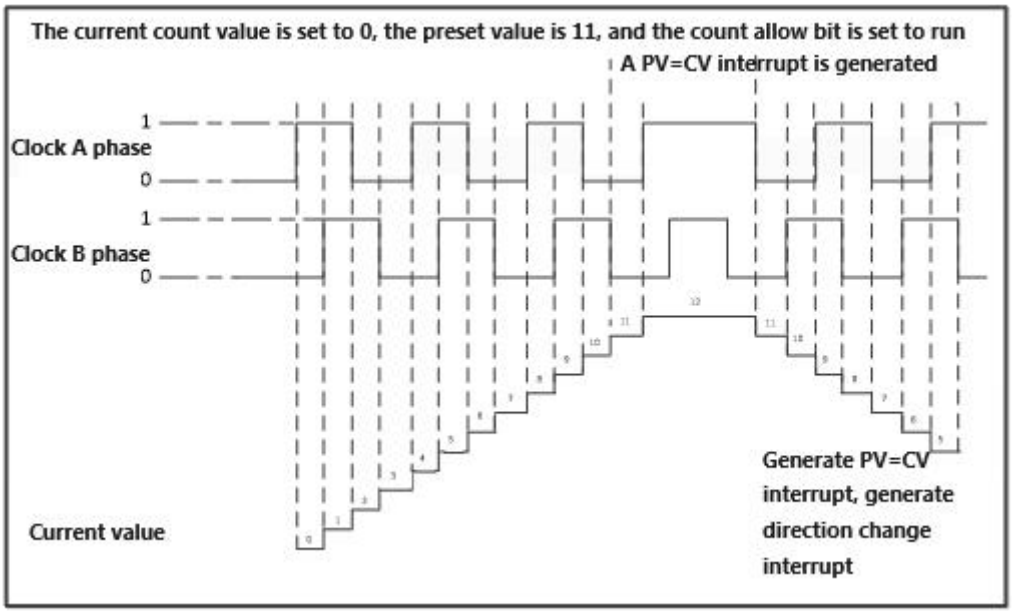


Figure 8-7 Timing diagram for modes 9 and 10 (orthogonal, 4x)

8.6 Control Register and Status Register

- **Control register**

In the SM area, the following control registers are provided for each high-speed counter to store its configuration data: a control byte (8 bits), a current value and a preset value (both are 32-bit signed double integers). The current value specifies the starting value of the count. If the current value is written to the high-speed counter, the high-speed counter will start counting from this value immediately. The following table describes these registers in detail.

HSC0	HSC1	HSC2	HSC3	Description
SM37.0	SM47.0	SM57.0	SM127.0	Valid level of reset signal: 0=high level; 1=low level
SM37.1	SM47.1	SM57.1	SM127.1	Valid level of start signal: 0=high level; 1=low level
SM37.2	SM47.2	SM57.2	SM127.2	Quadrature counter rate: 0=1x rate; 1=4x rate*
SM37.3	SM47.3	SM57.3	SM127.3	Counting direction: 0=count down; 1=count up.

SM37.4	SM47.4	SM57.4	SM127.4	Whether to write count direction to HSC: 0=no; 1=yes.
SM37.5	SM47.5	SM57.5	SM127.5	Whether to write new preset value to HSC: 0=no; 1=yes.
SM37.6	SM47.6	SM57.6	SM127.6	Whether to write new current value to HSC: 0=no; 1=yes.
SM37.7	SM47.7	SM57.7	SM127.7	Whether to allow the high-speed counter: 0=disable; 1=enable.
HSC0	**HSC1**	**HSC2**	**HSC3**	**Description**
SMD38	SMD48	SMD58	SMD128	new current value
SMD42	SMD52	SMD62	SMD132	new preset value

In addition, all high-speed counters except K5 series CPU allow to specify a maximum of 32 preset values (PV), and their control registers are described as follows:

HSC0	**HSC1**	**HSC2**	**HSC3**	**Description**
SM141.0	SM151.0	SM161.0	SM171.0	Whether to use multi-segment preset value: 0=no; 1=yes
SM141.1	SM151.1	SM161.1	SM171.1	Whether the preset value is relative or absolute: 0 = absolute; 1 = relative
SM141.2	SM151.2	SM161.2	SM171.2	Whether the preset value comparison ("CV=PV") interrupt is generated cyclically: 0=no; 1=yes. Note: Only relative value mode is allowed to be set to cycle generation。
SM141.3	SM151.3	SM161.3	SM171.3	reserve
SM141.4	SM151.4	SM161.4	SM171.4	Whether to update segment number and preset value: 0=no; 1=yes
SM141.5	SM151.5	SM161.5	SM171.5	Whether to reset the interrupt variable: 0=Yes; 1=No
SM141.6	SM151.6	SM161.6	SM171.6	reserve
SM141.7	SM151.7	SM161.7	SM171.7	reserve
HSC0	**HSC1**	**HSC2**	**HSC2**	**Description**
SMW142	SMW152	SMW162	SMW172	The starting position of the preset value table (represented by the byte offset relative to VB0), which must be an odd number.

It should be noted that not all control bits in the control byte are applicable to all operating modes. For example, the two control bits "count direction" and "whether to write count direction to HSC" are only used in modes 0, 1 and 2 (single-phase up/down counter with internal direction control). If the working mode of the high-speed counter is to use an external direction control signal, then these two control bits will be

ignored.

The default value of the current value and preset value of the control byte is 0 after power-on.

- **Status Register**

Each high-speed counter provides a status register in the SM area to indicate the current status information of the high-speed counter.

HSC0	HSC1	HSC2	HSC3	Description
SM36.0	SM46.0	SM56.0	SM126.0	reserve
SM36.1	SM46.1	SM56.1	SM126.1	reserve
SM36.2	SM46.2	SM56.2	SM126.2	reserve
SM36.3	SM46.3	SM56.3	SM126.3	Whether there is an error in the setting of the multi-segment PV value table: 0=No, 1=Yes.
SM36.4	SM46.4	SM56.4	SM126.4	reserve
SM36.5	SM46.5	SM56.5	SM126.5	Current counting direction: 0=decrease; 1=increase.
SM36.6	SM46.6	SM56.6	SM126.6	Whether the current count value is equal to the preset value: 0=No; 1=Yes.
SM36.7	SM46.7	SM56.7	SM126.7	Whether the current count value is greater than the preset value: 0=No; 1=Yes.
HSC0	HSC1	HSC2	HSC3	Description (K5 series not supported)
SMB140	SMB150	SMB160	SMB170	Running PV value segment number (starting from 0).

- **Read the current count value of the high-speed counter**

The count value of the high-speed counter is a 32-bit double integer and is read-only.

The Kinco-K series provides a high-speed counter area (HC area) in the memory area to store the count value of each high-speed counter. This area is addressed by the memory area type (HC) and the number of the high-speed counter. For example, HC2 represents the current count value of HSC2. **See 3.6.2.1 Direct Address Representation Format for more information.** The following figure describes this addressing mode.

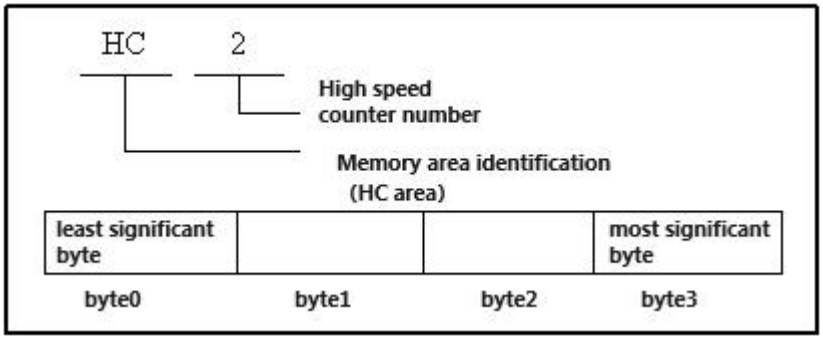


Figure 8-7 Example of reading the count value of the high-speed counter

8.7 High-speed counter interrupt

Each high-speed counter supports interrupts: When the count value is equal to the preset value, a "PV (preset value) = CV (count value)" interrupt will be generated. If the high-speed counter mode using the external reset signal is used, an "external reset" interrupt will be generated during reset; if the counter mode using the external direction control signal will be generated, the "direction change" interrupt will be generated when the counting direction changes.

The list of interrupt events for high-speed counters is as follows:

Event number	Interrupt description	Classification
191	The 32nd "CV=PV" when HSC3 uses multi-segment PV values	I/O Interrupt
…	... (add 1 in turn)	
161	The second "CV=PV" when HSC3 uses multi-segment PV values	
160	The first "CV=PV" when HSC3 uses multi-segment PV values	
159	The 32nd "CV=PV" when HSC2 uses multi-segment PV values	
…	... (add 1 in turn)	
129	The second "CV=PV" when HSC2 uses multi-segment PV values	
128	The first "CV=PV" when HSC2 uses multi-segment PV values	
127	The 32nd "CV=PV" when HSC1 uses multi-segment PV values	
…	... (add 1 in turn)	
97	The second "CV=PV" when HSC1 uses multi-segment PV values	
96	The first "CV=PV" when HSC1 uses multi-segment PV values	
95	The 32nd "CV=PV" when HSC0 uses multi-segment PV values	
…	... (add 1 in turn)	

65	The second "CV=PV" when HSC0 uses multi-segment PV values	
64	The first "CV=PV" when HSC0 uses multi-segment PV values	
18	When HSC0 uses a single PV value, CV=PV (current value=preset value)	
17	HSC0 input direction change	
16	HSC0 external reset	
15	When HSC1 uses a single PV value, CV=PV (current value=preset value)	
14	HSC1 input direction change	
13	HSC1 external reset	
12	When HSC2 uses a single PV value, CV=PV (current value=preset value)	
9	When HSC3 uses a single PV value CV=PV (current value=preset value)	

8.8 Use of PV Interrupts

There are two PV setting methods for Kinco-K series PLC: single PV setting method and multi-PV setting method.

The single PV setting method allows only 1 PV value to be set for each high-speed counter.

The multi-PV setting method allows each high-speed counter to set up to 32 PV values and allows you to choose whether the relationship between PVs is a relative value or an absolute value. The relative value method allows "CV=PV" to interrupt the cycle generation.

The PV setting methods supported by Kinco-K series PLC are shown in the following table:

Model	**PV setting method**		
	Single PV Setting	**Multi-PV Setting （32 PV value）**	
		Absolute value （cycle interrupt not supported）	**Relative value （cycle interrupt supported）**
K5 series	Supported	Not Supported	
K2 series	Supported	Supported	
KS series			
KW series			
MK series			
K6 series			

The following will take HSC0 as an example to describe the function and setting method of PV value in

detail.

8.8.1 Preset value (PV value) setting

- **How to choose single-segment and multi-segment PV values**

A control bit is provided in the control register of each high-speed counter to select whether to use the multi-segment preset value.

This control bit for HSC0 is SM141.0.

If SM141.0 is 0, it means that a single PV value is used, which is consistent with the usage of K5: SMD42 specifies a new PV value, and SM37.5 specifies whether to use this new PV value.

If SM141.0 is 1, it means that the multi-stage PV value mode is adopted. At this time, SM37.5 and SMD42 are invalid. Each PV value is stored in the PV value table (SMW142 is the starting address of the table), and SM141.4 indicates whether to use the data in the PV value table. If SM141.4 is 1, it means that after this startup, the high-speed counter will use the data in the PV value table. If SM141.4 is 0, it means that after this startup, the high-speed counter will use the last PV value data and ignore the data in the PV value table.

- **Table of multi PV value**

If multi-segment PV values are used, each PV value will use the data in the PV value table.

A control word is provided in the control register of each high-speed counter to store the starting address of the PV value table, **The starting address of the table must be an odd address in the V area, such as 301 (representing VB301)**

The format of the PV value table is as follows

	Data type	Description
0	BYTE	Number of PV values
1	DINT	1st PV value
5	DINT	2nd PV value
…	DINT	…

- All offsets are offset in bytes relative to the start of the table.

- When the relative value method is adopted, the mathematical absolute value of the PV value must

be greater than 1, otherwise the PLC considers that the number of segments has ended here, and counts the number of PV values based on this (priority to the set value of the number);

- When the absolute value method is adopted, the mathematical absolute value of the difference between two adjacent PV values must be greater than 1, otherwise the PLC considers that the number of segments ends here, and counts the number of PV values based on this (priority to the number set value)

- Users should pay attention when setting the PV value: "CV=PV" interrupts must be generated in sequence. That is to say, when the count value reaches the first PV value and generates an interrupt, the PLC will then compare it with the second PV value, and so on.

- The PV value setting must be reasonable. Taking relative values as an example, when counting up, the PV value must be greater than 0, otherwise the “CV=PV” interrupt corresponding to this value may never be generated; when counting down, the PV value must be less than 0, otherwise The "CV=PV" interrupt may never be generated.

8.8.2 Relative and absolute values

A control bit is provided in the control register of each high-speed counter to select whether the PV value is a relative value or an absolute value. This control bit for HSC0 is SM141.1.

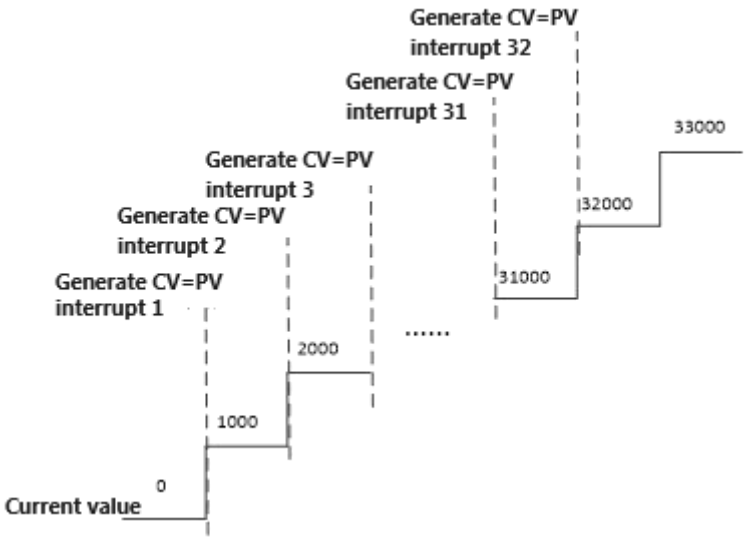
- **Absolute value method**

If SM141.1 is 0, it means that the PV value is an absolute value method. When the count value is equal to the PV value, the corresponding "CV=PV" interrupt will be generated. For example, the count value of the high-speed counter is 0 when it is started. If 32 PV values are set, the sequence is 1000, 2000, 3000...32000, then when the count value reaches 1000, the first "CV=PV" interrupt will be generated; When the count value reaches 2000, the second "CV=PV" interrupt will be generated; and so on.

Counter	CV (Count value)=PV (Preset value)break marker				
	Segment 1	Segment 2	……	Segment N	Segment 32

HC0	1000	2000	……	N*1000	32000

It can be seen from the table that an interrupt will be generated when the HC0 count value is equal to the preset value, which can be understood as the process of absolute positioning in motion control, and the HC0 count value is the value relative to the zero point. As shown below:

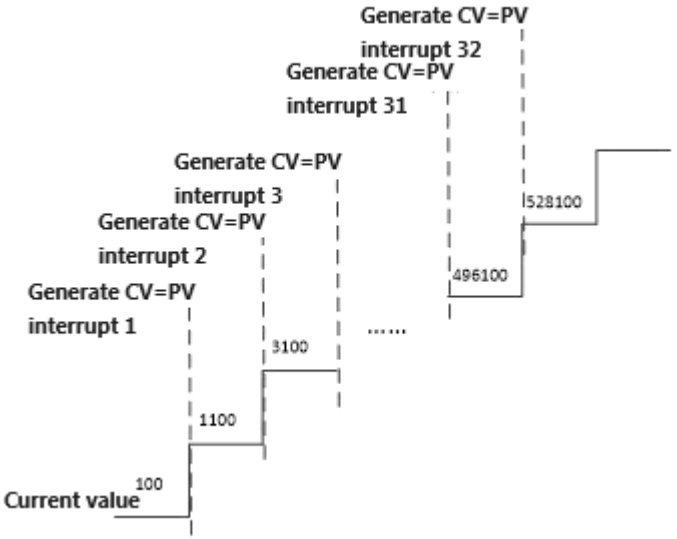


- **Relative value method**

If SM141.1 is 1, it means that the PV value is a relative value method. If the counter is based on the current count value and continues to count until the difference is equal to the PV value, a corresponding "CV=PV" interrupt will be generated. For example, if you set 32 PV values, which are 1000, 2000, 3000 to 32000, and the count value when the high-speed counter starts is 100, then when the count value reaches 1100, 3100, 6100 to 528100, the interrupt will be generated respectively

Counter	CV (Counter value)=PV (Preset value)break marker				
	Segment 1	Segment 2	……	Segment N	Segment 32
HC0	1100	3100	……	100+N* (N+1)*1000/2	528100

It can be seen from the table that an interrupt will be generated when the count value of HC0 is equal to the sum of the previous N times preset values. This can be understood as the process of relative positioning in motion control, and the count value of the next interrupt generated by HC0 is the increase of the relative preset value. As shown below:



- **"CV=PV" interrupt cycle generation in relative value mode**

Only when the PV value is in the relative value mode, it is allowed to set the cycle to generate an interrupt, otherwise it is invalid.

If SM141.0 is 0, it means that "CV=PV" interrupt is only generated once. It stops when all interrupts corresponding to PV values are completed. To continue generating, the corresponding register value must be modified and the HSC instruction called again.

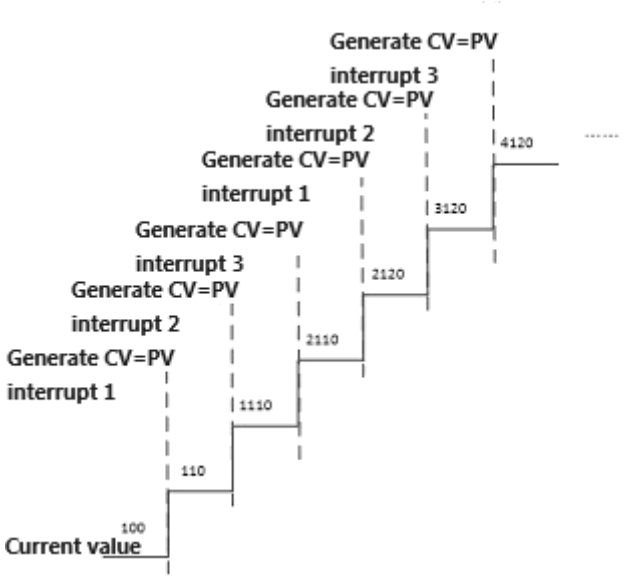
If SM141.0 is 1, it means that "CV=PV" interrupt will be generated cyclically. When the interrupt corresponding to the last PV value is completed, the PLC will take the current count value as the benchmark and add it to each PV value again to obtain the new value required for the interrupt. Then continue to compare with the count value, and generate the corresponding "CV=PV" interrupt. This process will keep

looping and never stop.

For example, if you set 3 PV values, 10, 1000, and 1000, and the count value when the high-speed counter starts is 100, the required values for each interrupt are as follows:

Current counter	Interrupt times	Value 1	Value 2	Value 3
100	1st	110	1110	2110
2110	3rd	2120	3120	4120
4120	6th	4130	5130	6130
…	nth	…	…	…

如下图:

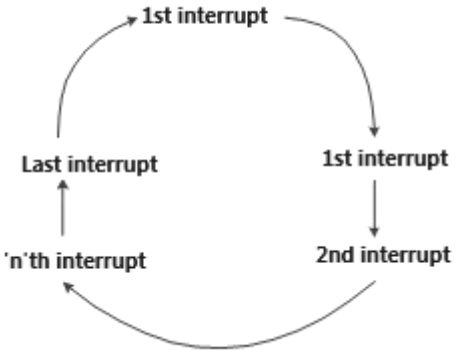


The break loop is only available in relative count mode. In interrupt loop mode, it will automatically restart after the interrupt ends. This mode is especially suitable for the following situations:

(1)Continuous reciprocating motion;

(2) Periodic interrupt is generated according to the specified pulse.

The sequence of the interrupt loop is as follows:



8.8.3 CV=PV interrupt number

When using the single PV value mode, other series of PLC high-speed counters are fully compatible with K5, including the number of "CV=PV" interrupts that are consistent with those in K5.

High-speed counter	Interrupt number	Description
HSC0	18	When HSC0 uses a single PV value, CV=PV (current value=preset value)
	17	HSC0 input direction change
	16	HSC0 external reset
HSC1	15	When HSC1 uses a single PV value, CV=PV (current value=preset value)
	14	HSC1 input direction change
	13	HSC1 external reset
HSC2	12	When HSC2 uses a single PV value, CV=PV (current value=preset value)
	11—10	reserve
HSC3	9	When HSC3 uses a single PV value CV=PV (current value=preset value)
	8—5	reserve

When using the multi-stage PV value mode, the high-speed counter assigns a new interrupt number to each of the 32 PV values, as shown in the following table

High-speed counter	Interrupt number	Description
HSC0	64	"CV=PV" interrupt for 1st PV value

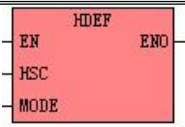
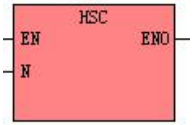
	65	"CV=PV" interrupt for 2nd PV value
	…	... (add 1 in turn)
	95	"CV=PV" interrupt for the 32nd PV value
HSC1	96	"CV=PV" interrupt for 1st PV value
	97	"CV=PV" interrupt for 2nd PV value
	…	... (add 1 in turn)
	127	"CV=PV" interrupt for the 32nd PV value
HSC2	128	"CV=PV" interrupt for 1st PV value
	129	"CV=PV" interrupt for 2nd PV value
	…	... (add 1 in turn)
	159	"CV=PV" interrupt for the 32nd PV value
HSC3	160	"CV=PV" interrupt for 1st PV value
	161	"CV=PV" interrupt for 2nd PV value
	…	... (add 1 in turn)
	191	"CV=PV" interrupt for the 32nd PV value

8.9 HDEF (High-speed counter definition)、HSC (High-speed counter)Instruction

The operation of the high-speed counter is not affected by the CPU scan cycle, so it can be used to count high-speed pulse inputs.

HDEF and HSC instructions are located in the [Instruction Set]->[Counter] group.

- Instruction and its operand description

	Name	Instructions format	Affect CR value	
LD	HDEF			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	HSC			
IL	HDEF	HDEF HSC, MODE	U	
	HSC	HSC *N*		

Parameter	Input/Output	Data type	Description
HSC	Input	INT	constant (High-speed counter number)

MODE	Input	INT	constant (0-10, High-speed counter operation mode)
N	Input	INT	constant (High-speed counter number)

The function of the HDEF instruction is to define a working mode MODE for the high-speed counter specified by the parameter HSC.

The function of the HSC instruction is: first read the working mode specified by the HDEF instruction and the value of the corresponding control register in the SM area and configure the characteristics of the high-speed counter N, and then start the high-speed counter N.

Note: The high-speed counter can be configured and started only by triggering the execution of the HSC instruction once when needed! It is recommended to use conditions such as rising edge and falling edge to trigger the HSC instruction. If the trigger condition remains "1" for a long time (for example, using SM0.0), the high-speed counter will always be initialized and cannot be used normally.

- LD

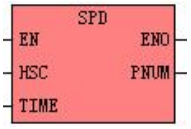
  If the EN value is 1, execute HDEF, HSC instructions, otherwise do not execute.。

- IL

  If the CR value is 1, execute HDEF, HSC instructions, otherwise do not execute. The implementation of HDEF and HSC does not affect the CR value。

8.10 SPD (pulse density) instruction

➢ Instruction and its operand description

	Name	Instruction format	Affect CR value	Suitable for
LD	SPD			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK

IL	SPD	SPD HSC, TIME, PNUM	U	☑ K6

Parameter	Input/Output	Data type	Description
HSC	Input	INT	Constant (High-speed counter number)
TIME	Input	WORD	I、Q、M、V、L、SM、constant
PNUM	Output	DINT	Q、M、V、L、SM

The SPD instruction is used to calculate the number of pulses received by the high-speed counter HSC within the time TIME (unit: ms), and write the result into PNUM.

- LD

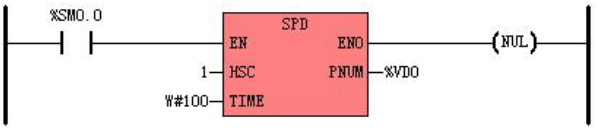
  If EN is 1, the instruction is executed.

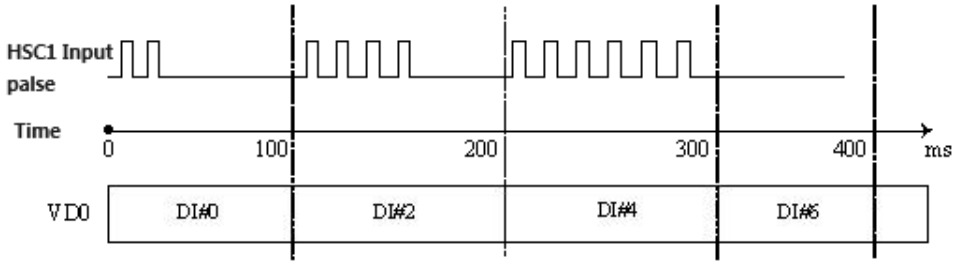
- IL

  If the CR value is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

➢ Instruction usage example

LD		SM0.0 is always 1, so the SPD instruction is always executed: calculate the pulse count value of HSC1, and assign the calculated pulse value within this time to VD0 every 100ms.

Res upt	Examples of results are as follows: 

8.11 How to use the high-speed counter

There are two ways to use the high-speed counter:

1. Use related instructions for programming: In this way, you need to set the corresponding control register in the program, and call HDEF instruction and HSC instruction for programming. This method is suitable for all Kinco-K series PLCs, suitable for single-stage PV setting method and multi-stage PV setting method (K5 series only supports single-stage PV setting method).

2. Use the HSC wizard to set: This method is simple and intuitive, just need to check and set according to the content provided in the wizard. This method is suitable for all PLCs except K5 series. It is suitable for single-stage PV setting method and multi-stage PV setting method. It is recommended to use the wizard method for PLCs other than K5 series to set, which is simple and convenient and saves programming time.

8.11.1 Programming with relevant instructions

The user can program a high-speed counter as follows:

1) Configure the control byte of the high-speed counter, and specify the current value (that is, the starting value of the count) and the preset value;

2)Use the HDEF instruction to define a high-speed counter and its working mode;

3)Use the HSC instruction to configure and start the high-speed counter;

4)(Optional) Use the ATCH instruction to connect the corresponding interrupt service routine for the high-speed counter interrupt;

Note: After the CPU enters RUN mode, only one HDEF instruction is allowed to be executed for each high-speed counter.。

The following will take HSC0 as an example to explain in detail how to program and use the high-speed counter.

It is recommended that the user try to write a separate initialization subroutine in the project, which contains the HDEF instruction and other initialization instructions, so that the entire user project can have a good structure.

8.11.1.1 Use a high-speed counter

- **Use a high-speed counter**

Mode 9 will be used as an example for description below, but the described steps are also applicable to other working modes in principle, and the user can make slight adjustments according to actual needs.

1) Set the control byte SMB37 according to the desired operation.

   For example, (1x count), SMB37 = B#16#F0 means:

   - Allow HSC0 to count;
   - Allow to write new current value and preset value to HSC0
   - Set the start signal and reset signal to be active high.

2) Assign the desired current value (32-bit double integer, that is, the starting value of the count) to SMD38.
3) Assign the desired preset value (32-bit double integer) to SMD42.
4) (Optional) Use the ATCH instruction to connect an interrupt service routine for the "PV = CV" interrupt event (event number 18) for fast response to the interrupt event.
5) (Optional) Use the ATCH instruction to connect an interrupt service routine for the "direction change" interrupt event (event number 17) for fast response to the interrupt event. (some modes do not support)
6) (Optional) Use the ATCH instruction to connect an interrupt service routine for the "external reset"

interrupt event (event number 16) for fast response to the interrupt event. (some modes do not support)

7) Execute the HDEF instruction, input 0 at the HSC end and 9 at the MODE end (mode 9 is used in this example);

8) Execute the HSC instruction to configure and start HSC0.

➢ **Change counting direction (mode 0, 1, 2)**

The following describes how to change the count direction of HSC0 (mode 0, 1, 2).

1) Set the control byte SMB37 according to the desired operation.

For example, SMB37 = B#16#90 shows:

- allow counting;
- Write the new count direction to HSC0;
- Set the count direction to count down.

2) Execute the HSC instruction to configure HSC0.

➢ **Change the current value (for all modes)**

The following describes how to change the current value of HSC0 (that is, the starting value of the count).

1) Set the control byte SMB37 according to the desired operation.

SMB37 = B#16#C0 means:

- allow counting.
- Allows writing a new current value to HSC0.

2) Assign the desired current value (32-bit double integer) to SMD38.

3) Execute the HSC instruction to configure HSC0.

➢ Change the preset value (for all modes)

The following describes how to change the preset value of HSC0.

1) Set the control byte SMB37 according to the desired operation.

SMB37= B#16#A0 means:

- allow counting.
- Allows updating a preset value to HSC0.

2) Assign the desired preset value (32-bit double integer) to SMD42.

3) Execute the HSC instruction to configure HSC0.

➢ Disable high-speed counter (applicable to all modes)

The following describes how to disable HSC0.

1) Set the control byte SMB37 according to the desired operation.

SMB37= B#16#00 means: disable the high-speed counter.

2) Execute the HSC instruction to make the CPU disable HSC0.

Note: When modifying the value of the control byte (such as SMB37 of HSC0), the 8 bits of the byte will be modified, and each bit represents a different meaning, so you need to pay attention to avoid incorrectly modifying a control bit. For example, the initial setting of SMB37= B#16#84 allows high-speed counting to count by 4 times the frequency. If the user modifies SMB37= B#16#C0 in order to change the current value, then the count multiple should be changed at the same time. To avoid this situation, the user can directly operate the corresponding function bits.

8.11.1.2 Example of use

1、The following takes HSC0 as an example to program a single-segment PV setting method.

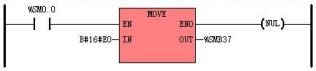
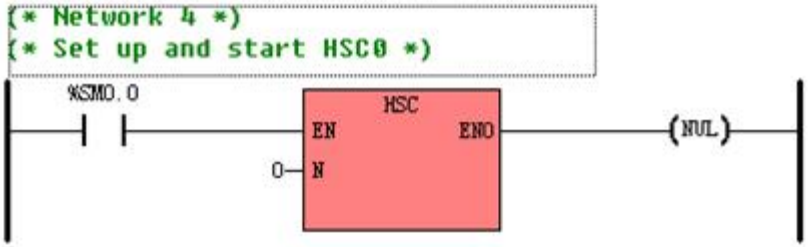
The example contains three programs: MAIN, Initialize, and ReachPV.

MAIN: call subroutine Initialize to initialize the high-speed counter HSC0.

Initialize: Set the working mode, counting mode, current value CV, preset value PV, the interrupt subroutine connected when the CV=PV interrupt is generated, etc., and start the counter.

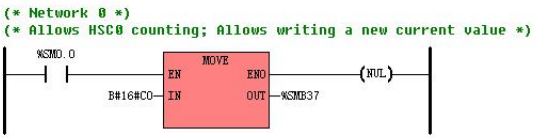
ReachPV: This program is executed when the CV=PV interrupt occurs, the current value CV is cleared to restart the counter to start a new count, and when CV=PV, the interrupt is continued to enter a

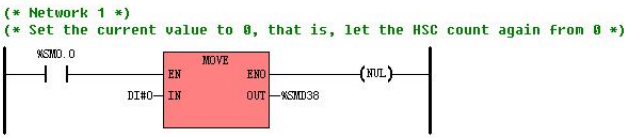
reciprocating cycle.

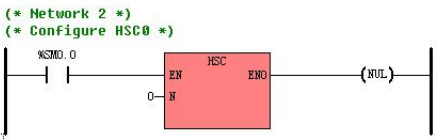
LD	Initialize (* Network 0 *) (* 1X counting mode, allowing counting, allowing updating the current value, preset value; Reset and start signals are active at high levels *)  (* Network 1 *) (* Set the current value to 0 and the preset value to 100 *) (* Network 2 *) (* Connect the CV=PV interrupt event to the reachPV interrupt service routine *) (* Network 3 *) (* Set the working mode of HSC0 to mode 9 *) 

LD

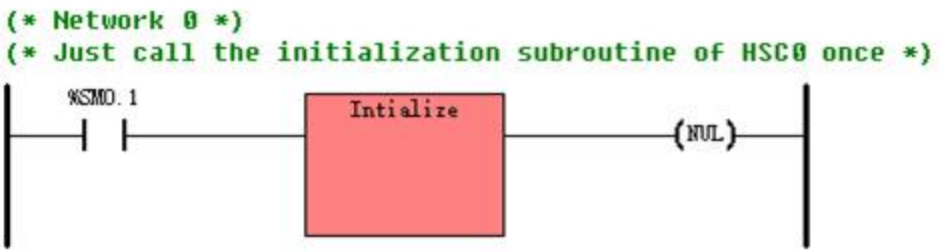
ReachPV







MAIN



IL

Initialize

```
(* Network 0 *)
LD      %SM0.0
MOVE    B#16#E0, %SMB37      (*set control byte *)
MOVE    DI#0, %SMD38          (*Set the current value to 0*)
MOVE    DI#100, %SMD42        (*Set the preset value to 100*)
ATCH    ReachPV, 18           (*Connect "CV=PV" interrupt and ReachPV interrupt program *)
HDEF    0, 9                  (*Set the working mode of HSC0 to mode 9 *)
HSC     0                     (*Set and start HSC0 *)
```

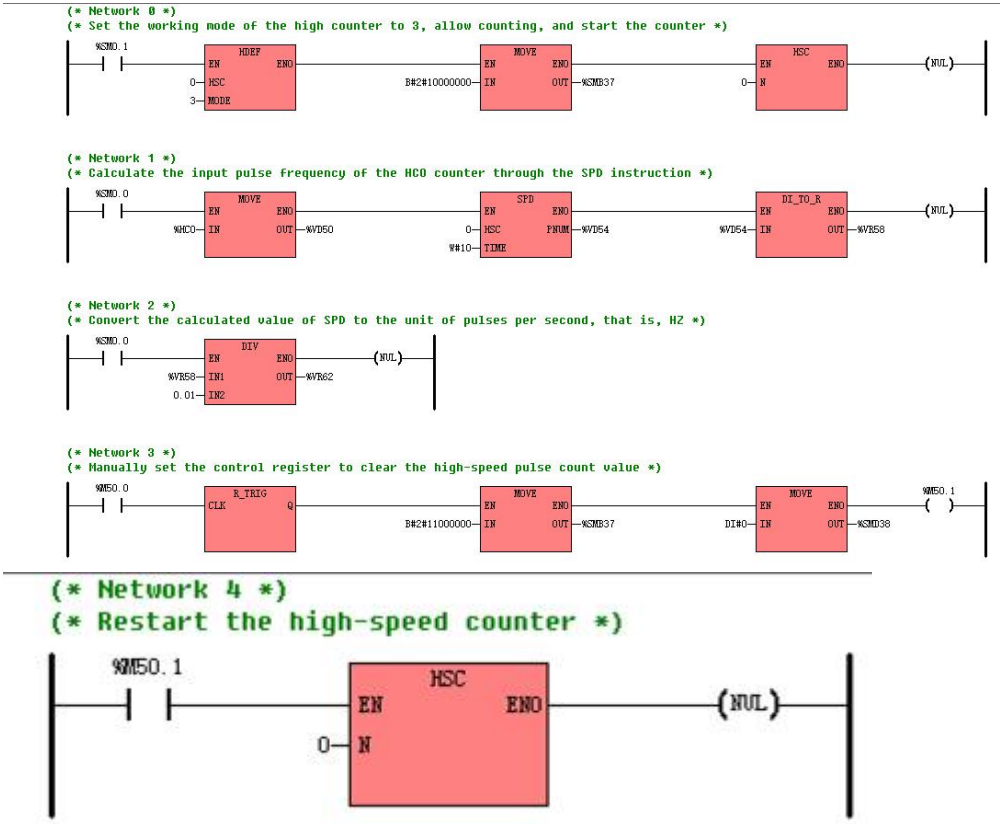
ReachPV

```
(* Network 0 *)
LD     %SM0.0
MOVE   B#16#C0, %SMB37          (*Allow HSC0 to count; allow to update current value *)
MOVE   DI#0, %SMD38             (*set the current value to 0 *)
HSC    0                        (* Set HSC0 *)

MAIN
LD     %SM0.1
CAL    Initialize               (*Just call the initialization subroutine of HSC0 once *)
```

2、The following is still taking HSC0 as an example to calculate the high-speed pulse input frequency and manual reset count value.

LD

Main



IL

```
(* Network 0 *)
(*Set the working mode of the high counter to 3, allow counting, and start the counter *)
LD      %SM0.1
HDEF    0, 3
MOVE    B#2#10000000, %SMB37
HSC     0
(* Network 1 *)
(*Calculate the input pulse frequency of the HCO counter through the SPD instruction *)
LD      %SM0.0
```

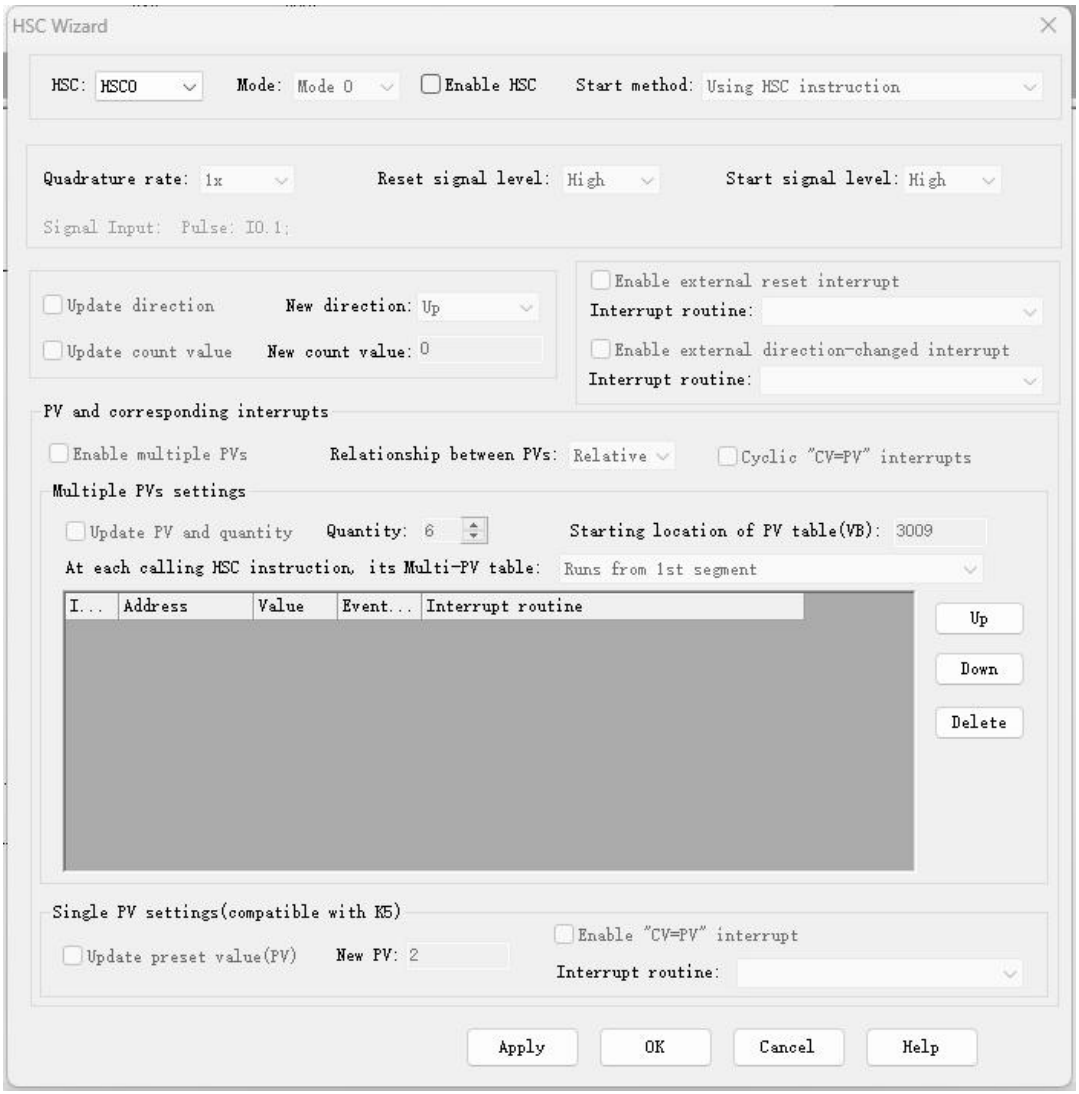
```
MOVE    %HC0, %VD50
SPD     0, W#10, %VD54
DI_TO_R %VD54, %VR58
(* Network 2 *)
(*Convert the calculated value of SPD to the unit of pulses per second, that is, HZ *)
LD      %SM0.0
MOVE    %VR58, %VR62
DIV     0.01, %VR62
(* Network 3 *)
(*Manually set the control register to clear the high-speed pulse count value *)
LD      %M50.0
R_TRIG
MOVE    B#2#11000000, %SMB37
MOVE    DI#0, %SMD38
ST      %M50.1
(* Network 4 *)
(*Restart the high-speed counter *)
LD      %M50.1
HSC     0
```

8.11.2 Using the HSC Wizard

8.11.2.1 HSC Wizard Setup

All PLC users except the K5 series can directly use the HSC wizard to configure the high-speed counter without complicated programming. The wizard is as shown below.

Even if the HSC is configured through the wizard, it is only the initial configuration of the high-speed counter. The user can modify, start and stop the parameters of the high-speed counter at any time according to "Method 1" in the program.



Use the HSC Wizard as follows:

- In [Counter], select the counter to be used which corresponds to the four-channel high-speed pulse counter introduced earlier.
- Select [Enable this counter], and then it will allow subsequent configuration to allow counting in the corresponding controller register (taking HSC0 as an example, it can be considered as the corresponding

control register SM37.7).

- In [Mode], select the counter mode to be used corresponding to the working mode in chapter 8.2 High-speed Counter Working Mode and Input Signal.
- In [Startup Mode], select the startup mode of the high-speed counter.。

  There are two ways to start:

  "Call the HSC instruction in the program": If this method is selected, the counter is started by calling the HSC instruction in the user program. Before calling the HSC instruction, there is no need to configure each register and call the HDEF instruction.

  "Start directly after the PLC is powered on": If this method is selected, the high-speed counter will run automatically after the PLC is powered on, without calling any instructions.
- Modifying the counting direction and modifying the count value corresponds to the modification of the counting direction and current value of the control register in the chapter 8.6 Control Register and Status Register.
- To use multi-stage PV value mode, select [Enable multi-preset value (PV) function], and then you can configure PV value, quantity, associated interrupt subroutine, etc. If [Modify PV Value and Quantity] is selected, the value in [PV Value Quantity] can be adjusted to modify the number of PV values.
- If you want to use the single PV value mode, first select [Modify PV value] in "Single PV value setting (compatible with K5)", and then you can modify the PV value and the associated interrupt subroutine.
- For other configuration items, please refer to the description above and configure according to actual needs.

8.11.2.2 Using example

In the following example, the setting method of multi-stage PV is adopted. The PV value is expressed as a relative value. The cycle generates CV=PV interrupt. The PV table is directly set in the wizard, and the interrupt subroutine is associated. The previous links such as setting the status register, multi-segment PV table, and calling some programs of HDEF and HSC instructions are omitted. After setting the wizard, you can directly program, which is easy to use.

The following examples illustrate how to use the wizard to configure the high-speed counter and write related interrupt programs.

1)Use the wizard

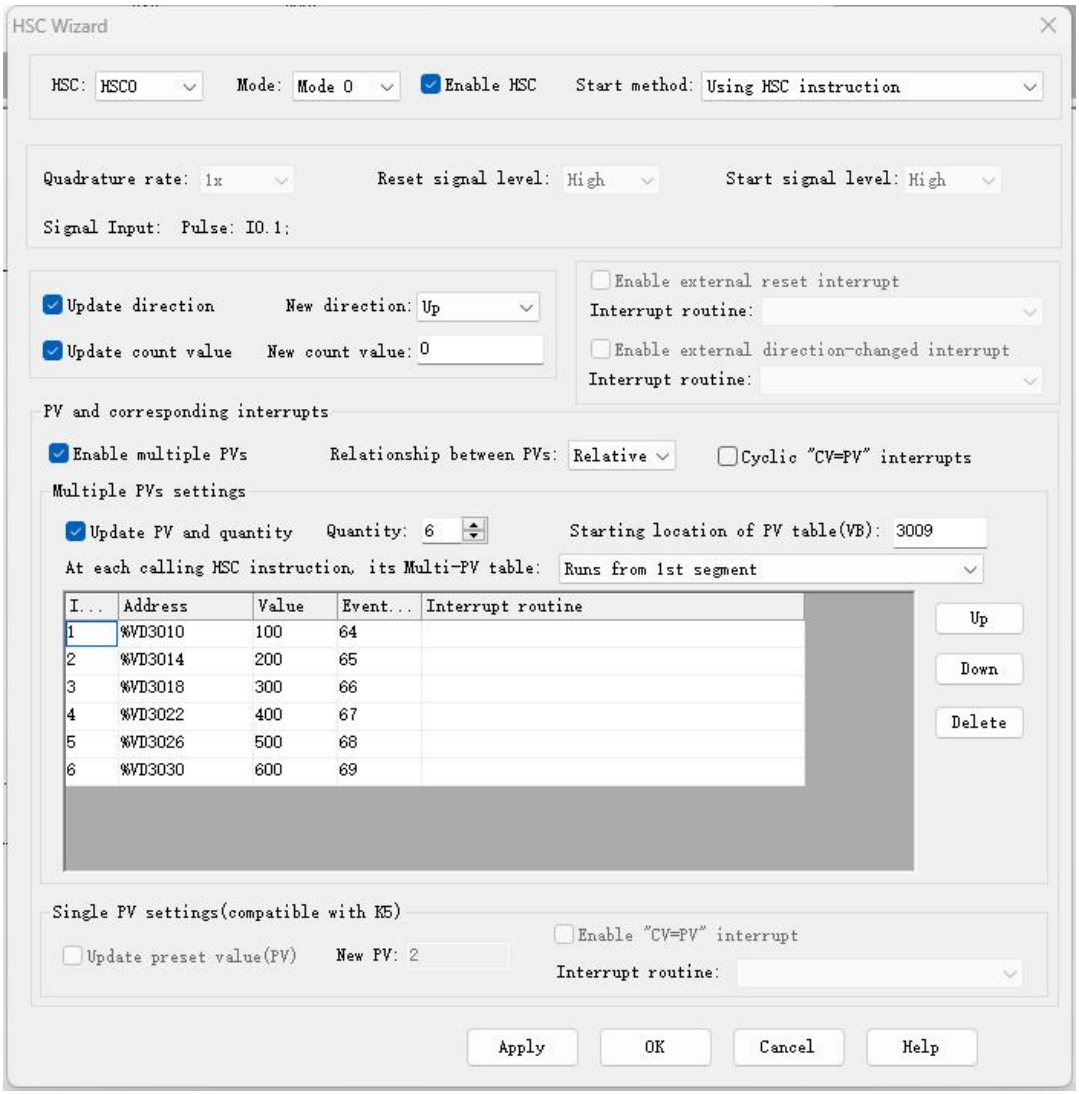
Use counter HSC0, mode 0 (single-phase up/down counter with internal direction control), and enable the counter.

Select "Start after PLC is powered on" as the startup mode, that is, there is no need to call the HSC instruction to start the counter.

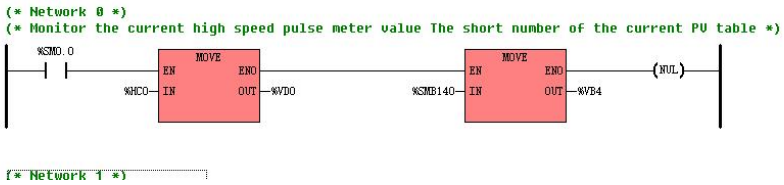
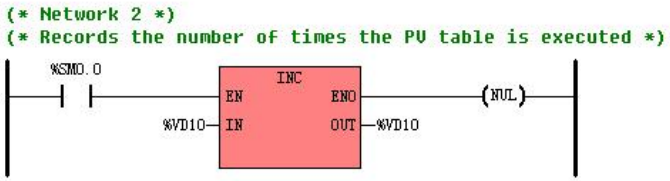
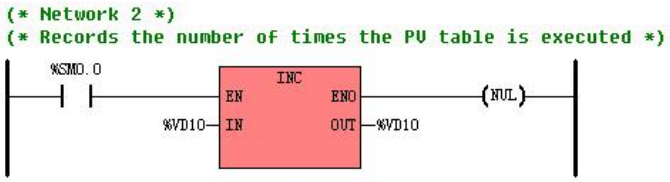
Set the initial count direction to increase, and the current count value is 0 (that is, counting from 0).

Enable the PV interrupt function and set the associated interrupt subroutine of the PV table (the interrupt subroutine needs to be established in advance to be associated). For the description of PV interrupt, please refer to 8.8 Use of PV Interrupt.

After completing the above configuration, click the [OK] button to exit, and the configuration information of HSC0 will be saved in the user engineering program. After the user downloads the program and the PLC runs again, the above configuration will take effect immediately, and HSC0 will start according to the configuration.

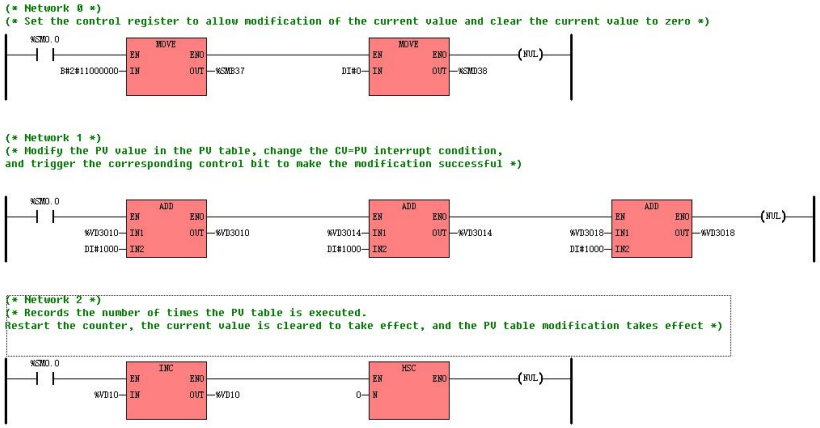


2)Write the required interrupt program (which has been associated with the count interrupt in the wizard) and monitor the high-speed count value in the MAIN program

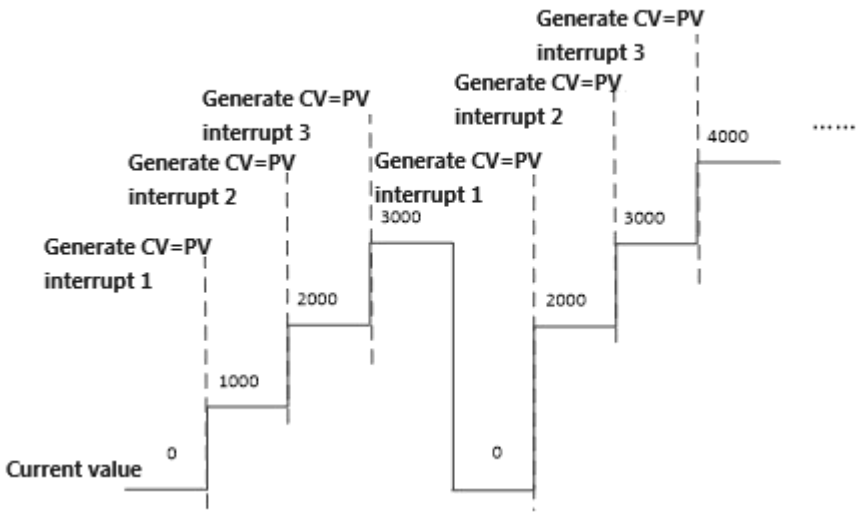
LD	MAIN 
LD	Interrupt subroutine 1: counts reach first segment of the PV table 
	Interrupt subroutine 2: The second segment of the PV table counts to 

Interrupt subroutine 3: The third segment of the PV table is counted, the current count value is cleared, and the PV table is modified to restart the counter to take effect.

Note: The wizard only sets the initial settings of the high-speed counter. If you need to modify the high-speed counter, you need to set the corresponding registers in the program and restart the high-speed counter to take effect.



Timing diagram as shown below:



Chapter 9 Use of high-speed pulse output function

This chapter mainly introduces the high-speed pulse output function of Kinco-K series PLC, including the usage and programming application of pulse output instruction, wiring of output terminals, precautions and related registers, etc.

9.1 Function Overview

Kinco-K series PLC has high-speed pulse output with two or more channels, and the maximum output frequency of each channel is not the same. By using different instruction programming methods, functions such as single-segment, multi-segment pulse train output, pulse width modulation, forward and reverse output, and position control can be realized. All channels support PTO (pulse train) and PWM (pulse width modulation) output.

The detailed configuration table of pulse output is as follows:

PLC Model	No. of Channel	Maximum output frequency				Output
		Q0.0	Q0.1	Q0.4	Q0.5	
K5 Series	2	200K	200K			PNP
K2 Series (K204ET)	3	200K	200K	200K		
K2 Series (K205)	3	50K	10K	10K		
K2 Series (K209EA)	3	200K	200K	10K		
K2 Series (K209M)	3	200K	200K	200K		
KS Series (KS101M has no output port for High-speed pulse)	4	200K	200K	200K	10K	
KW Series	2	200K	200K			
MK Series	4	50K	50K	50K	10K	
K6 Series	4	200K	200K	200K	10K	

Note: The CPU module of relay output model (the last digit of the order number is "R", such as K205-16DR) does not support the high-speed pulse output function!

The maximum output frequency of Q0.0, Q0.1, and Q0.4 channels requires that the load resistance

be no greater than 3KΩ..

9.2 Types of high-speed pulse output instructions

The instruction set of Kinco-K series PLC provides the following two instructions for high-speed output function:

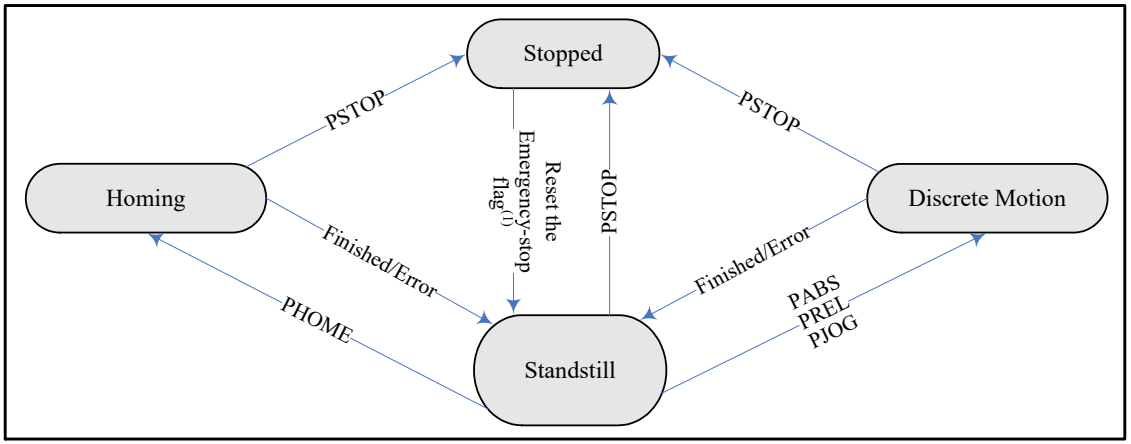
- position control instructions: including PREL (relative motion), PABS (absolute motion), PHOME (back to origin), PJOG (jog), PJOGA (jog with acceleration and deceleration), PREL_M (multi-stage relative motion), PSTOP (emergency stop), and PFLO_F (variable frequency pulse train output). Users can easily realize simple position control functions. **Note: When using the position control instruction, the output pulse frequency cannot be lower than 80Hz!**
- PLS instruction: can realize PTO (pulse train output single or multi-segment) and PWM (pulse width modulation) output function.

9.3 Use of position control Instructions

The position control instructions described in this chapter are a convenient way to use the high-speed pulse output function. Compared with PLS instructions, position control instructions are more suitable for position control applications, and users can use them to easily implement common position control functions. It should be noted that when using the position control instruction, the allowable range of the output pulse frequency is 125Hz~200KHz (the range of some CPUs is 125HZ-10KHZ). For details, please refer to 9.1 Function Overview The maximum output pulse frequency of each series of PLC high-speed pulse output ports , the user-specified output frequency must be within this range.

9.3.1 Model

The figure below is a model diagram of the position control. The user can understand the following information from the figure: when the position control instruction is executed, the state of the corresponding high-speed output channel; in each state, the position control instruction that the PLC allows to execute.



(1) The emergency stop flag bits are SM201.7/ SM231.7/SM251.7/SM221.7 respectively. They correspond to four high-speed pulse output channels in turn. When the PSTOP instruction is executed, this bit will be automatically set to 1. After that, this flag must be reset to continue using motion control instructions. For a description of these special registers, see 9.4.2.2 Control Register and Status Register.

9.3.2 Related variables of position control instructions

9.3.2.1 The direction output channel

For the Position Control instructions, KPLC specifies a direction output channel for each high-speed pulse output channel, and a control bit in the SM area to enable the direction output. Please see the following table.

Pulse output channel	Q0.0	Q0.1	Q0.4	Q0.5
Direction output channel	Q0.2	Q0.3	Q0.6	Q0.7
Direction control bit	SM201.3	SM231.3	SM251.3	SM221.3

The direction output channel is used to output the direction control signal of the motor. The output is 0 during forward rotation and 1 during reverse rotation.

The direction enable control bit is used to disable or enable the use of the corresponding direction output channel. The direction enable control bit has the highest priority. If it is set to disable, the direction control signal will not be output when the position control instruction is executed, and the corresponding output channel can be used as a common DO point.

Note: The number of high-speed pulse output channels supported by each series of PLCs is not the same, so the above corresponding registers are only valid for CPUs with high-speed pulse output channel.

9.3.2.2 Control Register and Status Register

For position control instructions, KPLC allocates a control byte for each high-speed output in the SM area, and users need to pay attention to setting this control byte in the application. In addition, a current value (DINT type) register is also allocated to store the number of pulses that have been output currently (increase during forward rotation and decrease during reverse rotation).

The following table describes these registers in detail.

Q0.0	Q0.1	Q0.4	Q0.5	Description
SMD212	SMD242	SMD262	SMD226	Read only. The current value (increase during forward rotation, decrease during reverse rotation) indicates the number of pulses that have been output currently.
SMD208	SMD238	SMD258	SMD222	RW. The new current value. In conjunction with the corresponding flag bits, it is used to modify the current value.
Q0.0	**Q0.1**	**Q0.4**	**Q0.5**	**Description**
SM201.7	SM231.7	SM251.7	SM221.7	RW. Emergency stop flag bit. If this bit is 1, it means that it is in an emergency stop state and does not execute any position control instructions. This bit is automatically set to 1 when the PSTOP (emergency stop) instruction is executed, and it must be reset by your program..
SM201.6	SM231.6	SM251.6	SM221.6	RW. Used to decide whether to reset the current value 1 - clears the current value to 0. 0 - The current value remains unchanged.
SM201.5	SM231.5	SM251.5	SM221.5	reserve
SM201.4	SM231.4	SM251.4	SM221.4	RW. Used to decide whether to modify the current value 1 - Modify ‘the current value’ to ‘the new current value’. 0 - The current value remains unchanged.
SM201.3	SM231.3	SM251.3	SM221.3	RW. Direction Control Bit. 1 - Disable the direction output, and the direction channel is

				used as a common DO. 0 - Enable direction output.
SM201.0 ~ SM201.2	SM231.0 ~ SM231.2	SM251.0 ~ SM251.2	SM221.0 ~ SM221.2	Reserve

➢ **How to modify the current value**

Each of the 4 high-speed output channels has a current value register, SMD212, SMD242, SMD262 and SMD226, which store the number of pulses output by the corresponding channel. The current value register is a read-only value and is not allowed to be modified directly in the program. If you need to modify the current value, you can take the following methods:

✧ **Method 1**

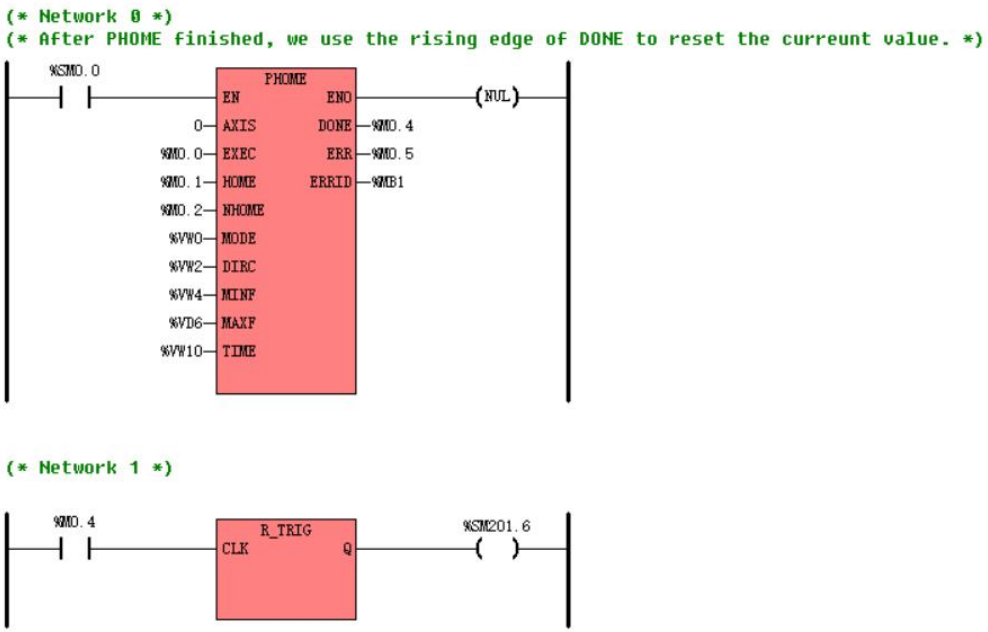
Use the reset control bit to clear the current value to 0.

The reset control bits of the 4 channels are SM201.6, SM231.6, SM251.6 and SM221.6 respectively.

As long as the reset control bit is 1, the PLC will clear the corresponding current value register to 0.

Therefore, the reset control bit only needs to be 1 for one scan period to play a role. When using it, pay attention to avoid the reset control bit being kept at 1 for a long time , and try to avoid resetting the current value during the movement process (including PHOME, PREL, PABS, PJOG, PFLO_F instructions are being executed) to avoid counting errors.

The following takes channel 0 as an example to illustrate how to reset the current value:



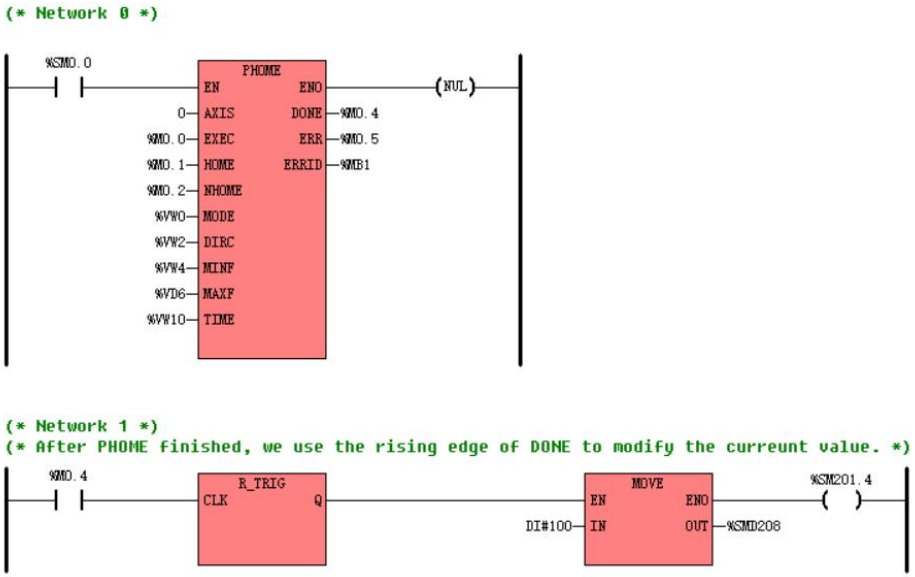
✧ **Method 2**

The current value can be set to any value using the following registers.

Q0.0	Q0.1	Q0.4	Q0.5	Description
SMD208	SMD238	SDM258	SMD222	RW. The new current value. In conjunction with the corresponding flag bits, it is used to modify the current value.
SM201.4	SM231.4	SM251.4	SM221.4	RW. Used to decide whether to modify the current value 1 - Modify 'the current value' to 'the new current value'. 0 - The current value remains unchanged.

Take channel 0 as an example to illustrate the usage method: if SM201.4 is 0, the current value SMD212 keep unchanged. If SM201.4 is 1, the value in SMD208 is assigned to SMD212. Pay attention to avoid modifying the current value register during the movement process (including the PHOME, PREL, PABS, PJOG, PFLO_F instructions are being executed) to avoid counting errors.

The following takes channel 0 as an example to illustrate how to modify the current value:



9.3.3 Position Control Instruction

9.3.3.1 Overview of position control Instructions

The following instructions are located in the [position control] group of the instruction set.

Name	Description
PHOME	back to origin
PABS	absolute movement
PREL	relative motion
PJOG	jog
PJOGA	Jog with acceleration and deceleration
PSTOP	emergency stop
PFLO_F	Variable frequency pulse train output
PREL_M	Multi-segment relative motion

1) Precautions

1、CPU output frequency range limit

- When using the position control instruction, the output pulse frequency cannot be lower than 125Hz and cannot be higher than the maximum output frequency of the CPU, otherwise the CPU will not be able to output pulses normally. If the output frequency is lower than the lowest frequency instruction of 125HZ or higher than the maximum frequency of 200K HZ, the execution of the instruction will directly report an error and output an error code, but it should be noted that some economical CPUs (such as K205, all-in-one series, etc.) cannot meet the hardware specifications of their own CPUs. 200K. At this time, the instruction will not report an error, but in actual use, the highest frequency specified by the CPU should be given priority (see 9.1 Function Overview for details), otherwise the pulse output will be wrong!

2、How to use the emergency stop flag bit

- KPLC has at most four high-speed pulse output channels, and the corresponding four emergency stop flags are SM201.7, SM231.7, SM251.7, SM221.7. When the PSTOP instruction is executed, this bit will be automatically set to 1, indicating that it is in an emergency stop state. At this time, it does not execute any position control instructions. If the user wants to execute the Position Control instruction again, this bit needs to be cleared to 0 in the program.

3、How to use the direction enable control bit

- The four high-speed pulse output channels of KPLC correspond to the four direction control signals, which can be used to disable or allow the use of the corresponding direction output channels through the direction enable control bit. If disabled, the direction output channel can be used as a common DO. For details, see chapter 9.3.2.1 Motor Direction Control Signal.

4、How to clear the pulse count value?

- For details, see chapter 9.3.2.2 Control Register and Status Register.

5、How to modify the maximum output frequency in the Position Control instruction?

- PREL and PABS instructions are not allowed to modify the output frequency MINF and MAXF values during operation, even if they are modified, they will not take effect.

- PHOME instruction will read the MAXF value in real time after running to the highest frequency segment and before the homing and origin signals are generated, and automatically calculate the number of acceleration or deceleration segments according to the new frequency value. After that, it will accelerate or decelerate to a new frequency value and then maintain a constant speed output.
- During the execution of the PJOG instruction, the MAXF value will be read in real time, and the frequency of the output pulse will be adjusted according to the new frequency value.
- During the execution of the PJOGA instruction, the effect of real-time adjustment of the output pulse frequency cannot be achieved by modifying the maximum frequency (MAXF) value.

- PFLO_F provides a variable frequency output at any time. See the instruction introduction for details.
- PREL_M users can customize the PV curve (speed-position relationship) according to the actual situation, and can define up to 8 segments, see the instruction introduction for details.

2) Instruction output parameter ERRID

The Position control instruction may generate a non-fatal error during execution. At this time, the CPU will generate an error code and output it to the ERRID parameter of the instruction. The following table lists these error codes and their descriptions.

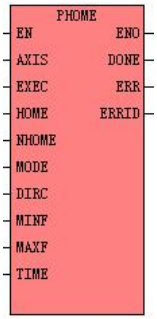
Error Code	Description
0	Normal
1	The acceleration/deceleration time is too short or the initial speed is too low, causing the initial pulse period to exceed the time per segment.
2	If the initial speed MINF exceeds the maximum allowable speed (200KHz), some CPUs themselves cannot reach 200K with the hardware limit, and the instruction will not report an error, but the actual pulse output specification in 9.1 Function Overview shall prevail.
3	The initial speed MINF is lower than the minimum allowable speed (125Hz).
4	The number of pulses required for acceleration and deceleration exceeds the total number of pulses.
5	The initial speed MINF exceeds the maximum speed MAXF.
6	In the PREL_M instruction, the segment value is 0 or exceeds the maximum allowed number of segments.
7	In the PREL_M instruction, the direction is wrong, that is, the running direction of all segments is not exactly the same.

8	In the PREL_M instruction, the address of various parameter input is wrong.
9	In the PREL_M instruction, there are multiple acceleration/deceleration processes between two adjacent segments, that is, the value of ADTIM2 of the nth segment and the value of ADTIM1 of the n+1th segment are not 0.
10	In the PREL_M instruction, it is impossible to assign all the acceleration/deceleration processes successfully according to the value of each input parameter.
11	An exception occurred during execution of the PREL_M instruction.
12	This channel has a pulse output instruction being executed, so the PREL_M instruction cannot be started.
13	The emergency stop flag of this channel is 1, so the PREL_M instruction cannot be started.

9.3.3.2 Position control instruction

9.3.3.2.1 PHOME (back to origin)

- Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to
LD	PHOME			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	PHOME	PHOME *AXIS, EXEC, HOME, NHOME, MODE, DIRC, MINF, MAXF, TIME, DONE, ERR, ERRID*	U	

Parameter	Input/outpour	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
HOME	Input	BOOL	I 、Q、V、M、L、SM、RS、SR
NHOME	Input	BOOL	I 、Q、V、M、L、SM、RS、SR
MODE	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、Constant
DIRC	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、

			Constant
MINF	Input	WORD	I、Q、M、V、L、SM、Constant
MAXF	Input	DWORD	I、Q、M、V、L、SM、Constant
TIME	Input	WORD	I、Q、M、V、L、SM、Constant
DONE	Output	BOOL	Q、M、V、L、SM
ERR	Output	BOOL	Q、M、V、L、SM
ERRID	Output	BYTE	Q、M、V、L、SM

MODE,DIRC,MINF,MAXF,TIME, Must be both constant types or both memory types

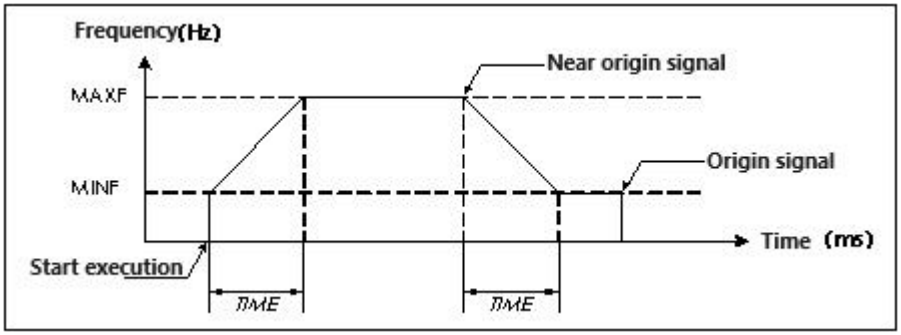
The following table describes the effect of each parameter in detail.

Parameter	Description
AXIS	The high-speed output channel used. 0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.
EXEC	If the rising edge transition of EXEC is detected, the PHOME instruction is triggered to execute.
HOME	Origin input signal.
NHOME	Near-origin input signal.
MODE	The control method of returning to the origin: 0 means using near origin and origin input signal for control; 1 means using only origin input signal for control
DIRC	The running direction of the motor: 0 means forward rotation; 1 means reverse rotation. For the output of the direction control signal, please refer to the description in 9.4.2.1 Motor Direction Control Signal.
MINF	The initial speed (ie initial frequency) of the output pulse, unit: Hz. MINF is not allowed to go below 125Hz, nor to exceed MAXF.
MAXF	The maximum speed (ie the highest frequency) of the output pulse, unit: Hz. The allowable range of MAXF is 125Hz to the maximum pulse frequency that the CPU allows to output. MAXF must be greater than or equal to MINF.
TIME	Acceleration/deceleration time, unit: ms. This instruction adopts the same acceleration time and deceleration time. Acceleration time is the time required to accelerate from MINF to MAXF, and deceleration time is the time required to decelerate from MAXF to MINF.
DONE	completion flag. When the instruction is executed normally, DONE jumps from 0 to 1.
ERR	Error flag bit. This flag is set to 1 if an error occurs during instruction execution.
ERRID	error code.

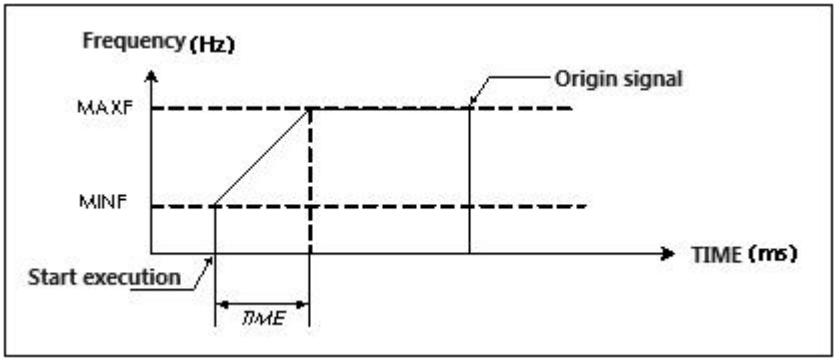
The PHOME instruction uses the near origin and origin input signals to control the return to the origin, and the parameter MODE defines the control method:

① If the control is carried out by using the near-origin and the origin signal, the deceleration starts when the near-origin signal is detected, and the pulse output stops when the origin signal is detected. Timing diagram is

as follows:



◴ If only the origin signal is used for control, the pulse output will be stopped when the origin signal is detected. Timing diagram is as follows:



When the PHOME instruction is executed, if DIRC is set to forward rotation, the current pulse count value will increase, and if DIRC is set to reverse rotation, the current pulse count value will decrease. **Note: When the back-to-origin movement is completed, the current pulse count value register is not automatically cleared, and the user needs to change the current value according to the actual requirements. See the description in 9.3.2.2 Control Register and Status Register for details.**

- LD

When EN is 1, if the rising edge of EXEC input terminal is detected, the instruction is triggered to execute.

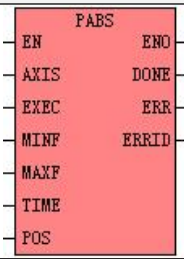
- IL

  When the value of CR is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to be executed.

  The execution of this instruction does not affect the CR value.

9.3.3.2.2 PABS (Absolute movement)

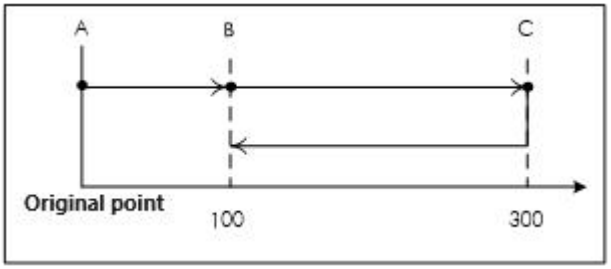
➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to
LD	PABS			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	PABS	PABS AXIS, EXEC, MINF, MAXF, TIME, POS, DONE, ERR, ERRID	U	

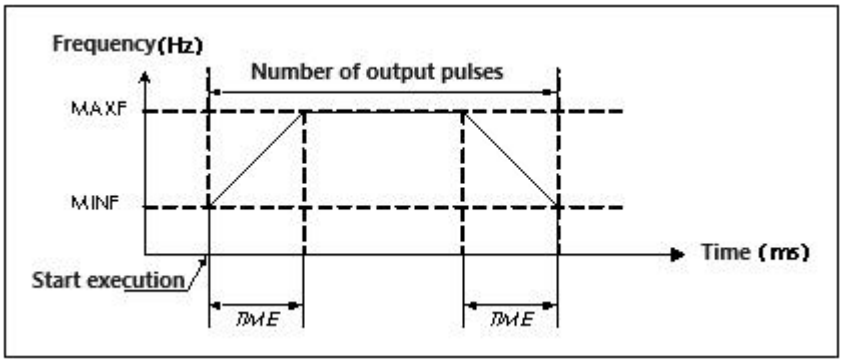
Parameter	Input/outpour	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
MINF	Input	WORD	I、Q、M、V、L、SM、constant
MAXF	Input	DWORD	I、Q、M、V、L、SM、constant
TIME	Input	WORD	I、Q、M、V、L、SM、constant
POS	Input	DINT	I、Q、M、V、L、SM、HC、constant
DONE	Output	BOOL	Q、M、V、L、SM
ERR	Output	BOOL	Q、M、V、L、SM
ERRID	Output	BYTE	Q、M、V、L、SM

MINF,MAXF,TIME,POS, Must be both constant type or memory type at the same time

The following table describes the effect of each parameter in detail.

Parameters	Description
AXIS	The high-speed output channel used. 0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.
EXEC	If the rising edge transition of EXEC is detected, the PABS instruction is triggered to execute.
MINF	The initial speed (ie initial frequency) of the output pulse, unit: Hz. MINF is not allowed to go below 125Hz, nor to exceed MAXF.
MAXF	The maximum speed (ie the highest frequency) of the output pulse, unit: Hz. The allowable range of MAXF is 125Hz to the maximum pulse frequency that the CPU allows to output. MAXF must be greater than or equal to MINF.
TIME	Acceleration/deceleration time, unit: ms. This instruction adopts the same acceleration time and deceleration time. Acceleration time is the time required to accelerate from MINF to MAXF, and deceleration time is the time required to decelerate from MAXF to MINF.
POS	The target value, that is, the number of pulses required between the origin and the position to be moved. As shown in the figure below, if the object is moved from A to B, the target value should be set to "100"; if it is moved from B to C, the target value should be set to "300"; At B, the target value is set to "100". 
DONE	completion flag. When the instruction is executed normally, DONE jumps from 0 to 1.
ERR	Error flag bit. This flag is set to 1 if an error occurs during instruction execution.
ERRID	error code.

The PABS instruction adopts absolute Position and outputs pulses according to the difference between the current value and the target value POS. The difference between the current value and the target value is the number of output pulses. The Timing diagram executed by the PABS instruction is as follows:



If the direction enable control bit is set to 0, the PABS instruction will output the direction control signal of the motor in the corresponding direction output channel: when the target value > the current value, the forward rotation signal will be output, and the current pulse count value will increase. When the target value is less than the current value, the reverse signal is output, and the current pulse count value will decrease.

- LD

  When EN is 1, if the rising edge of EXEC input terminal is detected, the instruction is triggered to execute.

- IL

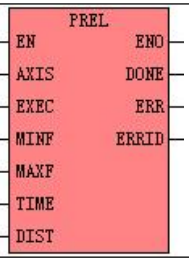
  When the value of CR is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to be executed.

  The execution of this instruction does not affect the CR value.

9.3.3.2.3 PREL (relative motion)

➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to

LD	PREL			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	PREL	PREL AXIS, EXEC, MINF, MAXF, TIME, DIST, DONE, ERR, ERRID	U	

Parameter	Input/outpour	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
MINF	Input	WORD	I、Q、M、V、L、SM、constant
MAXF	Input	DWORD	I、Q、M、V、L、SM、constant
TIME	Input	WORD	I、Q、M、V、L、SM、constant
DIST	Input	DINT	I、Q、M、V、L、SM、HC、constant
DONE	Output	BOOL	Q、M、V、L、SM
ERR	Output	BOOL	Q、M、V、L、SM
ERRID	Output	BYTE	Q、M、V、L、SM

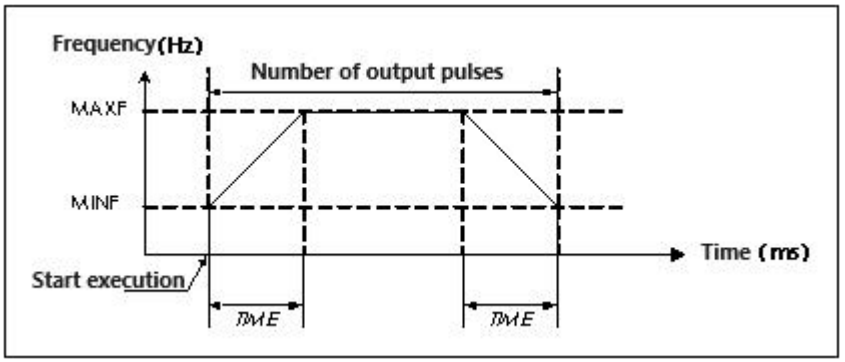
MINF, MAXF, TIME, DIST Must be both constant type or memory type at the same time

The following table describes the function of each parameter in detail

Parameter	Description
AXIS	The high-speed output channel used. 0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.
EXEC	If the rising edge transition of EXEC is detected, the PREL instruction is triggered to execute.
MINF	The initial speed (ie initial frequency) of the output pulse, unit: Hz. MINF is not allowed to go below 125Hz, nor to exceed MAXF.
MAXF	The maximum speed (ie the highest frequency) of the output pulse, unit: Hz. The allowable range of MAXF is 125Hz to the maximum pulse frequency that the CPU allows to output. MAXF must be greater than or equal to MINF.
TIME	Acceleration/deceleration time, unit: ms. This instruction adopts the same acceleration time and deceleration time. Acceleration time is the time required to accelerate from MINF to MAXF, and deceleration time is the time required to decelerate from MAXF to MINF.

DIST	The amount of movement, that is, the number of pulses required between the current position and the position to be moved. As shown in the figure below, if the object is moved from A to B, the movement amount should be set to "100"; if it is moved from B to C, the movement amount should be set to "200"; To B, the movement amount is set to "-200".
DONE	completion flag. When the instruction is executed normally, DONE jumps from 0 to 1.
ERR	Error flag bit. This flag is set to 1 if an error occurs during instruction execution.
ERRID	error code.

The PREL instruction adopts relative positioning, and the number of output pulses is the movement amount DIST. The Timing diagram executed by the PREL instruction is as follows:



If the direction enable control bit is set to 0, then the PREL instruction will output the direction control signal of the motor in the corresponding direction output channel: When the movement amount is positive, the forward signal is output, and the current pulse count value will increase; when the movement amount is negative, the reverse signal is output, and the current pulse count value will decrease.

- LD

When EN is 1, if the rising edge of EXEC input terminal is detected, the instruction is triggered to

execute.

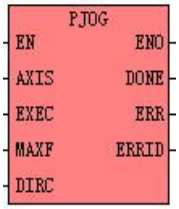
- IL

  When the value of CR is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to be executed.

  The execution of this instruction does not affect the CR value.

9.3.3.2.4 PJOG (Jog move)

➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to
LD	PJOG			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	PJOG	PJOG AXIS, EXEC, MAXF, DIRC, DONE, ERR, ERRID	U	

Parameter	Input/outpour	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
MAXF	Input	DWORD	I、Q、M、V、L、SM、constant
DIRC	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、constant
DONE	Output	BOOL	Q、M、V、L、SM
ERR	Output	BOOL	Q、M、V、L、SM
ERRID	Output	BYTE	Q、M、V、L、SM

MAXF, DIRC Must be both constant type or memory type at the same time

The following table describes the effect of each parameter in detail.

Parameters	Description
AXIS	The high-speed output channel used. 0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.
EXEC	If EXEC is 1, it will continue to output pulses; if it is 0, it will stop outputting.
MAXF	Frequency of output pulse, unit: Hz. The allowable range is 125Hz to the maximum pulse frequency that the CPU allows to output.
DIRC	The running direction of the motor: 0 means forward rotation; 1 means reverse rotation. For the output of the direction control signal, please refer to the description in 9.3.2.1 Motor Direction Control Signal.
DONE	completion flag. When the instruction is executed normally, DONE jumps from 0 to 1.
ERR	Error flag bit. This flag is set to 1 if an error occurs during instruction execution.
ERRID	Error code

If the EXEC input is 1, the PJOG instruction will continuously output a pulse train from the specified channel AXIS, and the frequency is MAXF. If MAXF is a variable, the output frequency value can be changed at any time during the pulse output process. If the EXEC input is 0, the output will be stopped immediately. During the operation of the instruction, the user can modify the output frequency value MAXF, and the PJOG instruction will immediately output the pulse train according to the new MAXF frequency value.

When the PJOG instruction is executed, if DIRC is set to forward rotation, the current pulse count value will increase; if DIRC is set to reverse rotation, the current pulse count value will decrease.

- LD

  When EN is 1, if the EXEC input is 1, the instruction is executed:

- IL

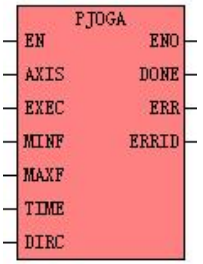
  When the CR value is 1, if the EXEC input is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

9.3.3.2.5 PJOGA (Jog with acceleration and deceleration)

➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to

LD	PJOGA			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	PJOGA	PJOGA AXIS, EXEC, MINF, MAXF, TIME, DIRC, DONE, ERR, ERRID	U	

Parameter	Input/outpour	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
MINF	Input	WORD	I、Q、M、V、L、SM、constant
MAXF	Input	DWORD	I、Q、M、V、L、SM、constant
TIME	Input	WORD	I、Q、M、V、L、SM、constant
DIRC	Input	INT	I、Q、V、M、L、SM、T、C、AI、AQ、constant
DONE	Output	BOOL	Q、M、V、L、SM
ERR	Output	BOOL	Q、M、V、L、SM
ERRID	Output	BYTE	Q、M、V、L、SM

MINF, MAXF, TIME, DIST Must be both constant type or memory type at the same time。

Note: Different from PJOG, PJOGA is a jog instruction with acceleration and deceleration control. Another point is that it is not allowed to modify the value of MAXF to achieve the effect of real-time modification of the running speed during the execution of PJOGA.

The following table describes the effect of each parameter in detail.

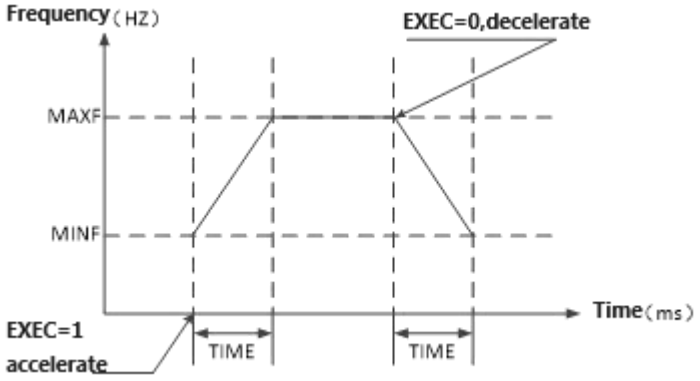
Parameters	Description
AXIS	The high-speed output channel used. 0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.
EXEC	If EXEC is 1, it will continue to output pulses; if it is 0, it will stop outputting.
MINF	The initial speed (ie initial frequency) of the output pulse, unit: Hz. MINF is not allowed to go below 125Hz, nor to exceed MAXF.
MAXF	The allowable range of MAXF is 125Hz to the maximum pulse frequency that the CPU allows to output.

	MAXF must be greater than or equal to MINF.
TIME	Acceleration/deceleration time, unit: ms. This instruction adopts the same acceleration time and deceleration time. Acceleration time is the time required to accelerate from MINF to MAXF, and deceleration time is the time required to decelerate from MAXF to MINF.
DIRC	The running direction of the motor: 0 means forward rotation; 1 means reverse rotation. For the output of the direction control signal, please refer to 9.3.2.1 Motor Direction Control Signal.
DONE	completion flag bit. When the instruction is executed normally, DONE jumps from 0 to 1.
ERR	Error flag bit. This flag is set to 1 if an error occurs during instruction execution.
ERRID	error code.

If the EXEC input is 1, the PJOGA instruction continues to output pulse trains from the specified channel AXIS. First, it goes through an acceleration process to increase from the minimum frequency MINF to the maximum frequency MAXF. The acceleration is determined by the acceleration and deceleration time TIME. If the EXEC input is 0, it will decelerate according to the set acceleration and deceleration time TIME until the output stops at the minimum frequency. If the EXEC input is 1 again during the deceleration process, it will re-enter the acceleration start process from the current speed.

When the PJOGA instruction is executed, if DIRC is set to forward rotation, the current pulse count value will increase; if DIRC is set to reverse rotation, the current pulse count value will decrease.

The Timing diagram executed by the PJOGA instruction is as follows:



- LD

When EN is 1, if the EXEC input is 1, the instruction is executed:

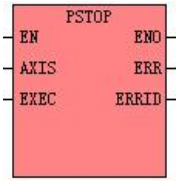
- IL

  When the CR value is 1, if the EXEC input is 1, the instruction is executed.

  The execution of this instruction does not affect the CR value.

9.3.3.2.6 PSTOP (Emergency stop)

➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to
LD	PSTOP			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	PSTOP	PSTOP AXIS, EXEC, ERR, ERRID	U	

Parameter	Input/outpour	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
ERR	Output	BOOL	Q、M、V、L、SM
ERRID	Output	BYTE	Q、M、V、L、SM

The following table describes the effect of each parameter in detail.

Parameters	Description
AXIS	The high-speed output channel used. 0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.
EXEC	If the rising edge transition of EXEC is detected, the PSTOP instruction is triggered to execute.
ERR	Error flag bit. This flag is set to 1 if an error occurs during instruction execution.
ERRID	error code.

The PSTOP instruction is used to immediately stop the pulse output of the AXIS channel, so that the current movement is stopped urgently, and the emergency stop flag is set at the same time. KPLC has at most four high-speed pulse output channels, and the corresponding four emergency stop flags are SM201.7/SM231.7/SM251.7/SM221.7 respectively. **The user needs to use the program to clear the emergency stop flag to 0, otherwise the CPU will not execute any Position control instructions.**

- LD

When EN is 1, if the rising edge of EXEC input terminal is detected, the instruction is triggered to execute.

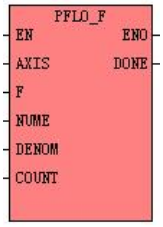
- IL

When the value of CR is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to be executed.

The execution of this instruction does not affect the CR value.

9.3.3.2.7 PFLO_F (Variable frequency pulse train output)

➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to
LD	PFLO_F			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	PFLO_F	PFLO_F AXIS, F, NUME, DENOM,COUNT, DONE	U	

Parameter	Input/outpour	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)

F	Input	DINT	L、M、V、constant
NUME	Input	INT	L、M、V、constant
DENOM	Input	INT	L、M、V、constant
DONE	Output	BOOL	Q、M、V、L
COUNT	Input	DWORD	L、M、V、constant

F, NUME, DENOM, COUNT Must be both constant type or memory type at the same time。

The following table describes the effect of each parameter in detail.

Parameters	Description
EN	enable terminal. If EN is 1, execute following pulse output, otherwise stop pulse output.
AXIS	The high-speed output channel used.0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.
F	Input frequency. Unit: Hz
NUME	Electronic gear molecules that output pulse frequency
DENOM	Electronic gear denominator of output pulse frequency
DONE	completion flag bit. If there is pulse output, it is 0; if pulse output is stopped, it is 1.

If EN is 1, execute the instruction to continuously output pulses from the designated channel AXIS. The output frequency is equal to the input frequency F multiplied by the electronic gear (NUME/DENOM), and a total of COUNT pulses are output. When the number of output pulses reaches COUNT, the output is completed, and the flag bit DONE is set to 1. If the input frequency F is less than 0, it means reverse rotation, and the corresponding direction channel output is 1; if F is greater than 0, it means forward rotation, and the corresponding direction channel output is 0.

Note: If the calculated output frequency is less than the minimum allowable frequency of 125Hz, the PLC will stop the output, and will continue to follow the output when the output frequency exceeds the minimum frequency again. The output frequency is not allowed to exceed the maximum output frequency of the CPU, please set the appropriate input frequency and electronic gear ratio.

- LD

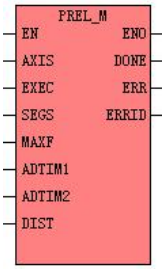
  When EN is 1, the instruction is executed. If the EN input is 0, the output will be stopped immediately.

- IL

When the CR value is 1, the instruction is executed. The execution of this instruction does not affect the CR value.

9.3.3.2.8 PREL_M (Multi-segment relative motion)

- Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to
LD	PREL_M			☑ MK ☑ K6
IL	PREL_M	PREL_M AXIS, EXEC, SEGS, MAXF, ADTIM1, ADTIM1, DIST, DONE, ERR, ERRID	U	

Parameter	Input/output	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
SEGS	Input	WORD	M、V、L、SM、constant
MAXF	Input	DINT	M、V、L、SM
ADTIM1	Input	WORD	M、V、L、SM
ADTIM1	Input	WORD	M、V、L、SM
DIST	Input	DINT	M、V、L、SM
DONE	Output	BOOL	Q、M、V、L、SM
ERR	Output	BOOL	Q、M、V、L、SM
ERRID	Output	BYTE	Q、M、V、L、SM

The following table describes the effect of each parameter in detail.

Parameters	Description
AXIS	The high-speed output channel used. 0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.
EXEC	If the rising edge transition of EXEC is detected, the PREL_M instruction is triggered to execute.
SEGS	The total number of segments (1-8) for the custom curve.

MAXF	The maximum speed (ie the highest frequency) of the output pulse, unit: Hz. The allowable range of MAXF is 125Hz to the maximum pulse frequency that the CPU allows to output. This parameter is the starting address of the frequency of each segment, that is, the address of the first segment, and the subsequent segments are stored continuously in the memory.
ADTIM1	Initial acceleration/deceleration time, unit: ms. 0 means no acceleration/deceleration time. This parameter is the starting address of the initial acceleration/deceleration time of each segment, that is, the address of the first segment, and the subsequent segments are stored continuously in the memory.
ADTIM2	End acceleration/deceleration time, unit: ms. 0 means no acceleration/deceleration time. This parameter is the start address of the end acceleration/deceleration time of each segment, that is, the address of the first segment, and the subsequent segments are stored continuously in the memory.
DIST	Movement amount (including acceleration and deceleration process), unit: pulse. Positive numbers represent the positive direction, and negative numbers represent the negative direction. This parameter is the starting address of the movement amount of each segment, that is, the address of the first segment, and the subsequent segments are stored continuously in the memory.
DONE	completion flag bit. When the instruction is executed normally, DONE jumps from 0 to 1.
ERR	Error flag bit. This flag is set to 1 if an error occurs during instruction execution.
ERRID	error code.

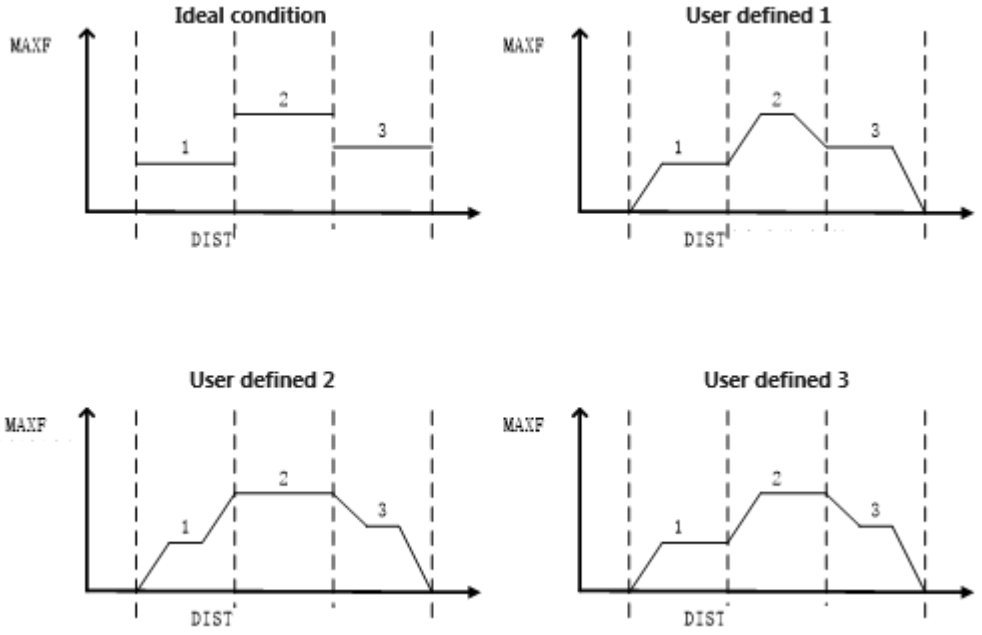
Note: This instruction starts from static by default, and the user can also modify the following SM registers to set the startup speed arbitrarily: Channel 0 is SMD204, Channel 1 is SMD234, Channel 2 is SMD254, and Channel 3 is SMD180. If the startup speed set by the user is less than 100Hz, the default is 100Hz, and the startup speed is not allowed to exceed the maximum speed (MAXF) of the first stage.。

This instruction supports continuous output of 1-8 segment pulse trains. The desired number of segments, as well as the number of pulses, frequency, acceleration time and deceleration time of each segment can be defined in the user program. When this instruction is executed, it will continuously output pulses according to the parameters set by the user, in which the first stage starts from a standstill or the starting speed set by the user, and the speed of the first stage will be used as the starting speed of the second stage, and so on. The speed of each segment can be higher or lower than the speed of the previous segment, and the instruction will automatically judge and generate an acceleration or deceleration process.

All curve segments of this instruction can only be in positive direction or negative direction at the same

time.

Note: Only one acceleration or deceleration process is allowed between two adjacent segments, that is, at least one of the value of ADTIM2 of the nth segment and the value of ADTIM1 of the n+1th segment must be 0. The following figure uses 3 segments as an example to illustrate several output Timing diagrams that can be customized:



- LD

  When EN is 1, if the rising edge of EXEC input terminal is detected, the instruction is triggered to execute.

- IL

  When the value of CR is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to be executed.

The execution of this instruction does not affect the CR value.

- PREL_M Instruction usage example

LD

```
(* Network 0 *)
(* Set the parameters of the first curve: operating frequency 1khz,
 acceleration time 1s, deceleration time none, output 10000 pulses *)
```

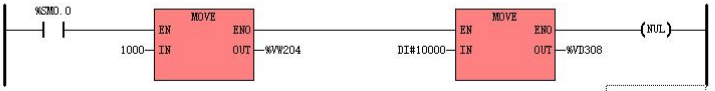
```
(* Network 1 *)
```

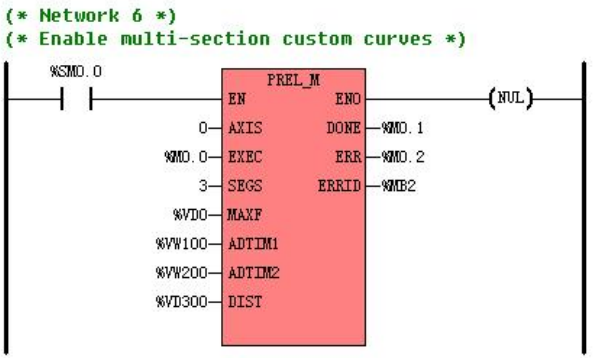
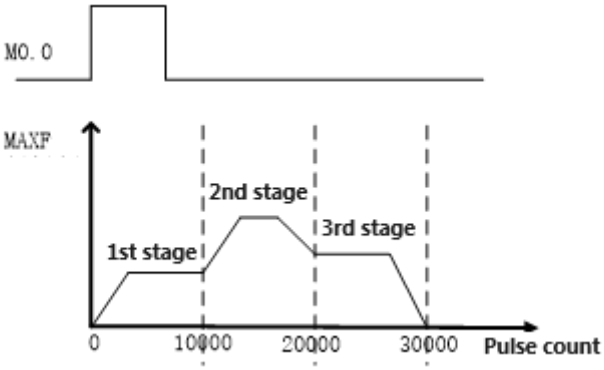
```
(* Network 2 *)
(* Set the parameters of the second curve: running frequency 2khz,
 acceleration time 1s, deceleration time 1s, output 10000 pulses *)
```

```
(* Network 3 *)
```

```
(* Network 4 *)
(* Set the parameters of the third curve: operating frequency 1.2khz,
 acceleration time none, deceleration time 1s, output 10000 pulses *)
```

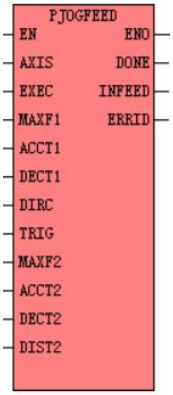
```
(* Network 5 *)
```



<table>
<tr><td>LD</td><td> 
</td></tr>
<tr><td>Description</td><td>Set a three-segment curve, the rising edge of M0.0 starts to execute this instruction, and runs according to the set speed-position relationship.
 
</td></tr>
</table>

9.3.3.2.9 PJOGFEED (break fixed length)

- Instruction and its operand description

	Name	Instruction format	Affected CR value	Adopt to
LD	PJOGFEED			☑ MK ☑ K6

Parameter	Input/outpour	Data type	Allowed memory area
AXIS	Input	INT	constant (0、1、2、3)
EXEC	Input	BOOL	I、Q、V、M、L、SM
MAXF1	Input	DWORD	M、V、L
ACCT1	Input	WORD	M、V、L
DECT1	Input	WORD	M、V、L
DIRC	Input	INT	M、V、L
TRIG	Input	BOOL	I、Q、V、M、L、SM
MAXF2	Input	DWORD	M、V、L
ACCT2	Input	WORD	M、V、L
DECT2	Input	WORD	M、V、L
DIST2	Input	DINT	M、V、L
DONE	Output	BOOL	Q、M、V、L
INFEED	Output	BOOL	Q、M、V、L
ERRID	Output	BYTE	Q、M、V、L

The following table describes the effect of each parameter in detail.

Parameters	Description
AXIS	The high-speed output channel used. 0 means using Q0.0, 1 means using Q0.1, 2 means using Q0.4, 3 means using Q0.5.

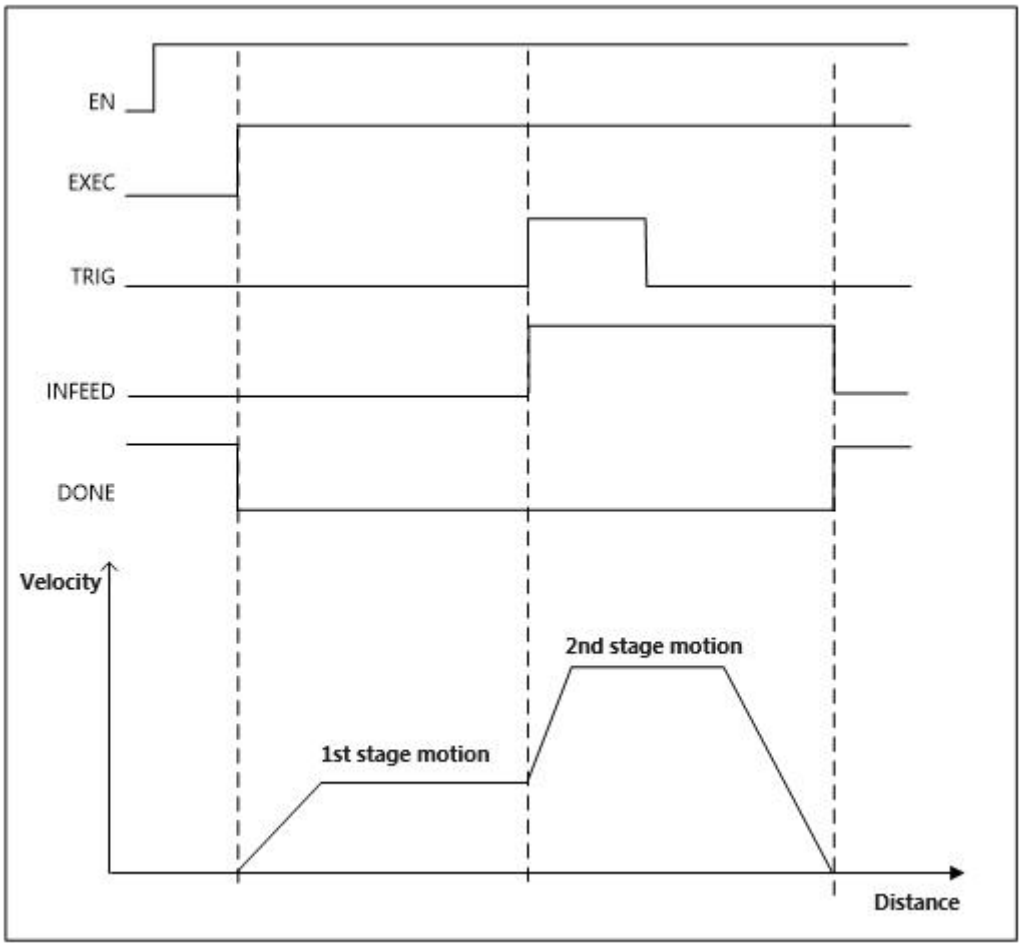
EXEC	The rising edge of EXEC triggers the execution of this instruction; the falling edge of EXEC triggers the movement to stop.
MAXF1	Maximum speed for the 1st movement (speed mode). Unit: Hz.
ACCT1	Acceleration time for the 1st movement (speed mode). Unit: ms.
DECT1	Deceleration time for the 1st movement (speed mode). Unit: ms.
DIRC	Movement direction, 0 means forward rotation, non-0 value means reverse rotation.
TRIG	Trigger signal for the second movement. In the process of executing the first movement, if the rising edge of TRIG is detected, this instruction immediately starts to execute the second movement (fixed-length movement).
MAXF2	The maximum speed for the 2nd movement (fixed-length movement). Unit: Hz.
ACCT2	Acceleration time for the second movement (fixed-length movement). Unit: ms.
DECT2	Deceleration time for the second movement (fixed-length movement). Unit: ms.
DIST2	The movement amount of the second movement (fixed-length movement), that is, the number of output pulses.
DONE	completion flag bit. When the instruction is executed normally, DONE jumps from 0 to 1.
INFEED	Fixed-length motion flag. 1 means that the fixed-length movement is being executed; 0 means that the fixed-length movement is not executed or has been executed.
ERRID	error code

If EN is 1, this instruction is scanned. When the rising edge of EXEC is detected, this instruction latches input parameters such as MAXF1, ACCT1, DECT1, MAXF2, ACCT2, DECT2, DIST2, and immediately starts executing the first motion (speed mode). Accelerate to the maximum speed MAXF1 within the acceleration time ACCT1 and maintain a constant speed at MAXF1. During the first movement, if the rising edge of TRIG is detected, this instruction immediately starts to execute the second movement (fixed-length movement), and stops the movement after completing the specified DIST2 pulses from the current position. The acceleration time of the second movement is ACCT2, the maximum speed is MAXF2, and the deceleration time is DECT2. When the second movement starts, the INFEED parameter is output as 1 immediately, and after the second movement is completed, the INFEED parameter is output as 0 immediately. The direction of both movements is specified by the parameter DIRC, 0 means forward rotation, non-zero value means reverse rotation.

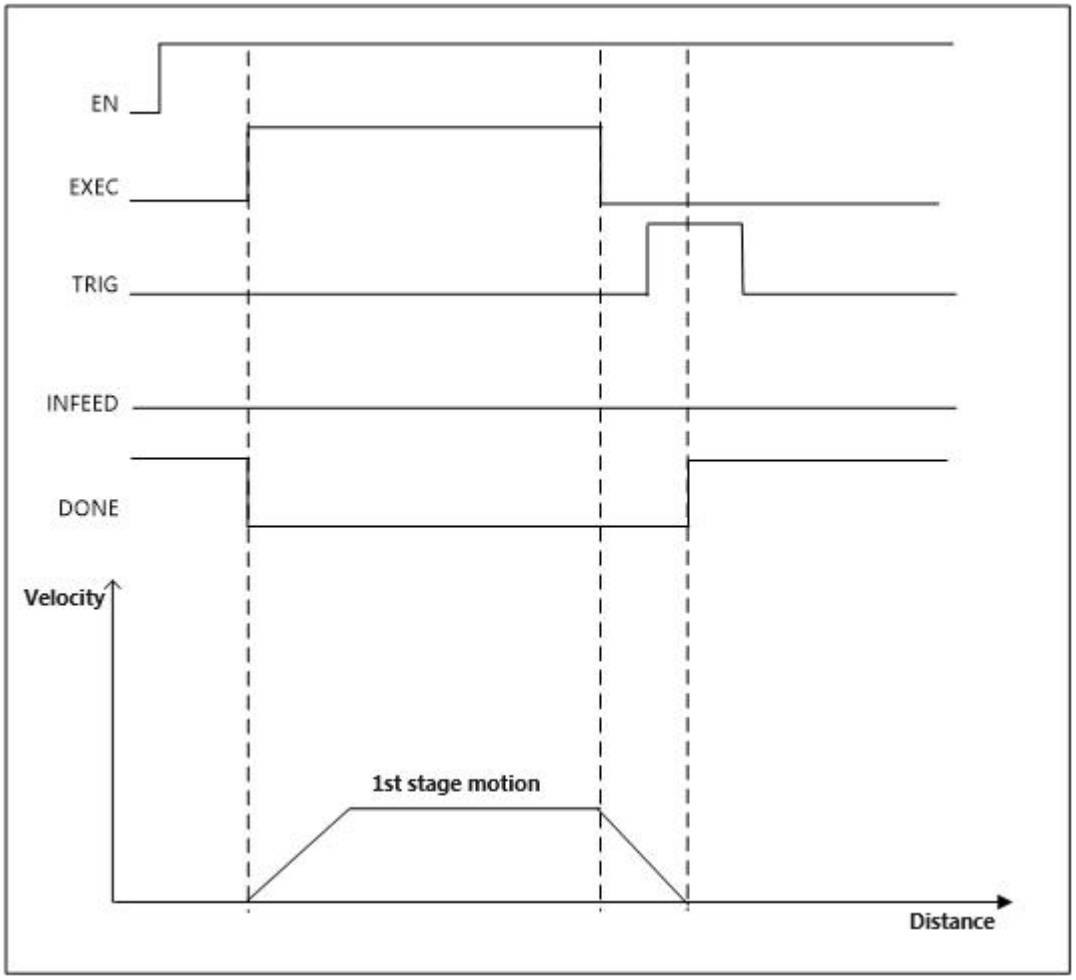
When the falling edge of EXEC is detected while EN is 1, or when EN becomes 0, this instruction stops motion immediately. At this time, if the first movement is being executed, go to the deceleration section to decelerate and stop at the deceleration time DECT1; if the second movement is being executed, go to the deceleration section and decelerate and stop at the deceleration time DECT2.

Timing diagrams of movements in common situations are listed below.

- **PJOGFEEDTiming diagram --- A complete execution process**

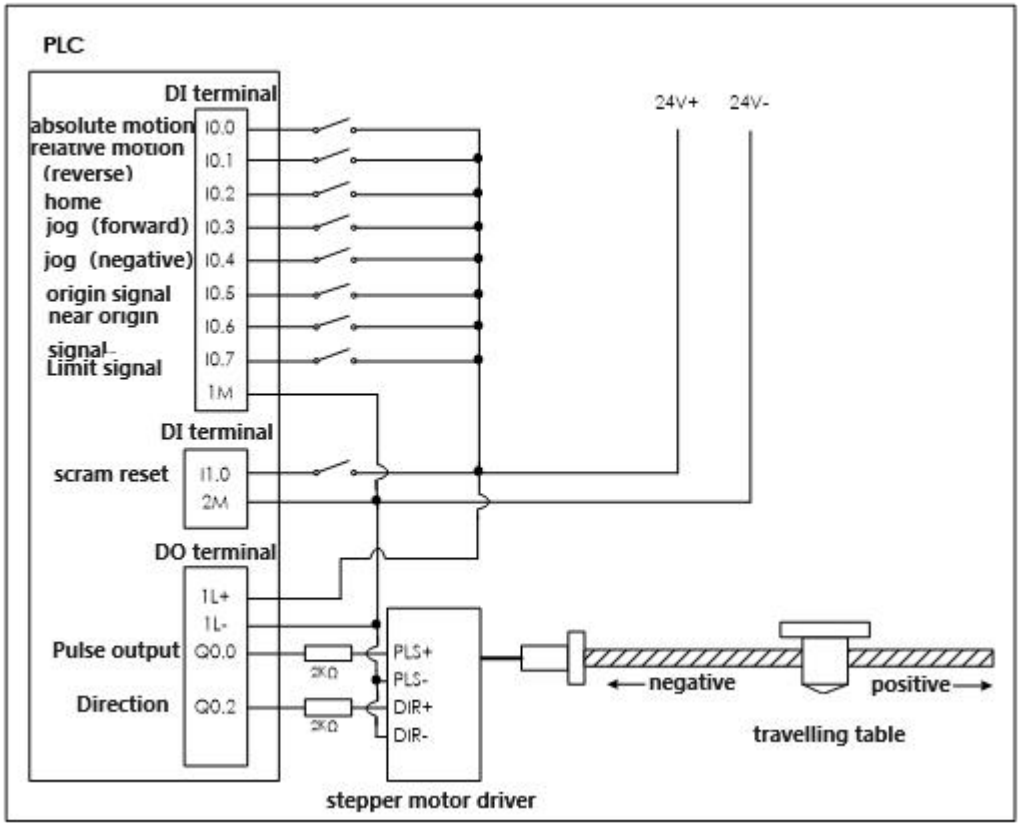


- **PJOGFEEDTiming diagram --- During the 1st movement, EXEC becomes 0 causing the movement to stop**



9.3.3.3 Example of PositionControl Instructions

- Wiring diagram

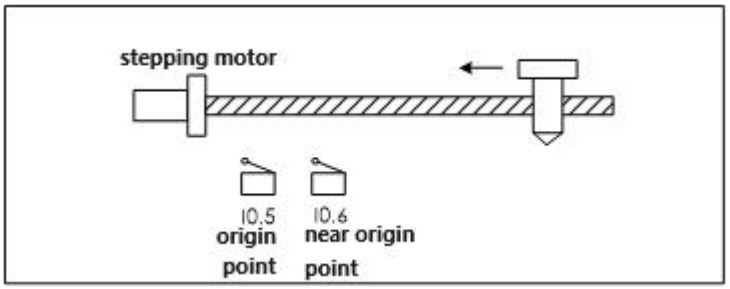
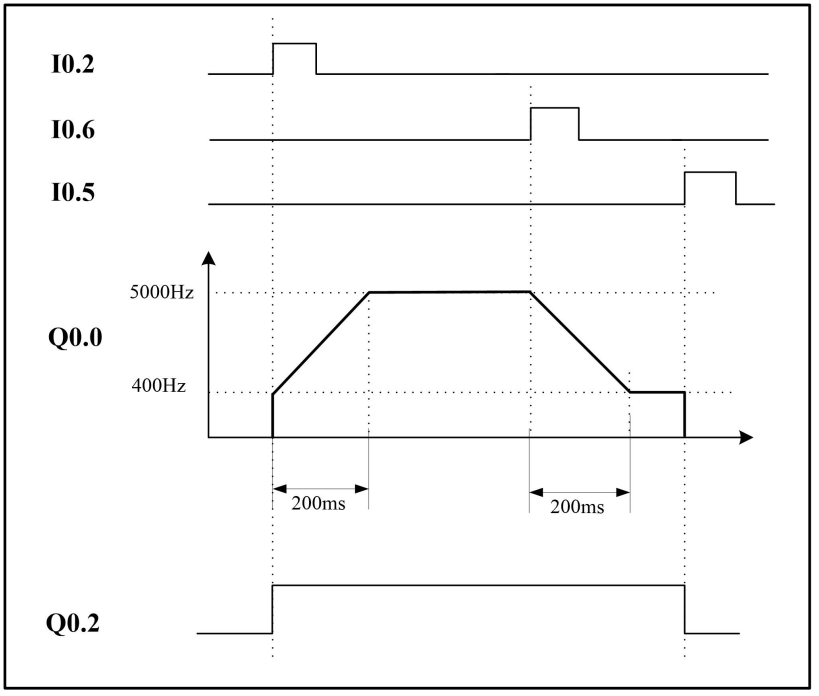


Below we will describe the use of PHOME, PREL, PABS, PJOG and PSTOP instructions based on this system.

This example uses the Q0.0 output channel, please pay attention to the use of related special registers, see 9.3.2.2 Control Register and Status Register for details

- Back to origin

1、The "return to origin" signal connected to I0.2 is used to start the movement back to the origin. It should be noted that the current value register (SMD212) is not automatically cleared when the back-to-origin movement is completed, and the user needs to change the current value according to actual requirements.

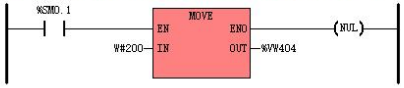
Description	Assume that the current workbench is in the following state:  If the rising edge of I0.2 is detected, the pulse will be output at Q0.0, the output of Q0.2 is 1, which means reverse rotation, and the motor will find the origin in the reverse direction. 

LD

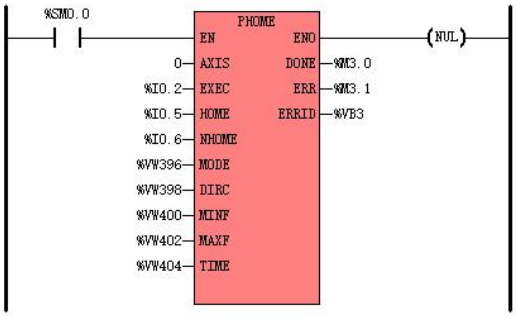
```
(* Network 0 *)
(* Use near origin signal and origin signal at the same time; Control motor reversal *)
```

```
(* Network 1 *)
(* Set the initial frequency and running frequency *)
```

```
(* Network 2 *)
(* Set the acceleration and deceleration time *)
```



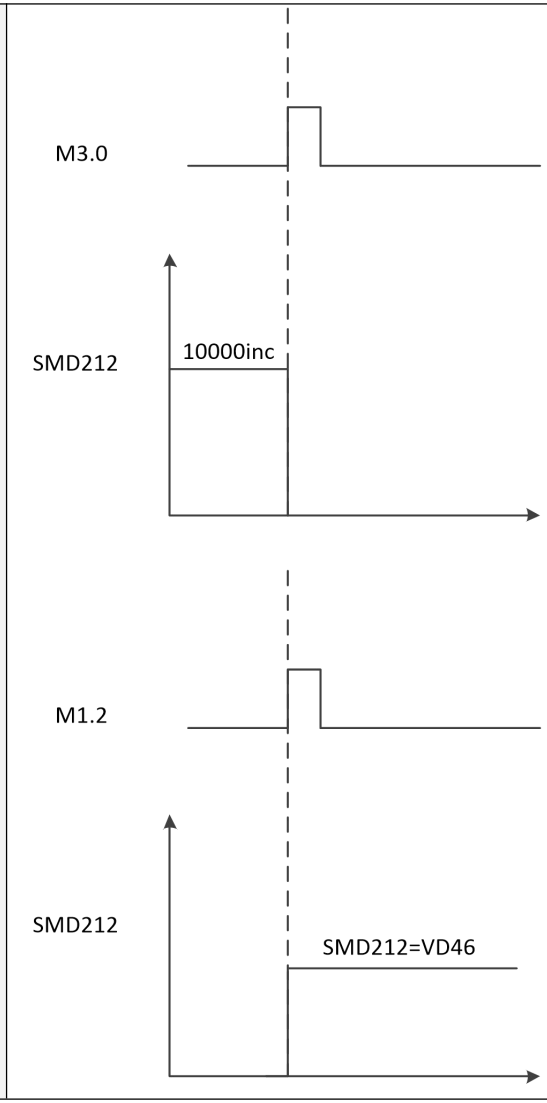

```
(* Network 3 *)
(* Call back to the original instruction *)
```

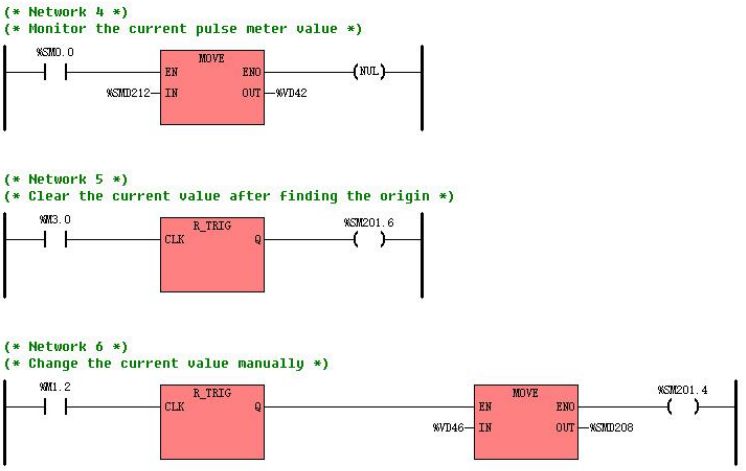


IL	(* Network 0 *) LD %SM0.1 MOVE 0, %VW396(* Both near origin and origin signals are used *) MOVE 1, %VW398(* Reverse the motor *) MOVE W#400, %VW400(* Set initial frequency *) MOVE W#5000, %VW402(*Set operating frequency *) MOVE W#200, %VW404(* Set up/down time *) (* Network 2 *) LD %SM0.0 PHOME 0, %I0.2, %I0.5, %I0.6, %VW396, %VW398, %VW400, %VW402, %VW404, %M3.0, %M3.1, %VB3

2、After the homing is completed, the current pulse count value can be cleared or given the desired value (of course, the modification of the current pulse count value is not only possible after the homing).

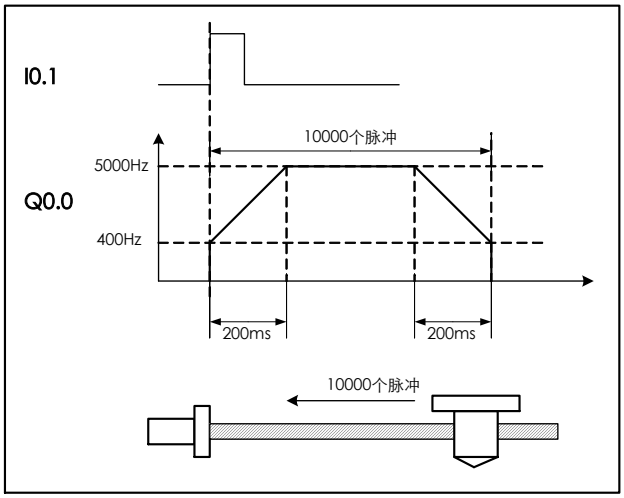
Description	Assuming that the current pulse count value is 10000 before the origin is found, it will be cleared instantly after the origin is found. M1.2 is to manually modify the current value, please watch this Timing diagram with the following program.

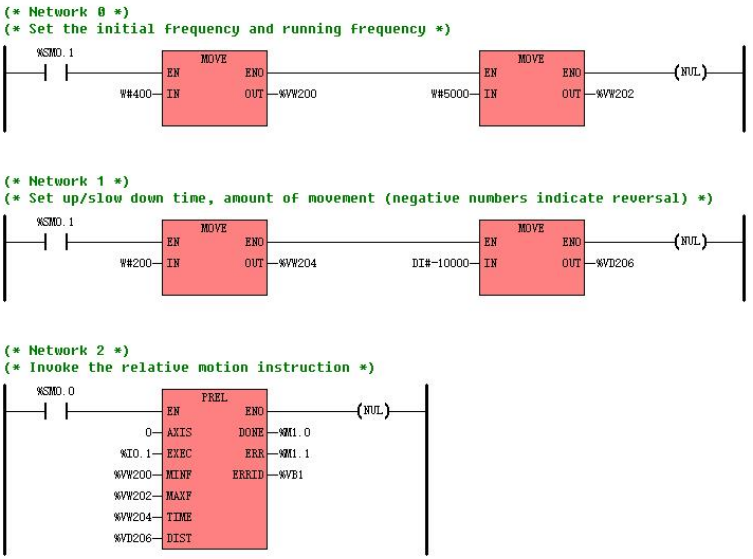

M3.0
SMD212
10000inc
M1.2
SMD212
SMD212=VD46

<table>
<tr><td>LD</td><td>Monitor the current pulse count value. After finding the origin, the current value is automatically cleared. You can also use M1.2 to cooperate with the corresponding register to manually modify the current value to the value you want to set.
 
</td></tr>
<tr><td>IL</td><td>(* Network 4 *)
(*Monitor the current pulse meter value*)
LD %SM0.0
MOVE %SMD212, %VD42
(* Network 5 *)
(*Clear the current value after finding the origin*)
LD %M3.0
R_TRIG
ST %SM201.6
(* Network 6 *)
(*Change the current value manually*)
LD %M1.2
R_TRIG
MOVE %VD46, %SMD208
ST %SM201.4</td></tr>
</table>

- Relative movement (reversal)

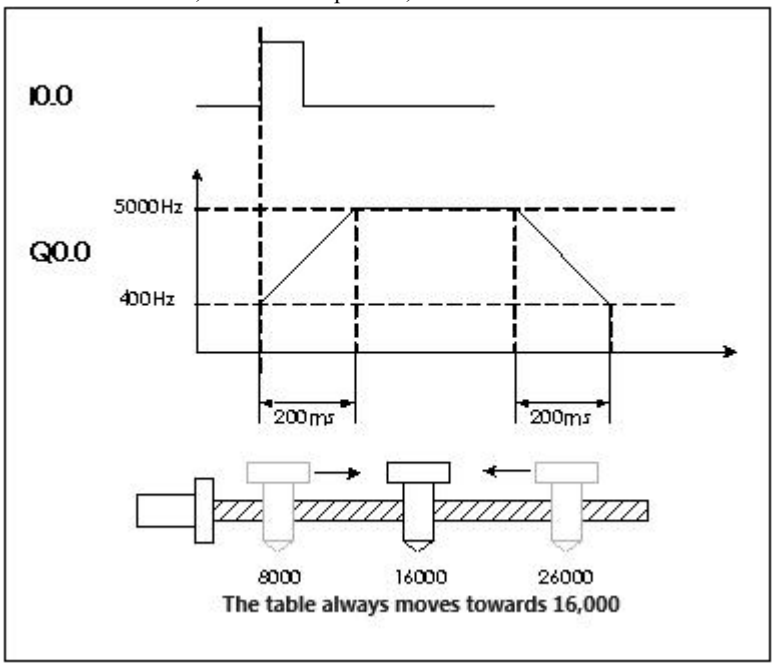
  The "relative motion (reverse)" signal connected to I0.1 is used to start the relative motion (reverse).

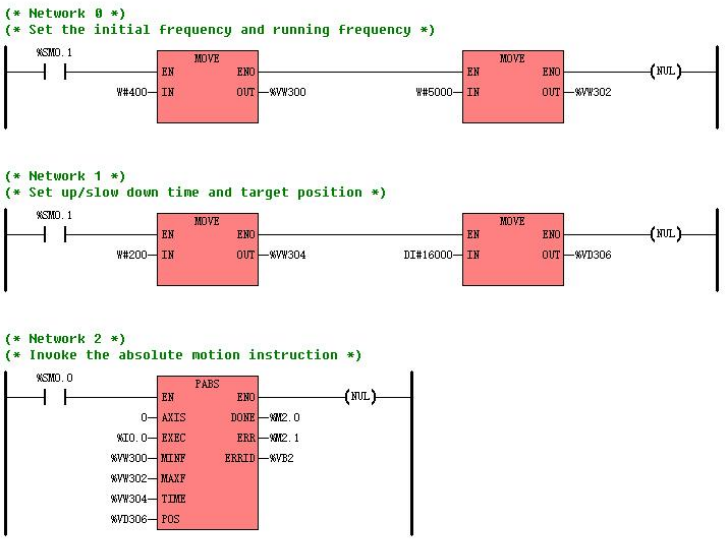
Description	If the rising edge of I0.1 is detected, a pulse is output at Q0.0, and the output of Q0.2 is 1, indicating inversion. 

<table>
<tr><td>LD</td><td> 
</td></tr>
<tr><td>IL</td><td>(* Network 0 *)
(*Set the initial frequency and running frequency*)
LD %SM0.1
MOVE W#400, %VW200
MOVE W#5000, %VW202
(* Network 1 *)
(*Set up/slow down time, amount of movement (negative numbers indicate reversal)*)
LD %SM0.1
MOVE W#200, %VW204
MOVE DI#-10000, %VD206
(* Network 2 *)
(*Invoke the relative motion instruction*)
LD %SM0.0
PREL 0, %I0.1, %VW200, %VW202, %VW204, %VD206, %M1.0, %M1.1, %VB1</td></tr>
</table>

- Absolute movement

  The "absolute motion" signal connected to I0.0 is used to start absolute motion.

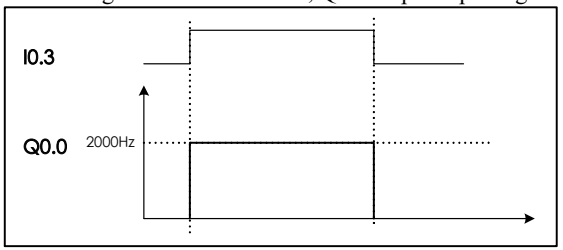
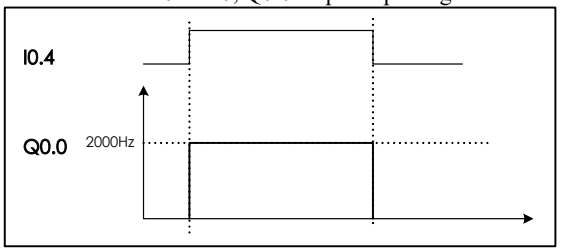
Description	If the rising edge of I0.0 is detected, a pulse will be output at Q0.0. If the output of Q0.2 is 0, it means forward rotation, and if the output is 1, it means reverse rotation. 

<table>
<tr><td>LD</td><td> 
</td></tr>
<tr><td>IL</td><td>(* Network 0 *)
(*Set the initial frequency and running frequency*)
LD %SM0.1
MOVE W#400, %VW300
MOVE W#5000, %VW302
(* Network 1 *)
(*Set up/slow down time and target position*)
LD %SM0.1
MOVE W#200, %VW304
MOVE DI#16000, %VD306
(* Network 2 *)
(*Invoke the absolute motion instruction*)
LD %SM0.0
PABS 0, %I0.0, %VW300, %VW302, %VW304, %VD306, %M2.0, %M2.1, %VB2</td></tr>
</table>

- Jog

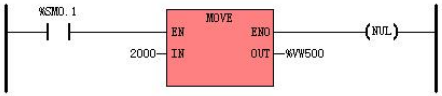
The "jog (forward)" signal connected to I0.3 is used for jog (motor forward). The "jog (reverse)" signal

connected to I0.4 is used for jog (motor reverse). If I0.3 and I0.4 are turned on at the same time, the last motor rotation will be used.

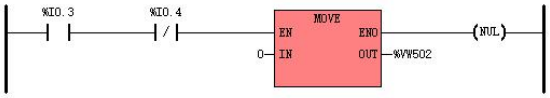
Description	If I0.3 is 1, Q0.0 continues to output pulse train. The output of Q0.2 is 0, indicating that the motor is rotating forward. If I0.3 is 0, Q0.0 stops outputting immediately.  If I0.4 is 1, Q0.0 continues to output pulse train. The output of Q0.2 is 1, indicating that the motor is reversed. If I0.4 is 0, Q0.0 stops outputting immediately. 

LD

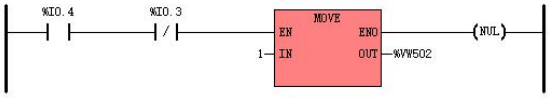
```
(* Network 0 *)
(* Set the operating frequency to 2000HZ *)
```




```
(* Network 1 *)
(* Set the motor direction to turn forward *)
```

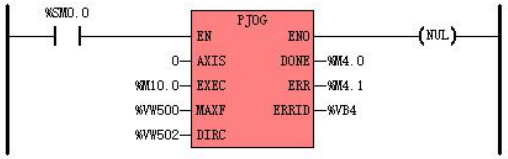



```
(* Network 2 *)
(* Set motor direction: reverse *)
```




```
(* Network 3 *)
(* Jog *)
```

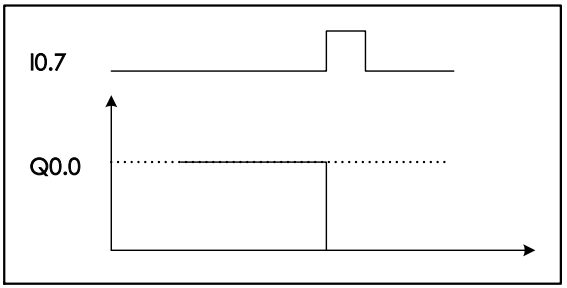
```
(* Network 4 *)
```

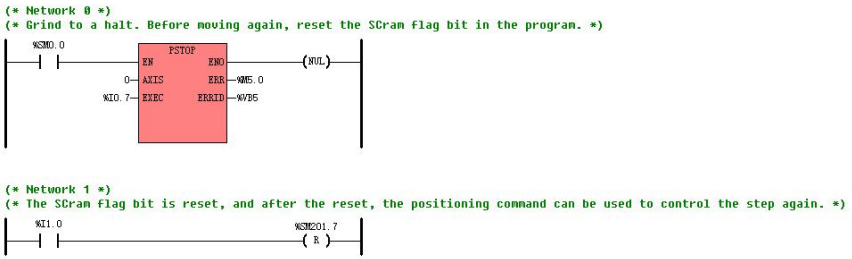


IL	(* Network 0 *)
	(*Set the operating frequency to 2000HZ*)
	LD %SM0.1
	MOVE 2000, %VW500
	(* Network 1 *)
	(*Set the motor direction to turn forward*)
	LD %I0.3
	ANDN %I0.4
	MOVE 0, %VW502
	(* Network 2 *)
	(*Set motor direction: reverse*)
	LD %I0.4
	ANDN %I0.3
	MOVE 1, %VW502
	(* Network 3 *)
	(*Jog*)
	LD %I0.3
	OR %I0.4
	ST %M10.0
	(* Network 4 *)
	LD %SM0.0
	PJOG 0, %M10.0, %VW500, %VW502, %M4.0, %M4.1, %VB4

- Emergency stop and emergency stop reset

There are two limit switches at both ends of the lead screw, these two switches are connected in parallel to I0.7 as the emergency stop signal. After calling the PSTOP instruction in the program, if you need to use the Position instruction to control the stepper motor again, you need to reset the emergency stop flag, and I1.0 is the emergency stop reset signal.

Description	If the rising edge of I0.7 is detected, Q0.0 stops outputting immediately. 

LD	
IL	(* Network 0 *) (*Grind to a halt. Before moving again, reset the SCram flag bit in the program.*) LD %SM0.0 PSTOP 0, %I0.7, %M5.0, %VB5 (* Network 1 *) (*The SCram flag bit is reset, and after the reset, the positioning command can be used to control the step again.*) LD %I1.0 R %SM201.7

9.4 Use of PLS Instructions

The PLS instruction can realize the PTO or PWM output function.

- PTO: Pulse Train Output, Pulse train output, fixed duty cycle of 50%, cannot be adjusted, but the number and period of pulses can be controlled. By controlling the number and period of pulses output by PTO, the effect of controlling motor speed and Position can be achieved.
- PWM: Pulse-Width Modulation, Pulse width modulation can adjust the pulse width (duty ratio 0-100%) and period, and the effect of controlling the output voltage can be achieved by adjusting the pulse width and period. It can be used to control the speed and torque of the motor, adjust the brightness of the light, etc., and can also adjust the duty cycle with the PID instruction to achieve the effects of temperature control and pressure control.

9.4.1 High-speed pulse output function (PTO/PWM)

Kinco-K series PLC supports up to 4 high-speed pulse outputs, and accordingly provides 4 PTO/PWM

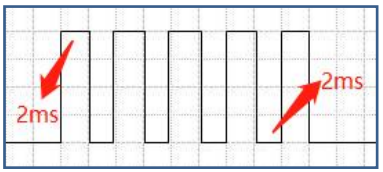
pulse generators for generating PTO/PWM outputs. Among them, the first pulse generator is assigned to Q0.0, called PWM0 or PTO0; the second is assigned to Q0.1, called PWM1 or PTO1; the third is assigned to Q0.4, called PWM2 or PTO2; The fourth is assigned at Q0.5 and is called PWM3 or PTO3.。

The memory addresses Q0.0, Q0.1 and Q0.4, Q0.5 are shared by the PTO/PWM generator and the DO image register. If the high-speed output instruction of a certain channel is called in the user program, the PTO/PWM generator will control the output channel and prohibit the output of ordinary DO.

Note: If Q0.0, Q0.1, Q0.4, Q0.5 are relay type, avoid using high-speed pulse output function!

(1) PTO

The PTO function can generate a pulse train square wave with a specified number of pulses (which can be understood as a PWM wave with a 50% duty cycle). The user can control the period of the output square wave and the number of output pulses. The unit of the pulse period is microseconds (μs) or milliseconds (ms), and the maximum period value is 65535. The range of the number of pulses is: 2 to 4,294,967,295. If the specified number of pulses is less than 2, the CPU will set the corresponding error flag and disable the output.



The above picture shows the waveform of a pulse train square wave with a period of 4ms and a pulse number of 5

The PTO function provides two modes of single-segment operation and multi-segment operation.

- **Single-segment operation**

In the single-segment operation mode, only one pulse train output will be performed after each execution of the PLS instruction, so the special register needs to be updated for the next pulse train. Once the start PTO segment is started, the special register must be changed according to the requirements of the second signal waveform, and the PLS instruction will be triggered again to execute the next pulse train after the first segment is executed.

- **Multi-stage operation**

In the multi-segment operation mode, the CPU automatically reads the set value of each PTO segment from the envelope table of the V area and executes the PTO of this segment according to the set value.

The settings of each segment in the envelope table occupy 8 bytes. It includes a period value (16-bit unsigned integer), reserved value (not used for now, 16-bit signed integer) and a pulse value (32-bit unsigned double integer). That is, in the same segment, the output frequency of all pulses is the same. Multi-segment operations are configured and started using the PLS instruction.

The starting position of the envelope table is stored in SMW168 (PTO0), SMW178 (PTO1), SMW218 (PTO2) and SMW248 (PTO3). The time base is set by SM67.3 (PTO0), SM77.3 (PTO1), SM97.3 (PTO2), and SM107.3 (PTO3). The time base can be selected in microseconds or milliseconds. All period values in the envelope table must use the same time base and cannot be changed while the envelope is executing.

The format of the envelope table is shown in the table below.

fset (1)	Length	Segment	Description
0	8bit		Number of Segment (1 to 64)
1	16bit	1st segment	Initial period (2 to 65535 time base)
3	16bit		reserve
5	32bit		Number of pulses (1 to 4,294,967,295)
9	16bit	2nd segment	Initial period (2 to 65535 time base)
11	16bit		reserve
13	32bit		Number of pulses (1 to 4,294,967,295)
…		…	…

All offsets are offset in bytes relative to the start of the envelope table.

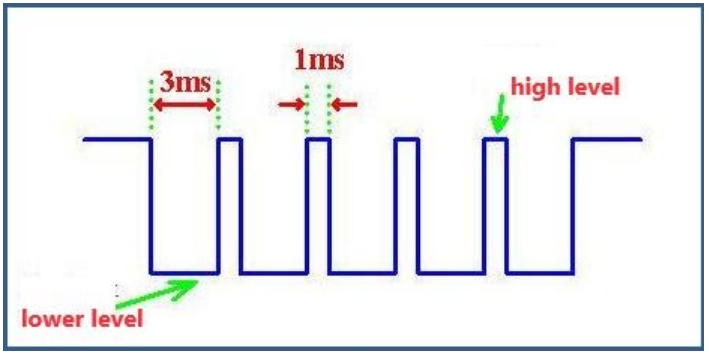
Note: The starting position of the envelope table must be an odd address in the V area, such as VB3001. And it should be noted that the maximum range should not exceed the maximum allowable area of memory, otherwise unpredictable errors will occur

(2) PWM

The PWM function provides continuous pulse output with adjustable duty cycle. The user can control the period and pulse width of the output.

The unit of period and pulse width can be selected from microseconds (μs) or milliseconds (ms), and the

maximum period value is 65535. When the pulse width is greater than or equal to the period, the duty cycle is automatically set to 100%, and the output is always on. When the pulse width is 0, the duty cycle is 0% and the output is disconnected.



The above picture shows a waveform with a period of 4ms, a high level (pulse width) of 1ms, a low level of 3ms, and a duty cycle of 25%.

There are two ways to change the characteristics of the PWM signal waveform::

1、Synchronous update: If it is not required to change the time base, synchronous update can be used. With synchronous updates, changes in signal waveform characteristics occur on periodic edges, providing smooth transitions.

2、Asynchronous update: Generally, for PWM operation, the pulse width changes while the period remains the same, so there is no requirement to change the time base. However, if the time base of the PTO/PWM generator needs to be changed, asynchronous updates are used. Asynchronous updates will cause the PTO/PWM function to be disabled momentarily, out of sync with the PWM signal waveform. This can cause the controlled device to vibrate. For this reason, a PWM synchronous update is recommended. Select a time base suitable for all cycle times.

9.4.2 PTO/PWM Control Register and Status Register

In the SM area, some control registers are provided for each PTO/PWM generator to store its

configuration data. See the table below.

Q0.0	Q0.1	Q0.4	Q0.5	Description	
SM67.0	SM77.0	SM97.0	SM107.0	PTO/PWM	Whether to update the period value: 0=no; 1=yes
SM67.1	SM77.1	SM97.1	SM107.1	PWM	Whether to update the pulse width value: 0=no; 1=yes
SM67.2	SM77.2	SM97.2	SM107.2	PTO	Whether to update the number of pulses: 0=no; 1=yes
SM67.3	SM77.3	SM97.3	SM107.3	PTO/PWM	Time base: 0=1μs; 1=1ms
SM67.4	SM77.4	SM97.4	SM107.4	PWM	Update method: 0 = asynchronous update; 1 = synchronous update
SM67.5	SM77.5	SM97.5	SM107.5	PTO	Operation mode: 0=single segment operation; 1=multiple segment operation
SM67.6	SM77.6	SM97.6	SM107.6	Function selection: 0=PTO; 1=PWM	
SM67.7	SM77.7	SM97.7	SM107.7	PTO/PWM	Enable or disable this function: 0=disable; 1=enable
Q0.0	**Q0.1**	**Q0.4**	**Q0.5**	**Description**	
SMW68	SMW78	SMW98	SMW108	PTO/PWM	Period value, range 2~65535
SMW70	SMW80	SMW100	SMW110	PWM	Pulse width value, range 0~65535
SMD72	SMD82	SMD102	SMD112	PTO	Number of pulses, range 1~4,294,967,295
SMW168	SMW178	SMW218	SMW248	The starting position of the envelope table (indicated by a byte offset relative to VB0), used only for PTO multi-segment operations.	

The default values of all control bytes, periods, and pulses are 0. The method for the user to modify the characteristics of the PTO/PWM waveform is: first set the corresponding control register, if it is a PTO multi-segment operation, the envelope table must also be set first, and then execute the PLS instruction.

In the SM area, a status byte is also provided for each PTO/PWM generator, and the user can know the current status information of the PTO/PWM generator by accessing the status byte. See the table below.

Q0.0	Q0.1	Q0.4	Q0.5	Description
SM66.0	SM76.0	SM96.0	SM106.0	Reserve
SM66.1	SM76.1	SM96.1	SM106.1	Reserve
SM66.2	SM76.2	SM96.2	SM106.2	Reserve
SM66.3	SM76.3	SM96.3	SM106.3	Whether the PWM is idle: 0=no; 1=yes
SM66.4	SM76.4	SM96.4	SM106.4	Whether the PTO period value and the number of pulses are set incorrectly: 0=No; 1=Yes Note: The period value and the number of pulses must be greater than 1.
SM66.5	SM76.5	SM96.5	SM106.5	Whether the PTO was terminated due to user instruction: 0=no; 1=yes
SM66.6	SM76.6	SM96.6	SM106.6	reserve

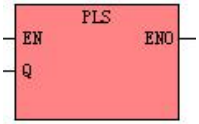
SM66.7	SM76.7	SM96.7	SM106.7	Is the PTO idle: 0=busy; 1=idle

The PTO idle bit and the PWM idle bit indicate whether the PTO output and PWM output have ended.

9.4.3 PLS instruction

The PLS instruction is located in the [Instruction Set]->[Counter] group.

Instruction and its operand description

	Name	Instruction format	CR value affected	Adopt to
LD	PLS			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ HP ☑ MK ☑ K6
IL	PLS	PLS *Q*	U	

Parameters	Input/Output	Datatype	Memory area allowed
Q	Input	INT	constant (0、1、2、3)

The function of the PLS instruction is to read the value of the corresponding control register in the SM area and configure the characteristics of the high-speed pulse output, and then start the high-speed pulse output until the specified pulse output function is completed. The pulse output channel is specified by the parameter Q, 0 means using Q0.0 output, 1 means using Q0.1 output, 2 means using Q0.4 output, 3 means using Q0.5 output.

Note: In the user program, only execute the PLS instruction once when needed. It is recommended to use the output result of the edge instruction to call the PLS instruction. If the EN terminal of PLS remains at 1, the PLS instruction will not be output normally.

(1) **LD**

If the EN value is 1, the PLS instruction is executed.

(2) **IL**

If the CR value is 1, the PLS instruction is executed. The execution of this instruction does not affect the CR value.

9.4.4 Using the PTO function

Take PTO0 as an example to introduce how to program and use the PTO function.

In general, using PTO involves two steps:

1. Set the relevant control registers and initialize the PTO;

2. Execute the PLS instruction.

It is recommended that the user try to write a separate initialization subroutine in the project, so that the entire user project can have a good structure. In addition, if possible, try to call this initialization subroutine with SM0.1 as the condition in the main program, so that the subroutine will only be called and executed once in the first scan after the CPU is powered on, which can reduce the scan of the CPU time.

9.4.4.1 PTO (single segment operation)

9.4.4.1.1 Execute PTO

1) The desired operation is achieved according to the set control byte SMB67.

   For example, SMB67 = B#16#85 shows:

   - Allow PTO/PWM function;
   - Choose to use PTO function, single segment operation;
   - The time base is selected as 1μs;
   - Allows updating the pulse count and period values.。

2) Assign the desired period value to SMW68.

3) Assign the desired number of pulses to SMD72。

4) (Optional) Use the ATCH instruction to connect an interrupt service routine for the "PTO0 complete"

interrupt event (event number 27) for fast response to the interrupt event.

5) Call the rising edge of m0.0 to start PTO0 by executing the PLS instruction according to the initialized PTO parameters.

9.4.4.1.2 Change PTO period

Follow the steps below to change the PTO0 period value:

1） The desired operation is achieved by setting the control byte SMB67:

For example, SMB67 = B#16#81 shows:

- Allow PTO/PWM function;
- Choose to use the PTO function, single-segment operation;
- The time base is selected as 1μs;
- Allows to update the period value.

2） Assign the desired period value to SMW68.

3） Execute the PLS instruction to configure and start PTO0, the PTO with the new period value will start immediately。

9.4.4.1.3 Change the number of PTO pulses

Follow the steps below to change the number of pulses output by PTO0:

1） The desired operation is achieved by setting the control byte SMB67:

For example, SMB67 = B#16#84 shows:

Allow PTO/PWM function;

Choose to use the PTO function, single-segment operation;

The time base is selected as 1μs;

Allows updating the number of pulses.。

2） Assign the desired number of pulses to SMD72.

3） Execute the PLS instruction to configure and start PTO0, and the newly specified number of pulses will be output immediately.

9.4.4.1.4 Example of use PTO

Different PTO output pulses and periods are selected by different conditions.

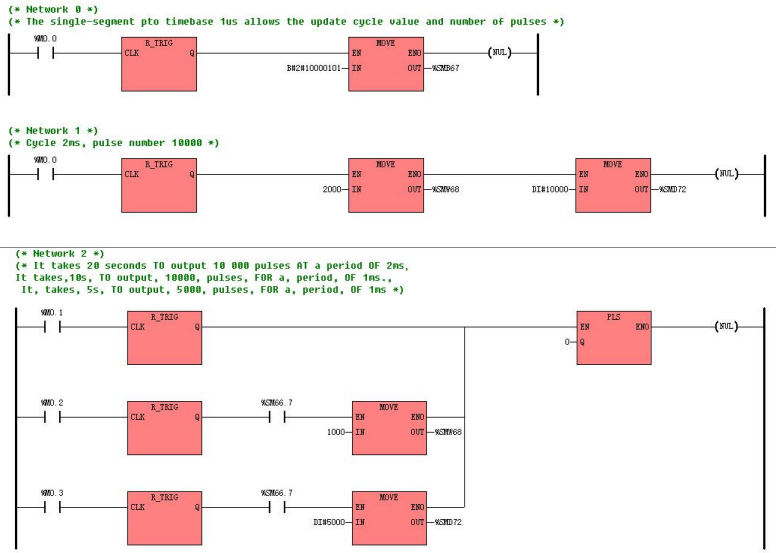
M0.0 Initializes the special register for the first time to set the period and pulse.

M0.1 Start the PTO output.

M0.2 Modify the period to restart the PTO output to take effect.

M0.3 Modify the number of output pulses to restart the PTO output to take effect.

LD



IL	(* Network 0 *)
	(*The single-segment pto timebase 1us allows the update cycle value and number of pulses*)
	LD %M0.0
	R_TRIG
	MOVE B#2#10000101, %SMB67
	(* Network 1 *)
	(*Cycle 2ms, pulse number 10000*)
	LD %M0.0
	R_TRIG
	MOVE 2000, %SMW68
	MOVE DI#10000, %SMD72
	(* Network 2 *)
	(*It takes 20 seconds to output 10,000 pulses at a period of 2ms
	It takes 10s to output 10000 pulses for a period of 1ms. It takes 5s to output 5000 pulses for a period of 1ms*)
	LD %M0.1
	R_TRIG
	OR(
	LD %M0.2
	R_TRIG
	AND %SM66.7
	MOVE 1000, %SMW68
	)
	OR(
	LD %M0.3
	R_TRIG
	AND %SM66.7
	MOVE DI#5000, %SMD72
	)
	PLS 0

9.4.4.2 PTO (Multi-segment operation)

9.4.4.2.1 Execute PTO

1） Set control byte SMB67 to achieve the desired operation。

For example, SMB67 = B#16#A0 shows:

- Allow PTO/PWM function;
- Option to use the PTO function
- Select multi-segment operation;

- The time base is selected as 1μs;

2） Assign the start position of the envelope table (odd number, indicating the byte offset of the start address of the envelope table relative to VB0) to SMW168.

Set the relevant value in the envelope table.

3） (Optional) Use the ATCH instruction to connect an interrupt service routine for the "PTO0 complete" interrupt event (event number 27) to achieve a fast response to the interrupt event.

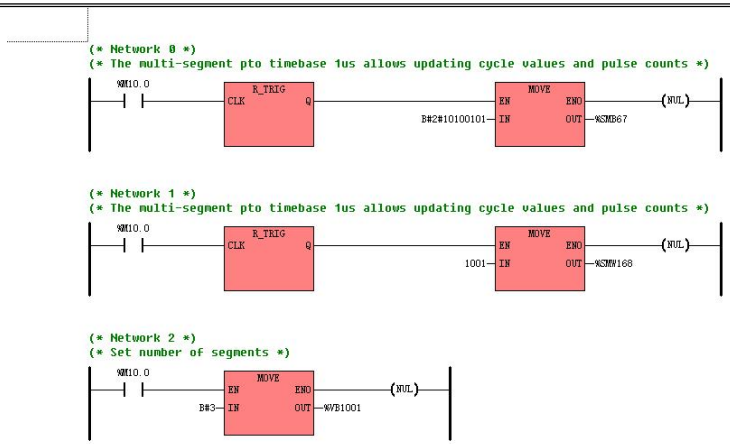
4） Execute the PLS instruction to configure and start PTO0.

9.4.4.2.2 Example of use

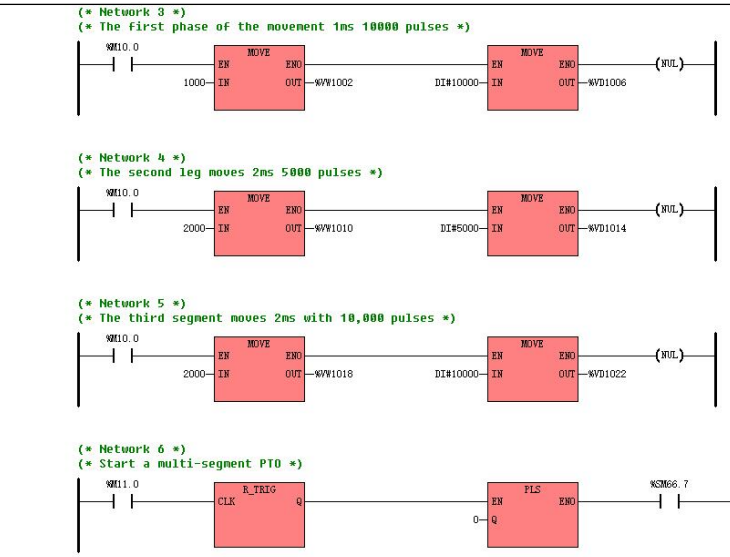
In the example, the parameters of the three-segment PTO are set in the envelope table to realize the three-segment motion of the stepper motor. The envelope table is as follows:

Segment	Description	
1st	period	1ms
	reserve	
	number of pulses	10000
2nd	period	2ms
	reserve	
	number of pulses	5000
3rd	period	2ms
	reserve	
	number of pulses	10000

LD


(* Network 0 *)
(* The multi-segment pto timebase 1us allows updating cycle values and pulse counts *)
%M10.0
R_TRIG
CLK
Q
MOVE
EN
ENO
B#2#10100101
IN
OUT
%SMB67
(NUL)
(* Network 1 *)
(* The multi-segment pto timebase 1us allows updating cycle values and pulse counts *)
%M10.0
R_TRIG
CLK
Q
MOVE
EN
ENO
1001
IN
OUT
%SMW168
(NUL)
(* Network 2 *)
(* Set number of segments *)
%M10.0
MOVE
EN
ENO
B#3
IN
OUT
%VB1001
(NUL)

LD


(* Network 3 *)
(* The first phase of the movement 1ms 10000 pulses *)
%M10.0
MOVE
EN
ENO
1000
IN
OUT
%VW1002
MOVE
EN
ENO
DI#10000
IN
OUT
%VD1006
(NUL)
(* Network 4 *)
(* The second leg moves 2ms 5000 pulses *)
%M10.0
MOVE
EN
ENO
2000
IN
OUT
%VW1010
MOVE
EN
ENO
DI#5000
IN
OUT
%VD1014
(NUL)
(* Network 5 *)
(* The third segment moves 2ms with 10,000 pulses *)
%M10.0
MOVE
EN
ENO
2000
IN
OUT
%VW1018
MOVE
EN
ENO
DI#10000
IN
OUT
%VD1022
(NUL)
(* Network 6 *)
(* Start a multi-segment PTO *)
%M11.0
R_TRIG
CLK
Q
PLS
EN
ENO
0
Q
%SM66.7
(NUL)

IL	(* Network 0 *)
	(*The multi-segment pto timebase 1us allows updating cycle values and pulse counts*)
	LD %M10.0
	R_TRIG
	MOVE B#2#10100101, %SMB67
	(* Network 1 *)
	(*The multi-segment pto timebase 1us allows updating cycle values and pulse counts*)
	LD %M10.0
	R_TRIG
	MOVE 1001, %SMW168
	(* Network 2 *)
	(*Set number of segments*)
	LD %M10.0
	MOVE B#3, %VB1001

IL	(* Network 3 *)
	(*The first phase of the movement 1ms 10000 pulses*)
	LD %M10.0
	MOVE 1000, %VW1002
	MOVE DI#10000, %VD1006
	(* Network 4 *)
	(*The second leg moves 2ms 5000 pulses*)
	LD %M10.0
	MOVE 2000, %VW1010
	MOVE DI#5000, %VD1014

IL	(* Network 5 *)
	(*The third segment moves 2ms with 10,000 pulses*)
	LD %M10.0
	MOVE 2000, %VW1018
	MOVE DI#10000, %VD1022
	(* Network 6 *)
	(*Start a multi-segment PTO*)
	LD %M11.0
	R_TRIG

	PLS 0 AND %SM66.7 ST %BR0.0

9.4.5 Using the PWM function

In general, using PWM consists of two steps:

1、setting the relevant control registers and initializing the PWM function.

2、Execute PLS instruction.

It is recommended that the user try to write a separate initialization subroutine in the project, so that the whole user project can have a good structure. In addition, if possible, try to call this initialization subroutine with SM0.1 in the main program, so that this subroutine will be called and executed only once in the first scan after the CPU is powered on, which can reduce the CPU scan time.

9.4.5.1 Execute PWM

1) Set the control byte SMB67 to achieve the desired operation。

For example, SMB67 = B#16#D3 shows:

·Allow PTO/PWM function;

· Choose to use PWM function;

· Choose to use the synchronous update method;

·The time base is selected as 1μs;

· Allows to update the number of pulses and period values。

2) Assign the desired period value to SMW68.

3) Assign the desired pulse width value to SMW70.

4) Execute the PLS instruction to configure and start PWM0.

9.4.5.2 Change PWM pulse width

The following describes how to change the pulse width of PWM0:

1） Set the control byte SMB67 to achieve the desired operation。

For example, SMB67 = B#16#D2 shows:

· Allow PTO/PWM function;

· Choose to use PWM function;

· Choose to use the synchronous update method;

· The time base is selected as 1μs;

· Allows to update the pulse width value.

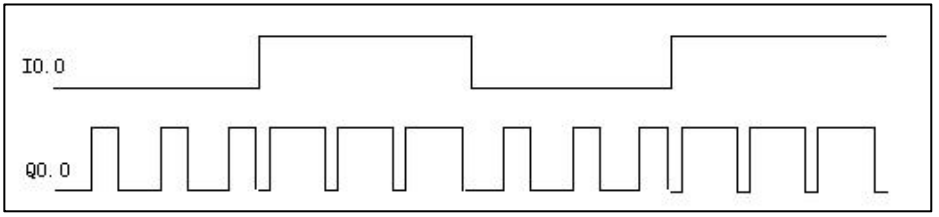
2) Assign the desired pulse width value to SMW70。

3) Execute the PLS instruction to configure and start PWM0.

9.4.5.3 Using example

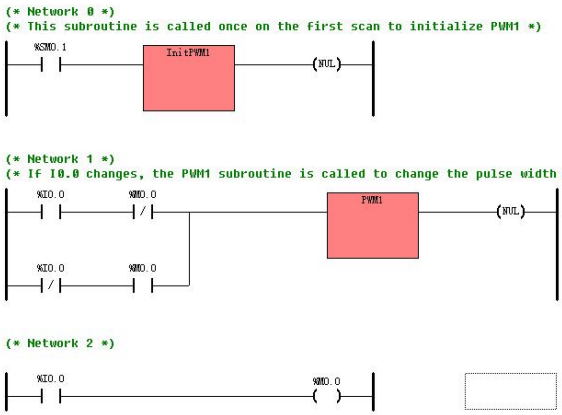
The example uses PWM1, Q0.1 output. The period is 10ms.

The duty cycle of the output pulse is changed by the I0.0 signal. If I0.0 is 0, the duty cycle is 40%; if I0.0 is 1, the duty cycle is 80%; the timing diagram is as follows:



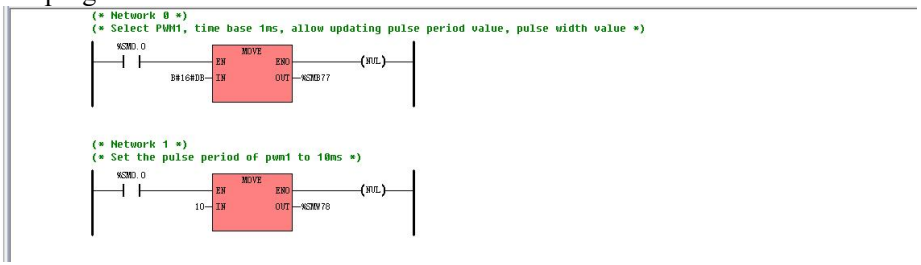
LD

MAIN:



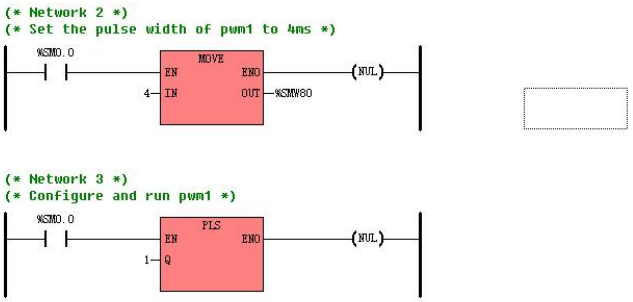
LD

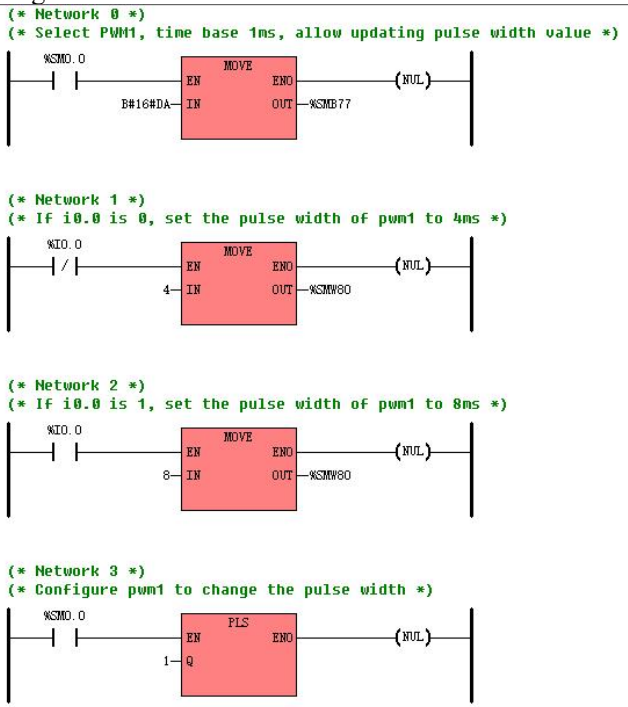
Subprogram InitPWM1:



LD

Subprogram InitPWM1:



<table>
<tr><td>LD</td><td>Subprogram PWM1:

</td></tr>
<tr><td>IL</td><td>MAIN:
(* Network 0 *)
(*This subroutine is called once on the first scan to initialize PWM1*)
LD %SM0.1
CAL InitPWM1
(* Network 1 *)
(*If I0.0 changes, the PWM1 subroutine is called to change the pulse width*)
LD %I0.0
ANDN %M0.0
OR(
LDN %I0.0
AND %M0.0
)</td></tr>
</table>

	CAL PWM1 (* Network 2 *) LD %I0.0 ST %M0.0

IL	Subroutine InitPWM1:
	(* Network 0 *)
	(*Select PWM1, time base 1ms, allow updating pulse period value, pulse width value*)
	LD %SM0.0
	MOVE B#16#DB, %SMB77
	(* Network 1 *)
	(*Set the pulse period of pwm1 to 10ms*)
	LD %SM0.0
	MOVE 10, %SMW78
	(* Network 2 *)
	(*Set the pulse width of pwm1 to 4ms*)
	LD %SM0.0
	MOVE 4, %SMW80
	(* Network 3 *)
	(*Configure and run pwm1*)
	LD %SM0.0
	PLS 1

IL	Subroutine PWM1: (* Network 0 *) (*Select PWM1, time base 1ms, allow updating pulse width value*) LD %SM0.0 MOVE B#16#DA, %SMB77 (* Network 1 *) (*If i0.0 is 0, set the pulse width of pwm1 to 4ms*) LDN %I0.0

```
MOVE    4, %SMW80
(* Network 2 *)
(*If i0.0 is 1, set the pulse width of pwm1 to 8ms*)
LD      %I0.0
MOVE    8, %SMW80
(* Network 3 *)
(*Configure pwm1 to change the pulse width*)
LD      %SM0.0
PLS     1
```

Chapter 10 Use of communication functions

10.1 Function Overview

KPLC supports many types of communication ports, including serial communication ports (RS232 and RS485), CAN communication ports, Ethernet communication ports and LORA wireless communication ports, and provides rich communication protocols and functions for each type of communication port.

The following section describes the functions and use of each communication port in detail.

10.2 Use of the serial communication port

The CPU module body provides the commonly used RS232 and RS485 serial communication ports.

The following table details the number of serial communication ports provided by each series PLC and the communication protocols they support.

Series		Number		communication protocols	
		RS232	RS485	RS232	RS485
MK		--	2	programming protocol, free communication, Modbus RTU Slave [3]	programming protocol [2]、free communication, Modbus RTU slaves [3]、Modbus RTU master, Kinco PLC Interconnection Protocol [4]
K2	K209EA	1	2		
	Others	--	2		
K6		1 [1]	3 [1]		
KS		1	1		
KW		1	1		
K6S		--	2		

Note: (1) The CPU module body of K6 provides 2 RS485 communication ports. Users can use the BD board of KB6-2COM to add 1 RS232 and 1 RS485 port. For the detailed description of the BD board, please refer to Appendix G Use of Extended BD Board.。

（2） Both PORT1 and PORT2 of K6 support programming protocol, and only PORT1 of other series supports programming protocol.

（3） All serial communication ports support Modbus RTU slave protocol, and default as Modbus RTU

slave, users can use it directly without programming.

（4） K6 and K6S PORT1 support Kinco PLC interconnection protocol master and slave station, MK series PORT1 support interconnection protocol slave station, for specific use, please refer to 10.6 Kinco Interconnection protocol communication.

（5）

Note: In the user program, the same serial communication port is not allowed to use multiple protocols at the same time!

All serial ports of the CPU module are numbered and named uniformly, among which the RS232 communication port is numbered 0 and named PORT0, the other RS485 interfaces are named PORT1, PORT2, etc., and so on. In addition, the first serial port (if there is PORT0, then PORT0 is the first; if there is no PORT0, then PORT1 is the first) supports the programming protocol. And in order to handle the abnormal situation, **the first serial port supports the following function: turn the "RUN/STOP" switch of CPU module to STOP position, and then re-power the CPU, the communication parameters of the first serial port will be automatically set to baud rate 9600, no parity, 8 data bits, 1 stop bit.**

10.2.1 Free communication

10.2.1.1 Overview

Free communication means that the communication process and the communication data format of the CPU communication port are completely controlled by the user program. The free communication method supports ASCII and binary data communication. Users can use the free communication method to write various custom communication protocols to communicate with other devices.

If the communication port uses the free communication function, the communication port is completely occupied by the free communication, and it can no longer use other communication protocols, including programming protocols, Modbus protocols, etc. At this time, if the user wants to use KincoBuilder software to download and upload programs, etc., he can put the CPU in the STOP

state, so that the CPU will stop executing the free communication instructions.

10.2.1.2 Instructions of free communication

10.2.1.2.1 Overview

The following instructions are located in the [Instruction Set]->[Communication Instruction] group.

Name	Function description	PLC type	communication port
XMT	send data	☑ K2 (Excluding K209M) ☑ KS (Excluding KS101M) ☑ KW (Excluding KW203) ☑ MK ☑ K6 ☑ K6S	RS232 RS485
RCV	Receive data		
COM_XMT	send data	☑ K209M ☑ KS101M ☑ KW203 ☑ MK ☑ K6 ☑ K6S	RS232 RS485 LORA
COM_RCV、COM_RCV2	Receive data		
COM_RESET	reset communication port		
COM_XMTRCV	Send and receive data	☑ K6S ☑ MK ☑ K6	RS232 RS485
COM_WPARAS	Modify serial port parameters	☑ K6S ☑ MK ☑ K6	RS232 RS485 LORA
COM_RPARAS	Read serial communication port parameters	☑ K6S ☑ MK ☑ K6	RS232 RS485 LORA

The instructions in the above table can be divided into two groups: XMT, RCV instructions are a group; COM_XMT, COM_RCV, COM_RCV2 and COM_RESET instructions are a group. The two sets of instructions are applicable to different types of CPUs. The similarities and differences between the two sets of instructions are described below.

Comparison of advantages and disadvantages of XMT, RCV instructions and COM_XMT, COM_RCV instructions:

1、The process of using the XMT and RCV instructions is relatively complicated. First, the free cov

mmunication register needs to be set to achieve the effect of the user-defined protocol, and then the XMT and RCV instructions need to be called for programming with the communication interruption.

2. The use of COM_XMT, COM_RCV, COM_RCV2 instructions does not require the above tedious processes. It is only necessary to directly call the instruction and set the input and output parameters. The programming is more convenient and easier to understand and use. This set of instructions was developed later, and only supported by some new series of PLCs, including K6, MK, KM (KS101M and KM209), KW203, etc.

3. When the communication logic requires strict verification, it is more reliable to use the XMT and RCV instructions to set the corresponding control registers, and users can choose according to their actual needs.

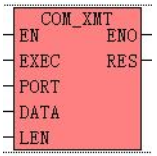
Because the serial port and the LORA wireless communication port are only different in the communication parameter settings, and the instructions are used in the same way, the following instructions are introduced according to the hardware configuration of the serial port. For the communication parameter settings of the LORA wireless communication port, please refer to the chapter 10.4 Use of the LPWAN Wireless Communication Port.

10.2.1.2.2 Use of COM_XMT and COM_RCV instructions

The following briefly describes the overall steps for the user to use the COM_XMT and COM_RCV instructions to program the serial port free communication:

1) Configure the communication parameters of the serial communication port used (including station number, baud rate, parity, etc.). The instruction defaults to the parameters defined in [Communication Settings] in [PLC Hardware Configuration]. If the operation is the RS485 port, the user can also use the COM_WPARAS instruction to set the communication parameters.
2) Call these two instructions directly in the program for programming. It should be noted that the free communication instructions use a half-duplex method to process communication data, so receiving and sending cannot be performed at the same time. Please pay attention when using these two instructions to program.

10.2.1.2.2.1 COM_XMT (Send data)

	Name	Instruction format	Adopt to	
LD	COM_XMT		☑ KM (KS101M and K209M) ☑ KW203 ☑ MK ☑ K6 ☑ K6S	

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
PORT	Input	INT	constant
DATA	Input	BYTE	V、M、L
LEN	Input	INT	V、M、L、constant
RES	Output	BYTE	V、M、L

The DATA and LEN parameters together form a memory block, which must be legal variable addresses.

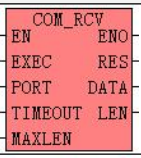
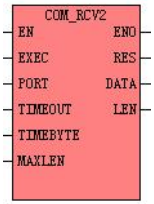
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
PORT	The communication port number used. 0 means PORT0, 1 means PORT1, 2 means PORT2, and so on. If the parameter value specifies a non-existing communication port, it is an illegal value, causing the instruction to report an error.
DATA	The first address of the storage area for the data to be sent.
LEN	The length of the data to be sent, unit: byte. A maximum of 248 bytes of data is allowed to be sent each time.
RES	Indicates the latest execution result. Its composition is as follows: Bit 7 ~ instruction status. This bit is set to 0 if the instruction is executing and is set to 1 immediately when the instruction is completed. Bit 4 ~ sending error. If an error occurs during sending, the sending will stop, and this bit will be set to 1. Bit 3~The communication port is busy. If the PORT is in the process of sending, the sending will not be started this time, and this bit will be set to 1. Bit 2～The data length is wrong. This bit is set to 1 if the LEN value is equal to 0 or exceeds the maximum length. Bit 1 ~ sending timeout. If the data has not been sent for more than 2 seconds, the sending is stopped, and this bit is set to 1. Bit 0~Illegal communication port. This bit is set to 1 if PORT is an illegal value.

	Other bit ~ Reserved

- LD

When the value of EN is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to execute once, and the data specified by the user is sent out through the PORT RES is set to 0 when the instruction is executed. When the instruction is completed (regardless of success or failure), the 7th bit of RES will be set to 1 immediately. In the program, the user can judge whether the instruction is completed according to the rising edge of 7 bits of RES, and then judge whether the transmission is successful or not according to the error value represented by other bits.

10.2.1.2.2.2 COM_RCV and COM_RCV2 (Receive data)

	Name	Instruction format	Adopt to
LD	COM_RCV		☑ KM (KS101M/K209M) ☑ KW203 ☑ MK ☑ K6 ☑ K6S
LD	COM_RCV2		

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
PORT	Input	INT	constant
TIMEOUT	Input	INT	V、M、L、constant
TIMEBYTE	Input	INT	V、M、L、constant
MAXLEN	Input	INT	V、M、L、constant
RES	Output	BYTE	V、M、L

DATA	Output	BYTE	V、M、L
LEN	Output	INT	V、M、L

Note: TIMEOUT and MAXLEN of the COM_RCV instruction must be both constant or the same memory type. The TIMEOUT, MAXLEN and TIMEBYTE of the COM_RCV2 instruction must be constant or the same memory type. Moreover, the DATA and LEN parameters together form a memory block, which must be legal variable addresses!

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
PORT	The communication port number used. 0 means PORT0, 1 means PORT1, 2 means PORT2, and so on. If the parameter value specifies a non-existing communication port, it is an illegal value, causing the instruction to report an error.
TIMEOUT	Indicates the maximum time allowed for this reception. Unit: ms。 If it exceeds the TIMEOUT value, exit the receiving state and set the corresponding flag bit of RES.
TIMEBYTE	Timeout between bytes. It will take effect from the receipt of the first byte. If the interval between any two adjacent bytes exceeds this timeout period, one reception will end. Otherwise, after receiving any byte, the timer will restart. Unit: ms.
MAXLEN	indicates the maximum number of bytes allowed for this reception, and the maximum value of MAXLEN is 248. Unit: byte. If the number of bytes received by the PLC exceeds MAXLEN, only the first MAXLEN bytes will be reserved and processed.
RES	The latest execution result. Its composition is as follows: Bit 7 ~ instruction status. This bit is set to 0 if the instruction is executing and is set to 1 immediately when the instruction is completed. Bit 4~ Receive error. If an error occurs during reception, the reception is stopped, and this bit is set to 1. Bit 3~The communication port is busy. If the PORT is in the process of receiving, the receiving will not be started this time, and this bit will be set to 1. Bit 2~MAXLEN parameter value is wrong. This bit is set to 1 if the MAXLEN value is greater than 248 or less than 0. Bit 1 ~ Receive timeout. If the receiving process time reaches TIMOUT, stop receiving, and this bit is set to 1. Bit 0~Illegal communication port. This bit is set to 1 if PORT is an illegal value. Other bits ~ reserved
DATA	The first address of the storage area for receiving data.
LEN	The length of received data, unit: byte.

The functions of the two instructions COM_RCV and COM_RCV2 are basically the same. The difference is: COM_RCV2 can set the timeout period between received characters through the

TIMEBYTE parameter, that is, the timeout period between any two bytes after receiving the first byte. If the next character is not received beyond this time, the instruction will complete this reception; while COM_RCV uses the default inter-character timeout of 3.5 characters.

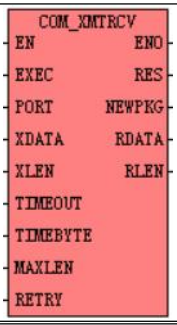
- LD

When the EN value is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to execute: The PORT enters the receiving state. If a frame message is received, it stops receiving and stores the received data in the receiving buffer (the first address of the buffer is DATA, and the length is LEN). The condition for the end of the receiving process is: if no new characters are received within the time of "3.5-character length" (The instruction will be automatically calculated according to the baud rate. The COM_RCV background uses 3.5 characters by default. COM_RCV2 can be set by TIMEBYTE input), the instruction considers that a complete frame of message has been received and completes the reception. If the receiving time reaches TIMEOUT, the instruction will also complete receiving immediately. After the receiving is completed, the instruction will judge the number of characters received. If it exceeds MAXLEN, the instruction will only keep the first MAXLEN characters, and discard the rest.

RES is set to 0 when the instruction is executed. When the instruction is completed (regardless of success or failure), the 7th bit of RES will be set to 1 immediately. In the program, the user can judge whether the instruction is completed according to the rising edge of the 7th bit of RES, and then judge whether the reception is successful according to the value of other bits: If the value of each bit is 0, the first bit is 1 or the second bit is 1, it can be considered that the reception is successful.

10.2.1.2.2.3 COM_XMTRCV (Send and receive data)

	Name	Instruction format	Adopt to

LD	COM_XMTRCV		☑ K6S ☑ MK ☑ K6

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
PORT	Input	INT	Constant
XDATA	Input	BYTE	V、M、L
XLEN	Input	INT	V、M、L、Constant
TIMEOUT	Input	INT	V、M、L、Constant
TIMEBYTE	Input	INT	V、M、L、Constant
MAXLEN	Input	INT	V、M、L、Constant
RETRY	Input	INT	V、M、L、Constant
RES	Output	BYTE	V、M、L
NEWPKG	Output	BOOL	V、M、L
RDATA	Output	BYTE	V、M、L
RLEN	Output	INT	V、M、L

Note: The XLEN, TIMEOUT, TIMEBYTE, MAXLEN, and RETRY parameters of this directive must be either constant or of the same memory type! In addition, the XDATA and XLEN parameters together form the send data memory block, and the RDATA and RLEN parameters together form the receive data memory block, which must be located in the valid variable address!

Parameters	Description
EXEC	If the rising edge of EXEC is detected, the instruction is triggered.
PORT	The number of the communication port used. 0 indicates PORT0, 1 indicates PORT1, 2 indicates PORT2, and so on. If the parameter value specifies a non-existent communication interface, it is an invalid value,

	resulting in an instruction error.
XDATA	The first address of the data area to be sent.
XLEN	Length of data to be sent.
TIMEOUT	The maximum time allowed for this execution (including sending and receiving). Unit: ms. If the TIMEOUT value is exceeded, exit the receiving state and set the RES flag bit.
TIMEBYTE	Timeout between received bytes. It takes effect from the receipt of the first byte. If the interval between any two adjacent bytes exceeds this timeout period, a receive ends. Unit: ms.
MAXLEN	Specifies the maximum number of bytes that can be received at this time. MAXLEN has a maximum value of 248. Unit: byte. If the PLC receives more bytes than MAXLEN, only the first MAXLEN bytes are retained and processed.
RETRY	The number of times the communication was retried after the communication failed. If the sending or receiving fails, the communication is restarted for a maximum of RETRY times.
RES	The latest execution result. Its composition is as follows: Bit 7 ~ instruction status. This bit is set to 0 if the instruction is executing, and to 1 immediately when the instruction is completed (either correctly or incorrectly). Bit 4 ~ Received error. If an error occurs during the receiving process, the receiving is stopped and the bit is set to 1. The third bit ~ The communication port is busy. If the PORT is receiving packets, the packets are not received at this time and the bit is set to 1. The second ~ MAXLEN parameter is incorrect. If MAXLEN is greater than 248 or less than 0, the bit is set to 1. Bit 1 ~ Receive timeout. If the receiving process time reaches TIMOUT, the receiving is stopped and the bit is set to 1. Bit 0 ~ Illegal communication interface. If PORT is an invalid value, the bit is set to 1. Other bits ~ reserved
NEWPKG	Indicates whether data was received during the execution. The 7th bit of RES indicates whether the execution is complete, and NEWPKG indicates that the execution is complete and the data is received.
RDATA	The first address of the receiving data storage area.
RLEN	Length of the received data, in bytes.

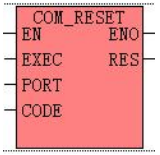
- LD

When the EN value is 1, if the rising edge of the input terminal of EXEC is detected, the command is triggered to execute: PLC first sends data through the PORT port, sends out the data in the sending area composed of XDATA and XLEN, and then immediately enters the receiving state and waits for receiving data. If the end of the receive condition is met, the receive is stopped and the received data is stored in the receive buffer (the first address of the buffer is RDATA and the length is RLEN). The receiving end conditions include: If no new characters are received within the specified TIMEBYTE after receiving a character, the

command considers that a complete frame has been received and the receiving is complete. If the instruction execution time reaches TIMEOUT, the instruction will also finish receiving immediately. After receiving, the instruction will determine the number of received characters, if more than MAXLEN, then the instruction only keeps the first MAXLEN characters, all the rest of the discard.

When the instruction is executed, RES is set to 0. When the instruction completes (whether successful or unsuccessful), the seventh bit of RES is immediately set to 1. In the program, the user can determine whether the instruction is complete based on the RES 7 bit or the rising edge of NEWPKG, and then determine whether there is a receiving error based on the value of other bits.。

10.2.1.2.2.4 COM_RESET (Reset communication port)

	Name	Instruction format	Adopt to
LD	COM_RESET		☑ KM (KS101M/K209M) ☑ KW203 ☑ MK ☑ K6 ☑ K6S

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
PORT	Input	INT	constant
CODE	Input	BYTE	V、M、L、constant
RES	Output	BYTE	V、M、L

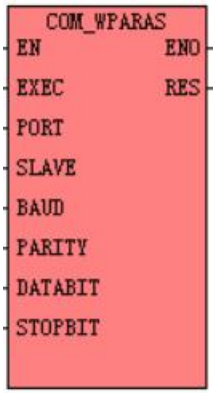
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
PORT	The communication port number to be reset. 0 means PORT0, 1 means PORT1, 2 means PORT2, and so on. If the parameter value specifies a non-existing communication port, it is an illegal value, causing the instruction to report an error.
CODE	Reset option, the meaning of its value is as follows: 1 ~ Only reset all software variables such as cache and status used by this communication port.

	2 ~ Reset all software variables used by this communication port and reset and initialize the chip hardware at the same time.
RES	Other values are illegal and will cause the instruction to report an error. The latest execution results. Its composition is as follows: Bit 7 ~ instruction status. This bit is set to 0 if the instruction is executing and is set to 1 immediately when the instruction is completed. Bit 1 ~ Illegal reset option. If CODE is an illegal value, this bit is set to 1. Bit 0~Illegal communication port. This bit is set to 1 if PORT is an illegal value.

- LD

When the value of EN is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to execute once, and the PORT communication port is reset according to the CODE option. RES is set to 0 when the instruction is executed. When the instruction is completed (regardless of success or failure), the 7th bit of RES will be set to 1 immediately. The user can judge whether the instruction is completed according to the 7th bit of RES in the program, and then judge whether it is successful or not according to the error value represented by other bits.

10.2.1.2.2.5 COM_WPARAS (Modify serial port parameters)

	Name	Instruction format	Adopt to
LD	COM_WPARAS		☑ K6S ☑ MK ☑ K6

Parameters	Input/Output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM

PORT	Input	INT	Constant
SLAVE	Input	BYTE	V、M、L、Constant
BAUD	Input	DINT	V、M、L、Constant
PARITY	Input	BYTE	V、M、L、Constant
DATABIT	Input	BYTE	V、M、L、Constant
STOPBIT	Input	BYTE	V、M、L、Constant
RES	Input	BYTE	V、M、L

Parameters	Description
EXEC	If a rising edge of EXEC is detected, the instruction is triggered.
PORT	Number of the communication port to be operated. 1 indicates PORT1, 2 indicates PORT2, and so on. This directive does not support modifying PORT0 parameters. If a non-existent communication interface is specified, it is an invalid value, resulting in an instruction error.
SLAVE	Station ID of the communication port in the communication network. The valid value ranges from 1 to 63.
BAUD	The valid baud rate is 2400, 4800, 9600, 19200, 38400, 57600, or 115200.
PARITY	Parity check, valid values are: 0 -- no check; 1 -- odd check; 2 -- even check.
DATABIT	Data bit, valid value bit 7 or 8.
STOPBIT	Stop bit. The valid value is 1 or 2.
RES	The latest execution result. Their meanings (0 being the lowest and 7 being the highest) are as follows: Bit 7 - instruction status. This bit is set to 0 if the instruction is executing and 1 immediately when the instruction is completed. 6th not -- Save new parameter result, 1 indicates failure to write new parameter to permanent memory. Bit 5 - whether the stop bit is correct, 1 indicates the wrong stop bit value. Bit 4 - whether the data bit is correct, 1 indicates the wrong data bit value. Bit 3 - whether the parity check is correct, 1 indicates an incorrect parity value. The second bit - whether the baud rate is correct, 1 indicates the incorrect baud rate value. First digit - Is the station number correct? 1 indicates the wrong station number value. Bit 0 -- PORT PORT number is correct. 1 indicates an incorrect port number.

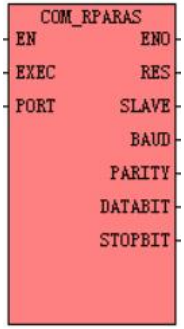
This instruction is used to modify the communication parameters of the serial communication interface. After the parameter modification is successful, the PLC will store this set of parameters into the permanent memory, which will be permanently valid. Every time the PLC is powered on, the stored parameters will be read and take effect preferentially.

Note: This instruction will write new parameters to the permanent memory when it is executed, and the memory life is 1 million times, after which the memory may have problems.

- LD

When the EN value is 1, if the rising edge of the EXEC input terminal is detected, the command is triggered to execute once, and the communication parameters of the PORT communication port are modified and stored according to the valid input parameter value. When the instruction is executed, RES is set to 0. When the instruction completes (whether successful or unsuccessful), the seventh bit of RES is immediately set to 1. In the program, the user can determine whether the instruction is complete according to the 7th bit of RES, and then determine whether it is successful according to the error value represented by the other bits.

10.2.1.2.2.6 COM_RPARAS (Read serial communication port parameters)

	Name	Instruction format	Adopt to
LD	COM_RPARAS		☑ K6S ☑ MK ☑ K6

Parameters	Input/Output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
PORT	Input	INT	Constant
RES	Output	BYTE	V、M、L
SLAVE	Output	BYTE	V、M、L、Constant
BAUD	Output	DINT	V、M、L、Constant
PARITY	Output	BYTE	V、M、L、Constant

DATABIT	Output	BYTE	V、M、L、Constant
STOPBIT	Output	BYTE	V、M、L、Constant

Parameters	Description
EXEC	If a rising edge of EXEC is detected, the instruction is triggered.
PORT	Number of the communication port to be operated. 1 indicates PORT1, 2 indicates PORT2, and so on. This directive does not support parameters that operate on PORT0. If a non-existent communication interface is specified, it is an invalid value, resulting in an instruction error.
SLAVE	Station ID of the communication port in the communication network. The valid value ranges from 1 to 63.
BAUD	The valid baud rate is 2400, 4800, 9600, 19200, 38400, 57600, or 115200.
PARITY	Parity check, valid values are: 0 -- no check; 1 -- odd check; 2 -- even check.
DATABIT	Data bit, valid value bit 7 or 8.
STOPBIT	Stop bit. The valid value is 1 or 2.
RES	The latest execution result. Their meanings (0 being the lowest and 7 being the highest) are as follows: Bit 7 - instruction status. This bit is set to 0 if the instruction is executing and 1 immediately when the instruction is completed. Positions 1-6 - Spare Bit 0 -- PORT PORT number is correct. 1 indicates an incorrect port number.

This instruction is used to read the communication parameters currently in use at the serial communication interface.

- LD

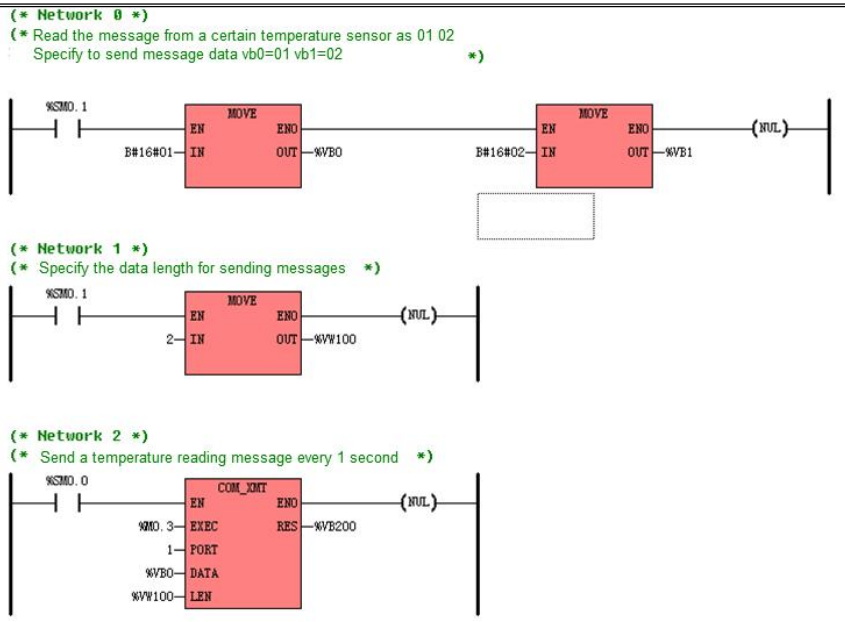
When the EN value is 1, if the rising edge of the EXEC input is detected, the command is triggered to execute once, reading the communication parameter values used by the PORT communication interface. When the instruction is executed, RES is set to 0. When the instruction completes (whether successful or unsuccessful), the seventh bit of RES is immediately set to 1. In the program, the user can determine whether the instruction is complete according to the 7th bit of RES, and then determine whether it is successful according to the error value represented by the other bits.

10.2.1.2.2.7 Instruction usage example

The following will illustrate the use of the free communication instructions COM_XMT and COM_RCV.

In the example, the PORT1 port of the PLC communicates with the temperature and humidity sensor to read the temperature value, and the PLC periodically sends the temperature reading message 01 02. After the sending is completed, enter the receiving state, put the received message into the designated register, and the user can find and analyze the register where the temperature value is stored according to the message protocol customized by the temperature and humidity sensor.

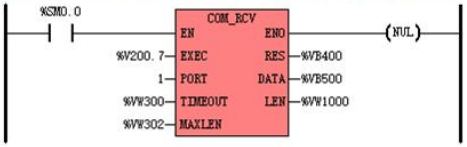
LD



LD

(* Network 3 *)
(* Specify a receive timeout of 500ms, with a maximum length of 200 bytes for receiving messages *)

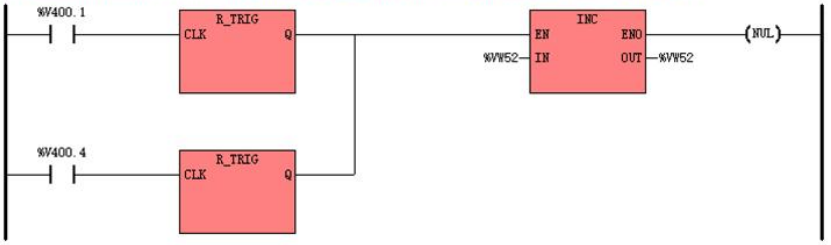
(* Network 4 *)
(* Put the received message data and message length into the specified register *)

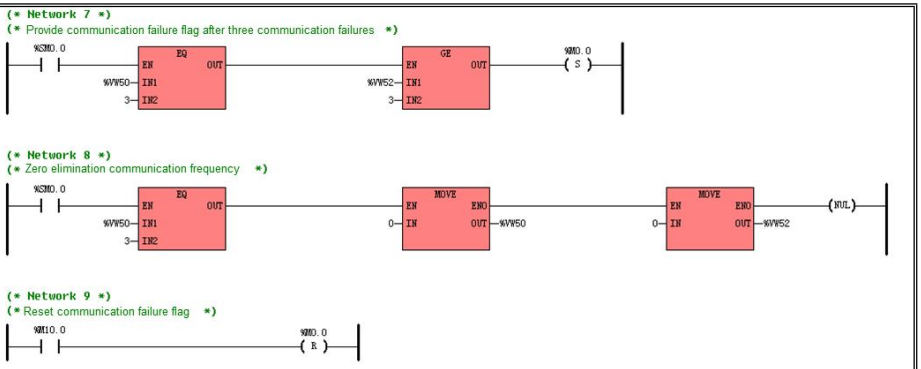


(* Network 5 *)
(* Record the number of times sent *)

(* Network 5 *)
(* Record the number of times sent *)

(* Network 6 *)
(* Record the number of communication failures caused by receiving timeouts and receiving errors *)



<table>
<tr><td></td><td> 
</td></tr>
<tr><td>IL</td><td>(* Network 0 *)
(*Read the message from a certain temperature sensor as 01 02, specify the message data to be sent as vb0=01 vb1=02*)
LD %SM0.1
MOVE B#16#01, %VB0
MOVE B#16#02, %VB1
(* Network 1 *)
(*Specify the data length for sending messages *)
LD %SM0.1
MOVE 2, %VW100</td></tr>
</table>

IL	(* Network 2 *)
	(*Send a temperature reading message every 1 second *)
	LD %SM0.0
	COM_XMT %M0.3, 1, %VB0, %VW100, %VB200
	(* Network 3 *)
	(*Specify a receive timeout of 500ms, with a maximum length of 200 bytes for receiving messages*)
	LD %SM0.1
	MOVE 500, %VW300
	MOVE 200, %VW302
	(* Network 4 *)
	(*Put the received message data and message length into the specified register*)
	LD %SM0.0
	COM_RCV %V200.7, 1, %VW300, %VW302, %VB400, %VB500, %VW1000
	(* Network 5 *)
	(*Record the number of times sent*)
	LD %V200.7
	R_TRIG
	INC %VW50
	(* Network 6 *)
	(*Record the number of communication failures caused by receiving timeouts and receiving errors*)
	LD %V400.1
	R_TRIG
	OR(
	LD %V400.4
	R_TRIG
	)
	INC %VW52

IL	(* Network 7 *)
	(*Provide communication failure flag after three communication failures *)
	LD %SM0.0
	EQ %VW50, 3
	GE %VW52, 3
	S %M0.0
	(* Network 8 *)
	(*Zero communication frequency *)
	LD %SM0.0
	EQ %VW50, 3
	MOVE 0, %VW50
	MOVE 0, %VW52
	(* Network 9 *)
	(*Reset communication failure flag*)
	LD %M10.0
	R %M0.0

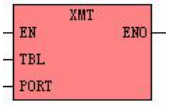
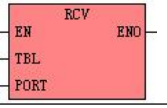
10.2.1.2.3 Use of XMT and RCV instructions

The following briefly describes the overall steps for users to use XMT and RCV instructions to program serial port free communication:

1) Set the serial communication parameters (including station number, baud rate, parity, etc.) of the communication port used in [PLC Hardware Configuration]. For details, please refer to the description in 4.3.4.1 CPU parameter configuration.
2) Set the free communication control register of the corresponding serial port in the program (define the start receiving character, end receiving character, timeout time, etc.), that is, customize the communication protocol between PLC and other devices. For details, see the description of the register definition in the status register and control register in 10.2.1.2.3.3. Note that different serial ports (PROT0, PROT1, PROT2) correspond to different control register addresses.
3) Call the XMT and RCV instructions and program with the status register of the serial port free communication and communication interruption. For the detailed use of the instructions, please refer to the description and sample program below.

10.2.1.2.3.1 XMT、RCV instruction (Send and receive data)

- Instruction and its operand description

	Name	Instruction format	CR value affected	
LD	XMT			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
	RCV			
IL	XMT	XMT TBL, PORT	U	
	RCV	RCV TBL, PORT		

Parameters	Input/Output	Data type	Memory area allowed
TBL	Input	BYTE	I、Q、M、V、L、SM

PORT	Input	INT	constant (0-2)

Note that the TBL parameter is a variable-length memory block parameter, and the entire block memory is not allowed to exceed the valid memory range, otherwise the result is unpredictable.

The XMT instruction is used to send the data stored in the data buffer. The parameter PORT defines the communication port used (0 means PORT0, 1 means PORT1, and so on). The parameter TBL defines the starting address of the data buffer. The first byte of the buffer defines the number of bytes to be sent this time (1--255), and the data bytes to be sent are stored in sequence in the back. If the number of bytes to send is set to 0, the XMT instruction does nothing.

The RCV instruction is used to receive data and store the received data in the data buffer. The parameter PORT defines the communication port used (0 means PORT0, 1 means PORT1, and so on). The parameter TBL defines the starting address of the data buffer. The first byte of the buffer stores the number of bytes received this time, and the following bytes store the received valid data bytes in turn.。

- LD

  If the EN value is 1, execute the XMT and RCV instructions, otherwise not execute.

- IL

  If the CR value is 1, then execute XMT, RCV instructions, otherwise not execute. The execution of the instruction does not affect the CR value.

10.2.1.2.3.2 communication interruption

KPLC provides a variety of serial port free communication interrupt solutions, which are only used in conjunction with XMT and RCV instructions. If users need to know more about interrupts, please refer to 6.12.1 How Kinco-K Series Handles Interrupt Events.

Users can use the control bit SM87.1, SM187.1 or SM287.1 to prohibit or allow the CPU to generate communication interrupts. If the interrupt control bit is set to 1, it is allowed to generate a communication interrupt: the CPU will generate a send completion interrupt when the last character in the buffer is sent. After the CPU exits receiving (whether it exits normally or abnormally), it will generate a reception completion interrupt.

10.2.1.2.3.3 status register, control register

KPLC provides multiple status registers and control registers for free communication in the SM area. When writing a communication program, the user must set these control registers. In addition, the CPU will automatically detect the communication status during the communication process, and write the detection results into the relevant status registers, and the user can read these status information and perform corresponding processing in the program.

(1) Receive status byte

Bit (read only)			Value	Meaning
PORT 0	PORT 1	PORT 2		
SM86.0	SM186.0	SM286.0	1	reserve.
SM86.1	SM186.1	SM286.1	1	Terminating receive: Reached the maximum number of bytes received.
SM86.2	SM186.2	SM286.2	1	Terminate receiving: Receive a character timeout.
SM86.3	SM186.3	SM286.3	1	Terminate receiving: the system receiving timeout.
SM86.4	SM186.4	SM286.4	-	reserve.
SM86.5	SM186.5	SM286.5	1	End Receive: A user-defined end character was received.
SM86.6	SM186.6	SM286.6	1	Termination of reception: parameter error, reception without start condition or no reception end condition, etc.
SM86.7	SM186.7	SM286.7	1	Terminate reception: The user has used the prohibit reception instruction.

(2) Receive control byte

Bit			Value	Description
PORT 0	PORT 1	PORT 2		
SM87.0	SM187.0	SM287.0	-	reserve.
SM87.1	SM187.1	SM287.1	0	Disable generation of corresponding interrupt when transmission or reception is complete.
			1	Allows generation of corresponding interrupts when a send or receive is complete.
SM87.2	SM187.2	SM287.2	0	Ignores the user-defined receive character timeout value in SMW92/SMW192/SMW292.
			1	Use the user-defined receive character timeout value in SMW92/SMW192/SMW292.
SM87.3	SM187.3	SM287.3	-	reserve.
SM87.4	SM187.4	SM287.4	0	Ignore the user-defined reception preparation time in SMW90/SMW190/SMW290.

			1	Use the user-defined reception preparation time in SMW90/SMW190/SMW290.
SM87.5	SM187.5	SM287.5	0	Ignore the user-defined receive end character in SMB89/SMW189/SMW289.
			1	Use the user-defined receive end character in SMB89/SMW189/SMW289.
SM87.6	SM187.6	SM287.6	0	Ignore the user-defined receive start character in SMB88/SMW188/SMW288.
			1	Use the user-defined receive start character in SMB88/SMW188/SMW288.
SM87.7	SM187.7	SM287.7	0	Receiving data is prohibited. This inhibit condition takes precedence over all other receive control words.
			1	Allow data to be received.

(3) Other control registers

Control bit (byte)			Description
PORT0	PORT1	PORT2	
SMB88	SMB188	SMB288	It is used to store user-defined receiving start character. After executing the RCV instruction, the CPU starts to enter the effective receiving state after receiving the start character, and the data received before will be discarded. The CPU regards the start character as the first valid data received. If you want this setting value to take effect, you need to set SM87.6/SM187.6/SM287.6 to 1.
SMB89	SMB189	SMB289	It is used to store user-defined receiving end characters. The CPU will take this character as the last byte received. After receiving this character, the CPU will immediately terminate the receiving state regardless of other end conditions. To make this setting effective, you need to set SM87.5/SM187.5/SM287.5 to 1.
SMW90	SMW190	SMW290	It is used to store the receiving preparation time defined by the user (the range is 0~60000ms). After executing the RCV instruction, after the time value, the CPU will automatically enter the valid receiving state regardless of whether the start character is received, and the data received thereafter will be considered as valid data. If you want this setting value to take effect, you need to set SM87.4/SM187.4/SM287.4 to 1.
SMW92	SMW192	SMW292	It is used to store the timeout value of receiving characters defined by the user (the range is 1~60000ms). After executing the RCV instruction and entering the effective receiving state, if no character is received within the timeout period, the CPU will end the receiving state, regardless of other end conditions.

			To make this setting effective, you need to set SM87.2/SM187.2/SM287.2 to 1.
SMB94	SMB194	SMB294	It is used to store the user-defined number of received characters each time (1~255). As long as the CPU has received this number of valid characters, regardless of other end conditions, it will immediately terminate the receiving state. This setting value is always valid. If the value is set to 0, the RCV instruction will exit directly.

In the serial port free communication, there is also a default system receive timeout, the time is 60 seconds, the function of this timeout value is as follows: after executing the RCV instruction, if no data is received on the serial port within this timeout period, the CPU The reception will be terminated immediately and the RCV instruction will be exited; in addition, after the CPU enters the effective reception state (that is, after receiving the start character defined in SMB88 or after the reception preparation time defined in SMW90), the CPU will give priority to the user defined in SMW92. If the timeout value for receiving characters is not defined by the user, the system will use the timeout value for receiving characters to decide whether to terminate the reception.

10.2.1.3.3.4Instruction usage example

The following will illustrate the use of serial port free communication instructions XMT and RCV.

In the example, the CPU will receive a string of data, with a carriage return character as the receiving end character. If the receiving is completed normally, the received data will be sent back and the receiving will be started again. If the receiving state is exited abnormally (such as communication error, receiving timeout, etc.), the received data will be ignored and the receiving will be started again.

The example includes the main program MAIN, the subroutine InitComn, the receiving completion interrupt subroutine EndReceive, and the sending completion interrupt subroutine EndSend.

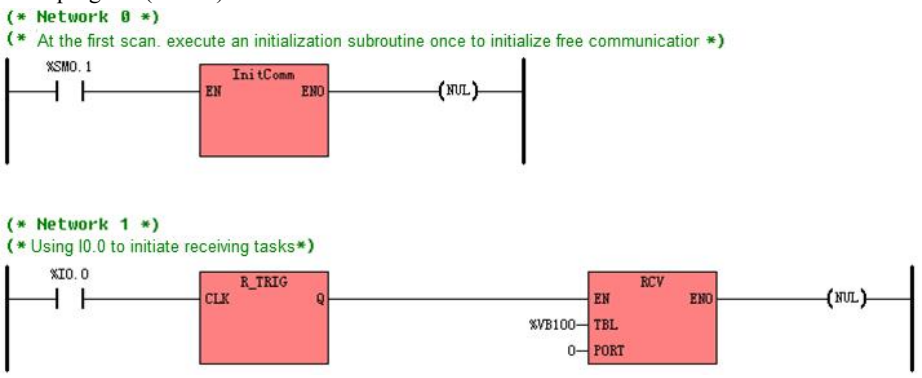
Main program MAIN: Trigger the receiving task and call the subroutine InitComn.

Subroutine InitComn: Set the control register (receive start, end condition, timeout time), that is, the custom protocol. Initialize free communication, and at the same time set the communication interrupt number and the corresponding interrupt. The subroutines are bound to generate the corresponding interrupts, and then

execute the corresponding sending and interrupting subroutines.

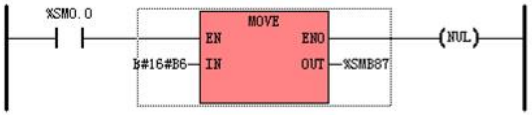
Receive complete interrupt subroutine EndReceive: receive complete interrupt generated, you can perform sending tasks in this program.

Send completion interrupt subroutine EndSend: send complete interrupt, restart the receiving task.

LD	Main program (MAIN): 

LD	InitComm (SBR00), initialization subroutine:

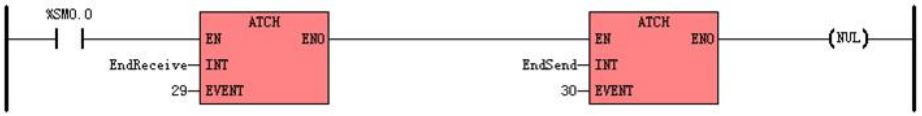
```
(* Network 0 *)
(* Set the starting and ending conditions for effective reception status. *)
```




```
(* Network 1 *)
(* Set the receiving preparation time to 1ms and the receiving end character to a carriage return *)
```

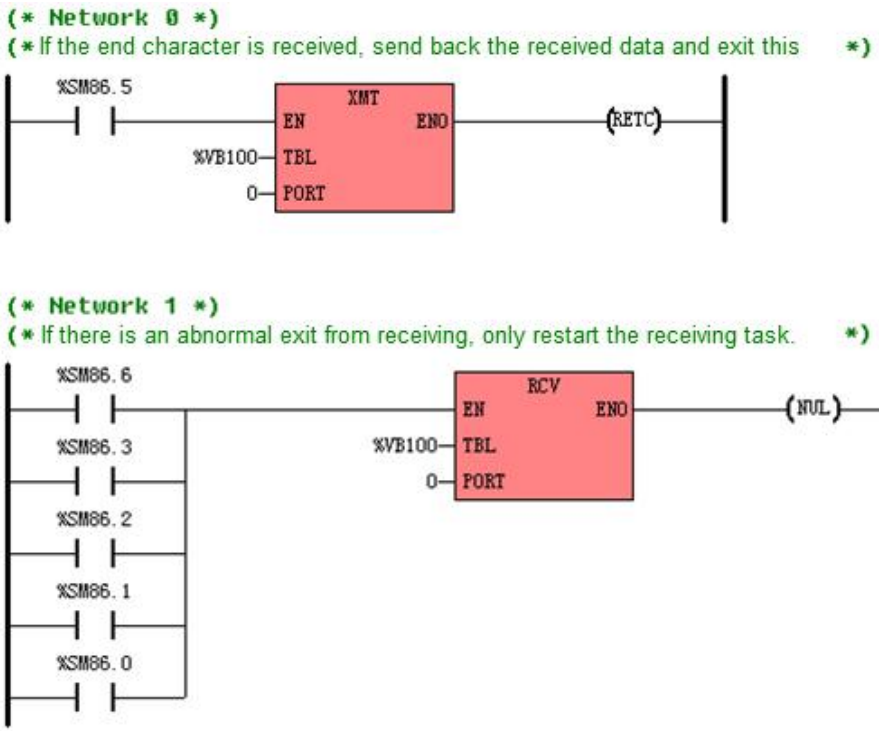
```
(* Network 2 *)
(* Set the timeout for receiving characters to 500ms, and set the maximum number of characters received *)
```

```
(* Network 3 *)
(* Bind EndReceive program to receive completion interrupt and EndSend program to send completion interrupt *)
```

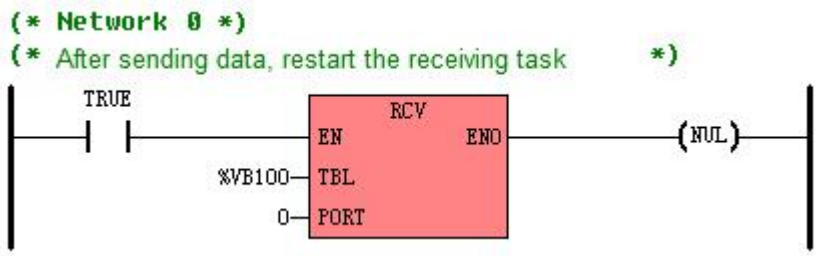


LD

EndReceive (INT00), Receive complete interrupt service routine:



EndSend (INT01), send Completing the interrupt service routine:



IL

主程序 (MAIN):

```
(* Network 0 *)
LD     %SM0.1
CAL    InitComm                (*At the first scan, execute the initialization subroutine once.*)
(* Network 1 *)
LD     %I0.0
R_TRIG
RCV    %VB100, 0               (*Use I0.0 to start receiving tasks.*)
```

InitComm (SBR00), Initialize subroutine:

```
(* Network 0 *)
LD     %SM0.0
MOVE   B#16#B6, %SMB87
MOVE   1, %SMW90               (* Set the starting and ending conditions for effective reception status *)
MOVE   B#16#D, %SMB89
MOVE   500, %SMW92             (* Reception preparation time set to 1ms *)
MOVE   B#100, %SMB94           (* Set the receiving end character to carriage return (ASCII is 13) *)
ATCH   EndReceive, 29
ATCH   EndSend, 30             (* Received character timeout set to 500ms *)
                               (* Set the maximum number of characters received each time to 100 *)
                               (* Bind EndReceive program to receive completion interrupt *)
                               (* Bind EndSend program to send completion interrupt *)
```

EndReive (INT00), Receiving completed interrupt service program:

```
(* NETWORK 0 *)
LD     %SM86.5
XMT    %VB100, 0               (* Send back the received data upon receiving the end character *)
RETC                           (* Then exit this program *)

(* NETWORK 1 *)
LD     %SM86.0
OR     %SM86.1
OR     %SM86.2
OR     %SM86.3
OR     %SM86.6                 (If the receiving status exits abnormally, restart the receiving task *)
RCV    %VB100, 0
```

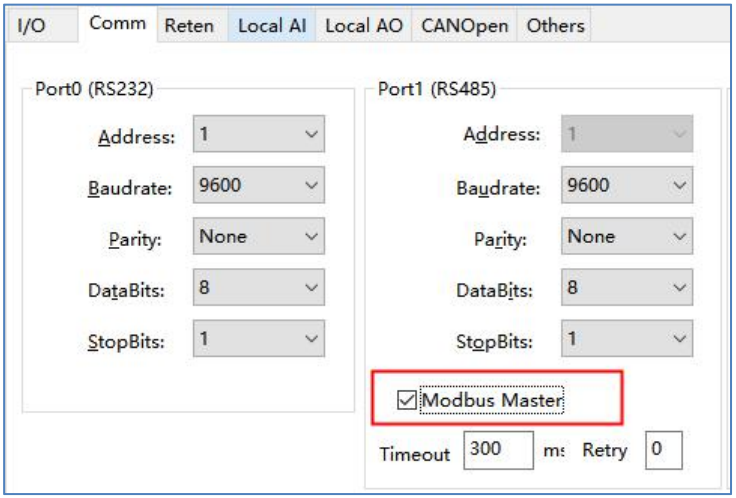
EndSend (INT01), Send completed interrupt service program:

```
(* NETWORK 0 *)
LD     TRUE
RCV    %VB100, 0                (* Restart receiving task after sending is completed *)
```

10.2.2 Modbus RTU communication function

10.2.2.1 Overview

The Modbus RTU protocol is one of the most widely used serial communication protocols at present, and it is a master-slave communication protocol. All serial communication ports of KPLC are defaulted as Modbus RTU slave stations, which can be used by users without programming. If the user needs PLC to be the Modbus RTU master station (only supported by RS485 port), you can check it in the hardware configuration and set the communication parameters.



10.2.2.2 KPLC memory area accessible by Modbus RTU master station

The KPLC memory areas accessible by the Modbus RTU master station are classified as follows:

Type	Modbus Function code	Corresponding PLC memory area
DO (switch output, 0XXXX)	1, 5, 15	Q area, M area
DI (digital input, 1XXXX)	2	Zone I, Zone M

AO (analog output, 4XXXX)	3, 6, 16	AQ area, V area
AI (analog input, 3XXXX)	4	AI area, V area
error log (16-bit unsigned integer)	3,4	PLC error record area

The maximum number of registers accessed by one instruction is as follows:

- Read "bit", a maximum of 1600 bits (200 bytes) can be read at a time. (function code 1, 2)
- Write "bit", a maximum of 800 bits can be written at a time. (function code 15)

3. Read "words", a maximum of 100 words can be read at a time. (function code 3, 4)

4. Write "words", a maximum of 100 words can be written at a time. (function code 16)

Note: When the size of the memory area to be operated is smaller than the above maximum value, it is only allowed to operate the entire memory area at one time. For example, for K506, the external master station reads 90 words in the AI area at a time, which is wrong, because the maximum number of words in the AI area is 32 words.

10.2.2.3 Modbus register number

If the registers required by the Modbus master station are numbered from 1, then add 1 directly to the register numbers in the table below. If the registers required by the Modbus master station are also numbered from 0, then directly correspond to the register numbers in the table below.

- Applicable to K5, KS, MK ,K6series

memory area	Range	Type	Corresponding Modbus register number*
I	I0.0 --- I31.7	DI	0 --- 255
Q	Q0.0 --- Q31.7	DO	0 --- 255
M	M0.0 --- M1023.7	DI/DO	320 -- 8511
AI	AIW0 --- AIW62	AI	0 --- 31
AQ	AQW0 --- AQW62	AO	0 --- 31
V	VW0 --- VW4094	AI/AO	100 -- 2147

Because the memory sizes of different series PLCs are different, the ranges allowed to be accessed are also different. For example, the length of V area of K6 is 16K bytes, so VW0--VW16382 can be accessed, but the length of V area of K2 is 4K bytes, so only VW0--VW4094 can be accessed.

However, different series of PLCs have the same starting Modbus register numbers for the same memory area, and the numbering rules are also consistent. For example, in all series of PLCs, the

register number of VW0 is 100, and the register number of VW2 is 101.

Please refer to 3.6.4 Address Range of Memory Area for the memory area range of each series of PLC.

In addition to the above memory areas, the memory area of KPLC's error records also supports reading through the Modbus RTU protocol. For details, please refer to 13.3 How to read the errors that have occurred in the PLC.

10.2.2.4 Basic format of Modbus RTU message

The CRC check code in the message is the high byte first, and the low byte last.

Message interval time not less than 3.5 characters long	target station number	Function code	Data	CRC check code
	1 byte	1 byte	N byte	2 byte

10.2.2.5 Introduction to Modbus RTU Instructions

The following description of the "response format" of each request instruction refers to the correct response format of the slave station. If it is an abnormal response, the "function code" part of the returned response frame becomes the following data: the data obtained after the highest position of the function code is 1. For example, if the function code is 0x01, if the slave responds abnormally, the returned function code is 0x81.

10.2.2.5.1 Function code 01: read coil (switch output)

Request format:

target station number	Function code	Start address		Read number		CRC
1 byte	01	High byte	Low byte	High byte	Low byte	2 byte

correct response format

Station number	Function code	The byte number of return data	byte 1 of return data	byte 2 of return data	…	CRC
1byte	01	1byte	1byte	1byte	…	2byte

10.2.2.5.2 Function code02: Read input status (switch input)

Request format:

Target station number	Function code	Start address		Read number		CRC
1byte	02	High byte	Low byte	High byte	Low byte	2byte

Correct response format:

Station number	Function code	The byte number of return data	byte 1 of return data	byte 2 of return data	…	CRC
1byte	02	1byte	1byte	1byte	…	2byte

10.2.2.5.3 Function code03: Read hold Register (analog output)

Request format:

Target station number	Function code	Start address		Read number		CRC
1byte	03	High byte	Low byte	High byte	Low byte	2byte

Correct response format:

Station number	Function code	The byte number of return data	Register1High byte	Register1Low byte	…	CRC
1byte	03	1byte	1byte	1byte	…	2byte

10.2.2.5.4 Function code04: Read input Register (analog input)

Request format:

Target station number	Function code	Start address		Read number		CRC
1byte	04	High byte	Low byte	High byte	Low byte	2byte

Correct response format:

Station number	Function code	The byte number of return data	Register1High byte	Register1Low byte	…	CRC
1byte	04	1byte	1byte	1byte	…	2byte

10.2.2.5.5 Function code05: Write single coil (switch output)

Request format:

Target station number	Function code	coil address		mandatory value		CRC
1byte	05	High byte	Low byte	High byte	Low byte	2byte

Note : mandatory value＝ 0xFF00, then set the coil to ON； mandatory value＝0x0000, Then set the coil to disconnect.。

Response format: If the setting is successful, the original message will be returned

10.2.2.5.6 Function code06: Write single hold Register (analog output)

Request format:

Target station number	Function code	Register address		mandatory value		CRC
1byte	06	High byte	Low byte	High byte	Low byte	2byte

Response format: If the setting is successful, the original message will be returned

10.2.2.5.7 Function code15: Write multi-coil (switch output)

Request format:

Target station number	Function code	Start address		Written number		mandatory value byte number	first byte of mandatory value	…	CRC
1byte	15	High byte	Low byte	High byte	Low byte	1byte	1byte	…	2byte

Correct response format:

Target station number	Function code	Start address		Written number		CRC
1byte	15	High byte	Low byte	High byte	Low byte	2byte

10.2.2.5.8 Function code16: Write multiple hold Register (analog output)

Request format:

Target Station number	Function code	Start address		Written number		mandatory value byte	mandatory value1 High byte	mandatory value1 Low byte	…	CRC
1byte	16	High byte	Low byte	High byte	Low byte	1byte	1byte	1byte	…	2byte

Correct response format:

Target station number	Function code	Start address		Written number		CRC
1byte	16	High byte	Low byte	High byte	Low byte	2byte

10.2.2.6 CRC Check Algorithm in Modbus Protocol

In the Modbus RTU protocol, CRC is used as the frame verification method. Below are two CRC algorithms written in C language.

10.2.2.6.1 Calculate CRC directly

```
/*Parameter: chData - const BYTE *, pointing to the first address of the storage area for the data to be verified
UNO - Number of bytes of data to be verified
Return value: WORD type, calculated CRC value*/
WORD  CalcCrc(constBYTE* chData, WORDuNo)
{
WORDcrc=0xFFFF;
    WORDwCrc;
    UCHARi,j;
    for (i=0; i<uNo; i++)
    {
        crc ^= chData[i];
        for (j=0; j<8; j++)
        {
            if (crc&1)
```

```
                {
                    crc>>= 1;
                    crc ^= 0xA001;
                }
                else
                    crc>>= 1;
            }
        }

        wCrc=( (WORD)LOBYTE(crc) )<<8;
        wCrc=wCrc|( (WORD)HIBYTE(crc) );
        return (wCrc);
}
```

10.2.2.6.2 Lookup table for quickly calculate CRC

```
/*High byte CRC table*/
const UCHAR auchCRCHi[] =
{
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
```

```
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;

/*Low byte CRC table*/
const UCHAR auchCRCLo[] =
{
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
```

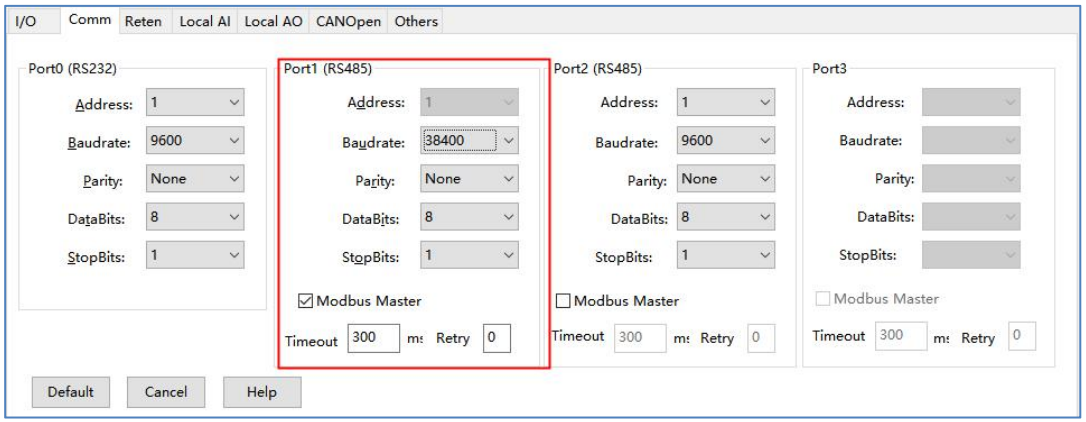
```
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;

/*Parameter: puchMSG - const BYTE *, pointing to the first address of the data storage area to be verified
UsDataLen - Number of bytes of data to be verified
Return value: WORD type, calculated CRC value*/
WORD CKINCOSerialCom::CalCrcFast(const BYTE* puchMsg , WORD usDataLen)
{
      BYTE uchCRCHi = 0xFF ;  /*CRC high byte initialization*/
   BYTE uchCRCLo = 0xFF ;  /*CRC low byte initialization*/
      WORD uIndex ;          /*Index of CRC lookup table*/
      while (usDataLen--)
      {
            uIndex = uchCRCHi ^ *puchMsg++ ;      /*Calculate CRC*/
            uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex];
            uchCRCLo = auchCRCLo[uIndex] ;
      }
      return (uchCRCHi << 8 | uchCRCLo) ;
}
```

10.2.2.7 Modbus RTU Master Instructions

In order to facilitate the user's application, KPLC provides Modbus RTU master station instructions (only for RS485 port), and the user can realize the function of the Modbus RTU master station by calling these instructions.

The following briefly describes the overall steps for the user to program the Modbus master station:

1) Set the serial communication parameters (including station number, baud rate, parity, etc.) of the communication port used in [PLC Hardware Configuration], and select the item [As Modbus master station], set the timeout time and the number of retries. For details, please refer to 4.3.4.1 CPU parameter configuration.



Timeout: within this period, if the master station does not receive a response message from a slave station, it will consider that the slave station has timed out, and the unit is ms.

Retry: When the master station receives a wrong response from a slave station or times out, it will continue to retry the number of times to communicate with the slave station.

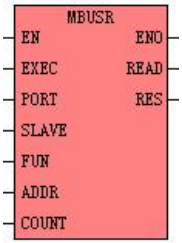
Directly call MBUSR and MBUSW instructions in the program for programming.

For a detailed description of the Modbus RTU protocol, please refer to 10.2.2.4 Modbus RTU message basic format.

The instructions described in this section are located in the [Instruction Set] -> [Communication Instructions] group.

10.2.2.7.1 MBUSR (Modbus master station read)

- Instruction and its operand description

	Name	Instruction format	CR value affected	
LD	MBUSR			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	MBUSR	MBUSR *EXEC, PORT, SLAVE, FUN, ADDR, COUNT, READ, RES*	U	

Parameters	Input/Output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
PORT	Input	INT	constant (0-2)
SLAVE	Input	BYTE	I、Q、M、V、L、SM、constant
FUN	Input	INT	constant (Modbus function code)
ADDR	Input	INT	I、Q、M、V、L、SM、AI、AQ、constant
COUNT	Input	INT	I、Q、M、V、L、SM、AI、AQ、constant
READ	Output	BOOL、WORD、INT	Q、M、V、L、SM、AQ
RES	Output	BYTE	Q、M、V、L、SM

Note: The parameters SLAVE, ADDR, COUNT must be both constant type or memory type at the same time; READ, COUNT parameters together form a memory block, and the memory addresses in the entire block must be legal addresses.

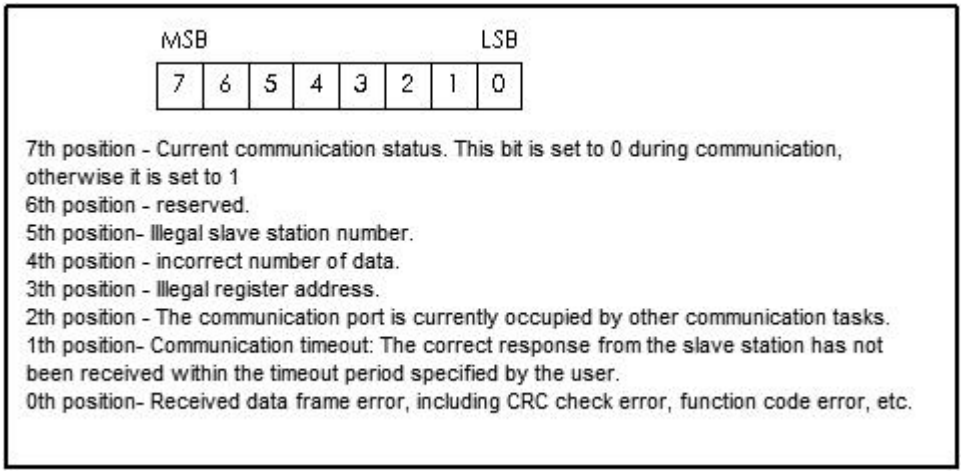
When KPLC is used as the Modbus RTU master station, the MBUSR instruction is used to read the data in the slave station. The applicable function codes of this instruction are 1 (read DO), 2 (read DI), 3 (read AO) and 4 (read AI).

The parameter PORT defines the communication port used. SLAVE defines the station number of the

target slave station, and the allowed station number range is 1~254. FUN defines the function code. ADDR defines the starting address of the register to be read. COUNT defines the number of registers to read, the maximum allowed is 32.

A rising edge transition on the EXEC input is used to initiate communication. When the MBUSR instruction is executed, if the rising edge transition of EXEC is detected, MBUSR will perform a communication. Organize the message according to the station number, function code and other parameters input by the user and complete the CRC check, then send the message and wait for the response from the slave station. After receiving the message returned from the slave station, CRC check, address check and function code check are performed on it. If it proves that the message is correct after verification, then the required data will be written into the data buffer, otherwise the received message will be discarded.

The parameter READ defines the starting address of the data buffer, and the read data (the number is COUNT) is stored in this area. READ must match the function code. If the function code is 1 or 2, then input a BOOL address variable; if the function code is 3 or 4, input an INT or WORD address variable. The parameter RES is used to store the current state information and the fault information of the latest communication, and its composition is as follows:



- LD

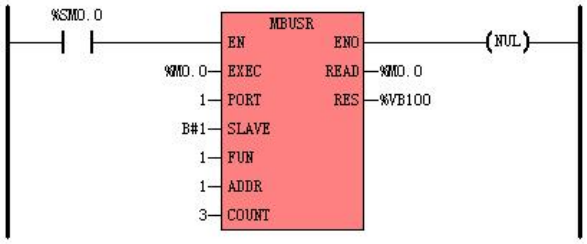
If EN is 1, the instruction is executed, otherwise it is not executed.

- IL

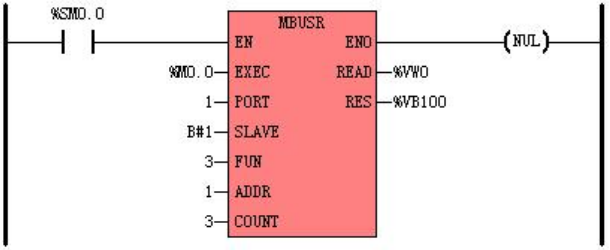
  If the CR value is 1, the instruction is executed, otherwise it is not executed.。

  The execution of this instruction does not affect the CR value.

The following example illustrates the relationship between the function code FUN, the number of COUNT, and READ in the instruction:

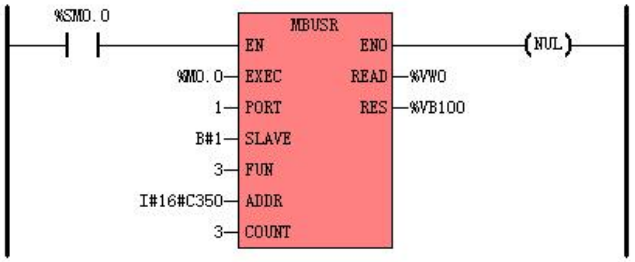


When the function code FUN is 1 or 2, the number COUNT refers to the number of bool type variables read, and the read data is stored in READ, and the corresponding READ should be filled with bool type address variables.



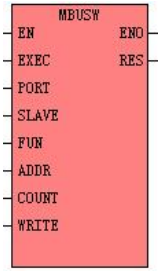
When the function code FUN is 3 or 4, the number COUNT refers to the number of read INT or WORD type variables, the read data is stored in READ, and the corresponding READ should be filled with INT or WORD type address variables.

In addition, it should be noted that the input of ADDR is INT, and the input range of the address is -32768-32767. When the address to be input exceeds 32767, it is necessary to fill in the hexadecimal value at the input end. For example, the address of ADDR is given as decimal 50000, which needs to be filled in as shown below:



10.2.2.7.2 MBUSW (Modbus master write)

- Instruction and its operand description

	Name	Instruction format	CR value affected	
LD	MBUSW			☑ K5 ☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	MBUSW	MBUSW *EXEC, PORT, SLAVE, FUN, ADDR, COUNT, READ, RES*	U	

Parameters	Input/Output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
PORT	Input	INT	constant (0-2)
SLAVE	Input	BYTE	I、Q、M、V、L、SM、constant

FUN	Input	INT	constant (Modbus function code)
ADDR	Input	INT	I、Q、M、V、L、SM、AI、AQ、constant
COUNT	Input	INT	I、Q、M、V、L、SM、AI、AQ、constant
WRITE	Input	BOOL、WORD、INT	I、Q、RS、SR、V、M、L、SM、T、C、AI、AQ
RES	Output	BYTE	Q、M、V、L、SM

Note: The parameters SLAVE, ADDR, COUNT Must be both constant type or memory type at the same time; READ, COUNT parameters together form a memory block, and the memory addresses in the entire block must be legal addresses.

When KPLC acts as the Modbus RTU master station, the MBUSW instruction is used to write data into the slave station. The applicable function codes of this instruction are 5 (write one DO), 6 (write one AO), 15 (write multiple DOs) and 16 (write multiple AOs).

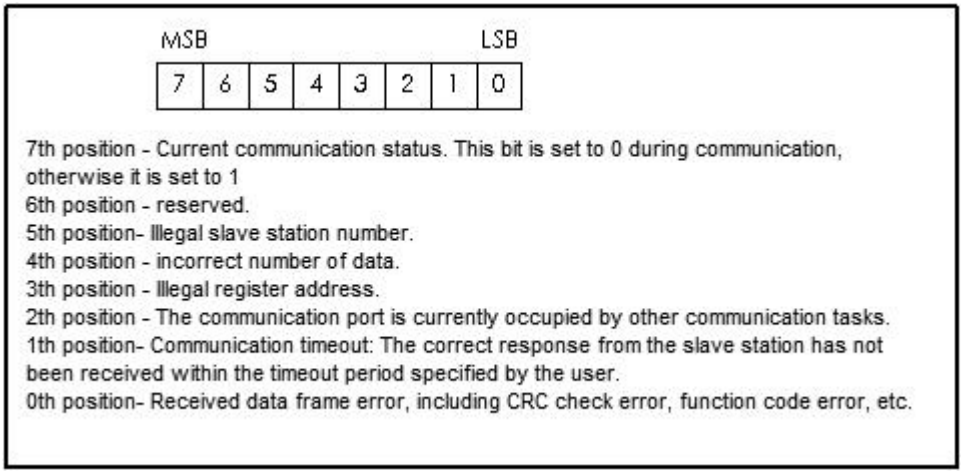
The parameter PORT defines the communication port used. SLAVE defines the station number of the target slave station, and the allowed station number range is 1~255. FUN defines the function code. ADDR defines the starting address of the register to be written. COUNT defines the number of write registers, the maximum allowed is 32.

The parameter WRITE defines the starting address of the data buffer, and the data to be written into the slave station is stored in this area. WRITE must match the function code. If the function code is 5, 15, then input a BOOL type address variable; if the function code is 6, 16, then input an INT or WORD type address variable.

A rising edge transition on the EXEC input is used to initiate communication. When the MBUSW instruction is executed, if the rising edge transition of EXEC is detected, MBUSW will conduct a communication. In this communication, the message is organized according to the target station number, function code, target register, quantity, written data and other parameters input by the user, and after completing the CRC check, the message is sent out and waits for the response from the slave station. When the message returned by the slave station is received, CRC check, address check and function code check are

performed on it to judge whether the slave station has correctly executed the write instruction just now.

The parameter RES is used to store the current state information and the fault information of the latest communication, and its composition is as shown in the figure below:



Note: If the value of the slave station number SLAVE is 255, it means this is a broadcast communication. All the slave stations in the network receive the message sent by the master station and perform the corresponding "write" register operation, but do not respond, so the master station will not wait to receive the response message.

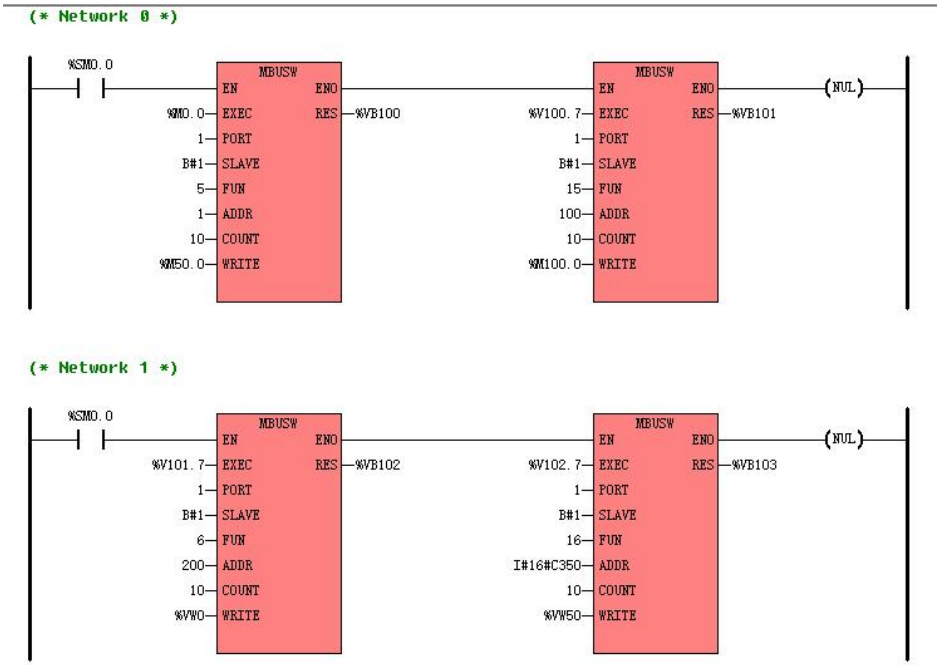
- LD

  If EN is 1, the instruction is executed, otherwise it is not executed.

- IL

  If the CR value is 1, the instruction is executed, otherwise it is not executed. The execution of this instruction does not affect the CR value.

  The following example illustrates the relationship between the function code FUN, the number of COUNT, and WRITE in the instruction:



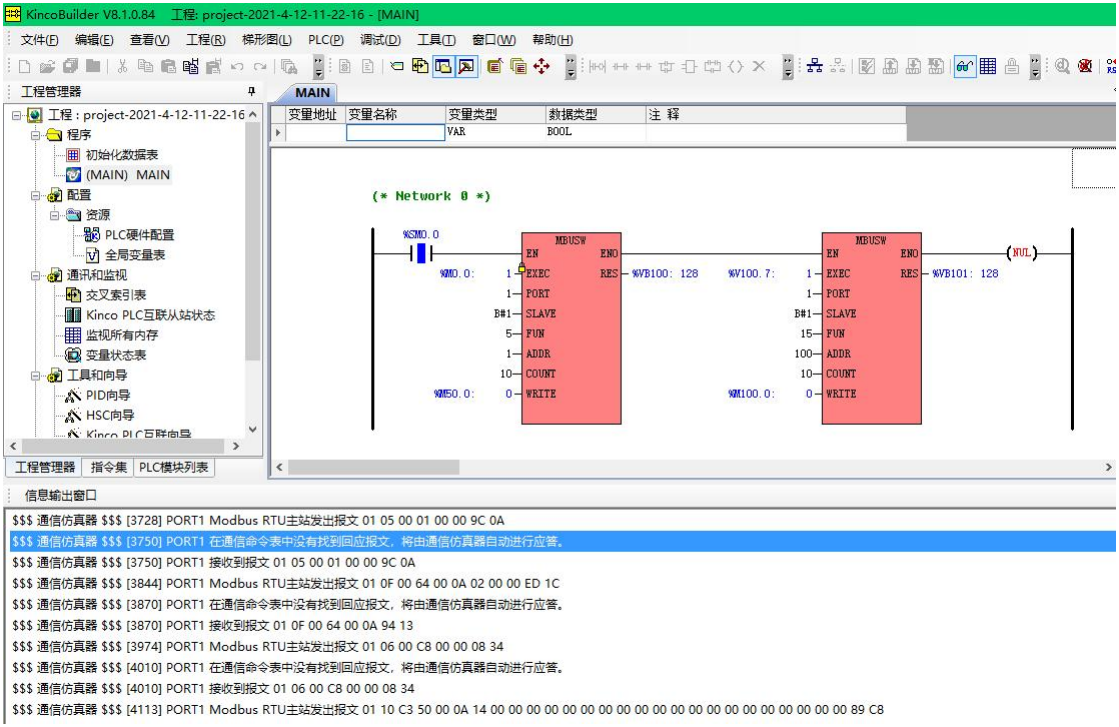
When the function code FUN is 5 or 6, the number COUNT has no effect, and the number is fixed at 1. When the function code FUN is 15, the number COUNT refers to the number of written BOOL variables. When the function code FUN is 16, the number COUNT refers to the number of INT or WORD type variables written.

When the function code FUN is 5 or 15, the start address of the corresponding WRITE should be a BOOL address variable.

When the function code FUN is 6 or 16, the start address of the corresponding WRITE should be an address variable of type INT or WORD.

In addition, it should be noted that the input of ADDR is INT, and the input range of the address is -32768-32767. When the address to be input exceeds 32767, it is necessary to fill in the hexadecimal value at the input end. For example, the address of ADDR is given as 50000 in decimal, which should be written as I#16#C350 in the above figure.

Note: When using Modbus instructions to read and write data with other devices, because the storage methods of the two devices are different, it may be necessary to exchange the high and low bytes of the data, so as to get the correct result. In a similar situation, you can use the serial port debugging tool to monitor the sending and receiving messages of the master and slave stations for analysis. KPLC fully complies with the standard Modbus RTU message format to send and receive messages. You can also use the offline simulation function 4.4 communication simulation that comes with the programming software to check the correct sending and receiving messages to find the problem, as shown in the figure below:

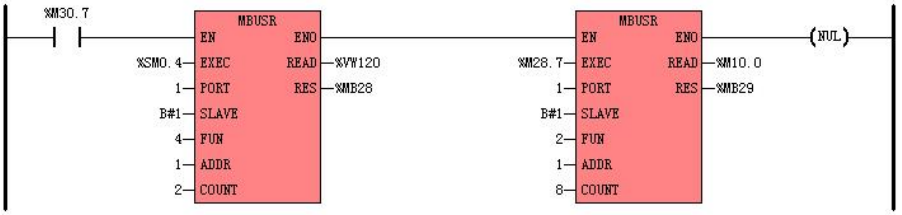


10.2.2.7.3 MBUSR、MBUSW using example

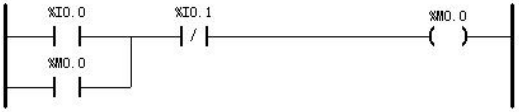
LD

```
(* Network 0 *)
(* M30.7 is used to mark whether the MBUSW command is communicating, and it will be updated after each communication. *)
```

```
(* Network 1 *)
(*If the communication port is not occupied by the MBUSW command, execute the MBUSR command:
  Read data from slave station 1 every 2 seconds: read AI registers 1 and 2 first, and then read DI registers 1-8 after completion. *)
```

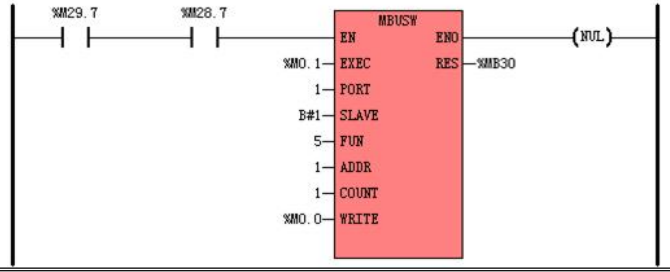



```
(* Network 2 *)
(* Example of MBUSW instruction *)
```




```
(* Network 3 *)
```

```
(* Network 4 *)
(*If the communication port is not occupied by the MBUSR command, execute the MBUSW command:
  Once I0.0 or I0.1 is turned on, it is immediately written to the 1 # D0 register of the 1 # slave statio *)
```

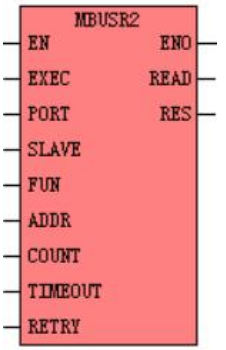


IL	(* Network 0 *) (M30.7 is used to mark whether the MBUSW instruction is communicating, and it will be updated after each communication.) LD %SM0.1 S %M30.7 (* Network 1 *) If the communication port is not occupied by the MBUSW instruction, execute the MBUSR instruction: (* Read data from 1 # slave station every 2 seconds: *) (* Read AI registers 1 and 2 first, and then read DI registers 1-8 after completion. *) LD %M30.7 MBUSR %SM0.4, 1, B#1, 4, 1, 2, %VW120, %MB28 MBUSR %M28.7, 1, B#1, 2, 1, 8, %M10.0, %MB29 (* Network 2 *) (*Example of MBUSW instruction *) LD %I0.0 OR %M0.0 ANDN %I0.1 ST %M0.0 (* Network 3 *) LD %I0.0 OR %I0.1 ST %M0.1 (* Network 4 *) (*If the communication port is not occupied by the MBUSR command, execute the MBUSW command: *) (*Once I0.0 or I0.1 is turned on, it is immediately written to the 1 # D0 register of the 1 # slave station*) LD %M29.7 AND %M28.7 MBUSW %M0.1, 1, B#1, 5, 1, 1, %M0.0, %MB30

10 .2.2.7.4 MBUSR2 (New Modbus master read command)

- instruction and its operand description

	Name	Instruction format	CR value affected	☑ MK

LD	MBUSR2			☑ K6
IL	MBUSR2	MBUSR2 *EXEC, PORT, SLAVE, FUN, ADDR, COUNT, TIMEOUT, RETRY, READ, RES*	U	

Parameter	Input/Output	Data Type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
PORT	Input	INT	Constant (0-2)
SLAVE	Input	BYTE	I、Q、M、V、L、SM、Constant
FUN	Input	INT	Constant (Modbus function code)
ADDR	Input	WORD	I、Q、M、V、L、SM、AI、AQ、Constant
COUNT	Input	INT	I、Q、M、V、L、SM、AI、AQ、Constant
TIMEOUT	Input	INT	I、Q、M、V、L、SM、AI、AQ、Constant
RETRY	Input	INT	I、Q、M、V、L、SM、AI、AQ、Constant
READ	Output	BOOL、WORD、INT	Q、M、V、L、SM、AQ
RES	Output	BYTE	Q、M、V、L、SM

Note: Parameters SLAVE, ADDR, COUNT, TIMEOUT, RETRY must be of either constant type or memory type. The READ and COUNT parameters together form a memory block, and the memory addresses in the whole block must be valid addresses.

When KPLC is used as the master station of Modbus RTU, the usage of MBUSR command is different, MBUSR2 and MBUSW2 take effect independently, and there is no need to set the master station in the communication setting page of hardware configuration.

This command applies to the following function codes: 1 (read DO), 2 (read DI), 3 (read AO), and 4 (read AI).

PORT defines the corresponding communication ports (1, 2, and 3, respectively corresponding to different serial ports).

SLAVE defines the ID of the target slave station. The allowed ID ranges from 1 to 255.

FUN defines the function codes (1, 2, 3, 4).

ADDR defines the starting address of the register of the slave station to be accessed.

COUNT defines the number of registers that need to be read. When the function code is 1 or 2, it supports a maximum read COUNT==1600 bits, that is, 200 bytes of valid data of the slave station. When the function code is 3 or 4: A maximum of 125 characters can be read at a time, that is, 250 bytes of valid data of the slave station.

TIMEOUT Specifies the timeout period of the packet. It is used only for the current packet.

RETRY defines the number of packet retries. If the value is set to 1 and the connection between the master station and the slave station fails, the master station sends two request packets and outputs one RES.7 rising edge only for the current packet.

EXEC A rising edge jump will trigger the execution of this instruction

When the MBUSR2 command is executed, if the rising edge of EXEC is detected, MBUSR2 will communicate once: organize the packet according to the station number, function code and other parameters entered by the user, complete CRC check, and then send the packet out and wait for the response from the slave station. After receiving the packets returned from the station, CRC, address, and function code checks are performed on the

packets. If the packets are correct, the required data is written into the data buffer. Otherwise, the received packets are discarded.

READ The parameter defines the starting address of the data buffer where the read data (COUNT) is stored. READ must match the function code. If the function code is 1 or 2, enter the address variable of type BOOL. If the function code is 3 or 4, enter an INT or WORD address variable.

RES The parameter is used to store the status information of the current command execution and the fault information of the latest communication.

Once the rising edge of EXEC is started, RES outputs res.7 == 0;

If the startup parameter is wrong, RES will output res.7 == 1 on the next scan.

When a packet in the queue is sent, RES outputs res.7 == 0.

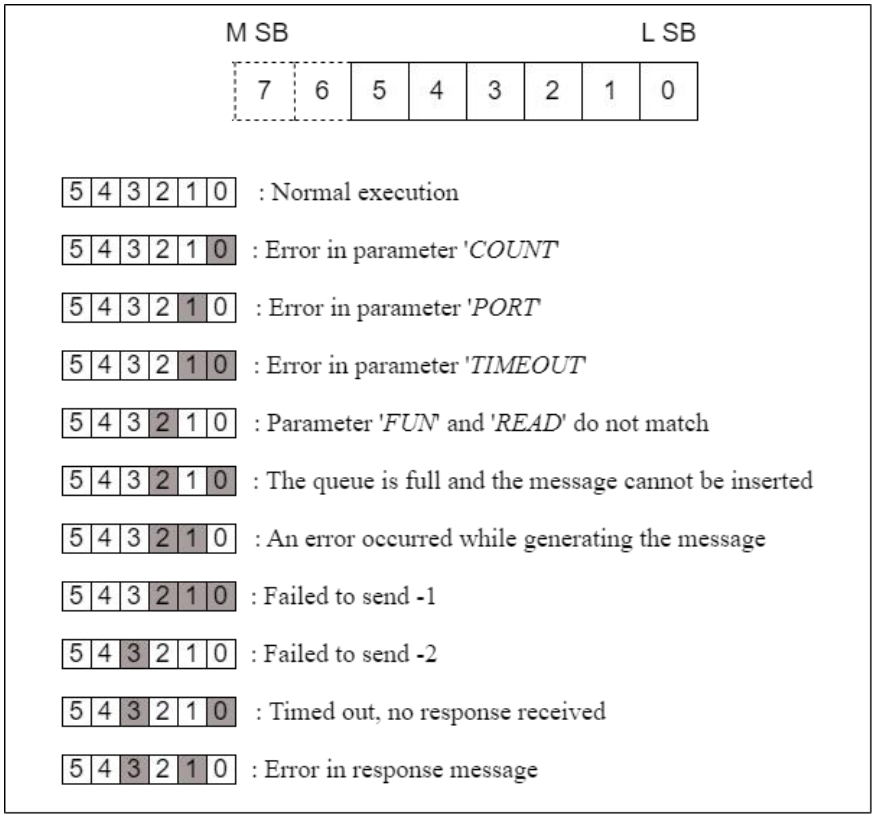
After the packet in the queue is completed (regardless of success or failure), RES outputs res.7 == 1;

Every time RES.7 is updated, RES.6 is updated, and RES.bit6 is used to indicate the current state:

RES.bit6 == 1: The current instruction block still has unsent packets in the queue.

RES.bit6 == 0: indicates that the packet submitted by the current block is normally sent.

When RES.7 rises, RES.0 to RES.5 indicates the following meanings:



- LD

  If EN is 1, the instruction is executed, otherwise it is not executed.

- IL

  If the CR value is 1, the instruction is executed, otherwise it is not executed.

  The execution of this command does not affect the CR value.

  other description:

  1. In the same serial port, such as Port1, this command cannot be used with XMT, RCV, COM_XMT, COM_RCV, COM_RCV2, MBUSR, MBUSW and other commands at the same time.

  2. MBUSR2 and MBUSW2 take effect independently. You do not need to set the primary

station on the communication Settings page of hardware configuration.

3. Function code 1,2, supports a maximum read COUNT==1600 bits, that is, read 200 bytes of valid data from the slave station;

Function code 3,4, supports a maximum read COUNT==125 words, that is, read 250 bytes of valid data from the slave station;

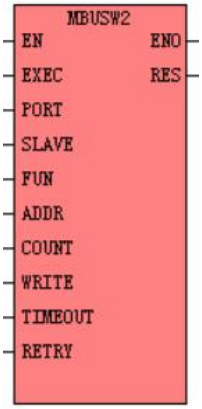
P.S.: COUNT indicates the maximum number of connected devices (that is, the upper limit). If the internal number of connected devices does not reach the upper limit, COUNT can only be set to the maximum number of currently connected devices. If the number exceeds the upper limit, a response message error will be caused.

4. Set the baud rate of the secondary and primary stations to the same baud rate.

5.Only KincoBuilder_V8.5.0.9 and later support offline emulation of MBUSW2 and MBUSR2 commands.

10.2.2.7.5 MBUSW2 (New Modbus master write command)

➢ instruction and its operand description

	Name	Instruction format	CR value affected	
LD	MBUSW2			☑ MK ☑ K6
IL	MBUSW2	MBUSW2 *EXEC, PORT, SLAVE, FUN, ADDR, COUNT, WRITE,*	U	

		TIMEOUT, RETRY, RES		

Parameter	Input/Output	Data Type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
PORT	Input	INT	Constant (0-2)
SLAVE	Input	BYTE	I、Q、M、V、L、SM、Constant
FUN	Input	INT	Constant (Modbus function code)
ADDR	Input	WORD	I、Q、M、V、L、SM、AI、AQ、Constant
COUNT	Input	INT	I、Q、M、V、L、SM、AI、AQ、Constant
WRITE	Input	BOOL、WORD、INT	I、Q、RS、SR、V、M、L、SM、T、C、AI、AQ
TIMEOUT	Input	INT	I、Q、M、V、L、SM、AI、AQ、Constant
RETRY	Input	INT	I、Q、M、V、L、SM、AI、AQ、Constant
RES	Output	BYTE	Q、M、V、L、SM

Note: Parameters SLAVE, ADDR, COUNT, TIMEOUT, RETRY must be of either constant type or memory type. The WRITE and COUNT parameters constitute a memory block. The memory addresses in the block must be valid addresses.

When KPLC is used as the master station of Modbus RTU, the usage of MBUSR command is different, MBUSR2 and MBUSW2 take effect independently, and there is no need to set the master station in the communication setting page of hardware configuration. This instruction applies to the function codes 5 (write a DO), 6 (write an AO), 15 (write multiple DO), and 16 (write multiple AO).

PORT The parameter defines the corresponding communication interface.

SLAVE Defines the ID of the target slave station. The allowed ID ranges from 0 to 255. Function code 5,6,15,16 Broadcast is supported. The address of the slave station is 0 during broadcast.

FUN Defines the function codes: 5 (write a DO), 6 (write an AO), 15 (write multiple DOS), and 16 (write multiple AOs).

ADDR Defines the starting address of the register to be written, corresponding to the function code's outgoing message format and the KPLC register address number, go to: 10.2.2.3 Modbus register number and 10.2.2.4 Modbus RTU message Basic Format.

COUNT Defines the number of registers to be written. If the function code is 5 or 6, only COUNT==1 is supported. If the function code is 15, COUNT==800 is supported, that is, 100 bytes of valid data is written to the slave station at a time. When the function code is 16, a maximum of COUNT==100 words is supported, that is, 200 bytes of valid data is written to the slave station at a time.

WRITE Defines the start address of the data buffer where data to be written to the slave station is stored. WRITE must match the function code. If the function code is 5 or 15, enter the address variable of type BOOL. If the function code is 6 or 16, enter an INT or WORD address variable.TIMEOUT 定义了报文超时时间, 仅仅用于当前报文。

RETRY Defines the number of packet retries. If the value is set to 1 and the connection between the master station and the slave station fails, the master station sends two request packets and outputs one RES.7 rising edge only for the current packet.

EXEC The rising edge jump will trigger the execution of this instruction, that is, add a message to the communication queue, and copy the values of M and V in the master station specified by the WRITE parameter to the message each time the rising edge will update the RES output. If there is an error in the parameter, Res.bit7 will perform a rising edge jump. RES.bit6 is used in conjunction with indicating the current state:

RES.bit6 == 1: The current instruction block still has unsent packets in the queue.RES.bit6 == 0: indicates that the packet submitted by the current block is normally sent

LD

If EN is 1, the instruction is executed, otherwise it is not executed.

IL

If the CR value is 1, the instruction is executed, otherwise it is not executed. The execution of this command does not affect the CR value.

Other description:

1. In the same serial port, such as Port1, this command cannot be used with XMT, RCV, COM_XMT, COM_RCV, COM_RCV2, MBUSR, MBUSW and other commands at the same time.

2.MBUSR2 and MBUSW2 take effect independently. You do not need to set the primary station on the communication Settings page of hardware configuration.

3. Function code 5,6, only COUNT==1 is supported.

Function code 15, support COUNT==800, that is, write 100 bytes of valid data from the slave station at a time;

Function code 16, the maximum support COUNT==100 words, that is, write 200 bytes of valid data from the slave station at a time.

P.S.:COUNT indicates the maximum number of connected devices (that is, the upper limit). If the internal number of connected devices does not reach the upper limit, COUNT can only be set to the maximum number of currently connected devices. If the number exceeds the upper limit, a response message error will be caused.

4. Function code 5,6,15,16 Broadcast is supported. The address of the slave station is 0 during broadcast.

5. The baud rate of the secondary and primary stations must be the same.

6.Only KincoBuilder_V8.5.0.9 and later support offline emulation of MBUSW2 and MBUSR2 commands.

10.2.3 Dynamic modification of RS485 communication port parameters

10.2.3.1 Overview

By default, KPLC needs to modify the parameters of each communication port in [PLC Hardware Configuration] and download them to the PLC to take effect.

At the same time, KPLC also provides the function of dynamically modifying the communication parameters of the RS485 port of the main body in the user program. PORT0 (RS232) may be frequently used as a programming port, so dynamic adjustment of communication parameters is not allowed.

- It is allowed to dynamically modify the three parameters of [PLC station number], [baud rate] and [parity check].
- Dynamically modified communication parameter values are stored in permanent memory and are permanently valid.
- The priority of the dynamically modified communication parameters is higher than the communication parameters in [PLC Hardware Configuration]. Even if the user re-downloads a new project, the PLC will give priority to the stored dynamic communication parameters. Users can use [PLC] -> [Clear...] menu instruction or the clear function mentioned below to clear the dynamic communication parameters.
- After the user modifies the communication parameters in the program, [PLC station number] will take effect immediately, but [baud rate] and [parity] are not necessarily. If the communication port is in idle state at that time, these two parameters will take effect immediately; if the communication port is in communication state at that time, these two parameters will be saved but will not take effect immediately.

  When the PLC is powered on next time, all modified communication parameters will take effect.

At present, there are two ways to modify the parameters of the RS485 communication port: one is to modify through related instructions, and the other is to modify through special registers. These two methods are applicable to different PLC models and are not supported at the same time.

10.2.3.2 Use instructions to read and write RS485 communication port parameters

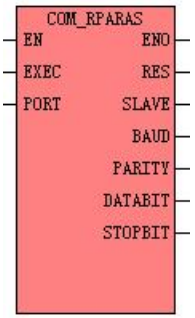
The following instructions are all located in the [Instruction Set] -> [Communication Instructions] group.

name	Functional description	Applicable PLC model
COM_RPARAS	Read RS485 communication port parameters	☑ K209M ☑ KS101M ☑ KW203 ☑ MK
COM_WPARAS	Modify RS485 communication port parameters	

		☑ K6

10.2.3.2.1 COM_RPARAS(Read RS485 communication port parameters)

- Instruction and its operand description

	Name	Instructions format	CR value affected	
LD	COM_RPARAS			☑ K209M ☑ KS101M ☑ KW203 ☑ MK ☑ K6
IL	COM_RPARAS	COM_RPARAS *EXEC, PORT, RES, SLAVE, BAUD, PARITY, DATABIT, STOPBIT*	U	

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
PORT	Input	INT	constant (1-2)
SLAVE	Output	BYTE	Q、M、V、L、SM
BAUD	Output	DINT	Q、M、V、L、SM
PARITY	Output	BYTE	Q、M、V、L、SM
DATABIT	Output	BYTE	Q、M、V、L、SM
STOPBIT	Output	BYTE	Q、M、V、L、SM
RES	Output	BYTE	Q、M、V、L、SM

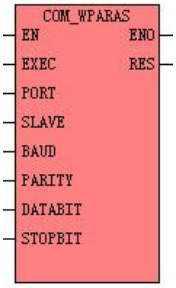
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
PORT	The communication port number used. 1 means PORT1, 2 means PORT2, and so on. If the parameter value specifies a non-existing communication port, it is an illegal value, causing the instruction to report an error.
SLAVE	The address where the read station number is stored
BAUD	The address where the read baud rate is stored
PARITY	The address where the read parity bit is stored

DATABIT	The address where the read data bits are stored
STOPBIT	The address where the read stop bit is stored
RES	RES is used to store the current state information and the latest fault information: Bit7- instructions execution is correct Bit0 - Unsupported PORT parameter

The rising edge transition of the EXEC input terminal is used to read the communication parameters of the RS485 port once. PORT represents the parameter of which PORT port to read. The read station number, baud rate, parity, data bits, and stop bits are stored in the addresses corresponding to SLAVE, BAUD, PARITY, DATABIT, and STOPBIT in sequence.

10.2.3.2.2 COM_WPARAS (Modify the RS485 communication port parameters)

- Instruction and its operand description

	Name	Instructions format	CR value affected	
LD	COM_WPARAS			☑ K209M ☑ KS101M ☑ KW203 ☑ MK ☑ K6
IL	COM_WPARAS	COM_RPARAS *EXEC, PORT, SLAVE, BAUD, PARITY, DATABIT, STOPBIT, RES,*	U	

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
PORT	Input	INT	constant (1-2)
SLAVE	Input	BYTE	Q、M、V、L、SM
BAUD	Input	DINT	Q、M、V、L、SM
PARITY	Input	BYTE	Q、M、V、L、SM
DATABIT	Input	BYTE	Q、M、V、L、SM
STOPBIT	Input	BYTE	Q、M、V、L、SM
RES	Output	BYTE	Q、M、V、L、SM

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
PORT	The communication port number used. 1 means PORT1, 2 means PORT2, and so on. If the parameter value specifies a non-existing communication port, it is an illegal value, causing the instruction to report an error.
SLAVE	The storage address of the station number that needs to be modified, ranging from 1-127
BAUD	The address where the baud rate needs to be modified is stored. Baud rate values can be 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
PARITY	The address where the parity bit to be modified is stored, 0-no parity, 1-odd parity, 2-even parity
DATABIT	The address where the data bit to be modified is stored, range 7, 8
STOPBIT	The address where the stop bit needs to be modified, range 1, 2
RES	RES is used to store the current state information and the latest fault information: Bit7- instruction execution is correct Bit6- Failed to write these parameters to permanent storage Bit5- unsupported STOPBIT parameter Bit4- Unsupported DATABIT parameter Bit3- PARITY parameter not supported Bit2- Unsupported BAUD parameters Bit1- unsupported SLAVE parameter, currently just cannot be 0 Bit0- Unsupported PORT parameter

The rising edge transition of the EXEC input terminal is used to modify the communication parameters of the RS485 port once. PORT represents which PORT parameter to modify. Write the parameters stored in SLAVE (station number), BAUD (baud rate), PARITY (parity check), DATABIT (data bit), STOPBIT (stop bit) to the corresponding PORT.

10.2.3.3 Use special registers to read and modify RS485 communication port parameters

Except for K209M, KS101M, KW203, MK, and K6, all other models of PLC need to use special registers to read and write RS485 communication port parameters.

10.2.3.3.1 Register description

SMB20--SMB25 is used to dynamically modify communication parameters. The meaning of each register is described in detail below.

- **Parameters value: SMB23、SMB24 and SMB25**

SMB	Description
SMB23	PLC station number value. Valid range of values: 1-31. If a write operation is performed, SMB23 is the new PLC station number value to be written; if a read operation is performed, SMB23 is the currently used PLC station number value read; if a clear operation is performed, SMB23 is ignored.
SMB24	Baud rate value. The valid range of values is 0-5: 0 means 2400, 1 means 4800, 2 means 9600, 3 means 19200, 4 means 38400, 5 means 1200. If a write operation is performed, SMB24 is the new baud rate value to be written; if a read operation is performed, SMB24 is the currently used baud rate value read; if a clear operation is performed, SMB24 is ignored.
SMB25	parity value. The valid range of values is 0-2, 0 means no test, 1 means odd test, 2 means even test. If a write operation is performed, SMB25 is the new parity value to be written; if a read operation is performed, SMB25 is the currently used parity value read; if a clear operation is performed, SMB25 is ignored.

- **Control byte: SMB20 and SMB21**

Bit	Description
SMB20: Specify the port number and operation mode to be operated.	
SM20.7	A value of 1 indicates that a write operation is enabled. After the PLC writes new parameters, it will automatically clear this bit to 0.
SM20.6	A value of 1 indicates that a read operation is enabled. After the PLC reads the parameters, it will automatically clear this bit to 0.
SM20.5	A value of 1 initiates a purge operation. After the PLC clears the parameters, it will automatically clear this bit to 0.
SM20.4	Reserved, must be assigned a value of 0.
SM20.3 SM20.0	The combined value of these 4 bits indicates the port number to be operated. 1 means PORT1, 2 means PORT2, if it is set to other invalid values, the PLC will report an error and exit the operation.
SMB21: Specify the communication parameters to be operated.	
SM21.7 -- SM21.3	spare. Must be assigned a value of 0.
SM21.2	A value of 1 means to modify or clear the parity value of the specified communication port.
SM21.1	A value of 1 means to modify or clear the baud rate value of the specified communication port.
SM21.0	A value of 1 means to modify or clear the PLC station number of the specified communication port.

At the same time, SM20.5, SM20.6, and SM20.7 only allow one bit to be 1, otherwise the PLC will report an error and exit the operation.

When performing a read operation, the value of SMB21 is ignored, and the PLC will read all

communication parameter values at one time.

When performing the clearing operation, the PLC allows clearing the station number, baud rate, and parity value separately or simultaneously. After a parameter is cleared, the PLC will automatically adopt the corresponding communication parameter value in the hardware configuration information.

- **Status byte: SMB22**

SMB22 stores the operation result of this dynamic adjustment of communication parameters.

Bit (read only)	Description
SM22.7	A value of 1 indicates that the operation is complete. After the PLC completes the operation specified by the user, whether it succeeds or fails, it will automatically set SM22.7 to 1. Only when the value of SM22.7 is 1, the values of other bits in SMB22 have practical significance.
SM22.6	If the value of SM22.7 is 1, then if the value of SM22.6 is 1, it means that the operation is successful; if the value of SM22.6 is 0, it means that the operation failed due to an error.
SM20.5 SM20.0	If this operation fails, the combined value of these 6 bits indicates the error code that occurred, and the meaning is as follows.

Error code	Error description
1	Wrong operation instruction. For example, SM20.7 and SM20.6 are set to 1 at the same time.
2	wrong port number
3	The value of SMB21 (the communication parameter to be operated) is wrong.
4	The value of SMB23 (new PLC station number) is wrong.
5	SMB24 (new baud rate) value is wrong.
6	SMB25 (new parity) value error.
10	Failed to read the stored PORT1 dynamic PLC station number from the permanent memory.
11	The dynamic PLC station number of PORT1 has not been set.
12	Failed to read stored PORT1 dynamic baud rate value from persistent memory.
13	The dynamic baud rate of PORT1 has not been set.
14	Failed to read stored PORT1 dynamic parity value from persistent storage.
15	The dynamic parity value of PORT1 has not been set.
20	Failed to read the stored PORT2 dynamic PLC station number from the permanent memory.
21	The dynamic PLC station number of PORT2 has not been set.
22	Failed to read stored PORT2 dynamic baud rate value from persistent memory.
23	The dynamic baud rate of PORT2 has not been set.
24	Failed to read stored PORT2 dynamic parity value from persistent storage.
25	The dynamic parity value of PORT2 has not been set.
61	Failed to write dynamic communication parameter value to permanent memory.

10.2.3.3.2 Using instruction

- Modify PLC communication parameters

1) Set the lower 4 bits of SMB20 to the port number to be operated.

   For example, SMB20=B#1 means that the parameters of PORT1 will be modified.

2) According to the parameter type to be modified, assign the corresponding value to SMB21.

   For example, SMB21=B#16#03 indicates that the PLC station number and baud rate value will be modified.。

3) Assign the desired new parameter value to the corresponding register: SMB23 is the new PLC station number value, SMB24 is the new baud rate value, and SMB25 is the new parity value.

   For example, SMB23=B#03 means to change the PLC station number to 3, and SMB24=B#3 means to change the baud rate to 19200.

4) (Optional) If the communication parameter operation (reading, writing or clearing) has been started once, then the completion flag bit SM22.7 must be checked first, and this operation can be started only when SM22.7 is 1.

5) Assign SM20.7 to 1 to start this write operation. PLC will automatically clear SM20.7 to zero after writing new parameters.

6) (Optional) Check the flag bits SM22.7 and SM22.6, the values of these two flag bits are both 1, indicating that the parameter modification is successful.

- Read PLC communication parameters

1) Set the lower 4 bits of SMB20 to the port number to be operated.

   For example, SMB20=B#1 means that the parameters of PORT1 will be read.

2) (Optional) If the communication parameter operation (reading, writing or clearing) has been started once, then the completion flag bit SM22.7 must be checked first, and this operation can be started only when SM22.7 is 1.

3) Assign SM20.6 to 1 to start this read operation. PLC will automatically clear SM20.6 to zero after reading

new parameters.

4) Check the flag bits SM22.7 and SM22.6, the value of these two flag bits is 1, which means that the parameter reading is successful. After reading successfully, the parameter value is stored in the following registers: SMB23 is the PLC station number value, SMB24 is the new baud rate value, and SMB25 is the parity check value.

- Clear PLC communication parameters

1) Set the lower 4 bits of SMB20 to the port number to be operated.
   For example, SMB20=B#1 means that the parameters of PORT1 will be modified.

2) According to the parameter type that needs to be cleared, assign the corresponding value to SMB21
   For example, SMB21=B#16#03 indicates that the dynamic PLC station number and baud rate stored in the permanent memory will be cleared.。

3) (Optional) If the communication parameter operation (reading, writing or clearing) has been started once, then the completion flag bit SM22.7 must be checked first, and this operation can be started only when SM22.7 is 1.

4) Set SM20.5 to 1 to start this clearing operation. PLC will automatically clear SM20.5 to zero after clearing.。

5) (Optional) Check the flag bits SM22.7 and SM22.6, the value of these two flag bits is 1, which means clearing the parameters is successful.

10.2.3.3.3 Example

The following example is to modify the dynamic station numbers of PORT1 and PORT2 on the HMI, and the PLC used is KS. The program adopts the IL language, and the user can directly copy it to the editor of KincoBuilder and execute the [Project] -> [LD Language] menu instruction to convert it into a ladder diagram. VW48 is the value of the new station number, which can be modified on the HMI. At the same time, the value of VW48 is also permanently stored in VW3690 of KS (different models of PLC have different permanent storage areas, refer to 12.1 Data retention and data backup for details). The PLC program checks the real-time

value of VW48 and compares it with the value stored in VW3690. If the value changes and VW48 is a legal value allowed, VW48 is used as the new station number value of PORT1 and PORT2 and the modification is started.

```
(* Network 0 *)
(*When powered on, initialize VW48 using the permanently stored values.*)
LD    %SM0.1
MOVE  %VW3690, %VW48
(* Network 1 *)
(*Determine whether the VB48 value has changed and whether the value is valid. Then save VW48 and initiate the modifications.*)
LD    %SM0.0
GE    %VB48, B#1
LE    %VB48, B#31
NE    %VW48, %VW3690
MOVE  %VW48, %VW3690
ST    %M999.7
(* Network 2 *)
(*Start modifying PORT1 station number.*)
LD    %M999.7
R_TRIG
MOVE  B#1, %SMB20
MOVE  B#1, %SMB21
MOVE  %VB48, %SMB23
S     %SM20.7
S     %M999.6
(* Network 3 *)
(*After successfully modifying the PORT1 parameter, modify the PORT2 station number.*)
LD    %M999.6
AND   %SM22.7
R_TRIG
AND   %SM22.6
```

```
MOVE    B#2, %SMB20
S       %SM20.7
R       %M999.6
```

10.3 Use of Ethernet port

10.3.1 Description

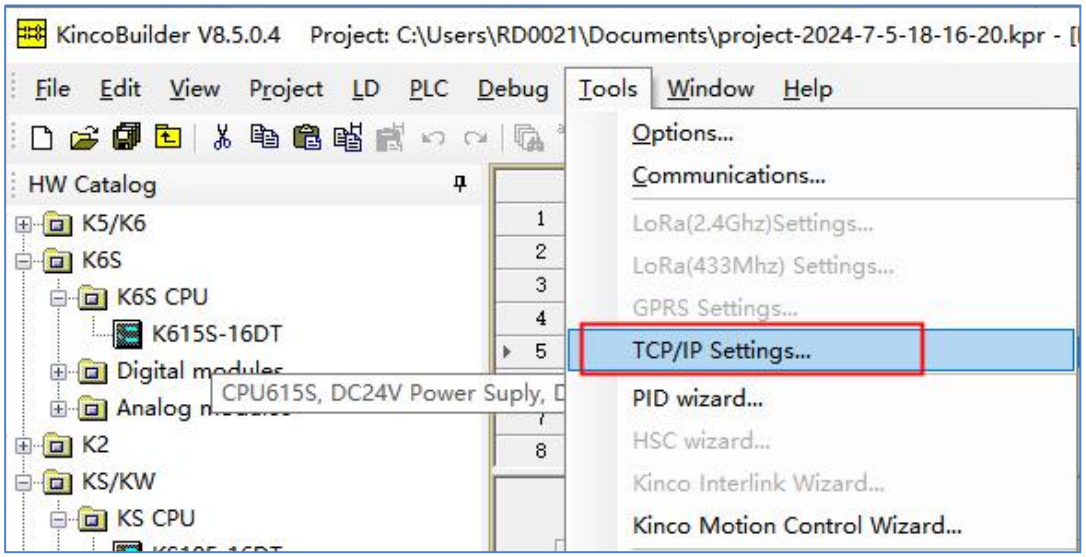
KPLC Ethernet port support Programming protocol, ModbusTCP protocol (master-slave), TCP and UDP free communication.

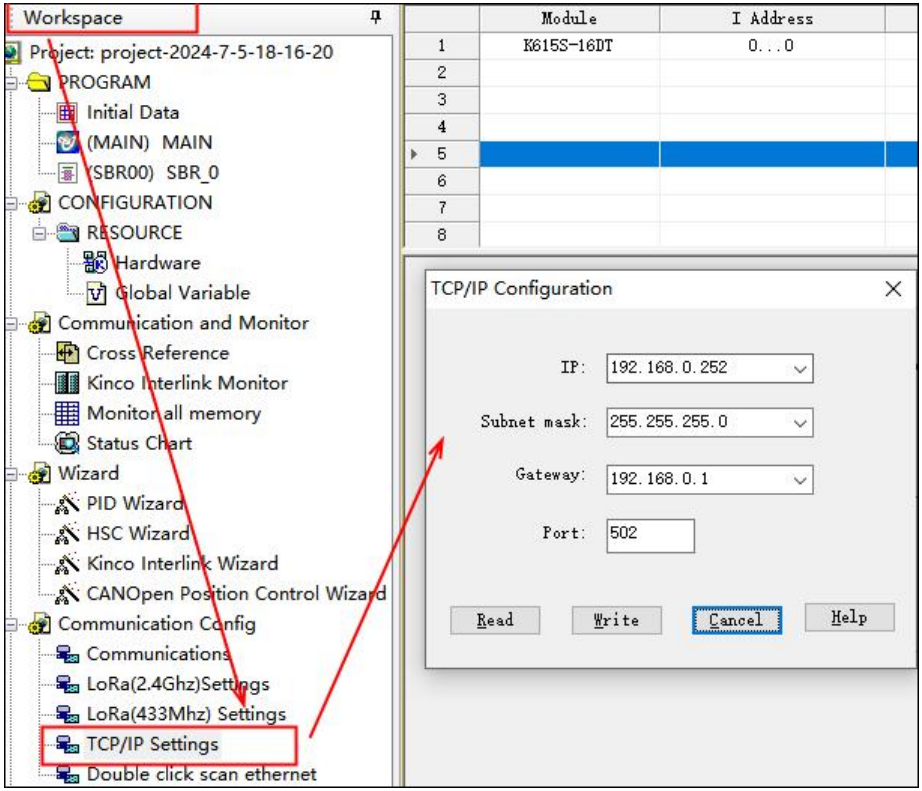
module	ModbusTCP Slave Protocol	ModbusTCP Master protocol	Free Communication
K6S	Supported	Supported	Supported
K6	Supported	Supported	Supported

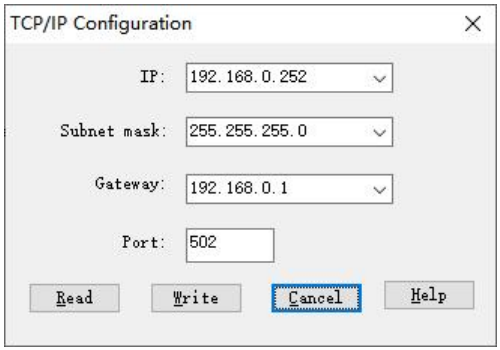
10.3.2 Network port parameter setting

10.3.2.1 Local network port parameters

Clicking 【Tool】-> 【TCP/IP Parameters】the TCP/IP interface parameter window will pop up, or click on [Project Manager] -> [Communication Parameters] on the left, you can read and write the parameters which include IP address, subnet mask, and gateway of the PLC

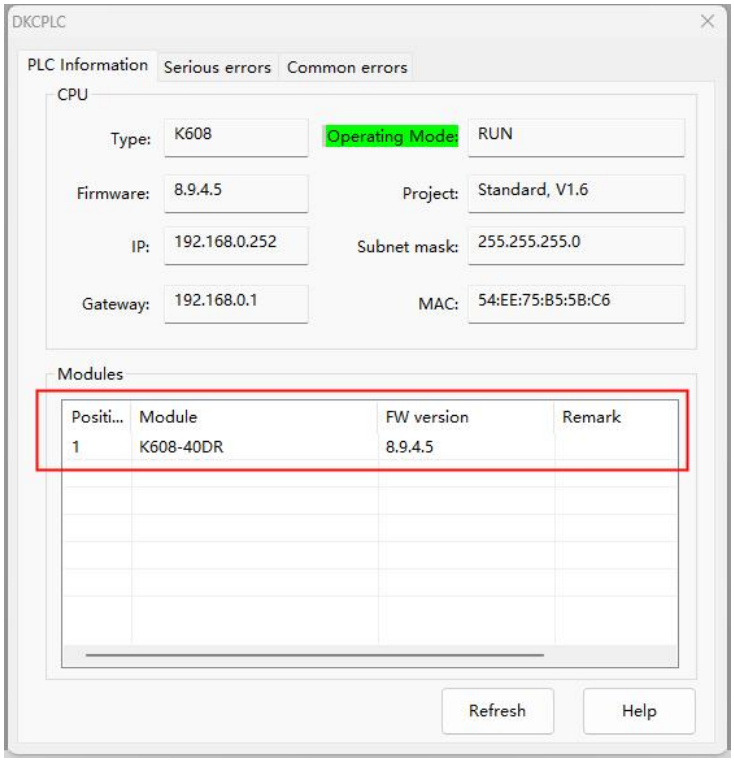

KincoBuilder V8.5.0.4 Project: C:\Users\RD0021\Documents\project-2024-7-5-18-16-20.kpr - [
File Edit View Project LD PLC Debug Tools Window Help
HW Catalog
K5/K6
K6S
K6S CPU
K615S-16DT
Digital modules
Analog modules
K2
KS/KW
KS CPU
CPU615S, DC24V Power Suply, D
Options...
Communications...
LoRa(2.4Ghz)Settings...
LoRa(433Mhz) Settings...
GPRS Settings...
TCP/IP Settings...
PID wizard...
HSC wizard...
Kinco Interlink Wizard...
Kinco Motion Control Wizard...





The above figures shows the default IP and port of the network PLC.. The network parameters and Mac address of the current real-time PLC can be viewed in the dialog window which is displayed after the

user connects the PLC through the programming port and executes the menu instruction [PLC] -> [PLC Information] in KincoBuilder as shown below. It should be noted that the Mac address of Kinco PLC does not currently support modification.



10.3.2.2 Ethernet communication setting

The CPU with the network port can realize the communication between the PC and the CPU through the network port. First, you need to open the communication setting screen as shown in Figure 1.

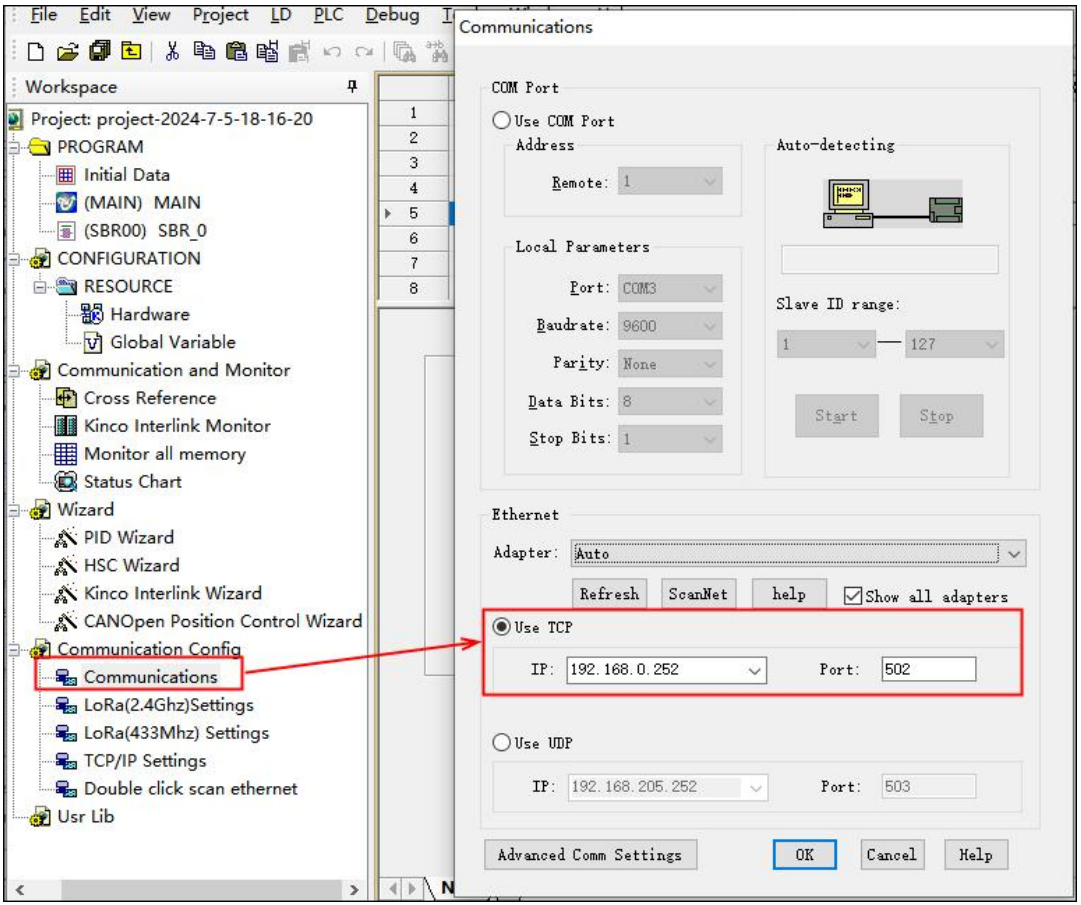


Figure 1

1、Use TCP/UDP: When the IP address and port number of the CPU are known, the parameters can be directly written, and the corresponding CPU or download program can be monitored. The transmission method can be selected from TCP or UDP.

Note: When there are multiple CPUs with the same IP address and port number in the same network, you cannot monitor or download programs through the network port, so as to avoid unexpected errors.

2、Scan network: In the same local area network, you can scan all devices in the network through the scan network function. This function supports scanning across network segments. as shown in figure 2:

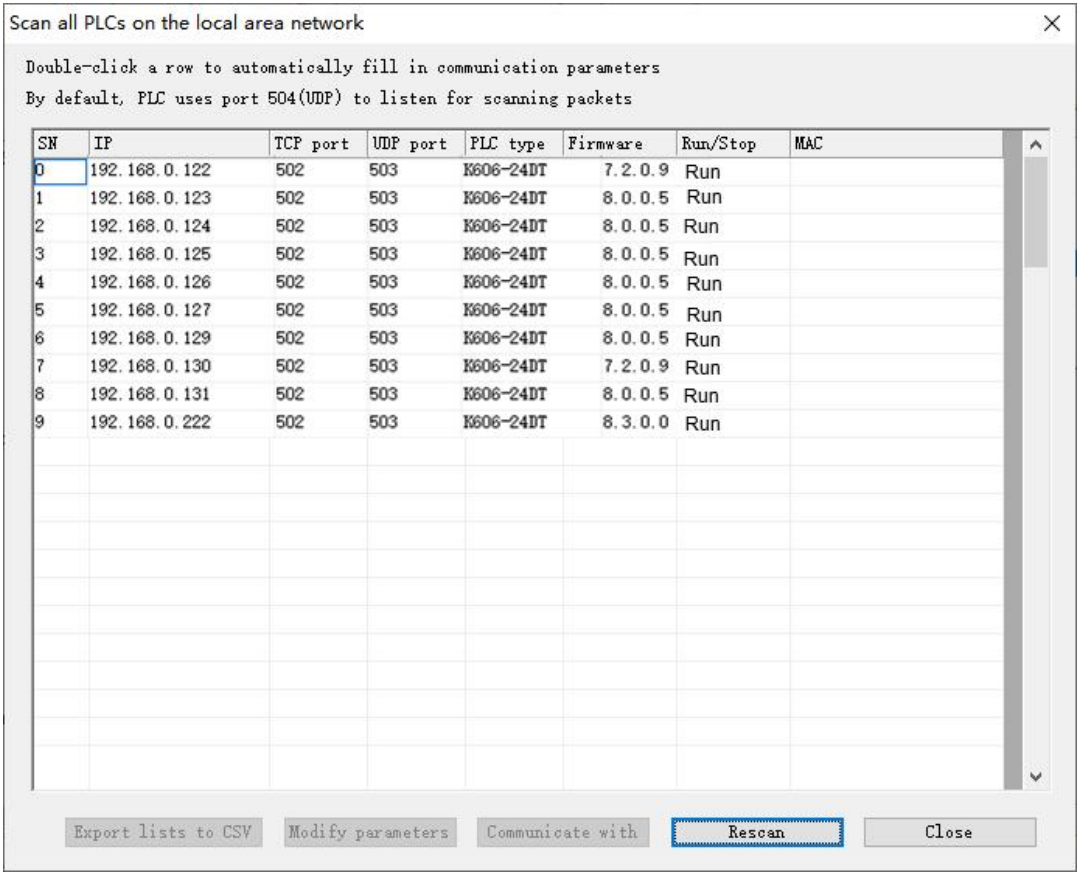

Scan all PLCs on the local area network
Double-click a row to automatically fill in communication parameters
By default, PLC uses port 504(UDP) to listen for scanning packets
SN IP TCP port UDP port PLC type Firmware Run/Stop MAC
0 192.168.0.122 502 503 K606-24DT 7.2.0.9 Run
1 192.168.0.123 502 503 K606-24DT 8.0.0.5 Run
2 192.168.0.124 502 503 K606-24DT 8.0.0.5 Run
3 192.168.0.125 502 503 K606-24DT 8.0.0.5 Run
4 192.168.0.126 502 503 K606-24DT 8.0.0.5 Run
5 192.168.0.127 502 503 K606-24DT 8.0.0.5 Run
6 192.168.0.129 502 503 K606-24DT 8.0.0.5 Run
7 192.168.0.130 502 503 K606-24DT 7.2.0.9 Run
8 192.168.0.131 502 503 K606-24DT 8.0.0.5 Run
9 192.168.0.222 502 503 K606-24DT 8.3.0.0 Run
Export lists to CSV
Modify parameters
Communicate with
Rescan
Close

Figure 2

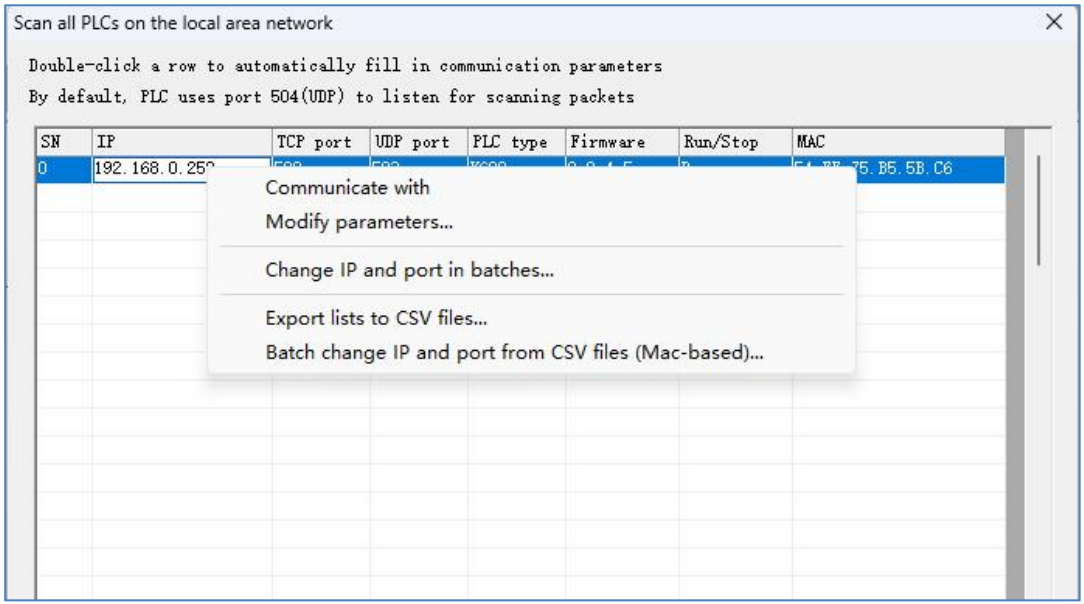


Figure 3

3、If you need to modify the parameters of this device: left-click to select the device serial number you want to modify in the above picture. Right-click to select modify the device parameters. The window will pop up as shown below (figure 4), and you can modify the Ethernet parameters of this device.

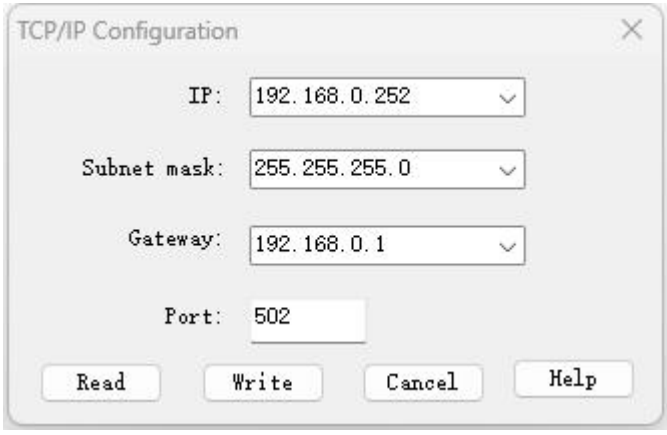


Figure 4

4 、 Communicate with device: Click Communicate with this device to jump back to Figure 1, the

communication parameters of TCP or UDP will automatically change to the IP address and port number of the device you want to communicate with.

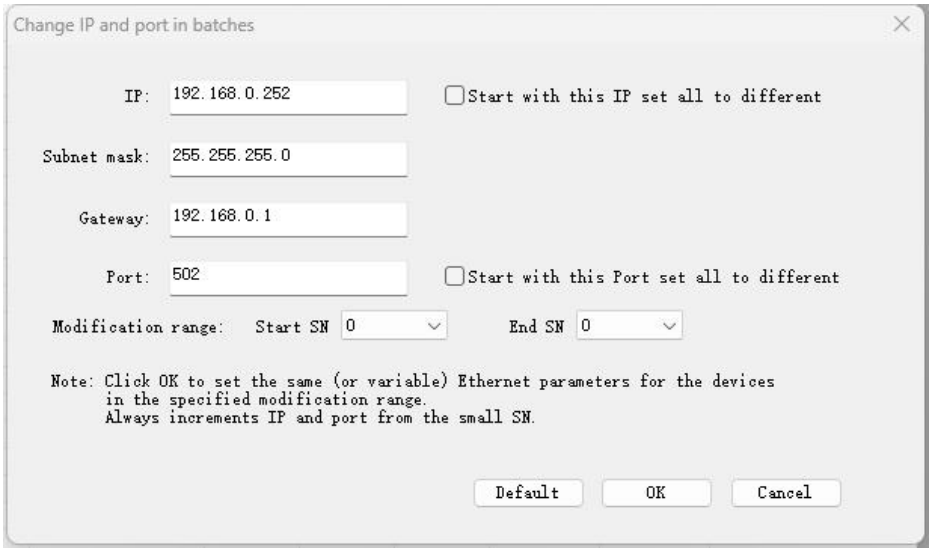
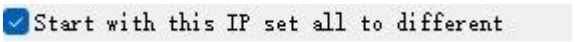


Figure 5

5、Batch modify IP and port directly in the software: Users can use this method to modify the parameters of all devices in the LAN in batches.

① Click yes when are not checked, all devices in the network will be modified to the same IP address, subnet mask, gateway, and port number. Figure 6.

② Click yes when are  checked, the device in the network will use this IP as the starting IP and this port as the starting port number to increase. In Figure 6, the IP addresses and port numbers of all devices in the network before modification are the same, and Figure 7 shows the effect after batch modification.

Note: Modification range: start serial number - end serial number refers to the PLC serial number in the

scanned network, you can only modify the parameters of the PLC with the specified serial number.

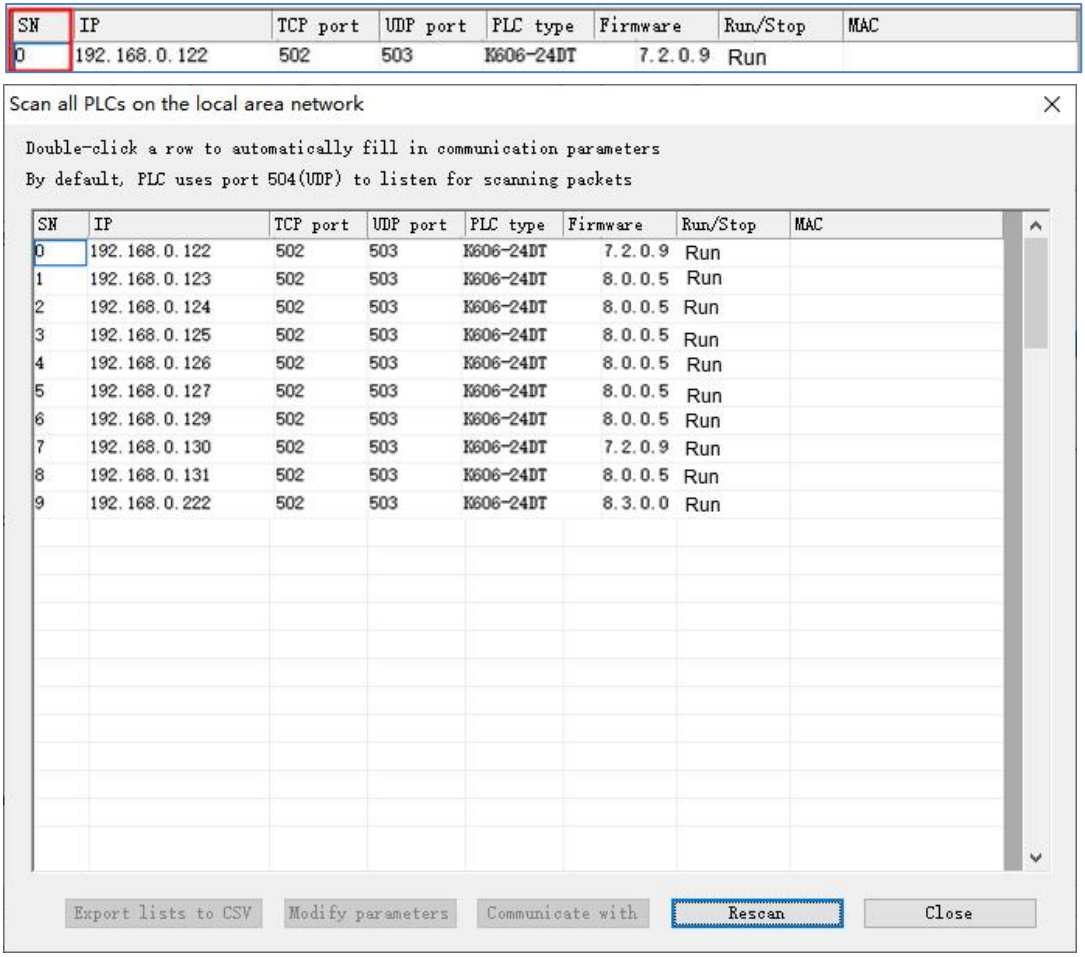


Figure 6

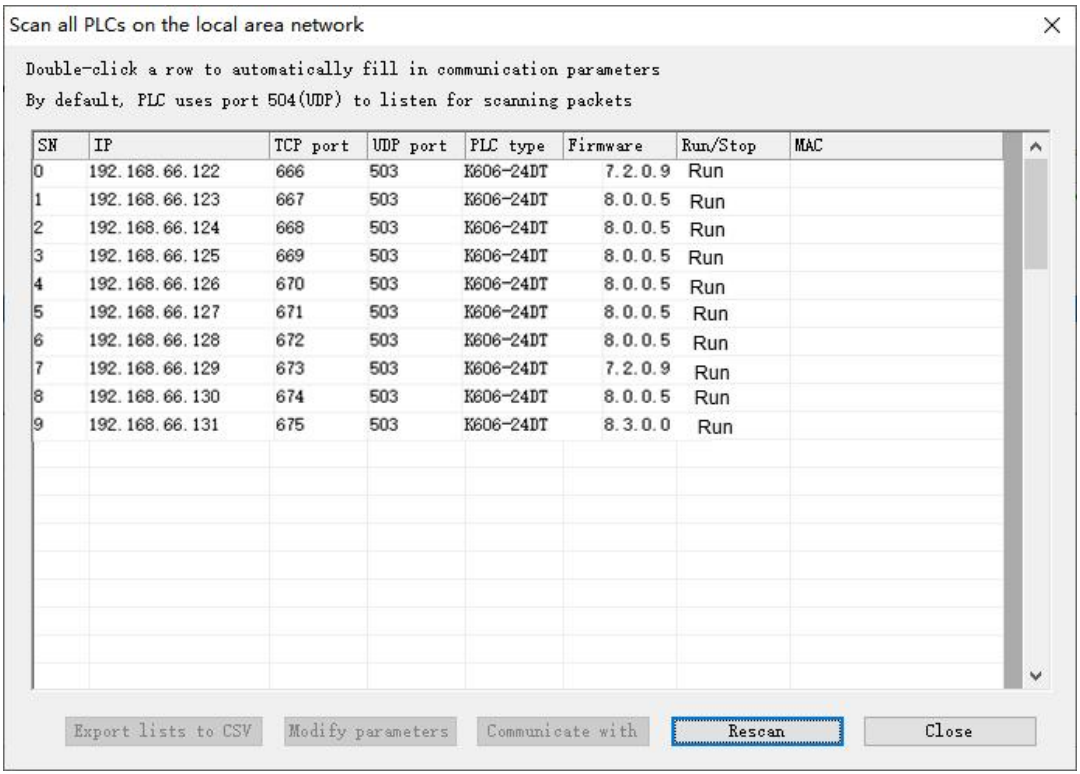
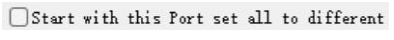


Figure 7

③ Click yes when



, The devices in the network increment with this IP as the starting IP, and the port numbers of all devices become the same.

④ Click yes when

,The devices in the network increment with this port as the starting port number, and the IP addresses of all devices become the same.

6、Batch modification of IP, port and other parameters through CSV files.

① Export the list of all PLC parameters in the network to a CSV file, as shown in Figure 1 and Figure 2.

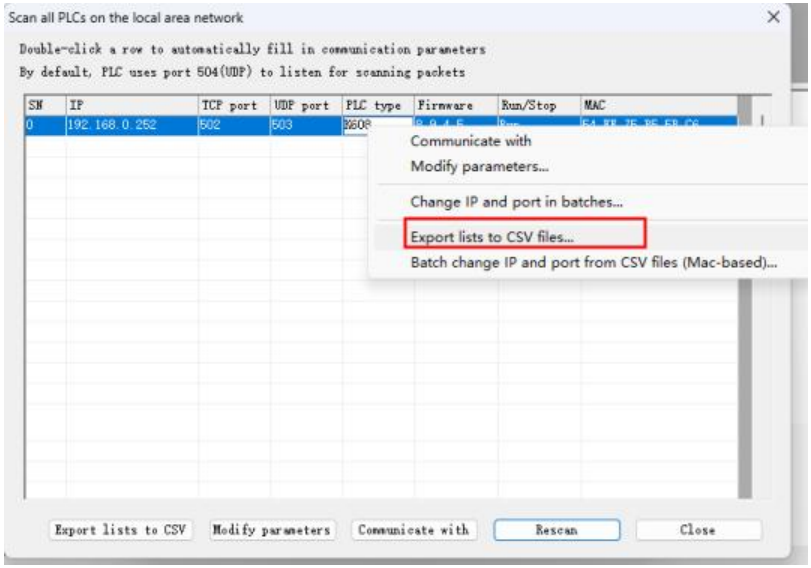
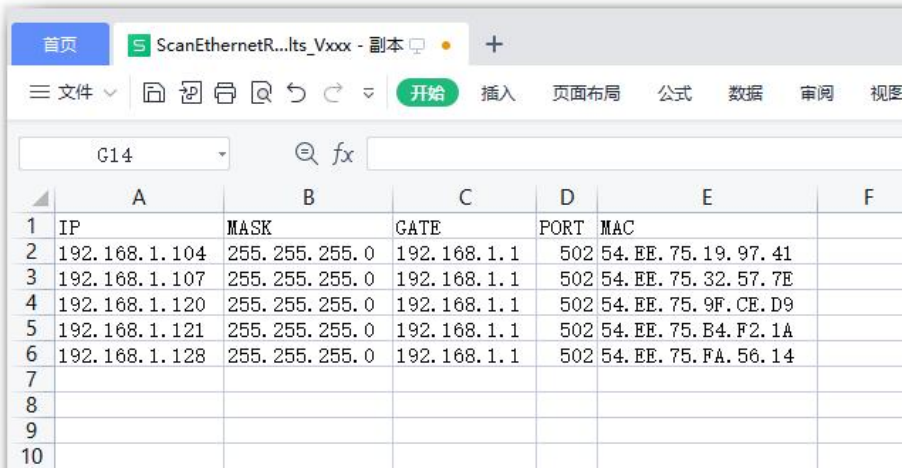


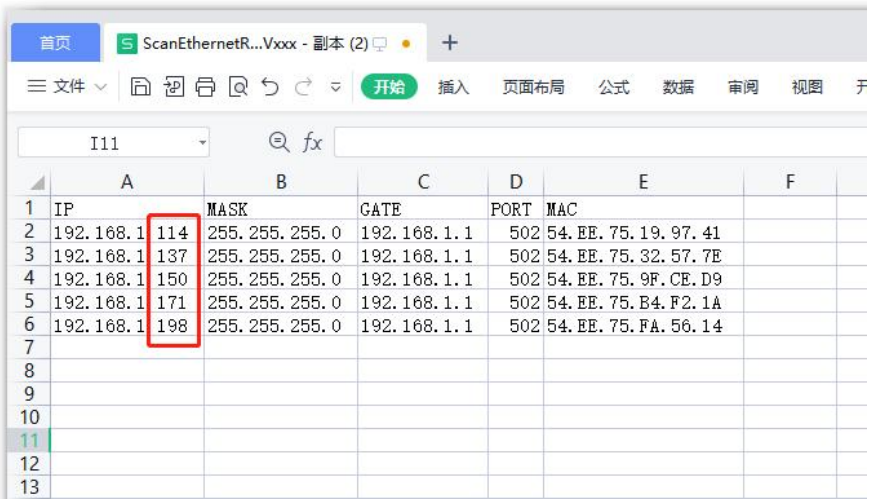
Figure 1



	A	B	C	D	E	F
1	IP	MASK	GATE	PORT	MAC	
2	192.168.1.104	255.255.255.0	192.168.1.1	502	54.EE.75.19.97.41	
3	192.168.1.107	255.255.255.0	192.168.1.1	502	54.EE.75.32.57.7E	
4	192.168.1.120	255.255.255.0	192.168.1.1	502	54.EE.75.9F.CE.D9	
5	192.168.1.121	255.255.255.0	192.168.1.1	502	54.EE.75.B4.F2.1A	
6	192.168.1.128	255.255.255.0	192.168.1.1	502	54.EE.75.FA.56.14	
7						
8						
9						
10						

Figure 2

② Modify related parameters in the exported CSV file.



	A	B	C	D	E	F
1	IP	MASK	GATE	PORT	MAC	
2	192.168.1.114	255.255.255.0	192.168.1.1	502	54.EE.75.19.97.41	
3	192.168.1.137	255.255.255.0	192.168.1.1	502	54.EE.75.32.57.7E	
4	192.168.1.150	255.255.255.0	192.168.1.1	502	54.EE.75.9F.CE.D9	
5	192.168.1.171	255.255.255.0	192.168.1.1	502	54.EE.75.B4.F2.1A	
6	192.168.1.198	255.255.255.0	192.168.1.1	502	54.EE.75.FA.56.14	
7						
8						
9						
10						
11						
12						
13						

Figure 3

③ Import the modified CSV file in.

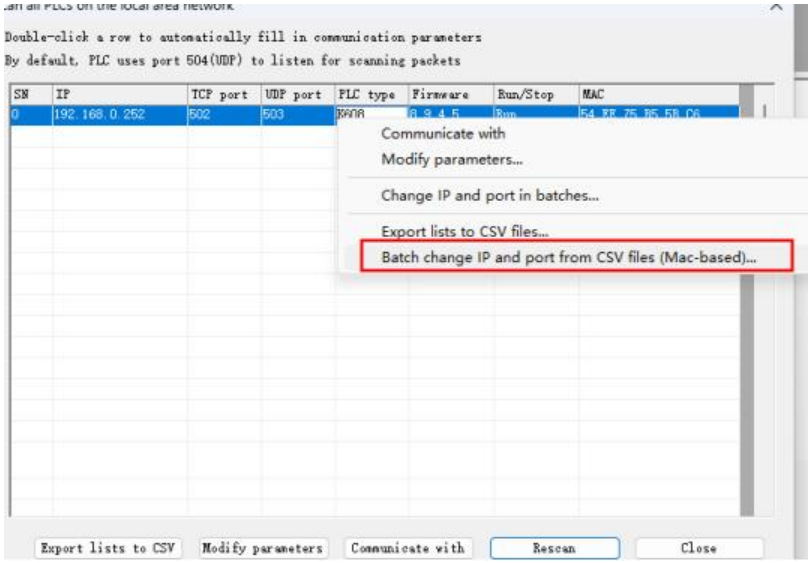


Figure 4

④ After the import is complete, a window for rescanning the observations will pop up.

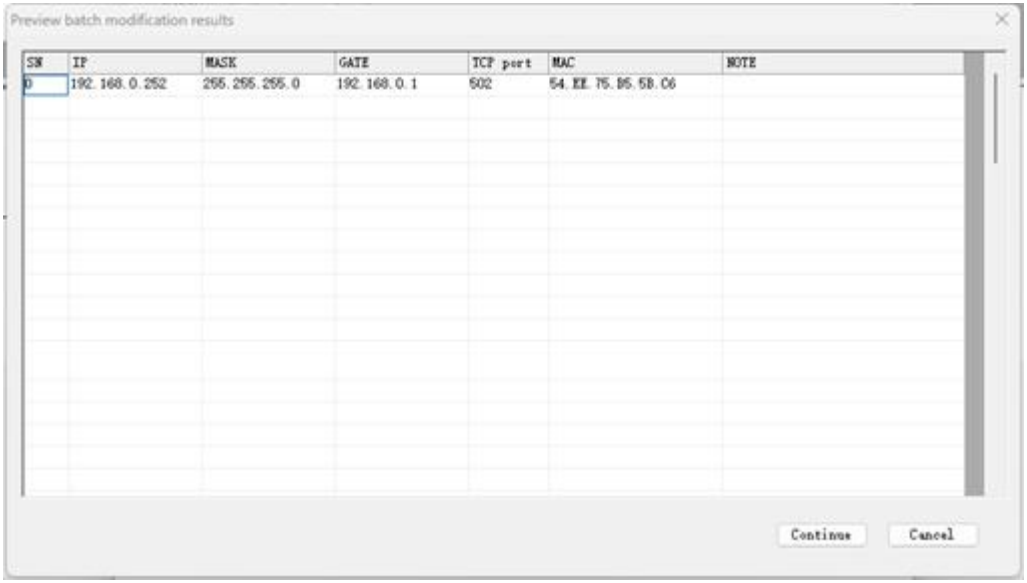


Figure 5

⑤ The parameters after successful modification are as shown in the figure below.

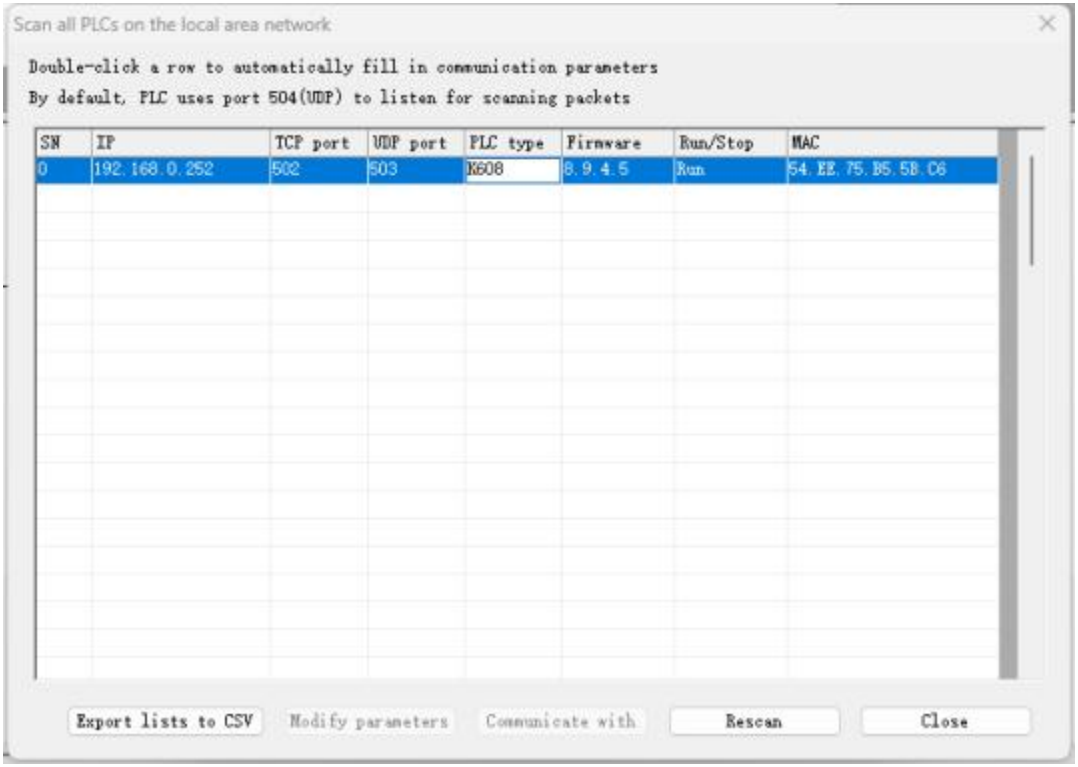


Figure 6

Note:

a) The sorting rule of the serial numbers in the scanning network is that the IP addresses are sorted from small to large, and when the IP addresses are the same, the MAC addresses are sorted from small to large.

Figure 6 shows sorted MAC address from small to large by same IP address

Figure 7 shows sorted IP address from small to large by different IP address

When some IP addresses are the same, all devices are sorted by IP address from small to large, and devices with the same IP address are sorted by MAC address from small to large.

b) When using batch modification of IP and port, the starting IP address and port number need to

be guaranteed to be a reasonable value, otherwise the address and port number may eventually exceed the maximum allowable value after incrementing. In this case, the modification will not be successful and The software will report an error message.

c) When the corresponding relationship between the IP address and the actual CPU in the network is unclear, you can connect a PLC to confirm it, or confirm the IP address by observing the running state of the CPU in the network by placing a PLC in a stopped state.

d) KincoBuilder occupies UDP port number 504 by default when scanning the network, and the communication connection defaults to port number 503, so when using UDP instructions, you must avoid these two port numbers to avoid unexpected errors.

e) When using TCP instructions in the program, be sure to avoid the port number that KincoBuilder communicates with the CPU through TCP to avoid unexpected errors.

10.3.3 Ethernet instructions

10.3.3.1Ehernet instructions description

Name	Description	Max No.
TCP_MBUSW	ModbusTCP master read instruction	32
TCP_MBUSR	ModbusTCP master write instruction	32
TCP_SERVER	Turn on TCP local server	16
TCPS_CLOSE	Turn off TCP local server	16
TCPS_XMT	Send instruction as a TCP server	32
TCPS_RCV	Receive instructions when acting as a TCP server	32
TCPS_XMTRCV	Send and receive instructions as a TCP server	32
TCP_CLIENT	Enable TCP local client	16
TCPC_CLOSE	Close the TCP local client	16
TCPC_XMT	Send instruction as a TCP client	32
TCPC_RCV	Receive instructions as a TCP client	32
TCPC_XMTRCV	Send and receive instructions as a TCP client	32
UDP_PEER	Create UDP_PEER	16
UDP _XMT	Send instruction when used as UDP_PEER	32
UDP _RCV	Receive instruction when used as UDP_PEER	32

UDP _XMTRCV	Send and receive instructions when used as UDP_PEER	32
ETH_STATUS	Instructions to monitor the current Ethernet status	16
ETH_RPARAS	Read the Ethernet parameters of the local CPU (IP, subnet mask, gateway, port number)	16
ETH_WPARAS	Modify the Ethernet parameters of the local CPU (IP, subnet mask, gateway, port number)	16
ETH_RESET	reset ethernet function	16

1) Notes:

When using these instructions, users need to pay attention to the following points:

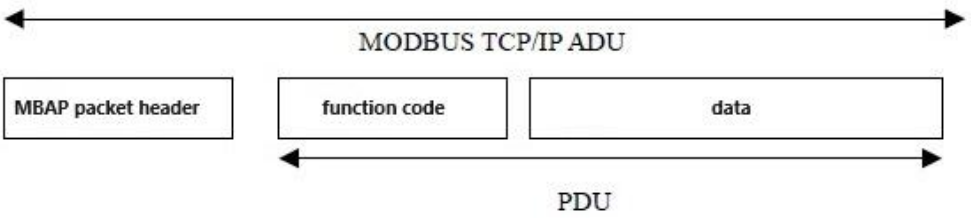
- • K series CPU provides 16 TCP server resources, 16 TCP client resources, and 16 UDP resources. In actual use, the corresponding maximum number of resources cannot be exceeded. It should be noted that when the CPU acts as a ModbusTCP slave station, it will occupy TCP Server resources, when the CPU acts as the ModbusTCP master station, will occupy the TCP client resources.
- When the TCP instruction uses the local port number, it is necessary to avoid the port number used for the communication between the programming software and the CPU, so as not to affect the communication.
- When using the local port number for the UDP instruction, it is necessary to avoid the port numbers (503 and 504) used for the communication between the programming software and the CPU, so as not to affect the communication.
- All instructions in Ethernet have a limit on the number of instructions. Please refer to the table above for the specific limit. When the number of instructions used in the program exceeds the upper limit, the PLC will report an error. You can see the error code in [PLC Information]→[Common Error]
- TCP_MBUSW, TCP_MBUSR, these two instructions will compete with the TCPC_xxx series of instructions for PLC client resources, try not to use them at the same time.
- TCP_MBUSW, TCP_MBUSR, these two instructions will automatically configure which PLC client is occupied at startup. When multiple instructions are running, they will be automatically connected in groups according to the two instructions R_IP and R_PORT input by the user.
- TCP_MBUSW, TCP_MBUSR, these two instructions are independent of all other Ethernet instructions, and each instruction can be used independently to realize its function.

- When the xxx_RCV and xxx _XMTRCV instructions are used at the same time, only the first instruction in the program can receive the message.

10.3.3.2 ModbusTCP instructions

ModbusTCP is a Modbus protocol based on Ethernet TCP/IP. The network port of the physical layer is different from the physical layer of ModbusRTU. It adopts master/slave mode of communication. Currently, Kinco-K series PLC supports both ModbusTCP master station and ModbusTCP slave station. By default, for ModbusTCP slave.

A simple explanation of the content of the Modbus TCP/IP protocol is to remove the modbus protocol itself, add the CRC check of the MBAP header, and add the MBAP header. The request/response of MODBUS over TCP/IP is shown in the following figure:



MBAP is the message head, the length is 7 bits. It consists of :

Transaction ID	Protocol ID	length	unit identifier
2 bits	2 bits	2 bits	1 bit

Content	Description
Transaction ID	It can be understood as the serial number of the message. Generally, 1 is added after each communication to distinguish different communication data messages.
Protocol ID	00 00 represents the ModbusTCP protocol.
length	Indicates the next data length, in bytes.
unit identifier	It can be understood as the device address.

The frame structure PDU consists of function code + data. It is consistent with the function code and

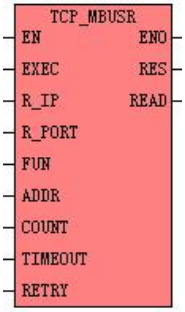
data field of serial link Modbus, you can refer to the introduction of serial Modbus function.

Unit identifier: the message sent by K series PLC as the ModbusTCP master station, the unit identifier is always 1, that is, the slave station address is ignored. When the PLC acts as a ModbusTCP slave station, the unit identifier is the same as that sent by the master station.

Note: The number of PLC memory areas and registers that can be accessed by the Ethernet port Modbus TCP master station is the same as that of the serial port Modbus RTU, which will not be repeated here. For details, please refer to 10.2.2.2 the memory of K series PLC area that Modbus RTU master station can access and 10.2.2.3 Modbus register numbers.

10.3.3.2.1 TCP_MBUSR instruction

- Instruction and its operand description

	Name	Instruction Format	Affected CR value	
LD	TCP_MBUSR			☑ K6
IL	TCP_MBUSR	TCP_MBUSR *EXEC, R_IP, R_PORT, FUN, ADDR, COUNT,TIMEOUT, RETRY, RES, READ*	U	

Parameters	Input/output	Data type	memory area allowed
EXEC	input	BOOL	I、Q、V、M、L、SM、RS、SR
R_IP	input	DWORD	M、V、L、SM、constant
R_PORT	input	WORD	M、V、L、SM、constant
FUN	input	INT	constant (Modbus Function code)

ADDR	input	WORD	M、V、L、SM、constant
COUNT	input	INT	M、V、L、SM、constant
TIMEOUT	input	INT	M、V、L、SM、constant
RETRY	input	INT	M、V、L、SM、constant
RES	output	BYTE	Q、M、V、L、SM
READ	output	BOOL、WORD、INT	Q、M、V、L、SM

Note: Parameters *R_IP, R_PORT, ADDR, COUNT, TIMEOUT, RETRY* must be both constant types or both memory types.

Attention, The READ parameter is a variable-length block memory parameter, and the entire block memory cannot fall into the illegal memory area, otherwise the result will be unpredictable.

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
R_IP	For the ip address of the remote server, you can directly enter the IP such as 192.168.210.100, or you can enter the hexadecimal DW#16#C0A8D264, that is, 192==C0 168==A8 210==D2 100==64.
R_PORT	The port of the remote server. 0 is not supported.
FUN	Modbus Function code。
ADDR	The first register address of the modbusTCP slave to access.
COUNT	How many consecutive registers to access, when the FUN parameter==1 or 2, the COUNT range is 1-1600, and other inputs trigger errors; when the FUN parameter==3 or 4, the COUNT range is 1-125, and other inputs trigger errors.
TIMEOUT	The timeout time of this execution, in ms, cannot be 0ms, cannot be greater than 32767ms, and cannot be a negative number. For example, if you enter 5000, it means that within 5 seconds, the following sequence of actions will be performed: a. Keep waiting for the connection to the server to succeed, and send the data to be sent immediately after the success. If the connection is not successful, return failure and end. b. After sending successfully, wait for the initial received data, if not, return failure and end. c. After the transmission is successful, the data is received, and the instruction ends successfully.
RETRY	refers to the number of retries. The range is 0-255. If the input exceeds this range, the value of the low byte of this parameter will be automatically intercepted. Indicates the number of times to retry after sending failed. For example, if the number of retries is 1, if each transmission fails, it will be sent twice after the transmission is started.
READ	When the FUN parameter is 1, 2, it must be a BOOL variable, which is the first bit of the received bit string, and the other bits received are placed in order When the FUN parameter is 3, 4, it must be a WORD or INT variable, which is the first bit of the received data, and the other bits received are placed in order.

RES	When execution starts, this parameter outputs 0, and when execution ends, bit7 == TRUE. The byte value of bit0-bit6 combined, indicating the error code 0. The instruction was executed successfully. 1. Unable to automatically allocate client resources 2. wrong length 3. Operation timed out and no receipt received 4. Unable to connect to the server 5. TIMEOUT parameter error 6. IP or PORT parameter error 7. Function code error 8. There is an error in the receipt received 9. There is too much TCP activity trying to connect (it is easy to appear when the Modbus TCP slave quickly accesses a non-existent slave or a disconnected slave, and it will disappear automatically when you slow down)

When the K series PLC is used as the ModbusTCP master, the TCP_MBUSR instruction is used to read the data in the slave. The applicable function codes for this instruction are 1 (read DO), 2 (read DI), 3 (read AO) and 4 (read AI).

The parameters R_IP and R_PORT represent the IP address and port number of the target slave station, and the port number range is 1~65535. FUN defines function codes. ADDR defines the starting address of the register to be read. COUNT defines the number of registers to be read. The maximum allowed value is 125 for read AI, 125 for read AO, 1600 for read DI, and 1600 for read DO.

A rising edge transition on the *EXEC* input is used to initiate communication. When the TCP_MBUSR instruction is executed, if the rising edge transition of EXEC is detected, TCP_MBUSR will conduct a communication: organize the message to send out according to the parameters input by the user and wait for the response from the slave station; after receiving the message returned by the slave station , the verification proves whether the message is correct, and if it is correct, the required data is written into the data buffer, otherwise the received message will be discarded.。

- LD

  If *EN* is 1, Then the instruction is executed, otherwise it is not executed.

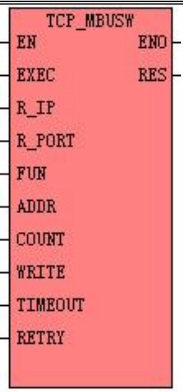
- IL

  If CR value is 1, Then the instruction is executed, otherwise it is not executed.

The execution of this instruction does not affect the CR value.

10.3.3.2.2 TCP_MBUSW instruction

- Instruction and its operand description

	Name	Instruction Format	Affect CR value	
LD	TCP_MBUSW			☑ K6
IL	TCP_MBUSW	TCP_MBUSW *EXEC, R_IP, R_PORT, FUN, ADDR, COUNT,WRITE,TIMEOUT, RETRY, RES*	U	

Parameter	Input/output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
R_IP	Input	DWORD	M、V、L、SM、Constant
R_PORT	Input	WORD	M、V、L、SM、Constant
FUN	Input	INT	Constant (Modbus Function code)
ADDR	Input	WORD	M、V、L、SM、Constant
COUNT	Input	INT	M、V、L、SM、Constant
WRITE	Input	BOOL、WORD、INT	Q、M、V、L、SM
TIMEOUT	Input	INT	M、V、L、SM、Constant
RETRY	Input	INT	M、V、L、SM、Constant
RES	Output	BYTE	Q、M、V、L、SM

Note: Parameters R_IP, R_PORT, ADDR, COUNT, TIMEOUT, RETRY must be both constant types or both memory types。

Attention, the WRITE parameter is a variable-length block memory parameter, and the entire block memory cannot fall into the illegal memory area, otherwise the result will be unpredictable.

Parameter	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
R_IP	For the ip address of the remote server, you can directly enter the IP such as 192.168.210.100, or you can enter the hexadecimal DW#16#C0A8D264, that is, 192==C0 168==A8 210==D2 100==64.
R_PORT	The port of the remote server. 0 is not supported. Modbus function code.
FUN	Modbus Function code.
ADDR	The first register address of the modbusTCP slave to access.
COUNT	Indicates how many consecutive registers to access. When FUN parameter == 6, COUNT can only be equal to 1. When FUN parameter == 16, COUNT range is 1-100, and other inputs trigger errors; when FUN parameter == 5, COUNT can only be equal to 1 Can be equal to 1. When the FUN parameter == 15, the COUNT range is 1-800, and other inputs will trigger an error.
WRITE	When the FUN parameter is 6, 16, it must be a WORD or INT variable, which is the first bit of the data to be sent, and the other bits of the data to be sent are placed in sequence; when the FUN parameter is 5, 15, it must be a BOOL variable, which is the variable to be sent. Send the first bit of the data, and place the other bits of the data to be sent in turn
TIMEOUT	The timeout time of this execution, in ms, cannot be 0ms, cannot be greater than 32767ms, and cannot be a negative number. For example, if you enter 5000, it means that within 5 seconds, the following sequence of actions will be performed: a. Keep waiting for the connection to the server to succeed, and send the data to be sent immediately after the success. If the connection is not successful, return failure and end. b. After sending successfully, wait for the initial received data, if not, return failure and end. c. After the transmission is successful, the data is received, and the instruction ends successfully.
RETRY	refers to the number of retries. The range is 0-255. If the input exceeds this range, the value of the low byte of this parameter will be automatically intercepted. Indicates the number of times to retry after sending failed. For example, if the number of retries is 1, if each transmission fails, it will be sent twice after the transmission is started.
RES	When execution starts, this parameter outputs 0, and when execution ends, bit7 == TRUE. The byte value of bit0-bit6 combined, indicating the error code 0. The instruction was executed successfully. 1. Unable to automatically allocate client resources 2. wrong length 3. Operation timed out and no receipt received 4. Unable to connect to the server 5. TIMEOUT parameter error 6. IP or PORT parameter error

	7. Function code error 8. There is an error in the receipt received 9. There is too much TCP activity trying to connect (it is easy to appear when the Modbus TCP slave quickly accesses a non-existent slave or a disconnected slave, and it will disappear automatically when you slow down)

When the K series PLC is used as the ModbusTCP master station, the TCP_MBUSW instruction is used to write data into the slave station. The applicable function codes for this instruction are 5 (write one DO), 6 (write one AO), 15 (write multiple DOs) and 16 (write multiple AOs).

The parameters R_IP and R_PORT represent the IP address and port number of the target slave station, and the port number range is 1~65535. FUN defines function codes. ADDR defines the starting address of the register to be written. COUNT defines the number of registers to be written. The maximum allowed value is 100 for writing AO and 800 for writing DO.

The parameter WRITE defines the starting address of the data buffer, and the data to be written into the slave is stored in this area. WRITE must match the function code. If the function code is 5, 15, input the address variable of BOOL type; if the function code is 6, 16, input the address variable of INT or WORD type.

A rising edge transition on the *EXEC* input is used to initiate communication. When the TCP_MBUSW instruction is executed, if the rising edge transition of EXEC is detected, MBUSW will conduct a communication: organize the message to send out according to the parameters input by the user and wait for the response from the slave station; after receiving the message returned by the slave station , to determine whether the slave station has correctly executed the write instruction just now.。

- LD

  IF *EN* is 1, Then the instruction is executed, otherwise it is not executed.。

- IL

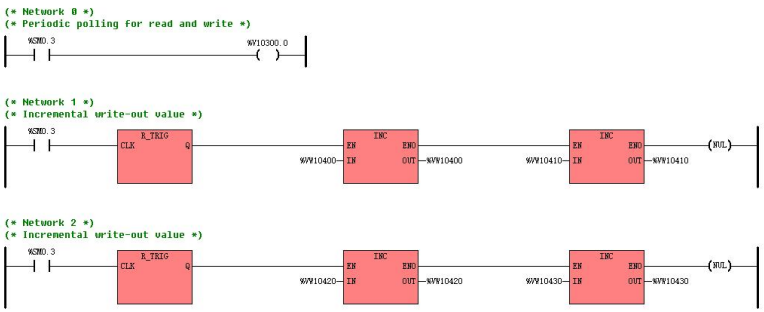
  If CR Value is 1, Then the instruction is executed, otherwise it is not executed.。

  The execution of this instruction does not affect the CR Value.

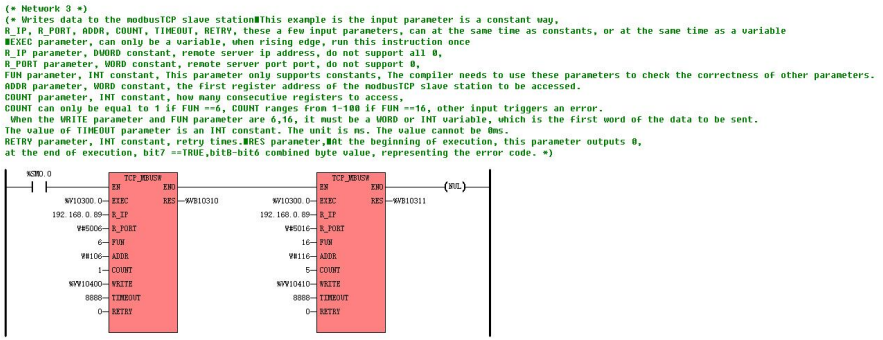
10.3.3.2.3 Example

This example monitors ModbusTCP messages through the network debugging assistant.

LD

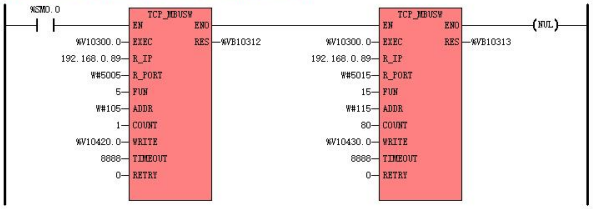


LD

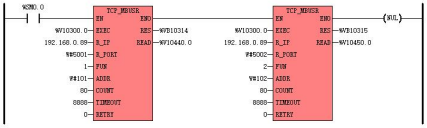


LD

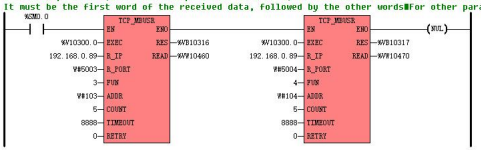
```
(* Network 4 *)
(* Writes data to the modbusTCP slave station
■COUNT parameter,INT constant, how many consecutive registers to access,
COUNT can only be equal to 1 if FUN ==5, cOUNT ranges from 1-800 if FUN ==15, other input triggers an error.
■If the WRITE parameter and FUN parameter are 5,15, the value must be BOOL variable,
which is the first bit of the data to be sent, and the other bits of the data to be sent■For other parameters,
see the description of FUN==6,FUN==16 above *)
```

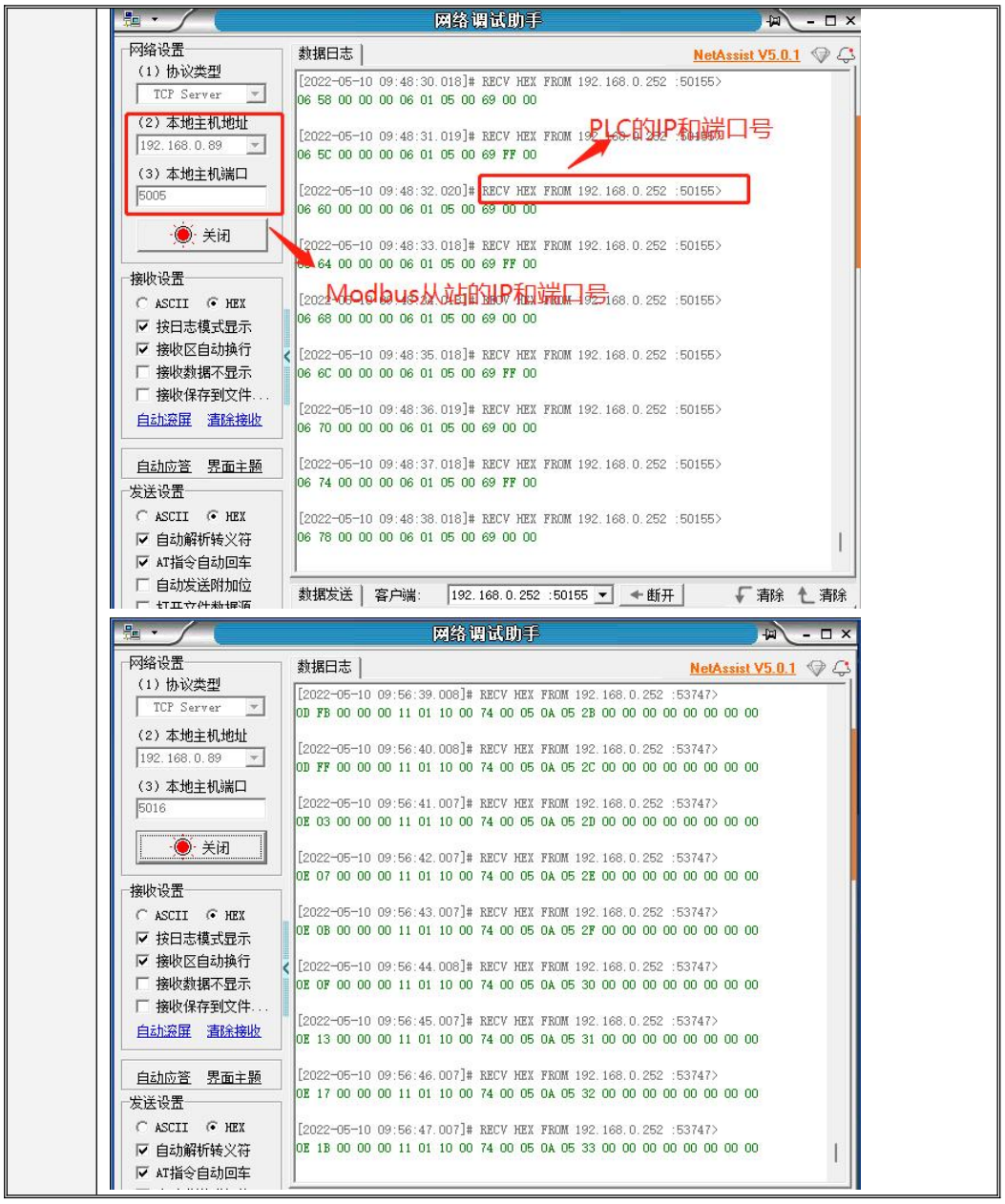



```
(* Network 5 *)
(* Writes data to the modbusTCP slave station
■This example is the input parameter is a constant way, R_IP, R_PORT, ADDR, COUNT, TIMEOUT, RETRY, these a few input parameters, can at the same time as constants,
or at the same time as a variable
■R_IP parameter DWORD constant, remote server ip address, do not support all 0.
R_PORT parameter, WORD constant, remote server port port, do not support 0,
FUN parameter, INT constant The number of constants support constant, The compiler needs to make some parameters check the correctness of others.
ADDR parameter WORD constant, the first register address of the modbusTCP slave station to be accessed.
COUNT parameter, INT constant, how many consecutive registers to access if FUN ==6, COUNT can only be equal to 1, if FUN ==16, COUNT ranges from 1-100, other input triggers an error.
When the WRITE parameter and FUN parameter are 6,16, it must be a WORD or INT variable, which is the first word of the data to be sent.
 The value of TIMEOUT parameter is an INT constant. The unit is ms. The value cannot be 0ms.
RETRY parameter, INT constant, retry times.
■RES parameter,■At the beginning of execution, this parameter outputs 0, at the end of execution, bit7 --TRUE,bit0-bit6 combined byte value, representing the error code. *)
```




```
(* Network 6 *)
(* Read data from modbusTCP slave station
■The COUNT argument,■INT constant, how many consecutive registers to access, COUNT range 1-125 when FUN parameter ==3 or 4, other input triggers an error
■The READ parameter,■When the FUN parameter is 3 or 4, it must be a WORD or INT variable.
 it must be the first word of the received data, followed by the other words■For other parameters, see the description of FUN==1,FUN==2 above *)
```




网络调试助手
网络设置
(1) 协议类型
TCP Server
(2) 本地主机地址
192.168.0.89
(3) 本地主机端口
5005
关闭
接收设置
ASCII HEX
按日志模式显示
接收区自动换行
接收数据不显示
接收保存到文件...
自动滚屏 清除接收
自动应答 界面主题
发送设置
ASCII HEX
自动解析转义符
AT指令自动回车
自动发送附加位
数据日志
NetAssist V5.0.1
[2022-05-10 09:48:30.018]# RECV HEX FROM 192.168.0.252 :50155>
06 58 00 00 00 06 01 05 00 69 00 00
[2022-05-10 09:48:31.019]# RECV HEX FROM 192.168.0.252 :50155>
06 5C 00 00 00 06 01 05 00 69 FF 00
[2022-05-10 09:48:32.020]# RECV HEX FROM 192.168.0.252 :50155>
06 60 00 00 00 06 01 05 00 69 00 00
[2022-05-10 09:48:33.018]# RECV HEX FROM 192.168.0.252 :50155>
06 64 00 00 00 06 01 05 00 69 FF 00
[2022-05-10 09:48:34.018]# RECV HEX FROM 192.168.0.252 :50155>
06 68 00 00 00 06 01 05 00 69 00 00
[2022-05-10 09:48:35.018]# RECV HEX FROM 192.168.0.252 :50155>
06 6C 00 00 00 06 01 05 00 69 FF 00
[2022-05-10 09:48:36.019]# RECV HEX FROM 192.168.0.252 :50155>
06 70 00 00 00 06 01 05 00 69 00 00
[2022-05-10 09:48:37.018]# RECV HEX FROM 192.168.0.252 :50155>
06 74 00 00 00 06 01 05 00 69 FF 00
[2022-05-10 09:48:38.018]# RECV HEX FROM 192.168.0.252 :50155>
06 78 00 00 00 06 01 05 00 69 00 00
数据发送 客户端: 192.168.0.252 :50155 断开 清除 清除
PLC的IP和端口号
Modbus从站的IP和端口号
网络调试助手
网络设置
(1) 协议类型
TCP Server
(2) 本地主机地址
192.168.0.89
(3) 本地主机端口
5016
关闭
接收设置
ASCII HEX
按日志模式显示
接收区自动换行
接收数据不显示
接收保存到文件...
自动滚屏 清除接收
自动应答 界面主题
发送设置
ASCII HEX
自动解析转义符
AT指令自动回车
数据日志
NetAssist V5.0.1
[2022-05-10 09:56:39.008]# RECV HEX FROM 192.168.0.252 :53747>
0D FB 00 00 00 11 01 10 00 74 00 05 0A 05 2B 00 00 00 00 00 00 00 00
[2022-05-10 09:56:40.008]# RECV HEX FROM 192.168.0.252 :53747>
0D FF 00 00 00 11 01 10 00 74 00 05 0A 05 2C 00 00 00 00 00 00 00 00
[2022-05-10 09:56:41.007]# RECV HEX FROM 192.168.0.252 :53747>
0E 03 00 00 00 11 01 10 00 74 00 05 0A 05 2D 00 00 00 00 00 00 00 00
[2022-05-10 09:56:42.007]# RECV HEX FROM 192.168.0.252 :53747>
0E 07 00 00 00 11 01 10 00 74 00 05 0A 05 2E 00 00 00 00 00 00 00 00
[2022-05-10 09:56:43.007]# RECV HEX FROM 192.168.0.252 :53747>
0E 0B 00 00 00 11 01 10 00 74 00 05 0A 05 2F 00 00 00 00 00 00 00 00
[2022-05-10 09:56:44.008]# RECV HEX FROM 192.168.0.252 :53747>
0E 0F 00 00 00 11 01 10 00 74 00 05 0A 05 30 00 00 00 00 00 00 00 00
[2022-05-10 09:56:45.007]# RECV HEX FROM 192.168.0.252 :53747>
0E 13 00 00 00 11 01 10 00 74 00 05 0A 05 31 00 00 00 00 00 00 00 00
[2022-05-10 09:56:46.007]# RECV HEX FROM 192.168.0.252 :53747>
0E 17 00 00 00 11 01 10 00 74 00 05 0A 05 32 00 00 00 00 00 00 00 00
[2022-05-10 09:56:47.007]# RECV HEX FROM 192.168.0.252 :53747>
0E 1B 00 00 00 11 01 10 00 74 00 05 0A 05 33 00 00 00 00 00 00 00 00

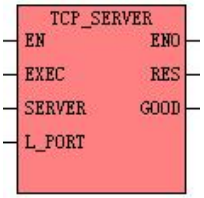
IL	(* Network 0 *) (*Periodic polling for read and write *) LD %SM0.3 ST %V10300.0 (* Network 1 *) (*Incremental write-out value *) LD %SM0.3 R_TRIG INC %VW10400 INC %VW10410 (* Network 2 *) (*Incremental write-out value *) LD %SM0.3 R_TRIG INC %VW10420 INC %VW10430 (* Network 3 *) LD %SM0.0 TCP_MBUSW %V10300.0, 192.168.0.89, W#5006, 6, W#106, 1, %VW10400, 8888, 0, %VB10310 TCP_MBUSW %V10300.0, 192.168.0.89, W#5016, 16, W#116, 5, %VW10410, 8888, 0, %VB10311 (* Network 4 *) LD %SM0.0 TCP_MBUSW %V10300.0, 192.168.0.89, W#5005, 5, W#105, 1, %V10420.0, 8888, 0, %VB10312 TCP_MBUSW %V10300.0, 192.168.0.89, W#5015, 15, W#115, 80, %V10430.0, 8888, 0, %VB10313
IL	(* Network 5 *) LD %SM0.0 TCP_MBUSR %V10300.0, 192.168.0.89, W#5001, 1, W#101, 80, 8888, 0, %VB10314, %V10440.0 TCP_MBUSR %V10300.0, 192.168.0.89, W#5002, 2, W#102, 80, 8888, 0, %VB10315, %V10450.0 (* Network 6 *) LD %SM0.0 TCP_MBUSR %V10300.0, 192.168.0.89, W#5003, 3, W#103, 5, 8888, 0, %VB10316, %VW10460 TCP_MBUSR %V10300.0, 192.168.0.89, W#5004, 4, W#104, 5, 8888, 0, %VB10317, %VW10470

10.3.3.3 TCP Server Instruction

10.3.3.3.1 TCP_SERVER Instruction

When you need to use this machine as a server, first use this instruction to open the server connection.

➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	TCP_SERVER			☑ K6
IL	TCP_SERVER	TCP_SERVER *EXEC,SERVER,L_PORT,RES,GOOD*	U	

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
SERVER	Input	INT	M、V、L、Constant
L_PORT	Input	WORD	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM
GOOD	Output	BOOL	Q、M、V、L、SM

Note: Parameter *SERVER, L_PORT* must be both constant types or both memory types.

➢ Parameter's description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute and the server connect.
SERVER	Server resource number, the range is 0--15, that is, a maximum of 16 TCP server resources are supported.
L_PORT	The local port of the server, 0 is not supported.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code. 0 Successful connected 1 SERVER parameter error 2 R_IP or R_PORT parameter error 3 Keep unused。 4 The client has been connected, but the client has been inactive for a long time, so it ends

GOOD	Every time EXEC is executed, the GOOD parameter is assigned to FALSE first. TRUE indicates that there is a remote client connected to this server, FALSE indicates that there is no. This parameter is updated once after the instruction is executed and will not be updated dynamically. You can view the change of dynamic connection through the ETH_STATUS instruction.

Note:

1、Each TCP_SERVER is only allowed to connect to one remote client and does not support multiple connections.

2、After the server is started, it will always be in the listen state and will not time out until the remote client connects to the server.

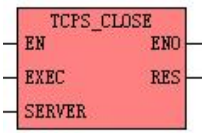
3、When the remote client connects to the server, and then the remote client disconnects, the server will automatically enter the listen state again, even if you do not execute the EXEC of this instruction again. At this time, after the remote client reconnects to this server, GOOD will not be updated, because this instruction has ended when the first connection is successful.

4、Because the PLC has only one network card, there is no need to input the local ip, and the local ip of the PLC is taken by default.

10.3.3.3.2 TCPS_CLOSE Instruction

This instruction is used to close the connection of this machine as a server.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	TCPS_CLOSE			☑ K6
IL	TCPS_CLOSE	TCPS_CLOSE	U	

		EXEC,SERVER,RES		

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
SERVER	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM

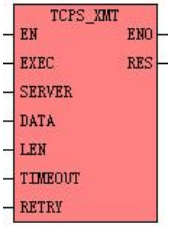
- Parameter's description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute, and the TCP server connection is closed.
SERVER	Server resource number, the range is 0--15, that is, a maximum of 16 TCP server resources are supported.
RES	When the instruction is executed, this parameter outputs 0. When the execution ends, bit7 becomes TRUE, and the combined value of bit0-bit6 represents the error code. 0 The shutdown server operation has been performed, and success is not guaranteed。 1 SERVER parameter error.

10.3.3.3.3 TCPS_XMT Instruction

This instruction is the sending instruction when the machine acts as a server.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	TCPS_XMT			☑ K6
IL	TCPS_XMT	TCPS_XMT *EXEC,SERVER,DATA, LEN, TIMEOUT, RETRY, RES*	U	

Parameters	Input/Output	Data type	Memory area allowed

EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
SERVER	Input	INT	M、V、L、Constant
DATA	Input	BYTE	M、V、L
LEN	Input	INT	M、V、L、Constant
TIMEOUT	Input	INT	M、V、L、Constant
RETRY	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM

Note: Parameters *SERVER, LEN, TIMEOUT, TRTRY* must be both constant types or both memory types.

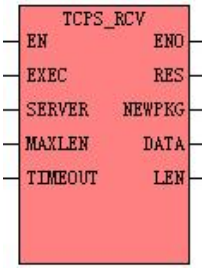
- Parameter's description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
SERVER	The number of the server resource that has been s tarted, in the range 0--15.
DATA	The first address of the data byte to be sent.
LEN	The number of bytes of data sent. The number of bytes cannot be 0, cannot be negative, and cannot be greater than 1460 bytes.
TIMEOUT	The timeout time of this execution, in ms, cannot be greater than 32767ms, and cannot be a negative number. 0 means not waiting, sending if there is a connection, and failing directly if there is no connection. When it is not 0, such as 5000, it means that it has been waiting for the connection to the client to succeed within 5 seconds, and it will be sent immediately after success. If there is no connection, it will return a timeout failure.
RETRY	refers to the number of retries. The range is 0-255. If the input exceeds this range, the value of the low byte of this parameter will be automatically intercepted. Indicates the number of times to retry after sending failed. For example, if the number of retries is 1, if each transmission fails, it will be sent twice after the transmission is started.
RES	At the beginning, this parameter outputs 0, at the end, bit7 becomes TRUE, and the combined value of bit0-bit6 indicates the error code. 0 Sending is completed, it only means that it has been submitted to the TCP protocol stack, it does not mean that the remote end has received it, and it does not mean that the remote end receive it. 1 SERVER Parameter error 2 send length error 3 The operation timed out and has not been sent yet 4 not connected to client yet 5 TIMEOUT Parameter, Input Error

10.3.3.3.4 TCPS_RCV Instruction

This instruction is the receiving instruction when the machine acts as a server。

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	TCPS_RCV			☑ K6
IL	TCPS_RCV	TCPS_RCV *EXEC, SERVER, MAXLEN, TIMEOUT, RES, NEWPKG, DATA, LEN*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
SERVER	Input	INT	M、V、L、Constant
MAXLEN	Input	INT	M、V、L、Constant
TIMEOUT	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM
NEWPKG	Output	BOOL	Q、M、V、L、SM
DATA	Output	BYTE	Q、M、V、L、SM
LEN	Output	WORD	Q、M、V、L、SM

Note: Parameter *SERVER, MAXLEN, TIMEOUT,* Must be both constant types or both memory types.

- Parameters description

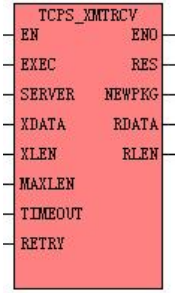
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
SERVER	The number of the server resource that has been opened, in the range 0--15.
MAXLEN	The maximum number of data bytes that can be received in a message. You cannot enter 0 or a negative number. When MAXLEN is greater than 1460, it will be ignored, and the

	maximum received 1460 bytes.
TIMEOUT	timeout period of this execution, in ms, cannot be greater than 32767ms, and cannot be a negative number. 0 means waiting forever, and is always in the state of waiting for client messages. If it is not 0, such as 5000, it means that it has been waiting for the message from the client within 5 seconds, and no longer receives the message after the timeout.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 Not used 1 SERVER Parameter error 2 MAXLEN Length error 3 Operation timed out, end receiving 5 TIMEOUT Parameter, Input error
NEWPKG	When the server receives a new message, NEWPKG generates a rising edge, which only lasts for 1 scan period.
DATA	The starting byte of the receiving buffer. Note that this buffer is matched with MAXLEN. For example, MAXLEN is 100 and DATA=VB0. At this time, VB0--VB99 are all buffers, which can only be read and cannot be used for other purposes. . Users can clear the buffer by themselves. Otherwise, when the length of the next received message is smaller than the last one, the buffer will have the data of the last message.
LEN	The length of the received message. When the actual message length is less than or equal to MAXLEN, LEN is the actual message length. When the actual message length exceeds MAXLEN, LEN=MAXLEN.

10.3.3.3.5 TCPS_XMTRCV Instruction

This instruction is the sending and receiving instruction when the machine acts as a server.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	TCPS_XMT RCV			☑ K6
IL	TCPS_XMT RCV	TCPS_RCV *EXEC, SERVER, XDATA, XLEN, MAXLEN, TIMEOUT, RETRY, RES, NEWPKG, RDATA, RLEN*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
SERVER	Input	INT	M、V、L、constant
XDATA	Input	BYTE	M、V、L
XLEN	Input	INT	M、V、L、constant
MAXLEN	Input	INT	M、V、L、
TIMEOUT	Input	INT	M、V、L、constant
RETRY	Input	INT	M、V、L、constant
RES	Output	BYTE	Q、M、V、L、SM
NEWPKG	Output	BOOL	Q、M、V、L、SM
RDATA	Output	BYTE	Q、M、V、L、SM
RLEN	Output	WORD	Q、M、V、L、SM

Note: parameters *SERVER, XLEN, MAXLEN, TIMEOUT, RETRY* Must be both constant types or both memory types.

- Parameter's description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
SERVER	The number of the server resource that has been opened, in the range 0--15.
XDATA	The first address of the data byte to be sent.
XLEN	The number of bytes of sent data, cannot be 0, cannot be negative, and cannot be greater than 1460 bytes.
MAXLEN	The maximum number of data bytes that can be received in a message. It cannot be entered as 0, and cannot be negative. The maximum number of bytes received is 1460 bytes. When MAXLEN is greater than 1460, it will be ignored.
TIMEOUT	The timeout time of this execution, in ms, cannot be 0ms, cannot be greater than 32767ms, and cannot be a negative number. For example, if you enter 5000, it means that within 5 seconds, the following sequence of actions will be performed: a. Always wait for the connection to the client to succeed, and send the data to be sent immediately after success. If the connection has not been successful, return failure and end. b. After sending successfully, wait for data to be received, if not, return failure and end. c. After the transmission is successful, the data is received, and the instruction ends successfully.
RETRY	The number of retries, if the range is 0-255, if the input exceeds this range, the value of the low byte of this parameter will be automatically intercepted.

	The number of retries after sending failure or receiving failure. For example, the number of retries is 1. If every sending fails, after starting EXEC, it will send twice.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 The instruction ends successfully, a message is sent, and a message is received 1 SERVER Parameter error 2 Length error, MAXLEN is 0, or XLEN is 0, or XLEN is greater than 1460 bytes. When MAXLEN is greater than 1460, it will be ignored, and only messages less than 1460 bytes will be accepted. 3 Timeout, no message received 4 Failed to send, has not been connected to the client 5 TIMEOUT parameter, input error
NEWPKG	This instruction EXEC will automatically become FALSE when it is started, and it will become TRUE when a message is successfully received until the next time this instruction is started.
RDATA	The starting byte of the receiving buffer. Note that this buffer is matched with MAXLEN. For example, MAXLEN is 100 and DATA=VB0. At this time, VB0--VB99 are all buffers, which can only be read and cannot be used for other purposes. Users can clear the buffer by themselves. Otherwise, when the length of the next received message is smaller than the last one, the buffer will have the data of the last message.
RLEN	The length of the received message. When the actual message length is less than or equal to MAXLEN, LEN is the actual message length. When the actual message length exceeds MAXLEN, LEN=MAXLEN.

Note: Only 1 message can be received when the rising edge of EXEC is started. Take the rising edge of NEWPKG to read the message.

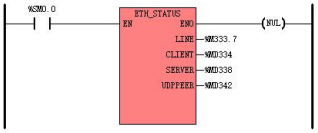
TCPS_RCV and TCPS_XMTRCV when the instructions are used at the same time, only the first instruction in the program can receive the message。

10.3.3.3.6 Example

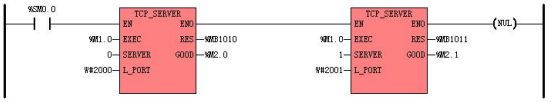
In this example, the network debugging assistant is used as a TCP client, and the PLC is used as a TCP server.

LD

```
(* Network 0 *)
(* LINE Parameter TRUE means network cable connected,   FALSE Indicates that the network cable is disconnected CLIENT parameter   DWORD,
 \each bit indicates a TCP client' s status SERVER Parameter  DWORD,  each bit indicates a TCP server' s status UDPPEER parameter  DWORD,
Each bit represents the status of a UDP peer *)
```



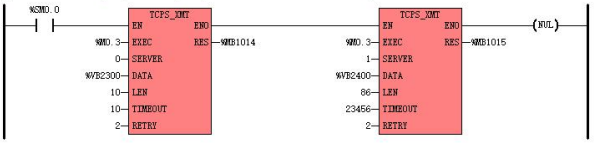

```
(* Network 1 *)
(* This example is the way that input parameter is constant,
SERVER, L_PORT, These input parameters could be constant at same time,
or start the TCP server for the variable at the same time *)
```



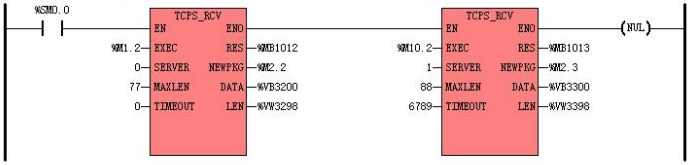
LD

```
(* Network 2 *)
(* Shut down the TCP server with the input parameters as constants *)
```

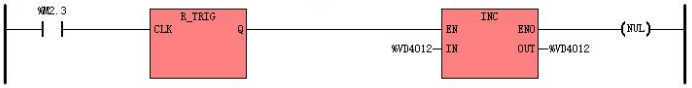
```
(* Network 3 *)
(* In this example, the input parameters are constant *)
```

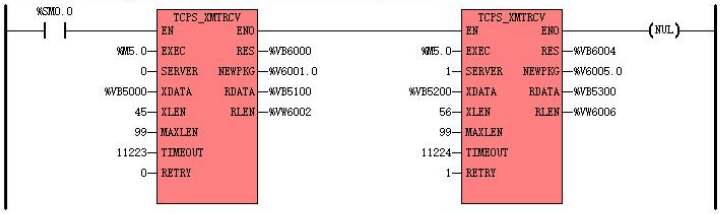



```
(* Network 4 *)
(* Sending data *)
```

<table>
<tr>
<td></td>
<td>
(* Network 5 *)
(* In this example, the input parameters are constant *)


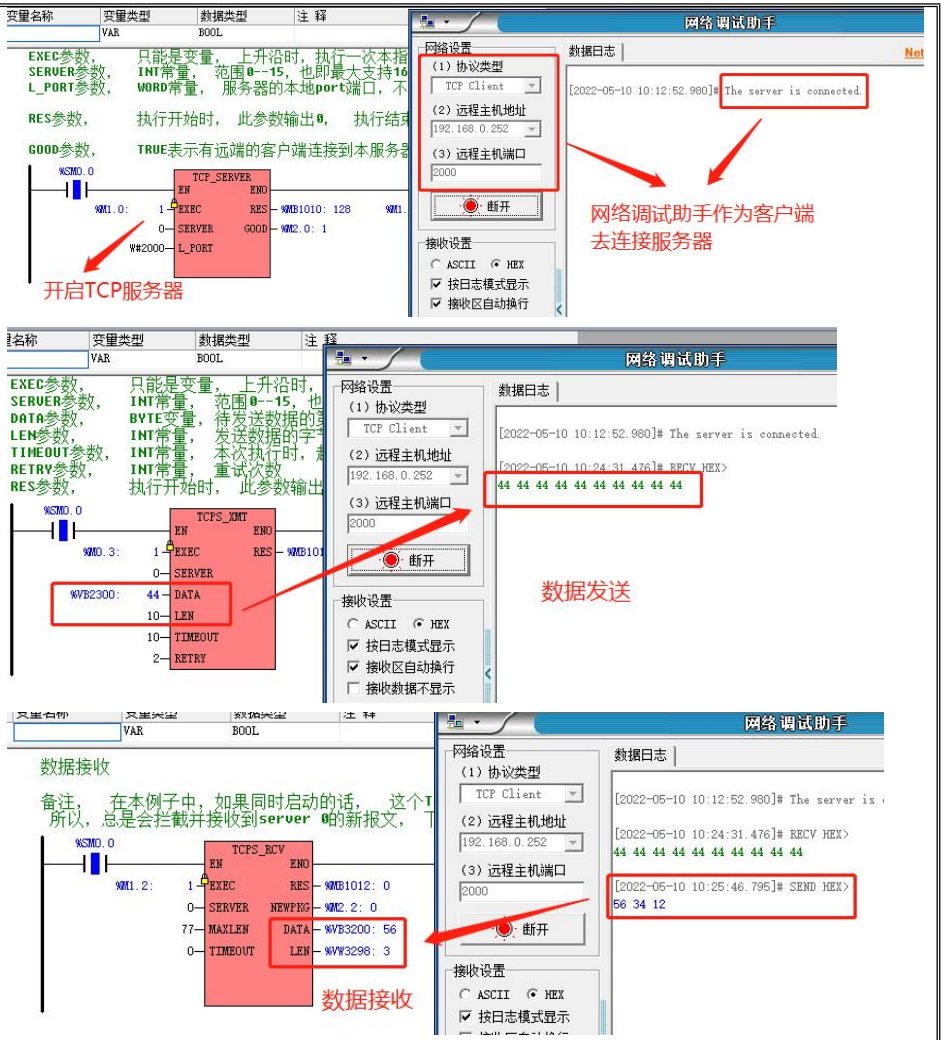
(* Network 6 *)
(* est statement Observe the number of packets received by the TCPS_RCV instruction from server 0 *)

(* Network 7 *)
(* Test statement Observe the number of packets received by the TCPS_RCV instruction from server 1 *)


</td>
</tr>
<tr>
<td>LD</td>
<td>
(* Network 8 *)
(* In this example, the input parameters are constant *)


(* Network 9 *)
(* Emitted string for TCPS_XMTRCV *)

</td>
</tr>
</table>



IL

```
(* Network 0 *)
(*LINE Parameter  TRUE means network cable connected,  FALSE Indicates that the network cable is disconnected CLIENT parameter  DWORD,  each bit indicates a TCP client's status SERVER Parameter  DWORD,  each bit indicates a TCP server's status UDPPEER parameter  DWORD,  Each bit represents the status of a UDP peer*)
LD    %SM0.0
ETH_STATUS %M333.7, %MD334, %MD338, %MD342
(* Network 1 *)
```

	(*This example is the way that input parameter is constant, SERVER, L_PORT, These input parameters could be constant at same time, or start the TCP server for the variable at the same time *) LD %SM0.0 TCP_SERVER %M1.0, 0, W#2000, %MB1010, %M2.0 TCP_SERVER %M1.0, 1, W#2001, %MB1011, %M2.1 (* Network 2 *) (*Shut down the TCP server with the input parameters as constants *) LD %SM0.0 TCPS_CLOSE %M1.4, 0, %MB1016 TCPS_CLOSE %M1.5, 1, %MB1017 (* Network 3 *) (*In this example, the input parameters are constant *) LD %SM0.0 TCPS_XMT %M0.3, 0, %VB2300, 10, 10, 2, %MB1014 TCPS_XMT %M0.3, 1, %VB2400, 86, 23456, 2, %MB1015 (* Network 4 *) (*Sending data*) LD %SM0.1 FILL B#16#44, %VB2300, B#73 FILL B#16#55, %VB2400, B#83
IL	(* Network 5 *) (*In this example, the input parameters are constant *) LD %SM0.0 TCPS_RCV %M1.2, 0, 77, 0, %MB1012, %M2.2, %VB3200, %VW3298 TCPS_RCV %M10.2, 1, 88, 6789, %MB1013, %M2.3, %VB3300, %VW3398 (* Network 6 *) (*est statement Observe the number of packets received by the TCPS_RCV instruction from server 0 *) LD %M2.2 R_TRIG INC %VD4008 (* Network 7 *) (*Test statement Observe the number of packets received by the TCPS_RCV instruction from server 1 *) LD %M2.3 R_TRIG INC %VD4012 (* Network 8 *) (*In this example, the input parameters are constant *) LD %SM0.0

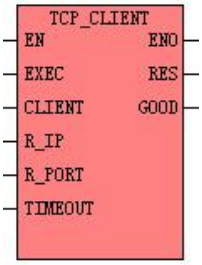
	TCPS_XMTRCV %M5.0, 0, %VB5000, 45, 99, 11223, 0, %VB6000, %V6001.0, %VB5100, %VW6002 TCPS_XMTRCV %M5.0, 1, %VB5200, 56, 99, 11224, 1, %VB6004, %V6005.0, %VB5300, %VW6006 (* Network 9 *) (*Emitted string for TCPS_XMTRCV *) LD %SM0.1 FILL B#16#33, %VB5000, B#43 FILL B#16#44, %VB5200, B#53
IL	(* Network 10 *) (*Observe the number of packets received by TCPS_XMTRCV from server 0 *) LD %V6001.0 R_TRIG INC %VD4016 (* Network 11 *) (*Observe the number of packets received by TCPS_XMTRCV from server 1 *) LD %V6005.0 R_TRIG INC %VD4020

10.3.3.4 TCP Client instructions

10.3.3.4.1 TCP_CLIENT Instruction

When you need to use this machine as a TCP client, first use this instruction to start the client connection.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	TCP_CLIENT			☑ K6
IL	TCP_ CLIENT	TCP_ CLIENT *EXEC,CLIENT,R_IP,R_PORT,TIMEOUT,RES,GOOD*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
CLIENT	Input	INT	M、V、L、Constant
R_IP	Input	DWORD	M、V、L、Constant
R_PORT	Input	WORD	M、V、L、Constant
TIMEOUT	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM
GOOD	Output	BOOL	Q、M、V、L、SM

Note: The parameters CLIENT, R_IP, R_PORT, TIMEOUT must be both constant types or memory types at the same time.

- Parameters description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute and the server connection is opened.

CLIENT	Client resource number, in the range 0--15, that is, a maximum of 16 TCP client resources are supported.
R_IP	The IP address of the remote server. All 0s are not supported. You can directly enter the IP address, for example: 192.168.210.100
R_PORT	The port of the remote server. 0 is not supported.
TIMEOUT	The timeout period, in ms, cannot be greater than 32767ms, and cannot be a negative number. 0 means that it has been waiting for the connection to succeed. When it is not 0, such as 5000, it means that the connection to the server is not successful within 5 seconds, and a timeout failure is returned.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 Successfully connected 1 CLIENT parameter error 2 R_IP 或 R_PORT parameter error 3 Timeout, and not connected successfully 4 Connected to the server, ended due to no operation 5 TIMEOUT Parameter error 6 Too much TCP activity trying to connect(prone when a Modbus TCP slave is quickly accessing a non-existent slave or a disconnected slave, slow down and it will disappear automatically)
GOOD	Each time EXEC is executed, the GOOD parameter is first assigned the value FALSE. TRUE indicates that the connection to the server was successful, and FALSE indicates that the connection failed. This parameter is updated once after the instruction is executed and will not be updated dynamically. You can view the change of dynamic connection through the ETH_STATUS instruction.

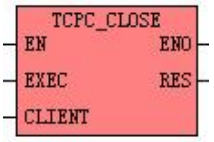
Note: When a remote server has been connected, and the IP or PORT is modified to point to the new remote server, executing this instruction again will automatically close the old connection, and then connect to the new remote server.

10.3.3.4.2 TCPC_CLOSE Instruction

This Instruction is used to close the connection of this machine as a client.。

- Instruction and its operand description

	Name	Instruction format	Affected CR value	☑ K6

LD	TCPC_CLOSE			
IL	TCPC_CLOSE	TCPC_CLOSE *EXEC,SERVER,RES*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
CLIENT	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM

➢ Parameter's description

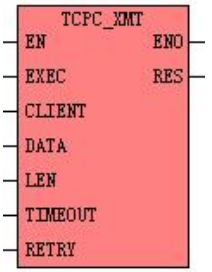
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute, and the TCP client connection is closed.
CLIENT	Client resource number, in the range 0--15, that is, a maximum of 16 TCP client resources are supported.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 A close client operation has been performed; success is not guaranteed. 1 CLIENT parameter error

10.3.3.4.3 TCPC_XMT Instruction

This instruction is the sending instruction when the machine acts as a client.

➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	☑ K6

LD	TCPC_XMT			
IL	TCPC_XMT	TCPC_XMT *EXEC,CLIENT,DATA, LEN, TIMEOUT, RES*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
CLIENT	Input	INT	M、V、L、Constant
DATA	Input	BYTE	M、V、L
LEN	Input	INT	M、V、L、Constant
TIMEOUT	Input	INT	M、V、L、Constant
RETRY	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM

Note: The parameters CLIENT, LEN, TIMEOUT, TRTRY must be constant types or memory types at the same time.

- Parameter description

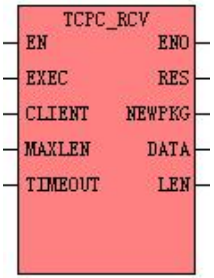
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
CLIENT	The number of the client resource that has been opened, in the range 0--15.
DATA	The first address of the data byte to be sent.
LEN	The number of bytes of sent data, cannot be 0, cannot be negative, and cannot be greater than 1460 bytes.
TIMEOUT	The timeout period, in ms, cannot be greater than 32767ms, and cannot be a negative number. 0 means that it has been waiting for the connection to succeed. When it is not 0, such as 5000, it means that the connection to the server is not successful within 5 seconds, and a timeout failure is returned.
RETRY	The number of retries, if the range is 0-255, if the input exceeds this range, the value of the low byte of this parameter will be automatically intercepted. The number of retries after sending failure or receiving failure. For example, the number of retries is 1. If every sending fails, after starting EXEC, it will send twice.

RES	At the beginning of execution, this parameter outputs 0. At the end of execution, bit7==TRUE, the byte value of bit0-bit6 combined, represents the error code 0 sending completed, only indicating submission to the TCP protocol stack, does not mean that the remote end has received it, nor does it necessarily mean that the remote end will receive it 1 CLIENT parameter error 2. Sending length error The operation has timed out and has not been sent yet 4 has not yet connected to the server 5 TIMEOUT parameter, input error

10.3.3.4.4 TCPC_RCV instruction

This instruction is the receiving instruction when the machine acts as a server.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	TCPC_RCV			☑ K6
IL	TCPC_RCV	TCPC_RCV *EXEC, CLIENT, MAXLEN, TIMEOUT, RES, NEWPKG, DATA, LEN*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
CLIENT	Input	INT	M、V、L、Constant
MAXLEN	Input	INT	M、V、L、Constant
TIMEOUT	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM
NEWPKG	Output	BOOL	Q、M、V、L、SM
DATA	Output	BYTE	Q、M、V、L、SM

LEN	Output	WORD	Q、M、V、L、SM

Note: The parameters CLENT, MAXLEN, TIMEOUT must be both constant types or memory types at the same time.

- Parameter description

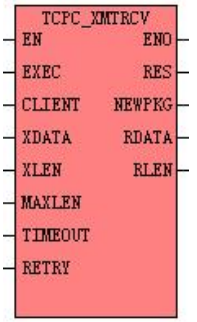
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
CLIENT	The number of the client resource that has been opened, in the range 0--15.
MAXLEN	The maximum number of data bytes that can be received in one message. It cannot be entered as 0 or as a negative number. The maximum received 1460 bytes, when MAXLEN is greater than 1460 will be ignored
TIMEOUT	The timeout period, in ms, cannot be greater than 32767ms, and cannot be a negative number. 0 means that it has been waiting for the connection to succeed. When it is not 0, such as 5000, it means that the connection to the server is not successful within 5 seconds, and a timeout failure is returned.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 not used 1 CLIENT parameter error 2 MAXLEN length error 3 Operation timed out, end receiving 5 TIMEOUT parameter, input error
NEWPKG	When the client receives a new message, NEWPKG generates a rising edge, which only lasts for one scan period.
DATA	The starting byte of the receiving buffer. Note that this buffer is matched with MAXLEN. For example, MAXLEN is 100 and DATA=VB0. At this time, VB0--VB99 are all buffers, which can only be read and cannot be used for other purposes. . Users can clear the buffer by themselves. Otherwise, when the length of the next received message is smaller than the last one, the buffer will have the data of the last message.
LEN	The length of the received packet. When the actual packet length is less than or equal to MAXLEN, LEN is the actual packet length. When the actual packet length exceeds MAXLEN, LEN=MAXLEN.

10.3.3.4.5 TCPC_XMTRCV instruction

This instruction is the sending and receiving instruction when the machine acts as a client.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	☑ K6

LD	TCPC_XMT RCV			
IL	TCPC_XMT RCV	TCPC_RCV *EXEC, CLIENT, XDATA, XLEN, MAXLEN, TIMEOUT, RETRY, RES, NEWPKG, RDATA, RLEN*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
CLIENT	Input	INT	M、V、L、Constant
XDATA	Input	BYTE	M、V、L
XLEN	Input	INT	M、V、L、Constant
MAXLEN	Input	INT	M、V、L、Constant
TIMEOUT	Input	INT	M、V、L、Constant
RETRY	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM
NEWPKG	Output	BOOL	Q、M、V、L、SM
RDATA	Output	BYTE	Q、M、V、L、SM
RLEN	Output	WORD	Q、M、V、L、SM

Note: The parameters CLIENT, XLEN, MAXLEN, TIMEOUT, RETRY must be constant type or memory type at the same time.

- Parameter's description

Parameters	Description
EXEC	If the rising edge instruction of EXEC is detected, the instruction is triggered to execute.
CLIENT	The number of the client resource that has been opened, in the range 0--15.
XDATA	The first address of the data byte to be sent.
XLEN	The number of bytes of sent data, cannot be 0, cannot be negative, and cannot be greater

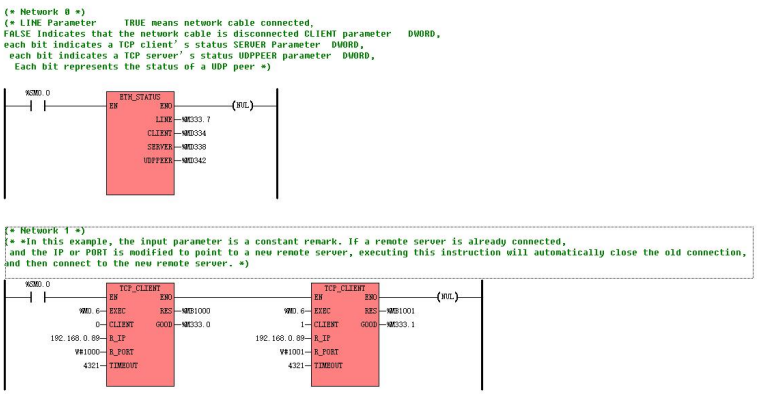
	than 1460 bytes.
MAXLEN	The maximum number of data bytes that can be received in a message. You cannot enter 0 or a negative number. When MAXLEN is greater than 1460, it will be ignored, and the maximum received 1460 bytes.
TIMEOUT	The timeout time of this execution, in ms, cannot be 0ms, cannot be greater than 32767ms, and cannot be a negative number. For example, if you enter 5000, it means that within 5 seconds, the following sequence of actions will be performed: a. It keeps waiting for the connection to the server to succeed, and immediately sends the data to be sent after the success. If the connection is not successful, it returns a failure message and ends. b. After the transmission is successful, wait for receiving data, and it has not waited until it returns failure and ends。 c. After the transmission is successful, data is received, and the instruction ends successfully.
RETRY	The number of retries, the range is 0-255. If the input exceeds this range, the value of the low byte of this parameter will be automatically intercepted. The number of retries after sending failure or receiving failure. For example, the number of retries is 1. Assuming that each sending fails, after starting EXEC, it will send a total of 2 times.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code. 0 The instruction ends successfully, a message is sent, and a message is received 1 CLIENT Parameter error 2 Length error, MAXLEN is 0, or XLEN is 0, or XLEN is greater than 1460 bytes, only the maximum length of 1460 is received. When MAXLEN is greater than 1460, it will be ignored 3 The operation timed out and no message was received 4 Failed to send, has not been connected to the server 5 TIMEOUT parameter, input error
NEWPKG	This instruction EXEC will automatically become FALSE when it is started, and it will become TRUE when a message is successfully received until the next time this instruction is started.
RDATA	The starting byte of the receiving buffer. Note that this buffer is matched with MAXLEN. For example, MAXLEN is 100 and DATA=VB0. At this time, VB0--VB99 are all buffers, which can only be read and cannot be used for other purposes. . Users can clear the buffer by themselves. Otherwise, when the length of the next received message is smaller than the last one, the buffer will have the data of the last message.
RLEN	The length of the received packet. When the actual packet length is less than or equal to MAXLEN, LEN is the actual packet length. When the actual packet length exceeds MAXLEN, LEN=MAXLEN.

Note: Only 1 message can be received after the rising edge of EXEC is started. Take the rising edge of NEWPKG to read the message.
When the TCPC_RCV and TCPC_XMTRCV instructions are used at the same time, only the first instruction in the program can receive the message.

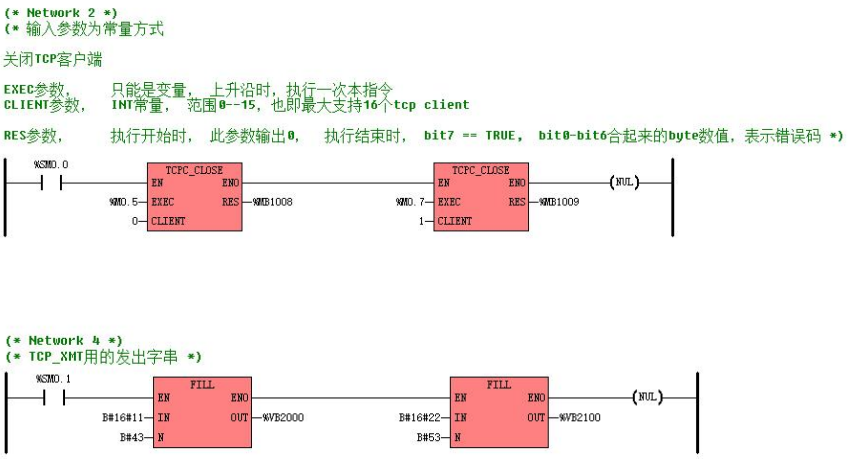
10.3.3.4.6 Example

In this example, the network debugging assistant is used as a TCP server, and the PLC is used as a TCP client.

LD



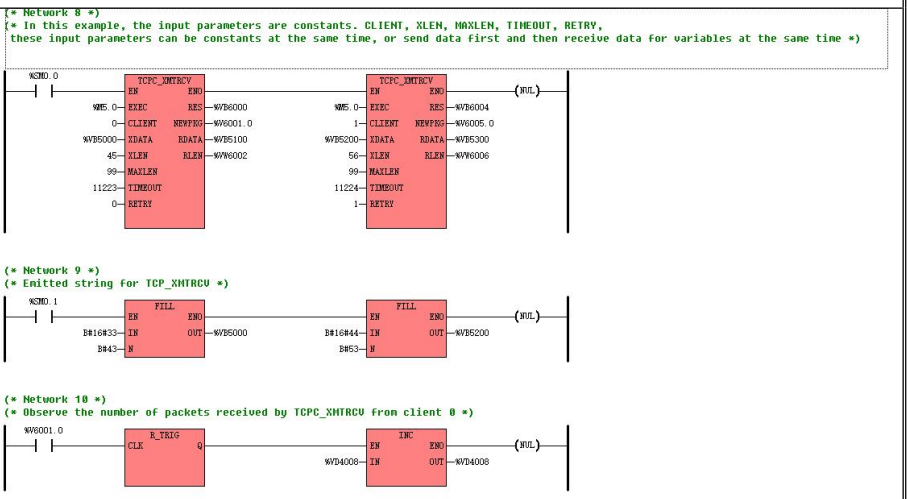
LD

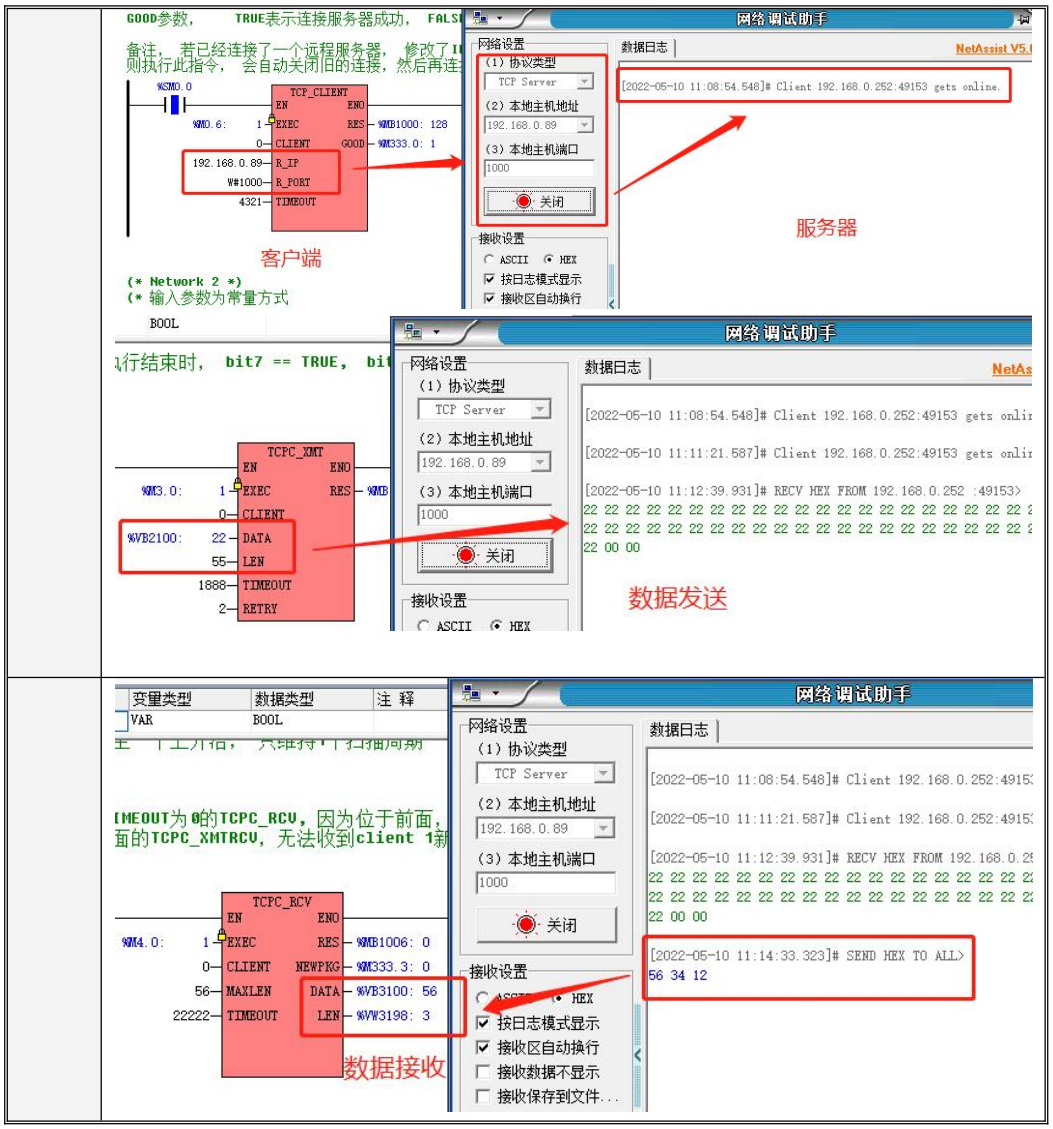


LD

(* Network 2 *)
(* The input parameter is a constant method to close the TCP client *)
%M0.0
TCPC_CLOSE
EN
ENO
%M0.5
EXEC
RES
%WB1008
0
CLIENT
%M0.7
EXEC
RES
%WB1009
1
CLIENT
(NUL)
(* Network 3 *)
(* In this example, the input parameters are constant, CLIENT, LEN, TIMEOUT, RETRY, these input parameters can be constants at the same time, or send data for variables at the same time *)
%M0.0
TCPC_EXT
EN
ENO
%M3.0
EXEC
RES
%WB1002
1
CLIENT
%WB2000
DATA
44
LEN
1111
TIMEOUT
1
RETRY
%M3.0
EXEC
RES
%WB1003
0
CLIENT
%WB2100
DATA
55
LEN
1888
TIMEOUT
2
RETRY
(NUL)
(* Network 4 *)
(* Outgoing string for TCP_XMT *)
%M0.1
FILL
EN
ENO
B#16#11
IN
OUT
%WB2000
B#43
N
B#16#22
IN
OUT
%WB2100
B#53
N
(NUL)
(* Network 5 *)
(* In this example, the input parameters are constants, CLIENT, MAXLEN, TIMEOUT, these input parameters can be constants at the same time, or receive data for variables at the same time *)
%M0.0
TCPC_RCV
EN
ENO
%M4.0
EXEC
RES
%WB1005
1
CLIENT
NEWFEG
%M33.2
49
MAXLEN
DATA
%WB3000
0
TIMEOUT
LEN
%WW3098
%M4.0
EXEC
RES
%WB1006
0
CLIENT
NEWFEG
%M33.3
56
MAXLEN
DATA
%WB3100
22222
TIMEOUT
LEN
%WW3198
(NUL)
(* Network 6 *)
(* Observe the TCPC_RCV instruction, the number of packets received from client 1 *)
%M33.2
R_TRIG
CLK
Q
INC
EN
ENO
%WD4000
IN
OUT
%WD4000
(NUL)
(* Network 7 *)
(* Observe the TCPC_RCV instruction, the number of packets received from client 0 *)
%M33.3
R_TRIG
CLK
Q
INC
EN
ENO
%WD4004
IN
OUT
%WD4004
(NUL)

LD


(* Network 8 *)
(* In this example, the input parameters are constants. CLIENT, XLEN, MAXLEN, TIMEOUT, RETRY,
these input parameters can be constants at the same time, or send data first and then receive data for variables at the same time *)
%M0.0
TCPC_XMTRCV
EN
ENO
%M5.0
EXEC
RES
%WB6000
0
CLIENT
NEWFEG
%WB6001.0
%WB5000
XDATA
RDATA
%WB5100
45
XLEN
RLEN
%WW6002
99
MAXLEN
11223
TIMEOUT
0
RETRY
TCPC_XMTRCV
EN
ENO
%M5.0
EXEC
RES
%WB6004
1
CLIENT
NEWFEG
%WB6005.0
%WB5200
XDATA
RDATA
%WB5300
56
XLEN
RLEN
%WW6006
99
MAXLEN
11224
TIMEOUT
1
RETRY
(NUL)
(* Network 9 *)
(* Emitted string for TCP_XMTRCV *)
%M0.1
FILL
EN
ENO
B#16#33
IN
OUT
%WB5000
B#43
N
FILL
EN
ENO
B#16#44
IN
OUT
%WB5200
B#53
N
(NUL)
(* Network 10 *)
(* Observe the number of packets received by TCPC_XMTRCV from client 0 *)
%WB6001.0
R_TRIG
CLK
Q
INC
EN
ENO
%WD4008
IN
OUT
%WD4008
(NUL)


GOOD参数， TRUE表示连接服务器成功， FALSE
备注， 若已经连接了一个远程服务器， 修改了IP
则执行此指令， 会自动关闭旧的连接，然后再连
%SM0.0
%M0.6:
TCP_CLIENT
EN
ENO
EXEC
RES
CLIENT
GOOD
192.168.0.89
R_IP
W#1000
R_PORT
4321
TIMEOUT
%MB1000: 128
%M333.0: 1
客户端
(* Network 2 *)
(* 输入参数为常量方式
BOOL
网络调试助手
网络设置
(1) 协议类型
TCP Server
(2) 本地主机地址
192.168.0.89
(3) 本地主机端口
1000
关闭
接收设置
ASCII
HEX
按日志模式显示
接收区自动换行
数据日志
NetAssist V5.
[2022-05-10 11:08:54.548]# Client 192.168.0.252:49153 gets online.
服务器
行结束时， bit7 == TRUE， bit
TCPC_XMT
%M3.0:
%VB2100:
EXEC
CLIENT
DATA
LEN
TIMEOUT
RETRY
22
55
1888
2
%MB
[2022-05-10 11:08:54.548]# Client 192.168.0.252:49153 gets onlir
[2022-05-10 11:11:21.587]# Client 192.168.0.252:49153 gets onlir
[2022-05-10 11:12:39.931]# RECV HEX FROM 192.168.0.252 :49153>
22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
22 00 00
数据发送
变量类型
数据类型
注 释
VAR
BOOL
IMEOUT为0的TCPC_RCV，因为位于前面，
面的TCPC_XMTRCV，无法收到client 1新
TCPC_RCV
%M4.0:
NEWPKG
MAXLEN
22222
%MB1006: 0
%M333.3: 0
%VB3100: 56
%VW3198: 3
数据接收
[2022-05-10 11:14:33.323]# SEND HEX TO ALL>
56 34 12
接收数据不显示
接收保存到文件...

IL	(* Network 0 *) (*LINE Parameter TRUE means network cable connected, FALSE Indicates that the network cable is disconnected CLIENT parameter DWORD, each bit indicates a TCP client's status SERVER Parameter DWORD, each bit indicates a TCP server's status UDPPEER parameter DWORD, Each bit represents the status of a UDP peer*) LD %SM0.0 ETH_STATUS %M333.7, %MD334, %MD338, %MD342 (* Network 1 *) (*In this example, the input parameter is a constant remark. If a remote server is already connected, and the IP or PORT is modified to point to a new remote server, executing this instruction will automatically close the old connection, and then connect to the new remote server.*) LD %SM0.0 TCP_CLIENT %M0.6, 0, 192.168.0.89, W#1000, 4321, %MB1000, %M333.0 TCP_CLIENT %M0.6, 1, 192.168.0.89, W#1001, 4321, %MB1001, %M333.1
IL	(* Network 2 *) (*The input parameter is a constant method to close the TCP client *) LD %SM0.0 TCPC_CLOSE %M0.5, 0, %MB1008 TCPC_CLOSE %M0.7, 1, %MB1009 (* Network 3 *) (*In this example, the input parameters are constant, CLIENT, LEN, TIMEOUT, RETRY, these input parameters can be constants at the same time, or send data for variables at the same time *) LD %SM0.0 TCPC_XMT %M3.0, 1, %VB2000, 44, 1111, 1, %MB1002 TCPC_XMT %M3.0, 0, %VB2100, 55, 1888, 2, %MB1003 (* Network 4 *) (*Outgoing string for TCP_XMT *) LD %SM0.1 FILL B#16#11, %VB2000, B#43 FILL B#16#22, %VB2100, B#53 (* Network 5 *) (*In this example, the input parameters are constants, CLIENT, MAXLEN, TIMEOUT, these input parameters can be constants at the same time, or receive data for variables at the same time *) LD %SM0.0

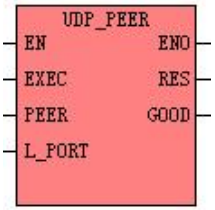
	TCPC_RCV %M4.0, 1, 49, 0, %MB1005, %M333.2, %VB3000, %VW3098 TCPC_RCV %M4.0, 0, 56, 22222, %MB1006, %M333.3, %VB3100, %VW3198 (* Network 6 *) (*Observe the TCPC_RCV instruction, the number of packets received from client 1 *) LD %M333.2 R_TRIG INC %VD4000
IL	(* Network 7 *) (*Observe the TCPC_RCV instruction, the number of packets received from client 0 *) LD %M333.3 R_TRIG INC %VD4004 (* Network 8 *) (*In this example, the input parameters are constants. CLIENT, XLEN, MAXLEN, TIMEOUT, RETRY, these input parameters can be constants at the same time, or send data first and then receive data for variables at the same time *) LD %SM0.0 TCPC_XMTRCV %M5.0, 0, %VB5000, 45, 99, 11223, 0, %VB6000, %V6001.0, %VB5100, %VW6002 TCPC_XMTRCV %M5.0, 1, %VB5200, 56, 99, 11224, 1, %VB6004, %V6005.0, %VB5300, %VW6006 (* Network 9 *) (*Emitted string for TCP_XMTRCV *) LD %SM0.1 FILL B#16#33, %VB5000, B#43 FILL B#16#44, %VB5200, B#53 (* Network 10 *) (*Observe the number of packets received by TCPC_XMTRCV from client 0 *) LD %V6001.0 R_TRIG INC %VD4008 (* Network 11 *) (*Observe the number of packets received by TCPC_XMTRCV from client 1 *) LD %V6005.0 R_TRIG INC %VD4012

10.3.3.5 UDP instruction

10.3.3.5.1 UDP_PEER instruction

When you need to use the native UDP resource to communicate, first use this instruction to enable UDP communication.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	UDP_PEER			☑ K6
IL	UDP_PEER	UDP_PEER *EXEC,PEER,L_PORT, RES,GOOD*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
PEER	Input	INT	M、V、L、Constant
L_PORT	Input	WORD	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM
GOOD	Output	BOOL	Q、M、V、L、SM

Note: The parameters PEER, L_PORT must be both constant types or memory types at the same time。

- Parameter description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute, and UDP communication is started.
PEER	UDP PEER resource number, in the range 0--15, that is, a maximum of 16 UDP PEER resources are supported.
L_PORT	The port of the local UDP PEER, 0 is not supported.

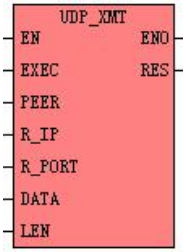
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 Created successfully 1 PEER parameter error 2 PORT parameter input error 3 Failed to create 4 The current PEER has been created
GOOD	Each time EXEC is executed, the GOOD parameter is first assigned the value FALSE. TRUE indicates that the connection to the server was successful, and FALSE indicates that the connection failed. This parameter is updated once after the instruction is executed and will not be updated dynamically. You can view the change of dynamic connection through the ETH_STATUS instruction.

Note: Because the PLC has only one network card, it is not necessary to input the local IP. The default is to take the local IP of the PLC.

10.3.3.5.2 UDP_XMT instruction

This instruction is the sending instruction during local UDP communication.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	UDP_XMT			☑ K6
IL	UDP_XMT	UDP_XMT *EXEC,PEER,R_IP,R_PORT,DATA, LEN,RES*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
PEER	Input	INT	M、V、L、Constant
R_IP	Input	DWORD	M、V、L、Constant

R_PORT	Input	WORD	M、V、L、Constant
DATA	Input	BYTE	M、V、L
LEN	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM

Note: The parameters PEER, R_IP, R_PORT, LEN must be constant types or memory types at the same time.

- Parameters description

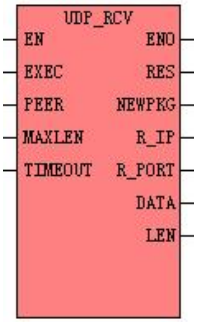
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
PEER	The number of the UDP PEER resource that has been enabled, in the range 0--15.
R_IP	The IP address of the remote UDP does not support all 0s. You can directly enter the IP address, for example: 192.168.0.250.
R_PORT	The port of the remote UDP. 0 is not supported.
DATA	The first address of the data byte to be sent.
LEN	The number of bytes of sent data, cannot be 0, cannot be negative, and cannot be greater than 1460 bytes.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 Sending is completed, it only means that it has been submitted to the UDP protocol stack, and does not mean that the remote end has received it. 1 PEER parameter error 2 Sending length error 3 Spare 4 No successfully created UDP PEER can be used to send data

10.3.3.5.3 UDP_RCV instruction

This instruction is the receiving instruction when the machine is used as UDP communication.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	☑ K6

LD	UDP_RCV			
IL	UDP_RCV	UDP_RCV *EXEC, PEER, MAXLEN, TIMEOUT, RES, NEWPKG, R_IP, R_PORT, DATA, LEN*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
PEER	Input	INT	M、V、L、Constant
MAXLEN	Input	INT	M、V、L、Constant
TIMEOUT	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM
NEWPKG	Output	BOOL	Q、M、V、L、SM
R_IP	Output	DWORD	Q、M、V、L、SM
R_PORT	Output	WORD	Q、M、V、L、SM
DATA	Output	BYTE	Q、M、V、L、SM
LEN	Output	WORD	Q、M、V、L、SM

Note: The parameters PEER, MAXLEN, TIMEOUT must be both constant types or memory types at the same time.

- Parameter description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
PEER	The number of the UDP PEER resource that has been enabled, in the range 0--15.
MAXLEN	The maximum number of data bytes that can be received in a message. It cannot be entered as 0 or negative. The maximum number of bytes received is 1460 bytes. When MAXLEN is greater than 1460, it will be ignored.
TIMEOUT	The timeout time of this execution, in ms, cannot be greater than 32767ms, and cannot be

	a negative number. 0 means waiting forever, and is always in the state of waiting for UDP packets. When it is not 0, such as 5000, it means that it has been waiting for the message from the remote UDP within 5 seconds, and no longer receives the message after the timeout.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 Spare 1 PEER parameter error 2 MAXLEN length error 3 Operation timeout, end receiving 4 Spare 5 The TIMEOUT parameter entered by the user is incorrect
NEWPKG	When a new UDP packet is received, NEWPKG generates a rising edge, which only lasts for one scan period.
R_IP	At the instant of the rising edge of NEWPKG, the IP address of the remote UDP that received the message.
R_PORT	At the instant of the rising edge of NEWPKG, the port of the remote UDP that received the message.
DATA	The starting byte of the receiving buffer. Note that this buffer is matched with MAXLEN. For example, MAXLEN is 100 and DATA=VB0. At this time, VB0--VB99 are all buffers, which can only be read and cannot be used for other purposes. . Users can clear the buffer by themselves. Otherwise, when the length of the next received message is smaller than the last one, the buffer will have the data of the last message.
LEN	The length of the received packet. When the actual packet length is less than or equal to MAXLEN, LEN is the actual packet length. When the actual packet length exceeds MAXLEN, LEN=MAXLEN.

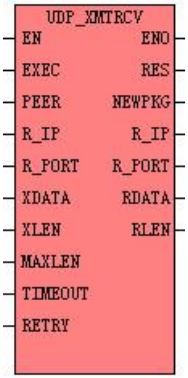
Note:
1、After a rising edge of EXEC is started, any number of messages can be received. Pay attention to observe NEWPKG.
2、When the NEWPKG parameter is on the rising edge, the output of these parameters R_IP, R_PORT, DATA, LEN is meaningful, and remains unchanged at other times.

10.3.3.5.4 UDP_XMTRCV instruction

This instruction is the sending and receiving instruction during the local UDP communication。

- Parameter description

	Name	Instruction format	Affected CR value	☑ K6

LD	UDP_XMTRCV			
IL	UDP_XMTRCV	UDP_RCV *EXEC, PEER, R_IP, R_PORT, XDATA, XLEN, MAXLEN, TIMEOUT, RETRY, RES, NEWPKG, R_IP, R_PORT, DATA, LEN*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
PEER	Input	INT	M、V、L、Constant
R_IP	Input	DWORD	M、V、L、Constant
R_PORT	Input	WORD	M、V、L、Constant
XDATA	Input	BYTE	M、V、L
XLEN	Input	INT	M、V、L、Constant
MAXLEN	Input	INT	M、V、L、Constant
TIMEOUT	Input	INT	M、V、L、Constant
RETRY	Input	INT	M、V、L、Constant
RES	Output	BYTE	Q、M、V、L、SM
NEWPKG	Output	BOOL	Q、M、V、L、SM
R_IP	Output	DWORD	Q、M、V、L、SM
R_PORT	Output	WORD	Q、M、V、L、SM
DATA	Output	BYTE	Q、M、V、L、SM
RLEN	Output	WORD	Q、M、V、L、SM

Note: Parameters PEER, R_IP, R_PORT, XLEN, MAXLEN, TIMEOUT, RETRY must be

constant type or memory type at the same time.

- Parameters description

Parameter	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
PEER	The number of the UDP PEER resource that has been opened, ranging from 0 to 15.
R_IP	The IP address of the UDP sent to the remote end. All 0s are not supported. You can directly enter the IP address, for example: 192.168.0.250.
R_PORT	The port sent to the remote UDP. 0 is not supported.
XDATA	The first address of the data byte to be sent.
XLEN	The number of bytes of sent data, cannot be 0, cannot be negative, and cannot be greater than 1460 bytes.
MAXLEN	The maximum number of data bytes that can be received in a message. It cannot be entered as 0 or negative. The maximum number of bytes received is 1460 bytes. When MAXLEN is greater than 1460, it will be ignored.
TIMEOUT	The timeout time of this execution, in ms, cannot be 0ms, cannot be greater than 32767ms, and cannot be a negative number: For example, if you enter 5000, it means that within 5 seconds, the following sequence of actions will be performed: a. Wait for the UDP PEER to be established successfully, and send the data to be sent immediately after the success. If the connection is not successful, return the failure and end. b. After sending successfully, wait for receiving data, if not, return failure and end. c. After successful transmission, a packet of data is received, and the instruction ends successfully.
RETRY	Number of retries, range 0-255. If the input exceeds this range, the value of the low byte of this parameter will be automatically intercepted. The number of retries after failure to send or fail to receive. For example, if the number of retries is 1, assuming that each sending fails, after starting EXEC, a total of 2 times are sent.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 The instruction ends successfully, a message is sent, and a message is received 1 PEER parameter error 2 Length error, MAXLEN is 0, or XLEN is 0, or XLEN is greater than 1460 bytes, only the maximum length of 1460 is received, when MAXLEN is greater than 1460, it will be ignored 3 operation time out, message not received 4 Failed to send, no UDP PEER has been established successfully 5 TIMEOUT parameter, input error
NEWPKG	This instruction EXEC will automatically become FALSE when it is started, and it will become TRUE when a message is successfully received until the next time this instruction is started.

R_IP	At the instant of the rising edge of NEWPKG, the IP address of the remote UDP that received this message. Because UDP is connectionless, you may send a message to slave A and receive a receipt from slave B while waiting for the receipt.
R_PORT	At the instant of the rising edge of NEWPKG, the remote UDP port that received this message. Because UDP is connectionless, you may send a message to slave A and receive a receipt from slave B while waiting for the receipt.
RDATA	The starting byte of the receiving buffer. Note that this buffer is matched with MAXLEN. For example, MAXLEN is 100 and DATA=VB0. At this time, VB0--VB99 are all buffers, which can only be read and cannot be used for other purposes. . The user can clear the buffer by themselves, otherwise, when the length of the next received message is less than the previous one, the buffer will remain the last message data。
RLEN	The length of the received packet. When the actual packet length is less than or equal to MAXLEN, LEN is the actual packet length. When the actual packet length exceeds MAXLEN, LEN=MAXLEN.

Note: Only 1 message can be received after the rising edge of EXEC is started. Take the rising edge of NEWPKG to read the message.

When the UDP_RCV and UDP_XMTRCV instructions are used at the same time, only the first instruction in the program can receive the message.

10.3.3.5.5 Example

In this example, the network debugging assistant communicates with the PLC through UDP.

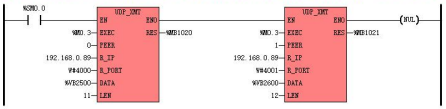
LD

```
(* Network 0 *)
(* The LINE parameter 'TRUE' indicates that the Ethernet cable is plugged in, 'False' indicates that the Ethernet cable is unplugged.
 The 'CLIENT' parameter is set to 'WORD', where each bit represents the status of a TCP client.
The 'SERVER' parameter, each bit represents the status of a TCP server. The 'UDPPEER' parameter is set to 'WORD', and each bit represents the status of a UDP peer *)
```

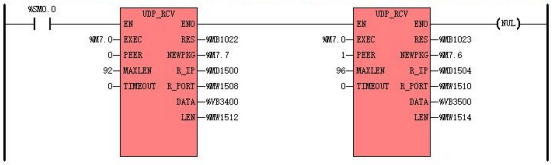
```
(* Network 1 *)
(* This example uses constant input parameters PEER and L_PORT. These input parameters can be both constants or UDP enabled for variables at the same time *)
```

```
(* Network 2 *)
(* Send string for UPD_XMT *)
```

```
(* Network 3 *)
(* This example uses constant input parameters, PEER, R_IP, R_PORT, LEN,  These input parameters can be constants at the same time or send data to variables at the same time *)
```



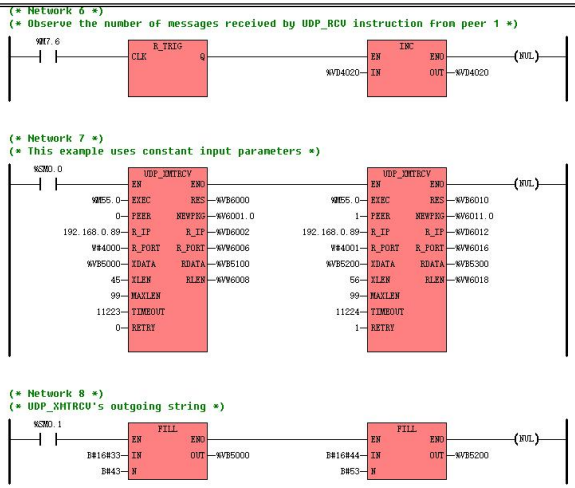

```
(* Network 4 *)
(* This example uses constant input parameters, PEER, MAXLEN, TIMEOUT, which can be constants or variables at the same time *)
```



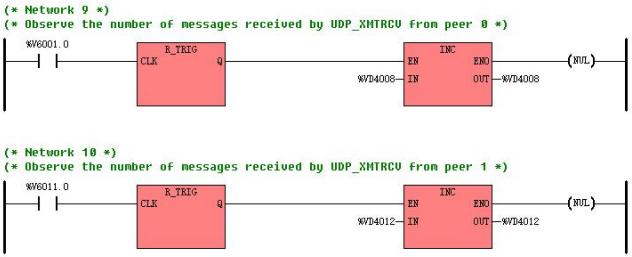

```
(* Network 5 *)
(* Observe the number of messages received by UDP_RCV instruction from peer 0 *)
```

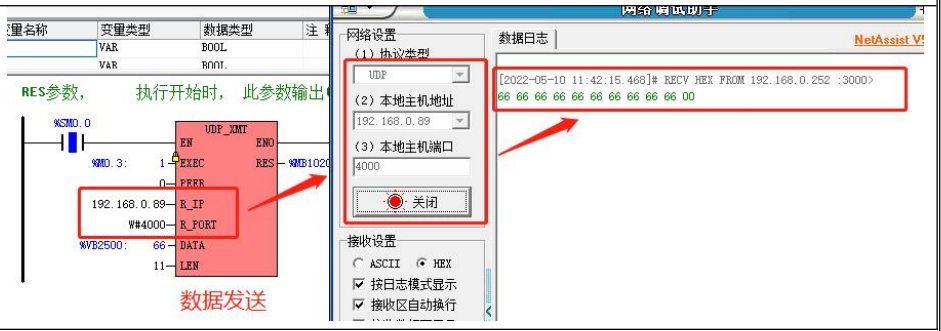
```
(* Network 6 *)
(* Observe the number of messages received by UDP_RCV instruction from peer 1 *)
```

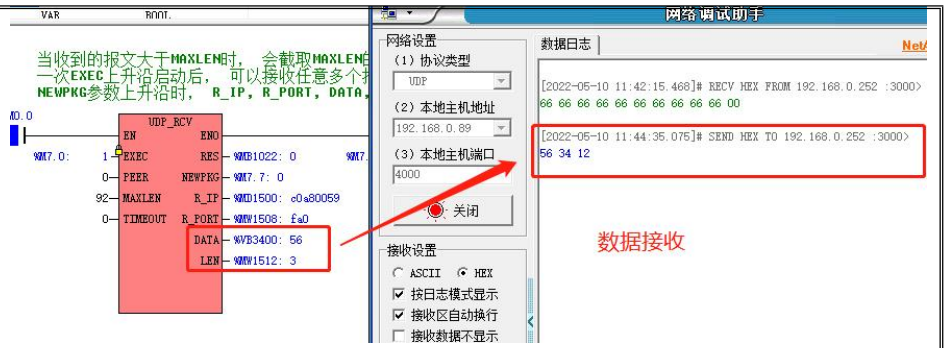
LD


(* Network 6 *)
(* Observe the number of messages received by UDP_RCV instruction from peer 1 *)
%M7.6
R_TRIG
CLK
Q
INC
EN
ENO
%VD4020
IN
OUT
(NUL)
(* Network 7 *)
(* This example uses constant input parameters *)
%SM0.0
UDP_XMTRCV
%M55.0
EXEC
RES
%VB6000
0
PEER
NEWPRG
%V6001.0
192.168.0.89
R_IP
%VD6002
W#4000
R_PORT
%VW6006
%VB5000
IDATA
%VB5100
45
XLEN
%VW6008
99
MAXLEN
11223
TIMEOUT
RETRY
%VB6010
%V6011.0
%VD6012
%VW6016
%VB5200
%VB5300
%VW6018
1
11224
56
(* Network 8 *)
(* UDP_XMTRCV's outgoing string *)
%SM0.1
FILL
B#16#33
B#43
N
B#16#44
B#53
%VB5200

LD


(* Network 9 *)
(* Observe the number of messages received by UDP_XMTRCV from peer 0 *)
%V6001.0
R_TRIG
CLK
Q
INC
EN
ENO
%VD4008
IN
OUT
(NUL)
(* Network 10 *)
(* Observe the number of messages received by UDP_XMTRCV from peer 1 *)
%V6011.0
%VD4012


变量名称
变量类型
数据类型
VAR
BOOL
RES参数，
执行开始时，
此参数输出
%SM0.0
UDP_XMT
%M0.3:
EXEC
RES
%MB1020
PEER
192.168.0.89
R_IP
W#4000
R_PORT
%VB2500:
66
DATA
11
LEN
数据发送
网络设置
(1) 协议类型
UDP
(2) 本地主机地址
192.168.0.89
(3) 本地主机端口
4000
关闭
接收设置
ASCII
HEX
按日志模式显示
接收区自动换行
数据日志
NetAssist V5
[2022-05-10 11:42:15.468]# RECV HEX FROM 192.168.0.252 :3000>
66 66 66 66 66 66 66 66 66 66 00

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>IL</td><td>(* Network 0 *)
(*The LINE parameter 'TRUE' indicates that the Ethernet cable is plugged in, 'False' indicates that the Ethernet cable is unplugged. The 'CLIENT' parameter is set to 'WORD', where each bit represents the status of a TCP client. The 'SERVER' parameter, each bit represents the status of a TCP server. The 'UDPPEER' parameter is set to 'WORD', and each bit represents the status of a UDP peer *)
LD %SM0.0
ETH_STATUS %M333.7, %MD334, %MD338, %MD342
(* Network 1 *)
(*This example uses constant input parameters PEER and L_PORT. These input parameters can be both constants or UDP enabled for variables at the same time *)
LD %SM0.0
UDP_PEER %M4.0, 0, W#3000, %MB1018, %M5.0
UDP_PEER %M4.0, 1, W#3001, %MB1019, %M5.1
(* Network 2 *)
(*Send string for UPD_XMT *)
LD %SM0.1
FILL B#16#66, %VB2500, B#10
FILL B#16#77, %VB2600, B#11
(* Network 3 *)
(*This example uses constant input parameters, PEER, R_IP, R_PORT, LEN, These input parameters can be constants at the same time or send data to variables at the same time *)
LD %SM0.0
UDP_XMT %M0.3, 0, 192.168.0.89, W#4000, %VB2500, 11, %MB1020
UDP_XMT %M0.3, 1, 192.168.0.89, W#4001, %VB2600, 12, %MB1021
(* Network 4 *)
(*This example uses constant input parameters, PEER, MAXLEN, TIMEOUT, which can be constants or variables at the same time *)</td></tr>
</table>

	LD %SM0.0 UDP_RCV %M7.0, 0, 92, 0, %MB1022, %M7.7, %MD1500, %MW1508, %VB3400, %MW1512 UDP_RCV %M7.0, 1, 96, 0, %MB1023, %M7.6, %MD1504, %MW1510, %VB3500, %MW1514
IL	(* Network 5 *) (*Observe the number of messages received by UDP_RCV instruction from peer 0 *) LD %M7.7 R_TRIG INC %VD4016 (* Network 6 *) (*Observe the number of messages received by UDP_RCV instruction from peer 1 *) LD %M7.6 R_TRIG INC %VD4020 (* Network 7 *) (*This example uses constant input parameters *) LD %SM0.0 UDP_XMTRCV %M55.0, 0, 192.168.0.89, W#4000, %VB5000, 45, 99, 11223, 0, %VB6000, %V6001.0, %VD6002, %VW6006, %VB5100, %VW6008 UDP_XMTRCV %M55.0, 1, 192.168.0.89, W#4001, %VB5200, 56, 99, 11224, 1, %VB6010, %V6011.0, %VD6012, %VW6016, %VB5300, %VW6018 (* Network 8 *) (*UDP_XMTRCV's outgoing string *) LD %SM0.1 FILL B#16#33, %VB5000, B#43 FILL B#16#44, %VB5200, B#53 (* Network 9 *) (*Observe the number of messages received by UDP_XMTRCV from peer 0 *) LD %V6001.0 R_TRIG INC %VD4008 (* Network 10 *) (*Observe the number of messages received by UDP_XMTRCV from peer 1 *) LD %V6011.0

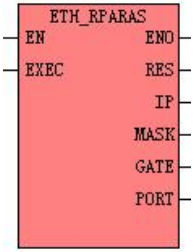
	R_TRIG INC %VD4012

10.3.3.6 Local Ethernet parameter read and write instructions

10.3.3.6.1 ETH_RPARAS instruction

This instruction is to read the local Ethernet parameter instruction.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	ETH_RPARAS			☑ K6
IL	ETH_RPARAS	ETH_RPARAS *EXEC, RES, IP, MASK, GATE, PORT*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
RES	Output	BYTE	Q、M、V、L、SM
IP	Output	DWORD	Q、M、V、L、SM
MASK	Output	DWORD	Q、M、V、L、SM
GATE	Output	DWORD	Q、M、V、L、SM
PORT	Output	WORD	Q、M、V、L、SM

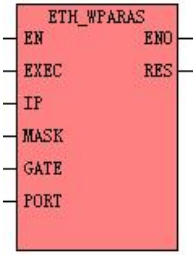
➢ Parameter description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 read successfully
IP	The IP address of the native Ethernet.
MASK	The subnet mask of the native Ethernet.
GATE	Gateway to native Ethernet.
PORT	The port number of the native Ethernet.

10.3.3.6.2 ETH_WPARAS instruction

This instruction is to modify the local Ethernet parameters.

➢ Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	ETH_WPARAS			☑ K6
IL	ETH_WPARAS	ETH_WPARAS *EXEC, IP, MASK, GATE, PORT, RES*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
IP	Input	DWORD	M、V、L、SM、constant
MASK	Input	DWORD	M、V、L、SM、Constant
GATE	Input	DWORD	M、V、L、SM、Constant
PORT	Input	WORD	M、V、L、SM、Constant

RES	Output	BYTE	M、V、L、SM、Constant

Note: Parameters IP, MASK, GATE, PORT must be both constant types or memory types at the same time.

- Parameter description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
IP	The IP address of the local Ethernet that needs to be modified. All 0s are not supported.
MASK	The subnet mask of the native Ethernet that needs to be modified.
GATE	Gateway to native Ethernet that needs to be modified.
PORT	The port number of the local Ethernet that needs to be modified. All 0s are not supported.
RES	At the beginning, this parameter outputs 0, at the end, bit7 == TRUE, the byte value of the combination of bit0-bit6, indicating the error code 0 Write success 1 Parameter error, there is an input parameter of 0, which is not supported. 2 An error occurred while writing new parameters to permanent storage.。

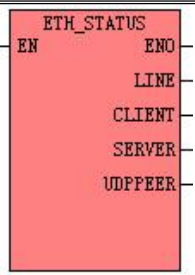
NOTE: After writing new parameters, the entire Ethernet hardware will reboot.

10.3.3.7 Local Ethernet Status Monitoring Instructions and Reset Instructions

10.3.3.7.1 ETH_STATUS instruction

This instruction is to monitor the local Ethernet status instruction.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	
LD	ETH_STATUS			☑ K6
IL	ETH_	ETH_ STATUS *LINE, CLIENT,*	U	

	STATUS	*SERVER, UDPPEER*		

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
LINE	Output	DWORD	M、V、L、SM
CLIENT	Output	DWORD	M、V、L、SM
SERVER	Output	DWORD	M、V、L、SM
UDPPEER	Output	DWORD	M、V、L、SM

- Parameter description

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
LINE	TRUE means the network cable is plugged in, FALSE means the network cable is unplugged.
CLIENT	Each bit represents the status of a TCP Client, such as bit0==TRUE, which means that CLIENT 0 is in a connected state, and bit0==FALSE, which means that CLIENT 0 is in a state of not being able to send and receive data.
SERVER	Each bit represents the status of a TCP Server. For example, bit9==TRUE, it means that the server 9 is in a connected state, and bit9==FALSE, it means that the server 9 is in a state where it cannot send and receive data.
UDPPEER	Each bit represents the status of a UDP peer, such as bit11==TRUE, which means that peer 11 has been created, and bit11==FALSE, which means that peer 11 is not used.

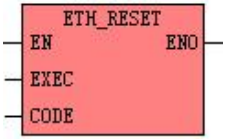
Note: LINE is the real-time display status. The three parameters of CLIENT, SERVER, and UDPPEER are not completely real-time, but are determined by the TCP protocol stack.

10.3.3.7.2 ETH_RESET instruction

This instruction is the reset Ethernet instruction.

- Instruction and its operand description

	Name	Instruction format	Affected CR value	☑ K6

LD	ETH_RESET			
IL	ETH_ RESET	ETH_ RESET *EXEC, CODE*	U	

Parameters	Input/Output	Data type	memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
CODE	Input	BYTE	Constant

- Parameter description

Parameter	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute, and the entire Ethernet function is reset once every rising edge is executed, and all connections will be invalid and need to be reconnected.
CODE	Can only be b#1, or b#2 When b#1, the hardware sending and receiving buffer of the Ethernet part of the MCU is not cleared. When b#2, clear the hardware transceiver buffer of the Ethernet part of the MCU.

Note:

1、This instruction is designed for unattended use and is generally not recommended.

2、When the device is only used for communication and unattended, if the PLC can judge that the necessary communication is abnormal, you can try to use this instruction to restart the Ethernet to eliminate the suspended animation when the device is running all the year round.

10.4 LPWAN Use of wireless communication port

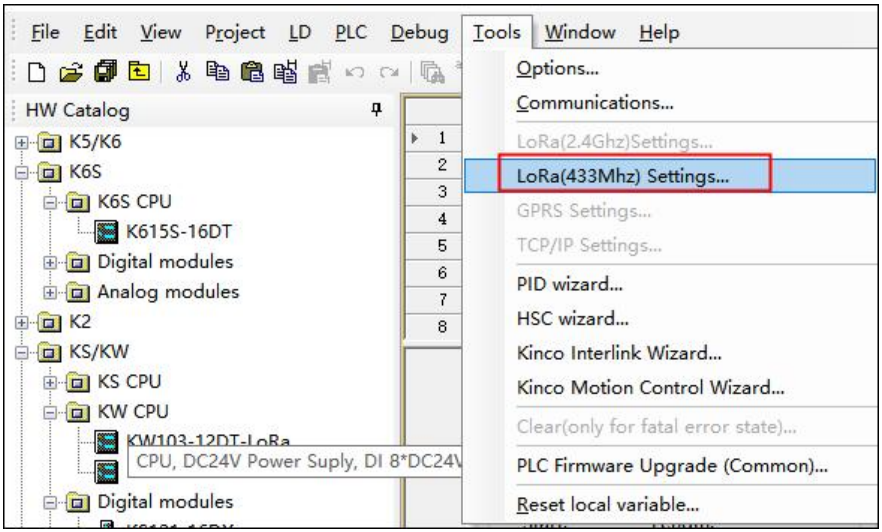
There are currently three types of CPUs that support LoRa wireless communication ports, two standard CPUs (KW103, KW203), and one simple CPU (KW213). The working frequency bands they support are as follows:

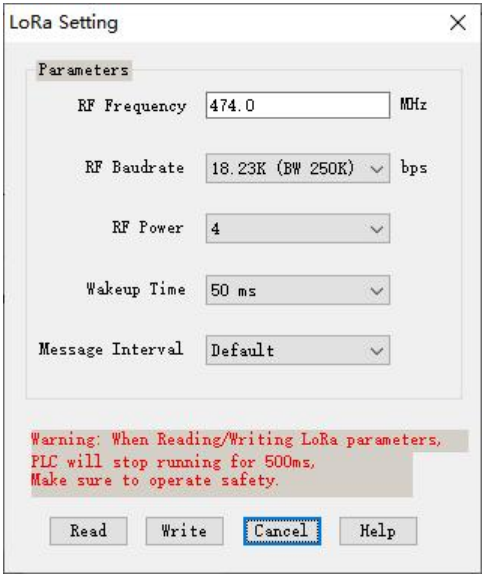
Type Working frequency

KW103	410～493MHz
KW203 KW213	2400～2500MHz

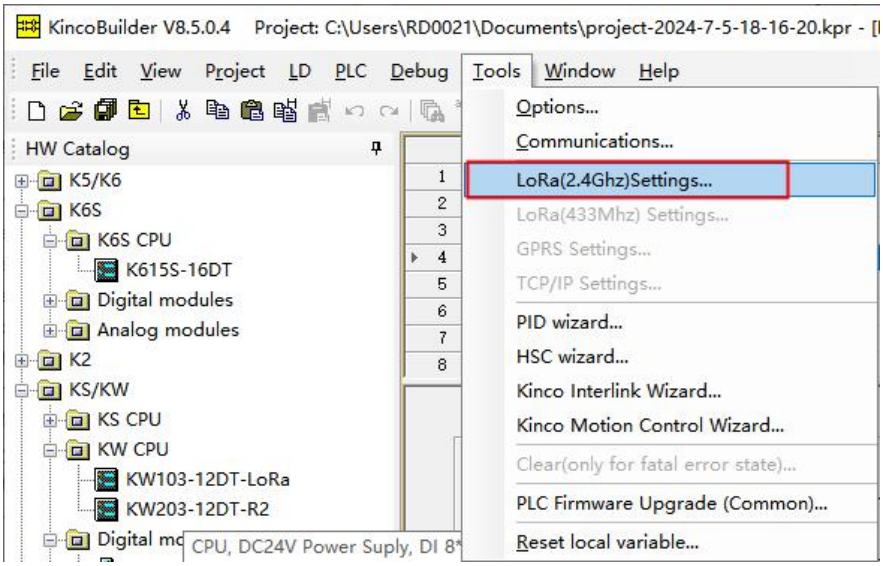
In addition to the different working frequency bands, KW103 and KW203 support some instructions that are not exactly the same. The free communication instruction KW103 supports the earliest XMT and RCV instructions, while KW203 supports the subsequent addition of COM_XMT and COM_RCV instructions, as well as dedicated read instructions for the LORA port. Only KW203 can use the write parameter instruction, which will be introduced in the following chapters.

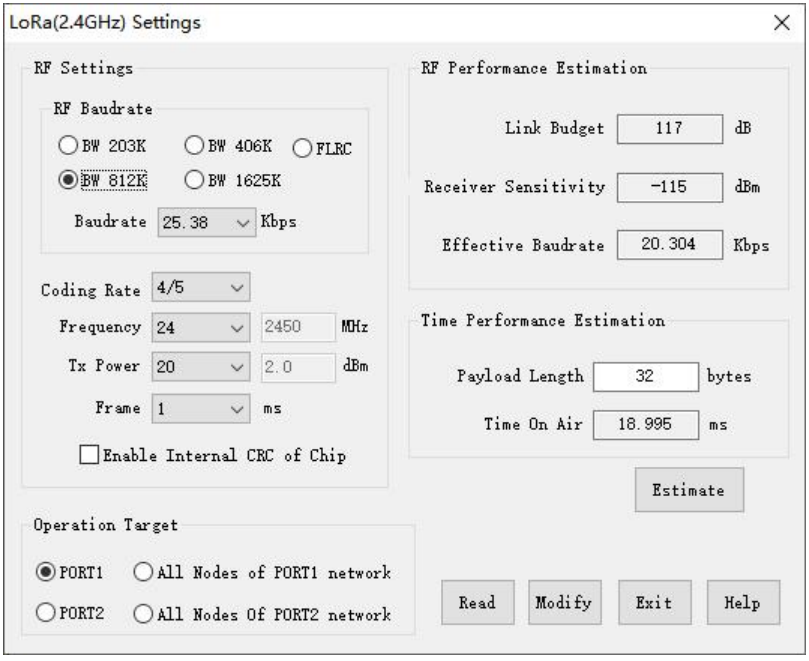
Regarding the communication parameter setting of KW103, the user executes the [Tools] -> [LoRa (433M) Parameter Settings] menu instruction, as shown in the figure below, in the pop-up window, the parameters such as frequency and air transmission speed can be read and written.





Regarding the communication parameter setting of KW203, the user executes the [Tools] -> [LoRa (2.4G) Parameter Setting] menu instruction, as shown in the figure below, and the LoRa communication parameters can be read and written in the pop-up window.


KincoBuilder V8.5.0.4 Project: C:\Users\RD0021\Documents\project-2024-7-5-18-16-20.kpr -
File Edit View Project LD PLC Debug Tools Window Help
HW Catalog
K5/K6
K6S
K6S CPU
K615S-16DT
Digital modules
Analog modules
K2
KS/KW
KS CPU
KW CPU
KW103-12DT-LoRa
KW203-12DT-R2
Digital mo
CPU, DC24V Power Suply, DI 8*
Options...
Communications...
LoRa(2.4Ghz)Settings...
LoRa(433Mhz) Settings...
GPRS Settings...
TCP/IP Settings...
PID wizard...
HSC wizard...
Kinco Interlink Wizard...
Kinco Motion Control Wizard...
Clear(only for fatal error state)...
PLC Firmware Upgrade (Common)...
Reset local variable...


LoRa(2.4GHz) Settings
RF Settings
RF Baudrate
BW 203K
BW 406K
FLRC
BW 812K
BW 1625K
Baudrate 25.38 Kbps
Coding Rate 4/5
Frequency 24 2450 MHz
Tx Power 20 2.0 dBm
Frame 1 ms
Enable Internal CRC of Chip
RF Performance Estimation
Link Budget 117 dB
Receiver Sensitivity -115 dBm
Effective Baudrate 20.304 Kbps
Time Performance Estimation
Payload Length 32 bytes
Time On Air 18.995 ms
Estimate
Operation Target
PORT1
All Nodes of PORT1 network
PORT2
All Nodes Of PORT2 network
Read
Modify
Exit
Help

10.4.1 Overview

LoRa is one of the LPWAN (Low Power Wide Area Network) communication technologies. It is a narrowband wireless communication solution based on spread spectrum technology and ultra-long distance. It has the advantages of wide coverage, strong anti-interference ability, and low transmission power. LoRa works in the license-free frequency band, and the deployment is convenient and flexible. Users can freely set up their own private network.

10.4.2 Common Wireless Communication Terminology

1） Decibel milliwatt (dBm)

Decibel milliwatt (dBm) is a value that characterizes the absolute value of power, and it is a ratio based on 1mW power. Its calculation formula is 10*lg(P), where P is the power (in mW). For dBm, there are several simple and intuitive empirical values:

- The power of 1mW is equal to 0dBm; the power of 1W is equal to 30dBm.
- An increase of 3dBm means that the power is multiplied by 2; a decrease of 3dBm means that the power is divided by 2.
- An increase of 10dBm means that the power is multiplied by 10; a decrease of 10dBm means that the power is divided by 10.

2） decibel (dB)

dB is a value that characterizes a relative value.

When considering how many dBs the power of A is larger or smaller than the power of B, the following calculation formula is used: 10*lg(A/B).

For example, the power of A is twice as high as that of B, then 10*lg (A power/B power)=10*lg2=3dB.

3） Bandwidth (BW)

The channel bandwidth is the difference between the lower limit frequency and the upper limit frequency of the signal allowed to pass through the channel, which can be understood as a frequency passband.

The bandwidth of LoRa is bilateral bandwidth. Assuming that the center frequency of a LoRa channel

is f and the bandwidth is BW, the actual frequency range of the channel is $[f - \frac{BW}{2}, f + \frac{BW}{2}]$.

4） Coding rate (CR)

LoRa uses cyclic error correction coding for forward error detection and error correction, which can effectively improve the reliability of the link in the case of severe interference, but at the cost of adding some redundant information during coding.

The coding rate refers to the proportion of the useful (non-redundant) part of the communication data stream. If the coding rate is k/n, then for every k bits of useful information, the encoder produces a total of n bits of data, where n-k is redundant.

5） Receiver Sensitivity

Receiving sensitivity refers to the minimum power of the signal that the receiver can receive correctly. When the signal energy is less than the nominal receiving sensitivity, the receiver will not receive.

Receiving sensitivity is to test the ability of the receiver to receive weak signals. The smaller the value, the stronger the receiving ability. For example, the receiving sensitivity of A is -85dBm, and that of B is -140dBm, then obviously the receiving ability of B is much stronger than that of A, and the wireless system using B will also cover a wider range than using A.

6） Link Budget

Under the premise of maintaining a certain communication quality, the maximum propagation loss allowed by the communication link.

Link budget is the main method to evaluate the coverage capability of wireless communication system. In a given environment, the larger the link budget value, the larger the coverage of wireless communication.

10.4.3 Parameter setting of LoRa communication port

The LoRa communication port provided by KW1 is different from the working frequency band of KW2, and the applicable scenarios are also different, but the usage methods of the two are basically the same, so the following will take KW2 as an example to illustrate.

The LoRa communication port supports programming protocol, Kinco PLC interconnection protocol, Modbus RTU protocol (master station and slave station) and free communication function. The programming software provides a variety of tools such as wizards, parameter configurations, and network status monitoring to facilitate user applications. In addition, various communication instructions are provided. Users can call these instructions in the program to realize online operation of LoRa communication. . These functions are described in detail below.

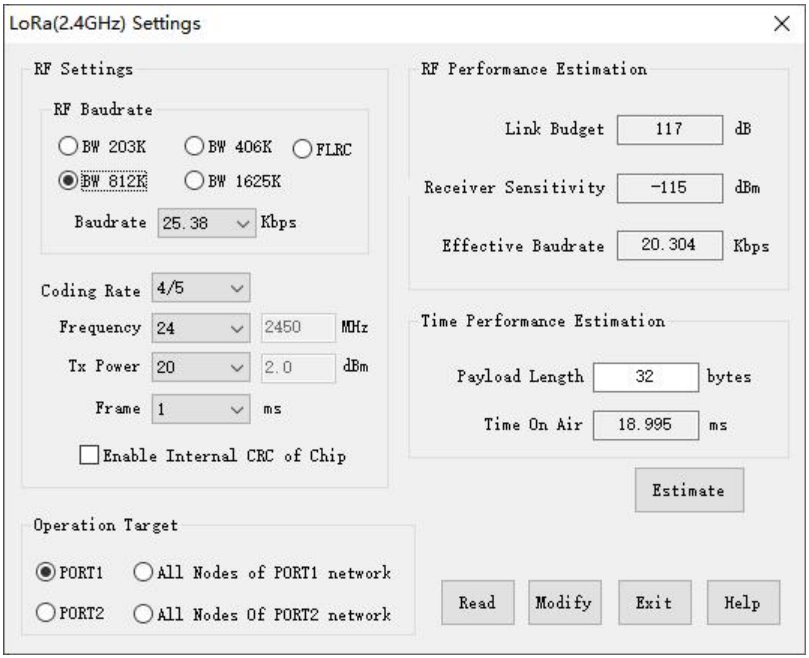
10.4.3.1 Configure LoRa parameters

KW provides two LoRa parameter configuration methods:

- Use the parameter configuration tool provided by KincoBuilder software.
- Use the parameter read and write instructions in the user program.

10.4.3.1.1 Use the parameter configuration tool

In the KincoBuilder software, execute the [Tools] -> [LoRa(2.4GHz) parameter configuration...] menu instruction to enter the configuration tool window. Users can configure the LoRa interface here. In addition, you can also intuitively see the estimated results of wireless communication performance when using different communication parameters, which helps you choose the most suitable parameter combination for your application.



➢ **Operational objectives**

Select the interface or device to be modified.

Different models of KW modules provide 1 or 2 LoRa interfaces and are assigned different numbers. If the module has only one LoRa port, the number of this port is 1. If the module has 2 LoRa ports, the number of each port is marked on the module silkscreen.

- [Local port 1]: Indicates the parameters of LoRa port 1 that will operate the module connected to the computer.
- [Local port 2]: Indicates the parameters of LoRa port 2 that will operate the module connected to the computer.
- [All nodes in port 1 network]: This option only supports [Write] operation, which means that the LoRa interface parameters of all modules that can communicate normally in the wireless network where LoRa port 1 of the module connected to the computer is to be modified.
- [All nodes in the port 2 network]: This option only supports the [Write] operation, which means that the

LoRa interface parameters of all modules that can communicate normally in the wireless network where the LoRa port 2 of the module connected to the computer is to be modified.

When choosing to modify the parameters of "all nodes in the network", please pay attention to:

1） The modification process will last about 3 seconds.

2） Modifying parameters may affect normal data communication, please modify under the premise of ensuring safety!

3） If the parameters of all nodes in the network are modified, other nodes in the network are not allowed to send data, otherwise the modification may fail! And because LoRa is a half-duplex communication mode, the configuration software cannot judge whether there is a node that failed to modify!

It is recommended that users execute this function through the master station in the network to avoid unsuccessful operation.

- **Communication parameters**

- [Air transmission rate]: refers to the wireless (in the air) data transmission rate, which can be understood as similar to the baud rate of wired communication. The higher the air rate, the faster the data transmission speed, but usually the transmission distance will be shorter.

  LoRa provides a variety of bandwidths, and each bandwidth provides a variety of air transmission rates. In the software, the user needs to select a bandwidth first, then the software will automatically list all the air rates in this group, and the user can make a choice.

  The factory default is 25.38Kbps.

- [Coding rate]: LoRa coding rate. The larger the selected value, the lower the encoding rate. The factory default is 1.

  The lower the code rate, the higher the reliability of the communication, but the correspondingly lower effective data transmission rate. In the case of serious interference, it is recommended to reduce the encoding rate (that is, choose a larger value).

- [Frequency channel]: The center frequency of LoRa work. Factory default is 2450MHz

  The 2.4GHz frequency band is a global free frequency band, which is used by WiFi, Bluetooth, Zigbee and other devices. LoRa has strong immunity to these communications, but it may be interfered by these

communications devices in extreme cases. If this happens, users are advised to adjust the frequency channel of KW to avoid the frequency band used by other communications devices within the field range .

- [Transmit Power]: LoRa transmit power. The factory default is 16 (-2dBm).
- [Subpacket time]: refers to the maximum interval time between two frames of packets. If the KW module does not receive data again within this time, it will process the received data as a complete frame. The factory default is 1ms.。

  The longer the packetization time, the lower the communication frequency that can be achieved.

  In practical applications, wireless signal transmission may be unstable due to reasons such as long communication distance, many obstacles, and serious interference, which may cause KW to segment the message incorrectly. If this happens, it is recommended that the user increase the subcontract appropriately time.
- [Enable chip built-in CRC]: LoRa chip has built-in CRC function. If CRC is enabled, the sending end LoRa will perform CRC check on the data before sending each time, and append the check code to the message and send it out. When the receiving end LoRa receives the message, it will first perform a CRC check. If the check is correct, the message will be stored, otherwise the message will be discarded.

  The Kinco interconnection protocol and Modbus communication protocol provided by KW all have their own CRC check function, so the built-in CRC of the chip is disabled by default.

➢ **Performance estimation**

According to the communication parameters selected by the user, the software will automatically estimate the possible wireless communication performance and display the results here for the user's reference.

- [Link Budget]: If the current communication parameters are used, the link budget that LoRa can theoretically reach. The larger the budget value, the wider the wireless coverage.
- [Receiving Sensitivity]: If the current communication parameters are used, the theoretical receiving sensitivity of LoRa. The smaller the sensitivity value, the stronger the receiving ability, so the distance between the transmitting end and the receiving end can be farther.
- [Effective air rate]: If the current communication parameters are used, LoRa theoretically transmits the

user's effective data rate.

The effective rate is related to the air transmission rate and coding rate: the higher the air transmission rate, the higher the effective rate; the lower the coding rate, the more redundant information in the communication data, so the lower the effective rate.

- **Time estimate**

After selecting the communication parameters, the user can enter the [transmission data length] here and click the [time estimation] button, the software will automatically calculate the [air transmission time] required for these data from the sending end to the receiving end.

10.4.3.1.2 Read and write instructions using parameters

KW provides LORA_RPARAS (read LoRa parameters) and LORA_WPARAS (modify LoRa parameters) instructions. In the user project, these instructions can be called to realize online reading and modification of LoRa parameters. The parameter values in the instruction correspond to the parameter option values in the LoRa parameter configuration tool.

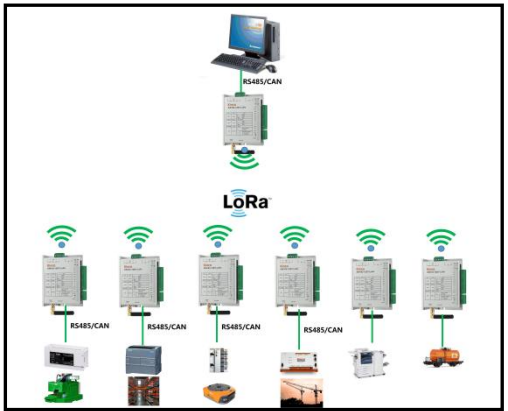
For a detailed description of the instruction, please refer to 10.4.5 LoRa function-related instructions.

10.4.4 Communication using LoRa

Before using LoRa communication, the user needs to configure the parameters of the LoRa communication port. In the same LoRa communication network, the following parameters of all nodes must be consistent: frequency, air transmission rate, encoding rate, and whether to enable the built-in CRC of the chip.

10.4.4.1 Build a LoRa communication network

LoRa is a half-duplex communication method, and receiving and sending cannot be performed at the same



time. In a LoRa network, only one node can be in the sending state at any one time. Therefore, it is recommended that users use LoRa to form a master-slave mode and star topology wireless communication network in practical applications: there is one and only one master station node in the network, and the rest of the nodes are all slave stations. The master station is responsible for scheduling, managing the network, and polling each slave station; the slave station can only respond after receiving the request from the master station. as the picture shows.

In practical applications, users can divide the modules in the same area into **different wireless networks** by **setting different communication frequency channels or air transmission rates**. For example, if there are 100 devices in a workshop, the communication frequency of 50 of them can be set to 2410MHz, and the frequency of the other 50 can be set to 2430MHz, then these devices can form two mutually independent wireless communication networks. It is better not to set the adjacent frequency bands into regular even numbers, but irregular odd numbers.

10.4.4.2 Communication using LoRa

The LoRa communication port supports programming protocol (master-slave mode), Kinco PLC interconnection protocol (master-slave mode) and free communication functions.

If the communication function is not used in the user program, the KW will support the programming protocol by default after power-on, and will also automatically act as a Kinco PLC interconnection slave station and a Modbus RTU slave station.

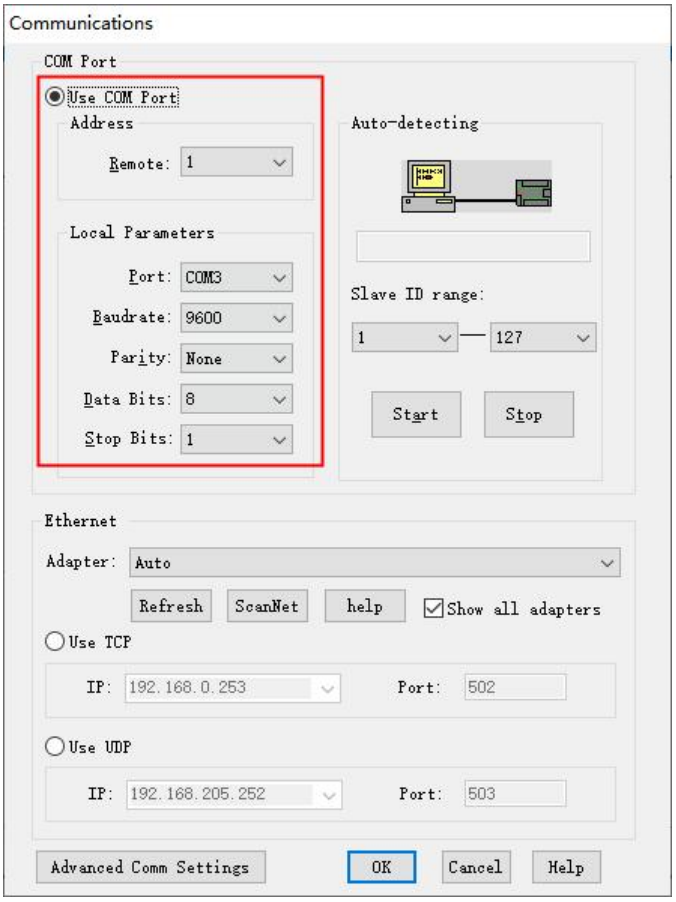
10.4.4.2.1 programming protocol

The LoRa communication port supports the programming protocol instructions in the KincoBuilder software (located in the [PLC] and [Debug] menus), including upload, download, online monitoring, forcing, etc., and also supports modifying the LoRa communication port parameters of all nodes in the network.

Kinco provides a dedicated debugging module to achieve the above functions. The method of use is as follows:

1) Connect a Kinco dedicated debugging module to the USB or RS232 port of the computer.

2) Open the [PC Communication Settings] window, select the station number of the [Target PLC] to be operated and the [COM port] used by the computer. The communication parameters of the RS232 port of the special debugging module are: baud rate 115200, no parity, 8 data bits, 1 stop bit. If the user uses the RS232 port, he needs to set the computer COM port parameters to the above parameters in . If using the USB port, no setting is required.
3) Configure the communication parameters of the LoRa port of the debugging module according to the method described above to ensure normal communication with the target PLC. The user does not need to care about the [Target PLC] station number when configuring parameters, and choosing any station number will not affect the configuration of the parameters of this module.
4) Finally, execute the required instructions in the KincoBuilder software.



Pay attention when using the wireless programming function: the LoRa communication port of the target PLC cannot use the free communication function; in the LoRa network where it is located, other nodes other than the debugging module are not allowed to actively send messages, that is, the communication of the master station in the network must be stopped!

10.4.4.2.2 free communication function

The so-called free communication means that the communication process and communication data are completely controlled by the user program. Users can use free communication methods to write various custom communication protocols to communicate with other devices.

When the free communication instruction in the user program is executed, the free communication mode is activated, and the corresponding communication port is completely occupied by free communication. When the free communication instruction is completed, the CPU automatically switches the communication port to the default protocol. If the PLC is in the STOP state, the free communication program is prohibited, and the PLC will restore the default programming protocol and Modbus RTU slave function.

For how to use the instruction, see 10.1 Free Communication.

10.4.5 LoRa function related instructions

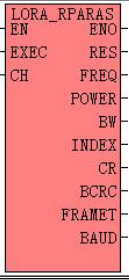
The instructions described in this section are located in the [Instruction Set] -> [Communication Instructions] group.

In the program written in IL language, the functions of all LoRa instructions are consistent with those in the LD program, and all follow the execution principle of the IL program: if the CR value is 1, the instruction is scanned and executed, and the execution result does not affect the CR value .

Therefore, only the instructions in the LD format will be described below, and no details will be given for the IL format.

10.4.5.1 LoRa port specific instructions

10.4.5.1.1 LORA_RPARAS (Read LoRa parameters)

	Name	Instructions format	Adopt to
LD	LORA_RPARAS		☑ KW203 ☑ KW213

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
CH	Input	INT	V、M、L、constant
RES	Output	BYTE	V、M、L
FREQ	Output	INT	V、M、L
POWER	Output	INT	V、M、L
BW	Output	INT	V、M、L
INDEX	Output	INT	V、M、L
CR	Output	INT	V、M、L
BCRC	Output	BOOL	V、M、L
FRAMET	Output	INT	V、M、L
BAUD	Output	REAL	V、L

The LoRa parameter value read by this instruction corresponds to each parameter option in the LoRa parameter configuration tool of the KincoBuilder software.

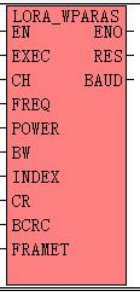
Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
CH	The LoRa port number of the parameter to be read. 0 means local port 1, 1 means use local port 2.
RES	The latest execution result. Its composition is as follows: Bit 7 ~ instruction status. This bit is set to 0 if the instruction is executing, and is set to 1 immediately when the instruction is completed. Bit 0 ~ illegal LoRa port. If the LoRa port number specified by CH is an illegal value, this bit will be set to 1. Other bits ~ reserved
FREQ	Read frequency channel option value.
POWER	Read transmit power option value.
BW	The value of the bandwidth option to read. This value corresponds to the BW option in the parameter configuration tool: 0 means BW203, 1 means BW406, 2 means BW812, 3 means BW1625, 4 means FLRC. 0-3 means working in LoRa mode, 4 means working in FLRC mode.
INDEX	Read the value of the over-the-air rate option. This value corresponds to the option in the "air transmission rate" list under the current BW option in the parameter configuration tool: 0 means the first item in the current list, 1 means the second item, and so on.
CR	The encoding rate value to read. This value corresponds to the option in the "Coding Rate" list in the parameter configuration tool. LoRa mode: The valid value range is 1-4. 1 means the first item in the list, and so on. FLRC mode: The valid value range is 0-2. 0 means the first item in the list, and so on.

BCRC	Read the option value of "Enable on-chip CRC".
FRAMET	The packet time option value to read.
BAUD	The over-the-air rate calculated from the read BW and INDEX option values.

When the EN value is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to execute once. RES is set to 0 when the instruction is executed. When the instruction is completed (regardless of success or failure), the 7th bit of RES will be set to 1 immediately. If the parameters are read successfully, KW will update the corresponding output parameters of this instruction according to the read LoRa parameter values, otherwise the relevant output parameter values will remain unchanged.

In the program, the user can judge whether the instruction is completed according to the rising edge of the 7th bit of RES, and then judge whether the reading is successful or not according to the error value represented by other bits.

10.4.5.1.2 LORA_WPARAS (Modify LoRa parameters)

	Name	Instructions format	Adopt to
LD	LORA_WPARAS		☑ KW203 ☑ KW213

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
CH	Input	INT	V、M、L、constant
FREQ	Input	INT	V、M、L
POWER	Input	INT	V、M、L
BW	Input	INT	V、M、L
INDEX	Input	INT	V、M、L
CR	Input	INT	V、M、L
BCRC	Input	BOOL	V、M、L

FRAMET	Input	INT	V、M、L
RES	Output	BYTE	V、M、L
BAUD	Output	REAL	V、L

The LoRa parameter input values in this instruction correspond to the parameter options in the LoRa parameter configuration tool of the KincoBuilder software.

Parameters	Description
EXEC	If the rising edge transition of EXEC is detected, the instruction is triggered to execute.
CH	LoRa port number to be modified. The meaning of the input value is as follows: 0 ～port 1 of this machine; 2 ～port 1 all nodes of the network; 1 ~ Port 2 of the local machine; 3 ~ Port 2 All nodes in the network.
FREQ	The new frequency channel option value.
POWER	New transmit power option value.
BW	New bandwidth option value. This value corresponds to the BW option in the parameter configuration tool: 0 means BW203, 1 means BW406, 2 means BW812, 3 means BW1625, 4 means FLRC. 0-3 means working in LoRa mode, 4 means working in FLRC mode.
INDEX	New over-the-air rate option value. This value corresponds to the option in the "air transmission rate" list under the current BW option in the parameter configuration tool: 0 means the first item in the current list, 1 means the second item, and so on.
CR	The new encoding rate value. This value corresponds to the option in the "Coding Rate" list in the parameter configuration tool. LoRa mode: The valid value range is 1-4. 1 means the first item in the list, and so on. FLRC mode: The valid value range is 0-2. 0 means the first item in the list, and so on.
BCRC	New "Enable on-chip CRC" option value.
FRAMET	New packet time option value.
RES	The latest execution result. Its composition is as follows: Bit 7 ~ instruction status. This bit is set to 0 if the instruction is executing, and is set to 1 immediately when the instruction is completed. No. 1 ~ modify the result. If the parameter modification fails, this bit is set to 1. Bit 0 ~ illegal LoRa port. If the LoRa port number specified by CH is an illegal value, this bit will be set to 1. Other bits ~ reserved
BAUD	Calculated over-the-air rate based on the entered BW and INDEX option values.

When the value of EN is 1, if the rising edge of the EXEC input terminal is detected, the instruction is triggered to execute once, and each communication parameter of the Lora port CH is updated with the new parameter value.

RES is set to 0 when the instruction is executed. When the instruction is completed (regardless of

success or failure), the 7th bit of RES will be set to 1 immediately. In the program, the user can judge whether the instruction is completed according to the rising edge of the 7th bit of RES, and then judge whether the modification is successful according to the error value indicated by other bits.

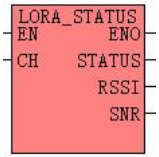
Users need to pay attention to the following points when using this instruction::

1）At the same time, only one LORA_WPARAS instruction is allowed to be executed!

2）Modifying parameters may affect the current normal data communication, please modify under the premise of ensuring safety!

3）If modifying the parameters of all nodes in the network, it is recommended that the user execute this instruction through the master station in the network, and it is necessary to ensure that all nodes can communicate normally! In addition, since LoRa is a half-duplex communication method, KW cannot accurately determine whether the parameters of each node have been successfully modified!

10.4.5.1.3 LORA_STATUS (Get LoRa signal quality)

	Name	Instructions format	Adopt to
LD	LORA_STATUS		☑ KW203

Parameters	Input/output	Data type	Memory area allowed
CH	Input	INT	V、M、L、constant
STATUS	Output	WORD	V、M、L
RSSI	Output	INT	V、M、AQW、L
SNR	Output	INT	V、M、AQW、L

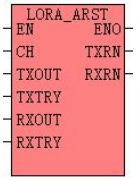
Parameters	Description
CH	The number of the LoRa port used. 0 means local port 1, 1 means use local port 2.
STATUS	The working status of the LoRa port. This parameter is reserved.

RSSI	The strength value of the received signal. The value is negative, and the closer to 0, the higher the signal strength.
SNR	The signal-to-noise ratio of the received signal. The larger the signal-to-noise ratio, the better the signal quality.

When the value of EN is 1, the instruction is executed. This instruction will get the signal strength RSSI and signal-to-noise ratio SNR detected when receiving data last time. Users can judge the quality of wireless communication according to these parameters. The larger the RSSI value, the higher the received signal strength, which means the smaller the attenuation of the signal during transmission; the larger the signal-to-noise ratio value, the better the signal quality and the less interference it receives.

10.4.5.1.4 Automatic reset LoRa communication port

- **LORA_ARST (Automatic reset due to transmit and receive timeouts)**

	Name	Instructions format	Adopt to
LD	LORA_ARST		☑ KW203 ☑ KW213

Parameters	Input/output	Data type	Memory area allowed
CH	Input	INT	V、M、L、constant
TXOUT	Input	DWORD	V、M、L、constant
TXTRY	Input	INT	V、M、L、constant
RXOUT	Input	DWORD	V、M、L、constant
RXTRY	Input	INT	V、M、L、constant
TXRN	Output	WORD	V、M、L
SNR	Output	WORD	V、M、L

Parameters	Description
CH	The number of the LoRa port used. 0 means local port 1, 1 means use local port 2.
TXOUT	Send timeout, in ms.
TXTRY	The allowed number of consecutive send timeouts. If the number of consecutive sending

	timeouts exceeds this value, the LoRa port will automatically reset.
RXOUT	Receive timeout, in ms.
RXTRY	The allowed number of consecutives receive timeouts. If the number of consecutives receive timeouts exceeds this value, the LoRa port will automatically reset.
TXRN	After this power-on, the number of resets of the LoRa port due to failure of sending timeout.
RXRN	After this power-on, the number of LoRa port resets caused by receiving timeout failures.

This instruction is used to specify the conditions for the automatic reset of the LoRa communication port. The rising edge of EN triggers the execution of this instruction once. After the instruction is executed, the PLC will store the reset conditions specified by the input parameters. In the future LoRa communication process, the PLC will automatically detect the timeout error of sending and receiving, and if the reset condition is met, the PLC will automatically reset the LoRa communication port. **Therefore, if the reset condition does not need to be adjusted multiple times, this instruction can be executed only once.**

The TXOUT and TXTRY parameters are used to specify the conditions for resetting caused by continuous sending overtime. When a transmission is started, the PLC starts to perform timeout detection. If no "send successfully" signal is detected within the TXOUT time, it is considered that a transmission has timed out. If the sending timeout occurs continuously, and the number of timeouts reaches the TXTRY value, the PLC will automatically reset the LoRa communication port, and the TXRN value will be increased by 1.

The RXOUT and RXTRY parameters are used to specify the condition that the continuous receiving timeout causes reset. When a reception is started, the PLC starts to perform timeout detection. If a message is received within the RXOUT time, it is considered as a reception timeout. If sending and receiving timeouts occur continuously, and the number of timeouts reaches the RXTRY value, the PLC will automatically reset the LoRa communication port, and the RXRN value will be increased by 1.

- **Automatic reset of continuous receiving message errors**

SMW26 is used to store the maximum allowable number of consecutive packet errors received by the LoRa communication port (local port 1). If the value is 0, the function is not enabled. If the value is greater than 0, the PLC will automatically detect the number of message errors (CRC check error, etc.). If there are continuous receiving message errors and the number of errors reaches the value of SMW26, the PLC will automatically reset the LoRa port.

SMW28 is used to record the number of resets of the LoRa port caused by continuous receiving message errors.

This reset condition has nothing to do with the LORA_ARST instruction and can be used alone.

- **Timing automatic reset**

SMW24 is used to store the timing reset period of the LoRa communication port (local port 1). If the value is 0, the function is not enabled. If the value is greater than 0, the PLC will automatically start a timer, and when the timer reaches the value of SMW24, the LoRa port will automatically reset once.

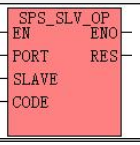
This reset condition has nothing to do with the LORA_ARST instruction and can be used alone.

10.4.5.2 Kinco Interconnection Protocol Dedicated Instructions

This group of instructions is only applicable to "Kinco PLC interconnection protocol".

In the [Communication Settings] page of the CPU in the [PLC Hardware Configuration] of the user project, the user can see the specific numbers of each communication port of the machine.

10.4.5.2.1 SPS_SLV_OP (The master pauses or restarts communication with the slave)

	Name	Instructions format	Adopt to
LD	SPS_SLV_OP		☑ KW203 ☑ KW213

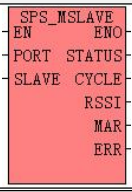
Parameters	Input/output	Data type	Memory area allowed
PORT	Input	INT	constant
SLAVE	Input	INT	V、M、L、constant
CODE	Input	BYTE	V、M、L、constant
RES	Output	BYTE	V、M、L

Parameters	Description
PORT	The communication port number used. 0 means PORT0, 1 means PORT1, 2 means PORT2, and so on. If the parameter value specifies a non-existing communication port, it is an illegal value, causing the instruction to report an error.

SLAVE	The station number of the target slave station. The slave station must already exist in the network, that is, it must be configured in the [Kinco PLC Interconnection Wizard].
CODE	Operation option, the meaning of its value is as follows: 1 ~ The master initiates communication with the target slave. 2 ~ The master stops communicating with the target slave.
RES	Execution result, the meaning of its value is as follows: 1 ~ Execution succeeded. 2 ~ The communication port used does not exist. 3 ~ The target slave station does not exist in the network. 4 ~ Illegal operation option.

When the value of EN is 1, the instruction is executed. For the Kinco Internet running on the PORT communication port, the user can call this instruction in the master station to suspend or restart the communication with the target slave station SLAVE.

10.4.5.2.2 SPS_MSLAVE (Read the communication status of the slave station)

	Name	Instructions format	Adopt to
LD	SPS_MSLAVE		☑ KW203

Parameters	Input/output	Data type	Memory area allowed
PORT	Input	INT	constant
SLAVE	Input	INT	V、M、L、constant
STATUS	Output	BYTE	V、M、L
CYCLE	Output	DWORD	V、M、L
RSSI	Output	INT	V、M、L
MAR	Output	WORD	V、M、L
ERR	Output	BYTE	V、M、L

Parameters	Description

PORT	The communication port number used. 0 means PORT0, 1 means PORT1, 2 means PORT2, and so on. If the parameter value specifies a non-existing communication port, it is an illegal value, causing the instruction to report an error.
SLAVE	The station number of the target slave station. The slave station must already exist in the network, that is, it must be configured in the [Kinco PLC Interconnection Wizard].
STATUS	The running status of the slave station, the meaning of its value is as follows: 1 ~ normal operation. 2 ~ The master failed to configure the slave. 3 ~ Offline (that is, no response message was received for the latest request). 4 ~The master station calls the SPS_SLV_OP instruction to suspend the communication with the slave station.
CYCLE	The real polling period of the master station to the slave station, unit: ms.
RSSI	The last time the master station received the signal strength of this slave station
MAR	The correct rate of the response message from the slave station, the output value is a 16-bit integer, meaning: correct rate × 100. Accuracy rate = the total number of correct response messages from the slave station ÷ the total number of request messages from the master station to the slave station
ERR	The most recent communication error that occurred in this slave station. 0 ~ no error. 1 ~ The slave station has timed out and has not responded. 2 ~ The length of the "write data" message is wrong. 3 ~ The length of the "read data" message is wrong. 4 ~ Wrong configuration message. 5~Wrong communication area configuration.

When the value of EN is 1, the instruction is executed. For the Kinco Internet running on the PORT communication port, the user can call this instruction in the master station to read the communication status of the slave station SLAVE.

In the [Communication Settings] page of the CPU in [PLC Hardware Configuration] in the project, the user can see the specific numbers of each communication port of the machine.

10.5 CAN Bus usage

10.5.1O Overview

Most series of KPLC products provide CAN bus communication ports, which support multiple communication protocols, making it convenient for users to connect various instruments, servo drives and other equipment. If there is only one CAN communication port, it will be named CAN1; if there are two

CAN communication ports, they will be named CAN1 and CAN2 respectively. The table below lists the CAN communication ports provided by each series of PLCs and the protocols they support:

Series		port provided	communication protocol					innating resistor (120Ω)
			Extended Protocol (3)	Free Communication (4)	Kinco Motion Control	CANOpen Master Station	CANOpen Slave Station	
K6 (1)(2)		AN2	--	Supported	Supported	Supported	--	t-in,Select with DIP switch
KS	105	CAN1	Supported	Supported	--	--	--	
	05C1	AN1	--	Supported	Supported	Supported	Supported	
	105C2	AN1	Supported	Supported	--	-	--	
	S101M	AN2	--	Supported	Supported	Supported	--	
KW		AN1	Supported	Supported	Supported	Supported	--	
MK		AN1	Supported	Supported	-	--	--	
K2	09M	N1	Supported	Supported	--	--	-	one
		N2	--	Supported	Supported	Supported	--	
	hers	--	--	--	--	--	--	
K5(2)		--	--	--	-	--	-	

Note: (1)The K6 version does not come standard with CAN2, users can use the KB6-CAN BD board to add this interface.

（2）K6/K5 series CPUs have dedicated expansion bus interfaces, which can be connected to various expansion modules of K6/K5. This expansion bus interface is not open to the outside world. For specific usage, please refer to the relevant descriptions in Appendix F Usage of Extension Modules.

（3）The expansion protocol here is a protocol customized by Kinco, which can be connected to KS series expansion modules. The KS series expansion modules adopt the RJ45 interface form, and users can use direct network cables to connect, so the expansion modules can be arranged in a distributed manner, and the total length from the CPU to the last expansion module is allowed to be up to 25 meters.。

（4）Free communication means that the user uses the sending and receiving instructions provided by KPLC to program, and the communication data format and communication process are completely controlled by the user program. Both CAN1 and CAN2 support free communication, and free communication can be used together with extended protocol and CANOpen master station protocol. Note: When used with other protocols at the same time, CAN free communication is relatively lower priority, and the CAN_INIT instruction is invalid, and the communication rate will adopt the rate of other protocols.

10.5.2 hardware wiring

The CAN bus interface usually uses the three pins CAN_H, CAN_L, and CAN_GND.

CAN_H, CAN_L are the signal data pins of the CAN bus. When in use, the user needs to connect the CAN_H and CAN_L pins of all nodes in the network together.

CAN_GND is the signal ground of the CAN bus, which provides a reference potential for the CAN bus. When in use, sometimes the CAN bus can work normally without connecting CAN_GND. However, it is recommended that users connect CAN_GND of all nodes in the network together, so that all nodes have a unified reference potential, thereby eliminating or reducing common-mode voltage and avoiding bus communication failures caused by common-mode interference.

When actually wiring the CAN bus, it is recommended to adopt a bus topology, and in order to reduce the signal reflection on the communication cable, it is recommended to add a 120Ω terminal resistance at the first and last ends of the bus. When the communication distance is long, the communication cable is recommended to use shielded twisted pair and the single end of the shielding layer is well grounded (control ground). And communication cables should be far away from strong interference sources, various high-power lines (including power cables of equipment), pulse signal lines with frequent switching, etc.

10.5.3 Extended bus functionality

10.5.3.1 Overview

Usually the CAN1 communication port of the CPU supports the expansion bus protocol, if the expansion module is configured in the [Hardware Configuration] of the user project, the CAN1 interface will work as the expansion bus interface.

In the factory default setting, the CPU module will automatically assign a unique ID to each expansion module and configure various parameters when it is powered on. Therefore, the CPU and all expansion modules need to be powered on at the same time or all expansion modules are powered on before the CPU

module, otherwise program execution errors may occur.

The KS series expansion modules adopt the RJ45 interface form, and the user can use a direct network cable to connect, so the KS expansion modules can be installed in a distributed manner, and the total length from the CPU to the last expansion module is allowed to be up to 25 meters. However, when KS expansion modules are installed in a distributed manner, each module may be installed on different devices, and the site cannot meet the above-mentioned default power-on sequence requirements. In response to this situation, KPLC provides the EX_ADDR instruction for the KS series expansion modules. Users can call this instruction to enable the CPU and KS expansion modules to be powered on or off in any order.

10.5.3.2 How to use the EX_ADDR instruction to realize the distributed application of the expansion module

In practical applications, if the expansion module is far away from the CPU module or installed on different devices, the default power-on sequence may not be guaranteed. In this case, the user can use the EX_ADDR instruction to modify the factory default configuration according to the following steps, so that each expansion module can be powered on or off at any time without causing PLC execution program errors.

1) In the user project, add each expansion module in the required order in [Hardware Configuration] and configure it according to actual needs, and call the EX_ADDR instruction (located in the [CAN instruction] group of the instruction set) in the program.
2) According to the order in [Hardware Configuration], connect the real CPU and all expansion modules, and then power on in the default order (CPU and all expansion modules are powered on at the same time or all expansion modules are powered on before the CPU module).
3) Download the user project to the CPU. After the CPU is running normally, modify the parameter value of the EX_ADDR instruction to 181 (decimal), and then execute the EX_ADDR instruction once. After the instruction is successfully executed, each expansion module will automatically save its own ID and various parameters (such as signal form, filtering method, etc.)
4) Power off the PLC system. Then the user can install the modules in the desired position, and **note that the connection order of each module (starting from the CPU) must still be consistent with the**

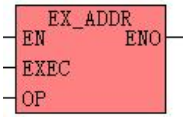
order in [Hardware Configuration]. Afterwards, when the expansion module is powered on again, it will automatically read the saved data and automatically enter the running state without CPU configuration, so it can be powered on or off at any time independently of the CPU.

5) If the user needs to restore the factory default power-on sequence, modify the parameter value of the EX_ADDR instruction in the program to 99 (decimal), and then let the EX_ADDR instruction execute once. After the instruction is successfully executed, each expansion module will clear the saved ID and channel parameters, and wait for the CPU to automatically assign ID and configure parameters when it is powered on again.

10.5.3.3 Extended bus instructions

The extended bus instructions are located in the [CAN instruction] group of the instruction set.

10.5.3.3.1 EX_ADDR (Modify the extension module configuration)

	Name	Instruction format	Adopt to
LD	EX_ADDR		☑ KS ☑ KW ☑ K209M ☑ MK

Parameters	Input/Output	Data type	Memory area allowed
EXEC	Input	BOOL	M、V、L、SM
OP	Input	INT	M、V、L、constant

Parameters	Description
EXEC	start side. The rising edge of EXEC triggers the execution of this instruction once, and it is necessary to ensure that EN is turned on before EXEC.
OP	Opcode (decimal). 181 --- Order all expansion modules to save their own ID and various parameters. 99 --- Instruction all expansion modules to clear saved IDs and various parameters.

This instruction will send the corresponding instruction to the connected expansion module according

to the OP value:

- If the OP value is 181, the expansion module will automatically save its own ID and various parameters (such as signal form, filter mode, etc.) after receiving the instruction. When the power is turned on again in the future, the expansion module will automatically read the saved data and enter the running state, without the need for CPU configuration, so it can be powered on or off at any time independently of the CPU.
- If the OP value is 99, the expansion module will clear the stored ID and channel parameters after receiving the instruction, and wait for the CPU to automatically assign the ID and configure the parameters when it is powered on again.

➢ **LD Format instruction description**

If EN is 1, the instruction is scanned, and if the rising edge of EXEC is detected, it will be executed once. If EN is 0, the instruction is not scanned and will not be executed.

10.5.4 CANOpen functions of master stations

10.5.4.1 Overview

The CANOpen bus has the advantages of good openness, high reliability, good real-time performance, strong anti-interference ability, and low cost. It is a commonly used field bus in industrial control and is currently being used more and more widely.

10.5.4.2 EDS Direction

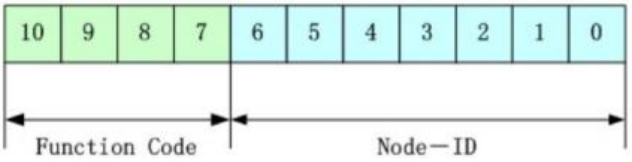
An EDS (Electronic Data Sheet) file is an identification file or similar for the slaves connected to the PLC. Through this file, you can identify the type of the slave station (which one is similar to 401, 402, 403, or which device belongs to 402). This file contains all the information of the slave station, such as the manufacturer, serial number, software version, supported baud rate type, OD that can be mapped, and the attributes of each OD. It is similar to the GSD file of Profibus. Therefore, before hardware configuration, we first need to import the EDS file of the slave station into the host configuration software.

10.5.4.3 Instruction of CANOpen communication objects

CANOpen Application Layer and Communication Specification (CiA DS301) is the core of the CANOpen protocol, applicable to all CANOpen devices. A variety of CANOpen communication objects are defined in DS301, and the services and protocols of these objects are also described in detail. In order to facilitate the user's application, we will introduce several key objects and their communication protocols below.

10.5.4.3.1 COB-ID Direction

COB-ID is a unique method of CANopen communication protocol, and its full name is Communication Object Identifier. These COB-IDs define the corresponding transmission levels for the PDO. With these transmission levels, the PLC programming software and the configuration of the slave station equipment define the same transmission level and the transmission content in it. In this way, after both the controller and the slave device adopt the same transmission level and transmission content, the data transmission is transparent. That is to say, both parties know the content of the data to be transmitted, and there is no need for the other party to reply whether the data transmission is successful when transmitting the data. The default ID allocation table is based on the 11-bit CAN-ID defined by CANOpen 2.0A (the COB-ID of the CANOpen 2.0B protocol is 27 bits). This consists of a 4-bit function code part and a 7-bit node ID (Node-ID) part, as shown in the figure.



Node-ID is defined by the system integrator, that is, the station number of the slave station, and the range of Node-ID is 1~127 (0 is not allowed to be used). Function Code is the function code of data transmission, which defines the transmission level of various PDO, SDO, and management messages. The smaller the function code, the higher the priority.

The figure below is the COB-ID allocation table of CANOpen predefined master/slave connection set:

CANopen predefines broadcast objects for master/slave connection			
object	Function code (ID-bits 10-7)	COB-ID	Index of communication parameters in OD
NMT Module Control	0000	000H	-
SYNC	0001	080H	1005H，1006H，1007H
TIME SSTAMP	0010	100H	1012H，1013H
Peer objects of CANopen master/slave connection set			
object	Function code (ID-bits 10-7)	COB-ID	Index of communication parameters in OD
urgent	0001	081H-0FFH	1024H，1015H
PDO1(send out)	0011	181H-1FFH	1800H
PDO1(receive)	0100	201H-27FH	1400H
PDO2 (send out)	0101	281H-2FFH	1801H
PDO2 (receive)	0110	301H-37FH	1401H
PDO3(send out)	0111	381H-3FFH	1802H
PDO3(receive)	1000	401H-47FH	1402H
PDO4(send out)	1001	481H-4FFH	1803H
PDO4(receive)	1010	501H-57FH	1403H
SDO (Sending/Server)	1011	581H-5FFH	1200H
SDO (Receiving/Customer)	1100	601H-67FH	1200H
NMT Error Control	1110	701H-77FH	1016H-1017H

Note: 1、The smaller the COB-ID, the higher the priority;

2、The function code in front of each COB-ID is in a fixed format;;

3、The COB-ID of the PDO in the figure is a value predefined by the CANOpen standard. But in actual application, users can also modify the COB-ID of PDO. Note: 00H, 80H, 100H, 701H-77FH, 081H-0FFH are occupied by the system, and users are not allowed to use these values.

10.5.4.3.2 Network management (NMT)

Network Management (NMT) is oriented to CANOpen devices and adopts a master-slave mode. NMT service can initialize, start, monitor, reset or stop CANOpen devices. There must be an NMT master station in a network, and the master station has the control right of the entire network, that is, the network

management (NMT) function. Several commonly used NMT services are introduced below.

10.5.4.3.2.1 NMT Node Control

The NMT master station controls the NMT status of each slave station (including stop, pre-operation, operation and initialization) through the NMT Node Control message. The slave device must support the NMT node control service. The format of the NMT node control message is as follows:

COB-ID	Byte 0	Byte 1
0x000	CS(Instruction Specifier)	Node ID

Among them, Node ID=the ID of the target slave station. If the Node ID is 0, it means that all slave stations on the network need to execute this instruction.

CS: instruction word, meanings of different values are:

1 means start the target node;

2 means stop the target node;

128 indicates that the target node enters the pre-operation state;

129 indicates reset target node

130 indicates that the target node resets the communication parameters

10.5.4.3.3 NMT Error Control

The error control service is used to detect network failures, including Node Guarding and Heartbeat. In practical applications, an error control method must be selected for a node.

By the way, the heartbeat service is newly added in the later version of DS301, and it is recommended by CiA.

- **NMT Node Guarding**

NMT master station sends remote frame (no data)::

COB-ID
0x700 + Node ID

The NMT slave station sends the following response message:

COB-ID	Byte 0
0x700 + Node ID	Bit7: Trigger bit, must be alternately set to "0" or "1" in each node guarding response. Bit0-6: The combined value indicates the status of the slave. Among them, 0 means Boot-up, 4 means STOPPED; 5 means Operational; 127 means Pre-Operational.

- **NMT Node Guarding**

If a node is configured as a heartbeat producer, it will periodically send heartbeat messages. One or more nodes in the network act as heartbeat consumers to process the heartbeat messages of each producer. Usually, the master acts as a heartbeat consumer and the other slaves act as heartbeat producers. The heartbeat message format is as follows:

COB-ID	Byte 0
0x700 + Node ID	The state value of this node. Among them, 0 means Boot-up, 4 means STOPPED; 5 means Operational; 127 means Pre-Operational.

10.5.4.3.4 SDO, Service Data Object

SDO communication is based on a "client-server" model.

By using index (index) and sub-index (sub-index), SDO enables a CANOpen device (as a client) to directly access objects in the object dictionary of other CANOpen devices (as a server). Usually, the master station acts as a client.

SDO has two transmission mechanisms: accelerated transmission, which transmits up to 4 bytes of data each time; segmented transmission, which allows segmented transmission of data exceeding 4 bytes. The following briefly introduces the packet format of the accelerated transmission mechanism SDO.

request message, Client -> Server:

COB-ID	Byte 0	Byte 1-2	Byte 3	Byte 4-7
0x600 + Node ID	SDO command word	object index	object sub index	data

response message, Server -> Client:

COB-ID	Byte 0	Byte 1-2	Byte 3	Byte 4-7
0x580 + Node ID	SDO command word	object index	object sub index	data

10.5.4.3.5 PDO, Process Data Object

PDO is used to transmit real-time data, and a PDO message contains up to 8 bytes of data.

PDO communication is based on the "producer-consumer" model. Distinguished by sending data or receiving data, PDO is divided into sending PDO (TPDO) and receiving PDO (RPDO). Producers support TPDO and consumers support RPDO.

There is no protocol regulation for PDO communication, and the content contained in a PDO message is pre-defined. During network configuration, the user defines the COB-ID of each PDO and the objects mapped therein, so both the producer and the consumer can know the content of the corresponding PDO to analyze the message.

Each PDO is described by communication parameters and mapping parameters in the object dictionary. The following describes the communication parameters of PDO.

- **COB-ID**

  Indicates the COB-ID used by this PDO.

- **Transfer type**

  Indicates the trigger mode of sending (or receiving) of this PDO. It is an 8-bit unsigned integer value. The transfer types are divided into the following categories:

  - Synchronization mode: trigger sending (or receiving) according to the count value of the SYNC object. The value of transmission type is 0 means "synchronous, acyclic" mode, and the value is 1-240 means "synchronous, cyclic" mode.
  - RTR-Only: Only applicable to TPDO, the sending of PDO is triggered by the received RTR message. A value of 252 for the transmission type means that a PDO is sent after SYNC and RTR are received. A value of 253 means that the PDO is sent immediately after the RTR is received.
  - Event-driven: PDO is sent immediately after the internal event of the CANOpen device occurs. The value of the transmission type is 254, indicating that the event is defined by the device manufacturer. A value of 255 means that it is an event defined by the device sub-protocol and the

application layer protocol, which generally refers to the change of the data value in the PDO or the expiration of the timer.

- **Prohibition time**

The prohibit time defines the minimum interval time when the PDO is sent continuously. The purpose of configuring the prohibition time is to avoid the problem that the high-priority PDOs are sent too frequently and always occupy the bus, so that other lower-priority messages cannot use the bus.

- **Event timer**

It is used to specify a period value for timing sending. It is a 16-bit unsigned integer in ms. PDO will trigger sending with this timing value as a period. If the value is 0, it means that the event timer is not used.

10.5.4.4 Using the CANOpen master function

The CANOpen master station function of KPLC has the following characteristics:

- Adopt CAN2.0A standard. Comply with CANOpen standard protocol DS301 V4.2.0.
- Support NMT network management services, including NMT Node Control and NMT Error Control, and act as the NMT master station.
- The KS series supports a maximum of 8 CANOpen slave stations, and the K6 series supports a maximum of 64 CANOpen slave stations.
- Allow users to configure the startup process for each slave in KincoBuilder;
- Each slave supports up to 8 TPDOs and 8 RPDOs; a total of up to 128 TPDOs and 128 RPDOs are supported.
- The PDO transmission type only supports 254 or 255 asynchronous transmission mode (event-driven mode)
- Support client SDO, and provide SDO read and write instructions, SDO instructions support standard accelerated transmission mode;

10.5.4.4.1 CANOpen network configuration tool

In KincoBuilder, enter [PLC Hardware Configuration], select the CPU module in the table in the upper part of the window, and then click [CANOpen Master Station] in the page in the lower part of the window to enter the network configuration page of CANOpen.

10.5.4.4.2 Process EDS files

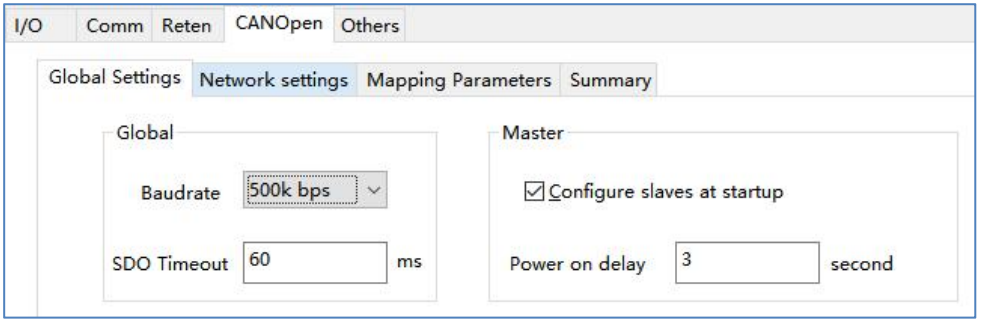
On the [CANOpen Master Station] -> [Network Configuration] page, the following buttons are provided to operate the EDS file:

- [Import EDS]: Click this button, select the desired EDS file, and then import it into KincoBuilder and store it. After importing an EDS, the corresponding slave device will be displayed in [All slave module list], and then configuration can be performed.
- [Delete]: Select a slave device in the [All Slave Module List] below, and then click the [Delete] button to delete the device from the list, and also remove its EDS file from KincoBuilder delete.
- [Export all EDS]: All existing slave EDS files in KincoBuilder can be merged and exported into one file (extension name is .ALLEDS). This function will be more useful when uninstalling KincoBuilder. Users can use this function to back up the EDS files of all slave stations before uninstalling, and then they can directly import the backed up .ALLEDS files.
- [Import all EDS]: You can import an EDS backup file (extension .ALLEDS) into KincoBuilder, after importing, all slave devices contained in the file will be displayed in the [All slave module list] below .

10.5.4.4.3 CANOpen network configuration process

1) Configure overall parameters

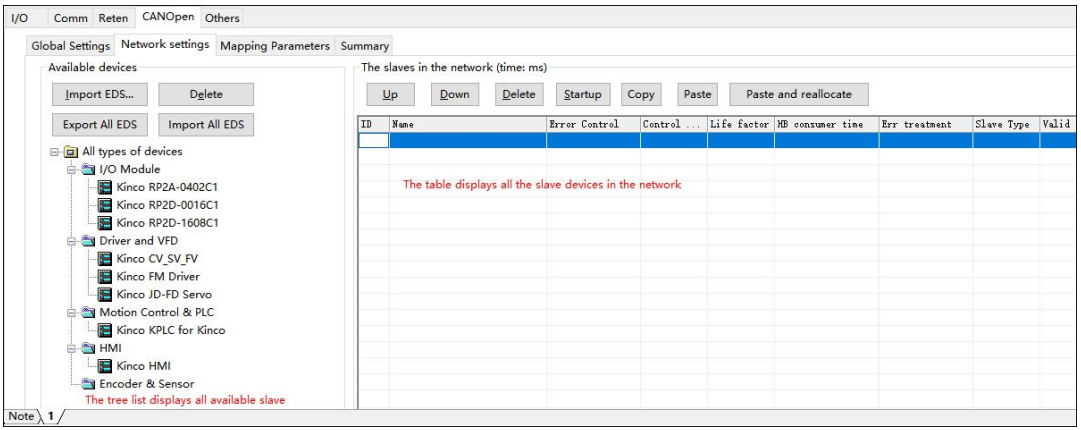
Enter the [master station and overall configuration] page, as shown in the figure below:



- [Baud rate]: Select the baud rate used by the master station. Note that the baud rate of all nodes on the network must be the same.
- [SDO timeout]: Set the timeout waiting time after the master station sends the SDO request message. If the response message from the corresponding slave station is not received beyond this time, an error will be reported. The SDO timeout value setting generally does not need to exceed 100ms.
- [Configure each slave station at startup]:: If this item is selected, in addition to controlling the NMT state transition of each slave station, the master station will also send corresponding configuration instructions to configure each slave station according to the parameter configuration of each slave station during startup(such as the error control method of the slave station, PDO mapping, etc.). If this item is not selected, the master station only controls the NMT state transition of each slave station.

2) Configure each slave station

Enter the [Network Configuration] page, and continue to configure the slave nodes and their parameters on the network, as shown in the figure below:



All function buttons on the page have corresponding right-click menu instructions. When the user clicks the right mouse button at the relevant position, the corresponding right-click menu will pop up, and menu instructions can be used at this time. The general procedure for configuring a slave is described below.

a) Add a slave device to the network

Double-click the type of slave station that needs to join the network from the tree list on the left, and a slave station device of this type will be added to the network and displayed in the table on the right.

b) Configure parameters such as the station number (ID) and supervision type of the slave device:

The [Address] column in the table on the right is the station number (ID) of the slave station. Line 1 is the location of station number 1.

After adding a slave device, its default configuration parameters are displayed. When adding, KincoBuilder defaults to adding devices to the table from top to bottom. The user can click the row in the table to select a slave station, and then click the [Move Up] and [Move Down] buttons to adjust its station number. You can also click the [Delete] button to remove this device from the network delete.

[Supervisory type] It is used to configure the NMT Error Control method of the node, including node protection and heartbeat. If the slave device supports these two methods at the same time, it is recommended to use the heartbeat method first.

[Supervision Time] Indicates the cycle value of the previously selected node protection mode or heartbeat mode. It is recommended that in practical applications, the period value should not be set too small,

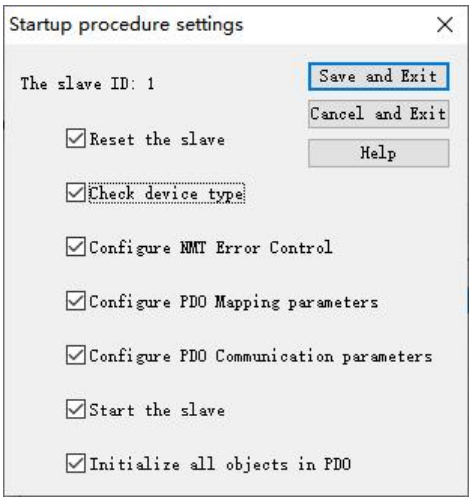
for example, it can be set above 2000.

[Heartbeat consumer time] The master station will regularly check whether it has received the heartbeat message from the slave station. If it still does not receive it after this "heartbeat consumer" time, it will consider the slave station to be offline and perform corresponding troubleshooting. It is recommended that in practical applications, the period value should not be set too small, for example, it can be set above 2000.

[Troubleshooting] It is used to select the processing method adopted when the master station detects the failure of the slave station, including three options: "none", "stop node" and "stop network". The faults that the master station can detect include SDO instruction timeout without response, node protection or heartbeat message timeout, receiving some types of emergency messages sent by the slave station, etc.

c) Configure the startup process of the slave station:

Click a slave station in the table with the mouse, and then click [Startup Process] to select what configuration the master station needs to perform on the slave station during the network startup process.



[Reset Node]: Whether the master station should send the "reset node" instruction before sending the configuration instruction to the slave station.

[Check device type]: Whether the master reads the device information for checking before sending configuration instructions to the slave.

[Configure node supervision mode]: Whether the master station needs to configure the supervision type and

parameters of the slave station.

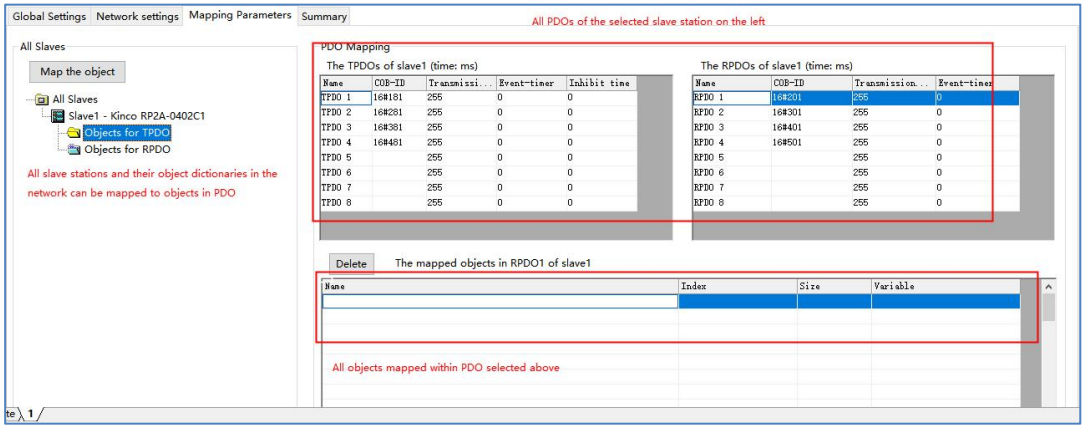
[Configure PDO mapping parameters]: Whether the master station needs to configure the PDO mapping parameters of the slave station.

[Configure PDO communication parameters]: Whether the master station needs to configure the PDO communication parameters of the slave station.

[Start the node]: After the configuration is completed, whether the master station needs to send the "start node" instruction to the slave station.

[Initialize the PDO data of the slave station]: After starting the slave station, whether the master station needs to clear all the data in the RPDO of the slave station and send it once immediately.

d) Configure the PDO of each slave station:



Enter the [Object Dictionary Mapping] page to configure PDO for all slave stations in the network.

The [Added Slave Station List] on the left side of the page shows all the slave stations that have joined the network, and the objects that are allowed to be mapped to PDO in the object dictionary of each slave station. Among them, the objects in [Send PDO] can only be mapped to the TPDO of the slave station, and the objects in the [Receive PDO] list can only be mapped to the RPDO of the slave station.

Click a slave station in [Added slave station list], then all PDOs of the slave station will be displayed on the right side, and the user can configure each PDO:

- Communication parameters

In the table on the right, select a PDO to modify its communication parameters such as timing time and prohibition time.

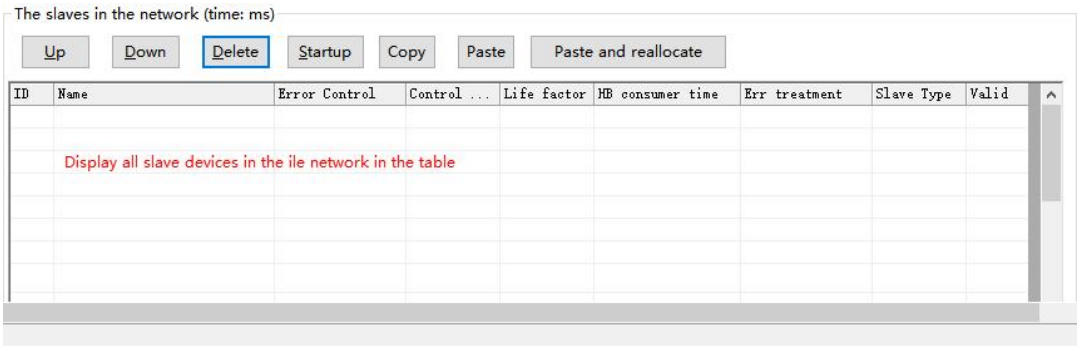
Among them, the COB-IDs of the first 4 TPDOs and RPDOs in the slave station are not allowed to be modified, and the default values in the predefined connection set in DS301 are adopted. The latter 4 TPDOs and RPDOs allow users to input valid COB-ID values by themselves.

- Mapping parameters

In the object list on the left, double-click an object, and this object will be added to the current PDO, and KincoBuilder will automatically assign a PLC V area address to this object, such as VW1006. The user operates the V area address in the program, which is equivalent to operating the corresponding object.

3)copy slave station, paste slave station

On the [Network Configuration] page, there are three buttons: [Copy Slave], [Paste Slave] and [Paste Slave (re-divide memory)]. This function is often used to replace the PLC model. Copy the configuration of the old model in advance and paste it to the new model, as shown in the figure below.



[Copy slave station]:Select a configured slave station and click this button to copy all the information of the slave station (including all PDO communication parameters, mapping parameters, etc.). If the selected slave does not have any PDOs configured, the copy fails with a corresponding message.

[Paste slave station]: After successfully copying a slave station, click to select a blank row in the table, and then click this button, the slave station information just copied will be pasted to this row and a new slave

station will be generated. Note: The PLC memory address corresponding to each mapping object in the PDO of the new slave station is still consistent with that in the source slave station. There is no reallocation, and the user needs to modify it by himself.

[Paste Slave (re-divide memory)]: The operation method is the same as [Paste Slave], but the difference is that the PLC memory address of each mapping object in the PDO of the new slave station will be automatically allocated without modification by the user.

10.5.4.5 CANOpen related instructions

10.5.4.5.1 CANOpen function and related instructions

The functions supported by CANOpen mainly include:

· Support NMT management message;

Support CANOpen predefined connection set mode, the total number of PDOs is 128TxPDO and 128RxPDO, and the configurable number of PDOs for each slave station is 1-8TxPDO, 1-8RxPDO;

8 (KS series) or 64 (K6 series) slave stations can be connected, and the slave station number 1-126 can be set arbitrarily;

· Support emergency message, node protection, heartbeat message;

· It is possible to set the startup process of the slave station;

· Support EDS import function;

Realize the operation of SDO through SDO_WRITE and SDO_READ instruction words;

· Support 254, 255 two PDO transmission modes, and has the function of sending PDO at regular intervals, can send 8 PDO data at the same time, and the timing time can be set;

· Support various baud rate communication: 10K/20K/50K/125K/250K/500K/800K/1M;

· Check the status of the master and slave stations on the network through the system word;

· With multi-PLC networking capability through CAN bus;

· It has the error handling function of the slave station. When the slave station makes an error, there are three processing methods that the master station can choose, namely: stop the node, stop the network, and

none.

SDO Instruction:

SDO (Service Data Object) is a communication object (message) defined by CANOpen, which is mainly used to transmit low-priority data between devices. Typically, it is used to configure and manage slave devices, such as modifying the PID parameters of the servo current loop, speed loop, and position loop, and PDO configuration parameters. This kind of data transmission is the same as the Modbus method, that is, after the master station sends it, the slave station needs to return a data response. This kind of data is only suitable for parameter setting, not suitable for data transmission that requires high real-time performance. For the specific message format of SDO, please refer to 10.5.4 CANOpen Master Station Function.

Other instructions:

CO_STATE (obtain the status and error of the slave station): CANOpen master station will detect and record the status, faults and emergency messages sent by all slave stations during operation.

CO_RESTART (reconfigure and start the slave station): During the normal operation of the CANOpen master station, the user can call this instruction to reconfigure and start the target slave station.

10.5.4.5.2 SDO Instruction

The SDO instruction is located in the [CAN instruction] group of the instruction set.

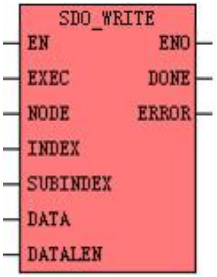
When using Kinco motion control function or CANOpen master station function, SDO instruction can be used.

In one user project, up to 64 SDO instructions are allowed.

10.5.4.5.2.1 SDO_WRITE (SDO write operation)

- Instruction and its operand description

	Name	Instruction format	Cr values affected	☑ KS

LD	SDO_WRITE			☑ KW ☑ K209M ☑ K6
IL	SDO_WRITE	SDO_WRITE *EXEC, NODE, INDEX, SUBINDEX, DATA, DATALEN, DONE, ERROR*	U	

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
NODE	Input	BYTE	I、Q、V、M、L、SM、constant
INDEX	Input	WORD	I、Q、V、M、L、SM、constant
SUBINDEX	Input	BYTE	I、Q、V、M、L、SM、constant
DATA	Input	BYTE	I、Q、V、M、L、SM
DATALEN	Input	BYTE	I、Q、V、M、L、SM、constant
DONE	Output	BOOL	Q、M、V、L、SM
ERROR	Output	DWORD	Q、M、V、L、SM

Note: NODE, INDEX, SUBINDEX, and DATALEN must be constant or variable at the same time; DATA and DATALEN parameters together form a variable-length memory block, and the memory block must all be legal memory addresses.

For the specific usage instructions of each parameter, see the following table:

Parameters	Functions
EN	enable terminal. If EN is 1, the instruction is enabled and allowed to execute.
EXEC	start side. The rising edge of EXEC is used to start SDO communication. It is best to ensure that EN is turned on before EXEC.
NODEID	The address of the node to be visited
Index	Index of the object to be accessed in OD
SubIndex	The sub-index of the object to be accessed in OD
Data	The starting byte of the data to be sent

DataLen	The length of the data to be sent, unit: byte, range 1, 2, 4, related to the object to be accessed. If the data type of the object to be accessed is byte, the length is 1, the data type is word length 2, and the data type is double word length 4
DONE	Execution result indication. If SDO is executing, DONE is 0; if SDO communication ends (received response or timeout), DONE is 1.
ERROR	error message. see table below

SDO error message description::

Error code	Description
0	No errors.
1	Master is not enabled
2	target node does not exist
3	Input parameter value error (such as data length)
4	The target node's last instruction has not been responded to
5	The send or receive buffer of the PLC is full
6	The instruction timed out and no response message was received
7	Wrong response received (not expected response, wrong length, etc.)
8	Termination message received
9	In this project, the number of SDO instructions exceeds the limit
10	BD board without KB6-CAN installed

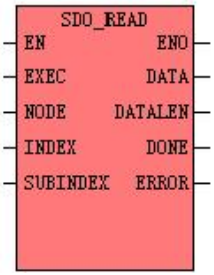
· LD

If EN is 1, the instruction is executed, otherwise it is not executed.

10.5.4.5.2.2 SDO_READ (SDO read operation)

- Instruction and its operand description

	Name	Instruction format	Cr values affected	☑ KS

LD	SDO_READ			☑ KW ☑ K209M ☑ K6
IL	SDO_READ	SDO_READ *EXEC, NODE, INDEX, SUBINDEX, DATA, DATALEN, DONE, ERROR*	U	

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM、RS、SR
NODE	Input	BYTE	I、Q、V、M、L、SM、constant
INDEX	Input	WORD	I、Q、V、M、L、SM、constant
SUBINDEX	Input	BYTE	I、Q、V、M、L、SM、constant
DATA	Output	BYTE	I、Q、V、M、L、SM
DATALEN	Output	BYTE	I、Q、V、M、L、SM
DONE	Output	BOOL	Q、M、V、L、SM
ERROR	Output	DWORD	Q、M、V、L、SM

Note: NODE, INDEX and SUBINDEX must be constant or variable at the same time; DATA and DATALEN parameters together form a variable-length memory block, which must be legal memory addresses.

For the specific usage instructions of each parameter, see the following table:

Parameters	Functions
EN	enable terminal. If EN is 1, the instruction is enabled and allowed to execute.
EXEC	start side. The rising edge of EXEC is used to start SDO communication. It is best to ensure that EN is turned on before EXEC.
NODEID	The address of the node to be visited
Index	Index of the object to be accessed in OD
SubIndex	The sub-index of the object to be accessed in OD
Data	The start byte of received data storage

DataLen	The length of the received data, unit: byte, range 1, 2, 4, related to the object to be accessed, if it is 4, it means that the read data occupies the length of 4 bytes starting from the start byte stored in Data .
DONE	Execution result indication. If SDO is executing, DONE is 0; if SDO communication ends (received response or timeout), DONE is 1.
ERROR	error message. see table below

SDO error message description:

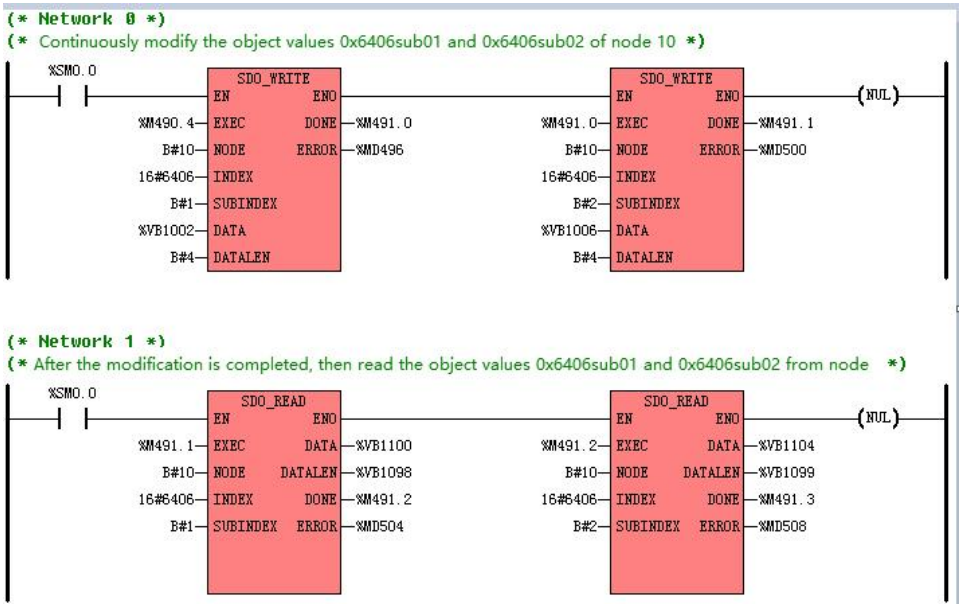
Error code	Description
0	No errors.
1	Master is not enabled
2	target node does not exist
3	Input parameter value error (such as data length)
4	The target node's last instruction has not been responded to
5	The send or receive buffer of the PLC is full
6	The instruction timed out and no response message was received
7	Wrong response received (not expected response, wrong length, etc.)
8	Termination message received
9	In this project, the number of SDO instructions exceeds the limit
10	No BD board coded KB6-CAN installed

· LD

If EN is 1, the instruction is executed, otherwise it is not executed.

10.5.4.5.2.3 SDO_WRITE、SDO_READ example

LD

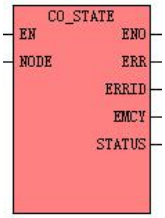


IL

```
(* Network 0 *)
(*Continuously modify the object values 0x6406sub01 and 0x6406sub02 of node 10 *)
LD      %SM0.0
SDO_WRITE %M490.4, B#10, 16#6406, B#1, %VB1002, B#4, %M491.0, %MD496
SDO_WRITE %M491.0, B#10, 16#6406, B#2, %VB1006, B#4, %M491.1, %MD500
(* Network 1 *)
(*After the modification is completed, then read the object values 0x6406sub01 and 0x6406sub02 from node 10 *)
LD      %SM0.0
SDO_READ %M491.1, B#10, 16#6406, B#1, %VB1100, %VB1098, %M491.2, %MD504
SDO_READ %M491.2, B#10, 16#6406, B#2, %VB1104, %VB1099, %M491.3, %MD508
```

10.5.4.5.3 CO_STATE (Get the status and errors of the slave station)

- Instruction and its operand description

	Name	Instruction format	Cr values affected	
LD	CO_STATE			☑ KS ☑ KW ☑ K209M ☑ K6
IL	CO_STATE	CO_STATE*NODE, ERROR, ERRID, EMCY, STATUS*	U	

Parameters	Input/output	Data type	Memory area allowed
NODE	Input	INT	V、M、L、constant
ERR	Output	BOOL	Q、M、V、L
ERRID	Output	BYTE	V、M、L
EMCY	Output	WORD	V、M、L
STATUS	Output	BYTE	V、M、L

For the specific usage instructions of each parameter, see the following table:

Parameters	Functions
EN	enable terminal. If EN is 1, the instruction is enabled and allowed to execute.
NODE	The address of the node to be visited
ERR	Indicates whether the target slave has failed. 1 means that the slave station has failed once, and 0 means that no failure has occurred.
ERRID	Indicates the fault code of the slave failure. See the following list for specific meanings.
EMCY	Indicates the fault code in the emergency message sent by the slave station. Please refer to the DS301 standard for the specific meaning of the fault code.
STATUS	Indicates the current status of the slave. The status value comes from the heartbeat of the slave station or the node guarding message. For the meaning of status value, please refer to 10.5.3.3.3 Error Control.

For the description of the error code (ERRID), see the table below:

Error code	Description

0	No errors.
1	The slave station responded to the SDO request message with a termination message (instruction code 0x80).
2	The response of the slave station to the SDO request message is wrong (such as message length error, object error, etc.).
3	This slave station has sent an emergency message.
4	The slave's heartbeat timeout or node guard timeout.
5	The slave does not respond to the SDO request message.
6	This slave station is in STOP state due to failure of other nodes. Note: If the user sets the "fault handling" mode of a slave station to "stop network" in the hardware configuration, then when the master station detects that the slave station is faulty, it will send a STOP instruction to make all slave stations in the network enter the STOP state.

The CANOpen master station will detect and record the status, faults and emergency messages sent by all slave stations during operation. If EN is 1, this instruction is executed, and the record value of the target slave station (station number NODE) will be read and output to the corresponding parameters (ERRID, EMCY, STATUS).

· LD

If EN is 1, the instruction is executed, otherwise it is not executed.

· IL

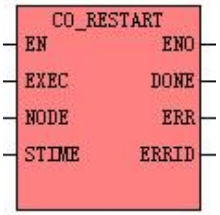
If the CR value is 1, the instruction is executed, otherwise it is not executed.

The execution of this instruction does not affect the CR value.

10.5.4.5.4 CO_RESTART (Reconfigure and start the slave)

- Instruction and its operand description

	Name	Instruction format	Cr values affected	☑ KS

LD	CO_RESTART			☑ KW ☑ K209M ☑ K6
IL	CO_RESTART	CO_RESTART*EXEC,NODE, STIME, DONE, ERR, ERRID*	U	

Parameters	Input/output	Data type	Memory area allowed
EXEC	Input	BOOL	I、Q、V、M、L、SM
NODE	Input	INT	V、M、L、constant
STIME	Input	INT	V、M、L、constant
DONE	Output	BOOL	Q、M、V、L
ERR	Output	BOOL	Q、M、V、L
ERRID	Output	BYTE	V、M、L

Note: The NODE and STIME parameters must be both constant or both variables. In a user project, a maximum of 32 CO_Restart instructions are allowed.

For the specific usage instructions of each parameter, see the following table:

Parameters	Functions
EN	enable terminal. If EN is 1, the instruction is enabled and allowed to execute.
EXEC	start side. EN must be turned on before EXEC. The rising edge of EXEC will start this instruction and execute a process of restarting the slave station. The falling edge of EXEC will stop the execution of this instruction and terminate the process of restarting the slave station.
NODE	The address of the node to be visited
STIME	The maximum time for the master station to wait for the response from the slave station after sending the SDO request message.
DONE	Instruction execution result flag bit. If the instruction is being executed, the output of DONE is 0. DONE immediately becomes 1 if the instruction execution is complete (whether it succeeds or fails).
ERR	Instruction execution error flag. Set to 1 if an error occurs during instruction execution.

ERRID	The error code of instruction execution. If an error occurs during instruction execution, ERRID is the specific error message.

For the description of the error code (ERRID), see the table below::

Error code	Description
0	No errors.
1	The station number of the target slave station is wrong.
2	The target slave is executing the process of restarting the slave, and it is forbidden to execute it again until it is completed.
3	The CAN message sent by the master station is wrong (maybe due to the line, hardware interface, software buffer being full, etc.).
4	The SDO request message sent by the master station, the target slave station did not respond after timeout.
5	The SDO request message sent by the master station, the target slave station responds with an error.
6	The user actively stops the execution of this instruction (set EXEC to 0).

· LD

If EN is 1, the instruction is executed, otherwise it is not executed.

· IL

If the CR value is 1, the instruction is executed, otherwise it is not executed.

The execution of this instruction does not affect the CR value.

During the normal operation of the master station, the user can call this instruction to reconfigure and start the target slave station (the station number is NODE). The various parameters of the slave station will adopt the parameters that have been configured for the slave station in [Hardware Configuration] -> [CANOpen Master Station], including the startup process, error monitoring parameters, PDO communication and mapping parameters, etc.

Note: If the user selects the "failure handling" mode of a slave station as "stop network" in [hardware configuration], then this instruction may not be executed normally when the slave station fails!

If EN is 1, then the rising edge of EXEC will trigger the execution of this instruction. First, the output of DONE is 0, and then the master station will read the configured slave station parameters in [Hardware Configuration], and perform the following operations on the target slave station in sequence :

1)Send a instruction to let the target slave enter the pre-operational state

2)According to the user's configuration of the slave station [startup process] in the hardware configuration, the master station will continue to operate. If "reset node" is selected, the master station will send a "reset communication" instruction to the slave station.

- If "Configure node supervision mode" is selected, the master station will send SDO to configure the heartbeat or node guard parameters of the target slave station.
- If "Configure PDO Mapping Parameters" is selected, the master station will send SDO to configure the mapping parameters of all PDOs of the target slave station.
- If "Configure PDO Communication Parameters" is selected, the master station will send SDO to configure the communication parameters of all PDOs of the target slave station.
- If "Start this node" is selected, then after the above process is completed, the master station will send the "Start node" instruction to the target slave station. Note: Regardless of configuration or not, this instruction will not send "PDO initialization data" to the target slave station.

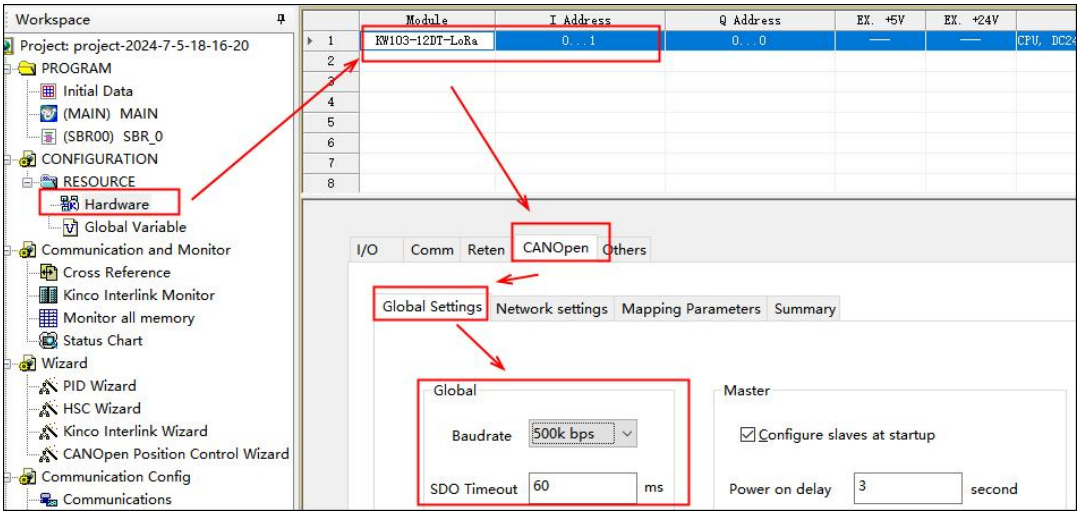
When the above process is successfully executed, this instruction will exit, and immediately output DONE as 1, and both ERR and ERRID as 0.

If any error occurs during the execution of the instruction, or EXEC becomes 0, the instruction will also exit, and DONE and ERR will be output as 1 immediately, and ERRID will output the corresponding error code. In addition, for specific errors (including SDO timeout no response, SDO response error), this instruction will be processed according to the "fault handling" method selected by the user in [Hardware Configuration]: if "None" is selected, no processing will be performed; If "Stop Node" is selected, the "Stop Node" instruction will be sent to the target node; if "Stop Network" is selected, the "Stop Node" instruction will be sent to all nodes in the network.

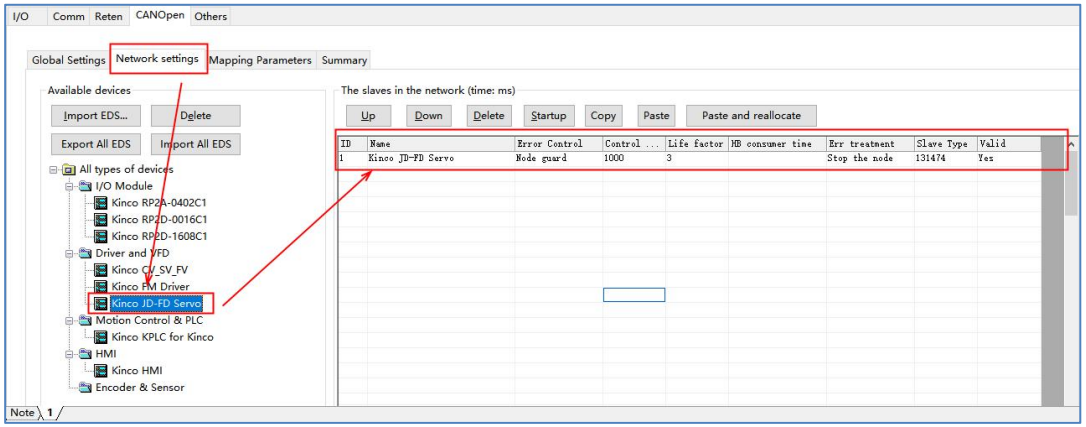
10.5.5 CANOpen Example

The following takes KS105C2-16DT with Kinco servo as an example to illustrate the entire configuration process and application:

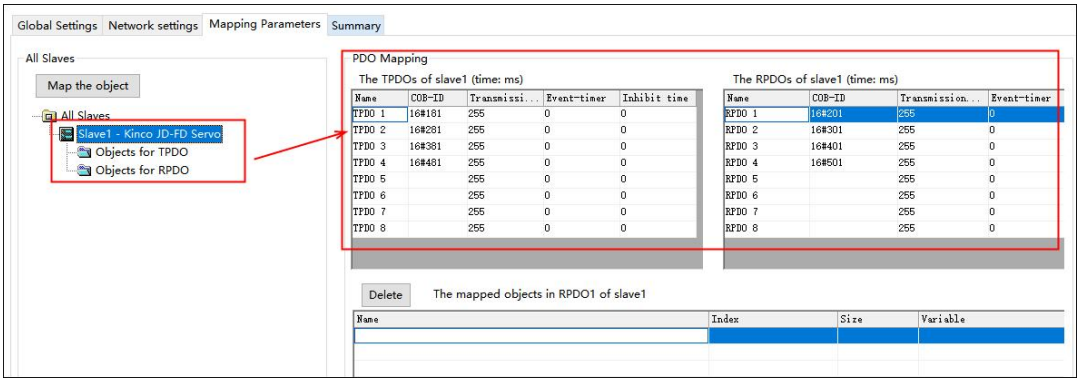
1、Set the communication parameters of the CANOpen master station as shown in the figure (the baud rate of the master and slave stations needs to be consistent).



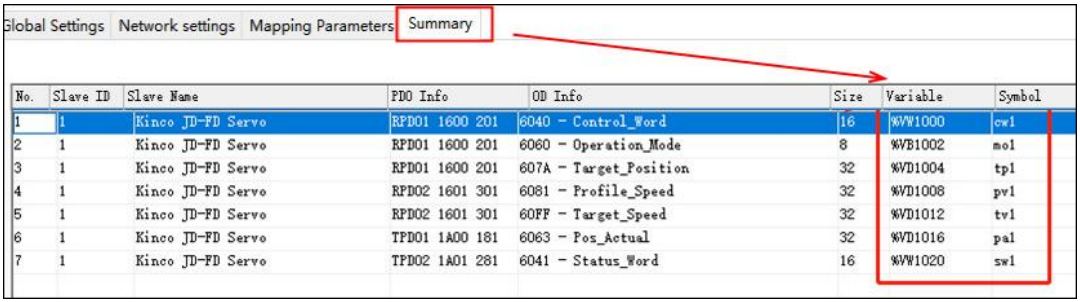
2、Set network configuration, add CANOpen slave device, and set supervision type and fault handling method at the same time.



3、For object dictionary mapping, the objects that need to be configured into the corresponding sending PDO and receiving PDO, the objects that change in real time such as actual position and actual current need to be configured with corresponding prohibition time, otherwise the CAN bus load will be too large and communication failure will occur.



4、The configured address list, the PLC memory address corresponding to the servo object is shown on the right, and the corresponding global variable name can be named for programming.

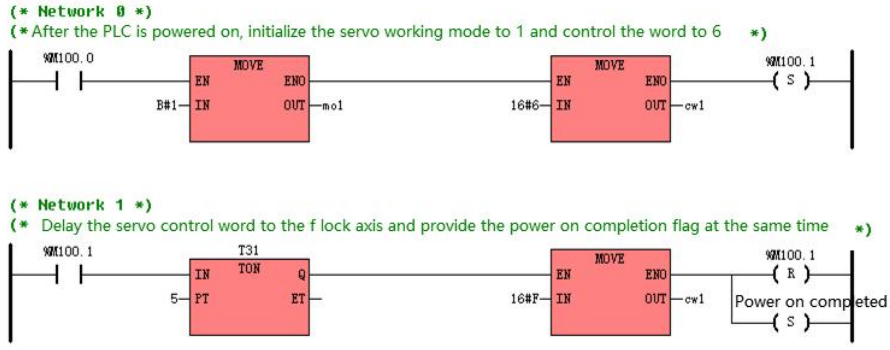


No.	Slave ID	Slave Name	PDO Info	OD Info	Size	Variable	Symbol
1	1	Kinco JD-FD Servo	RPD01 1600 201	6040 - Control_Word	16	%VW1000	cw1
2	1	Kinco JD-FD Servo	RPD01 1600 201	6060 - Operation_Mode	8	%VB1002	mo1
3	1	Kinco JD-FD Servo	RPD01 1600 201	607A - Target_Position	32	%VD1004	tp1
4	1	Kinco JD-FD Servo	RPD02 1601 301	6081 - Profile_Speed	32	%VD1008	pv1
5	1	Kinco JD-FD Servo	RPD02 1601 301	60FF - Target_Speed	32	%VD1012	tv1
6	1	Kinco JD-FD Servo	TPD01 1A00 181	6063 - Pos_Actual	32	%VD1016	pa1
7	1	Kinco JD-FD Servo	TPD02 1A01 281	6041 - Status_Word	16	%VW1020	sw1

5、Write the program to control the servo, including power-on initialization, origin return, speed control, absolute positioning, display part subroutine and MAIN main program. The main function of the main program is to call the subroutine. The procedure is as follows:

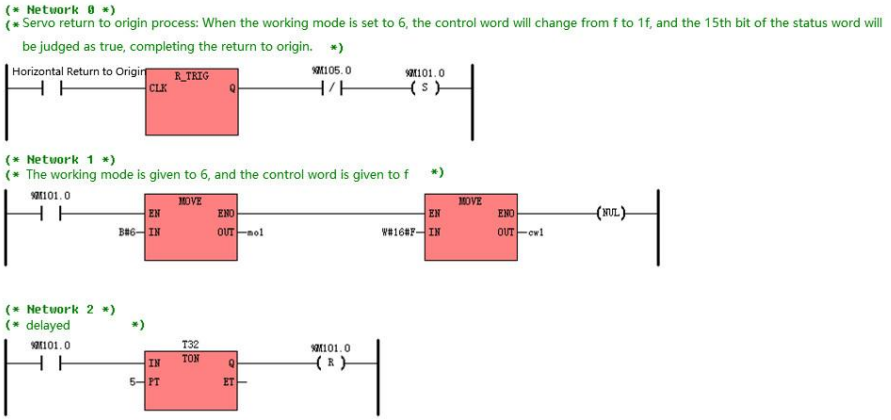
LD

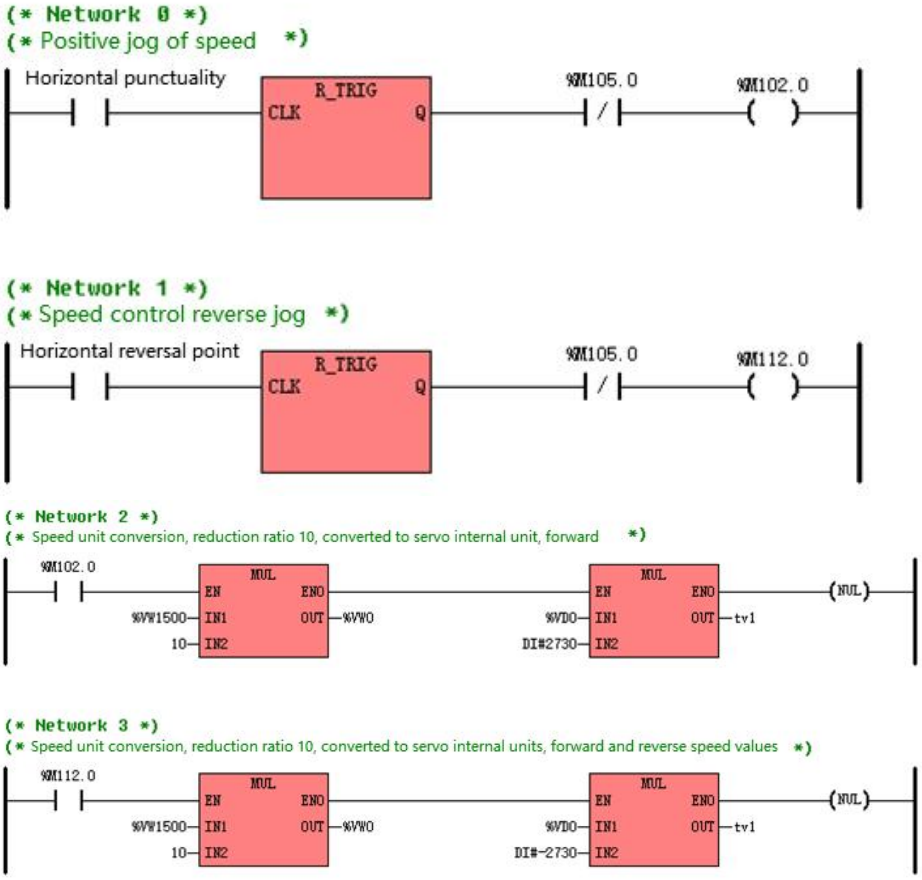
Power-on initialization subroutine: The process of PLC power-on to servo initialization, the working mode is 1, the control word is 6, the delay control word is F servo lock axis, and the power-on completion sign is given.

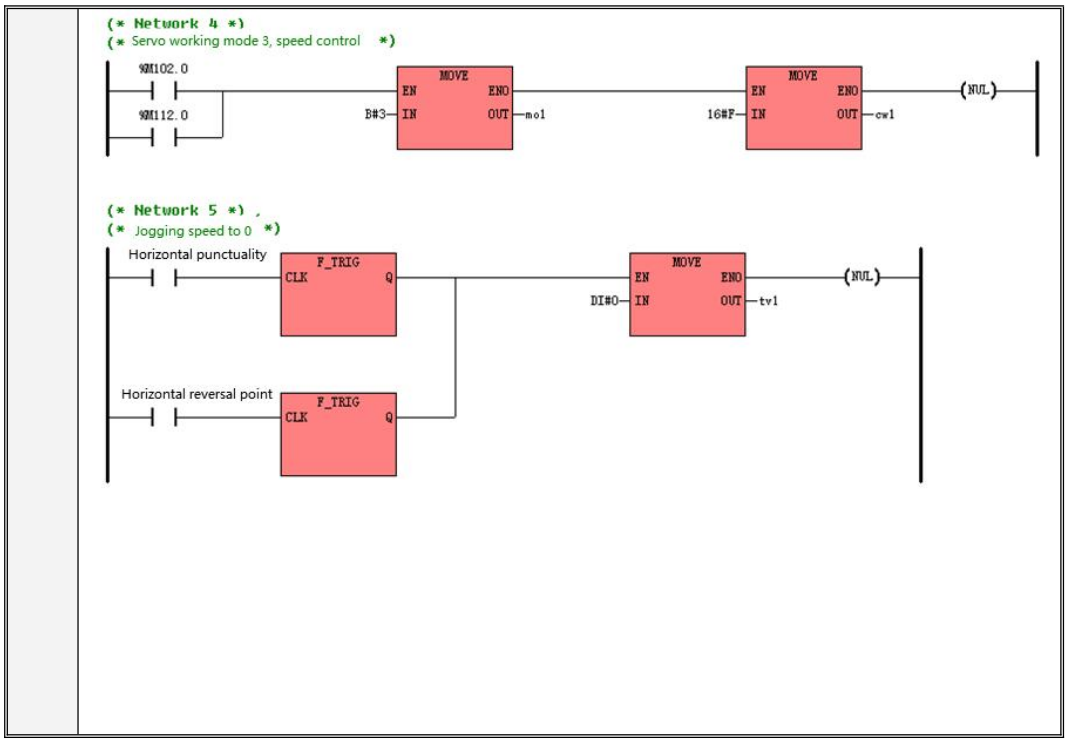


LD

Return to the origin subroutine: set the working mode to 6, the control word produces a change from f to 1f, wait for the 15th bit of the status word to change from false to true, and return to the origin is completed.

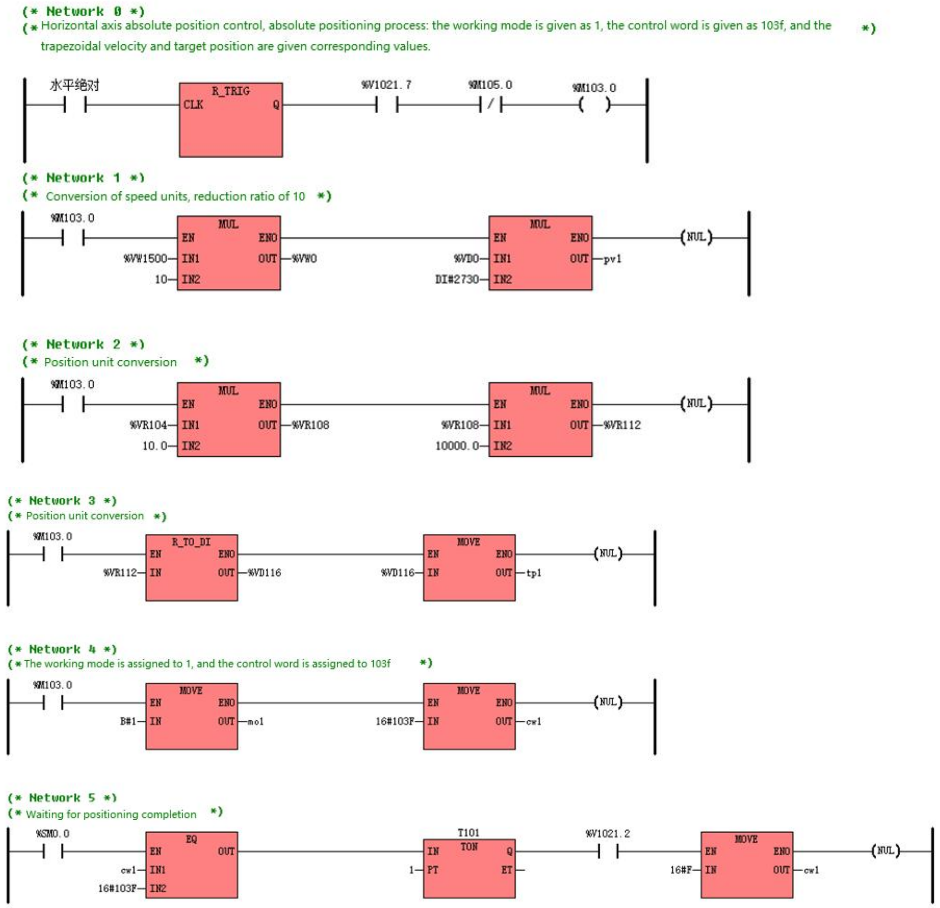


LD	Speed control subroutine: jogging control servo moves in speed mode, the working mode is 3, the control word is f, and the target speed needs to be converted to the internal value of the servo according to the engineering unit. 


(* Network 4 *)
(* Servo working mode 3, speed control *)
%M102.0
%M112.0
MOVE
EN
ENO
B#3
IN
OUT
mo1
MOVE
EN
ENO
16#F
IN
OUT
cw1
(NUL)
(* Network 5 *) ,
(* Jogging speed to 0 *)
Horizontal punctuality
F_TRIG
CLK
Q
MOVE
EN
ENO
DI#0
IN
OUT
tv1
(NUL)
Horizontal reversal point
F_TRIG
CLK
Q

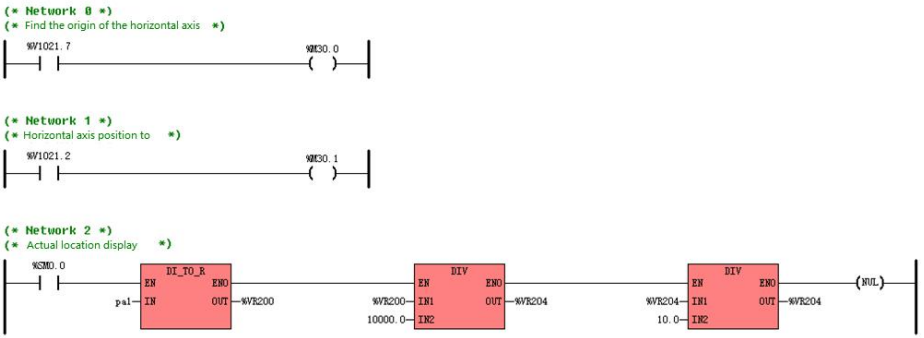
LD

Absolute Position subroutine: set the working mode to 1, the control word to 103f, the trapezoidal speed and target position need to be converted from the engineering unit to the servo internal unit, and wait for the 10th bit of the status word to change from false to true to complete the positioning.



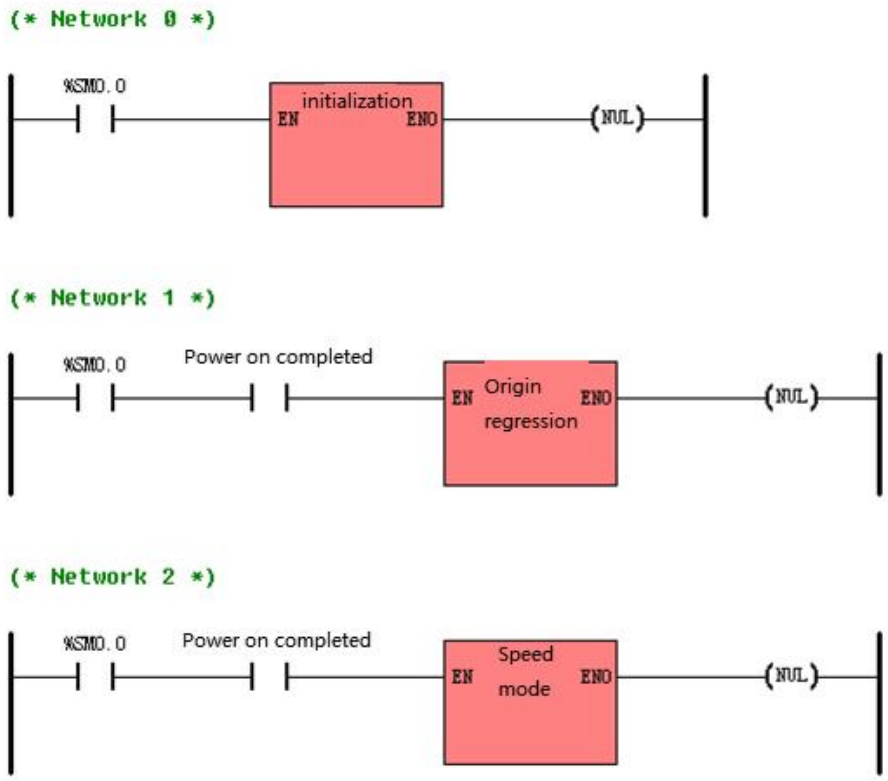
LD

Display part: display the origin found mark, positioning complete mark, actual position.

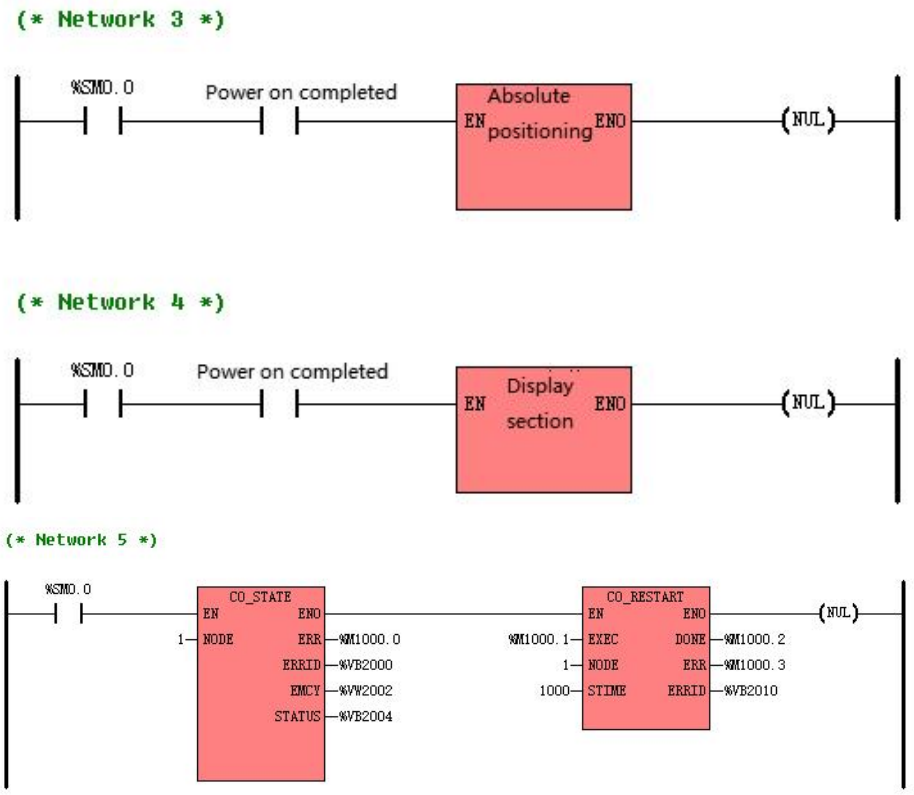


LD

Main program MAIN: call each subroutine



LD



CO_STATE: Monitor the status of the slave station: whether it is offline, emergency message, node protection status.

CO_RESTART: Restart the node. When the node stops due to a failure or the master station is powered on before the slave station, you can restart the node through this instruction to make the slave station enter the operating state.

IL	Power-on initialization subroutine: The process of PLC power-on to servo initialization, the working mode is 1, the control word is 6, the delay control word is F servo lock axis, and the power-on completion sign is given.
	(* Network 0 *)
	(*After the PLC is powered on, initialize the servo working mode to 1 and control the word to 6*)
	LD %M100.0
	MOVE B#1, mo1
	MOVE 16#6, cw1
	S %M100.1
	(* Network 1 *)
	(*Delay the servo control word to the f lock axis and provide the power on completion flag at the same time *)
	LD %M100.1
	TON T31, 5
	MOVE 16#F, cw1
	R %M100.1
	S Power on completed

IL	Return to the origin subroutine: set the working mode to 6, the control word produces a change from f to 1f, wait for the 15th bit of the status word to change from false to true, and return to the origin is completed.
	(* Network 0 *)
	(*Servo return to origin process: When the working mode is set to 6, the control word will change from f to 1f, and the 15th bit of the status word will be judged as true, completing the return to origin.*)
	LD Horizontal Return to Origin
	R_TRIG
	ANDN %M105.0
	S %M101.0
	(* Network 1 *)
	(*The working mode is given to 6, and the control word is given to f *)
	LD %M101.0
	MOVE B#6, mo1
	MOVE W#16#F, cw1
	(* Network 2 *)
	(*delayed *)
	LD %M101.0
	TON T32, 5
	R %M101.0
	(* Network 3 *)
	(*Control word for 1f *)
	LD T32
	MOVE W#16#1F, cw1
	(* Network 4 *)
	(*Waiting for Return to Origin to Complete *)
	LD %SM0.0
	EQ cw1, W#16#1F
	TON T100, 1
	AND %V1021.7
	MOVE W#16#F, cw1

IL	Speed control subroutine: jogging control servo moves in speed mode, the working mode is 3, the control word is f, and the target speed needs to be converted to the internal value of the servo according to the engineering unit.
	(* Network 0 *)
	(*Positive jog of speed control *)
	LD Horizontal punctuality
	R_TRIG
	ANDN %M105.0
	ST %M102.0
	(* Network 1 *)
	(*Speed control reverse jog *)
	LD Horizontal reversal point
	R_TRIG
	ANDN %M105.0
	ST %M112.0
	(* Network 2 *)
	(*Speed unit conversion, reduction ratio 10, converted to servo internal unit, forward speed value *)
	LD %M102.0
	MOVE %VW1500, %VW0
	MUL 10, %VW0
	MOVE %VD0, tv1
	MUL DI#2730, tv1
	(* Network 3 *)
	(*Speed unit conversion, reduction ratio 10, converted to servo internal units, forward and reverse speed values *)
	LD %M112.0
	MOVE %VW1500, %VW0
	MUL 10, %VW0
	MOVE %VD0, tv1
	MUL DI#-2730, tv1

IL	(* Network 4 *)
	(*Servo working mode 3, speed control *)
	LD %M102.0
	OR %M112.0
	MOVE B#3, mo1
	MOVE 16#F, cw1
	(* Network 5 *)
	(*Jogging speed to 0*)
	LD Horizontal punctuality
	F_TRIG
	OR(
	LD Horizontal reversal point
	F_TRIG
	)
	MOVE DI#0, tv1

IL

```
(* Network 1 *)
(*Conversion of speed units, reduction ratio of 10*)
LD      %M103.0
MOVE    %VW1500, %VW0
MUL     10, %VW0
MOVE    %VD0, pv1
MUL     DI#2730, pv1
(* Network 2 *)
(*Position unit conversion *)
LD      %M103.0
MOVE    %VR104, %VR108
MUL     10.0, %VR108
MOVE    %VR108, %VR112
MUL     10000.0, %VR112
(* Network 3 *)
(*Position unit conversion *)
LD      %M103.0
R_TO_DI %VR112, %VD116
MOVE    %VD116, tp1
(* Network 4 *)
(*The working mode is assigned to 1, and the control word is assigned to 103f *)
LD      %M103.0
MOVE    B#1, mo1
MOVE    16#103F, cw1
(* Network 5 *)
(*Waiting for positioning completion *)
LD      %SM0.0
EQ      cw1, 16#103F
TON     T101, 1
AND     %V1021.2
MOVE    16#F, cw1
```

IL	Display part: display the origin found mark, positioning complete mark, actual position.
	(* Network 0 *)
	(*Find the origin of the horizontal axis *)
	LD %V1021.7
	ST %M30.0
	(* Network 1 *)
	(*Horizontal axis position to *)
	LD %V1021.2
	ST %M30.1
	(* Network 2 *)
	(*Actual location display *)
	LD %SM0.0
	DI_TO_R pa1, %VR200
	MOVE %VR200, %VR204
	DIV 10000.0, %VR204
	DIV 10.0, %VR204
	(* Network 3 *)
	LD %M666.0
	MOVE 16#86, cw1

IL	Main program MAIN: call each subroutine (* Network 0 *) LD %SM0.0 CAL initialization (* Network 1 *) LD %SM0.0 AND Power on completed CAL Origin regression (* Network 2 *) LD %SM0.0 AND Power on completed CAL speed mode (* Network 3 *) LD %SM0.0 AND Power on completed CAL Absolute positioning (* Network 4 *) LD %SM0.0 AND Power on completed CAL Display Section (* Network 5 *) LD %SM0.0 CO_STATE 1, %M1000.0, %VB2000, %VW2002, %VB2004 CO_RESTART %M1000.1, 1, 1000, %M1000.2, %M1000.3, %VB2010

message analysis

1、When the PLC is powered on for the first time, it is necessary to configure the servo and send NMT management messages to make the servo enter the pre-operation state and send SDO to map the objects in PDO. The configuration messages sent are as follows:

系统时间	时间标识	CAN通道	传输方向	ID号	帧类型	帧格式	长度	数据
18:03:23.	0x3AB480(	ch2	接收	0x0701	数据帧	标准帧	0x01	x 00
18:03:26.	0x3ABC20(	ch2	接收	0x0000	数据帧	标准帧	0x02	x 80 00
18:03:26.	0x3ABC26E	ch2	接收	0x0000	数据帧	标准帧	0x02	x 81 01
18:03:26.	0x3ABC273	ch2	接收	0x0701	数据帧	标准帧	0x01	x 00
18:03:26.	0x3ABC45E	ch2	接收	0x0000	数据帧	标准帧	0x02	x 80 01
18:03:26.	0x3ABC45E	ch2	接收	0x0601	数据帧	标准帧	0x08	x 2B 0C 10 00 E8 03 00 00
18:03:26.	0x3ABC462	ch2	接收	0x0581	数据帧	标准帧	0x08	x 60 0C 10 00 E8 03 00 00
18:03:26.	0x3ABC464	ch2	接收	0x0601	数据帧	标准帧	0x08	x 2F 0D 10 00 03 00 00 00
18:03:26.	0x3ABC46A	ch2	接收	0x0581	数据帧	标准帧	0x08	x 60 0D 10 00 03 00 00 00
18:03:26.	0x3ABC46(	ch2	接收	0x0601	数据帧	标准帧	0x08	x 23 00 18 01 81 01 00 80
18:03:26.	0x3ABC470	ch2	接收	0x0581	数据帧	标准帧	0x08	x 60 00 18 01 81 01 00 80
18:03:26.	0x3ABC472	ch2	接收	0x0601	数据帧	标准帧	0x08	x 2F 00 1A 00 00 00 00 00
18:03:26.	0x3ABC477	ch2	接收	0x0581	数据帧	标准帧	0x08	x 60 00 1A 00 00 00 00 00
18:03:26.	0x3ABC479	ch2	接收	0x0601	数据帧	标准帧	0x08	x 23 00 1A 01 20 00 63 60
18:03:26.	0x3ABC47E	ch2	接收	0x0581	数据帧	标准帧	0x08	x 60 00 1A 01 20 00 63 60
18:03:26.	0x3ABC480	ch2	接收	0x0601	数据帧	标准帧	0x08	x 2F 00 1A 00 01 00 00 00
18:03:26.	0x3ABC484	ch2	接收	0x0581	数据帧	标准帧	0x08	x 60 00 1A 00 01 00 00 00
18:03:26.	0x3ABC486	ch2	接收	0x0601	数据帧	标准帧	0x08	x 2F 00 18 02 FF 00 00 00

First of all, the first message in the red box is 0000 80 01, 0x0000 is the COB-ID of the NMT management message, the data 0x80 means that the node enters the pre-operation state, and the data 0x01 means the node 1, which is the device with the station number 1 .

The COB-ID of the second message in the red frame is 0x601 and 0x581, 0x601 is the SDO message sent by PLC, and 0x581 is the message of servo response. The SDO message sent in the figure is to configure TPDO and RPDO, and map the PDO in the PLC hardware configuration to the servo.

The meaning of the message in the second red box is to map the object of 606300 actual position to the servo. The figure below shows the TPDO and RPDO mapped to the servo.

TPDOSet

TPDO1 | TPDO2 | TPDO3 | TPDO4 | TPDO5 | TPDO6 | TPDO7 | TPDO8

N	Index	Type	Name	Value	Unit
0	1A0000	uint8	TPDO1映射组	1	DEC
1	1A0001	uint32	TPDO1映射1	60630020	HEX
2	1A0002	uint32	TPDO1映射2	00000000	HEX
3	1A0003	uint32	TPDO1映射3	00000000	HEX
4	1A0004	uint32	TPDO1映射4	00000000	HEX
5	1A0005	uint32	TPDO1映射5	00000000	HEX
6	1A0006	uint32	TPDO1映射6	00000000	HEX
7	1A0007	uint32	TPDO1映射7	00000000	HEX
8	1A0008	uint32	TPDO1映射8	00000000	HEX
9	180001	uint32	TPDO1站号	00000181	HEX
10	180002	uint8	TPDO1传输类型	255	DEC
11	180003	uint16	TPDO1禁止时间	10	DEC
12	180005	uint16	TPDO1事件时间	0	DEC

RPDOSet

RPDO1 | RPDO2 | RPDO3 | RPDO4 | RPDO5 | RPDO6 | RPDO7 | RPDO8

N	Index	Type	Name	Value	Unit
0	160000	uint8	RPDO1映射组	3	DEC
1	160001	uint32	RPDO1映射1	60400010	HEX
2	160002	uint32	RPDO1映射2	60600008	HEX
3	160003	uint32	RPDO1映射3	607A0020	HEX
4	160004	uint32	RPDO1映射4	00000000	HEX
5	160005	uint32	RPDO1映射5	00000000	HEX
6	160006	uint32	RPDO1映射6	00000000	HEX
7	160007	uint32	RPDO1映射7	00000000	HEX
8	160008	uint32	RPDO1映射8	00000000	HEX
9	140001	uint32	RPDO1站号	00000201	HEX
10	140002	uint8	RPDO1传输类型	255	DEC
11	140003	uint16	RPDO1禁止时间	0	DEC

2、After the above configuration is completed, the PLC sends an NMT management message to start the node. After the node is started, the servo can be controlled by operating the PDO in the PLC program. The

messages before starting the node are configuration messages automatically sent by the PLC in the background without any operation by the user.

0x3ABC52A	ch2	接收	0x0601	数据帧	标准帧	0x08	x	23 01 14 01 01 03 00 00
0x3ABC52E	ch2	接收	0x0581	数据帧	标准帧	0x08	x	60 01 14 01 01 03 00 00
0x3ABC95C	ch2	接收	0x0000	数据帧	标准帧	0x02	x	01 01
0x3ABC956	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF FF FF FF
0x3ABC957	ch2	接收	0x0281	数据帧	标准帧	0x04	x	31 42 00 00
0x3ABC95F	ch2	接收	0x0000	数据帧	标准帧	0x02	x	01 00
0x3ABC962	ch2	接收	0x0181	数据帧	标准帧	0x04	x	00 00 00 00
0x3ABC964	ch2	接收	0x0281	数据帧	标准帧	0x04	x	31 42 00 00
0x3ABC974	ch2	接收	0x0201	数据帧	标准帧	0x07	x	00 00 00 00 00 00 00
0x3ABC97E	ch2	接收	0x0281	数据帧	标准帧	0x04	x	70 42 00 00
0x3ABC983	ch2	接收	0x0301	数据帧	标准帧	0x08	x	00 00 00 00 00 00 00 00
0x3ABC9CE	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF FF FF FF
0x3ABD561	ch2	接收	0x0181	数据帧	标准帧	0x04	x	00 00 00 00
0x137712	ch2	接收	0x0181	数据帧	标准帧	0x04	x	01 00 00 00
0x1378C9	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF FF FF FF
0x137937	ch2	接收	0x0181	数据帧	标准帧	0x04	x	01 00 00 00
0x137B4C	ch2	接收	0x0201	数据帧	标准帧	0x07	x	0F 00 03 00 00 00 00
0x137B4E	ch2	接收	0x0301	数据帧	标准帧	0x08	x	00 00 00 00 68 2A 04 00
0x137B52	ch2	接收	0x0281	数据帧	标准帧	0x02	x	37 52
0x137B5C	ch2	接收	0x0181	数据帧	标准帧	0x04	x	02 00 00 00
0x137B5E	ch2	接收	0x0281	数据帧	标准帧	0x02	x	37 56
0x137B84	ch2	接收	0x0281	数据帧	标准帧	0x02	x	37 52
0x137BC0	ch2	接收	0x0281	数据帧	标准帧	0x02	x	37 42
0x137BCA	ch2	接收	0x0181	数据帧	标准帧	0x04	x	1E 00 00 00

The message 0000 01 01 in the first red box means to start node 1, so that the device enters the operating state.

The COB-ID of the message in the second red frame is 0x181, 0x281, 0x201, and 0x301, which are the COB-ID of PDO. The data in these messages correspond to the objects configured by PDO in the hardware configuration. 0x181 and 0x281 are the sending PDO of the servo, and 0x201 and 0x301 are the receiving PDO of the servo. We configure the status word in 0x281. From the message, we can see that the value of the status word is 0x4231, and the data in the PDO needs to be read backwards.

The message in the third red box is the data sent by PLC to the servo. The configuration in 0x201 is the control word, working mode, and target position in sequence. 0x301 is configured with trapezoidal speed and target speed. The message data in 0x201 is 0F 00 03 00 00 00 00 00. This message is sent by triggering the speed control subroutine in the PLC program. The control word occupies a length of one word corresponding to 00 0F, the working mode occupies a length of one byte 03, and the target position occupies a length of two

words 00 00 00 00 Because it is speed control, the target position has no value.

3、Node protection message

0x3ABEC8E	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FE	FF	FF	FF
0x3ABECFC	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3ABF04A	ch2	接收	0x0701	远程帧	标准帧	0x00					
0x3ABF04E	ch2	接收	0x0701	数据帧	标准帧	0x01	x	05			
0x3ABF5FF	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FD	FF	FF	FF
0x3ABF66E	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3ABFA49	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FD	FF	FF	FF
0x3ABFAB7	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3AC0127	ch2	接收	0x0181	数据帧	标准帧	0x04	x	00	00	00	00
0x3AC0195	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3AC0428	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FD	FF	FF	FF
0x3AC0496	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3AC0873	ch2	接收	0x0181	数据帧	标准帧	0x04	x	00	00	00	00
0x3AC08E1	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3AC094F	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FE	FF	FF	FF
0x3AC09BC	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3AC0B06	ch2	接收	0x0181	数据帧	标准帧	0x04	x	00	00	00	00
0x3AC0B74	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3AC109A	ch2	接收	0x0181	数据帧	标准帧	0x04	x	00	00	00	00
0x3AC1108	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3AC14E5	ch2	接收	0x0181	数据帧	标准帧	0x04	x	00	00	00	00
0x3AC15C1	ch2	接收	0x0181	数据帧	标准帧	0x04	x	FF	FF	FF	FF
0x3AC1744	ch2	接收	0x0701	远程帧	标准帧	0x00					
0x3AC1746	ch2	接收	0x0701	数据帧	标准帧	0x01	x	85			
0x3AC1778	ch2	接收	0x0181	数据帧	标准帧	0x04	x	00	00	00	00

The master station periodically sends remote frames with a COB-ID of 0x701 according to the set supervision time and life factor, and the servo loop returns to states 05 and 85 during normal operation.

4、emergency message

16:03:21.	0x11BA7F2	ch2	接收	0x0081	数据帧	标准帧	0x08	x	31	73	20	00	02	40	00	00
16:03:21.	0x11BA7F5	ch2	接收	0x0281	数据帧	标准帧	0x02	x	38	02						
16:03:21.	0x11BA7F6	ch2	接收	0x0000	数据帧	标准帧	0x02	x	02	01						
16:03:21.	0x11BA7F9	ch2	接收	0x0081	数据帧	标准帧	0x08	x	31	73	20	00	02	40	00	00
16:03:21.	0x11BA7FE	ch2	接收	0x0000	数据帧	标准帧	0x02	x	02	01						

应急报文说明

当设备内部出现致命错误将触发应急报文，由应用设备以最高优先级发送到其他设备。一条应急报文由 8 字节组成。

表 10-22 应急报文格式

COB-ID	Byte 0-1	Byte2	Byte4-5	Byte6-7
紧急报文站号 0x101400	应急错误代码 0x603F00	错误寄存器(0x100100)	错误状态 0x260100	错误状态 0x260200

表 10-23 应急错误代码 0x603F00

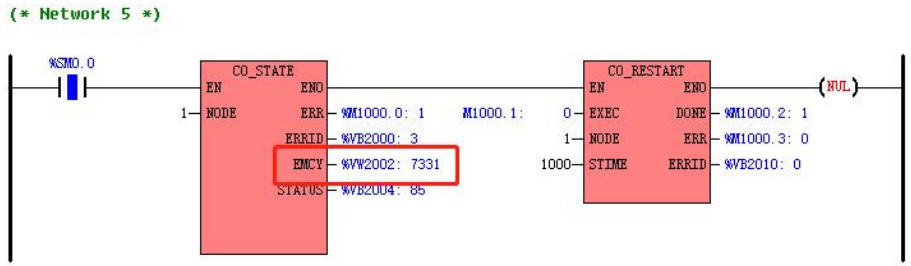
报警内容	应急错误代码(Hex)	报警内容	应急错误代码(Hex)
通讯式编码器没有连接	0x7331	电流传感器故障	0x5210
通讯式编码器多圈错误	0x7320	软件看门狗复位	0x6010
通讯式编码器校验错误	0x7330	异常中断	0x6011
驱动器温度过高	0x4210	MCU 故障	0x7400
驱动器总线电压过高	0x3210	电机型号配置错误	0x6320
驱动器总线电压过低	0x3220	电机动力线缺相	0x6321
驱动器功率部分短路或电机短路	0x2320	预使能报警	0x5443
电流采样饱和	0x2321	正限位报错	0x5442
驱动器制动电阻异常	0x7110	负限位报错	0x5441
实际跟随误差超过允许	0x8611	SPI 故障	0x6012
逻辑电压低	0x5112	总线通讯错误	0x8100
电机或驱动器过载	0x2350	总线通讯超时	0x81FF
输入脉冲频率过高	0x8A80	全闭环检查错误	0x8A81

The first picture shows the message captured by unplugging the servo encoder cable. 0x81 corresponds to the COB-ID of the emergency message of No. 1 slave, and the data is 31 73 20 00 02 40 00 00.

The second picture is the description of kinco servo emergency message (different manufacturers have different definitions of emergency message, please refer to the actual application). From the data, we can know that the data 0x7331 of the emergency message corresponds to the alarm content. The communication encoder is not connected. .

Through the CO_STATE instruction in the figure below, you can monitor the emergency message codes and node protection status that have occurred. In addition, the node can be restarted through the CO_RESTART instruction, which is applicable to situations where the slave station is in a non-operating state due to alarms, communication interruptions, and the master station being powered on before the slave

station. You can use this instruction to restart the node to make the slave station re-enter the operating state.



Note: The explanation of message parsing needs to be understood in conjunction with the NMT, PDO, SDO, node protection, etc. in the CANopen master function introduced in the previous chapters.

10.5.6 CANOpen slave station function

10.5.6.1 Overview

KS105C1-16DT supports CANOpen slave station function and has the following characteristics:

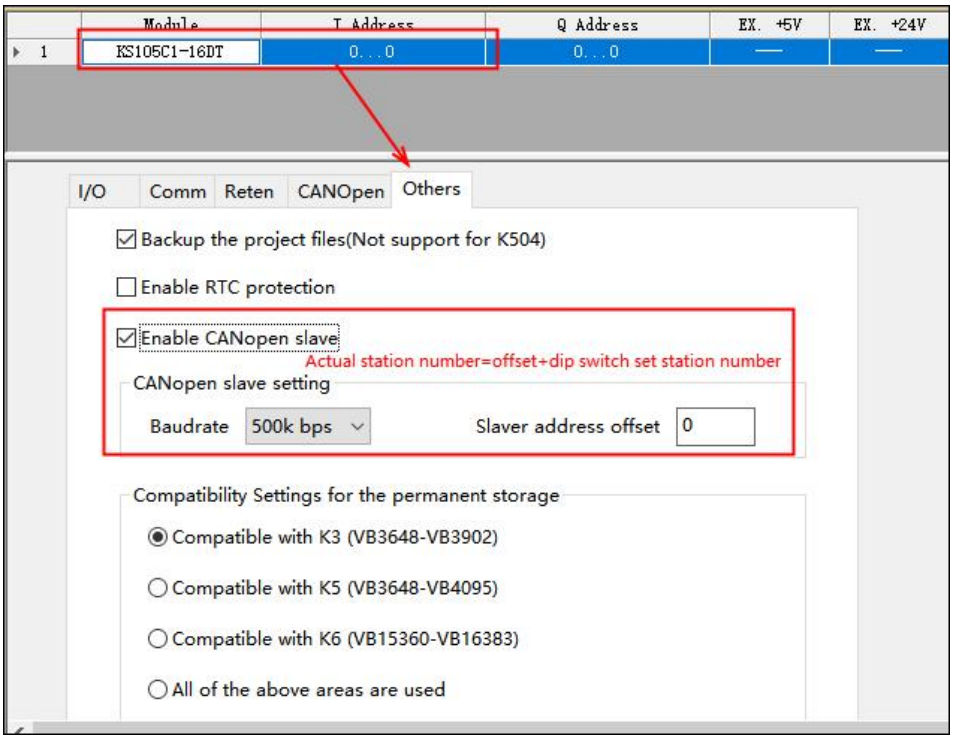
- CAN interface adopts CAN2.0A standard.
- Compliant with DS301 V4.2.0 and DS405 V2.0 protocols
- Support NMT network management service, support heartbeat protocol.
- Support SDO Server, speed up transfer mode.
- It supports up to 16 TPDOs and 16 RPDOs, and the communication parameters and mapping parameters of PDOs can be freely configured.

Note: When using, the serial numbers of PDO must be continuous. For example, it is legal to use TPDO1, TPDO2, and TPDO3, but it is illegal to use TPDO1, TPDO3, and TPDO4.

10.5.6.2 Enable CANOpen slave function

In the user project, enter [PLC Hardware Configuration], and in the [Other] page of the CPU module,

the user can configure the CANOpen slave function.



[Enable CANOPEN slave station]: If this option is selected, the PLC will act as a CANOpen slave station.

[Baud rate] : Select the baud rate used by the CAN interface. The baud rate of all nodes in a network must be the same.

[Slave station address offset]: The real station number of this slave station is equal to "combined value of the 1st to 3rd digit DIP switch" plus "slave station address offset". For example, if the combined value of the 1st to 3rd bit DIP switches is 5, and the slave station address offset is set to 8, then the station number of this slave station is 13.。

10.5.6.3 Object dictionary (EDS file)

Users can go to Kinco's official website to download the eds file of this slave device for free.

The EDS file of this device has been integrated in the KincoBuilder software: in the [CANOpen Master

Station]->[Network Configuration] page of the hardware configuration. This device is displayed in the "Motion Controller and PLC" group of "All Slave Module List", and its name is displayed as "Kinco KPLC for Kinco". If the user uses Kinco PLC as the master station and slave station, there is no need to import the eds file again.

This device has temporarily opened the following memory areas as CANOpen communication objects, which can be mapped to PDO for use. If you need to use a larger memory area, please contact Kinco to provide another eds file.

object index	Data type	PLC memory area	Quantity	R&W attributes
0xA040	BYTE	IB0～IB9	10	Read only
0xA0C0	INT16	AIW0～AIW18	10	Read only
0xA4C0	BYTE	QB0～QB9	10	Read and write
0xA4C1	BYTE	MB0～MB9	10	Read and write
0xA4D2	BYTE	VB508～VB517	10	Read and write
0xA540	INT16	AQW0～AQW18	10	Read and write
0xA550	INT16	VW0～VW18	10	Read and write
0xA641	INT32	VD1016～VD1052	10	Read and write
0xA6C2	FLOAT	VR2032～VR2068	10	Read and write

10.5.7 CAN free communication function

10.5.7.1 Overview

KPLC provides a set of CAN communication instructions, which can initialize the CAN port, send and receive data through the CAN port, etc. Users can use these instructions to freely write communication programs to communicate with other devices. The CAN free communication function can be used simultaneously with the expansion bus function and the CANOpen master station function.

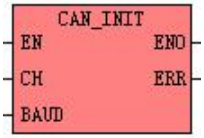
CAN communication instructions support CAN2.0A and CAN2.0B standards, and these instructions only support data frames, not remote frames. CAN data frame format is as follows:

ID	Byte 1-8
11 bits (CAN2.0A, standard frame) or 29 bits (CAN2.0B, extended frame)	1-8 byte length data

10.5.7.2 CAN Free Communication Instruction

10.5.7.2.1 CAN_INIT (Initialize the CAN port)

- Instruction and its operand description

	Name	Instruction format	Cr values affected	
LD	CAN_INIT			☑ KS ☑ KW ☑ K209M ☑ K6
IL	CAN_INIT	CAN_INIT CH, BAUD	U	

Parameters	Input/output	Data type	Memory area allowed
EN	Input	BOOL	I、Q、V、M、L、SM
CH	Input	INT	constant
BAUD	Input	INT	L、M、V、constant
ERR	Output	BOOL	L、M、V、constant

Parameters	Description
EN	enable terminal.
CH	The CAN port used. 0 means CAN1, 1 means CAN2, 2 means K541 module
BAUD	CAN's baud rate。 8 --- 1000K 7 --- 800K 6 --- 500K 5 --- 250K 4 --- 125K 3 --- 50K 2 --- 20K 1--- 10K
ERR	Whether the instruction execution was successful. 0 means success, 1 means there is an error (such as parameter error)

The rising edge transition of the EN input will trigger the execution of this instruction, which is used to

initialize the specified CAN interface (CH), and set the CAN baud rate to the value represented by BAUD.

- LD

  The rising edge of EN will trigger the execution of the instruction once, otherwise it will not be executed.

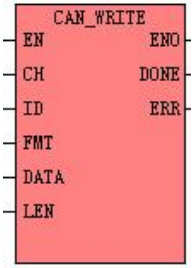
- IL

  The rising edge of CR will trigger the instruction to be executed once, otherwise it will not be executed.

The execution of this instruction does not affect the CR value.

10.5.7.2.2 CAN_WRITE (Send a CAN message)

➢ Instruction and its operand description

	Name	Instruction format	Cr values affected	
LD	CAN_WRITE			☑ KS ☑ KW ☑ K209M ☑ K6
IL	CAN_WRITE	CAN_WRITE CH,ID,FMT,DATA,LEN,DONE,ERR	U	

Parameters	Input/output	Data type	Memory area allowed
CH	Input	INT	constant
ID	Input	DWORD	L、M、V、constant
FMT	Input	BYTE	L、M、V、constant
DATA	Input	BYTE	L、M、V
LEN	Input	BYTE	L、M、V、constant

DONE	Output	BOOL	L、M、V
ERR	Output	BOOL	L、M、V

Parameters	Description
EN	enable terminal.
CH	The CAN interface used. 0 means CAN1, 1 means CAN2, 2 means K541 module
FMT	message format. 0 means standard frame, 1 means extended frame.
DATA	The starting byte address of the data to be sent.
LEN	The length of the data to be sent. Unit: byte.
DONE	Whether the message has been sent. DONE is set to 0 during execution, and DONE is set to 1 when the transmission is completed.
ERR	Whether there is an error in sending the message. If sending fails (usually because the send buffer is full), ERR is set to 1.

Note: ID, FMT, LEN, Must be both constant type or memory type at the same time. The DATA and LEN parameters together form a variable-length memory block, and the addresses in the entire memory block must be legal addresses.

The CAN message to be sent is specified by ID number, format (FMT, indicating extended frame or standard frame), data (DATA, indicating the starting address of message data storage), and length LEN.

The rising edge transition of the EN input terminal will trigger the execution of the instruction once, write the message to be sent into the send buffer inside the PLC, and then send it out through the specified CAN interface CH by PLC scheduling. If the instruction sends the message successfully, set DONE to 1 and ERR to 0. If the send buffer is full or the packet fails to be sent, set DONE and ERR to 1 at the same time.

In one project, the total number of CAN_TX instructions and CAN_WRITE instructions can be up to 64!

- LD

  The rising edge of EN will trigger the execution of the instruction once, otherwise it will not be executed.

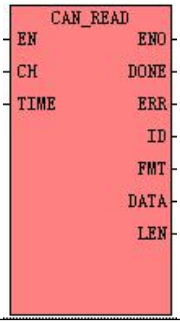
- IL

  The rising edge of CR will trigger the instruction to be executed once, otherwise it will not be executed.

The execution of this instruction does not affect the CR value.

10.5.7.2.3 CAN_READ (Receive a CAN message)

- Instruction and its operand description

	Name	Instruction format	Cr values affected	
LD	CAN_READ			☑ KS ☑ KW ☑ K209M ☑ K6
IL	CAN_READ	CAN_READ CH, TIME, DONE, ERR, FMT, DATA, LEN	U	

Parameters	Input/output	Data type	Memory area allowed
CH	Input	INT	constant
TIME	Input	INT	L、M、V、constant
DONE	Output	BOOL	L、M、V
ERR	Output	BOOL	L、M、V
ID	Output	DWORD	L、M、V
FMT	Output	BYTE	L、M、V
DATA	Output	BYTE	M、V
LEN	Output	BYTE	L、M、V

Parameters	Description
EN	enable terminal.
CH	The CAN interface used. 0 means CAN1, 1 means CAN2, 2 means K541 module
TIME	overtime time. After starting receiving, if no message is received within the specified time, it will exit the receiving state after timeout and set ERR to 1

ID	The ID number of the received message.
FMT	Received message format. 0 means standard frame, 1 means extended frame.
DATA	The starting byte address where the received message data is stored.
LEN	Received packet data length. Unit: byte.
DONE	Whether the reception is complete. DONE is set to 0 during execution, and DONE is set to 1 when the transmission is completed.
ERR	Whether there is an error in receiving the message. If the reception fails, ERR is set to 1.

Note: The DATA and LEN parameters together form a variable-length memory block, and the addresses in the entire memory block must be legal addresses.

The rising edge transition of the EN input terminal will trigger the execution of the instruction: start the receiving state, and receive any message from the specified CAN interface CH.

After the reception is started, if a CAN message is received within the specified timeout time TIME, the PLC will place the relevant data of the message in the output parameters ID, FMT, DATA, and LEN respectively. At the same time, set DONE to 1 to quit receiving. If no message is received within the specified timeout period, it will exit the receiving state after timeout, and set DONE and ERR to 1.

Note: 1) CAN_READ instruction has lower priority than CAN_RX instruction.

2) After the CAN_READ instruction is started, any message on the bus will be received by it, so if multiple CAN_READ instructions are called in the program, the last started instruction will take effect.

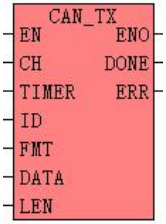
- LD

The rising edge of EN will trigger the execution of the instruction once, otherwise it will not be executed.

- IL

The rising edge of CR will trigger the instruction to be executed once, otherwise it will not be executed. The execution of this instruction does not affect the CR value.

10.5.7.2.4 CAN_TX (Automatically send CAN messages)

	Name	Instruction format	Adopt to
LD	CAN_TX		☑ KS ☑ KW209M ☑ K6

Parameters	Input/output	Data type	Memory area allowed
CH	Input	INT	constant
TIMER	Input	INT	L、M、V、constant
ID	Input	DWORD	L、M、V、constant
FMT	Input	INT	L、M、V、constant
DATA	Output	BYTE	M、V
LEN	Output	BYTE	L、M、V
DONE	Output	BOOL	L、M、V
ERR	Output	BOOL	L、M、V

Parameters	Description
EN	enable terminal.
CH	The CAN interface used. 0 means CAN1, 1 means CAN2, 2 means K541 module
TIMER	Period for sending messages to be sent at regular intervals, in ms. 0 means that the timing sending function is not enabled.
ID	ID of the message to be sent
FMT	The format of the message to be sent. 0 means standard frame, 1 means extended frame.
DATA	The first address where the data of the message to be sent is stored.
LEN	The data length of the message to be sent, unit: byte.
DONE	Send completed flag. After each successful transmission, DONE is automatically set to 1 and lasts for at least one scan cycle.
ERR	Send error flag. 1 means that the sending failed this time.

Note: ID, FMT, TIMER and LEN parameters must be constant or variable at the same time; DATA and LEN parameters form a variable-length memory block, and this memory block must all be located in the legal memory area, otherwise the result is unpredictable.

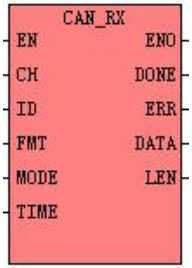
The PLC maintains a message automatic sending list. When the message in the list meets the sending conditions, the PLC will automatically send the message. The conditions for automatic sending of a message are: if the data in the message changes, it will be sent once immediately; if the set sending cycle time is up, it will be sent once immediately. After the message is sent, the output parameter DONE is automatically set to 1 and automatically becomes 0 after one scan. If the sending fails (because the sending buffer is full or the message sending fails), the output parameter ERR is automatically set to 1. The maximum length of the sending message list is 64 items.

The CAN_TX instruction is used to add a message to the send message list, and the message is specified by the message ID number, format, DATA, and LEN. The TIMER parameter value indicates the period (ms) of regular sending. If the TIMER value is 0, it means that it will not be sent regularly.

A rising edge transition at the EN input triggers the execution of the instruction. After this instruction is executed, the PLC immediately adds the message specified by the instruction to the automatic sending list. **Therefore, in a project, a CAN_TX instruction only needs to be executed once, and there is no need to execute it multiple times. In addition, in a project, the total number of CAN_TX instructions and CAN_WRITE instructions is allowed to be 64 at most!**

10.5.7.2.5 CAN_RX (Automatically receive CAN messages with a specific ID)

- Instruction and its operand description

	Name	Instruction format	Cr values affected	
LD	CAN_RX			☑ KS ☑ KW ☑ K209M ☑ K6

IL	CAN_RX	CAN_RX CH,ID,FMT, MODE,TIME, DONE,ERR, FMT,DATA,LEN	U	

Parameters	Input/output	Data type	Memory area allowed
CH	Input	INT	constant
ID	Input	DWORD	L、M、V、constant
FMT	Input	INT	L、M、V、constant
MODE	Input	INT	L、M、V、constant
TIME	Input	INT	L、M、V、constant
DONE	Output	BOOL	L、M、V
ERR	Output	BOOL	L、M、V
DATA	Output	BYTE	M、V
LEN	Output	BYTE	L、M、V

Parameters	Description
EN	enable terminal.
CH	The CAN interface used. 0 means CAN1, 1 means CAN2, 2 means K541 module
ID	ID of the message to be received
FMT	The format of the message to be received. 0 means standard frame, 1 means extended frame.
MODE	receive mode. 0 means permanent receiving mode, 1 means single receiving mode
TIME	Receive timeout. unit ms
DONE	In the single receiving mode, DONE is the flag bit of receiving success.
ERR	Receive timeout flag.
DATA	The first address where the last received message data is stored.
LEN	The data length of the latest message received, unit: byte.

Note: ID, FMT, MODE and TIME parameters must be constant or variable at the same time; DATA and LEN parameters form a variable-length memory block, and this memory block must all be located in the legal memory area, otherwise the result is unpredictable.

The PLC automatically maintains a received message filtering list. The CPU will filter the received CAN messages, and only the messages whose message ID and format (standard frame or extended frame) match the list value will be received by this instruction.

The CAN_RX instruction is used to add a message waiting to be received to the received message

filtering list, and the message is determined by the specified ID and FMT.

The MODE parameter indicates the receiving mode. If the MODE is 1, it is a single receiving mode. The instruction only receives the specified message once, and exits after receiving it. If MODE is 0, it is permanent receiving mode, and the instruction will always receive the specified message.

A rising edge transition at the EN input triggers the execution of the instruction. After this instruction is executed, the specified ID number (ID) and format (FMT) value will be added to the receiving filter list immediately, and the PLC will immediately enter the receiving state, and at the same time clear DONE and ERR to 0. If it is a single receiving mode, if the specified message is received within the time TIME, set DONE to 1, and the instruction exits the receiving state. If the specified message is not received within the time TIME, then DONE and ERR are set to 1, and the instruction exits the receiving state. If it is in the permanent receiving mode, then this instruction will monitor the CAN interface CH and receive all the specified messages after it is started. If the specified message is not received again within TIME after a successful reception, ERR will be set to 1. After that, if it is successfully received again, ERR will be cleared to 0.

In a project, a CAN_RX instruction only needs to be executed once. In addition, in a project, a maximum of 64 CAN_RX instructions are allowed!

- LD

The rising edge of EN will trigger the execution of the instruction and enter the receiving state, otherwise it will not be executed.

10.5.7.2.6 Example

The following will illustrate the use of the CAN free communication function with an example.

一、An example of CAN free communication between PLC and inclination sensor to read the inclination angle value

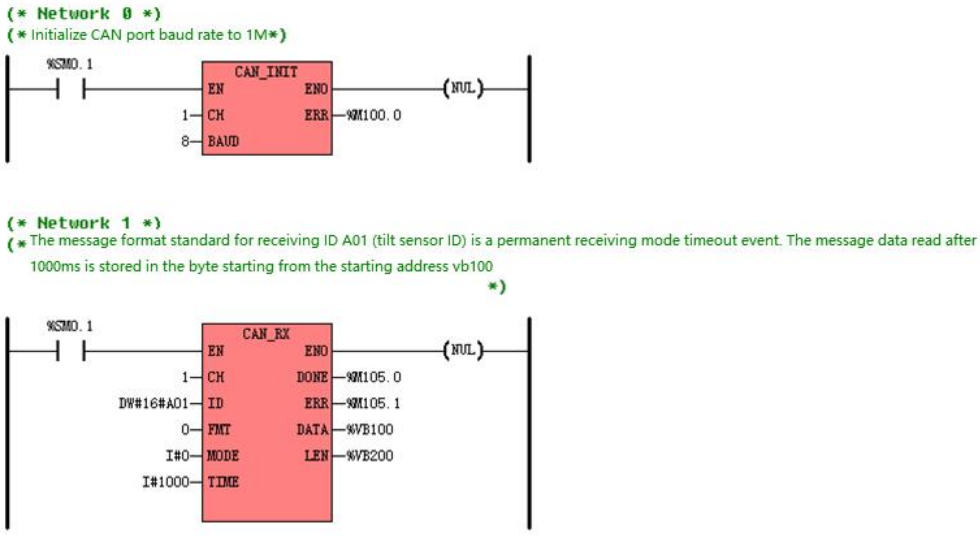
1、Power-on initialization, use the CAN_INIT instruction to initialize the CAN port and set the baud

rate.

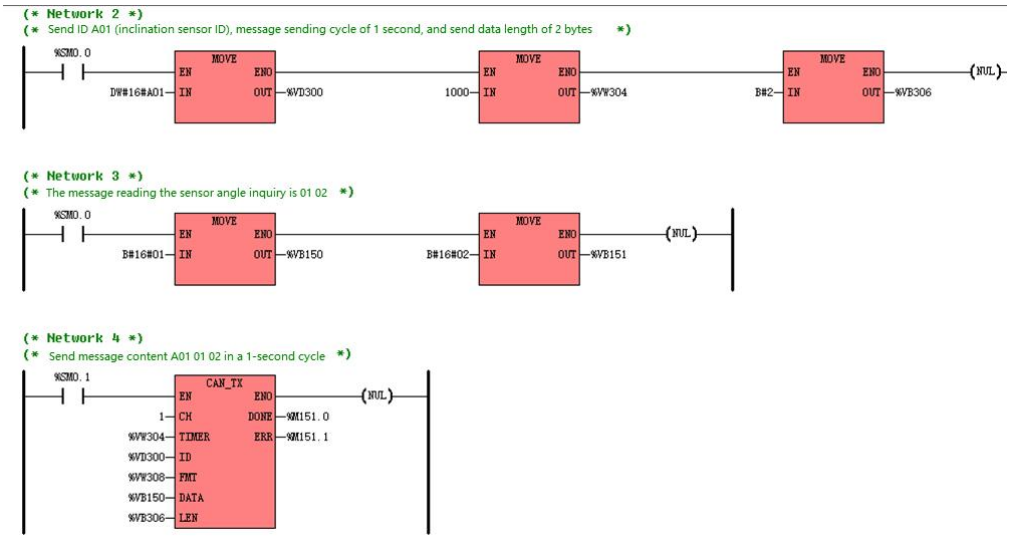
2、Set the parameters of the CAN_RX instruction, specify the ID of the received message as 0xA01 (inclination sensor ID), enter the permanent receiving mode, and set the timeout period. When the CPU is working normally, it will always receive the message with ID 0xA01, and the data information of the message is stored in the byte starting from the start address vb100, and the user can analyze the received data information according to the custom protocol of the inclination sensor.

3、Set the parameters of the CAN_TX instruction, the sending cycle is 1s, and the sending message ID is 0xA01 (inclination sensor ID). The length of the sent data is 2 bytes, and the content of the sent message is 01 02 (representing the meaning of reading the angle value of the inclination sensor), that is, sending a message to read the angle value of the inclination sensor every 1s.

LD



LD



IL

```
(* Network 0 *)
(*Initialize CAN port baud rate to 1M *)
LD     %SM0.1
CAN_INIT 1, 8, %M100.0
(* Network 1 *)
(*Receive ID A01 message timeout of 1000ms *)
LD     %SM0.1
CAN_RX  1, DW#16#A01, 0, I#0, I#1000, %M105.0, %M105.1, %VB100, %VB200
(* Network 2 *)
(*Send ID confirmation, send cycle of 1 second, data length of 2 bytes *)
LD     %SM0.0
MOVE   DW#16#A01, %VD300
MOVE   1000, %VW304
MOVE   B#2, %VB306
(* Network 3 *)
(*Read the angle message of the tilt sensor as 01 02*)
LD     %SM0.0
MOVE   B#16#01, %VB150
MOVE   B#16#02, %VB151
(* Network 4 *)
(*Send message content A01 01 02 in a 1-second cycle*)
LD     %SM0.1
CAN_TX  1, %VW304, %VD300, %VW308, %VB150, %VB306, %M151.0, %M151.1
```

(2)、Example of KS101M and KS105C2 networking through CAN free communication

1、In KS101M, the CAN1 port is first initialized and the baud rate is set to 1M. After power-on, it enters the permanent receiving mode to receive messages with specified ID201 and 202 (sent by KS105C2), and send messages with specified ID101 and 102. The data in the ID101 message contains 8 bytes in length, that is, two double words VD100 and VD104. The data in the ID102 message contains 4 bytes, that is, a double word VD120, where VD100 adds 1, VD104 adds 2, and VD120 adds 3. The messages monitored by the USB to CAN tool are as follows:

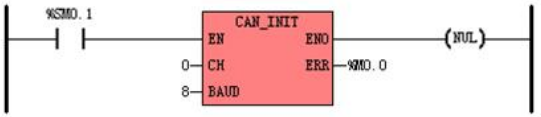
序号	系统时间	时间标识	CAN通道	传输方向	ID号	帧类型	帧格式	长度	数据
00000	09:29:05.672	0x505CF	ch2	接收	0x0101	数据帧	标准帧	0x08	x 01 00 00 00 02 00 00 00
00001	09:29:05.672	0x505D0	ch2	接收	0x0102	数据帧	标准帧	0x04	x 03 00 00 00
00002	09:29:06.513	0x525DD	ch2	接收	0x0101	数据帧	标准帧	0x08	x 02 00 00 00 04 00 00 00
00003	09:29:06.513	0x525DE	ch2	接收	0x0102	数据帧	标准帧	0x04	x 06 00 00 00
00004	09:29:07.502	0x54CD4	ch2	接收	0x0101	数据帧	标准帧	0x08	x 02 00 00 00 04 00 00 00
00005	09:29:07.502	0x54CD5	ch2	接收	0x0102	数据帧	标准帧	0x04	x 06 00 00 00
00006	09:29:07.502	0x54CE2	ch2	接收	0x0101	数据帧	标准帧	0x08	x 03 00 00 00 06 00 00 00
00007	09:29:07.502	0x54CE3	ch2	接收	0x0102	数据帧	标准帧	0x04	x 09 00 00 00
00008	09:29:08.493	0x573D6	ch2	接收	0x0101	数据帧	标准帧	0x08	x 03 00 00 00 06 00 00 00
00009	09:29:08.493	0x573D7	ch2	接收	0x0102	数据帧	标准帧	0x04	x 09 00 00 00
00010	09:29:08.493	0x573E4	ch2	接收	0x0101	数据帧	标准帧	0x08	x 04 00 00 00 08 00 00 00
00011	09:29:08.493	0x573E5	ch2	接收	0x0102	数据帧	标准帧	0x04	x 0C 00 00 00
00012	09:29:09.511	0x59ADB	ch2	接收	0x0101	数据帧	标准帧	0x08	x 04 00 00 00 08 00 00 00
00013	09:29:09.511	0x59ADC	ch2	接收	0x0102	数据帧	标准帧	0x04	x 0C 00 00 00
00014	09:29:09.511	0x59AE7	ch2	接收	0x0101	数据帧	标准帧	0x08	x 05 00 00 00 0A 00 00 00

Through the message, you can clearly see the data changes in the message in the specified ID number.

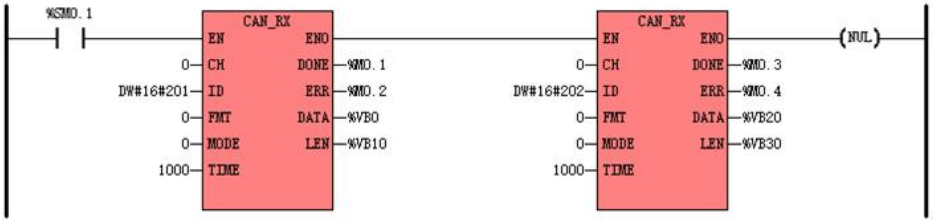
2、The program of KS101M is as follows:

LD

```
(* Network 0 *)
(* Initialize CAN1 port, baud rate 1M *)
```



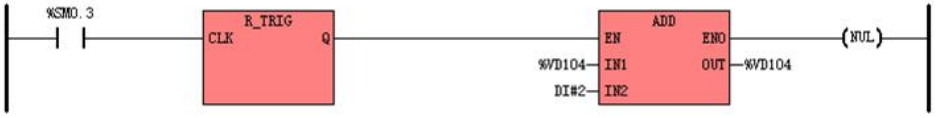

```
(* Network 1 *)
(* After the PLC is powered on, it enters permanent receiving mode, and the background automatically receives standard frame
   messages with specified IDs 201 and 202, without the need for the user to trigger this command again *)
```



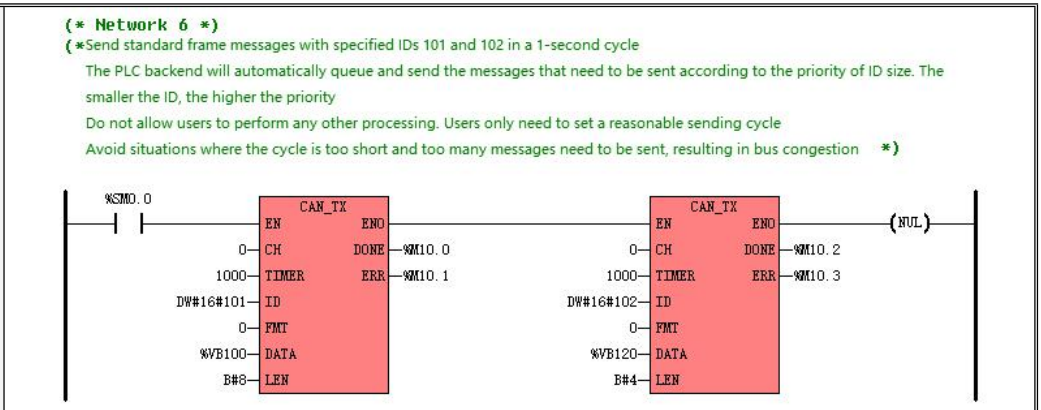
LD

```
(* Network 3 *)
(* Vd100 and vd104 are 8 bytes in length from sending ID101, with self increment 1 and self increment 2 *)
```

```
(* Network 4 *)
```




```
(* Network 5 *)
(* Vd120 is the length of 4 bytes in sending ID102, plus 3 bytes *)
```



IL

```
(* Network 0 *)
(*Initialize can1 port with a baud rate of 1M *)
LD     %SM0.1
CAN_INIT 0, 8, %M0.0
(* Network 1 *)
(*After the PLC is powered on, it enters permanent receiving mode, and the background automatically receives standard frame messages with specified IDs 201 and 202, without the need for the user to trigger this command again *)
LD     %SM0.1
CAN_RX  0, DW#16#201, 0, 0, 1000, %M0.1, %M0.2, %VB0, %VB10
CAN_RX  0, DW#16#202, 0, 0, 1000, %M0.3, %M0.4, %VB20, %VB30
(* Network 2 *)
LD     %SM0.0
ST     %BR0.0
(* Network 3 *)
(*Vd100 and vd104 are 8 bytes in length from sending ID101, with self increment 1 and self increment 2 respectively*)
LD     %SM0.3
R_TRIG
ADD    DI#1, %VD100
(* Network 4 *)
LD     %SM0.3
R_TRIG
ADD    DI#2, %VD104
(* Network 5 *)
(*Vd120 is the length of 4 bytes in sending ID102, plus 3 bytes*)
LD     %SM0.3
R_TRIG
```

	ADD DI#3, %VD120
IL	(* Network 6 *) (*Send standard frame messages with specified IDs 101 and 102 in a 1-second cycle. The PLC backend will automatically queue and send the messages that need to be sent according to the priority of the ID size. The smaller the ID, the higher the priority. Users do not need to perform any other processing. They only need to set a reasonable sending cycle to avoid situations where the cycle is too short, too many messages need to be sent, and bus congestion occurs *) LD %SM0.0 CAN_TX 0, 1000, DW#16#101, 0, %VB100, B#8, %M10.0, %M10.1 CAN_TX 0, 1000, DW#16#102, 0, %VB120, B#4, %M10.2, %M10.3

3、In KS105C2, first initialize the CAN1 port and set the baud rate to 1M. After power on, it enters the permanent receiving mode to receive messages with specified ID101 and 102 (sent by KS101M), and send messages with specified ID201 and 202. The data in the ID201 message contains 8 bytes in length, that is, two double words VD100 and VD104. The data in the ID202 message contains 4 bytes in length, that is, a double word VD120, where VD100 adds 1, VD104 adds 2, and VD120 adds 3. The messages monitored by the USB to CAN tool are as follows:

序号	系统时间	时间标识	CAN通道	传输方向	ID号	帧类型	帧格式	长度	数据
00000	09:46:10.143	0xE3402	ch2	接收	0x0201	数据帧	标准帧	0x08	x 01 00 00 00 02 00 00 00
00001	09:46:10.143	0xE3403	ch2	接收	0x0202	数据帧	标准帧	0x04	x 03 00 00 00
00002	09:46:11.133	0xE5AF7	ch2	接收	0x0201	数据帧	标准帧	0x08	x 01 00 00 00 02 00 00 00
00003	09:46:11.133	0xE5AF8	ch2	接收	0x0202	数据帧	标准帧	0x04	x 03 00 00 00
00004	09:46:11.133	0xE5B07	ch2	接收	0x0201	数据帧	标准帧	0x08	x 02 00 00 00 04 00 00 00
00005	09:46:11.133	0xE5B08	ch2	接收	0x0202	数据帧	标准帧	0x04	x 06 00 00 00
00006	09:46:12.121	0xE81F9	ch2	接收	0x0201	数据帧	标准帧	0x08	x 02 00 00 00 04 00 00 00
00007	09:46:12.121	0xE81FA	ch2	接收	0x0202	数据帧	标准帧	0x04	x 06 00 00 00
00008	09:46:12.121	0xE8209	ch2	接收	0x0201	数据帧	标准帧	0x08	x 03 00 00 00 06 00 00 00
00009	09:46:12.121	0xE820A	ch2	接收	0x0202	数据帧	标准帧	0x04	x 09 00 00 00
00010	09:46:13.144	0xEA8FD	ch2	接收	0x0201	数据帧	标准帧	0x08	x 03 00 00 00 06 00 00 00
00011	09:46:13.144	0xEA8FE	ch2	接收	0x0202	数据帧	标准帧	0x04	x 09 00 00 00
00012	09:46:13.144	0xEA909	ch2	接收	0x0201	数据帧	标准帧	0x08	x 04 00 00 00 08 00 00 00
00013	09:46:13.144	0xEA90A	ch2	接收	0x0202	数据帧	标准帧	0x04	x 0C 00 00 00
00014	09:46:14.132	0xED002	ch2	接收	0x0201	数据帧	标准帧	0x08	x 04 00 00 00 08 00 00 00

Through the message, you can clearly see the data changes in the message in the specified ID number.

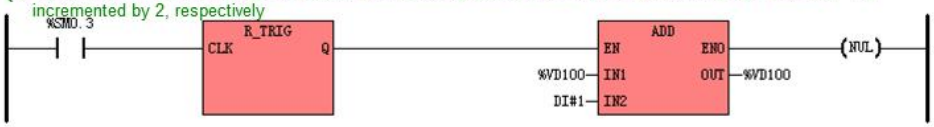
4、The program of KS105C2 is as follows:

LD

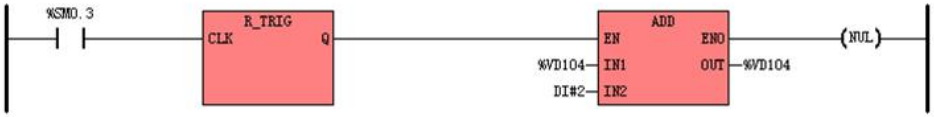
```
(* Network 0 *)
(* Initialize can1 port with a baud rate of 1M *)
%SM0.1
CAN_INIT
EN
ENO
0 CH
ERR %M0.0
8 BAUD
(NUL)

(* Network 1 *)
(* After the PLC is powered on, it enters permanent receiving mode, and the background automatically receives standard
frames with specified IDs other than 101 and 102 to report gratitude. There is no need for the user to trigger this
command again
*)
%SM0.1
CAN_RX
EN
ENO
0 CH
DONE %M0.1
DW#16#101 ID
ERR %M0.2
0 FMT
DATA %VB0
0 MODE
LEN %VB10
1000 TIME
CAN_RX
EN
ENO
0 CH
DONE %M0.3
DW#16#102 ID
ERR %M0.4
0 FMT
DATA %VB20
0 MODE
LEN %VB30
1000 TIME
(NUL)
```

LD

```
(* Network 3 *)
(* Vd100 and vd104 are the length of 8 bytes in the sending ID201, which are self incremented by 1 and self *)
   incremented by 2, respectively
```

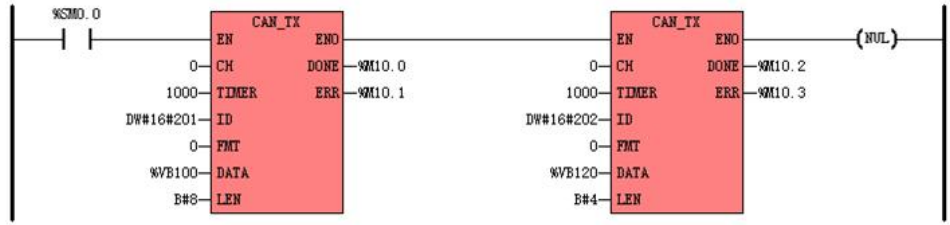



```
(* Network 4 *)
```




```
(* Network 5 *)
(*  Vd120 is the length of 4 bytes in the sending ID202, plus 3 bytes  *)
```

```
(* Network 6 *)
(* Send standard frame messages with specified IDs 201 \ 202 in a 1-second cycle
   The PLC backend will automatically queue and send the messages that need to be sent according to the
   priority of ID size. The smaller the ID, the higher the priority
   No further processing is required by the user, they only need to set a reasonable sending cycle
   Avoid situations where the cycle is too short and too many messages need to be sent, resulting in bus
   congestion  *)
```



IL	(* Network 0 *)
	(*Initialize can1 port with a baudrate of 1M *)
	LD %SM0.1
	CAN_INIT 0, 8, %M0.0
	(* Network 1 *)
	(*After the PLC is powered on, it enters permanent receiving mode, and the background automatically receives standard frame messages with specified IDs 101 and 102, without the need for the user to trigger this command again *)
	LD %SM0.1
	CAN_RX 0, DW#16#101, 0, 0, 1000, %M0.1, %M0.2, %VB0, %VB10
	CAN_RX 0, DW#16#102, 0, 0, 1000, %M0.3, %M0.4, %VB20, %VB30
	(* Network 2 *)
	LD %SM0.0
	ST %BR0.0
	(* Network 3 *)
	(*Vd100 and vd104 are the length of 8 bytes in the sending ID201, which are self incremented by 1 and self incremented by 2, respectively*)
	LD %SM0.3
	R_TRIG
	ADD DI#1, %VD100
	(* Network 4 *)
	LD %SM0.3
	R_TRIG
	ADD DI#2, %VD104
	(* Network 5 *)
	(*Vd120 is the length of 4 bytes in the sending ID202, plus 3 bytes*)
	LD %SM0.3
	R_TRIG
	ADD DI#3, %VD120

IL	(* Network 6 *) (*Send standard frame messages with specified IDs 201 and 202 in a 1-second cycle. The PLC backend will automatically queue and send the messages that need to be sent according to the priority of the ID size. The smaller the ID, the higher the priority. Users do not need to perform any other processing. They only need to set a reasonable sending cycle to avoid situations where the cycle is too short and there are too many messages that need to be sent, causing bus congestion *) LD %SM0.0 CAN_TX 0, 1000, DW#16#201, 0, %VB100, B#8, %M10.0, %M10.1 CAN_TX 0, 1000, DW#16#202, 0, %VB120, B#4, %M10.2, %M10.3

5、When a CAN communication problem occurs, first check the external causes: whether the wiring is correct, whether terminal resistors are added to the nodes at both ends, and external interference and other factors. If there is no external problem, you can go to find the program problem, whether the corresponding physical wiring CAN port is selected when the several communication devices use the initialization CAN

port instruction, whether the baud rate is the same, and so on.

6、The receiving instruction CAN_RX in CAN free communication automatically enters the receiving state in the background after being executed once, and the user does not need to do other processing. Send the instruction CAN_TX to periodically send the specified ID message. The background will queue up all the packets that need to be sent with ID priority (following the principle that the smaller the ID, the higher the priority), the user needs to set a reasonable sending cycle and number of packets to avoid bus congestion.

10.5.8 Kinco motion control function

10.5.8.1 Overview

The Kinco motion control function is used to control Kinco's motion control products (servo and stepper drives) with CAN interface. Based on the CANOpen protocol, it encapsulates the details of the CANOpen communication with the drive, etc., and combines the actual application requirements to provide users with a set of motion control instructions and corresponding network configuration tools. This function is easy to use, even if the user is not familiar with the details of the CANOpen protocol, he can easily communicate with the driver and perform position control.

This function supports parameter upload (download), motor shaft lock, loose shaft, homing, jog (speed mode), absolute positioning, relative positioning and other operations for motion control products. Torque mode and master-slave follow mode are not supported for the time being. operate. In addition, this function can be used in all third-party motion control products that support the standard CANOpen protocol in principle, please consult Kinco technical personnel before use.

The number of motion control axes that can be controlled by different types of CPUs with this function is different, please refer to the table below. In practical applications, the user can determine the actual number of connected units according to the required program space and network load rate.

Type	Maximum number of connected axes	Maximum number of MC instructions called in the user program
K209M		
KS101M	16	192
K6		
KS(except KS101M)	16	192

KW

➢ CAN bus load rate

The CAN bus load rate refers to the ratio of the number of bits actually transmitted on the bus per unit time to the maximum number of bits that can be transmitted theoretically.

The CAN bus load rate refers to the ratio of the number of bits actually transmitted on the bus per unit time to the maximum number of bits that can be transmitted theoretically.

The higher the load rate, the busier the CAN bus, and the more data transmitted per unit time. Usually, in practical applications, the industry generally requires that the load rate of the CAN bus does not exceed 30%. When this value is exceeded, the probability of transmission delay of low-priority messages will increase—it may even never get the chance to be sent, which may lead to abnormal control of some device nodes in the bus network. But the value of "30%" is just an empirical value with a long history. If the network and nodes can be optimized to meet the response time requirements of each node, the CAN bus can still guarantee normal communication when the load rate exceeds 70%.

KPLC provides the following load rate registers for the CAN2 interface:

SMW122	CAN2 load rate minimum	Load factor = Register value × 100
SMW124	CAN2 load rate maximum	

The user can directly read the register value in the program to get the real-time bus load rate of the CAN2 interface. If the user finds that some node devices in the CAN network occasionally behave abnormally in practical applications, the real-time load rate can be read to determine whether the low-priority message may be lost due to the high bus load. If the bus load rate is too high, the number of connected nodes in the network should be reduced.

NOTE: The CAN bus has a bit stuffing mechanism. As long as the CAN controller chip detects that there are 5 consecutive identical bits in the bit stream when sending, it will automatically insert a

complementary code bit in the back, and the receiver will automatically delete the inserted bit. When CAN sends different data, the phenomenon of bit stuffing may or may not occur, this is done automatically by the CAN controller chip, and the external software cannot know this information. Therefore, when calculating the load rate, KPLC calculates a minimum and maximum value of the load rate according to the two conditions of each frame message without bit filling and bit filling, and the real load rate should be between these two values.

➢ How to use Kinco motion control function

Users can follow the steps below to use the Kinco motion control function:

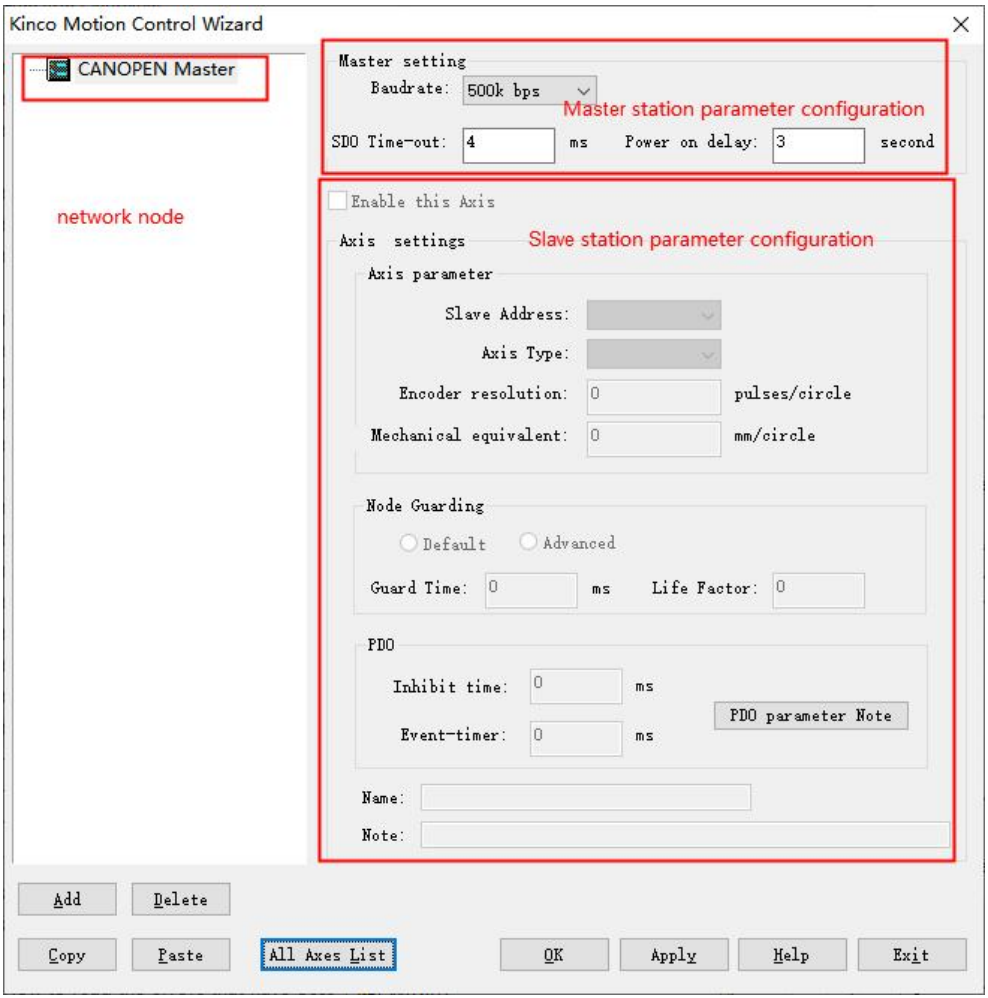
1) In the user project, enter the [Kinco Motion Control Network Configuration] wizard window and complete the basic configuration of the network and axis parameters.
2) Call motion control instructions for programming according to actual needs.
3) Download the project to the PLC, then the PLC will run as the master station after startup, manage the communication of the entire network, and execute the position control program.

10.5.8.2 Kinco motion control network configuration

The motion control function of Kinco is based on the CANOpen protocol, with the PLC as the master station and each driver as the slave station. Before calling the instruction, the user must configure the actual CANOpen network. According to the habit of field application, we call the slave station "axis" in the software.

In the [Project Manager] of the KincoBuilder software, double-click the [Kinco Motion Control

Network Configuration] node to enter the configuration window, and complete the network configuration in this window.



The window is divided into three parts: the tree list of network nodes, the parameters of the master station and the parameters of the axis (slave station).

1) Operation of network node tree

In the network node tree, the root node is [CANOpen master station], and each child node below is the axis (slave station) in the network.

There are four buttons [Add], [Delete], [Copy] and [Delete] below, and the software also provides corresponding shortcut keys and right-click menu functions. Users can use these functions to operate network nodes.

- Add a new axis

Click the [Add] button; or right-click on any node, and then execute the [Add] menu instruction; or use the ALT+N shortcut key. The newly added axes using the above three methods all adopt the default parameters at the beginning.

- copy and paste

The user can first copy an existing axis, and then paste it into the network to generate a new axis. Except for the axis number (slave station address), other parameters of the new axis are consistent with the copied axis. This is convenient for applications where all axes function the same.

First click an axis in the tree to select it, and then click the [Copy] button, or use the shortcut key of Ctrl+C; or right-click an axis, and execute the [Copy] menu instruction. There are several ways to duplicate this axis.

After copying, click the [Paste] button, or use the shortcut key of Ctrl+P, or right-click any axis and execute the [Paste] menu instruction to generate a new axis in the network.

- Delete an axis

First click an axis to select it, and then click the [Delete] button, or use the DELETE shortcut key to delete the axis.

Right-click an axis and execute the [Delete] menu instruction to delete the axis

2) Master station parameters

Click the [CANOpen master station] node, all parameters of the master station will be modifiable, and all parameters of the axis (slave station) will be grayed out and cannot be modified.

- [Baud rate]: Select the baud rate used by the master station. Note that the baud rates of all nodes (master station and slave station) on the network must be consistent.
- [SDO Timeout]: The timeout waiting time after the master station PLC sends the SDO request message. If the response message from the corresponding slave station is not received beyond this time, a timeout

error will be reported. When selecting a different baud rate, the software will automatically recommend a SDO timeout time, and the user can modify it based on this value.

3) Axis (slave station) parameters

Click on an axis node, all parameters of the axis will be modifiable, and all parameters of the master station will be grayed out and cannot be modified.

（1）[Axis No.]: CANOpen slave address of the axis, **the slave station numbers in this system must be assigned continuously from 1.**

（2）[Type]: Depending on the function of the axis, the user can choose a linear axis or a rotary axis.

（3）[Encoder resolution]: The resolution of the encoder of the shaft or stepper driver, that is, the number of pulses sent by the encoder for one revolution.

（4）[Mechanical equivalent per revolution]: The length (linear axis, mm) or the angle of rotation (rotation axis, °) that the mechanical load moves for each revolution of the motor shaft.

（5）[Node Guard]: Set the node guard time of this axis. Users can adopt the default value, or click "Advanced" to modify it by themselves.

（6）[PDO Prohibition Time]: Multiple PDOs are automatically established for each axis in the PLC to transmit position, speed, status and other information. Because the position and speed of the axis change rapidly, the PDO transmission is very frequent, and the PDO prohibition time must be set. Users can adopt the default value or modify it by themselves.

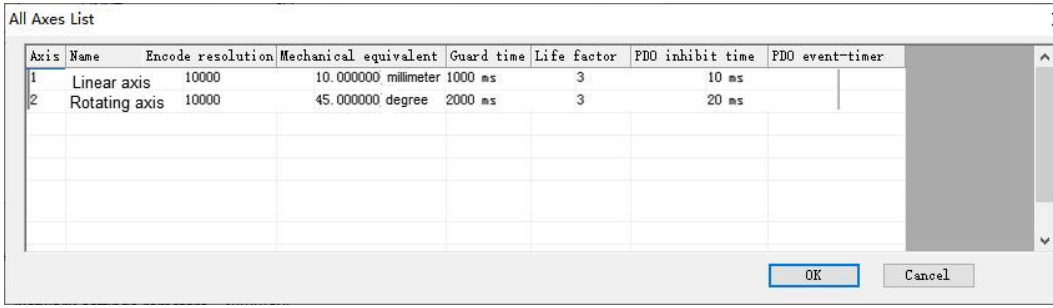
4) Other operations

（1）【OK】: Save the parameters configured in the current interface and exit the interface

（2）【Cancel】: Only save the parameters configured in the current interface and have been clicked to apply, and then exit the interface

（3）【Application】: Save the parameters configured in the current interface

（4）【Axis list】: The axis list is mainly used to view the parameters of all configured and enabled axis configurations for checking



10.5.8.3 Kinco Motion Control Instructions

10.5.8.3.1 Overview of Motion Control Instructions

The following instructions are located in the [Kinco Motion Control] group of the instruction set.

Name	Function description
MC_RPARAS	Read the parameters in the servo driver (see the parameter table below for details)
MC_WPARAS	Modify the parameters in the servo driver
MC_POWER	Control lock shaft, loose shaft
MC_RESET	Reset the error message on the axis and set the axis state to stand still and wait
MC_HOME	Control axis back to origin
MC_JOG	Control axis jog
MC_MABS/ MC_MABSE	Control axes for absolute positioning movements
MC_MREL/ MC_MRELE	Control axis for relative positioning movement
MC_STATE	Read the status values of the drive
MC_RESTART	Reconfigure and start the slave
MC_MIOT	Read the serial number, software version, IIT, temperature and other equipment information of the target axis

2) Precautions

When using these instructions, users need to pay attention to the following points:

- In a user project, please refer to 10.5.8.1 Overview for the maximum number of axes allowed and the maximum number of dedicated instructions used.
- For the same axis, when a dedicated instruction is being executed but not completed, it is not allowed to

start executing another dedicated instruction. If the user program starts another dedicated instruction at this time, then this instruction will end directly and report an error.

- For the same axis, before the user program executes motion commands (excluding read and write parameter commands), the MC_POWER command must be executed first to lock the axis. After the axis is locked successfully, the homing, relative movement, absolute movement or jog command can be continued. If there is no shaft lock, the execution of these types of commands will end directly and report an error.
- For the same axis, the user program resets with the MC_RESET instruction. After the reset is successful, the axis will be in the static waiting state of the loose axis. It is necessary to execute the MC_POWER instruction to lock the axis before continuing to execute the homing, relative movement, absolute movement or jog instructions.
- For homing, relative movement, absolute movement or jog commands, the acceleration and deceleration used are the acceleration and deceleration set inside the driver, and the user can also set it through the MC_WPARAS command.
- The output results of calling individual motion control instructions in the user program are independent of each other. If an instruction executes in error, its output parameter ERRID will give an error code. And this error result will not be refreshed again until the next instruction is executed again, and the execution results of other instructions will not affect the execution result of this instruction!
- After a node goes offline (the ONLINE output of the MC_STATE command is 1), for safety reasons, the PLC will not automatically reconnect to the node! After the user has eliminated the error, power off and restart the PLC or call the MC_RESTART command to re-establish the connection!
- For the same axis, this function does not support simultaneous execution of multiple motion control actions. At the same time, there may be a certain delay in the DONE signal output of this group of commands. If the user uses the DONE signal as the basis for judging the mutual action interlocking of each command, the possible delay must be considered and dealt with, otherwise there may be results that do not conform to the expected process!

3) Command output parameter ERRID

Each command provides an ERRID output parameter. If the instruction is executed successfully, the ERRID output is 0. If the command execution fails, ERRID will be set to a different error code value to explain the cause of the error.

The following is a description of each error code value. **Note that the error codes here are not suitable for the MC_RPARAS and MC_WPARAS instructions. The error codes of these two instructions have special meanings,** please refer to the instructions for instructions.。

Error Code	Description
0	no error
1	The target axis is not enabled, or the axis does not exist in the network
2	The target axis is not locked.
3	The target axis is executing another motion control instruction and is not standstill.
4	The CAN message sending buffer inside the PLC is full, and the CAN message cannot be sent
5	The PLC sent an SDO request message to the target axis, but did not receive a response after timeout
6	The PLC sent an SDO request message to the target axis, but received a wrong response message
7	The instruction is executed normally, but the PLC continues to detect the status word returned by the target axis, and finally does not detect the correct state value.
8	The MC_REL. MC_RELE . MC_ABS or MC_ABSE instruction has finished executing, but the 'Position Reached' bit in the status word from the target axis is wrong.
9	The target servo drive did not find excitation.

In motion control, the axis is divided into several logic states, and the transition between logic states requires specific conditions or specified MC operation control instructions. The advantage of dividing the processing in this way is that it is convenient for the axes to be controlled according to the movement mode. The axes can only be in one logical state at a time, and the transfer of the logical state needs to be carried out according to the rules, so that the operation will not be confused due to false triggers of different MCs

The logic state of the axis is divided into the following six types:

0: Power_off (Disabled): The axis is not powered on or enabled, and the MC_Power command needs to be executed

1: Errorstop: The axis is in the fault stop state, and the MC_Reset instruction needs to be executed

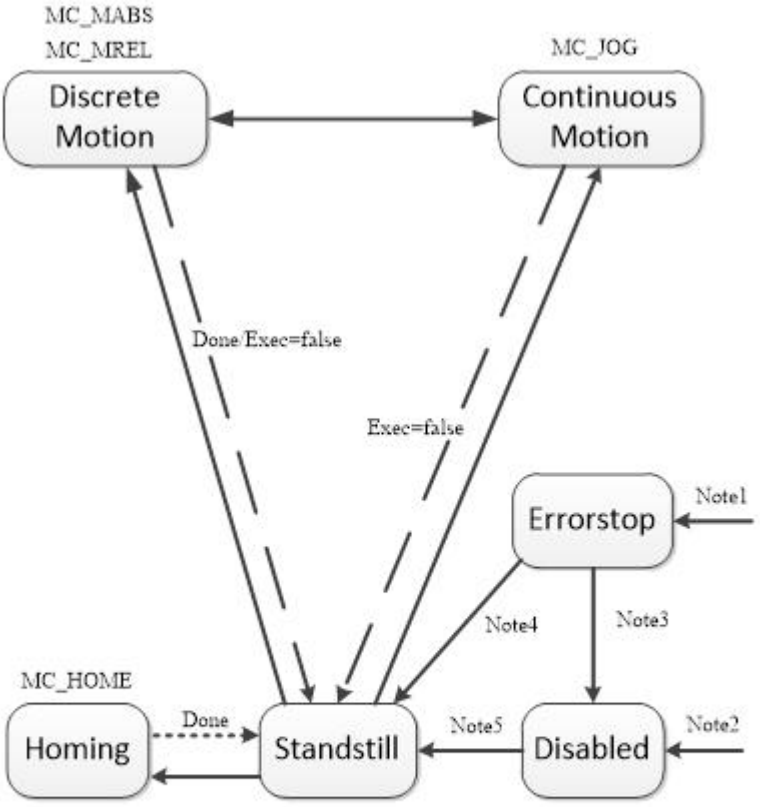
2: Standstill: The Axis is at rest

3: Discrete_Motion: The axis is in discrete operating state

4: Continuous_Motion:The axis is in continuous operation

5: Homing: The axis is in the zero return running state, waiting for the zero return operation to complete

The axis state transition diagram is as follows:



Note1: From any state, An error in the axis occurred.
Note2: From any state, MC_power.exec=false and there is no error in the axis.
Note3: MC_reset and MC_power.exec=false and there is no error in the axis.
Note4: MC_reset and MC_power.exec=true and there is no error in the axis.
Note5: MC_power.exec=true.

10.5.8.3.2 Motion Control Instructions

10.5.8.3.2.1 MC_RPARAS (read parameter)& MC_WPARAS (Change parameters)

The purpose of this group of instructions is to facilitate the user to operate the drive parameters in batches, for example, the user can set the drive parameters at one time at the initial stage of debugging. Please refer to the driver operation manual for how to set the specific parameters. **Improper setting may cause abnormal operation. Please operate with caution.**

Note: Parameters not in the list below can be read and written by calling the SDO command to send the SDO message. For the instructions of the SDO command, please refer to 10.5.4.5.2 SDO command.

1) List of operable drive parameters

Through the drive read and write commands, the following parameters of the drive can be operated, and all parameters can be read and written. Each command operates on up to 32 parameters at a time. The process data type in the table, REAL means single-precision floating-point number, UINT32 means unsigned 32-bit number, INT32 means signed 32-bit number, and so on.

The "serial number" value in the table is fixed, each parameter has a serial number, and the user can operate the corresponding parameter by inputting the serial number in the command. "Technical units" refers to the units used in the command parameters. "Drive value range" refers to the internal value range of the drive (this instruction will automatically convert the actual process parameter value required by the user into the data format used inside the drive, such as acceleration, speed, position, etc.).

Serial number	Parameters	CANOpen objectives	Type of technical data	Technical units	Drive value range
0	**trapezoidal acceleration**	0x60830020	REAL	linear axis: mm/s² Rotation axis: 1/s²	[0,268435455]
1	**trapezoidal deceleration**	0x60840020			

2	**find origin speed**	0x60990120	REAL	linear axis: mm/min Rotation axis: Degree/min	[-2147483648, 2147483647]
3	**find origin mode**	0x60980008	INT8	DEC	[-128,127]
4	**Speed loop proportional gain 0**	0x60F90110	UINT16	DEC	[0,32767]
5	**Speed loop integral gain 0**	0x60F90210	UINT16	DEC	[0,32767]
6	**Position loop proportional gain 0**	0x60FB0110	REAL	HZ	[0,32767]
7	**Position loop speed feedforward**	0x60FB0210	REAL	%	[0,1024]
8	**Speed loop proportional gain 1**	0x23400410	UINT16	DEC	[0,32767]
9	**Speed loop integral gain 1**	0x23400510	UINT16	DEC	[0,32767]
10	**Position loop proportional gain 1**	0x23400610	REAL	HZ	[0,32767]
11	**target current limit**	0x60730010	UINT16	DEC	[0,2048]
12	**maximum speed limit**	0x607F0020	REAL	linear axis: mm/min Rotation axis: Degree/min	[-2147483648, 2147483647]
13	**Origin Offset Mode**	0x60990508	UINT8	No units	[0,255]
14	**Motor direction**	0x607E0008	UINT8	No units	0 and 1
15	**motor model**	0x64100110	UINT16	No units	[0,65535]
16	**Soft limit positive setting**	0x607D0120	REAL	linear axis: mm Rotation axis: Degree	[-2147483648, 2147483647]
17	**Soft limit negative setting**	0x607D0220			
18	**smoothing filter**	0x60FB0510	UINT8	No units	[0,255]
19	**maximum following error**	0x60650020	UINT32	DEC	[0,268435455]
20	**target position window**	0x60670020	UINT32	DEC	[0,268435455]
21	**location window**	0x60680010	UINT16	DEC	[0,32767]

	time				
22	**Velocity Feedback Filtering**	0x60F90508	REAL	HZ	[0,45]
23	**speed feedback mode**	0x60F90608	UINT8	No units	[0,85]
24	**Input Polarity**	0x20100110	UINT8	No units	[0,255]
25	**Input port 1 function**	0x20100310	UINT16	No units	[0,65535]
26	**Input port 2 function**	0x20100410	UINT16	No units	[0,65535]
27	**Input port 3 function**	0x20100510	UINT16	No units	[0,65535]
28	**Input port 4 function**	0x20100610	UINT16	No units	[0,65535]
29	**Input port 5 function**	0x20100710	UINT16	No units	[0,65535]
30	**Input port 6 function**	0x20100810	UINT16	No units	[0,65535]
31	**Input port 7 function**	0x20100910	UINT16	No units	[0,65535]
32	**Input port 8 function**	0x20101D10	UINT16	No units	[0,65535]
33	**Output Polarity**	0x20100D10	UINT8	No units	[0,255]
34	**Output port 1 function**	0x20100F10	UINT16	No units	[0,65535]
35	**Output port 2 function**	0x20101010	UINT16	No units	[0,65535]
36	**Output port 3 function**	0x20101110	UINT16	No units	[0,65535]
37	**Output port 4 function**	0x20101210	UINT16	No units	[0,65535]
38	**Output port 5 function**	0x20101310	UINT16	No units	[0,65535]
39	**Output port 6 function**	0x20101E10	UINT16	No units	[0,65535]
40	**Output port 7 function**	0x20101F10	UINT16	No units	[0,65535]
41	**Pre-gear pulse data**	0x25080420	INT32	DEC	[-2147483648, 2147483647]
42	**pulse mode**	0x25080308	UINT8	No units	[0,255]
43	**save parameters**	0x10100120	UINT32	No units	16#65766173
44	**Initialization parameters**	0x10110120	UINT32	No units	16#64616f6C

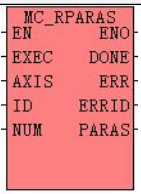
2) ERRID parameter description

Both read and write parameter commands provide ERRID (DWORD type) output parameter.

This parameter value is an error code, which indicates an error occurred during the execution of the instruction.

Error code	**Meaning**
0xFFFFFFFF	Errors have occurred that prevent instructions from being executed, including: 1) The axis number entered by the user is wrong, and the number of parameters is wrong 2) There are other Kinco special commands running 3) The instruction needs to operate 32 parameters, but all the 32 parameters fail to operate.
Other value	Each bit of ERRID represents the operation result of the corresponding parameter, and each bit corresponds to the parameter specified in the ID parameter serial number table. Bit0 indicates the result of the first parameter of this operation, bit1 indicates the operation result of the second parameter, and so on. A value of 1 indicates that the operation of the corresponding parameter fails, otherwise it indicates that the operation of the corresponding parameter succeeds.

3) MC_RPARAS (read parameter)

	Name	Instructions format	Adopt to
LD	MC_RPARAS		☑ KS ☑ KW ☑ K209M ☑ K6

Parameters	Input/Output	Data type	Memory area allowed	Description
EXEC	Input	BOOL	M、V、L、SM	If the rising edge of EXEC is detected, the instruction is triggered to execute.
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
ID	Input	BYTE	V、M、L	The starting address of the ordinal table of parameters to be read.
NUM	Input	INT	V、M、L、constant	number of parameters to read
DONE	Output	BOOL	M、V、L	Completion flag. DONE transitions from 0 to 1 when the instruction is executed.
ERR	Output	BOOL	M、V、L	Error flag. Set to 1 if an error occurs during instruction execution.

ERRID	Output	DWORD	V、M、L	error code
PARAS	Output	DWORD	V、M、L	The storage starting address of all parameter values read.

Note: AXIS and NUM must be both constant type or memory type at the same time. In addition, the ID and PAPAS parameters together form a variable-length memory block. This memory block must all be located in a legal memory area, otherwise the result is unpredictable .

The three parameters of ID, PARAS, and NUM together form a parameter table. Among them, ID is the starting address of the serial number table, and the serial numbers of the parameters to be operated are stored successively from this address (that is, the "serial number" in the previous parameter list). Each serial number occupies 1 byte. PARAS is the starting address of the parameter value table. From this address, the value of each parameter read is stored successively, and each value occupies 4 bytes. NUM is the number of parameters to operate on. For example, in the following example, assuming that the ID parameter is VB100, the PARAS parameter is VD1200, and the NUM parameter is 3, then VB100, VB101, and VB102 respectively store the serial numbers of the three parameters of this secondary operation. After the instruction is executed, the read 3 parameter values are stored in VD1200, VD1204, and VD1208 respectively.

For PARAS, it should be noted that although the parameter value table uniformly adopts DWORD addresses, the actual data types of each process parameter are not the same, so in the user program in the table, the user must process the data in the parameter table according to the actual data type.

- ✧ If the actual process data type is REAL type, then directly operate the parameter memory with the floating point number address. For example, in the following example, the serial number 0 of the first parameter value is a REAL parameter, stored in VD1200, then VR1200 can be directly operated. Because VD1200 and VR1200 actually occupy the same memory address in PLC.
- ✧ If the actual process data type is other data types than REAL, and the corresponding parameter memory is not mandatory to define the data type in the global variable table, then directly read the parameter memory. This is because the instruction automatically handles various signed and unsigned integers. For example, in the following example, the serial number of the third parameter value is 8 and stored in VD1208, but the actual type is INT32 or UINT32, then directly operate VD1208.

1) LD Format instruction description

If EN is 1, the instruction is triggered to execute on the rising edge of the EXEC input. According to the parameter table to be read specified by ID and NUM, the command sends SDO to the driver to read the corresponding object in sequence, and puts the read data into the value table specified by PARAS in turn, and sets the corresponding bit of ERRID to 0 . If the SDO response of a parameter is wrong or there is no response after timeout, the data of the corresponding address in PARAS remains unchanged, and the corresponding bit of ERRID is set to 1, and then continue to read the next parameter. When all parameters are read, DONE is set to 1, and ERR and ERRID are set to different values according to the execution results.

If EN is 0, the instruction will not be executed. When EXEC becomes 0 during the execution of the instruction, the instruction will stop reading the parameters that have not been completed, and set DONE to 1, and ERR and ERRID will maintain the executed results.

If the PLC detects an error when the command starts (such as the axis is not enabled, the axis is executing other commands, etc.), it will exit directly, set DONE, ERR to 1, and ERRID to the corresponding error code.

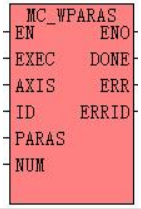
- **Example**

This example uses IL format. In KincoBuilder, first select [IL] format in the [Project] menu, then copy and paste the example into the editor, and then select [LD format], the program can be displayed in LD format.

```
(* Network 0 *)
(*Set the parameter table to indicate that parameters 0, 3, and 8 will be read this time*)
LD    %SM0.0
MOVE  B#0, %VB100
MOVE  B#3, %VB101
MOVE  B#8, %VB102
(* Network 1 *)
(*Call the instruction. This time, AXIS and NUM parameters are both constants, and they also support the format of full memory addresses.*)
```

```
LD      %M0.0
MC_RPARAS %M0.1, 1, %VB100, 3, %M0.2, %M0.3, %MD8, %VD1200
(* Network 2 *)
(*The read parameter values are stored in the parameter value table starting from the PARAS parameter (in this example, starting from 1200 in Zone V). The first data in the table is the value of the first parameter read, which is parameter 0 in the table. Since it is of the REAL type, it reads the floating-point memory address.*)
LD      %SM0.0
MOVE    %VR1200, %VR300
(* Network 3 *)
(*The second data in the table is the second parameter value read, which is parameter 3 in the table. This parameter has a signed 8-bit number and is processed as an integer because this data type is not provided in the PLC.*)
LD      %SM0.0
DI_TO_I %VD1204, %VW304
(* Network 4 *)
(*The third data in the table is the third parameter value read, which is parameter 8. This parameter is an unsigned 16 bit number, but the maximum range is 32767, so it can be processed as INT or WORD type in the program, but it is best to first check whether the value is within the allowable range.*)
LD      %SM0.0
DI_TO_I %VD1208, %VW308
NE      %VW308, 0
ST      %M3.0
```

4) MC_WPARAS (Change parameters)

	Name	Instructions format	Adopt to
LD	MC_WPARAS		☑ KS ☑ KW ☑ K209M ☑ K6

Parameters	Input/Output	Data type	Memory area allowed	Description
EXEC	Input	BOOL	M、V、L、SM	If the rising edge of EXEC is detected, the instruction is triggered to execute.
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
ID	Input	BYTE	V、M、L	The start address of the ordinal table of parameters to be modified.
PARAS	Output	DWORD	V、M、L	The storage starting address of all parameter values modified by reading.
NUM	Input	INT	V、M、L、constant	the number of parameters to modify
DONE	Output	BOOL	M、V、L	Completion flag. DONE transitions from 0 to 1 when the instruction is executed.
ERR	Output	BOOL	M、V、L	Error flag. Set to 1 if an error occurs during instruction execution.
ERRID	Output	DWORD	V、M、L	error code

Note: AXIS and NUM must be both constant type or memory type at the same time. In addition, the ID and PAPAS parameters together form a variable-length memory block. This memory block must all be located in a legal memory area, otherwise the result is unpredictable .

The three parameters of ID, PARAS, and NUM together form a parameter table. Among them, ID is the starting address of the serial number table. From this address, the serial numbers of the parameters to be operated (that is, the "serial number" in the previous parameter list) are stored successively, and each serial number occupies 1 byte. PARAS is the starting address of the parameter value table. From this address, the values of each parameter are stored successively, and each value occupies 4 bytes; NUM is the number of parameters to be operated. For example, in the following example, assuming that the ID parameter is VB200, the PARAS parameter is VD2000, and the NUM parameter is 3, then VB200, VB201, and VB202 respectively store the serial numbers of the three parameters of this secondary operation. VD2000, VD2004, and VD2008 respectively store the parameter values to be modified.

For PARAS, it should be noted that although the parameter value table uniformly adopts DWORD addresses, the actual data types of each process parameter are not the same, so in the user program in the table, the user should assign values to the corresponding addresses in the parameter table according to the actual data type.

✧ If the actual process data type is REAL type, then directly operate the parameter memory with the

floating point number address. For example, if the parameter value is expected to be stored in VD2000, then VR2000 can be operated directly. VD2000 and VR2000 actually occupy the same memory address in the PLC, and this command will automatically perform type conversion.

✧ If the actual process data type is other data types than REAL, then directly operate the parameter memory, and the instruction will automatically perform type conversion according to the data type of the parameter. For example, in this example, the data types of parameters 3 and 8 to be operated are UINT8 and UINT16, then directly assign a legal value to VD2004 and VD2008

1) LD Format instruction description

If EN is 1, the instruction is triggered to execute on the rising edge of the EXEC input. According to the parameter table specified by ID, PARAS, and NUM, the instruction sends the value in PARAS to the driver through SDO to modify the corresponding object in turn, and sets the corresponding bit of ERRID to 0 at the same time. If the SDO response of a parameter is wrong or there is no response after timeout, set the corresponding bit of ERRID to 1, and then continue to write the next parameter. When all parameters are written, DONE is set to 1, and ERR and ERRID are set to different values according to the execution results.

If EN is 0, the instruction will not be executed. If EN becomes 0 during the execution of the instruction, the instruction will stop writing unfinished parameters, and set DONE to 1, and ERR and ERRID will maintain the executed results.

If the PLC detects an error (such as the axis is not enabled, the axis is executing other instructions, etc.) when the instruction is started and executed, it will exit directly, set DONE, ERR to 1, and ERRID to the corresponding error code.

➢ **Example**

This example uses IL format. In KincoBuilder, first select [IL] format in the [Project] menu, then copy and paste the example into the editor, and then select [LD format], the program can be displayed in LD format.

```
(* Network 0 *)
```

```
(*Set the parameter table to indicate that parameters 0, 3, and 8 should be written this time.*)
LD    %SM0.0
MOVE  B#0, %VB200
MOVE  B#3, %VB201
MOVE  B#8, %VB202
(* Network 1 *)
(*Set the numerical values to be written for each parameter. Note that the data type and each parameter automatically occupy a 32-bit address. This example is equivalent to writing 1200.0 for parameter 0, 8 for parameter 3, and 2000 for parameter 8 in the table*)
LD    %SM0.0
MOVE  1200.0, %VR2000
MOVE  DI#8, %VD2004
MOVE  DI#2000, %VD2008
(* Network 2 *)
(*Call instruction *)
LD    %SM0.0
MC_WPARAS %M1.1, 1, %VB200, %VD2000, 8, %M1.2, %M1.3, %MD14
```

10.5.8.3.2.2 MC_POWER (Lock and Loose)

	Name	Instructions format	Adopt to
LD	MC_POWER		☑ KS ☑ KW ☑ K209M ☑ K6

Parameters	Input/Output	Data type	Memory area allowed	Description
EXEC	Input	BOOL	M、V、L、SM	The rising edge triggers the axis lock command, and the falling edge triggers the axis loose command.
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
DONE	Output	BOOL	M、V、L	Completion flag. DONE transitions from 0 to 1 when the instruction is executed.

ERR	Output	BOOL	M、V、L	Error flag. Set to 1 if an error occurs during instruction execution.
ERRID	Output	BYTE	V、M、L	error code

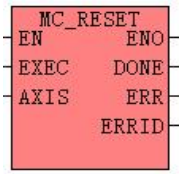
- LD Format instruction description

If EN is 1, then the rising edge of EXEC will trigger the execution of the axis lock command, and the falling edge of EXEC will trigger the execution of the axis loosening command.

When the instruction is executed, the PLC first sends a command to control the axis to enter the standby state and check the actual return state of the driver within the 5S timeout period. If it is successfully executed, it means that the instruction is executed successfully, then DONE is set to 1, ERR is set to 0, and ERRID is set to 0. If an error occurs (It may be an error in the execution of the command itself, or it may be an error that the driver did not perform the action correctly during the execution process, see the error code for details), the execution of the instruction will fail, and the execution of the instruction will stop. At the same time, DONE will be set to 1, ERR will be set to 1, and ERRID will be assigned the corresponding error code.。

If EN is 0, the instruction will not be executed.

10.5.8.3.2.3 MC_RESET (Reset drive alarm)

	Name	Instructions format	Adopt to
LD	MC_RESET		☑ KS ☑ KW ☑ K209M ☑ K6

Parameters	Input/Output	Data type	Memory area allowed	Description
EXEC	Input	BOOL	M、V、L、SM	The rising edge triggers this instruction to be executed once.
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
DONE	Output	BOOL	M、V、L	Completion flag. DONE transitions from 0 to 1 when the instruction is executed.

ERR	Output	BOOL	M、V、L	Error flag. Set to 1 if an error occurs during instruction execution.
ERRID	Output	BYTE	V、M、L	error code

When the axis makes an error during operation, this command can be called to reset the error information on the axis, and at the same time, set the axis to the loose axis static waiting state. **If you need to continue to execute other motion commands after the reset is successful, you should first call the MC_POWER command to lock the axis!**

Note: This command only resets the alarm error information of the drive, but does not reset the output results of each command!

- LD Format instruction description

  if EN is 1, then the rising edge of EXEC will trigger the execution of this instruction.

When the instruction is executed, the PLC first sends an instruction to reset the driver alarm, and checks the actual status of the driver within the 2-second timeout period. If it is successfully reset, it means that the instruction is executed successfully, then DONE is set to 1, ERR is set to 0, and ERRID is set to 0. If an error occurs (It may be an error in the execution of the command itself, or it may be an error that the driver did not perform the action correctly during the execution process, see the error code for details), the execution of the instruction will fail, and the execution of the instruction will stop. At the same time, DONE will be set to 1, ERR will be set to 1, and ERRID will be assigned the corresponding error code.

If EN is 0, the instruction will not be executed.

10.5.8.3.2.4 MC_HOME (Homing)

	Name	Instructions format	Adopt to
LD	MC_HOME		☑ KS ☑ KW ☑ K209M ☑ K6

Parameters	Input/Output	Data type	Memory area allowed	Description
EXEC	Input	BOOL	M、V、L、SM	Rising edge triggers this instruction to execute once;
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
POS	Input	REAL	V、M、L、constant	The offset position of the origin, unit: mm or °.
TIME	Input	DWORD	V、M、L、constant	Timeout time, if the origin is not found within this time, it will report an error and exit.
DONE	Output	BOOL	M、V、L	Completion flag. DONE transitions from 0 to 1 when the instruction is executed.
ERR	Output	BOOL	M、V、L	Error flag. Set to 1 if an error occurs during instruction execution.
ERRID	Output	BYTE	V、M、L	Error code

Execute this command to make the target axis return to the origin. The POS parameter sets the offset value of the origin coordinates.

Note: This command uses the internal homing mode of the driver. It is necessary to set 60980008 homing mode and homing speed and other related parameters in the driver first (it can also be written through the MC_WPARAS command). For details, please refer to the driver manual.

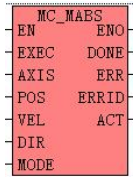
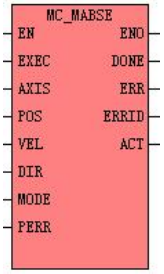
- LD Format instruction description

If EN is 1, then the rising edge of EXEC will trigger the execution of this instruction.

When the instruction is executed, the PLC first sends the command to let the axis start to find the origin; after the sending is completed, check the status returned by the driver. The check will last for TIME (the timeout period set by the user, in ms). If the axis successfully finds the origin within this time, it means that the command is executed successfully. At this time, DONE is set to 1, ERR is set to 0, and ERRID is set to 0. If an error occurs (It may be an error in the execution of the command itself, or it may be an error that the driver did not perform the action correctly during the execution process, see the error code for details), the execution of the instruction will fail, and the execution of the instruction will stop. At the same time, DONE will be set to 1,

ERR will be set to 1, and ERRID will be assigned the corresponding error code.

10.5.8.3.2.5 MC_MABS (absolute motion)

	Name	Instructions format	Adopt to
LD	MC_MABS		☑ KS ☑ KW ☑ K209M ☑ K6
	MC_MABSE		

Parameters	Input/Output	Data type	Memory area allowed	Description
EXEC	Input	BOOL	M、V、L、SM	Rising edge triggers this instruction to execute once; falling edge triggers pause motion
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
POS	Input	REAL	V、M、L、constant	Absolute target position, unit: mm or °
VEL	Input	REAL	V、M、L、constant	The maximum speed increased during the movement (>0), unit: mm/min or °/min.
DIR	Input	INT	V、M、L、constant	direction of motion. Reserved, the function has not been implemented yet, just keep it as 0.
MODE	Input	INT	V、M、L、constant	Movement mode: single execution or permanent execution. 0 means a single execution, and the command will exit after the axis completes this absolute positioning. 1 means permanent execution. After the axis completes an absolute positioning, the command will not exit. If a new target position is found, a command will be sent to let the axis continue to perform a new absolute positioning.
PERR	Input	REAL	V、M、L、constant	The judgment difference between the target position and the actual position. When the absolute value of the target

				position minus the actual position is less than or equal to RERR, DONE jumps from 0 to 1.
DONE	Output	BOOL	M、V、L	Completion flag. DONE transitions from 0 to 1 when the instruction is executed.
ERR	Output	BOOL	M、V、L	Error flag. Set to 1 if an error occurs during instruction execution.
ERRID	Output	BYTE	V、M、L	error code
ACT	Output	BOOL	M、V、L	MODE=0, when single execution, ACT indicates whether the single positioning command is correctly activated. 1 means active, 0 means inactive. MODE=1, when executing permanently, ACT indicates whether the permanent positioning command is correctly activated. 1 means active (it will always be 1 when a single positioning is completed), 0 means inactive.

This command is used to control the target axis to move to the target position (absolute position). When moving, the speed starts from the current value of VEL, and the speed is zero when reaching the target position. This command allows to pause during motion.

The difference between the MC_MABS and MC_MABSE instructions: the condition for the MC_MABS instruction to give the DONE signal is to judge the "position arrives" status word sent by the servo. The DONE condition given by the MC_MABSE instruction is to judge the status word "position arrived" from the servo, and at the same time to judge whether the absolute value of the difference between the target position and the actual position is less than or equal to RERR. **It is recommended that the user use the MC_MABSE command, the DONE signal given by this command is more reliable.**

Note: This command uses the internal acceleration and deceleration of the drive. It is necessary to first set the relevant parameters such as 60830020 acceleration and deceleration in the drive (it can also be written through the MC_WPARAS command). For details, please refer to the drive manual.

- LD Format instruction description

If EN is 1, then the rising edge of EXEC will trigger the execution of this instruction.

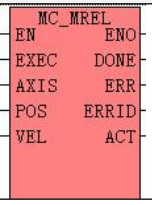
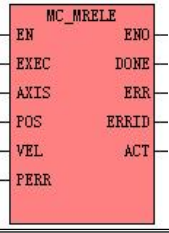
When the instruction is executed, the PLC controls the axis to start absolute positioning according to the target position (POS) and motion speed (VEL) parameter values input by the user. During the motion, the instruction will keep scanning the target position and target speed parameter values, if there is any change, it

will be sent to the axis immediately. That is to say, new speed parameters and position parameter values can be accepted at any time (for example, to perform a pause, set the speed to 0 during the movement to pause, and give the speed value again to resume the movement). At the same time, the PLC will keep checking the return status of the axis. If the target position of this positioning is successfully reached, it means that this positioning is completed, then DONE is set to 1, ERR is set to 0, and ERRID is set to 0. After this positioning is completed, the instruction will judge the MODE value. If the MODE value is set to single-run mode, the command exits directly. If the MODE value is set to the permanent operation mode, then the instruction does not exit, and scans the target position value at any time. If the target position changes, it will be sent to the axis, allowing the axis to perform a new absolute positioning.

If an error occurs, the execution of the instruction will fail, and the execution of the instruction will stop. At the same time, DONE will be set to 1, ERR will be set to 1, and ERRID will be assigned the corresponding error code.

If EN is 0, the instruction will not be executed. If EN becomes 0 during the execution, the execution of the instruction will stop, and the axis will be in the state of static lock and axis waiting.

10.5.8.3.2.6 MC_MREL (relative motion)

	Name	Instructions format	Adopt to
LD	MC_MREL		☑ KS ☑ KW ☑ K209M ☑ K6
	MC_MRELE		

Paramet	Input/Outpu	Data type	Memory area	Description

ers	t		allowed	
EXEC	Input	BOOL	M、V、L、SM	Rising edge triggers this instruction to execute once; falling edge triggers pause motion
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
POS	Input	REAL	V、M、L、constant	Relative distance to move, unit: mm or °. A positive number indicates movement in a positive direction; a negative number indicates movement in a negative direction.
VEL	Input	REAL	V、M、L、constant	The maximum speed increased during the movement (>0), unit: mm/min or °/min.
PERR	Input	REAL	V、M、L、constant	The judgment difference between the target position and the actual position. When the absolute value of the target position minus the actual position is less than or equal to RERR, DONE jumps from 0 to 1.
DONE	Output	BOOL	M、V、L	Completion flag. DONE jumps from 0 to 1 when the instruction is executed.
ERR	Output	BOOL	M、V、L	Error flag. Set to 1 if an error occurs during instruction execution.
ERRID	Output	BYTE	V、M、L	error code
ACT	Output	BOOL	M、V、L	Whether the command is activated correctly. 1 means active, 0 means inactive.

This instruction controls the target axis to move the specified distance POS (with the current position as a reference, that is, the current position as the starting position). When moving, the speed starts from the current value of VEL, and the speed is zero when reaching the target position. This instruction allows pause.

The difference between the MC_MREL and MC_MRELE instructions: the condition for the MC__ instruction to give the DONE signal is to judge the "position arrives" status word sent by the servo. The condition of DONE given by the MC_MRELE instruction is to judge the status word "position arrived" sent by the servo, and at the same time to judge whether the absolute value of the difference between the target position and the actual position is less than or equal to RERR. **It is recommended that the user use the MC_MRELE instruction, the DONE signal given by this command is more reliable.**

Note: This instruction uses the internal acceleration and deceleration of the drive. It is necessary to first set the relevant parameters such as 60830020 acceleration and deceleration in the drive (it can also be written through the MC_WPARAS command). For details, please refer to the drive manual.

- LD Format instruction description

If EN is 1, then the rising edge of EXEC will trigger the execution of this instruction.

When the instruction is executed, the PLC controls the axis to start relative positioning (using the current position as a reference) according to the target position (POS) and motion speed (VEL) parameter values input by the user. During the movement, the instruction will keep scanning the target speed parameter value, and if there is any change, it will be sent to the axis immediately, that is, the new speed parameter value can be accepted at any time (For example, to perform a pause, set the speed to 0 during the movement, and then re-give the speed value to resume the movement). At the same time, the PLC will keep checking the return status of the axis. If the target position of this positioning is successfully reached, it means that this positioning is completed, then DONE is set to 1, ERR is set to 0, and ERRID is set to 0. If an error occurs (It may be an error in the execution of the command itself, or it may be an error that the driver did not perform the action correctly during the execution process, see the error code for details), the execution of the instruction will fail, and the execution of the instruction will stop. At the same time, DONE will be set to 1, ERR will be set to 1, and ERRID will be assigned the corresponding error code.。

If EN is 0, the instruction will not be executed. If EN becomes 0 during the execution, the execution of the instruction will stop, and the axis will be in the state of static lock and axis waiting.

10.5.8.3.2.7 MC_JOG (JOG)

	Name	Instructions format	Adopt to
LD	MC_JOG		☑ KS ☑ KW ☑ K209M ☑ K6

Parameters	Input/Output	Data type	Memory area allowed	Description
EXEC	Input	BOOL	M、V、L、SM	Rising edge triggers this instruction to execute once; falling edge triggers pause motion

AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
VEL	Input	REAL	V、M、L、constant	Movement speed, unit: mm/min or °/min. A positive number indicates a positive direction, and a negative number indicates a negative direction.
DIR	Input	INT	V、M、L、constant	direction of motion. Reserved, the function has not been implemented yet, just keep it as 0.
DONE	Output	BOOL	M、V、L	Completion flag. DONE transitions from 0 to 1 when the instruction is executed.
ERR	Output	BOOL	M、V、L	Error flag. Set to 1 if an error occurs during instruction execution.
ERRID	Output	BYTE	V、M、L	error code
ACT	Output	BOOL	M、V、L	Whether the command is activated correctly. 1 means active, 0 means inactive.

This instruction controls the target axis to run at the target speed specified by Vel.

Note: This command uses the internal acceleration and deceleration of the drive. It is necessary to first set the relevant parameters such as 60830020 acceleration and deceleration in the drive (it can also be written through the MC_WPARAS command). For details, please refer to the drive manual.

- LD Format instruction description

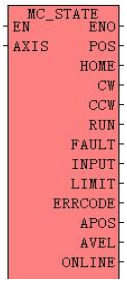
If EN is 1, then the rising edge of EXEC will trigger the execution of this instruction.

When the instruction is executed, the PLC controls the axis to start the jog operation according to the motion speed (VEL) parameter value input by the user. During the movement of the axis, the instruction will keep scanning the target speed parameter value, and if there is any change, it will be sent to the axis immediately, that is, the new speed parameter value can be accepted at any time.

If an error occurs (it may be an error in the execution of the command itself, or it may be an error in which the driver did not execute the action correctly during the execution process, see the error code for details), the execution of the command will fail. The execution of the instruction will stop, and at the same time, DONE will be set to 1, ERR will be set to 1, and ERRID will be assigned the corresponding error code.

If EN is 0, the instruction will not be executed. If EXEC becomes 0 during execution, the execution of the instruction will stop, and the axis will be in the state of static locking and axis waiting.

10.5.8.3.2.8 MC_STATE (Read the status values of the drive)

	Name	Instructions format	Adopt to
LD	MC_STATE		☑ KS ☑ KW ☑ K209M ☑ K6

Parameters	Input/Output	Data type	Memory area allowed	Description
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave station)
POS	Output	BOOL	V、M、L	"Position arrived" signal
HOME	Output	BOOL	V、M、L	"Origin found" signal
CW	Output	BOOL	M、V、L	"Motor CW" signal
CCW	Output	BOOL	M、V、L	"Motor reverse" signal
RUN	Output	BOOL	M、V、L	"Motor running" sign
FAULT	Output	BOOL	M、V、L	"Axis alarm" sign
INPUT	Output	WORD	V、M、L	The status of the digital input port of the axis, BIT0 corresponds to the DIN1 of the axis, which are stored in sequence, and the specific number of digital input ports can be found in the driver manual
LIMIT	Output	BOOL	M、V、L	"Limit to" sign
ERRCODE	Output	WORD	V、M、L	Axis alarm error code
APOS	Output	REAL	M、V、L	The current actual position of the machine, mm or °.
AVEL	Output	REAL	M、V、L	The current actual speed of the machine, mm/min or °/min.
ONLINE	Output	BYTE	M、V、L	"Axis On Line" flag. 1 means the axis is not online, 0 means the axis is online.

This command scans the status of the drive all the time, obtains various status flags and outputs them to the corresponding output parameters. **But there may be a certain delay, users need to be cautious when using each output as the basis for judging actions**

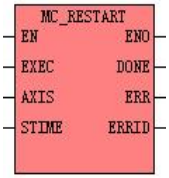
Note: The two signals "Position Arrived" and "Origin Found" will become 0 again during the

execution of the action (positioning or finding the origin), and will not be reset to 1 until the action is executed correctly!

- LD Format instruction description

If EN is 1, this command will be executed. If EN is 0, the instruction will not be executed, and various output parameters will not be refreshed.

10.5.8.3.2.9 MC_RESTRAT (Reconfigure and start the slave station)

	Name	Instructions format	Adopt to
LD	MC_RESTRAT		☑ KS ☑ KW ☑ K209M ☑ K6

Parameters	Input/Output	Data type	Memory area allowed	Description
EXEC	Input	BOOL	I、Q、V、M、L、SM	The rising edge of EXEC will start this command and execute a process of restarting the slave station. The falling edge of EXEC will stop the execution of this command and terminate the process of restarting the slave station.
AXIS	Input	INT	V、M、L、constant	The axis number of the target axis (that is, the address of the CANOpen slave).
STIME	Input	INT	V、M、L、constant	The maximum time for the master station to wait for the response from the slave station after sending the SDO request message.
DONE	Output	BOOL	Q、M、V、L	Instruction execution result flag bit. If the command is being executed, the output of DONE is 0. DONE immediately becomes 1 if the instruction execution is complete (whether it succeeds or fails).
ERR	Output	BOOL	Q、M、V、L	Instruction execution error flag. Set to 1 if an error occurs during command execution.
ERRID	Output	BYTE	V、M、L	The error code of instruction execution. If an error occurs during instruction execution, ERRID is the specific error

				message.

Note: AXIS and STIME parameters must be constant or variable at the same time. In a user project, a maximum of 32 MC_Restart instructions are allowed.

For the description of the error code (ERRID), see the table below:

Error code	Description
0	no error
1	Wrong target axis number
2	The target axis is executing the process of restarting the slave station, and it is forbidden to execute it again until it is completed.
3	The CAN message sent by the master station is wrong (maybe due to the line, hardware interface, software buffer being full, etc.).
4	The SDO request message sent by the master station, the target axis timed out and did not respond.
5	The SDO request message sent by the master station, the target axis responds with an error.
6	The user actively stops the execution of this instruction (set EXEC to 0).

- LDFormat instruction description

If EN is 1, this command will be executed. If EN is 0, the instruction will not be executed, and various output parameters will not be refreshed.

During the normal operation of the master station, the user can call this command to reconfigure and start the target axis (the station number is AXIS). The various parameters of the axis will adopt the parameters that have been configured for the slave station in [Kinco motion control network configuration], including error monitoring parameters and so on.

If EN is 1, then the rising edge of EXEC will trigger the execution of this command. First, the DONE output is 0, and then the master station will read the configured axis parameters, and perform the following operations on the target axis in sequence::

1) Send a command to put the target axis into the pre-operational state.

2) Send SDO to configure the node guarding parameters of the target axis.

3) Send SDO to configure the mapping parameters of all PDOs of the target axis.

4) Send SDO to configure the communication parameters of all PDOs of the target axis.

5) Send the "start node" command to the target axis.

When the above process is successfully executed, this command will exit, and immediately output DONE as 1, and both ERR and ERRID as 0.

If any error occurs during the execution of the instruction, or EXEC becomes 0, the instruction will also exit, and DONE and ERR will be output as 1 immediately, and ERRID will output the corresponding error code.

10.6 Kinco Internetwork protocol communication

Kinco PLC Interconnect communication protocol is a communication protocol designed for efficient networking communication between Kinco PLC, users can use RS485 or LoRa communication interface to use Kinco interconnect communication function.

The following table describes each series of communication interfaces that support Kinco interconnect communication.

Series	Supports the communication interface of Kinco interconnection communication
KW1	LoRa, can be used as a master or slave station
KW2	
K6	RS485 (PORT1), can be used as master or slave station
K6S	
MK	RS485 (PORT1), can be used as master or slave station

Kinco interconnection protocol adopts the master-slave mode. In the network, only one master station must exist, and other nodes must serve as slave stations. The master station accesses the slave station according to the conditions set by the user, and the slave station can only respond after receiving the request from the master station. In addition, the master station can send write data broadcast packets. All slave stations process broadcast packets after receiving them, but do not need to respond to broadcast packets.

If no other communication protocol functions are started, the PLC will default as the Kinco interconnection protocol slave station after powering on, and the user does not need to program. For the master station, the Kinco interconnection protocol function has the highest priority, and once the PLC is operated as

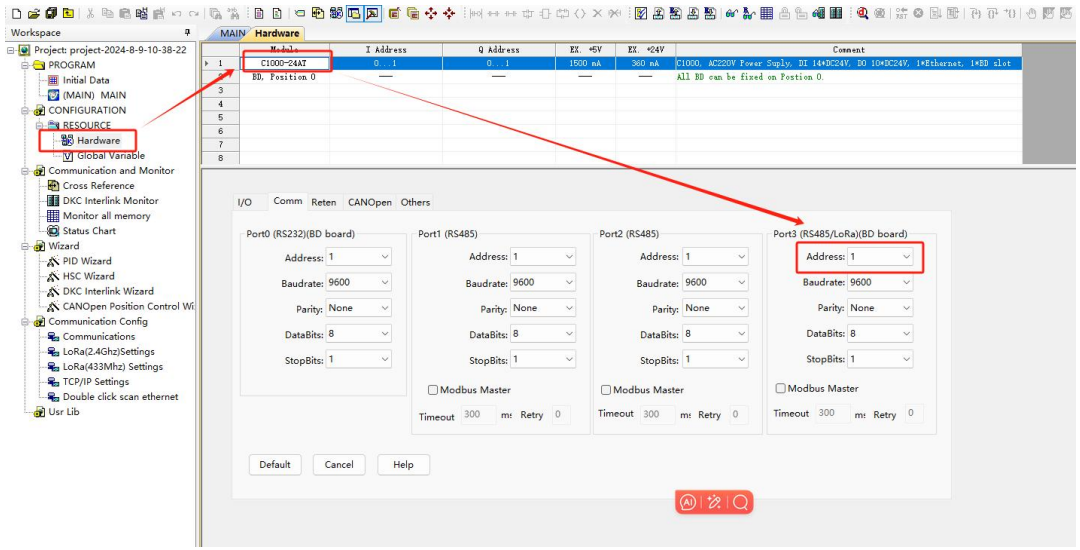
the interconnection protocol master station, the communication interface will no longer process the messages of other communication protocols.

10.6.1 Method of application

Users can enable the Kinco PLC Interconnect Protocol function by following the steps below.

1) Select the communication port used, and configure the communication parameters of the master station and each slave station to ensure normal communication.

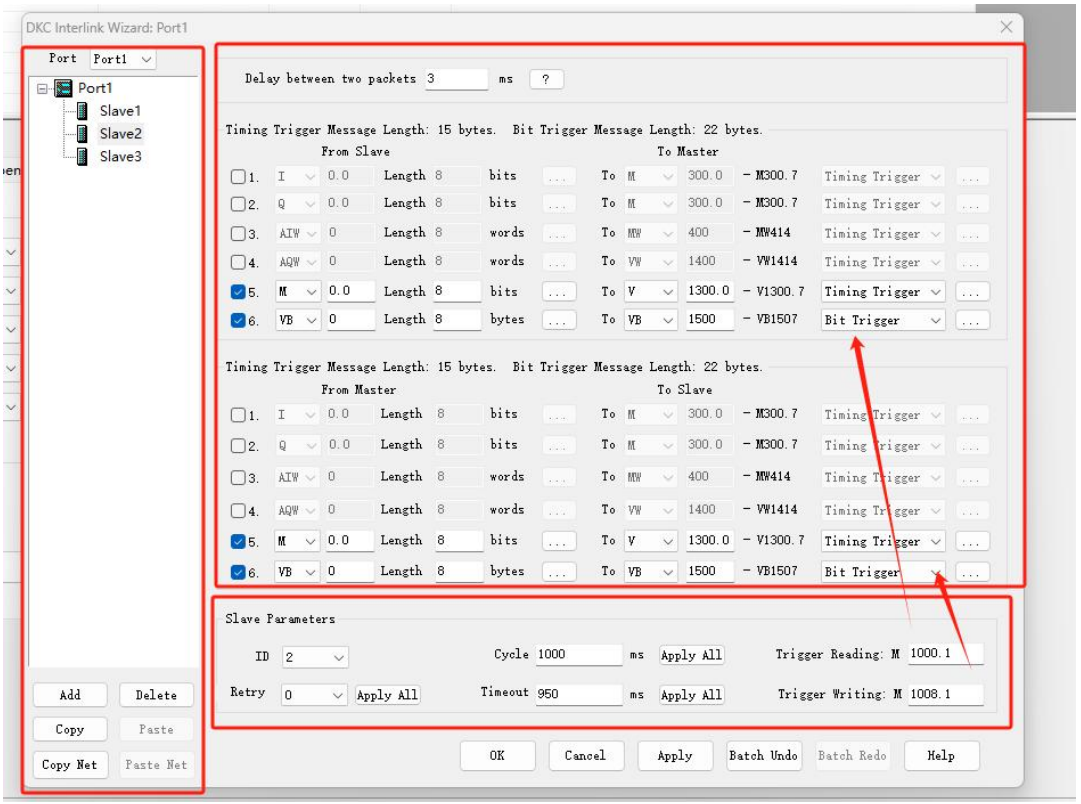
2) Each slave station needs to specify a station number for the communication port in "Hardware Configuration" -> "Communication Settings". The valid station number ranges from 1 to 63. In the same network, the station number of each slave station must be unique. Duplicate station numbers are not allowed.



3)In the user project of the master station, use the Kinco PLC interconnection wizard to configure each slave station, and download the project to a PLC after the configuration is completed, then the PLC will operate as the master station, start and manage the communication of the entire network. There is no need for programming and configuration in each slave user project.

10.6.2 Kinco PLC Interconnection Wizard

In the KincoBuilder software, the user can double-click the Kinco PLC Interconnection Wizard node in the Project Manager to enter the wizard window, where the communication data and communication parameters of all nodes in the network can be set.



- List of all slave stations

Select the communication port to be used, edit the slave stations to be connected in this network, and display them in a tree list.

The user must add all the slave stations in the network, and the master will only access the slave stations that exist in the list.

Click a slave node, the right side of the window will display the communication data and

communication parameters of the local slave. Each slave station must be configured with at least one data packet. The slave station without any data packet is ignored. When the slave address is changed, the title of the tree node will automatically change based on the new address.

If there is a problem with the configuration of the slave station, a small red "? "is displayed before the tree node title. The usual reason for the warning is that the data read from multiple slave stations overlaps in the address area written to the master station.

- Communication Settings

The master station accesses the slave station periodically according to the user-set period, and the access period of each slave station can be set separately.

Each data transmission message of each slave station can be set to "timing trigger" and "edge trigger" respectively. When the timing time of a slave station is up, the master station will decide whether to transfer data between the slave station and the slave station according to the trigger form configured by the user: for the data triggered by the timing, the master station will immediately start a transmission; For the data triggered by the edge, the master station will first determine the trigger bit. If the trigger bit value is "0", the master station will not start the transmission; if the trigger bit value is "1", the master station will immediately start a transmission, and the master station will automatically reset the trigger bit to "0" after completion.

Slave Address: indicates the address of the current slave.

[Access period] : indicates the timing period for the primary station to access the secondary station, in ms. The maximum allowed timing time for KW is approximately 49 days.

Users can set the access period based on actual requirements. If you need to quickly refresh the data from the slave station, then the access cycle can be set as short as possible, but it should be noted that if the access cycle is shorter than the time required for data transmission, then the actual access cycle is the time required for data transmission at the station. If the data does not need to be refreshed frequently, the user can set the access period as long as possible to reduce the communication frequency of the local slave station and reduce the network pressure.

[Timeout period] : After the master station sends a request, the master station must receive a response packet from the slave station within this period. Otherwise, the master station records a communication

timeout error of the local slave station.

[Retries] : After the primary station sends a request, if a communication error occurs (no response is timed out, or an error is received), the primary station will retry the request until the communication succeeds or the number of retries set by the user is reached.

[Rising edge of triggered read] : Specifies a variable in the M zone of the master station, which is used to trigger the master station to read the data configured as "edge triggered" in the local slave station. After reading is completed, the master station will automatically clear the bit "0", so in the master station user program, the moment before the need to transmit data can be set to "1", avoid keeping it "1" all the time.

[Rising edge of triggered write] : Specifies a variable in the M zone of the master station, which is used to trigger the master station to write data configured as "edge triggered" mode in the local slave station. The master station will automatically clear the bit "0" after the completion of writing, so in the master station user program, the moment before the need to transmit data can be set to "1", avoid keeping it "1" all the time.

- Transfer data Settings

Here, you can configure the data to be transmitted by the local slave station. A maximum of six groups of data can be set respectively in the data area for the master station to read and write. The triggering mode of each group of data can be set respectively (timing triggering or edge triggering).

To improve the communication efficiency, the master station combines the data areas configured by the user. All the data triggered by timing is combined into one packet, and all the data triggered by edge is combined into another packet. Note: The maximum length of a communication message is 246 bytes! When you configure communication data, the software automatically displays the combined packet length. Keep the combined packet length within 246 bytes. A packet contains the user's valid data and the protocol data added by the software, so the maximum length of the user's valid data in the packet is about 226 bytes.

- broadcast message

The station number 64 is fixed to send broadcast packets, which are used to modify the data of all the slave stations in the network at the same time.

Slave station 64 does not actually exist in the network. The master station sends the data configured by the user on slave station 64 as broadcast packets. After receiving the packets, all slave stations on the network

process the data without responding.

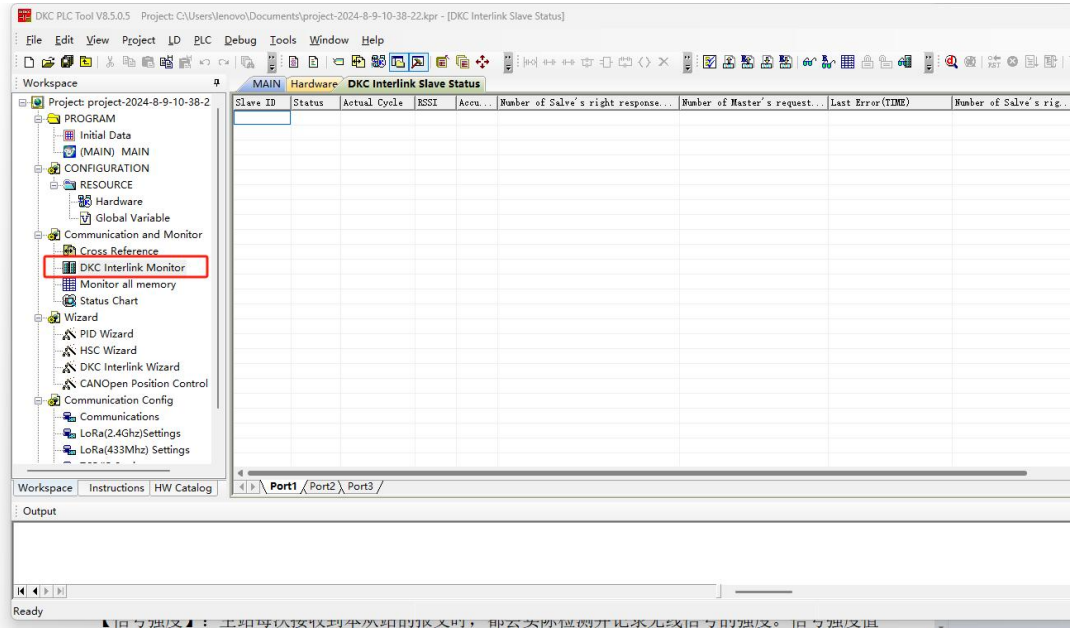
- Delay between packets from the primary station

If the RS485 communication port is used and the baud rate is greater than 9600bps, an appropriate delay must be added between two consecutive request packets from the master station. Otherwise, the slave station may fail to receive the request packets. The delay time is longer than the scanning period of the slave station.

10.6.3 Kinco PLC connected slave station status

"Kinco PLC Interconnect Slave Station Status" is used to monitor the operation information of all slave stations in the current interconnect communication network.

In the project manager of KincoBuilder software, users can double-click the [Kinco PLC Interconnect slave Station status] node to open it.



[Slave station address] : indicates the number of the slave station to which the bank information belongs.

[Running Status] : Indicates the communication status of the slave station. The value can be normal, offline, or not configured successfully.

When the network is started, the master station sends packets to the slave stations to configure the communication data area. After the configuration is successful, the master station and the slave station continue to interact with each other. If the configuration fails, the master station records that the slave station is not configured successfully and keeps trying to reconfigure the slave station until the configuration is successful. During normal data exchange, if the local slave station responds incorrectly, the master station will record an "offline" error once and record it as "normal" after the next normal communication.

[Actual access period] : indicates the actual cycle time of polling visits by the primary station to the secondary station, in ms.

[Signal strength] : Every time the master station receives a message from the local slave station, it will actually detect and record the strength of the wireless signal. The signal strength value is negative, and the closer the value is to 0, the higher the signal strength. In practical applications, if the user finds that the communication error rate of a slave station is relatively high, and the signal strength is relatively low, it is necessary to consider improving the installation of the slave station, such as adjusting the antenna position, replacing the antenna with a higher gain, and so on.

[Communication accuracy rate] : The proportion of correct response packets received by the master station from the slave station after the master station is powered on.

If LoRa communication interface is used, because wireless communication cannot guarantee 100% accuracy, we believe that the accuracy rate of more than 85% is good communication.

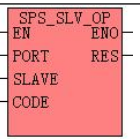
10.6.4 Kinco PLC link protocol special instruction

This set of instructions applies only to the "Kinco PLC Interconnection Protocol".

In the user engineering [PLC Hardware configuration] CPU [communication Settings] page, the user can see the specific number of each communication port of the machine.

10.6.4.1 SPS_SLV_OP (The master stops or restarts the communication with the slave)

	Name	Instructions format	Adopt to

LD	SPS_SLV_OP		☑ KW2 ☑ K6 ☑ K6S

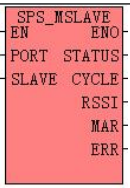
Parameters	Input/Output	Data type	Memory area allowed
PORT	Input	INT	Constant
SLAVE	Input	INT	V、M、L、Constant
CODE	Input	BYTE	V、M、L、Constant
RES	Output	BYTE	V、M、L

Parameters	Description
PORT	The number of the communication port used. 0 indicates PORT0, 1 indicates PORT1, 2 indicates PORT2, and so on. If the parameter value specifies a non-existent communication interface, it is an invalid value, resulting in an instruction error.
SLAVE	Target slave station number. The slave must already exist in the network, that is, it must have been configured in the Kinco PLC Interconnection Wizard.
CODE	Operation options, whose values have the following meanings: 1 ~ The master station initiates communication with the target slave station. 2 ~ The master stops communicating with the target slave.
RES	Execution result, the meaning of its value is as follows: 1 ~ Execution succeeds. 2 ~ The used communication port does not exist. 3 ~ The target slave does not exist in the network. 4 ~ Illegal operation options.

When the EN value is 1, the instruction is executed. For the Kinco interconnection network running on the PORT communication interface, the user can call this command in the master station to suspend or restart the communication with the target SLAVE station.

10.6.4.1 SPS_MSLAVE (Read the communication of the slave station)

	Name	Instructions format	Adopt to

LD	SPS_MSLAVE		☑ KW2 ☑ K6 ☑ K6S

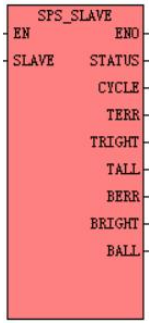
Parameters	Input/Output	Data type	Memory area allowed
PORT	Input	INT	Constant
SLAVE	Input	INT	V、M、L、Constant
STATUS	Output	BYTE	V、M、L
CYCLE	Output	DWORD	V、M、L
RSSI	Output	INT	V、M、L
MAR	Output	WORD	V、M、L
ERR	Output	BYTE	V、M、L

Parameters	Description
PORT	The number of the communication port used. 0 indicates PORT0, 1 indicates PORT1, 2 indicates PORT2, and so on. If the parameter value specifies a non-existent communication interface, it is an invalid value, resulting in an instruction error.
SLAVE	Target slave station number. The slave must already exist in the network, that is, it must have been configured in the Kinco PLC Interconnection Wizard.
STATUS	The running status of the slave station, its value has the following meaning: 1 ~ Normal operation. 2 ~ The primary station fails to configure the secondary station. 3 ~ Offline (that is, no response packet is received for the latest request). 4 ~ The master stops the communication with the slave by invoking the SPS_SLV_OP command.
CYCLE	Real polling period of the master station to the slave station, in ms.
RSSI	The last signal strength received from the master station to the local slave station. Note: Valid for LoRa communication only.
MAR	The correct rate of the local slave station's response packet. The output value is a 16-bit integer, meaning: Correct rate x 100. Correct rate = Total number of correct response messages from the slave station ÷ total number of request messages from the master station to the slave station
ERR	The last communication error occurred at the local slave station. 0 to no error. 1 ~ Local slave station timeout no response.

	2 ~ The length of the Write Data packet is incorrect. 3 ~ The length of the Read Data packet is incorrect. 4 ~ Incorrect configuration packets. 5 ~ Incorrect communication area configuration.

When the EN value is 1, the instruction is executed. For the Kinco interconnection network running on the PORT communication interface, the user can call this command in the master station to read the communication situation of the SLAVE station.

10.6.4.2 SPS_SLAVE (Read the communication of the slave station)

	Name	Instructions format	Adopt to
LD	SPS_SLAVE		☑ KW2 ☑ K6 ☑ K6S

Parameters	Input/Output	Data type	Memory area allowed
SLAVE	Input	INT	V、M、L、Constant
STATUS	Output	BYTE	V、M、L
CYCLE	Output	DWORD	V、M、L
TERR	Output	BYTE	V、M、L
TRIGHT	Output	DWORD	V、M、L
TALL	Output	DWORD	V、M、L
BERR	Output	BYTE	V、M、L
BRIGHT	Output	DWORD	V、M、L
BALL	Output	DWORD	V、M、L

Parameters	Description

SLAVE	Target slave station number. The slave must already exist in the network, that is, it must have been configured in the Kinco PLC Interconnection Wizard.
STATUS	The running status of the slave station, its value has the following meaning: 1 ~ Normal operation. 2 ~ The primary station fails to configure the secondary station. 3 ~ Offline (that is, no response packet is received for the latest request). 4 ~ The master stops the communication with the slave by invoking the SPS_SLV_OP command.
CYCLE	Real polling period of the master station to the slave station, in ms.
TERR	The latest error occurred in the communication of the slave station in timed trigger mode. 0 to no error. 1 ~ Local slave station timeout no response. 2 ~ The length of the Write Data packet is incorrect. 3 ~ The length of the Read Data packet is incorrect. 4 ~ Incorrect configuration packets. 5 ~ Incorrect communication area configuration.
TRIGHT	This parameter specifies the total number of correct packets that are transmitted by the local slave station in timed triggering mode.
TALL	Total number of packets transmitted by the local slave station in timed triggering mode.
BERR	The last error occurred when the local slave station used edge-triggered communication. 0 to no error. 1 ~ Local slave station timeout no response. 2 ~ The length of the Write Data packet is incorrect. 3 ~ The length of the Read Data packet is incorrect. 4 ~ Incorrect configuration packets. 5 ~ Incorrect communication area configuration.
BRIGHT	The cumulative number of correct packets in the communication of the local slave station in edge-triggered mode.
BALL	Indicates the total number of packets transmitted by the local slave station in edge-triggered mode.

When the EN value is 1, the instruction is executed.

For the Kinco interconnection network running on the default communication port (LoRa port for the KW series, PORT1 port for the K6 and K6S series), the user can call this command in the master station to read the communication of the SLAVE station.

Chapter 11 Use of Analog and PID

This chapter mainly introduces the application of Kinco-K series PLC analog quantity and PID instruction, including instruction call, parameter setting, use attention points, program samples, etc.

11.1 Function overview

The body of some KPLC CPU models provides AI (analog input) and AO (analog output) channels, supporting 0-20mA, 0-10V and other signal forms. At the same time, KPLC provides expansion modules that support various analog input signals (including current, voltage, RTD, TC, etc.) and analog output signals (including current and voltage).

The use of PID is usually accompanied by the application of analog quantities. Simply put, PID is to form a control deviation based on a given value and an actual output value, and to control the controlled object by making the deviation proportional, integral, and differential through a linear combination.

The PID instruction is a digital PID controller. In Kinco-K series PLC, PID instruction can be called to realize PID control function. In one CPU, users can use up to 8 PIDs at the same time. In addition, PID control can be separated from the connection with the expansion module, which expands the flexibility of this function.

Note that the PID operation takes a long time, it is recommended to avoid calling the PID function in the interrupt program.

11.2 Use of analog

11.2.1 Analog Introduction

The analog quantity is a quantity that changes continuously within a certain range, that is, it can take any value within a certain range.

The digital quantity is a discrete quantity and can only take several discrete values. For example, a binary digital variable can only take two values of 0 and 1.

What we usually call DI and DO refer to digital quantities, while AI and AO refer to analog quantities.

Currently KPLC supports the following types of analog input signals: current signal (0-20mA/4-20mA), voltage signal (0-10V/1-5V/-10V-+10V), thermal resistance signal (Pt100/Cu50/Pt1000), thermocouple signal (J type/K type/E type/S type). Support the following types of analog output signals: current signal (0-20mA/4-20mA), voltage signal (0-10V/1-5V/-10V-+10V). Users can refer to the hardware manuals of each series to select the required modules.

11.2.2 Signal form, measurement range and expression form of analog quantity

11.2.2.1 Signal form, measuring range and representation of the analog input

After the input signal of the analog channel is sampled by the ADC and linearly calculated, the calculation result is sent as a measured value to the AI image area of the CPU module through the expansion bus for user program access.

Each signal form has a certain measurement range. If the measured value exceeds the measurement range, the module will alarm and send a fault message to the CPU module through the expansion bus. **It is recommended that the user short-circuit the terminals of unused channels and set the signal form to [0-10V], [0-20mA] in the programming software (if the signal form of the RD module is set to resistance). This way these unused channels will not cause an alarm.**

The following table is the measurement range and measurement value representation format of the input analog quantity, where I represents the input current value, and V represents the input voltage value. For example, if the signal form is current, the read analog value is 9965, and the actual corresponding current value is 9.965mA.

signal form	Measuring range	Measurement value representation format
4-20mA	3.92-20.4mA	I×1000
0-20mA	0-20.4mA	

1-5V	0.96-5.1V	V×1000
0-10V	0-10.2V	

The following table shows the measurement range and measurement value representation format of the RD module. The RD module can be connected to a thermal resistor (Pt100, Pt1000, Cu50) to measure the temperature, and can also directly measure the resistance of the resistor. Each channel can be mixed with different resistor types, and supports two-wire and three-wire wiring. For specific wiring, please refer to the hardware manual. In the table below, T represents the measured temperature value, and R represents the measured resistance value.

For example, if the signal type is Pt100, the read analog value is 365, and the actual corresponding temperature value is 36.5°C.

signal form	Measuring range	Measurement value representation format
Pt100	-200~850℃	T*10
Cu50	-50~150℃	
Pt1000	-50~300℃	R*10
resistance	0-2000Ω	

The following table shows the measurement range of the TC module and the format of the measurement value. The TC module can be connected to different types of thermocouples and supports two wiring types: two-wire system and three-wire system. For specific wiring, please refer to the hardware manual. T in the table below represents the measured temperature value.

signal form	Measuring range	Measurement value representation format
J	-210~1200℃	T*10
K	-270~1300℃	
E	-270~1000℃	
S	-50~1600℃	

11.2.2.2 Signal form, measuring range and representation of the analog output

The AQ output value specified in the user program is first sent to the corresponding AO module through the expansion bus, and then it is calculated, transformed and output on the specified channel through the

DAC.

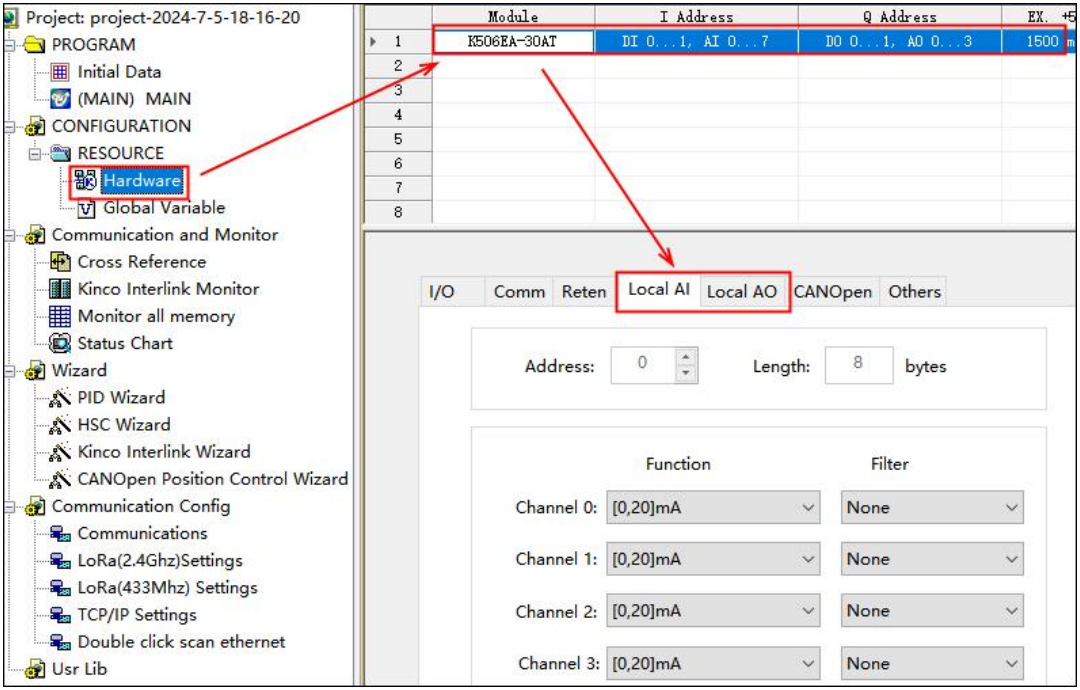
The output range of various signal forms is limited. If the output value specified in the user program exceeds the upper and lower limits of the selected range, the output signal will keep the limit value unchanged.

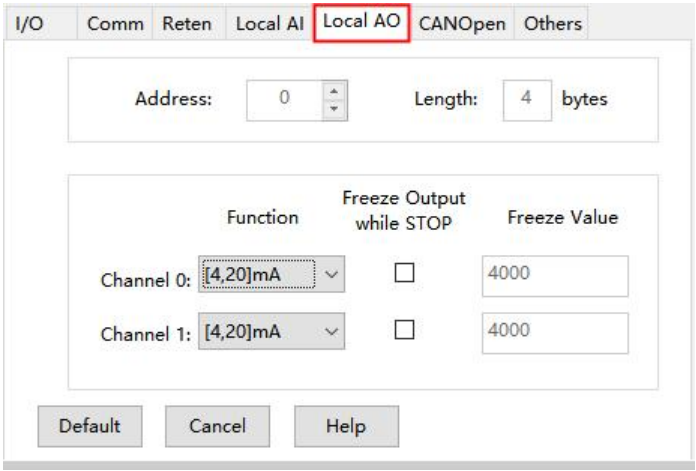
The following table is the format of the output range and output value, where I represents the actual current value, and V represents the actual voltage value.

Signal form	Output range	Output value specified in the user program
4-20mA	3.92-20.4mA	I×1000
0-20mA	0-20.4mA	
1-5V	0.96-5.1V	V×1000
0-10V	0-10.2V	
-10-+10V	-10.2V-+10.2V	

11.2.3 Setting of analog quantity in hardware configuration

We can select the form and filtering method of the analog signal of the CPU body or the analog expansion module in the PLC hardware setting of the KincoBuilder software. The following figure is an example of configuration that supports current and voltage signal forms:


Project: project-2024-7-5-18-16-20
PROGRAM
Initial Data
(MAIN) MAIN
CONFIGURATION
RESOURCE
Hardware
Global Variable
Communication and Monitor
Cross Reference
Kinco Interlink Monitor
Monitor all memory
Status Chart
Wizard
PID Wizard
HSC Wizard
Kinco Interlink Wizard
CANOpen Position Control Wizard
Communication Config
Communications
LoRa(2.4Ghz)Settings
LoRa(433Mhz) Settings
TCP/IP Settings
Double click scan ethernet
Usr Lib
Module
I Address
Q Address
1
K506EA-30AT
DI 0...1, AI 0...7
DO 0...1, AO 0...3
1500
I/O
Comm
Reten
Local AI
Local AO
CANOpen
Others
Address:
0
Length:
8
bytes
Function
Filter
Channel 0:
[0,20]mA
None
Channel 1:
[0,20]mA
None
Channel 2:
[0,20]mA
None
Channel 3:
[0,20]mA
None


I/O
Comm
Reten
Local AI
Local AO
CANOpen
Others
Address:
0
Length:
4
bytes
Function
Freeze Output while STOP
Freeze Value
Channel 0:
[4,20]mA
4000
Channel 1:
[4,20]mA
4000
Default
Cancel
Help

- Address

The memory area corresponding to the analog input is the %AIW area, and the %AQW area corresponding to the output. When using the analog memory area, it is related to the starting address and length in the hardware configuration.

• **initial address**: Specify the starting byte address of the address space occupied by the module in the AI area (that is, the address of the first channel). Each AI point occupies 2 bytes in the AI area. Therefore, the address must be even.

• **Length**: The length of the address space that this module occupies. This is a fixed value that depends on the number of AI channels on the module.

As above, the starting address of the analog input is specified as %AIW0, and this CPU has 4 AI channels. Therefore, the addresses of its four channels are %AIW0, %AIW2, %AIW4, and %AIW6. The starting address of the analog output is specified as %AQW0, and the CPU has 2 AQ channels, so the addresses of its 2 channels are %AQW0 and %AQW2 in turn.

➢ Channel settings

- **Signal Type**: Select the input signal type for each channel. The sampled value will be automatically converted linearly in the CPU. For the data conversion format, please refer to 11.2.3 Signal form, measurement range and expression form of analog quantity.
- **Filtering method**: Select software filters for individual channels.

Using a filter for a fast-changing analog signal can make the measured value more stable.

Note: If the system needs to respond quickly to an AI signal, then the software filter at this point should not be enabled.

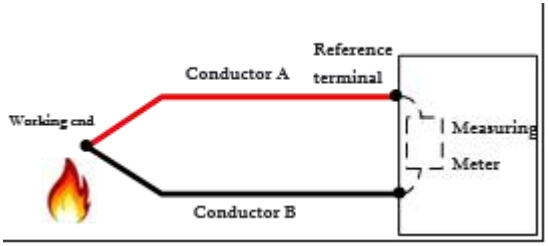
The input sampling of the software filter adopts sliding mode. The software filter has the following options:

- ➢ None --- Do not enable the software filter.。
- ➢ Arithmetic mean --- Take the arithmetic mean of a certain number of signal sample values.
- ➢ Median average --- After removing the maximum and minimum values from a certain number of signal sampling values, take the average of the remaining numbers.

11.2.4 Introduction to thermocouple measurement and external compensation instructions

11.2.4.1 Introduction to the principle and application of thermocouple measurement

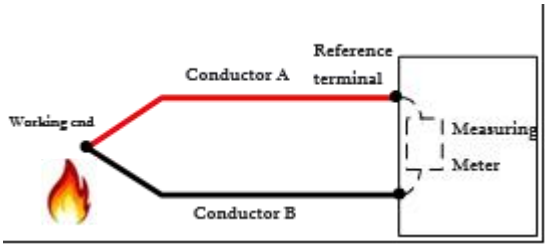
Thermocouple (referred to as TC) is a commonly used temperature measuring element, its working principle is two different materials of uniform conductors composed of a closed loop, when there is a temperature difference at both ends, there will be current through the loop, at this time there is an electromotive force between the two ends - thermoelectric motive force, which is the so-called Seebeck effect. The polarity and magnitude of the thermoelectric motive force are only related to the conductor material and the temperature difference between the two ends.



One end of the conductor A and B is welded to form a joint and placed in the measured medium, which is called the working end or the hot end, assuming that the measured temperature is T at this time; The other end is connected to the measuring instrument, called the reference end or the cold end, assuming that the cold end temperature is T0, for the application in the figure above, the cold end is the terminal of the measuring instrument. The actual value of the thermoelectric motive force measured in the measuring instrument is E(measured value) = E(T)+E(T0), and the actual temperature value is finally calculated according to the measured value of the electromotive force, the cold end temperature and the corresponding thermocouple index table, so the cold end temperature T0 must be accurately measured for the temperature compensation in the above calculation. In addition, the cold end temperature should be kept stable in the application, such as avoiding the position of strong convection when installing, avoiding the position of direct fan blowing, etc., if the cold end temperature is unstable, then obviously the TC measurement results will fluctuate greatly.

According to the cold end temperature measurement method is different, the measuring instrument usually provides the user with internal compensation and external compensation two ways to choose:

- internal compensation: a temperature measuring element is built in the instrument near the terminal, which can measure the temperature where the element is located in real time and is used by the instrument. The internal compensation method is simple to use, but because the cold end temperature measuring element is located inside the instrument, the nearby temperature will be affected by the heating of the instrument during operation, so the temperature measured at the cold end will be higher than the actual temperature at the external terminal, resulting in measurement errors.
- External compensation: The user uses other thermometers to measure the cold end temperature and provides it to the TC thermometer for use. For example, for KPLC, the user can add a thermal resistance module and install a thermal resistance at the terminal position to measure the temperature here.



In practical applications, the distance between the measured object and the measuring instrument is relatively far, so a compensation wire is required from the thermocouple sensor terminal to the measuring instrument terminal, as shown in the figure above. The so-called compensation wire refers to the wire made of a material with the same characteristics as the material of the thermocouple used in a certain temperature range. In the application of thermocouples, there is a "homogeneous conductor law", by the same homogeneous material welded at both ends of the closed loop, no matter how large the conductor cross-sectional area and how the temperature distribution, will not produce contact potential, temperature difference potential offset, the total potential in the loop is zero, so for the same material (or characteristics) of the

thermocouple or compensation wire, Even if a connection point is generated in the middle (such as C, D in the figure above) or there are more connection points, the total potential is unrelated to the temperature at the connection point, and ultimately only related to the hot end temperature T and the cold end temperature T0, which is also the basic principle of normal measurement of various thermocouple thermometers. In practical applications, there are some basic principles for the selection and wiring of compensation wires:

1）The compensation wire must be based on the type of thermocouple used. For example, the type K thermocouple must use the compensation wire of the type K thermocouple.

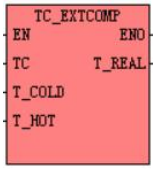
2)The compensation wire has a suitable temperature range and error, which should be selected according to the application requirements. For example, the applicable temperature range is -20-100 ° C, some are -20-200 ° C, the common grade error is ±2.5 ° C, and the more precise error is ±1.5 ° C, so it is necessary to choose according to the application needs.

3)Compensation wire should not be too long, according to general experience within 15 meters is appropriate.

4)Thermocouple signal is mV level of small signal, highly susceptible to interference, in the wiring, the compensation wire should be away from the interference source, AC signal and power line, power line, etc., and it is best to use shielding compensation wire, shielding layer single-end reliable grounding.

11.2.4.2External compensation instruction for thermocouple temperature measurement

- instruction and its operand description

	Name	Format	CR	
LD	TC_EXTCOMP			☑ K6 ☑ K6S
IL	TC_EXTCOMP	TC_EXTCOMPT TC, T_COLD, T_HOT,T_REAL	P	

Parameter	Described	Input/Output	Data Type	Allowed memory area

TC	The meaning of the thermocouple type used is as follows: 0--J type; 1-- Type K; 2--E type; 3-- Type S; 4--T type; Other values -- type J.	Input	INT	V、L、Constant
T_Cold	cold end temperature	Input	INT	V、L、AI、Constant
T_Hot	Measurement value of the hot end, that is, measurement value of the TC module	Input	INT	V、L、AI、Constant
T_Real	The calculated true temperature value.	Output	INT	V、L

The values of parameters TC_Cold, TC_Hot, and T_Real are the corresponding temperature values *10. The data types of parameters TC, TC_Cold, and TC_Hot must be constant or constant.

This instruction is used to calculate the external compensation of the thermocouple measurement, and calculates the real temperature value according to the input TC type, cold end temperature and hot end measurement value. Note: When using this directive, the compensation form of the corresponding TC module channel must be selected as "external compensation"!

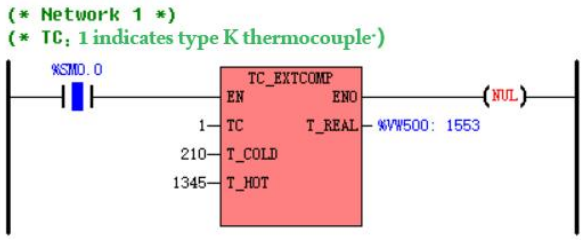
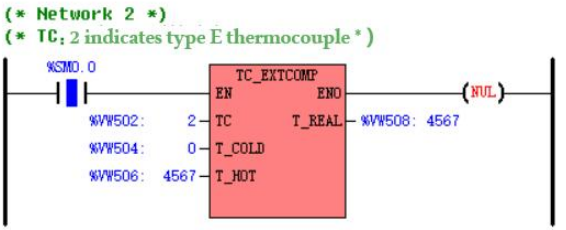
- LD

  If EN is 1, the instruction is executed, otherwise it is not executed.

- IL

  If the CR value is 1, the instruction is executed, otherwise it is not executed.

  - an example of how to use the command

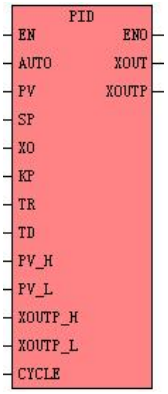
LD		Since SM0.0 is fixed at 1, the instruction is always executed, and the calculated T_Real value is 1553, representing 155.3 ° C
		Since SM0.0 is fixed at 1, the instruction is always executed, and the calculated T_Real value is 4567, representing 456.7 ° C.

11.3 Introduction to PID instruction

The following instructions are all located in the 【Instruction Set】->【PID Loop】group.

- Instruction and its operand description

	Name	Instruction format	Cr values affected	☑ K5

LD	PID			☑ K2 ☑ KS ☑ KW ☑ MK ☑ K6
IL	PID	PID AUTO, PV, SP, XO, KP, TR, TD, PV_H, PV_L, XOUTP_H, XOUTP_L, CYCLE, XOUT, XOUTP	U	

Parameters	Input/Output	Data type	Memory area allowed	Description
AUTO	Input	BOOL	I、Q、V、M、SM、L	Manual/auto flag. 0=manual state, 1=automatic state.
PV	Input	INT	AI、V	PV value, the measured value of the controlled quantity.
SP	Input	INT	V	Set value, the target value of the controlled quantity.
XO	Input	REAL	V	Manual value, ranging from 0.0 to 1.0. In manual state, it is directly the output value of PID.
KP	Input	REAL	V	Scale factor
TR	Input	REAL	V	Integration time, unit: second. 0 means no points.
TD	Input	REAL	V	Differential time, unit: second. 0 means no differentiation.
PV_H	Input	INT	V	Upper limit of PV value
PV_L	Input	INT	V	Lower limit of PV value
XOUTP_H	Input	INT	V	Upper limit of XOUTP value
XOUTP_L	Input	INT	V	Lower limit of XOUTP value

CYCLE	Input	DINT	V	Sampling period, unit: ms. According to the sampling period, the PLC cycles to sample the PV value and execute the PID operation.
XOUT	Output	REAL:	V	The output value of PID, the range is 0.0～1.0.
XOUTP	Output	INT	AQ、V	The output value after linearizing XOUT.

The PID instruction adopts the position type algorithm and the control mode of continuous output.

- LD

If the EN value is 1, execute the PID instruction: every sampling cycle time (CYCLE), the PLC will sample the PV value once and perform PID calculation and output.

- IL

If the CR value is 1, the PID command is executed: every sampling cycle time (CYCLE), the PLC samples the PV value once and performs PID calculation and output.

PID The execution of the instruction does not affect the CR value.

➢ PID detailed instruction

- Manual/automatic status

If the value of the manual/automatic flag bit AUTO is 0, it means that the PID enters the manual state. In the manual state, the PID instruction does not perform any operation, and directly sends the manual value XO specified by the user to the output variable.

If the value of the manual/automatic flag bit AUTO is 1, it means that the PID enters the automatic state. In the automatic state, the PID command performs normal PID calculation and output according to the value of each input parameter.

When the PLC control system is running normally, the PID should be in the automatic state.

- Normalization of PV and SP values

The user needs to input the parameters of PV value, SP value, upper limit of PV value (PV_H) and lower limit of PV value (PV_L). Before performing PID calculation, PLC will automatically perform normalized calculation on PV value and SP value according to these parameters, which is convenient for users to use. These parameters must have the same dimension.

The parameters of PV value and SP value are all INT type, and users can flexibly input values linearly related to them.

For example, assuming that a certain pressure is controlled, the range of the pressure transmitter used for pressure measurement is 0~40MPa, the corresponding output is 4~20mA, and the set value of automatic control is 25MPa. The output of the pressure transmitter is directly connected to the channel AIW0 of the AI module (the signal form is set to 4~20mA, and the conversion value in the PLC is 4000~20000). Then the parameters of the PID instruction can be set as:

	Actual parameters	Description
PV	AIW0	The measured value of AIW0 has a linear relationship with the actual pressure value, so it can be directly used as the PV value.
SP	14000	Indicates 14mA, because the measured value obtained by AIW0 measuring 25MPa is 14mA.
PV_L	4000	The lower output limit of the pressure transmitter.
PV_H	20000	Output upper limit of the pressure transmitter.

- PID output value

The PID instruction has two output values: XOUT and XOUTP.

The range of XOUT is 0.0 to 1.0 (that is, 0.0 to 100.0% in general).

XOUTP is an integer value obtained after linear transformation according to the user-specified output upper limit (XOUTP_H) and output lower limit (XOUTP_L). The calculation formula of XOUTP is as follows:

XOUTP ＝ (XOUTP_H － XOUTP_L)× XOUT ＋ XOUTP_L

The XOUTP parameter is convenient for the user to directly send the output of the PID to a certain channel of the AO module. For example, assuming that the output of the PID needs to be sent to a control

valve through the channel AQW0 of the AO module (the signal form is set to 4~20mA), then the parameters of the PID instruction can be set as:

	Actual parameters	Description
XOUTP	AQW0	AQW0 directly controls the regulating valve, and its value has a linear relationship with the opening of the valve, so AQW0 can be directly used as the output parameter of PID.
XOUTP_L	4000	Output lower limit of AQW0.
XOUTP_H	20000	Output upper limit of AQW0.

- About proportional action, integral action and derivative action

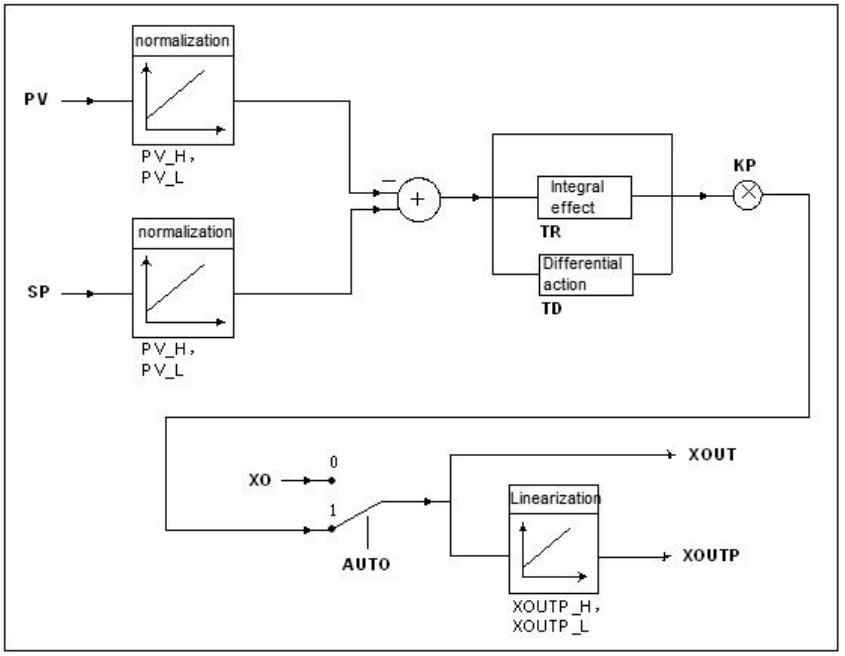
The proportional action is to reflect the deviation of the controlled quantity according to the proportion. Once a deviation occurs, the proportional regulator will immediately produce an adjustment action to reduce the deviation. The user sets the proportional coefficient of PID through the KP parameter. If KP is greater than 0, then PID is positive; if KP is less than 0, then PID is negative. If the proportional coefficient is large, the adjustment speed is fast, but it is easy to cause overshoot and reduce the stability of the system.

The integral action can eliminate the deviation of the controlled quantity. As long as there is a deviation, the integral adjustment will be carried out until the deviation is eliminated. The user sets the integral time constant of PID through the TR parameter (if TR is 0, it means canceling the integral function). The smaller the integral time constant, the stronger the integral action; the larger the integral time constant, the weaker the integral action. A strong integral action also tends to degrade the stability of the system. Integral action is usually combined with the other two control rates to form a PI or PID controller.

The differential action reflects the change rate of the deviation, and can predict the change trend of the deviation, and produce an effective correction before the deviation becomes large. Therefore, the differential action helps to improve the dynamic performance of the system and increase the stability of the system. The user sets the differential time constant of PID through the TD parameter (if TD is 0, it means canceling the differential action). The larger the differential time constant, the stronger the differential action; the smaller the differential time constant, the weaker the differential action. Derivative action is usually combined with the other two control rates to form a PD or PID controller. Generally, in a system with a large lag (such as

temperature control), it is necessary to add differential control.

- PID instruction block diagram



11.4 How to use PID

There are two ways to use PID:

1. Use relevant instructions for programming: call the PID instruction directly in the program, set the corresponding parameters and start programming.

2. Use the PID wizard to set: This method is simple and intuitive, just need to check and set the parameters according to the content provided in the wizard. After setting, the wizard automatically generates the corresponding subroutine, which can be called directly by the user when programming, which saves the tedious programming process of calling the PID instruction.

11.4.1 Programming with PID instructions

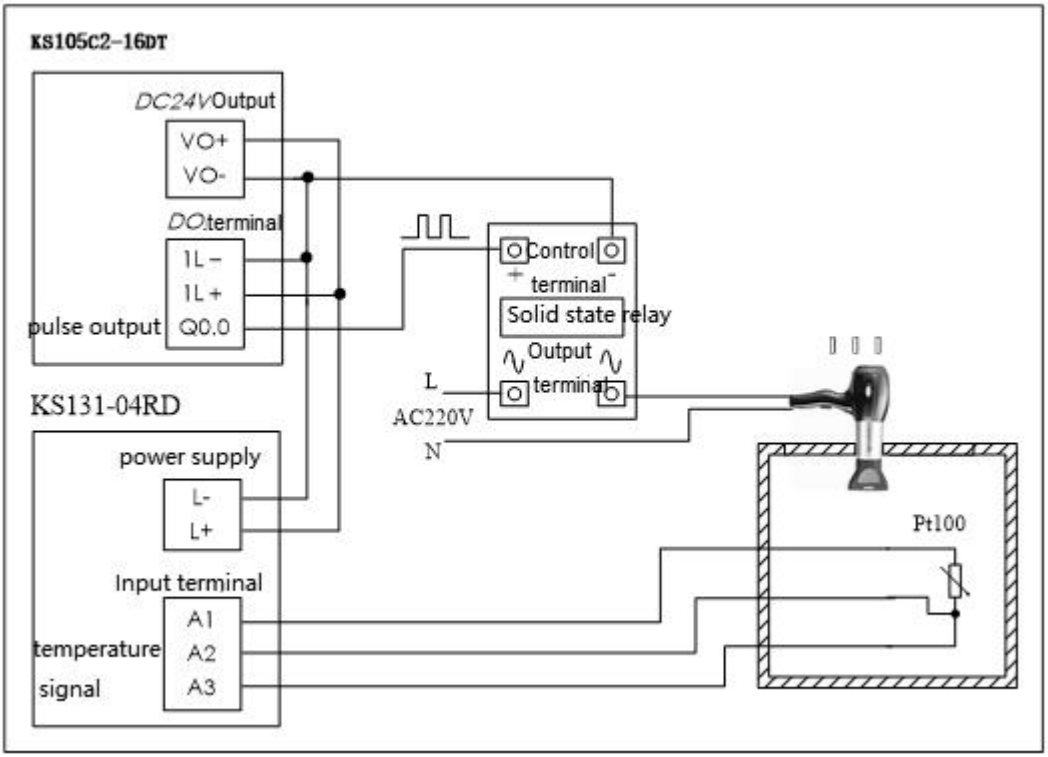
In order to better help users understand how to use the PID command, the following example directly illustrates the use of the PID command.

11.4.1.1 Example of PID usage

First, establish the following simulation system: control the temperature in a box, rely on hair dryer to heat up the temperature, and rely on natural cooling to cool down. The configuration of PLC is: KS105C2-16DT +K S131-04RD. In addition, the modification of parameters such as SP, KP, TR, and TD is completed through an HMI.

In this system, a Pt100 is used to measure the temperature, and its signal is connected to the AIW0 channel of the RD module. The Q0.0 output pulse string is connected to the control terminal of the solid-state relay, and the output of the PID is used to adjust the pulse width accordingly, so as to control the on-off time of the hair dryer and realize temperature control.

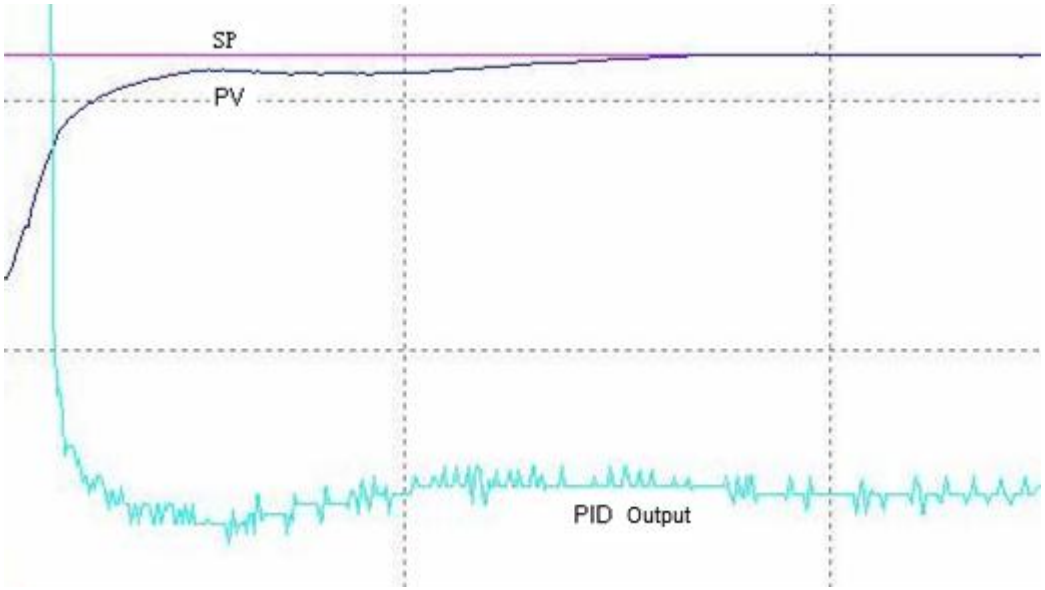
The simulation system is shown in the figure below:



The following is a brief description of the programming ideas in this example:

1) A simple PWM output is implemented using a timer.。

2) The following control methods are used in the program:If the PV value is lower than the SP value and exceeds 5°C, use the PID manual mode and output 100%; if the deviation between the PV value and the SP value is less than 5°C, use the PID automatic mode. The advantage of this method is: when the deviation is large, the PID manual mode is used, which avoids the problem of large overshoot caused by the excessive integral accumulation value in the automatic mode; in addition, it is also beneficial to reduce the adjustment time. When the SP is set to 50°C, the control effect is as shown in the figure below.

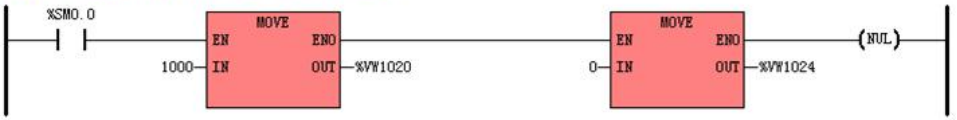


The program is as follows:

LD

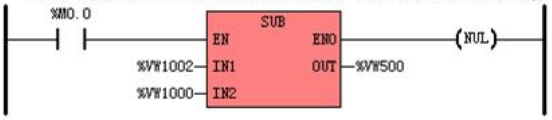
```
(* Network 0 *)
(* The sampling period is an important parameter that can affect the control effect and needs to be set
   according to the actual characteristics of the controlled object.                                  *)
```

```
(* Network 1 *)
(* The maximum and minimum values of PV *)
```

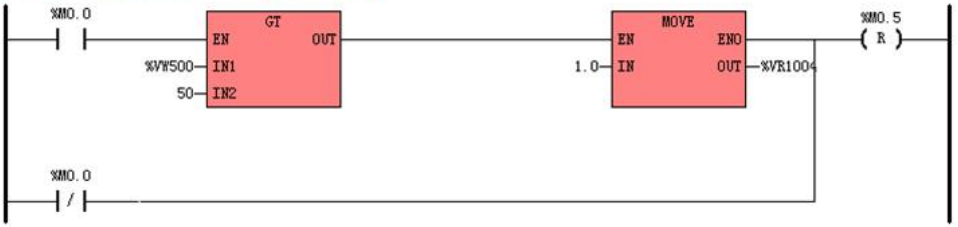



```
(* Network 2 *)
(* The maximum and minimum values of XOUT_P *)
```

```
(* Network 3 *)
(* M0.0- Start/stop automatic control button on HMI.
   Control method: If the PV value is lower than the SP value by more than 5 ° C, use
   PID manual mode and output 100%;                                                    *)
   If the deviation between PV value and SP value is less than 5 ° c, use PID automatic
```

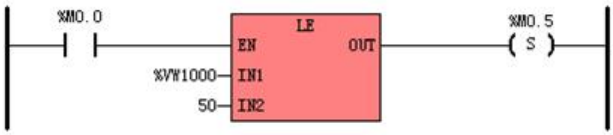



```
(* Network 4 *)
(* M0.5--PID manual/automatic parameter *)
```

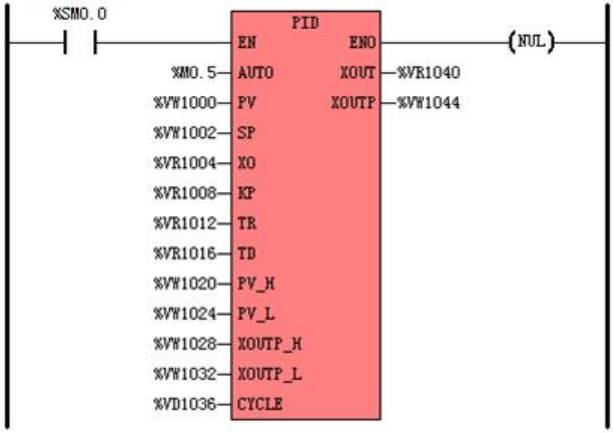


LD

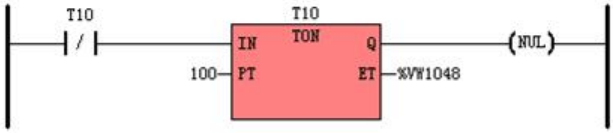
```
(* Network 5 *)
```



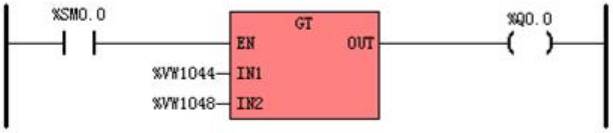

```
(* Network 6 *)
(* SP, X0, KP, TR, TD and other parameters need to be set through the HMI. *)
```




```
(* Network 7 *)
(* Output a pulse train with a period of 1 second, and adjust the pulse width using the output of PID. *)
```




```
(* Network 8 *)
```



IL

```
(* Network 0 *)
(* Set PV value and sampling period. *)
(The sampling period is an important parameter that can affect the control effect and needs to be set according to the actual characteristics of the controlled object.)
LD      %SM0.0
MOVE    %AIW0, %VW1000
MOVE    DI#1000, %VD1036
(* Network 1 *)
(*The maximum and minimum values of PV *)
LD      %SM0.0
MOVE    1000, %VW1020
MOVE    0, %VW1024
(*Maximum and minimum values of XOUT-P *)
MOVE    100, %VW1028
MOVE    0, %VW1032
(* Network 3 *)
(* M0.0-- Start/stop automatic control button on HMI. *)
(* Control method: If the PV value is lower than the SP value by more than 5 ℃, use PID manual mode and output 100%; *)
(If the deviation between PV value and SP value is less than 5 ℃, use PID automatic mode.)
LD      %M0.0
MOVE    %VW1002, %VW500
SUB     %VW1000, %VW500
(* Network 4 *)
(*M0.5---  PID manual/automatic mode parameters *)
LD      %M0.0
GT      %VW500, 50
MOVE    1.0, %VR1004
ORN     %M0.0
R       %M0.5
(* Network 5 *)
LD      %M0.0
LE      %VW1000, 50
S       %M0.5
(* Network 6 *)
(*SP, X0, KP, TR, TD and other parameters need to be set through the HMI.*)
LD      %SM0.0
PID     %M0.5, %VW1000, %VW1002, %VR1004, %VR1008, %VR1012, %VR1016, %VW1020
            , %VW1024, %VW1028, %VW1032, %VD1036, %VR1040, %VW1044
(* Network 7 *)
(*Output a pulse train with a period of 1 second, and adjust the pulse width using the output of PID.*)
LDN     T10
TON     T10, 100
(* Network 8 *)
LD      %SM0.0
GT      %VW1044, T10
ST      %Q0.0
```

11.4.2 Using the PID Wizard

11.4.2.1 PID wizard setting and usage example

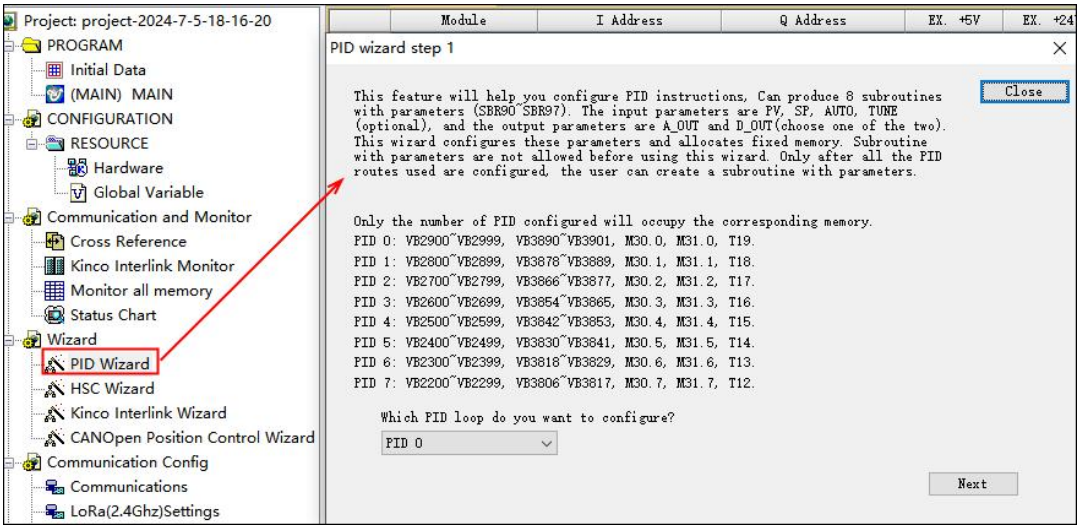
The following also takes the simulation system in 11.4.1.1 usage example of PID as an example to introduce the setting of the PID wizard.

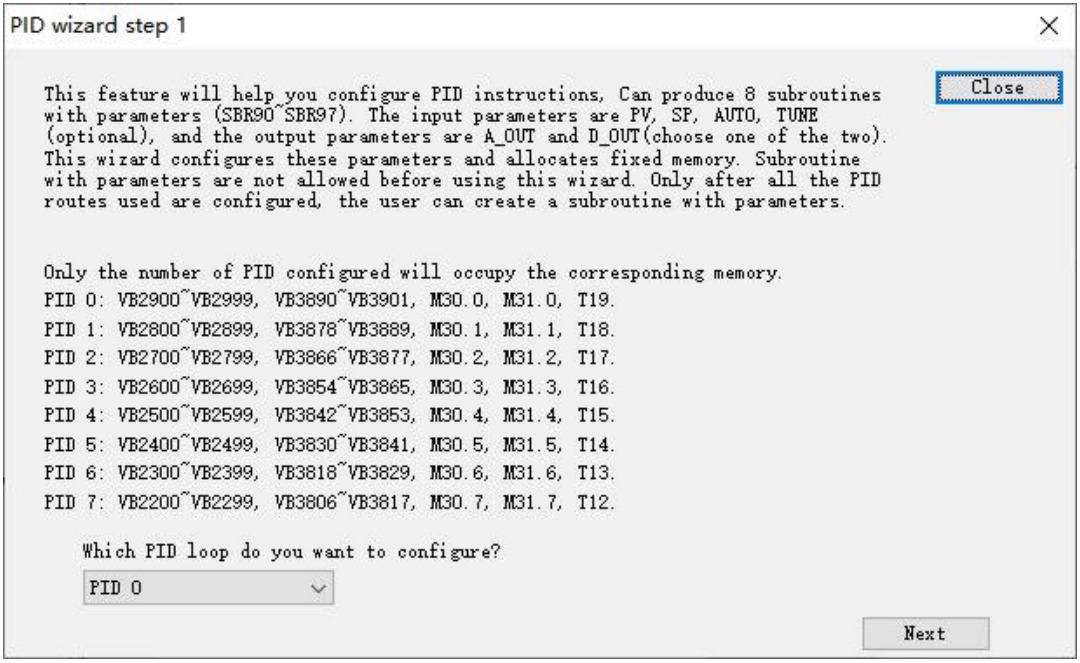
For the parameters of the following PID instructions, please refer to the relevant content in the chapter 11.3 PID instruction introduction, and will not be introduced in detail here.。

Kinco-K series PLC provides 8 channels of PID output and setting PID needs to occupy a part of the memory address. The following table shows some memory addresses used by each PID (mainly for customer modification). The parameters set by yourself include PID input terminals X0, KP, TR, and TD, and the sub-meters represent manual output values, P ratio, I integral, and D differential.

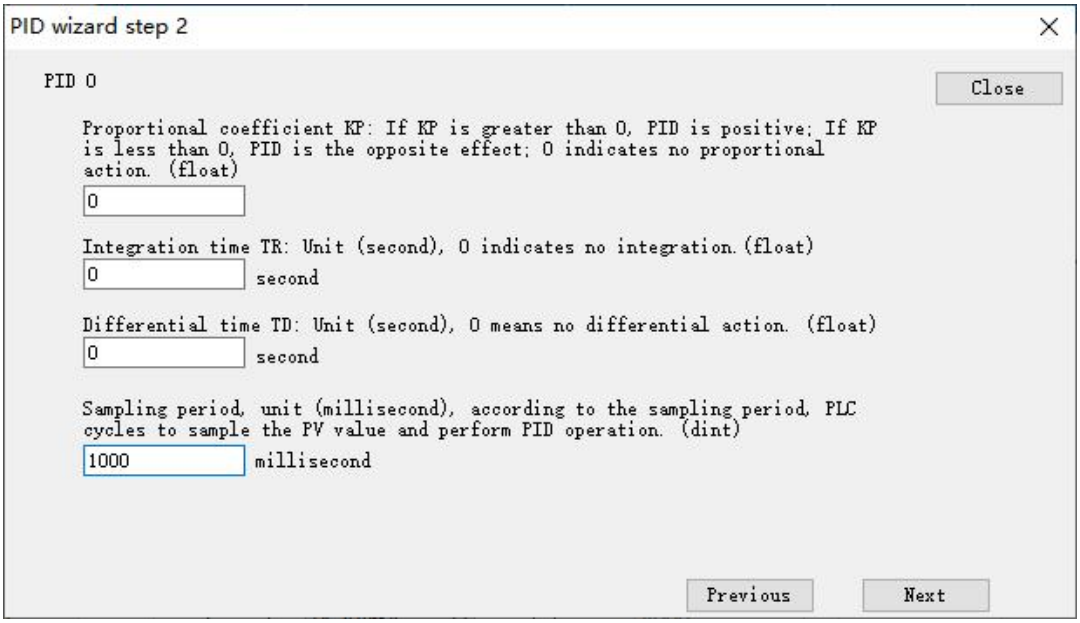
PID loop	Manual value X0	KP ratio	TR integration time	TD differential time
loop0PID	%VR2916	%VR3898	%VR3894	%VR3890
loop1PID	%VR2816	%VR3886	%VR3882	%VR3878
loop2PID	%VR2716	%VR3874	%VR3870	%VR3866
loop3PID	%VR2616	%VR3862	%VR3858	%VR3854
loop4PID	%VR2516	%VR3850	%VR3846	%VR3842
loop5PID	%VR2416	%VR3838	%VR3834	%VR3830
loop6PID	%VR2316	%VR3826	%VR3822	%VR3818
loop7PID	%VR2216	%VR3814	%VR3810	%VR3806

1、Follow the wizard prompts to select the number of PIDs you want to use. Setting the PID will occupy a part of the memory address, and selecting different output channels will occupy different memory addresses. Users need to pay attention to these occupied memory areas when programming later.



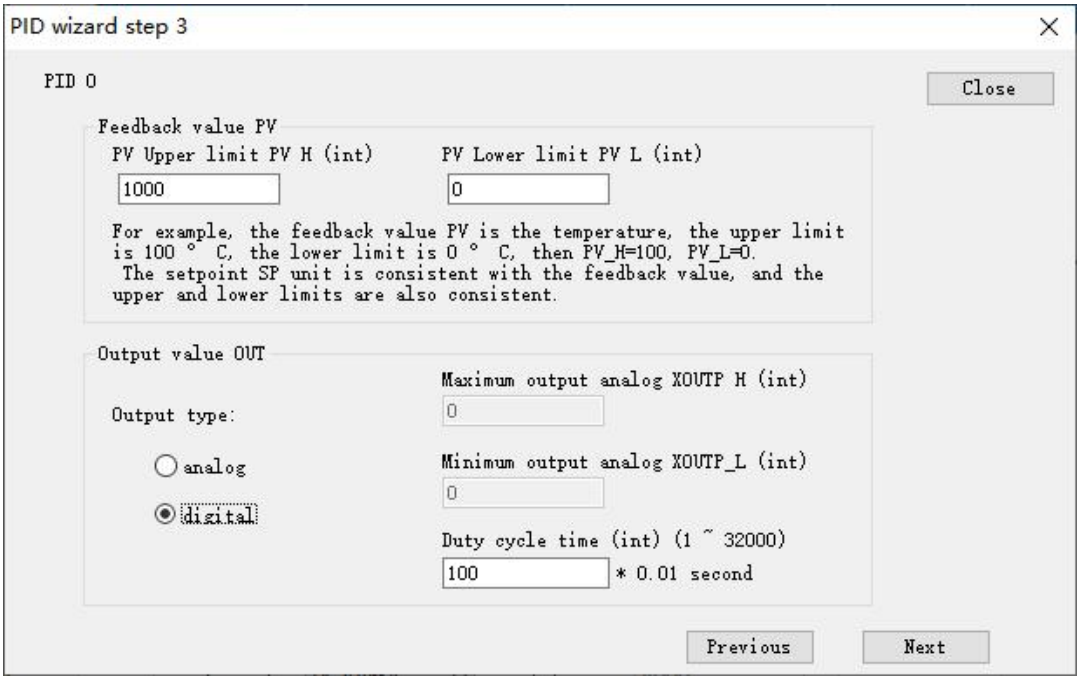


2、Set the sampling period of PID and the initial PID parameters. The address of PID parameters can be corresponding to the HMI to facilitate later debugging and adjustment of parameters.

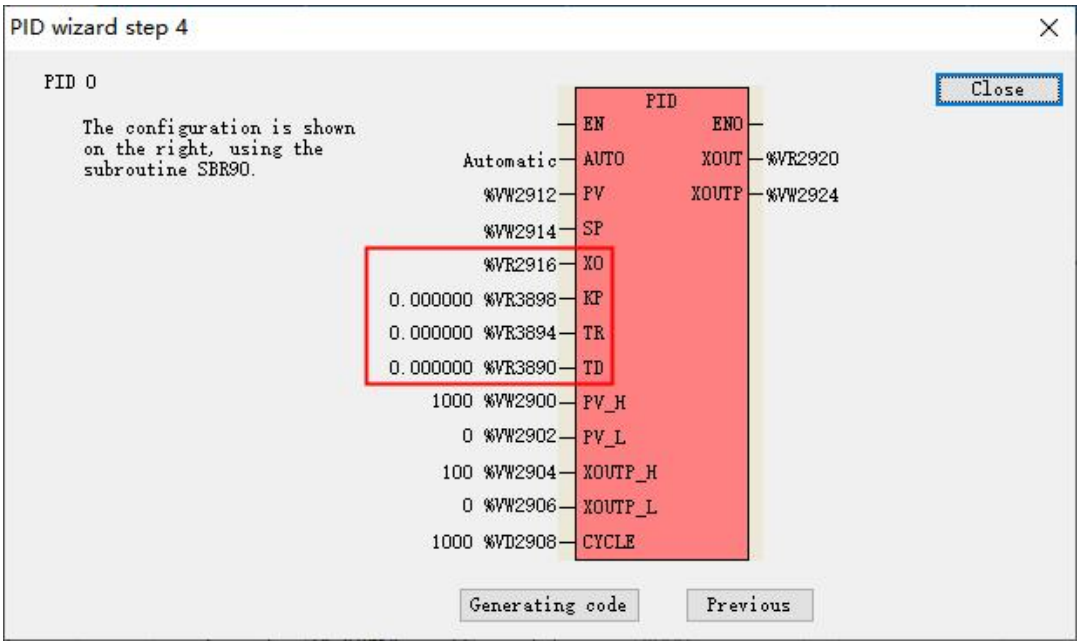


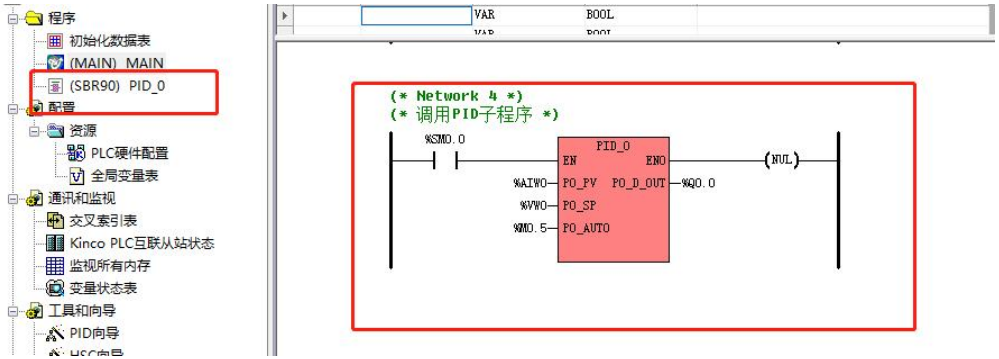
3、Set the upper and lower limits of the PV value, and the output type can be set to analog or digital. The analog quantity is the form of output voltage or current to control the external temperature, pressure, flow and other controlled objects. Digital output refers to controlling the external controlled object by adjusting the duty cycle of the PWM wave in real time in a form similar to the output of the PWM wave. The duty cycle time can be regarded as the cycle of the PWM wave.

The set values of PV_L, PV_H, XOUTP_L, and XOUTP_H are the expression forms of PLC internal values. For details, please refer to 11.2.3 Signal form, measurement range and expression form of analog quantity.



4、The PID command and its parameters generated by the wizard are as follows. After confirming that it is correct, click Generate Code. The corresponding subroutine will be generated in the software as shown in the figure below. The parameters we need to set by ourselves include PID input terminals X0, KP, TR, and TD, which respectively represent manual output value, P ratio, I integral, and D differential. Because the specific PLC addresses corresponding to the 8-way PID are different, please refer to the PID command generated in the wizard or directly check the address of the input terminal of the PID command in the generated subroutine, so as to map these addresses to the HMI for easy debugging.





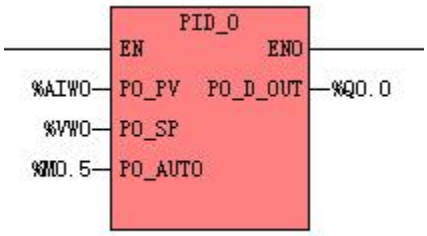
5、Introduction to the input and output parameters of the PID subroutine generated by the wizard:

PO_PV: The measured value of the external controlled object.

PO_SP: The setting value of the controlled object.

PO_AUTO: Manual automatic selection of PID control, 0-manual, 1-automatic.

PO_D_OUT: PID control output (analog/digital), consistent with the settings in the wizard.



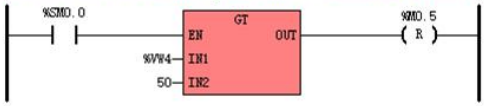
5、Write the corresponding PID manual and automatic switching program in the MAIN main program and call the PID subroutine PID_0 to realize the same temperature control function as the simulation system in 11.4.1.1 PID usage example.

LD

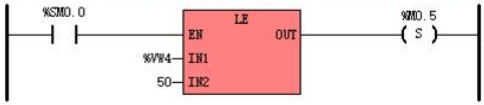
MAIN:

(* Network 0 *)
(* Obtain the difference between the measured PV value and the set value SP and place it in VW4, where VW0 represents the set value *)

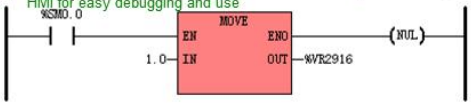
(* Network 1 *)
(* When the SP-PV value is greater than 5 ° C, manual PID control is adopted *)



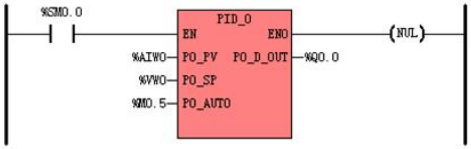
(* Network 2 *)
(* When the SP-PV value is less than 5 ° C, automatic PID control is adopted *)



(* Network 3 *)
(* The output values of VR2916 manual PID, including P, I, and D parameters, can be given in HMI for easy debugging and use *)



(* Network 4 *)
(* Calling PID subroutine, M0.5 manual PID selection enable bit
Q0.0 digital output is used to generate PWM wave ratio duty cycle for temperature control *)



Chapter 12 Data Retention and Data Backup

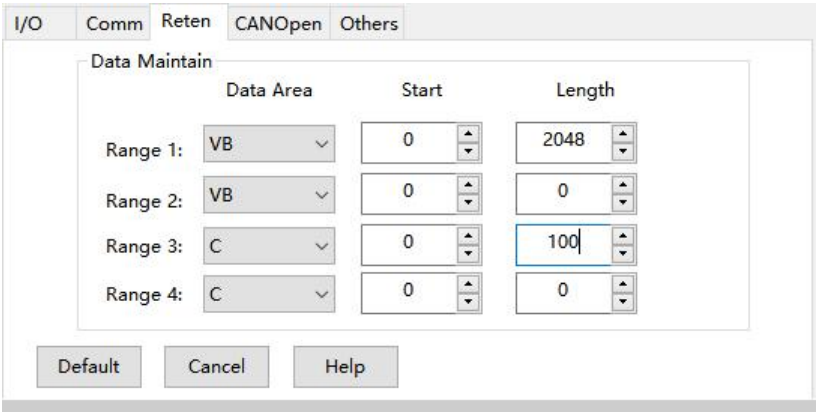
12.1 Data Retention and Data Backup

The data retention function is to use the backup battery to maintain the power supply after the PLC is powered off to maintain the RAM memory data. Because the internal RAM is operated, the data read and write speed is fast, and the memory life is unlimited. But when the battery runs out, all data will be lost.。

The data backup function is that the PLC saves the data in the permanent memory, which can be stored permanently. However, permanent memory has limitations in lifespan and writing speed, so it cannot be written frequently and quickly.

- **Data retention**

Data retention means that the data in the RAM of the CPU module remains in the state at the moment of power failure after power failure, and will be used continuously when the CPU is powered on next time. A backup lithium battery (non-rechargeable) is provided inside the CPU module for data retention. During power outages, the backup battery powers the RAM and maintains the data in the RAM. Users need to use KincoBuilder software to select the type of data area to be kept (such as V area, C area, etc.) and the start and end addresses and length in the [PLC Hardware Configuration] of the user project. At most, all V areas and C areas can be maintained. As shown below:



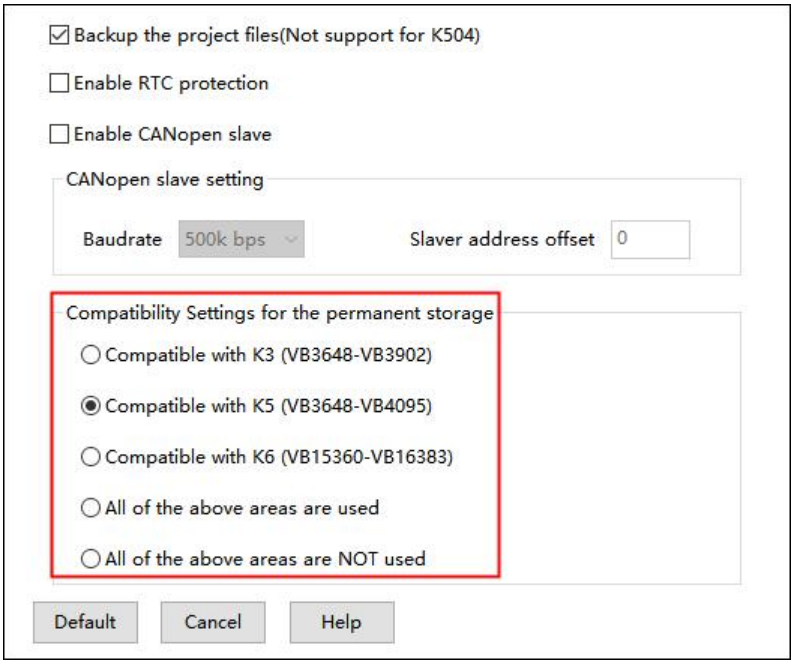
- **Data backup**

Data backup means that the CPU module creates an area in the permanent memory to store user data. The data in this area will never be lost when the power is off, and will be used by the CPU when the power is turned on next time. **The CPU module provides E2PROM memory for data backup function, since E2PROM only has a writing life of 1 million times, so users should pay attention to avoid permanently backing up frequently-changing data!**

K series PLC provides a data backup area in the V area, the data in this area will be automatically written into the permanent memory, and the user can directly use these memory areas. The lengths of the data retention area and data backup area of each type of PLC are not the same. The following table lists the lengths of the data retention area and data backup area of each model::

It should be noted that for the permanent data backup area, if it is also set to data retention in [PLC Hardware Configuration], then the permanent data backup function is given priority. That is, the data in the permanent backup memory will overwrite the data in the battery-backed RAM.

Because the K series PLC is constantly being upgraded, it can be seen from the above table that the data backup lengths of each CPU are not exactly the same. If you need to be compatible with the previous K3, K5 series, etc. without changing the program, you need to set it on the [Other] page of the CPU module in [PLC Hardware Configuration], as shown in the figure below:



- [Compatible with K3's permanent storage settings] - all supported

Selecting this item means that VB3648-3092 will take effect as a data backup area, and the data in this area will be automatically written into the permanent memory.

- [Permanent storage settings compatible with K5] -all supported except K3

Selecting this item means that VB3648-4095 will take effect as a data backup area, and the data in this area will be automatically written into the permanent memory.

- [Permanent storage settings compatible with K6]-only supported by K6, KS101M, K209M

Selecting this item means that VB15360-16383 will take effect as a data backup area, and the data in this area will be automatically written into the permanent memory.

- [All the above areas are automatically and permanently stored] - only supported by K6

This item means that VB3648-4095 and VB15360-16383 take effect at the same time as the data backup area, and the data in this area will be automatically written into the permanent memory

- [All the above areas are not permanently stored] - only supported by K6

Selecting this item means that all V areas do not have permanent storage areas.

- [Backup the entire user project to the permanent storage of the PLC]

By default, the K series only saves the hardware configuration and user program information in the user project in the PLC, but does not save variable names (global and local), program names, comments, etc.

If this option is selected, complete user project information will be saved in the PLC, including hardware configuration, user program, program name, all variable names, comments, etc. The project uploaded by the user from the PLC will be exactly the same as the downloaded project.

12.2 Backup battery and battery power indicator

Kinco-K series PLC uses lithium batteries of specific specifications as backup batteries. When the power is off, the backup battery is used to supply power to the real-time clock to maintain the operation of the clock, and also to supply power to the RAM for data retention

The backup battery can be removed, but it needs to be replaced by opening the cover. After opening the case, you can see the battery as shown on the right, and the user can replace it by himself. The battery is a CR2032 3V lithium battery with a connector, the shape is shown on the right, and the user can order the battery separately.

At room temperature, the typical life of the battery is 5 years, and the accumulative time of power failure is not less than 3 years.

KPLC provides the following registers in the SM area to indicate the "battery power" information:

SM0.7	read only. SM0.7 is TRUE means the battery voltage is low, SM0.7 is FALSE means the battery voltage is normal. **(Applicable to other series except K5)**
SMW10	read only. Store the voltage value of the backup battery, unit: 0.01V. **(Only supported by K5)** If the power supply of the backup battery continues to be lower than 2.6V, the PLC will generate an alarm of "low charge of the backup battery".

Chapter 13 Troubleshooting

Kinco-K series PLC divides the errors that occur during operation into three levels: fatal errors, serious errors, and general errors. When an error occurs when the PLC is running, different processing measures will be taken according to the level of the error, and the error codes will be stored in sequence according to the order of occurrence for the user to read and analyze.

Regardless of the level of errors that have occurred, we recommend that users analyze and check in time after discovering error indications to avoid unnecessary losses!

Only when the PLC can run, the PLC can detect and analyze various errors that occur! If the PLC hardware is damaged so badly that it cannot run various software programs at all, all functions will fail.

13.1 error type

13.1.1 fatal error

The reason for the fatal error is that the PLC detects that the hardware chip has an unknown fault that may endanger the correct operation, or the watchdog of the PLC is triggered (for example, the amount of FOR loop in the program is too large, the JMP instruction causes an infinite loop), etc... Fatal errors may cause the PLC to fail to continue working, and we cannot confirm whether unexpected and dangerous errors will occur again in the PLC. Therefore, the goal of dealing with fatal errors is to immediately put the PLC into a safe state.

When a fatal error occurs, the PLC will automatically take the following measures: Immediately exit the normal scanning state, and directly reset or enter the independent safety subsystem operation according to the value of SM2.0. If the value of SM2.0 is 0, when a fatal error occurs, the PLC will enter the independent safety subsystem operation; if the value of SM2.0 is 1, when a fatal error occurs, the PLC will directly reset and restart. **SM2.0 defaults to 0 after power-on, it is recommended that users do not modify its value**

unless necessary, so that the PLC can enter the security subsystem!

13.1.1.1 About the Security Subsystem

When a fatal error occurs, the PLC cannot operate normally. At this time, if the value of SM2.0 is 0, the PLC will automatically enter an independent safety subsystem to operate. In the safety subsystem, the PLC can no longer perform normal scanning, but it will allow all output channels to output the user-specified [Shutdown Hold] value, and provide some methods so that the user can try to eliminate software errors.

- **Status indication**

After entering the security subsystem, the indicator light of the PLC will indicate with a special state.

The following table lists the status of each indicator light of each series of PLC under the safety subsystem. Users can judge whether the PLC has entered the safety subsystem due to a fatal error based on the status of these lights.

Series	indicator light			
	Run	Stop	Comm	Err
K6/K2/K5	always off	always on	always off	flashing
KS/KW	always off	--	--	flashing
MK Series	flashing	--	--	flashing

Note: "--" indicates that there is no such indicator light or its status is ignored.

- **Function**

After entering the security subsystem, the PLC provides the following functions:

- All output points (DO, AO) immediately output the value of "Stop Output" defined by the user in "PLC Hardware Configuration".
- PORT1 (RS485) can work, and can receive a special "clear" command to clear the data in the PLC, so that the PLC can be restored to the factory state. At this time, the communication parameters of PORT1 are: baud rate 9600, no parity, 8 data bits, 1 stop bit.
- Control the status of each indicator light (as described above) to inform the user that a fatal error has just occurred in the PLC.

13.1.1.2 How to resolve fatal errors

Possible causes and solutions for fatal errors are as follows:

1) Key hardware chips (such as MCU, RAM, etc.) are damaged, causing the PLC to completely fail to operate normally. If the fatal error caused by this reason cannot be resolved by itself, the PLC can only be returned to the factory for repair.
2) There is a bug in the PLC system program. If the bug is triggered during the application process, it may cause a fatal error in the PLC. Solution: Contact Kinco, and Kinco will solve the bug of the PLC system program and provide the new version of the program to the user after the solution. The user can upgrade the faulty PLC to the new version of the system program by himself. For the upgrade method, please refer to Appendix B to update the system program; or after determining the cause of the bug, the user modifies the program to avoid using related functions, and then downloads the new program to the PLC to run.
3) There is a serious error in the user program. For example, watchdog failures caused by excessive FOR loops, JMP instruction errors caused by infinite loops, etc.; illegal addresses are manipulated when using pointers; when using some instructions that need to operate memory blocks, the parameter values are unreasonable, resulting in access to illegal memory addresses, etc. Solution: After determining the cause, the user corrects the errors in the program and re-downloads it to the PLC for operation.

When a fatal error occurs, the PLC will automatically enter the safety subsystem to run. Whether the user wants to update the system program or modify the user program and download it again, he must first let the PLC leave the safety subsystem and enter the normal system operation process.

The user can clear all the data in the PLC, restore the PLC to the factory initial state, and then the PLC can run normally, and the user can download the modified user program to the PLC to run or update the PLC to a new version of the system program. Please refer to 13.5 How to restore the CPU to the factory default state? Learn more. There are several operation methods to clear the PLC:

13.1.1.2.1 For PLCs with "RUN/STOP" switch

The user can power off the PLC first, turn the RUN/STOP switch to the STOP position, and then power on the PLC again. After power-on, the user needs to observe the status of the indicator lights of the PLC first, and deal with it according to the following situations:

- If the indicator light still indicates the status of the above-mentioned security subsystem, the user can first try the third method described below. That is, in the KincoBuilder software, use the [Tools] → [Clear (only for fatal errors)] menu command to clear the PLC. If the clearing fails, it indicates that the fatal error may be caused by a core chip failure, so the user cannot consult and solve it.
- If the status of the indicator light is: the RUN light is off; if there is a STOP light, the STOP light is always on at the same time; the ERR light is always on or off. This indicator light status indicates that the PLC has normally entered the STOP state, and this fatal error is caused by software, so the user can repair it. At this time, the parameters of each communication port of the PLC are as follows:

Communication port	Communication parameters
USB	It is virtualized as a serial port, and the previous serial port number remains unchanged in Windows. [PLC station number] is 1, other communication parameters can be ignored.
Ethernet port	All communication parameters (including IP addresses, port numbers, etc.) remain unchanged from previous values.
serial port	The parameters of the first serial port (if there is PORT0, it is PORT0; if there is no PORT0, it is PORT1) are automatically set by the PLC: baud rate 9600, no parity, 8 data bits, 1 stop bit.

Users can choose a communication port to connect PLC and computer. Configure the parameters of the corresponding communication port of the computer according to the parameters in the above table, and then execute the [PLC]→[Clear] menu command in the KincoBuilder software to restore the PLC to the factory state, and then power off and restart the PLC.

13.1.1.2.2 For PLCs with "CLR" clear button

Some series of KPLC products (including MK series) provide a CLR clear button. Users can directly operate the CLR button to clear all data in the PLC and

restore it to the factory state.

The following takes the MK series all-in-one machine as an example to fully describe the clearing steps:

① Step 1: Press the CLR button and hold it for more than 5 seconds, the ERR indicator will start flashing at a cycle of about 2 seconds.

② Step 2: Release the CLR button, after about 2 seconds, the ERR indicator light will start to flash rapidly with a period of about 600ms. At this time, the PLC is in a waiting state, and the user needs to perform the next operation within 60 seconds, otherwise the PLC will end this process and return to the state before operating the CLR button.

③ Step 3: Press the CLR button and hold it for more than 5 seconds, then the PLC will enter the clearing process, at this time the ERR light is always on, and the user can release the button. Depending on the series of PLCs, the time required for the clearing process will vary, usually from a few seconds to about 20 seconds. During the clearing process, the PLC will not respond to any other commands. When the clearing process is over, if the clearing is successful, the ERR light will flash twice at a cycle of 1 second, and then turn off; if the clearing fails, the ERR light will flash four times at a cycle of 600ms, and then turn off.

④ Step 4: Power off and restart the PLC, and the PLC will return to the factory state and run.

13.1.1.2.3 For all PLCs

In the safety subsystem, PORT1 (RS485) ports of all series of PLCs will start working, and users can use PORT1 ports to clear the data in the PLC.

In the KincoBuilder software, the user executes the [Tools] → [Clear (only for fatal errors)] menu command, and the window shown in the figure below will pop up, and the user can operate step by step according to the prompt information on the interface.

13.1.2 Serious error

Severe errors will cause the PLC to fail to perform one or several important functions, and cannot continue to run the user program correctly, but the result is predictable. When a serious error occurs, the PLC will automatically take the following measures:

1) Put the PLC in the STOP state immediately, and all output points (DO, AO) immediately output the "stop output" value set by the user.
2) The ERR indicator is always on, the RUN indicator is off, and if there is a STOP indicator, the STOP indicator is always on.
3) Record fault codes in sequence and allow users to read these records through KincoBuilder and Modbus RTU protocol.

13.1.3 general error

A general error is that an error occurs when the PLC executes a certain function, but the PLC can tolerate this error and can continue to run other parts of the user program correctly, and the result is predictable. When a general error occurs, the PLC will automatically take the following measures:

1) The PLC continues to run.
2) The ERR indicator is always on.

3) Record fault codes in sequence and allow users to read these records through KincoBuilder and Modbus RTU protocol.

13.2 Error code

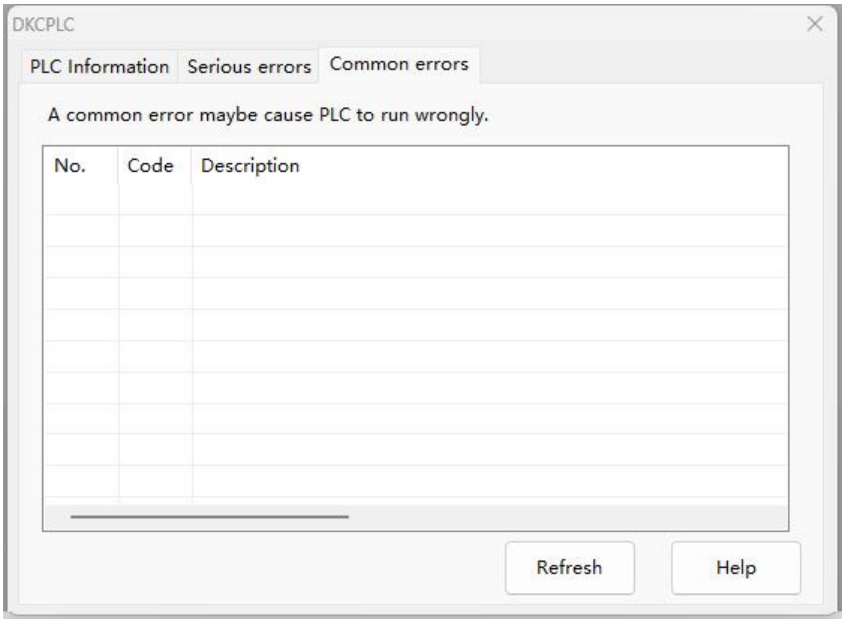
Code	Description
Serious errors	
20	The CPU type in "PLC Hardware Configuration" is inconsistent with the actually connected CPU type.
21	There is an incorrect expansion module (a module of another series is used) in the "PLC Hardware Configuration".
25	When powering on, reading the PLC protection type failed.
26	Failed to read target file (plaintext) at power-on.
27	When powering on, reading the target file (ciphertext) failed.
28	When the power is turned on, the CRC of the target file is wrong.
29	When powering on, check that there are unknown commands in the PLC program.
30	When powering on, check that the number of parameters in the PLC program exceeds the limit.
35	At power-up, reading persistent memory data failed.
40	At runtime, the JMP instruction failed to jump.
41	At runtime, the subroutine call failed.
42	While running, an interrupt subroutine call failed.
60	When powering on, the first expansion module times out and does not respond.
61	On power up, the 1st module responds with an error.
62	The first expansion module does not match the type in the hardware configuration.
65	When powering on, the second expansion module timed out and did not respond.
66	On power up, the 2nd module responds with an error.
67	The 2nd expansion module does not match the type in the hardware configuration.
70	When powering on, the third expansion module timed out and did not respond.
71	On power up, the 3rd module responds with an error.
72	The 3rd expansion module does not match the type in the hardware configuration.
75	When powered on, the 4th expansion module timed out and did not respond.
76	On power up, the 4th module responds with an error.
77	The 4th expansion module does not match the type in the hardware configuration.
80	When powered on, the fifth expansion module timed out and did not respond.
81	On power up, the 5th module responds with an error.
82	The 5th expansion module does not match the type in the hardware configuration.
85	When powered on, the 6th expansion module timed out and did not respond.
86	On power up, the 6th module responds with an error.
87	The 6th expansion module does not match the type in the hardware configuration.
90	When powered on, the 7th expansion module timed out and did not respond.
91	On power up, the 7th module responds with an error.
92	The 7th expansion module does not match the type in the hardware configuration.

100	When powered on, the 8th expansion module timed out and did not respond.
101	On power up, the 8th module responds with an error.
102	The 8th expansion module does not match the type in the hardware configuration.
105	When powered on, the 9th expansion module timed out and did not respond.
106	On power up, the 9th module responds with an error.
107	The 9th expansion module does not match the type in the hardware configuration.
110	When powered on, the 10th expansion module timed out and did not respond.
111	On power up, the 10th module responds with an error.
112	The 10th expansion module does not match the type in the hardware configuration.
115	When powered on, the 11th expansion module timed out and did not respond.
116	On power up, the 11th module responds with an error.
117	The 11th expansion module does not match the type in the hardware configuration.
120	When powered on, the 12th expansion module timed out and did not respond.
121	On power up, the 12th module responds with an error.
122	The 12th expansion module does not match the type in the hardware configuration.
95	When powering on, the CPU module fails to send extended communication packets.
96	When powered on, the CPU module expansion bus enters the error passive state.
97	When powered on, the CPU module expansion bus enters the bus off state.
General errors	
136	When powering on, the calibration value of the AI channel failed to be read.
137	When powering on, the calibration value of the AO channel failed to be read.
138	During calibration, the calibration value of the AI channel failed to be written.
139	During calibration, the calibration value of the AO channel failed to be written.
150	During operation, the heartbeat of the first expansion module times out, and the expansion bus communication of this module may be interrupted.
151	During operation, a fault message from the first expansion module is received.
154	During operation, the heartbeat of the second expansion module times out, and the expansion bus communication of this module may be interrupted.
155	During operation, a fault message from the second expansion module is received.
158	During operation, the heartbeat of the third expansion module times out, and the expansion bus communication of this module may be interrupted.
159	During operation, a fault message from the third expansion module is received.
162	During operation, the heartbeat of the fourth expansion module times out, and the expansion bus communication of this module may be interrupted.
163	During operation, a fault message from the fourth expansion module is received.
166	During operation, the heartbeat of the fifth expansion module times out, and the expansion bus communication of this module may be interrupted.
167	During operation, a fault message from the fifth expansion module is received.
170	During operation, the heartbeat of the sixth expansion module times out, and the expansion bus communication of this module may be interrupted.
171	During operation, a fault message from the sixth expansion module is received.
174	During operation, the heartbeat of the seventh expansion module times out, and the expansion bus communication of this module may be interrupted.
175	During operation, a fault message from the seventh expansion module is received.
178	During operation, the heartbeat of the eighth expansion module times out, and the

	expansion bus communication of this module may be interrupted.
179	During operation, a fault message from the eighth expansion module is received.
182	During operation, the heartbeat of the ninth expansion module times out, and the expansion bus communication of this module may be interrupted.
183	During operation, a fault message from the ninth expansion module is received.
186	During operation, the heartbeat of the tenth expansion module times out, and the expansion bus communication of this module may be interrupted.
187	During operation, a fault message from the 10th expansion module is received.
190	During operation, the heartbeat of the 11th expansion module times out, and the expansion bus communication of this module may be interrupted.
191	During operation, a fault message from the 11th expansion module is received.
194	During operation, the heartbeat of the 12th expansion module times out, and the expansion bus communication of this module may be interrupted.
195	During operation, a fault message from the 12th expansion module is received.
300	When running, the AI channel of the main body once had a DMA error.
301	During runtime, the sampling conversion process of the AI channel of the main body has stopped.
320	At runtime, a frame format error occurred in the expansion bus communication.
321	At runtime, the expansion bus communication has entered the error active state.
322	At runtime, the expansion bus communication ever went into error passive state.
323	During runtime, the expansion bus was turned off. NOTE: Bus shutdown is caused by excessive communication errors on the bus.
324	While running, extended communication failure: receive buffer full.
325	At runtime, extended communication failure: send buffer full.
326	When running, extended communication failure: Failed to send message.
327	During operation, a CANOpen slave failure is detected (heartbeat timeout or node guard timeout, SDO no response, etc.).
329	During operation, the following error occurred: Divided by 0.
330	During operation, the following error occurred: type conversion instruction (I_TO_B, DI_TO_I) overflow.
331	During operation, the following error occurred: the input value of the LN instruction is 0 or a negative number.
332	During operation, the following error occurred: the input value of the LOG instruction is 0 or a negative number.
333	During operation, the following error occurred: The input value of the SQRT instruction is a negative number.
334	During operation, the following error occurred: Invalid input value for I_TO_BCD instruction.
335	During operation, the following error occurred: Invalid input value for A_TO_H instruction.
336	During operation, the following error occurred: Invalid input value for R_TO_A instruction.
341	At runtime, the following error occurred: Invalid input value for instruction.
350	While running, the following error occurred: Failed to save persistent storage data.
351	At power-up, loss of retentive data in RAM is detected.
360	Low backup battery alarm.

13.3 How to read the errors that have occurred in the PLC

When an error occurs in the PLC, it will automatically record the error codes one by one, and the user can use the following methods to read general error and serious error information. For how to deal with fatal errors, please refer to 13.1.1.2 How to resolve Fatal Errors.



1) Use KincoBuilder

Users can execute [PLC] -> [PLC Serious Error...] or [PLC Common Error...] menu command in KincoBuilder, and view it in the dialog window, as shown in the right figure. In the dialog box, the user can click the [Refresh] button to refresh and display the latest error record information in the PLC

2) Use Modbus RTU communication

The user can use the Modbus RTU communication command (function code 3, 4) to read the error record information through the Port0, Port1 or Port2 communication port. What the user reads is the error code recorded by the PLC in the form of words, and one or more records can be read at a time.

The Modbus register information of the error record is as follows:

Modbus register	Description
9000-9127	After the PLC is powered on this time, the 128 general error codes that occurred recently. Among them, 9000 is the latest error, 9001 is the next latest error, and so on.
9128-9255	After the PLC is powered on this time, the 128 serious error codes that occurred recently. Among them, 9128 is the latest error, 9129 is the next latest error, and so on.
9256-9383	The last 128 general error codes that occurred during the last power-on process of the PLC. Among them, 9256 is the latest error, 9257 is the next latest error, and so on.
9384-9511	The last 128 serious error codes that occurred during the last power-on process of the PLC. Among them, 9384 is the latest error, 9385 is the next latest error, and so on.

13.4 Error register

For error handling, K series PLC provides a control byte and various error status registers in the SM area. When an error occurs in the PLC, the value of the corresponding error register will be automatically set. User programs can directly read these register values and perform corresponding processing.

- SMB2: Control Bytes

bit (readable and writable)	Functional description
SM2.0	Its value determines the action of the PLC when a fatal error occurs. The initial value is 0 after power-on. If the value is 0: When a fatal error occurs, the PLC enters the independent safety subsystem to run. If the value is 1: When a fatal error occurs, the PLC will reset directly.

- SMB0 and SMB1: Memory, instruction error

SM	Function description
SMB0 (Read only)	
SM0.2	If the PLC detects that the power-off retention data in the RAM is lost after power-on, set this bit to 1, otherwise set it to 0.
SMB1 (Read only)	
SM1.0	1 means that the following errors have occurred: DIV, MOD instructions are divided by 0.

SM1.1	1 means that the following errors have occurred: the parameters of LN, LOG, and SQRT instructions are illegal values (0 or negative numbers).
SM1.2	1 means that the following errors have occurred: I_TO_B, DI_TO_I instruction conversion result overflow.
SM1.3	1 means that the following error has occurred: I_TO_BCD command invalid input BCD code.
SM1.4	1 means that the following error has occurred: there are unrecognizable characters in the input string of the A_TO_H instruction.
SM1.5	1 means that the following error has occurred: R_TO_A instruction conversion result overflow.
SM1.6	1 indicates that the following error has occurred: The value of the input parameter of the FOR instruction is invalid.

- SMB3, SMB5 and SMB96-SMB110: Expansion module failure (CPU that does not support expansion is not supported)

If the PLC detects an expansion bus communication failure or receives a failure message sent by the expansion module, the PLC will set the corresponding expansion bus failure flag, and store the failure code in the corresponding expansion bus failure register for query. If no fault is detected, the fault flag and fault register are set to 0.

Note: If the PLC detects that the expansion bus or the expansion module is faulty during startup, it will enter the STOP state and light the ERR indicator at the same time, but the corresponding register will not be set. Because the CPU does not execute the user program at this time, the register value cannot be read.

SM	**Function description**
SMB3, SMB5 (Read only): expansion bus error flag	
SMB3.0	This bit is set to 1 if the first expansion module fails.
SMB3.1	This bit is set to 1 if the 2nd expansion module fails.
SMB3.2	This bit is set to 1 if the 3rd expansion module fails.
SMB3.3	This bit is set to 1 if the 4th expansion module fails.
SMB3.4	This bit is set to 1 if the 5th expansion module fails.
SMB3.5	This bit is set to 1 if the 6th expansion module fails.
SMB3.6	This bit is set to 1 if the 7th expansion module fails.
SMB3.7	This bit is set to 1 if the 8th expansion module fails.
SMB5.0	This bit is set to 1 if the 9th expansion module fails.
SMB5.1	This bit is set to 1 if the 10th expansion module fails.
SMB5.2	This bit is set to 1 if the 11th expansion module fails.

SMB5.3	This bit is set to 1 if the 12th expansion module fails.
SMB5.7	This bit is set to 1 if the CPU detects an expansion bus communication failure.
SMB96 - SMB110 (read only): Expansion bus error codes	
SMB96	If the first expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB97	If the second expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB98	If the third expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB99	If the fourth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB100	If the fifth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB101	If the sixth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB102	If the seventh expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB103	If the eighth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB104	If the ninth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB105	If the tenth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB106	If the eleventh expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB107	If the twelfth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB110	If the expansion bus at the CPU end fails, store its fault code. The meaning is shown in Table 2.

Error Code	Description
0	No glitches.
6	During operation, the heartbeat packet of the expansion module times out. The expansion module will regularly send heartbeat messages to the CPU module. If the CPU does not receive the heartbeat message from the module within the timeout, it means that the communication of the module may be abnormal.
10	An error occurred in the ADC conversion of the AI channel (voltage, current input).
11	The module saved the calibration value incorrectly.
12	The module reads the calibration value incorrectly.
14	The input signal of the first channel of the analog input module exceeds the set measurement range.
15	The input signal of the second channel of the analog input module exceeds the set measurement range.
16	The input signal of the third channel of the analog input module exceeds the set measurement range.

17	The input signal of the fourth channel of the analog input module exceeds the set measurement range.

Table 1 Meaning of error codes in the expansion module

Error codes	Descriptions
0	No glitches.
1	Communication frame format error.
2	The expansion bus enters the error alarm state.
3	The expansion bus goes into error passive state.
4	The expansion bus entered the bus off state and just recovered automatically.
5	The expansion bus receive buffer is full.
6	The expansion bus send buffer is full.
7	The CPU failed to send the packet.

Table 2 Meaning of CPU module expansion bus communication error codes

13.5 How to restore the factory to the original state?

The user can clear all data in the PLC and restore the PLC to the factory default state. There are several ways to clear it:

1) For all series of PLCs, the user can clear all the data in the CPU module by executing the [PLC] → [Clear...] menu command in the KincoBuilder programming software. This includes user programs, configuration data, passwords, etc., thereby restoring the CPU to its factory default state.

2) For PLCs with a "CLR" clear button, such as MK series all-in-ones, users can directly use the CLR hardware button to clear. For the specific operation method, please refer to 13.1.1.2.2 For PLC with "CLR" clear button.

After clearing, the internal state of the CPU is as follows:

- Previous user programs, all memory areas (including data retention and data backup areas), passwords, communication port parameters set by users using COM_WPARAS and LORA_WPARAS instructions, data in special storage areas written by users using UNID_W instructions, etc., are all cleared.
- According to the model of the machine, the CPU automatically generates an "empty project" inside. Among it, the parameters of all serial communication ports are configured as follows: PLC station

number is 1, baud rate is 9600, no checksum, 8 data bits, and 1 stop bit.

- In principle, the parameters of the Ethernet communication port set by the user remain unchanged. After the clearing process is complete, the CPU will check the previously stored Ethernet communication parameters. If it is legal data, it will remain unchanged; if the data is illegal, the CPU will automatically configure the parameters to factory default values: IP address is 192.168.0.252, subnet mask is 255.255.255.0, gateway address is 192.168.0.1, port for 502.
- The real time clock value (date and time) remains unchanged.

Chapter 14 Intellectual Property Protection

14.1 password protection

Kinco-K series PLC provides a password protection function, which allows users to restrict the use of specific functions by encrypting the CPU.

KPLC password length allows 8-14 characters, can be composed of numbers, letters, underscores, letters are case-sensitive.

If the user sets the PLC protection level to "Level 3: Highest Protection" and sets a password or enables the "Prohibit Upload" function. The user programs stored in the PLC will all be encrypted and stored in the form of ciphertext to prevent it from being cracked by directly reading the chip data.

If the CPU is encrypted, the user will be required to enter a password before using the restricted functions. At this time, if the password entered by the user is correct, the CPU allows the operation; if the password entered by the user is incorrect, the CPU will prohibit the operation. The entered password is only valid for the current operation, and it is still required to enter the password when using the restricted function in the future. The number of incorrect password input is limited. When the incorrect password input reaches the limited number of times, the PLC must be restarted to continue to perform operations with passwords to avoid brute force cracking.

14.1.1 protection level

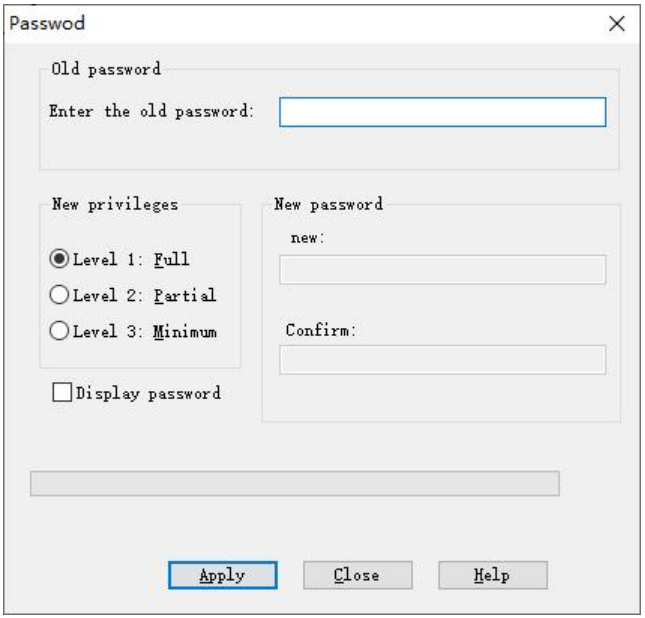
The password protection of the CPU provides the following 3 levels:

- **Level 1**: No protection. There is no limit to the usage of CPU functions. This is the default level.
- **Level 2**: Partial protection. Users are required to provide a password when performing the download function.
- **Level 3**: Highest protection. Users need to provide passwords when performing upload and download functions. After the password is set, the user program stored in the PLC will be encrypted in full bytes

and stored in the form of ciphertext to prevent it from being cracked by directly reading the hardware.

14.1.2 Change password and protection level

Execute the [PLC]→[Change Password...] menu command to enter the "CPU Password Protection" window for modification. As shown below:



1） Old password verification

If the connected CPU has already been set up for password protection, you must correctly enter the original old password here for verification.

If no password protection has been set before, just keep the input box empty.

2） New protection level and new password: Here you can set a new protection level and password for the connected CPU.

- **Protection level:** You can choose one of level 1, level 2 and level 3
- **New Password:** Enter a new password here.

The password allows 8-14 characters, which can be composed of numbers, letters, and underscores. Letters are case-sensitive. This input box takes effect when the protection level is selected as level 2 or level 3.

- **New password confirmation:** Here the user needs to enter the new password again.

After the user completes all the above settings, click the [Apply] button, the new settings will be written into the connected CPU, and KincoBuilder will pop up a dialog box to prompt the modification result.

14.1.3 What to do if you forget your password

If the user forgets the password in the CPU, all restricted functions will not be available.

At this point, the user can execute the [PLC] → [Clear...] menu command to clear all the contents in the CPU memory, and then continue to use the PLC. After executing the [Clear...] command, the user program, configuration data, password, etc. will all be cleared, and the CPU will return to the factory default state.

Appendix A Definition of common system memory SM area

This appendix describes in detail the relevant definitions in the commonly used system storage area SM area. The system storage area is used to assist the Kinco-K series PLC to realize specific command functions, and the user can also use the system storage to read certain states of the PLC. It should be noted that not all CPUs support all SMs below. For example, some CPU bodies do not have their own analog quantities, so naturally they do not support SM2.1 functions.

1. SMB0: system status byte

SM0.0-SM0.7is assigned by the running software of the CPU and is not controlled by the user program. In the user program, these bits can only be called (read-only). For detailed function description, please refer to the table below.。

Bit(read-only)	Description
SM0.0	always "1"
SM0.1	first scan bit. It is "1" when the CPU scans for the first time, and then cleared to "0". Usually used for user program initialization.
SM0.2	If the power-off retention data in RAM is lost, the PLC will set this bit to 1 in the first scan, and then clear it to 0.
SM0.3	A continuous pulse train with a period of 1s and a duty cycle of 50%.
SM0.4	A continuous pulse train with a period of 2s and a duty cycle of 50%.
SM0.5	A continuous pulse train with a period of 4s and a duty cycle of 50%.
SM0.6	A continuous pulse train with a period of 60s and a duty cycle of 50%.
SM0.7	SMO.7 is TRUE, the battery voltage is low; SMO.7 is FALSE, the battery voltage is normal.

2. error register

For error handling, KPLC provides a control byte and various error status registers in the SM area. When an error occurs in the PLC, the value of the corresponding error register will be automatically set. User

programs can directly read these register values and perform corresponding processing.

- **SMB2: Control byte**

Bit (R/W)	Function description
SM2.0	Its value determines the action of the PLC when a fatal error occurs. The initial value is 0 after power-on. If the value is 0: When a fatal error occurs, the PLC enters the independent safety subsystem to run. If the value is 1: When a fatal error occurs, the PLC will reset directly.
SM2.1	Its value determines the working status of AI and AO channels in the system. The initial value is 0 after power-on. If the value is 0: the AI and AO channels of the main body work normally. If the value is 1: the AI and AO channels of the body enter the calibration state.

- **SMB1 (read only): Instruction error**

SM	function description
SM1.0	1 means that the following errors have occurred: DIV, MOD instructions are divided by 0.
SM1.1	1 means that the following errors have occurred: the parameters of LN, LOG, and SQRT instructions are illegal values (0 or negative numbers).
SM1.2	1 means that the following errors have occurred: I_TO_B, DI_TO_I instruction conversion result overflow.
SM1.3	1 means that the following error has occurred: I_TO_BCD command invalid input BCD code.
SM1.4	1 means that the following error has occurred: there are unrecognizable characters in the input string of the A_TO_H instruction.
SM1.5	1 means that the following error has occurred: R_TO_A instruction conversion result overflow.
SM1.6	1 indicates that the following error has occurred: The value of the input parameter of the FOR instruction is invalid.

- **SMB3, SMB and SMB96-SMB110: Expansion module failure (only for CPUs that support expansion)**

If the PLC detects an expansion bus communication failure or receives a failure message sent by the expansion module, the PLC will set the corresponding expansion bus failure flag and store the failure code in the corresponding expansion bus failure register for query. If no fault is detected, the fault flag and fault register are set to 0.

Note: If the PLC detects the failure of the expansion bus or expansion module during startup, it will enter the STOP state and light the ERR light, but the corresponding register will not be set. This is because the CPU does not execute the user program at this time, and the user cannot read the register value.

SM	Function description
SMB3, SMB5 (read only): Expansion bus error flag	
SMB3.0	This bit is set to 1 if the first expansion module fails.
SMB3.1	This bit is set to 1 if the 2nd expansion module fails.
SMB3.2	This bit is set to 1 if the 3rd expansion module fails.
SMB3.3	This bit is set to 1 if the 4th expansion module fails.
SMB3.4	This bit is set to 1 if the 5th expansion module fails.
SMB3.5	This bit is set to 1 if the 6th expansion module fails.
SMB3.6	This bit is set to 1 if the 7th expansion module fails.
SMB3.7	This bit is set to 1 if the 8th expansion module fails.
SMB5.0	This bit is set to 1 if the 9th expansion module fails.
SMB5.1	This bit is set to 1 if the 10th expansion module fails.
SMB5.2	This bit is set to 1 if the 11th expansion module fails.
SMB5.3	This bit is set to 1 if the 12th expansion module fails.
SMB5.7	This bit is set to 1 if the CPU detects an expansion bus communication failure.
SMB96 - SMB110 (read only): Expansion bus error codes	
SMB96	If the first expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB97	If the second expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB98	If the third expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB99	If the fourth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB100	If the fifth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB101	If the sixth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB102	If the seventh expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB103	If the eighth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB104	If the ninth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB105	If the tenth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB106	If the eleventh expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB107	If the twelfth expansion module fails, store its fault code. The meaning is shown in Table 1.
SMB110	If the expansion bus at the CPU end fails, store its fault code. The meaning is shown in Table 2.

3. SMD12 and SMD16: Period of timer interrupt

KPLC provides two timing interrupts with a time base of 0.1ms: timing interrupt 0, interrupt number 3; timing interrupt 1, interrupt number 4.。

SMD12 is used to specify the period value of timer interrupt 0, the unit is 0.1ms. If SMD12 is set to 0,

timer interrupt 0 is prohibited. The default value of SMD12 is 0.

SMD16 is used to specify the period value of timer interrupt 1, the unit is 0.1ms. If SMD16 is set to 0, timer interrupt 1 is disabled. The default value of SMD16 is 0.

Timing interrupts will be generated periodically, which can be used to complete periodic tasks. Timing interrupts are not affected by the PLC scan cycle and can be used for precise timing.

4. Control and Status Registers for High-Speed Counters

- **Control Register**

In the SM area, the following control registers are provided for each high-speed counter to store its configuration data: a control byte (8 bits), a current value and a preset value (both are 32-bit signed double integers). The current value specifies the starting value of counting. If the current value is written into the high-speed counter, the high-speed counter will immediately start counting from this value. The table below describes these registers in detail.

HSC0	HSC1	HSC2	HSC3	Description
SM37.0	SM47.0	SM57.0	SM127.0	Active level of reset signal: 0=high level; 1=low level
SM37.1	SM47.1	SM57.1	SM127.1	Active level of start signal: 0=high level; 1=low level
SM37.2	SM47.2	SM57.2	SM127.2	Quadrature counter rate: 0=1x rate; 1=4x rate*
SM37.3	SM47.3	SM57.3	SM127.3	Counting direction: 0=down counting; 1=up counting.
SM37.4	SM47.4	SM57.4	SM127.4	Whether to write counting direction to HSC: 0=no; 1=yes.
SM37.5	SM47.5	SM57.5	SM127.5	Whether to write new preset value into HSC: 0=no; 1=yes.
SM37.6	SM47.6	SM57.6	SM127.6	Whether to write new current value to HSC: 0=no; 1=yes.
SM37.7	SM47.7	SM57.7	SM127.7	Whether to allow the high-speed counter: 0=prohibited; 1=allowed.
HSC0	**HSC1**	**HSC2**	**HSC3**	**Description**
SMD38	SMD48	SMD58	SMD128	The current value
SMD42	SMD52	SMD62	SMD132	preset value

In addition, all high-speed counters except KPLC allow to specify a maximum of 32 preset values (PV), and their control registers are described as follows:

HSC0	HSC1	HSC2	HSC3	Description
SM141.0	SM151.0	SM161.0	SM171.0	Whether to use multi-segment preset values: 0=no; 1=yes
SM141.1	SM151.1	SM161.1	SM171.1	Whether the preset value is relative or absolute: 0 = absolute; 1 = relative
SM141.2	SM151.2	SM161.2	SM171.2	Whether the preset value comparison ("CV=PV") interrupt is generated cyclically:
SM141.3	SM151.3	SM161.3	SM171.3	0=no; 1=yes.
SM141.4	SM151.4	SM161.4	SM171.4	Note: Only the relative value mode is allowed to be set as cycle generation.
SM141.5	SM151.5	SM161.5	SM171.5	reserve
SM141.6	SM151.6	SM161.6	SM171.6	Whether to update the number of segments and preset values: 0=no; 1=yes
SM141.7	SM151.7	SM161.7	SM171.7	Whether to reset the interrupt variable: 0=yes; 1=no
HSC0	**HSC1**	**HSC2**	**HSC2**	**Description**
SMW142	SMW152	SMW162	SMW172	The starting position of the preset value table (expressed by the byte offset relative to VB0), must be an odd number.

- **Status Register**

Each high-speed counter provides a status register in the SM area to indicate the current status information of the high-speed counter.

HSC0	HSC1	HSC2	HSC3	Description
SM36.0	SM46.0	SM56.0	SM126.0	reserve
SM36.1	SM46.1	SM56.1	SM126.1	reserve
SM36.2	SM46.2	SM56.2	SM126.2	reserve
SM36.3	SM46.3	SM56.3	SM126.3	Whether there is an error in the setting of the multi-segment PV value table: 0=no, 1=yes.
SM36.4	SM46.4	SM56.4	SM126.4	reserve
SM36.5	SM46.5	SM56.5	SM126.5	Current counting direction: 0=decrease; 1=increase.
SM36.6	SM46.6	SM56.6	SM126.6	Whether the current count value is equal to the preset value: 0=no; 1=yes.
SM36.7	SM46.7	SM56.7	SM126.7	Whether the current count value is greater than the preset value: 0=no; 1=yes.
HSC0	HSC1	HSC2	HSC3	Description (K5 series does not support)
SMB140	SMB150	SMB160	SMB170	The serial number of the running PV segment (starting from 0).

For details about the use of the high-speed counter function, please refer to Chapter 8 Use of High-speed Counter Function.

5. Control register and status register for high-speed pulse output

5.1 Use of PLS instruction

For details about the use of the high-speed pulse output PTO/PWM function, please refer to Chapter 9.3 Use of the PLS instruction.

- **PTO/PWM control register**

In the SM area, some control registers are provided for each PTO/PWM generator to store its configuration data. See the table below.

Q0.0	Q0.1	Q0.4	Q0.5	Description	
SM67.0	SM77.0	SM97.0	SM107.0	PTO/PWM	Whether to update the period value: 0=no; 1=yes
SM67.1	SM77.1	SM97.1	SM107.1	PWM	Whether to update the pulse width value: 0=no; 1=yes
SM67.2	SM77.2	SM97.2	SM107.2	PTO	Whether to update the number of pulses: 0=no; 1=yes
SM67.3	SM77.3	SM97.3	SM107.3	PTO/PWM	Time base: 0=1μs; 1=1ms
SM67.4	SM77.4	SM97.4	SM107.4	PWM	Update method: 0=asynchronous update; 1=synchronous update
SM67.5	SM77.5	SM97.5	SM107.5	PTO	Operation mode: 0=single-segment operation; 1=multi-segment operation
SM67.6	SM77.6	SM97.6	SM107.6	Function selection: 0=PTO; 1=PWM	
SM67.7	SM77.7	SM97.7	SM107.7	PTO/PWM	Enable or disable this feature: 0=disable; 1=allow
Q0.0	**Q0.1**	**Q0.4**	**Q0.5**	**Description**	
SMW68	SMW78	SMW98	SMW108	PTO/PWM	Period value, range 2~65535
SMW70	SMW80	SMW100	SMW110	PWM	Pulse width value, range 0~65535
SMD72	SMD82	SMD102	SMD112	PTO	Number of pulses, range 1~4,294,967,295
SMW168	SMW178	SMW218	SMW248	The starting position of the envelope table (indicated by byte offset relative to VB0), only used for PTO multi-segment operation.	

- **PTO/PWM status register**

A status byte is also provided for each PTO/PWM generator in the SM area, and the user can know the current status information of the PTO/PWM generator by accessing the status byte. See the table below.

Q0.0	Q0.1	Q0.4	Q0.5	Description

SM66.0	SM76.0	SM96.0	SM106.0	reserve
SM66.1	SM76.1	SM96.1	SM106.1	reserve
SM66.2	SM76.2	SM96.2	SM106.2	reserve
SM66.3	SM76.3	SM96.3	SM106.3	Is PWM idle: 0=no; 1=yes
SM66.4	SM76.4	SM96.4	SM106.4	Whether there is an error in the setting of the PTO period value and the number of pulses: 0=no; 1=yes **Note: The period value and the number of pulses must be greater than 1.**
SM66.5	SM76.5	SM96.5	SM106.5	Whether PTO terminated due to user command: 0=No; 1=Yes
SM66.6	SM76.6	SM96.6	SM106.6	reserve
SM66.7	SM76.7	SM96.7	SM106.7	Is PTO idle: 0=busy; 1=idle

5.2 Using position control instructions

Please refer to 9.4 Use of position control Instructions for details about the function usage of positioning instructions.

For position control commands, KPLC allocates a control byte for each high-speed output in the SM area, and the user needs to pay attention to setting the control byte in the application. In addition, a current value (DINT type) register is also assigned to store the number of pulses that have been output currently (increase when forward rotation, decrease when reverse rotation). The table below describes these registers in detail.

Q0.0	**Q0.1**	**Q0.4**	**Q0.5**	**Description**
SMD212	SMD242	SMD262	SMD226	read only. The current value indicates the number of pulses that have been output currently. Increase during forward rotation and decrease during reverse rotation.
SMD208	SMD238	SDM258	SMD222	read and write. new current value. Cooperate with the corresponding flag bit to modify the current value.
Q0.0	**Q0.1**	**Q0.4**	**Q0.5**	**Description**
SM201.7	SM231.7	SM251.7	SM221.7	read and write. Emergency stop sign. If this bit is 1, it means that it is in emergency stop state and does not execute any position control command. This bit will be automatically set to 1 when the PSTOP (emergency stop) instruction is executed. User needs to clear this bit to 0 by program.
SM201.6	SM231.6	SM251.6	SM221.6	read and write. Used to decide whether to reset the current value 1 - Clear the current value to zero. 0 - The current value remains unchanged.

SM201.5	SM231.5	SM251.5	SM221.5	Reserve
SM201.4	SM231.4	SM251.4	SM221.4	read and write. Used to decide whether to modify the current value 1 - Modify the current value to the value in the "New Current Value" register above. 0 - The current value remains unchanged.
SM201.3	SM231.3	SM251.3	SM221.3	Direction enable control bit. 1 Disable the direction output, and the direction channel is used as ordinary D0. 0 - Enable direction output.
SM201.0 ~ SM201.2	SM231.0 ~ SM231.2	SM251.0 ~ SM251.2	SM221.0 ~ SM221.2	Reserve

6. Control registers and status registers related to free communication instructions XMT/RCV

Please refer to Chapter 10.2.1.2 Free Communication Instructions for details about the function usage of the free communication XMT/RCV instructions.

KPLC provides multiple status registers and control registers for free communication in the SM area. When using XMT/RCV instructions to write communication programs, users must set these control registers. In addition, the CPU will automatically detect the communication status during the communication process, and write the detection results into the relevant status registers, and the user can read these status information and perform corresponding processing in the program.

- Receive status byte

Bit (read only)			Value	Meaning
PORT 0	PORT 1	PORT 2		
SM86.0	SM186.0	SM286.0	1	Received characters have parity errors, but reception does not stop.
SM86.1	SM186.1	SM286.1	1	Terminating receive: Reached the maximum number of bytes received.
SM86.2	SM186.2	SM286.2	1	Terminate receiving: Receive a character timeout.
SM86.3	SM186.3	SM286.3	1	Terminate receiving: the system receiving timeout.
SM86.4	SM186.4	SM286.4	-	reserve.
SM86.5	SM186.5	SM286.5	1	End Receive: A user-defined end character was received.
SM86.6	SM186.6	SM286.6	1	Termination of reception: parameter error, reception without

				start condition or no reception end condition, etc.
SM86.7	SM186.7	SM286.7	1	Terminate reception: The user has used the prohibit reception command.

- Receive control byte

Bit			Value	Description
PORT 0	PORT 1	PORT 2		
SM87.0	SM187.0	SM287.0	-	reserve.
SM87.1	SM187.1	SM287.1	0	Disable generation of corresponding interrupt when transmission or reception is complete.
			1	Allows generation of corresponding interrupts when a send or receive is complete.
SM87.2	SM187.2	SM287.2	0	Ignores the user-defined receive character timeout value in SMW92/SMW192/SMW292.
			1	Use the user-defined receive character timeout value in SMW92/SMW192/SMW292.
SM87.3	SM187.3	SM287.3	-	reserve.
SM87.4	SM187.4	SM287.4	0	Ignore the user-defined reception preparation time in SMW90/SMW190/SMW290.
			1	Use the user-defined reception preparation time in SMW90/SMW190/SMW290.
SM87.5	SM187.5	SM287.5	0	Ignore the user-defined receive end character in SMB89/SMW189/SMW289.
			1	Use the user-defined receive end character in SMB89/SMW189/SMW289.
SM87.6	SM187.6	SM287.6	0	Ignore the user-defined receive start character in SMB88/SMW188/SMW288.
			1	Use the user-defined receive start character in SMB88/SMW188/SMW288.
SM87.7	SM187.7	SM287.7	0	Receiving data is prohibited. This prohibition condition takes precedence over all other receiving control words.
			1	Allow data to be received.

- Other control registers

control word (byte)			Description
PORT0	PORT1	PORT2	
SMB88	SMB188	SMB288	It is used to store user-defined receiving start character. After executing the RCV instruction, the CPU starts to enter the effective receiving state after receiving the start character, and the data received before will be discarded. The CPU regards the start character as the first

			valid data received. If you want this setting value to take effect, you need to set SM87.6/SM187.6/SM287.6 to 1.
SMB89	SMB189	SMB289	It is used to store user-defined receiving end characters. The CPU will take this character as the last byte received. After receiving this character, the CPU will immediately terminate the receiving state regardless of other end conditions. To make this setting effective, you need to set SM87.5/SM187.5/SM287.5 to 1.
SMW90	SMW190	SMW290	It is used to store the receiving preparation time defined by the user (the range is 1~60000ms). After executing the RCV instruction, after the time value, the CPU will automatically enter the valid receiving state regardless of whether the start character is received, and the data received thereafter will be considered as valid data. If you want this setting value to take effect, you need to set SM87.4/SM187.4/SM287.4 to 1.
SMW92	SMW192	SMW292	It is used to store the timeout value of receiving characters defined by the user (the range is 1~60000ms). After executing the RCV instruction and entering the effective receiving state, if no character is received within the timeout period, the CPU will end the receiving state, regardless of other end conditions. To make this setting effective, you need to set SM87.2/SM187.2/SM287.2 to 1.
SMW94	SMW194	SMW294	It is used to store the user-defined number of received characters each time (1~255). As long as the CPU has received this number of valid characters, regardless of other end conditions, it will immediately terminate the receiving state. This setting value is always valid. If the value is set to 0, the RCV instruction will exit directly.

In the serial port free communication, there is also a default system receiving timeout setting, the time is 90 seconds. The function of this timeout value is as follows: after executing the RCV instruction, if the serial port does not receive any data within the timeout period, the CPU will immediately terminate receiving and exit the RCV instruction. In addition, after the CPU enters the effective receiving state (that is, after receiving the start character defined in SMB88 or after the receiving preparation time defined in SMW90), the timeout value for receiving characters defined by the user in SMW92 will be used preferentially. If the user does not define it, use the system to receive the timeout value to decide whether to terminate the reception.

7. Dynamically modify the relevant registers of RS485 communication port parameters

For details about the function of dynamically modifying the parameters of the RS485 communication port, please refer to 10.2.3 Dynamically Modifying the Parameters of the RS485 Communication Port.

Except for K209M, KS101M, KW203, MK, and K6, other series of PLCs support the use of SMB20--SMB25 to dynamically modify communication parameters. The meaning of each register is described in detail below.

- **Parameter value: SMB23、SMB24 and SMB25**

SMB	Description
SMB23	PLC station number value. Valid range of values: 1-31. If a write operation is performed, SMB23 is the new PLC station number value to be written; if a read operation is performed, SMB23 is the currently used PLC station number value read; if a clear operation is performed, SMB23 is ignored.
SMB24	Baud rate value. The valid range of values is 0-5: 0 means 2400, 1 means 4800, 2 means 9600, 3 means 19200, 4 means 38400, 5 means 1200. If a write operation is performed, SMB24 is the new baud rate value to be written; if a read operation is performed, SMB24 is the currently used baud rate value read; if a clear operation is performed, SMB24 is ignored.
SMB25	parity value. The valid range of values is 0-2, 0 means no test, 1 means odd test, 2 means even test. If a write operation is performed, SMB25 is the new parity value to be written; if a read operation is performed, SMB25 is the currently used parity value read; if a clear operation is performed, SMB25 is ignored.

- **Control byte: SMB20 and SMB21**

Bit	Description
SMB20: Specify the port number and operation mode to be operated	
SM20.7	A value of 1 indicates that a write operation is enabled. After the PLC writes new parameters, it will automatically clear this bit to 0.
SM20.6	A value of 1 indicates that a read operation is enabled. After the PLC reads the parameters, it will automatically clear this bit to 0.
SM20.5	A value of 1 initiates a purge operation. After the PLC clears the parameters, it will automatically clear this bit to 0.
SM20.4	Reserved, must be assigned a value of 0.
SM20.3 SM20.0	The combined value of these 4 bits indicates the port number to be operated. 1 means PORT1, 2 means PORT2, if it is set to other invalid values, the PLC will report an error and exit the operation.

SMB21: Specifies the communication parameters to be manipulated.	
SM21.7 -- SM21.3	spare. Must be assigned a value of 0.
SM21.2	A value of 1 means to modify or clear the parity value of the specified communication port.
SM21.1	A value of 1 means to modify or clear the baud rate value of the specified communication port.
SM21.0	A value of 1 means to modify or clear the PLC station number of the specified communication port.

At the same time, SM20.5, SM20.6, and SM20.7 only allow one bit to be 1, otherwise the PLC will report an error and exit the operation.

When performing a read operation, the value of SMB21 is ignored, and the PLC will read all communication parameter values at one time.

When performing the clearing operation, the PLC allows clearing the station number, baud rate, and parity value separately or simultaneously. After a parameter is cleared, the PLC will automatically adopt the corresponding communication parameter value in the hardware configuration information.

- **Status byte: SMB22**

SMB22 stores the operation result of this dynamic adjustment of communication parameters.

Bit (read only)	Description
SM22.7	A value of 1 indicates that the operation is complete. After the PLC completes the operation specified by the user, whether it succeeds or fails, it will automatically set SM22.7 to 1. Only when the value of SM22.7 is 1, the values of other bits in SMB22 have practical significance.
SM22.6	If the value of SM22.7 is 1, then if the value of SM22.6 is 1, it means that the operation is successful; if the value of SM22.6 is 0, it means that the operation failed due to an error.
SM20.5 ∽SM20.0	If this operation fails, the combined value of these 6 bits indicates the error code that occurred, and the meaning is as follows.

error code	Error description
1	Wrong operation command. For example, SM20.7 and SM20.6 are set to 1 at the same time.
2	wrong port number
3	The value of SMB21 (the communication parameter to be operated) is wrong.
4	The value of SMB23 (new PLC station number) is wrong.
5	SMB24 (new baud rate) value is wrong.
6	SMB25 (new parity) value error.

10	Failed to read the stored PORT1 dynamic PLC station number from the permanent memory.
11	The dynamic PLC station number of PORT1 has not been set.
12	Failed to read stored PORT1 dynamic baud rate value from persistent memory.
13	The dynamic baud rate of PORT1 has not been set.
14	Failed to read stored PORT1 dynamic parity value from persistent storage.
15	The dynamic parity value of PORT1 has not been set.
20	Failed to read the stored PORT2 dynamic PLC station number from the permanent memory.
21	The dynamic PLC station number of PORT2 has not been set.
22	Failed to read stored PORT2 dynamic baud rate value from persistent memory.
23	The dynamic baud rate of PORT2 has not been set.
24	Failed to read stored PORT2 dynamic parity value from persistent storage.
25	The dynamic parity value of PORT2 has not been set.
61	Failed to write dynamic communication parameter value to permanent memory.

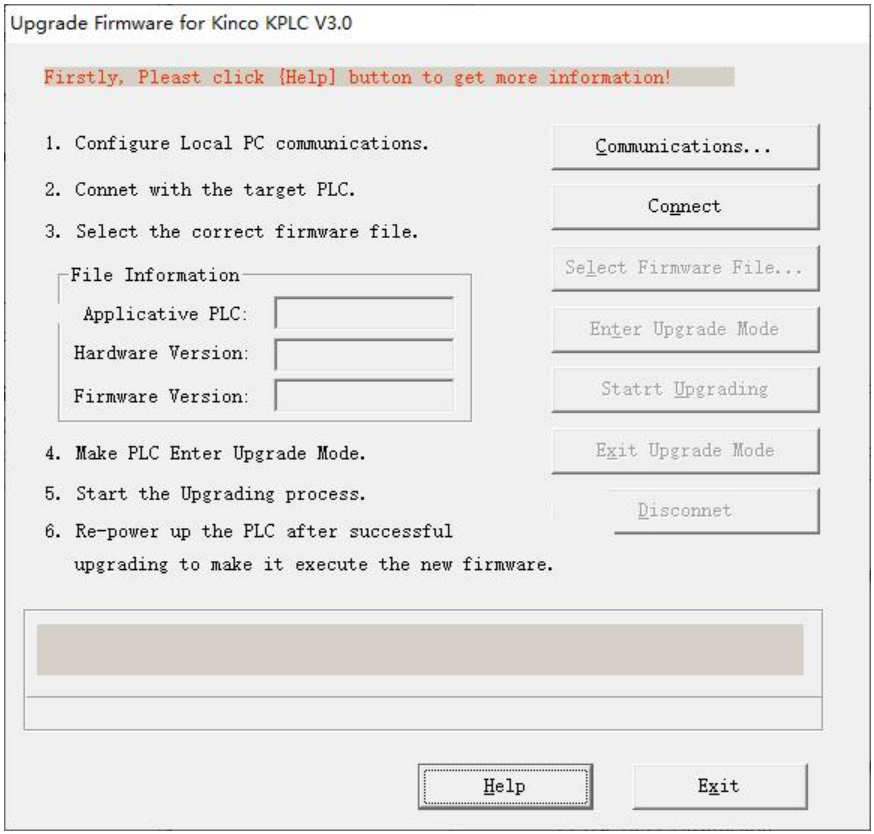
8. Other commonly used function variables

<table>
<tr><th>SM</th><th colspan="2">description</th></tr>
<tr><td>SMB6</td><td colspan="2">read only. Store the latest PLC scan time, unit: ms.</td></tr>
<tr><td>SMW10</td><td colspan="2">read only. Store the voltage value of the backup battery, unit: 0.01V. (only for K5)
If the power supply of the backup battery continues to be lower than 2.6V, the PLC will generate a "low battery" alarm.</td></tr>
<tr><td>SMW122</td><td>CAN2 port bus load rate minimum value.</td><td rowspan="2">Load factor = SMW value ÷ 1000</td></tr>
<tr><td>SMW124</td><td>CAN2 port bus load rate maximum.</td></tr>
<tr><td>SMB274-
SMB285</td><td colspan="2">The combined value of these 12 bytes indicates the ID value of this CPU module.
Each CPU module is given a unique ID value when leaving the factory, and the ID values of each module will not be the same.</td></tr>
</table>

Appendix B Updating Programs of the System

KPLC provides bootloader, allowing users to update system programs by themselves. Kinco continues to upgrade and improve KPLC, and releases the latest PLC system program. If the user needs the latest version of the PLC system program, he can contact Kinco to obtain it.

In KincoBuilder software, users can enter the small tool software for updating system programs by executing the [Tools] -> [Update System Program] menu command. The steps to update the PLC system program are briefly indicated in the software interface, and users can also click the [Help] button to view more detailed instructions.



PLC has two modes of normal work and system update. The user must first let the PLC enter the system update mode, and then can update its system program.

Because the type and number of communication ports improved by each series of PLCs are different, the communication ports that can be used by different models of products are different. In addition, due to historical reasons, some models need to enter the STOP state to be updated. These information are detailed in the help of updating the software, and are summarized here as follows:

Type	PORT0 (RS232)	PORT1 (RS485)	PORT2 (RS485)	USB	net	Enter STOP state or not
K5	Support	/	/	/	/	Yes
K2	/	/	/	Support	/	Yes
KS	Support	Support	/	/	/	Yes
K209M KS101M	Support	Support	/	Support	/	No
KW	Support	Support	/	Support	/	No
MK	/	Support	/	Support	/	No
K6	/	Support	/	/	Support	No

1、Steps and Precautions for Updating PLC System Program

Users can also refer to the relevant instructions in the [Help] of updating the system program software.

1)Use a communication cable to connect the PLC to the computer, and then power on the PLC.

Note: Except for MK series, other products with USB interface support direct power supply via USB data cable, and no external power supply is required for updating.

2)Click the [Set PC Communication Parameters] button to enter the configuration window, the user can select the computer communication port and configure the communication parameters.

Note: If you use the USB port to update, because it is a virtual serial port, there is no need to modify the communication parameters; if you use the serial port to update, please note that the baud rate is 115200, no parity, 8 data bits, and 1 stop bit. First change the communication parameters of the corresponding PORT in the user project [Hardware Configuration] to the above values, and then download the new project to the PLC, and the new parameters will take effect after the PLC runs successfully.

3） (Optional) First put the PLC in "STOP" state. **Note: K6, MK, KM, KW series do not need to enter the STOP state first.**

4） Click the [Connect Target Module] button. If the user uses the serial port or USB port to update, the software will open the corresponding communication port; if the user uses the Ethernet port to update, the software will establish a TCP connection with the PLC.。

5） Click the [Select System Program File...] button, select the appropriate file in the pop-up dialog box and open it.

If it is a legal file, the applicable PLC model and hardware version will be prompted in [File Information], and the software version number of the system program in the file will be prompted at the same time.

6)Click the [PLC Enter Update Mode] button, and after a delay of about 2 seconds, the PLC will enter the system update mode.

In the system update mode, the "RUN" indicator of the PLC will flash quickly, and other indicators will go out.

If you use the USB port to update, after entering the update mode, the original USB virtual serial port will be destroyed, and the software will automatically re-establish the virtual communication port. If the automatic establishment fails, the user can unplug the USB cable first, plug it in again, and then click the [PLC enter update mode] button.

If you use the Ethernet port to update, after entering the update mode, the original TCP connection will be destroyed, and the software will automatically re-establish the TCP connection with the PLC. If the automatic connection fails, the user can click the [PLC enters update mode] button again.

7) Click the [Start Update System Program] button to start the PLC system update process.

The update process includes erasing PLC system program data, writing new program data, verifying written data, and so on. In the [Operation Information Prompt] in the lower part of the window, the software will prompt the ongoing operation and progress.

If an error occurs during the update process, the software will stop the update and prompt the reason for the failure. **After the update fails, the PLC will stay in the system update mode permanently (the RUN indicator light is flashing at this time). The user can try to click the [Start to update the system program] button again to restart the update, or restart the PLC after powering off for at least 10**

seconds (note that the previous RUN/STOP state is maintained), and then perform the above process again.

8)When the system prompts that the update is successful, power off and restart the PLC, and the PLC can run the new system program.

Other Notes:

1) The system program file must match the model and hardware version of the PLC to be updated!
2) The extension of the PLC system program file is ".kfw", which has its own specific encoding format. Be careful not to modify this file! If an illegal system program is written into the PLC, the PLC may run abnormally and cannot be repaired by the user!
3) If the PLC system update fails, the PLC will remain in the system update mode no matter whether the user powers off and restarts (the RUN indicator flashes at this time), and the user needs to update the PLC system correctly! Only after the update is successful, the PLC will exit the system update mode.

Appendix C Use of PLC Offline Simulator

One、Introduction

KincoBuilder software provides KPLC off-line simulator, which can simulate real PLC operation and debug user programs in KincoBuilder. KPLC's off-line simulator has powerful functions and provides advanced debugging functions such as breakpoint, single-step execution, single-network execution, pause/continue operation, and serial communication simulator.

Before officially using the PLC, it is necessary to repeatedly debug and modify the project to achieve the desired effect, so debugging and testing are very important tasks. The real debugging requires hardware foundation. When the hardware is not available, the user can use the offline simulation function of the PLC programming software KincoBuilder to carry out preliminary debugging and simulate the running effect of the PLC. This description describes some specific usage methods.

Two、run the offline simulator

2.1 startup steps

1、Open the KincoBuilder software and open the project to be debugged

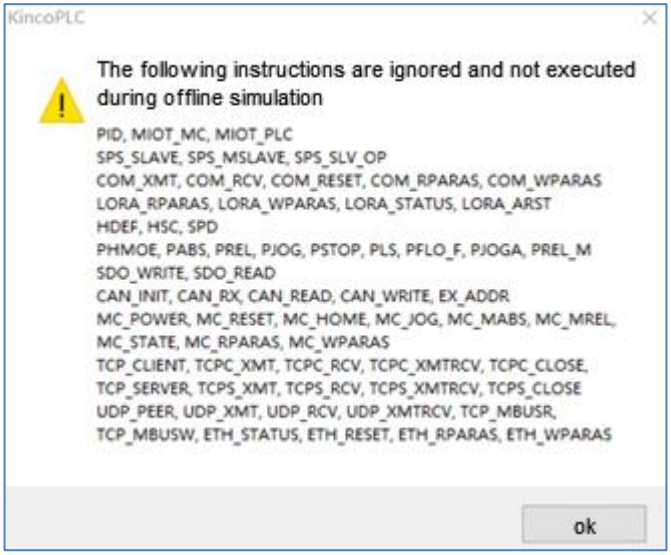
2、Click on the menu bar icon to compile the project, and then the [Information Output Window] at the bottom of the screen will prompt that the compilation is successful. Note: Some instructions do not support offline simulation for the time being, so such instructions cannot be included in the offline simulation program.。

3、After the compilation is successful, click the icon in the menu bar , start the offline simulator. Or start the offline simulator directly, and the simulator will automatically compile when starting.

2.2 Instructions that do not support simulation

Offline simulation does not support some commands for now.

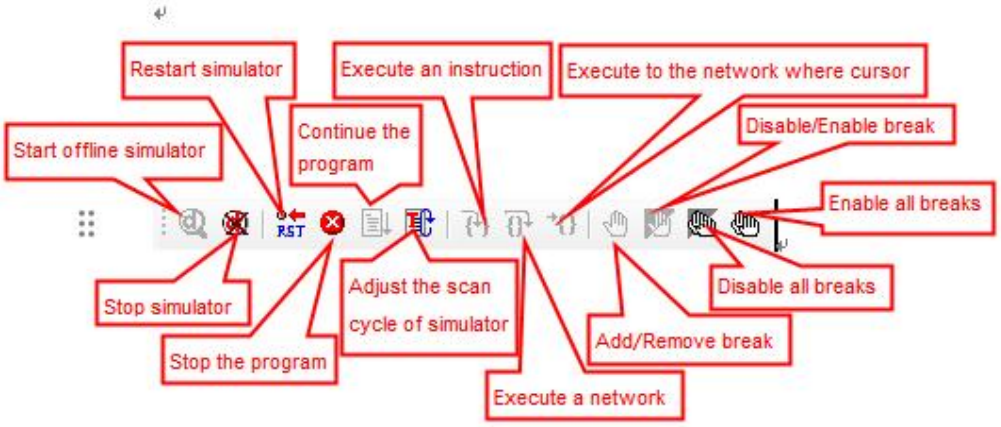
If the program contains instructions that do not support simulation, a prompt box as shown in the figure below will pop up. After the user clicks the [OK] button, the simulator will ignore the unsupported instructions and continue to run.



The commands that do not support offline simulation for the time being are:

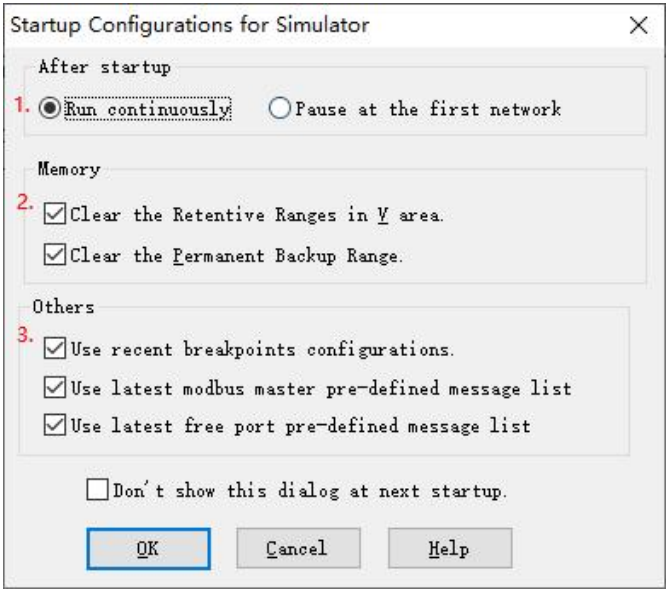
- Various high-speed pulse output functions and instructions, including PLS, position control instructions, etc.
- Various high-speed pulse input functions and instructions, including HDEF, HSC, SPD, high-speed counter wizard, etc.
- Various CAN communication functions and commands, including CAN free communication, CANOpen master station, Kinco motion control, etc.
- Various Ethernet functions and related instructions, including Ethernet free communication, Modbus TCP Client, etc.
- Kinco interconnection protocol functions and related commands.
- LoRa communication function and related instructions.

2.3 simulation tools



Three、Offline Simulation Startup Options

[Debug]--[Start Offline simulator] in the menu bar of the software or click the shortcut icon the interface [Offline Simulation Startup Options] will pop up by default



When starting the simulator, you can set the startup options, three options, one of the selected options will

execute the function

1、Status when the simulator starts: continuous running or paused on the first network

(1) Continuous operation. After entering the simulator, the program will cycle in sequence until other operations are performed or the emulation is exited.

(2) Pause on the first network, after entering the simulator, the program will pause on the first instruction of the first network

If it is set to pause on the first network, for example, when the scan period is set to 5 seconds, you can see that after starting the simulator for 5 seconds, it will start the first scan and pause on the first network. (Set the scan period to 1 second or more to see this effect).

2、Access area: choose whether to clear the data retention area and permanent data backup area in the V area at startup

What is cleared is the setting of the V area in the "Data Hold" section of the "PLC Hardware Configuration"

3、other options::

Use the latest breakpoint setting: refers to the situation where the latest breakpoint takes effect, instead of deleting the breakpoint and not using the latest power-off setting. After startup, all breakpoints are in an invalid state, and users can modify them as needed.

Use the last predefined communication message table of the modbus master station: use the last predefined communication message of the modbus master station and its effective status. If the message is not used, it will not be deleted, but it will be in an invalid state.

Use the latest free protocol predefined communication message table: Use the latest free protocol predefined communication message and its effective status. If the message is not used, it will not be deleted, but it will be in an invalid state.

Remarks: This dialog box will not pop up at the next startup, and it will be saved with the user project after setting. If you want to open the dialog box again, go to the menu bar--tools--software settings--offline simulation settings, re-check the direction before the startup option dialog box pops up at startup, and confirm. It will pop up when you start the simulator again. **Note: The two options in [Software Settings] - [Offline Simulation Settings] are saved with the user project.**

Four、Offline simulation mainly includes four functions

4.1 basic simulation

4.1.1 The running state is distinguished by the color change of the node

a. Red: Indicates nodes that are being processed and have no results

b. Blue: Indicates the node that has been executed (CR is true during execution, a color)

c. Orange: Indicates the node that has been executed (CR is false during execution is a color, in fact, it has not been executed)

d. Green: Indicates the state of the network being executed and executed during single-step operation, and the color of the left and right thick vertical lines turns green

4.1.2 Introduction to basic simulation operation instructions

1、【Start Offline Simulator】: Click the icon

2、【Turn-off Offline Simulator】: Click the icon

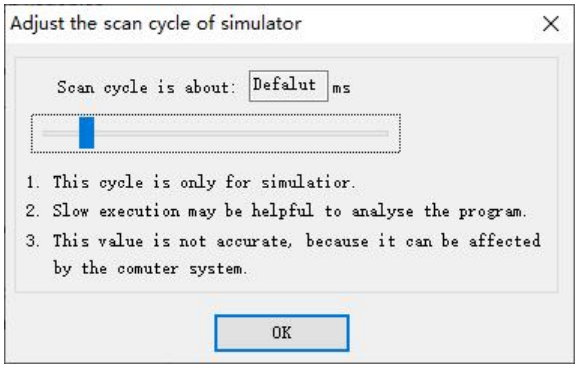
3、【Restart simulator】: Click the icon

4、【Stop program】: Click the icon

5、[Continue to run the program] When the program status is in the paused state, click

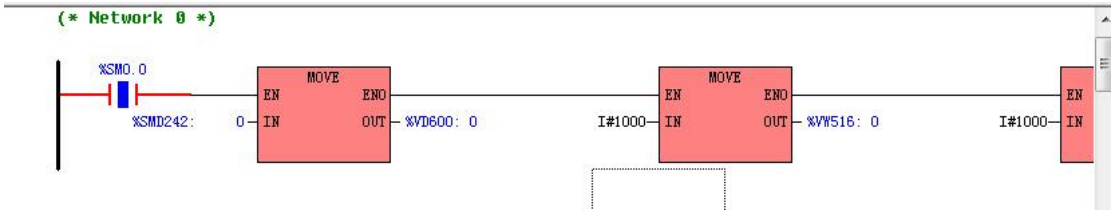
6、[Adjust the scan cycle of the offline simulator]: click the icon

Adjust the scan cycle of the offline simulator, if not adjusted, the default value will be selected automatically. The scan period can only be modified when the offline simulator is started, and the time cannot be input independently, and the time can be set by dragging the box. Time settings are optional: fastest (is it determined by the hardware, how much is it generally?), default (20ms?), 50ms, 100ms, 200ms, 500ms, 1000ms, 2000ms, 5000ms, 10000ms. This scan cycle is only for the simulator and does not affect the operation of the real PLC.

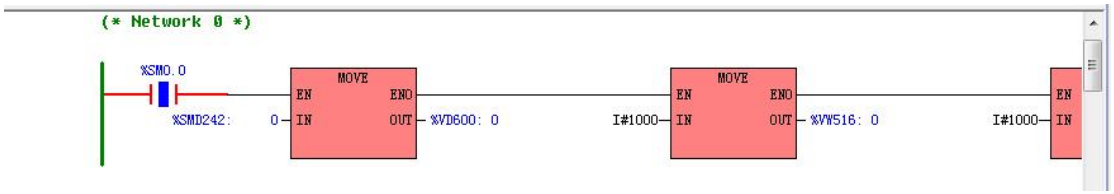


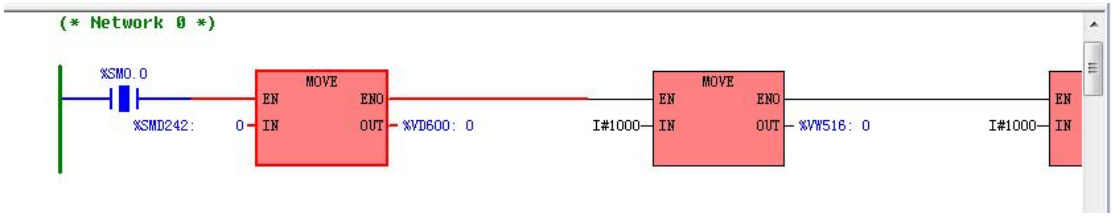
7、【Execute an instruction】: Click the icon

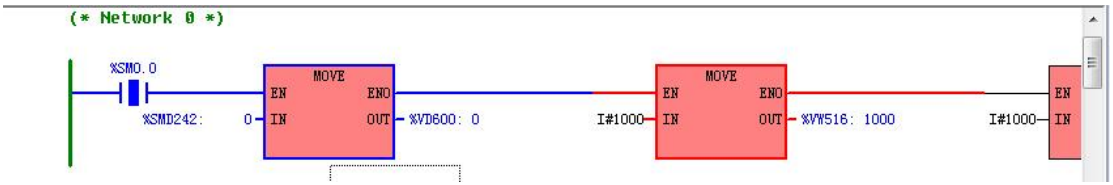
For example: first pause on the first network, and then click to execute an instruction, the effect is as follows, if the front switch is turned on, after the execution of the MOVE instruction, the color of the function block will change to blue



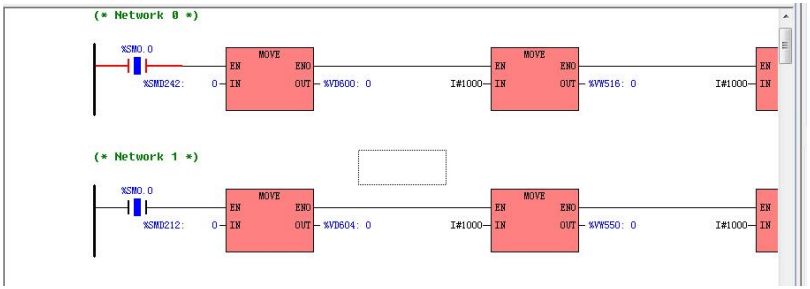
The diagram shows: paused on the network 0







8、【Execute a network】: click icon to execute a network that has not been executed at the current location.



9、[Execute to the network where the cursor is located]: click icon, execute to the network where the cursor is located, and can only execute downward.

When the cursor position is not in any network, the icon is gray and no operation can be performed; when the cursor is placed in the same network as the current execution position, click to execute to the network where the cursor is located, and there will be no action. If the cursor is placed on the network next to the current execution position, the program will execute to this network; if the cursor is placed on the previous network of the current execution position, click to execute to the network where the cursor is located, and nothing will happen.

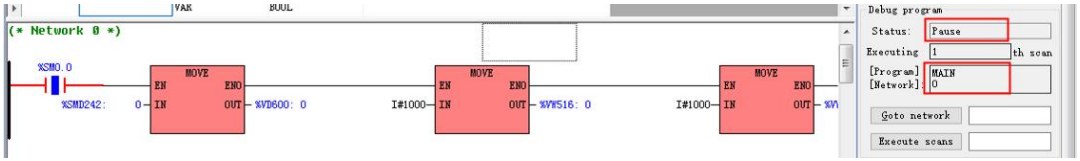
10、The program debugging part of the emulator interface includes: current state, N scans being executed, program Network, forward to the network, forward N scans.

The current status, the N times of scans being executed, and the program Network are all used as status indicators for users to view.

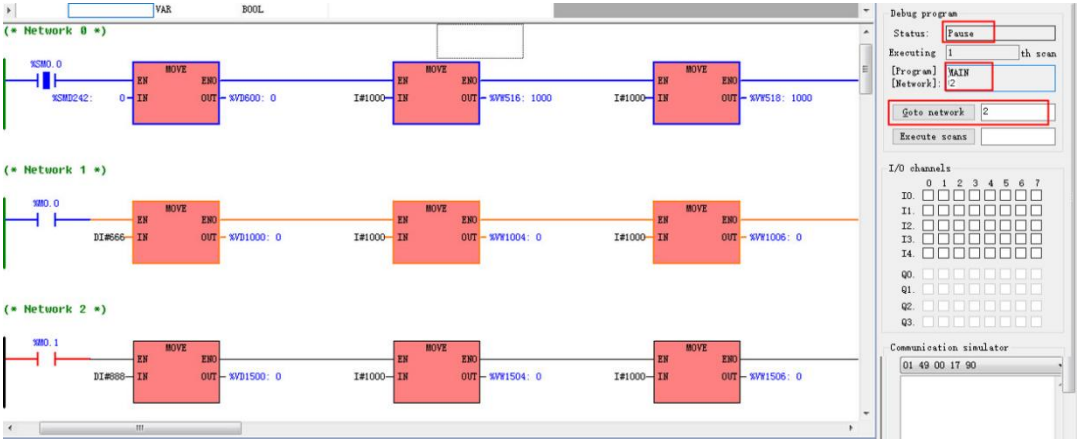
Advance to the network (valid in the paused state): In the paused state, you can enter the number of networks to advance in the blank space behind, and then click forward to the network, and the program will execute to the specified network and pause.

For example: now the network is paused at Network 0, now we enter 2 in the blank space after "advance

to network", that is, to advance to Network 2, the operation effect is as follows:



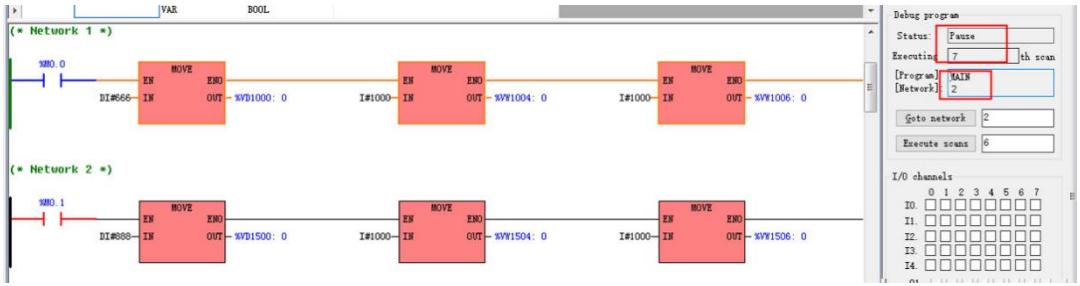
The picture above shows the first scan, paused at Network 0



The picture above shows that after the setting advances to the network, it pauses at Network 2

Forward N scans (valid in paused state): In the paused state, advance N scans. Then enter in the blank space after "Forward N scans", and then click Forward N scans, the program shows that it is scanning N+1 and pauses.

For example: the above example is to suspend Network 2 in the first scan, if you want to set forward N scans again, enter 6 in the blank space after "Forward N scans". After performing this operation, Network 2 in the seventh scan will be suspended. The effect is as follows:



4.2 breakpoint debugging

4.2.1 Network interruption points and their state change indications

Dark red circle in the upper left corner of the left vertical line of the network: if there is, it means that there is a breakpoint in the network, if it is not, it means that there is no breakpoint in the network.

The circle has two states: (1), if the circle is hollow , indicating that the network interruption point is prohibited

(2), if the circle is solid , Indicates that the network breakpoint is enabled

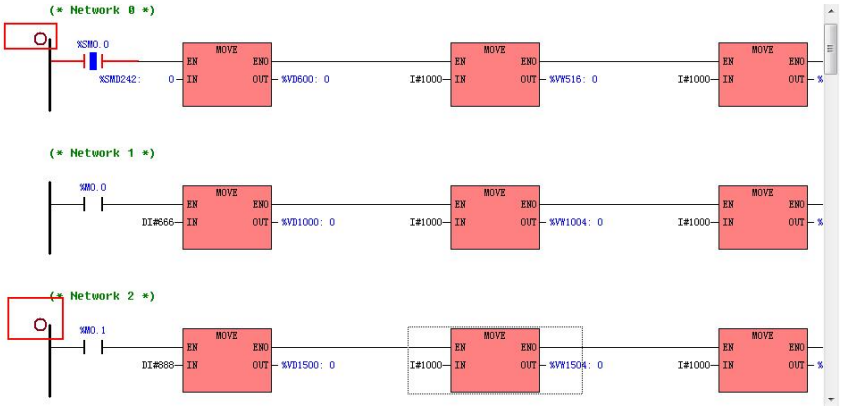
4.2.2 breakpoint operation

1、[Add/Delete Breakpoint]: Click Add/delete breakpoints, when the cursor is not on any node in the network, the is gray and cannot be operated. Put the cursor in any network to add/delete breakpoints. Put the cursor anywhere in the network, click the join/delete breakpoint, join and delete are executed alternately. The first click is to add and enable breakpoints, and the second click is to delete breakpoints. It is judged that the sign of adding and deleting breakpoints is in the upper left corner of the left vertical line of the operation network, and there will be a solid circle or no circle. A solid circle indicates that a breakpoint is enabled, and a circle without a circle indicates that the network has no breakpoint.。

2、【Disable/Enable Breakpoint】: Click icon to disable/enable the breakpoint, which works alternately, and can only be used for the network where the current cursor is located and has a breakpoint. The breakpoint can be enabled or disabled and can be judged to be enabled according to the status of the circle in the upper left corner of the left vertical line of the network. Still prohibited.

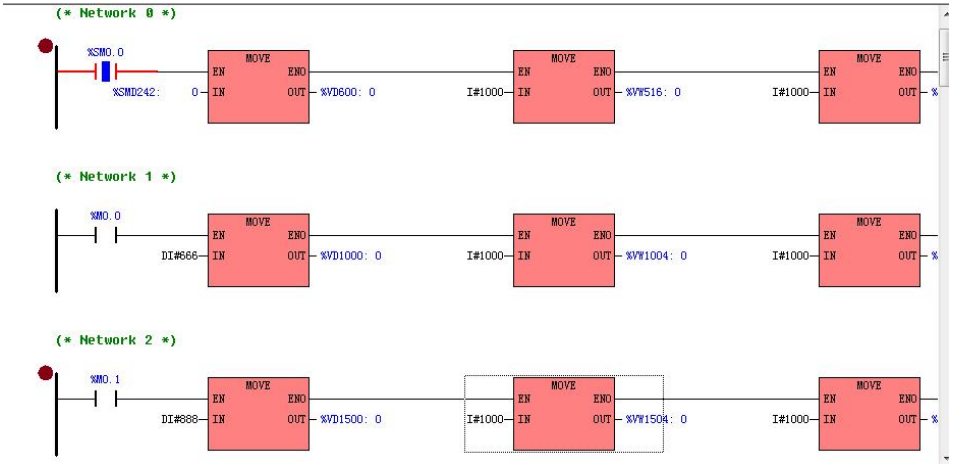
3、【Prohibit all breakpoints】: Click the icon to prohibit all breakpoints

Prohibition of all breakpoints is to prohibit all breakpoints in the program. After prohibition, you can see that there is a hollow circle in the upper left corner of the left vertical line of the network with breakpoints, as marked in the red box in the figure below



4、【Enable all breakpoints】: Click to enable all breakpoints

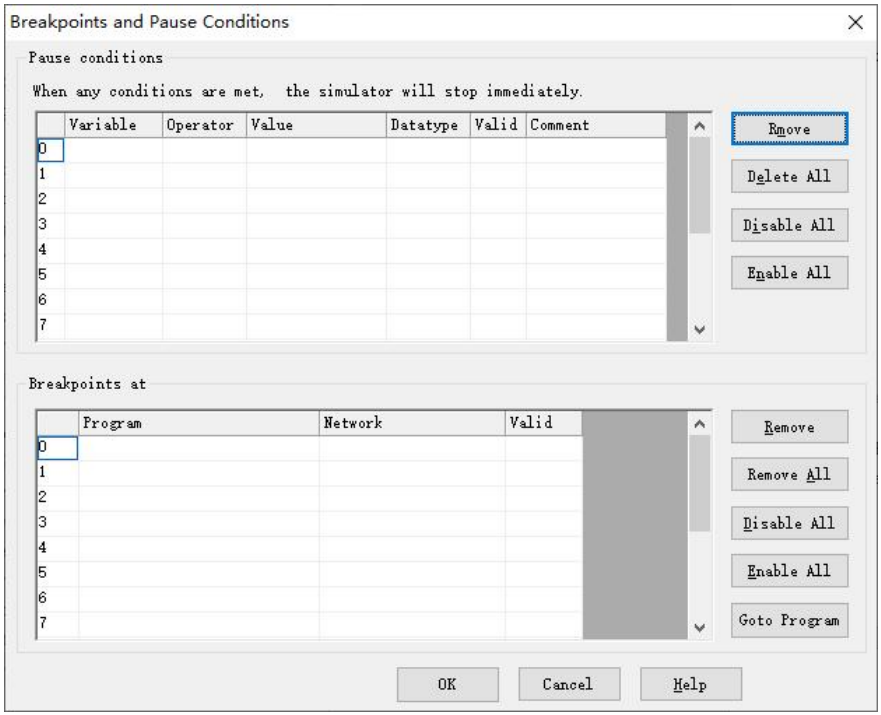
Enabling all breakpoints is to enable all breakpoints in the program. After clicking the icon, you can see that there is a solid circle in the upper left corner of the network front end with breakpoints, as shown in the figure below:



5、Breakpoints and pause conditions

This operation is opened in the menu bar [Debug]--[Power-off and Pause Conditions]. Click this interface to view and edit the pause conditions and breakpoint list in a unified manner. In the pause condition, the memory address needs to be self-input. Set the data type and value corresponding to the memory address. The operator can be selected, including: greater than, greater than or equal to, less than, less than or equal to, equal to, not

equal to. Whether it takes effect can be set individually, or you can use all enable or disable all on the right.



4.3 IO

Here the simulation needs to be run continuously to see the effect, and the effect cannot be seen by single-step running while paused.

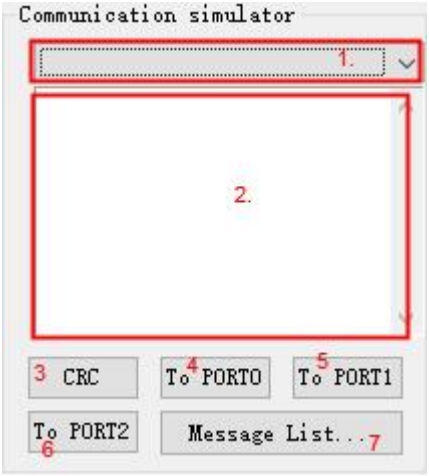
To simulate the input signal and output signal of the I/O channel, manually check the input I to simulate the input signal. Output Q changes with program control

Example: When I0.3 and I1.1 are turned on, Q0.0 and Q0.2 are also turned on; when I0.3 and I1.1 are turned off, Q0.0 and Q0.2 are turned off

Figures:

4.4 communication simulation

For message format, please refer to 【Help】--【Help Topics】in the menu bar of the software. Appendix A Using the Modbus RTU protocol to communicate with third-party equipment -- 2. The basic format of the Modbus RTU message, which will not be explained in this description.



1. Browse or select the recently sent message

2. Fill in the message

3. Click to automatically generate the CRC check code for filling in the message in 2 places

4. Send the message filled in 2 to PORT0

5. Send the message filled in 2 to PORT1

6. Send the message filled in 2 to PORT2

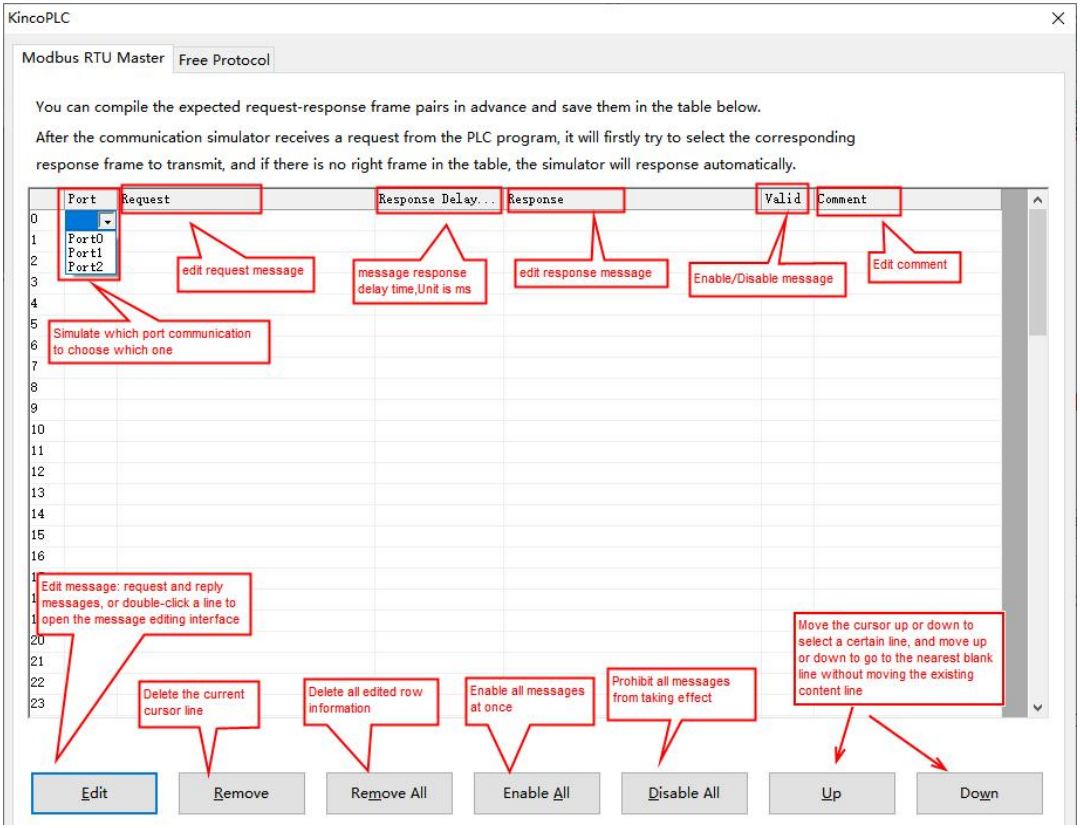
7. Open the communication message table

The messages in the communication message table can be regarded as the messages sent to the PLC by the equipment communicating with the PLC.

The communication message table can set the Modbus RTU master station and free communication (the message edited here will be saved with the project)

The communication emulator preferentially uses the messages set by the user in the table below. If the received message does not exist in the list, the emulator will automatically send the correct response message.

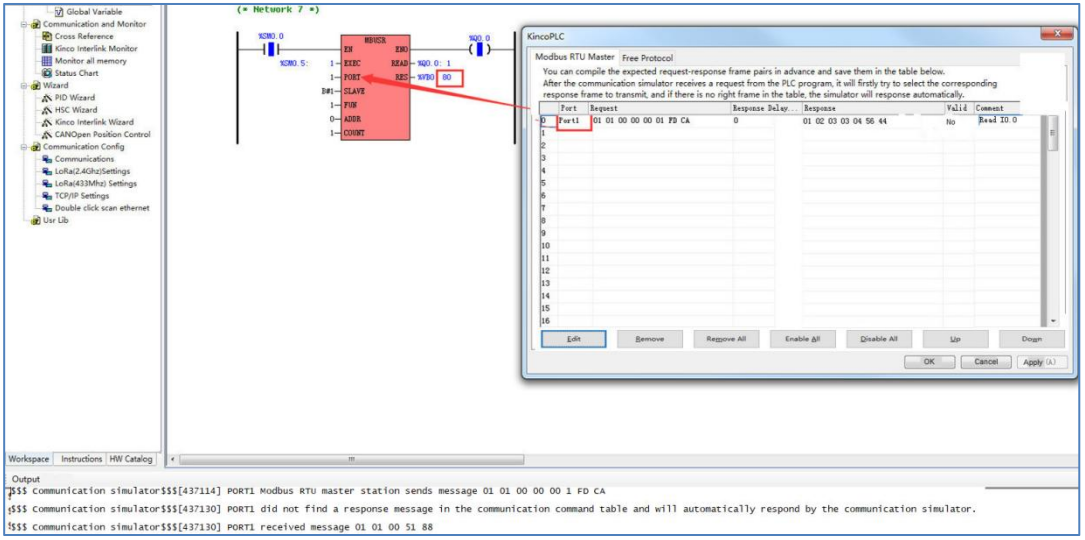
Open the communication message table and enter the free communication table first



(1) Modbus RTU main station

For example: In the PLC hardware setting, PORT1 is used as the Modbus master station, and the program reads the status of I0.0 of the No. 1 slave station

a. The message settings edited by the message table are invalid, and you can see [information output window]



$$$ Communication Emulator $$$ [437114] PORT1 Modbus RTU master send message 01 01 00 00 00 01 FD CA

$$$ COMMUNICATION EMULATOR $$$ [437130] PORT1 No response message found in the communication command table, will be answered automatically by the communication emulator.

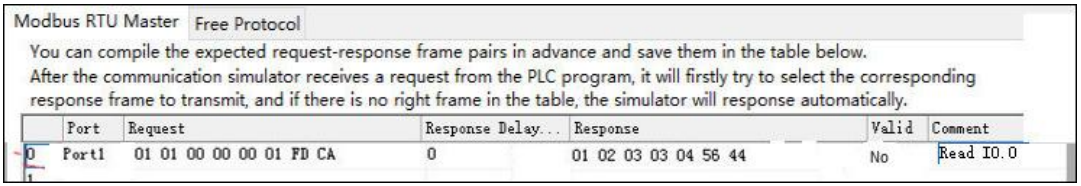
$$$ Communication Emulator $$$ [437130] PORT1 received message 01 01 01 00 51 88

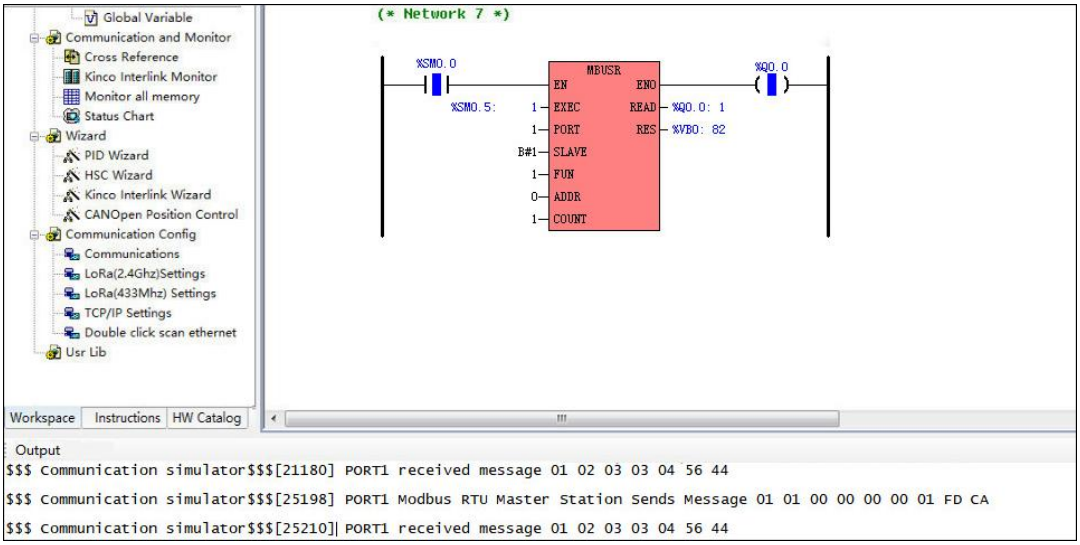
At the same time, the return value VB0 bit 80 of the MBUSR instruction indicates that the communication is successful.

Note: [Information Output Window]$$$ Communication Simulator $$$ [437114] PORT1 Modbus RTU master sends message 01 01 00 00 00 01 FD CA. Among them, [437114] represents the time from the start of offline simulation this time, in milliseconds.

[437114]--[437130] It takes 16 milliseconds to process a communication message in offline simulation

b、The message settings for message table editing are valid





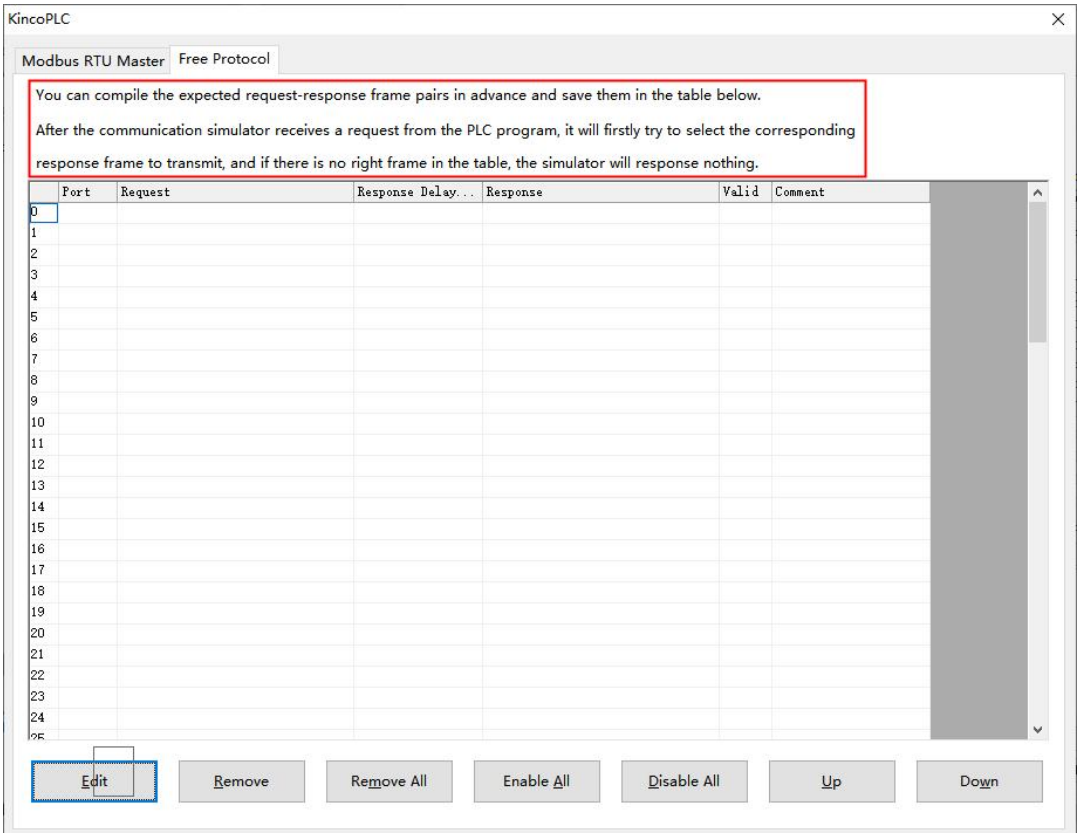
You can see the [Output]

$$$Communication Simulator $$$ [25198] PORT1 Modbus RTU Master Station Sends Message 01 01 00 00 00 00 01 FD CA

$$$Communication simulator$$$ [25210] PORT1 received message 01 02 03 03 04 56 44

Meanwhile, the return value VB0 bit 82 of the MBUSR instruction indicates a communication timeout.

(2)Free communication (only XMT RCV command is supported, COM_XMT COM_RCV is currently not supported)



The usage method is the same as Modbus RTU, except that the Modbus command in the program is changed to XMT command, so I won't repeat it here.

Appendix D Introduction to the screen part of the all-in-one machine

Kinco integrated controllers are divided into two series: HP series and MK series. The configuration software of the HMI part of the two is not the same software. The following will briefly introduce the use of the HMI part of the two series.

One、Instructions for using the HMI part of the HP series

1.1 Use of HMI

HMI programming software is: Kinco HPBuilder. To download, please log on to the official website -- [Software] -- [PLC Programming Software] under the download directory, and select the all-in-one machine HMI programming software to download. The official website website: www.kinco.cn.

1.1.1 Engineering production

The following describes the steps to use Kinco HPBuilder to create a project (taking HP043 as an example).

1. Create a project

Start Kinco HPBuilder

1、New construction:

(1) Menu bar【File】--【New Project】;

(2) Enter the project name;

(3) Select the save path of the project folder;

(4) Click [Create] to complete the project creation

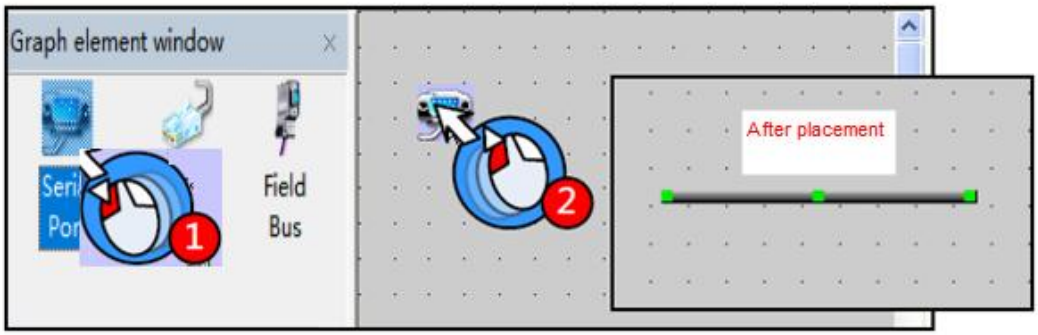
2、Device selection, connection and parameter setting

(1)Equipment selection

①Device selection--select communication method

Select "Serial Port" from [Component Library Window]--[Communication Connection] and drag it to the

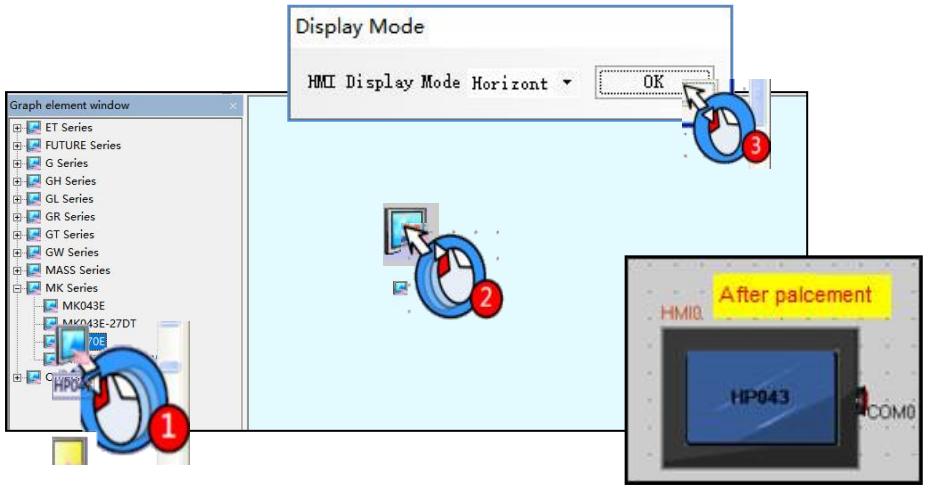
topology window and place it



②Alternate option--select HMI model

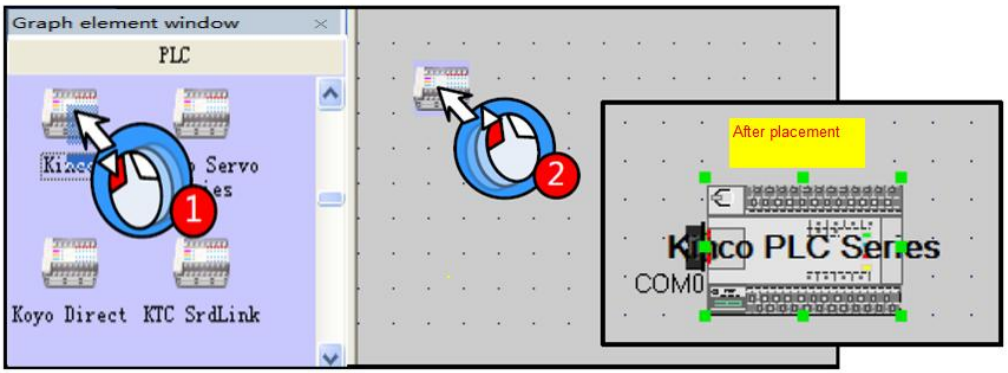
Select "HP043" from [Component Library Window]--[HMI], drag it to the topology window and place it

When dragging and dropping the HMI, the system will pop up the [Display Mode] dialog box as shown in the figure below, requiring the user to select the display mode of the touch screen. The system provides two options of "Horizontal" and "Vertical" (in this example, "Horizontal" is selected), click [OK]



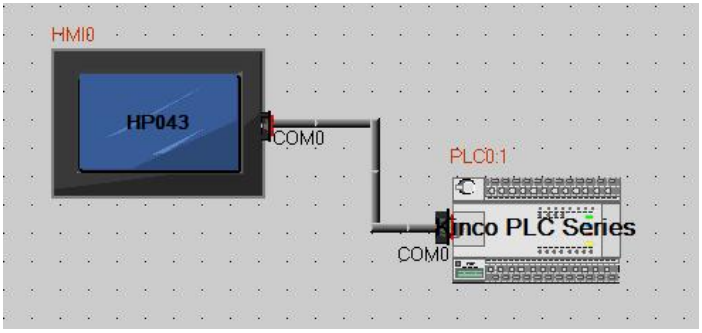
③Device selection--select PLC model (communication protocol)

Select "Kinco PLC Series" from [Component Library Window]--[PLC] and drag it to the topology window and place it



(2)device connection

Select the HMI and drag its "COM" port close to the left end of the communication connection. Until the HMI is dragged, one end of the communication cable also moves with the "COM port". At this point, the connection between the HMI and the serial port cable is successful. Use the same method to connect PLC and serial cable



(3)Parameter setting--HMI setting

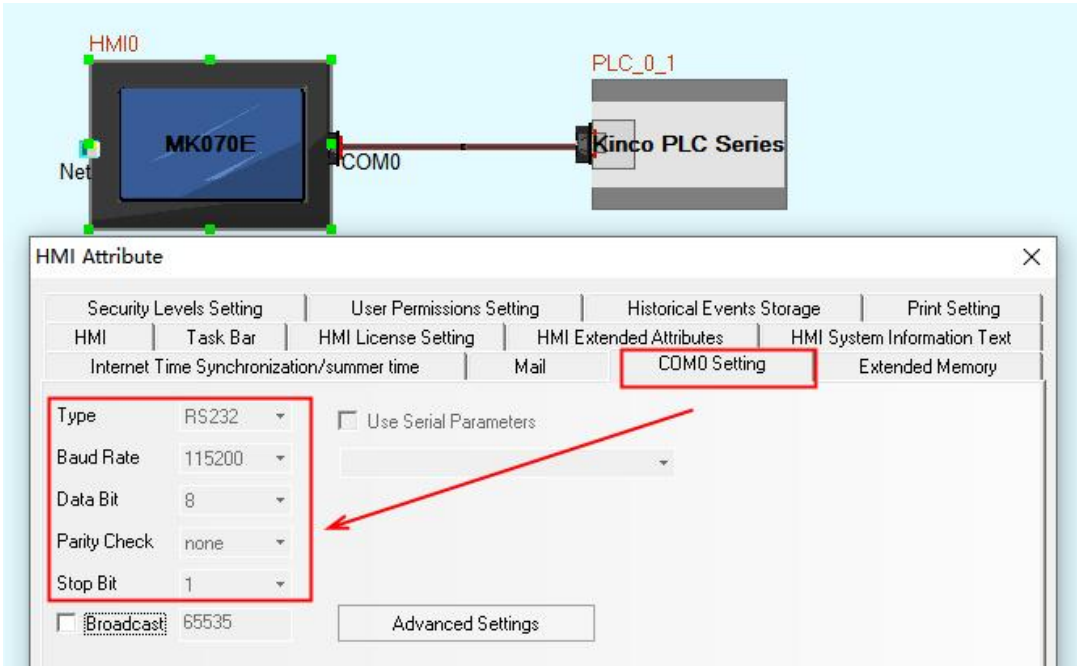
①Double-click HMI, the system will pop up [HMI Properties] property box

②Switch to [Taskbar] property page

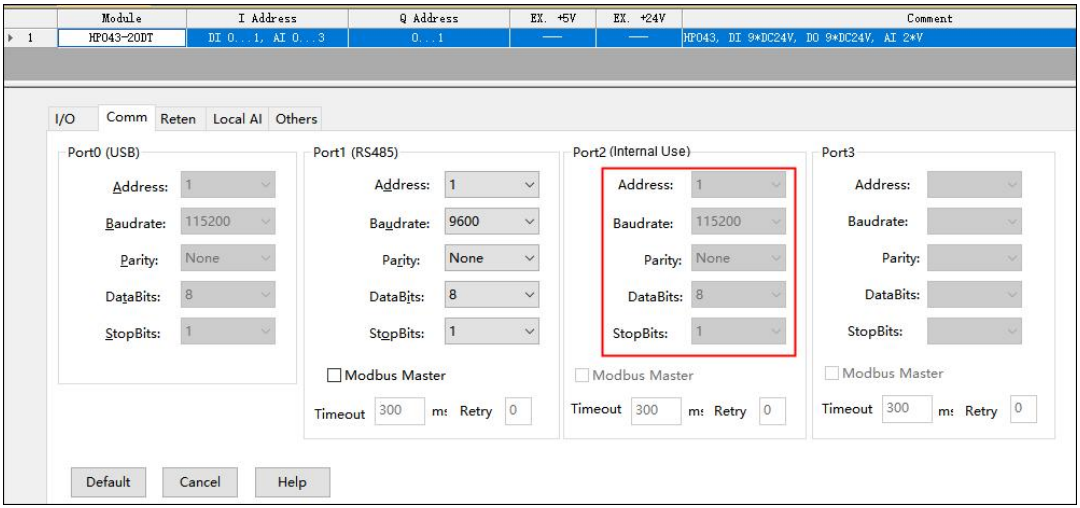
③Uncheck the "Show taskbar" option in the [Taskbar] property page

④Switch the [Serial Port 0 Setting] property page to configure the parameters of the serial port 0 according to the actual PLC communication parameters, and other options adopt the default settings

(3)HMI serial port 0 setting



Communication parameters of the actual PLC



Edit the configuration screen, you can refer to the Kinco HMIware user manual.

HMI provides external USB master-slave ports. For details, refer to the Kinco DTools configuration editing

software user manual.

1.1.2 HMI manual download link

DL link:

https://www.kinco.cn/Download/usermanual/HMI/Kinco%20DTools%E7%BB%84%E6%80%81%E7%BC%96%E8%BE%91%E8%BD%AF%E4%BB%B6%E4%BD%BF%E7%94%A8%E6%89%8B%E5%86%8C20200330.pdf

Two、Instructions for using the HMI part of the MK series

2.1 Use of HMI

HMI programming software is: Kinco DTools. To download, please log in to the official website - [Software] - [HMI Programming Software] under the download directory, select Kinco DTools to download. Official website: www.kinco.cn.

2.1.1 Use of Project

The following describes the steps of using Kinco DTools to create a project (taking MK070E as an example).

一、Create Project

Open Kinco DTools

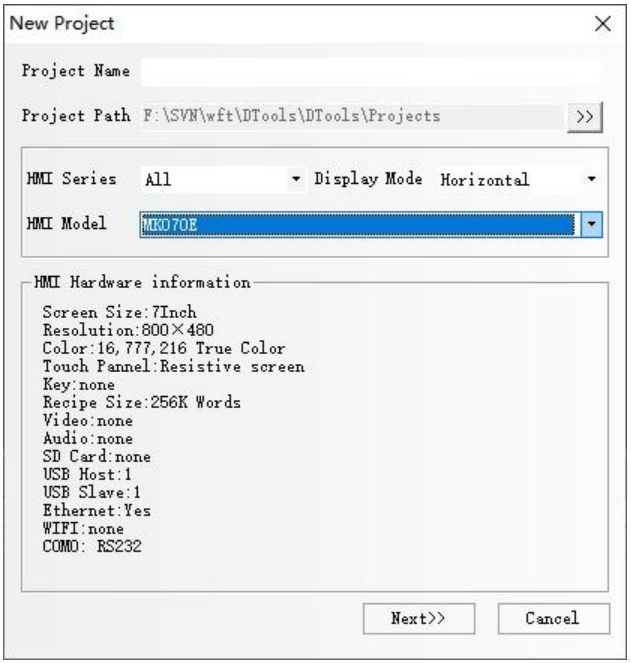
1、Create Project:

（1）Menu bar【File】--【New Project】;

（2）Enter the project name.

（3）Select the save path of the project folder.

（4）Select HMI Model MK070E



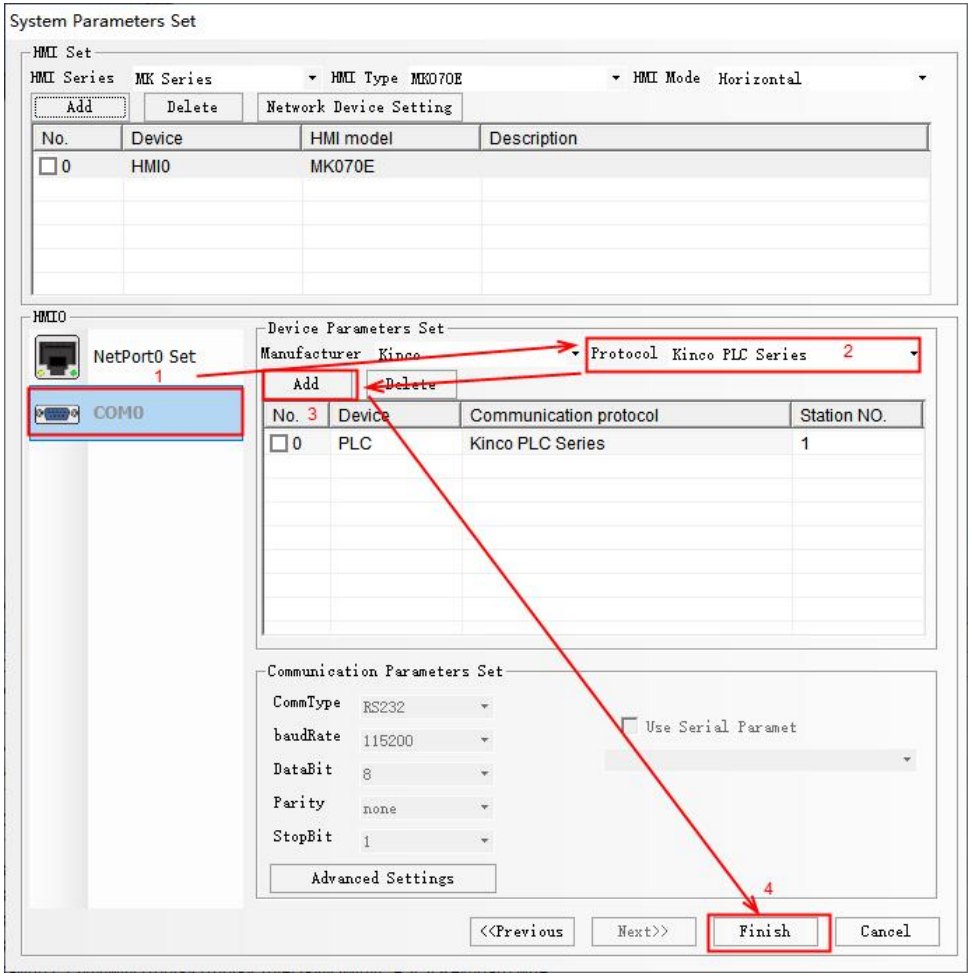
(5)Click Next to enter the following system parameter setting screen:

① Click the serial port 0 setting (note that it must be COM0. COM2 is not available)

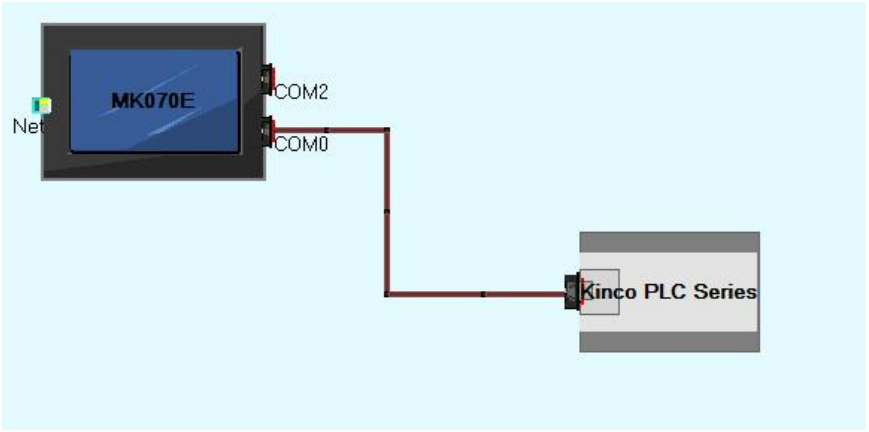
② Select Kinco PLC Series communication protocol

③ Click Add (communication parameters have been solidified and do not need to be set manually)

④ click finish

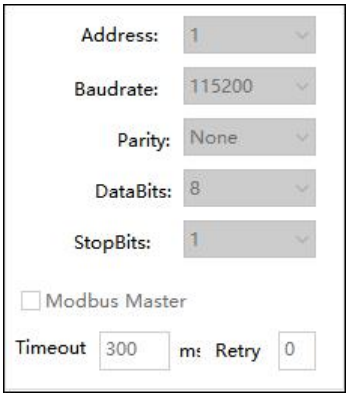


The completed hardware configuration connection diagram is as follows:



Notice:

The communication parameters of the actual PLC part are as shown in the figure below. The communication parameters have been internally solidified in the PLC and do not need to be set in the engineering hardware configuration of KincoBuilder.



For editing the configuration screen, please refer to the Kinco DTools configuration editing software user manual.

HMI provides external network ports and USB master-slave ports. For specific usage, refer to the Kinco DTools configuration editing software user manual

2.1.2 HMI manual download link

DL link:

https://www.kinco.cn/Download/usermanual/HMI/Kinco%20DTools%E7%BB%84%E6%80%81%E7%BC%96%E8%BE%91%E8%BD%AF%E4%BB%B6%E4%BD%BF%E7%94%A8%E6%89%8B%E5%86%8C20200330.pdf

Appendix E Introductory case (steps to create a project)

It is assumed that the user has already prepared the required hardware.

In order to help users quickly understand and master Kinco small PLC, we will give an example step by step how to create and debug a specific project. Please note that these are only descriptions of common steps. If users need to learn more about the specific functions of a certain aspect, please refer to the relevant chapters.

Suppose you want to create the following project:

- Project name: project
- Hardware: one Kinco-KS105-16DTCPU module, one switch module KS122-12XR, one analog module KS133-06IV.
- Realize the function: start Q0.0 --- Q0.7 in sequence, and execute this process cyclically, output 0-10V analog quantity to control the servo to move at a fixed speed, and input I0.0 to control the start and stop of the whole process.

In order to ensure a good program structure, we use three POUs in the project:

A subroutine named "Demo", the purpose is to achieve the required control function, cycle start Q0.0 --- Q0.7, control the servo movement;

An interrupt program named "INT_0" controls the start and stop of Q0.0 --- Q0.7 cyclic shift and the start and stop of servo;

A main program named "Main", which is the main loop task of the CPU, will call the "Demo" subroutine in the main program to execute the interrupt program.

1) Start KincoBuilder.

2) If necessary, use the following methods to set some default values used in KincoBuilder software.

Execute the [Tools] → [Software Settings...] menu command to enter the "Software Settings" dialog box. In this dialog box, the user can set some software default values, such as [default programming language], [default CPU model], etc. After clicking the [OK] button, the set default value will take effect and be automatically saved by KincoBuilder, and there is no need to set it every time.

Here we set [Default Programming Language] to be "Ladder Diagram LD".

3) Use any of the following methods to create a new project:

- Execute the [File] -> [New Project..] menu command;
- Click the icon on the toolbar

Then the "New Project" dialog box will appear. Here the user names the new project and selects the storage path. After the path and project name are determined, click the [Save] button to create a new project. In this example, we choose D:\temp as the project storage directory, and the project name is "project".

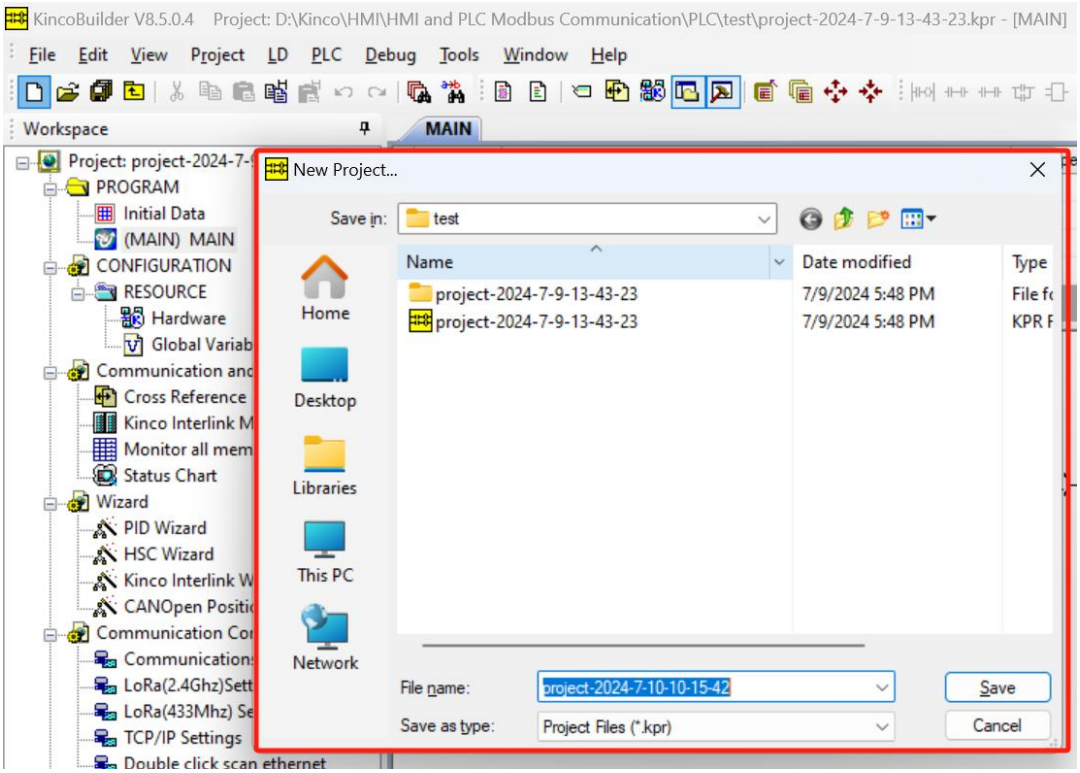


Figure 2-11 "New Project" dialog box

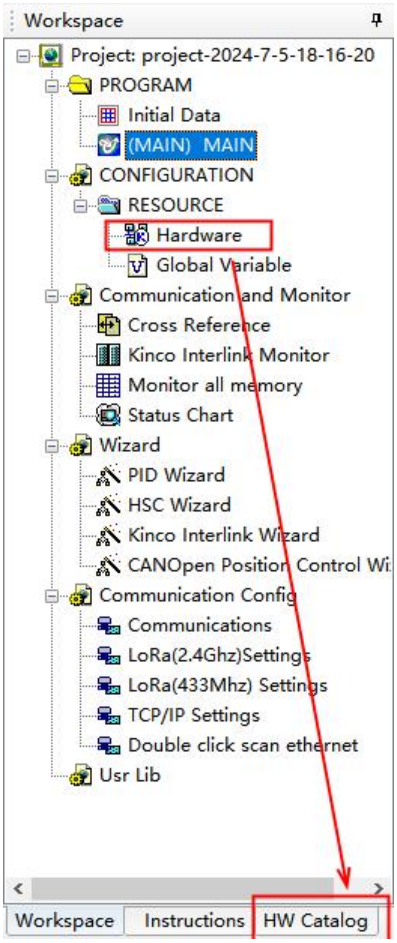
4) Perform PLC hardware configuration (a main program called MAIN is automatically generated after a new project is created)

When the user creates a new project, KincoBuilder will automatically add a CPU module of the [Default CPU Model] specified by the user in the "Software Settings" dialog box. Users can modify hardware configuration parameters at any time, but since hardware configuration is necessary information in a project, it is recommended that users complete the hardware configuration first in the project.

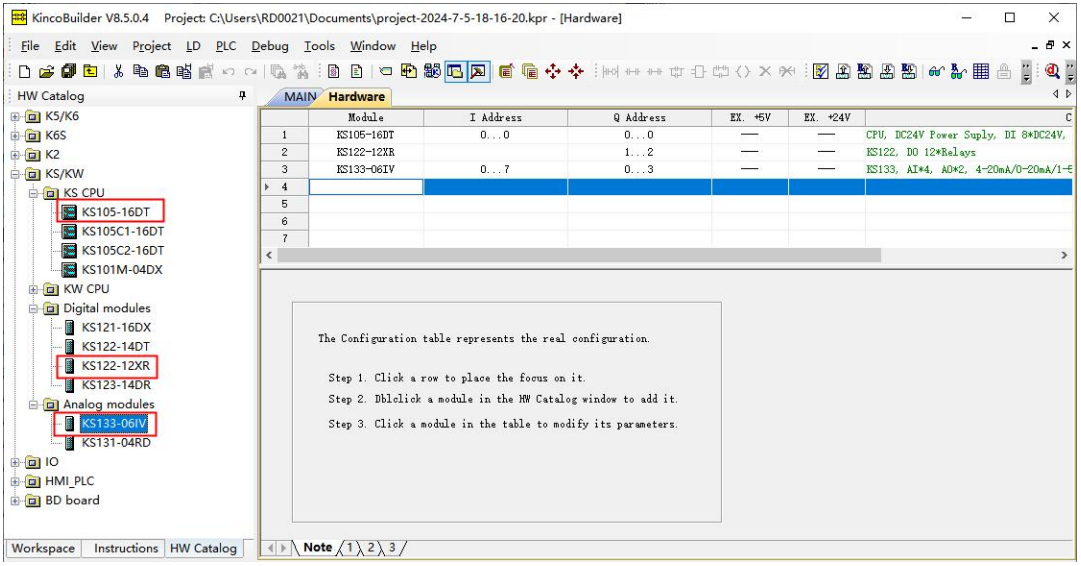
The figure below shows the process of hardware configuration in this project.

① Start hardware configuration

② Start PLC module list

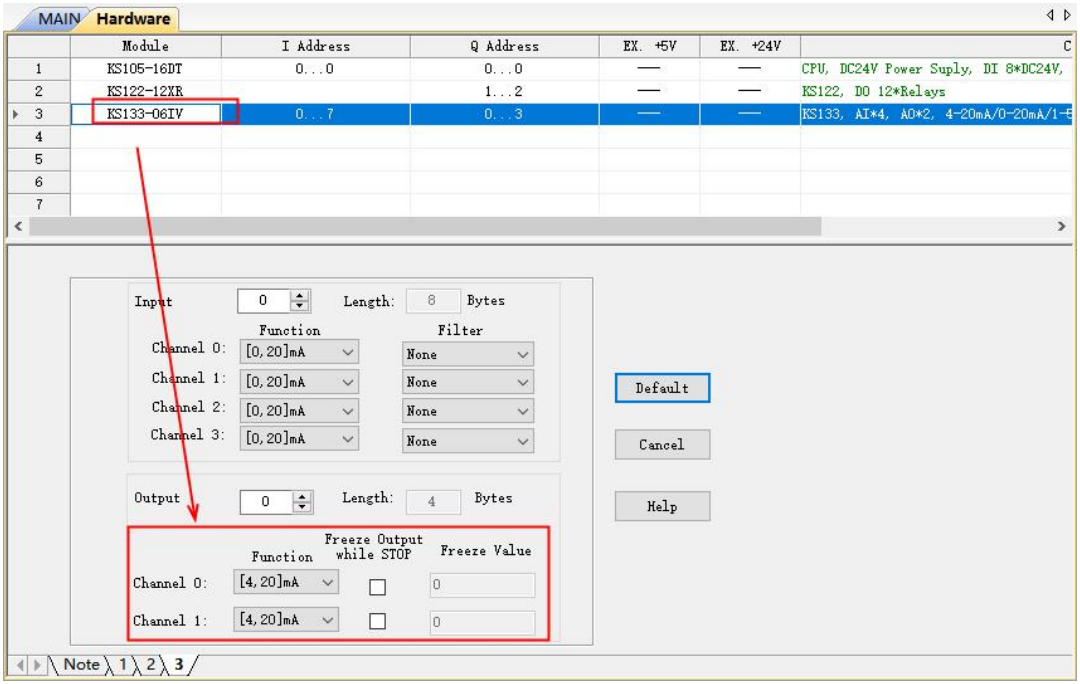


③ In the PLC module list, double-click to select the CPU module and expansion module to be used. In the hardware configuration, select the relevant CPU module and expansion module and right-click to delete it.



④ Select the module or CPU that needs to be set to set the hardware parameters

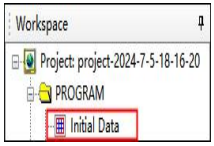
⑤ Set the signal form of analog output channel 0 to 0-10V, which is used to control the servo.



The above only sets the configurations that need to be used. For details, please refer to 4.3 PLC hardware configuration to understand the specific steps of hardware configuration, and I will not repeat them here.

5) About initializing the data table

According to actual needs, users can fill in the initialization data form at any time. In the initialization data table, the user can specify the initial value for the BYTE, WORD, DWORD, INT, DINT, and REAL data in the V area. Before entering the main loop when the CPU is powered on, the initialization data table will be processed once, and the initial value specified by the user will be assigned to the corresponding address. Refer to 4.4 Initialization Data Table.



Note: "Initialization data table", "data retention" of hardware configuration, and the memory area permanently saved by instructions in the user program will be restored or assigned after power-on

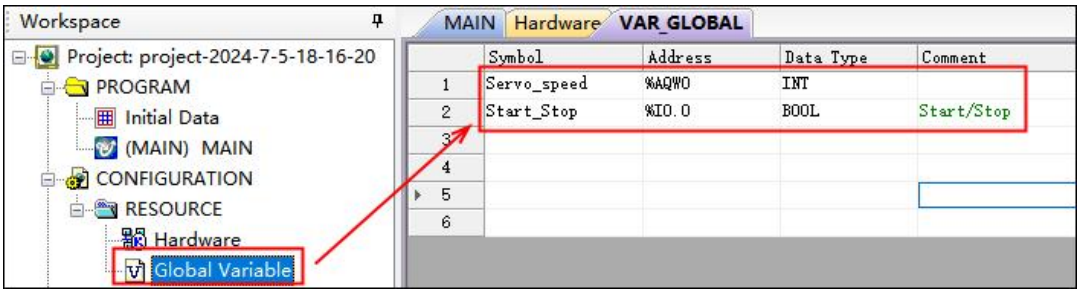
before entering the main loop. The sequence is: restore the memory data defined in the "data retention", assign initial values to the memory area defined in the "initialization data table", and restore the data permanently saved by the user using instructions.

6) Global variable table

In order to facilitate the user to write programs and enhance the readability of the program, the user can enter the global variable table to define the variables that need to be used, refer to 4.5 Global variable table for usage.

① Open the global variable table

② Define variable names and addresses, etc.



7) Write programs according to actual functional requirements

User programming can use two languages: IL or LD. The user can execute the [Project] → [IL language (instruction list)] or [Project] → [LD language (ladder diagram)] menu command to switch the language of the current program at any time. Note: Any LD program can be converted into an IL program, but only an IL program written according to certain rules can be converted into an LD program.

Below we will write a main program "Main", a subroutine "Demo" and an interrupt program "INT_0" for this example.

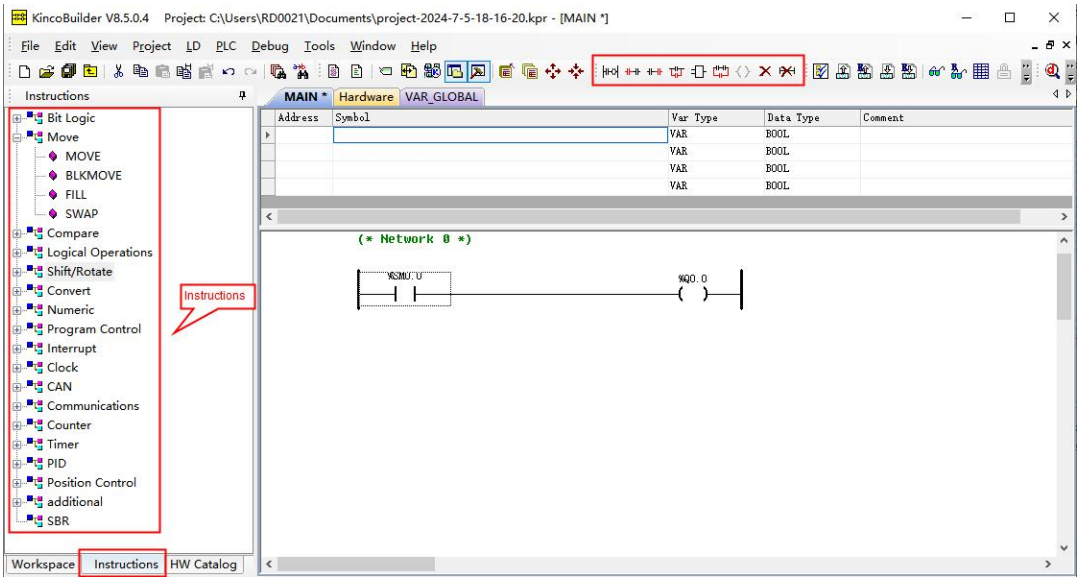
① Main program

When the user creates a new project, the software will automatically create a main program named "MAIN".

② Since only the subroutine "Demo" needs to be called in the main program, but the subroutine has not been established, so now we ignore the main program and continue to create the subroutine "Demo".

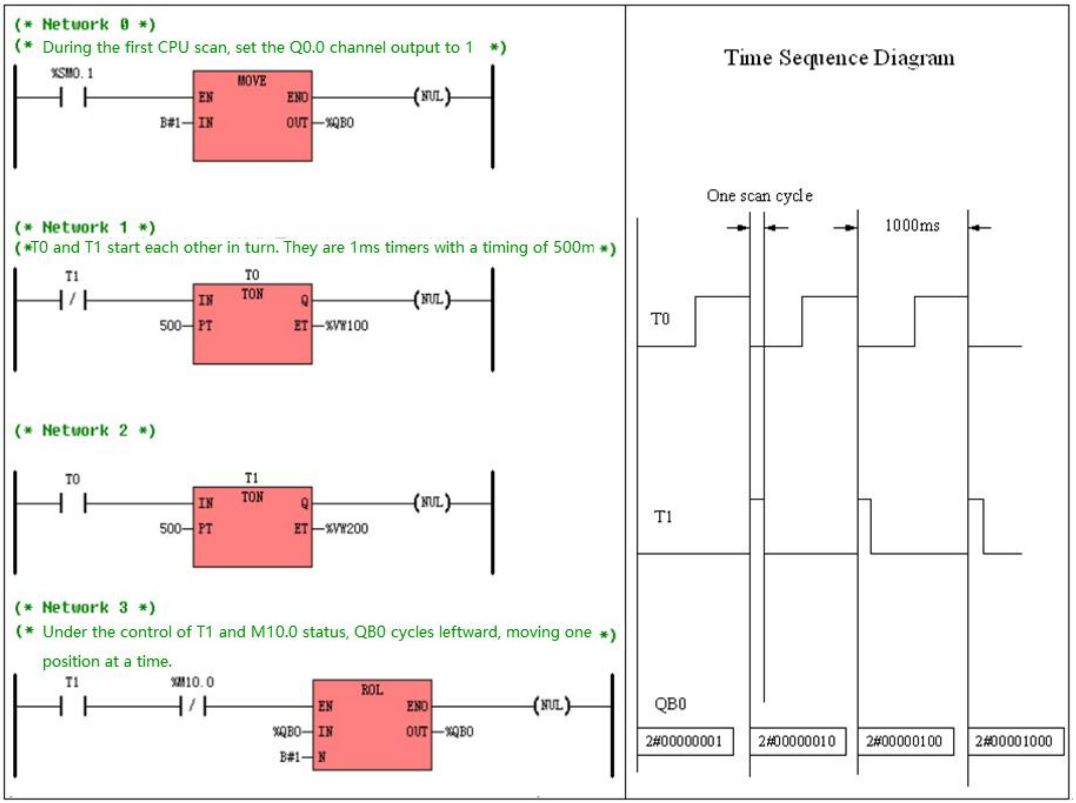
- First create a new subroutine using any of the following methods:
- Execute the [Project] → [New Subroutine] menu command;;
- Click the icon on the toolbar
- Right-click on the [Program] group node in the project manager, and execute the [New Subprogram] command in the pop-up menu.

Then a new subroutine will be created, whose name is "SBR_0" by default.

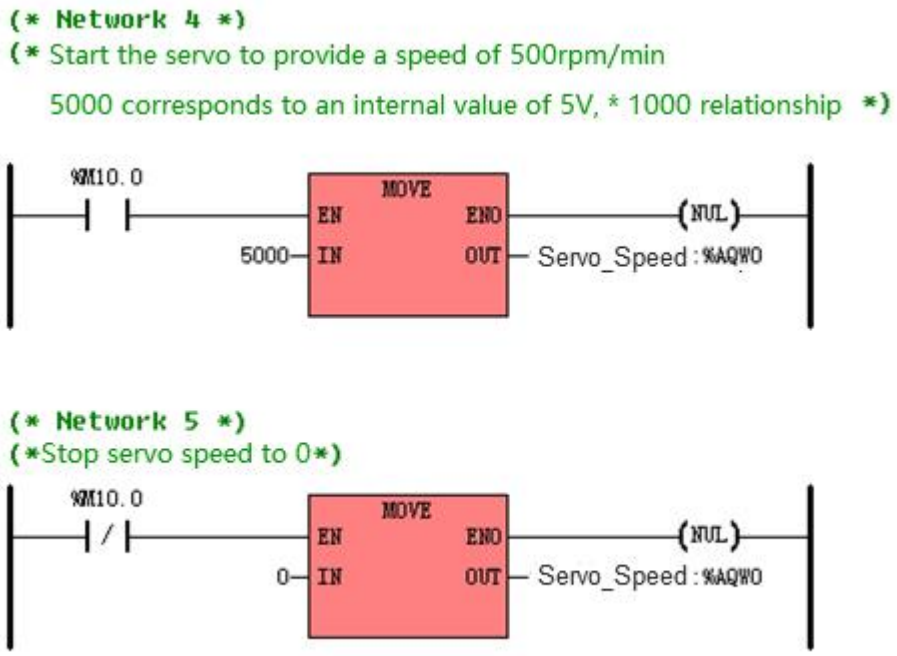


In the red box, you can choose to add network, add contacts, connect contacts in parallel and add corresponding instructions, etc. For details, please refer to 5.2 LD programming, which will not be described in detail here.

Now you can enter the program as shown in the figure below, the right part of the figure is the Timing diagram



Cycle start Q0.0 --- Q0.7 part of the program



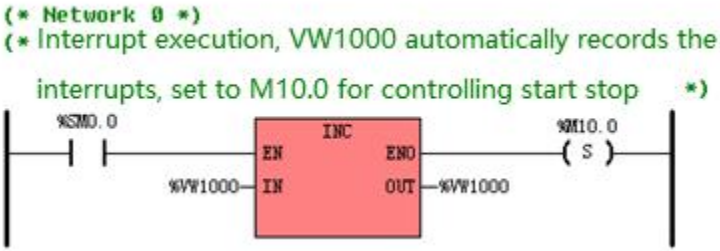
ontrol the servo start and stop part of the program

After the program is written, close the subroutine, and then click the right mouse button on the [(SBR00) SBR_0] node in the [Program] group of the project manager to pop up a menu. Execute the [Rename] command to change the name to "Demo", or execute the [Properties...] command, and modify it in the program's "Properties" dialog box.

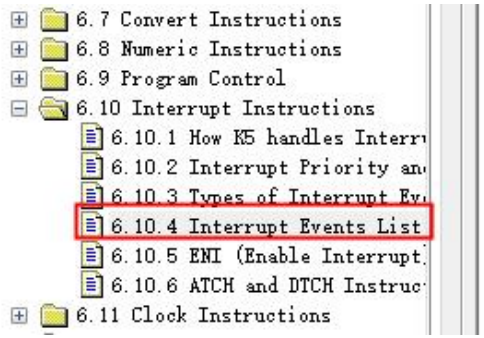
③interrupt program

- First use any of the following methods to create a new interrupt routine:
- Execute the [Project] → [New Interrupt Program] menu instructions;
- Click the icon on the toolbar
- ight-click the [Program] group node in the project manager, and execute the [New Interrupt Program] command in the pop-up menu.

Then a new interrupt routine will be created, whose name is "INT_0" by default.



For interrupt events, you can refer to the help to understand the corresponding event classification, list, etc.



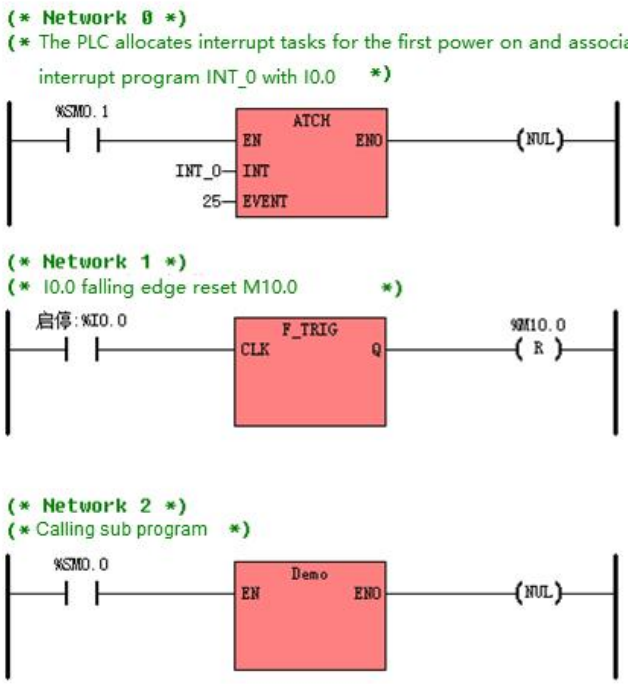
26	I0.0, Falling edge
25	I0.0, Rising edge
24	I0.1, Falling edge
23	I0.1, Rising edge
22	I0.2, Falling edge
21	I0.2, Rising edge
20	I0.3, Falling edge
19	I0.3, Rising edge
18	HSC0 CV=PV
17	HSC0 direction changed

This example does not rename the interrupt program. Users who need to rename the interrupt program can refer to the subroutine for the renaming method.

④ Re-modify the main program

Now that we have compiled the subroutine "Demo", we need to return to the main program to add its call instruction.

Switch to the editing window of the main program, and enter the following program:



8) Compile the project

After writing the project, it can be compiled. Before compilation, the software will automatically save the project to ensure that the latest input from the user is compiled. However, users are still advised to save their work at any time to avoid accidents.

Users can use any of the following methods to start the compilation process:

- Execute the [PLC] → [Compile Current Project] menu instructions;
- Click the icon on the toolbar ;
- Use the shortcut key F7.。

The compilation result will be in the "compilation information" page in the lower information output window. If there is an error in the project, double-clicking the error message in the compilation information will automatically locate the error. The user needs to modify the project according to the error message until

the compilation is successful.

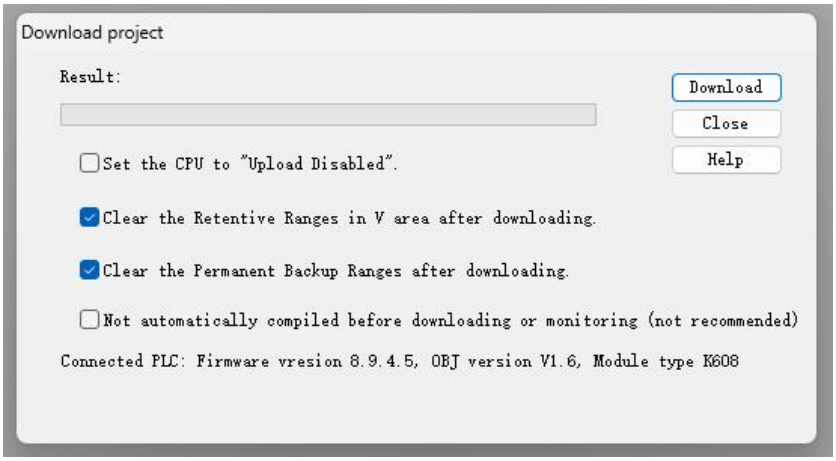
9) Download project

After the compilation is successful, the current project can be downloaded to the PLC.

Supplement: If you need to specify the communication parameters of the PC serial port, please refer to 2.5 How to connect the computer and PLC.

- Users can use any of the following methods to open the "Download User Project" dialog box:
- Execute【PLC】→【Download..】menu command;
- Click the icon on the toolbar ;
- Use the shortcut key F8.

➢ Download User Project" dialogue



✧ [The currently downloaded user project is prohibited from being uploaded]

If this option is selected, the CPU will prohibit anyone from uploading the project after this download.

If this item is not selected, the CPU will allow the user to upload according to the original authority after this download.

✧ [Clear the data holding area in the V area after downloading]

If this item is selected, after this download, the data in the V and C areas of the data holding area will be all cleared.

If this option is not selected, after this download, the data in the data holding area V and C will remain unchanged.

✧ [Clear the permanent data backup area after downloading]

If this option is selected, all data in the permanent data backup area will be cleared after this download.

If this option is not selected, the data in the permanent data backup area will remain unchanged after this download.

After the settings in the "Download User Project" dialog box are complete, click the [Start Download] button to download the project to the PLC. Now you can put the running switch of the CPU to the "RUN" position to see how the program is running.

10) Program debugging

Users can use the functions such as online monitoring and forcing provided by KincoBuilder to debug the program.

➢ Online monitoring

There are two modes of online monitoring:

- Monitor in variable status table. Users can enter the variables they want to monitor in the table for monitoring.
- Monitoring in the program. Users can observe the running status of the current program. Program modification is not allowed during online monitoring.

The online monitoring command can only take effect after opening the variable state table, LD program or IL program. Note that the online monitoring command is a multi-select command, if you want to leave the online state, just execute the online monitoring command again.

Users can use any of the following methods to enter the online monitoring state:

- xecute the [Debug] -> [Online Monitoring] menu command;
- Click the icon on the toolbar
- Use the shortcut key F6.

➢ Mandatory

The user can use the force function to forcibly modify the value of the variable in the I, Q, M, V, AI or AQ area of the PLC. The variables in the I, Q, M and V areas can be forced in bit, byte, word or double word mode, and the variables in the AI and AQ areas can be forced in words.

KincoBuilder allows forcing up to 32 variables simultaneously. Immediate instructions do not allow coercion.

Users can perform mandatory functions in any of the following ways:

- Coercion in the variable state table. The user can input the variables to be monitored and their mandatory values in the table, and then execute the menu commands (or right-click menu commands) [Force], [All Force], etc. to realize the corresponding functions.
- After entering the online monitoring state, directly force the variables used in the program.

Switching value: Right-click on the contact or coil, and execute the [Force to 0], [Force to 1] or [Force...] command in the pop-up menu;

Non-switching value: right-click on the variable name, and execute the [Force...] command in the pop-up menu.

At the same time, a variable may have the following values: external input signal (I, AI) or the result of instruction execution in the user program (Q, AQ, M, V), mandatory value specified by the user. Therefore, the following principles for mandatory values to take effect are stipulated:

✓ For the variables in the M area and V area, the forced value and the instruction execution result have the same priority. When forced by the user, the forced value will take effect; but the forced value will only take effect once in a scan cycle, and then the instruction execution result will take effect.

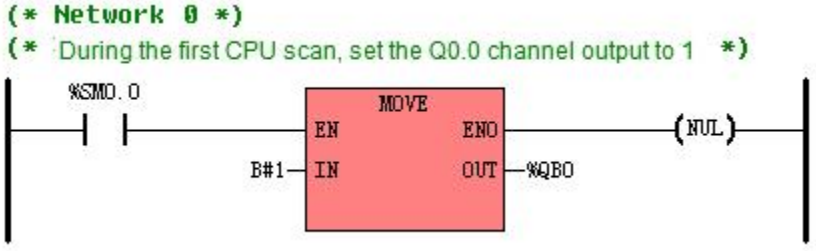
- ✓ For the variables in the I area and AI area, the forced value takes precedence over the external signal input value. If the user specifies a mandatory value, the mandatory value will take effect in the user program first.。
- ✓ For the variables in the Q area and AQ area, the execution result of the instruction is given priority during program execution, but the forced value will be given priority in the output task at the end of the scan cycle.

The user can execute the [Cancel Force] menu command to cancel the force state of a variable, or execute the [All Cancel Force] command to cancel the force state of all variables.

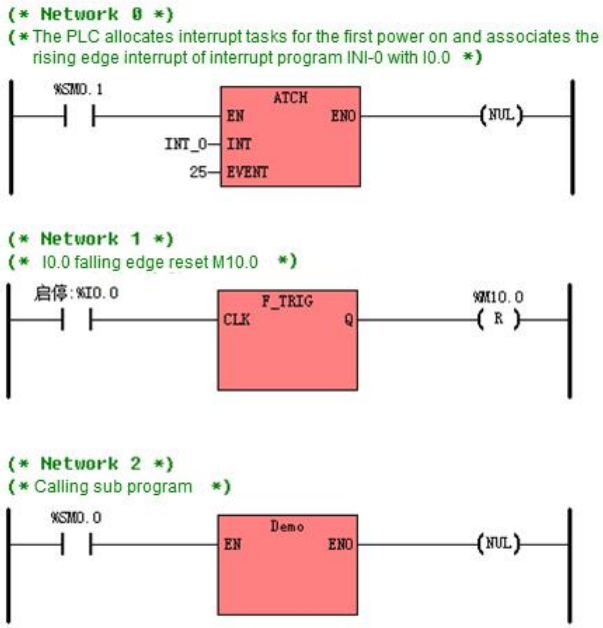
When the CPU is cold-started (re-powered), the forced state of all variables will be canceled.

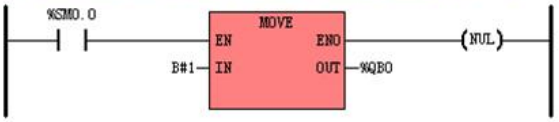
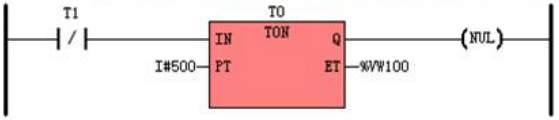
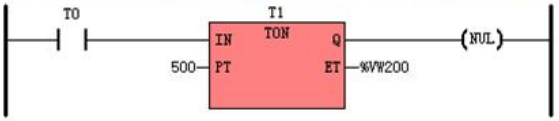
The above describes the common steps for users to create a new project and debug in the form of a summary, providing a guide for first-time users. For a more detailed description of each part, please refer to the relevant chapters.

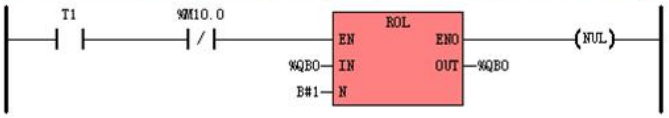
Note: For the constant in the program instruction, please indicate the relevant data type, otherwise an error will be reported when compiling. For example, the figure below needs to indicate that constant1 is a byte, so it needs to indicate B#1. For related instructions, please refer to 3.4 constant introduction.

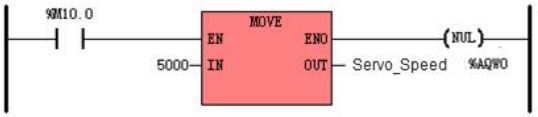


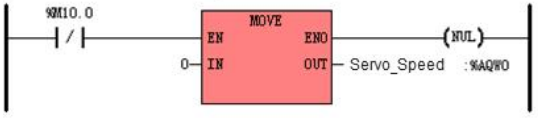
The complete program in the routine is as follows:

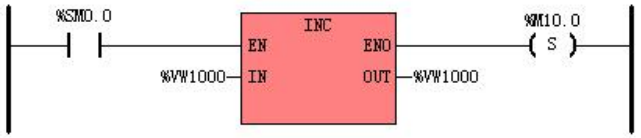
LD	MAIN: The main program is used to assign interrupt tasks and call subroutines. The rising edge of I0.0 generates an interrupt set and the falling edge of M10.0 resets, controls the output QB0 circular shift, and controls the analog output to set the servo speed. 

LD	Subroutine Demo: Self-made 1s pulse, every 1s QB0 shifts to the left, M10.0 controls whether to execute the shift, M10.0 is controlled by I0.0, and at the same time controls the analog output to control the servo start and stop. (* Network 0 *) (* During the first CPU scan, set the Q0.0 channel output to 1 *)  (* Network 1 *) (* T0 and T1 take turns starting each other, they are 1ms timers with a timing of 500ms *)  (* Network 2 *) (* T0 and T1 take turns starting each other, they are 1ms timers with a timing of 500ms *) 

<table>
<tr>
<td></td>
<td>
(* Network 3 *)
(* Under the control of T1 and M10.0 states, QB0 cycles left, moving one bit at a time *)


(* Network 4 *)
(* Start the servo at a speed of 500rpm/min
5000 corresponds to an internal value of 5V, * 1000 relationship *)


(* Network 5 *)
(* Stop servo speed to 0 *)


</td>
</tr>
<tr>
<td>LD</td>
<td>
Interrupt program INT_0: record the number of interrupts, and set M10.0 at the same time, which is used to control the start and stop of program functions.
(* Network 0 *)
(* Interrupt execution, vw1000 self recording interrupt count, set to M10.0 for
controlling start stop *)


</td>
</tr>
</table>

IL	MAIN: The main program is used to assign interrupt tasks and call subroutines. The rising edge of I0.0 generates an interrupt set and the falling edge of M10.0 resets, controls the output QB0 circular shift, and controls the analog output to set the servo speed. (* Network 0 *) (*The PLC allocates interrupt tasks for the first power on and associates the rising edge interrupt of interrupt program INT_0 with I0.0 *) LD %SM0.1 ATCH INT_0, 25 (* Network 1 *) (*I0.0 falling edge reset M10.0*) LD start_stop F_TRIG R %M10.0 (* Network 2 *) (*Calling sub program*) LD %SM0.0 CAL Demo

IL	Subroutine Demo: Self-made 1s pulse, every 1s QB0 shifts to the left, M10.0 controls whether to execute the shift, M10.0 is controlled by I0.0, and at the same time controls the analog output to control the servo start and stop.
	(* Network 0 *)
	(*During the first CPU scan, set the Q0.0 channel output to 1*)
	LD %SM0.0
	MOVE B#1, %QB0
	(* Network 1 *)
	(*T0 and T1 take turns starting each other, and they are 1ms timers with a timing of 500ms each *)
	LDN T1
	TON T0, I#500
	__CR_EQ_1
	MOVE T0, %VW100
	__CR_RESTORE
	ST %BR0.0
	(* Network 2 *)
	(*T0、T1 轮流相互启动, 他们是 1ms 定时器, 定时均为 500ms*)
	LD T0
	TON T1, 500
	__CR_EQ_1
	MOVE T1, %VW200
	__CR_RESTORE
	ST %BR0.0
	(* Network 3 *)
	(*T0 and T1 take turns starting each other, and they are 1ms timers with a timing of 500ms each *)
	LD T1
	ANDN %M10.0
	ROL %QB0, B#1
	(* Network 4 *)
	(*Start the servo and give a speed of 500rpm/min5000 as the internal value corresponding to 5V, *

	1000 relationship *) LD %M10.0 MOVE 5000,Servo_Speed
IL	(* Network 5 *) (*Stop servo speed to 0*) LDN %M10.0 MOVE 0, Servo_Speed Interrupt program INT_0: record the number of interrupts, and set M10.0 at the same time, which is used to control the start and stop of program functions. (* Network 0 *) (*Interrupt execution, VW1000 automatically records the number of interrupts, set to M10.0 for controlling start stop *) LD %SM0.0 INC %VW1000 S %M10.0

Appendix F Use of Expansion Modules

1. Overview

Each series of KPLC products provides corresponding CPU modules and various types of expansion modules.

The CPU module is the core of the PLC system, and is used to execute the main scanning tasks cyclically. Since the CPU module body integrates a communication port and a certain number of IO points, it can be used independently. If the system scale is large and the IO points of the CPU module cannot meet the system requirements, users can use the CPU module and expansion modules to flexibly combine small and medium-sized control systems suitable for various applications.

The following table lists the types and quantities of expansion modules that can be connected to various types of PLCs.

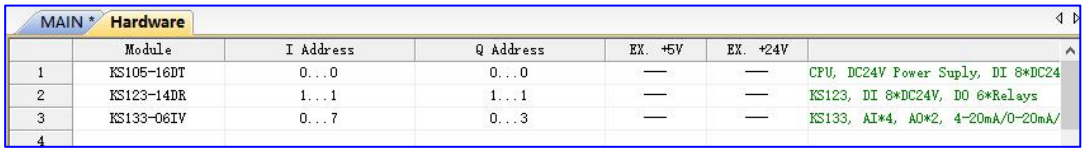
Series	Expansion module type	maximum quantity
K6 K5 MK KW KS(except KS105C1)	K6 and K5 expansion module	14 4(K504EX)、14(other CPU)
	KS extension module	14
K209M K2(except K209M) KS105C1	Not supported	0

Note: The quantity in the table is the maximum number of modules that can be connected to each series of PLC, and the actual number that can be connected to a single model is also limited by the size of the corresponding IO image area. For example, the length of the AI area of K508 is 64 bytes, and it supports up to 32 AI points, so it can only connect up to 8 K531-04IV modules. Please refer to chapter 3.6.4 Address Range of Memory Area for the memory area size of each series of PLC.

2. Introduction

1)Hardware Configuration

In practical applications, the user needs to first add the CPU and modules to be used in the module list according to the actual model and order in the [Hardware Configuration] of the user project and configure related parameters. Moreover, the model and order of the modules actually used must be consistent with those in the [Hardware Configuration] module list, otherwise the PLC will prompt a serious error and enter the STOP state!


MAIN * Hardware

	Module	I Address	Q Address	EX. +5V	EX. +24V	
1	KS105-16DT	0...0	0...0	—	—	CPU, DC24V Power Suply, DI 8*DC24
2	KS123-14DR	1...1	1...1	—	—	KS123, DI 8*DC24V, DO 6*Relays
3	KS133-06IV	0...7	0...3	—	—	KS133, AI*4, AO*2, 4-20mA/0-20mA/
4						

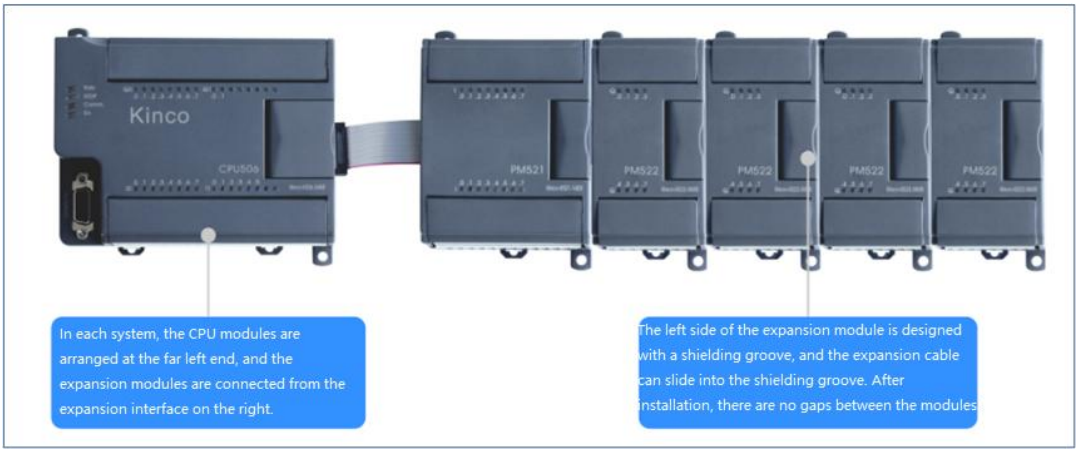
As shown in the figure above, the above three module models are configured in [Hardware Configuration] of the user project, and KS123-14DR is in the first position after the CPU module, and KS133-06IV is in the second position of the CPU module (that is, the position immediately after KS123-14DR). Therefore, the same product model and connection sequence must be followed when connecting the actual modules.

1)actual example of connection

- **K6 and K5 series**

K5/K6 series is the standard product series in KPLC. K6 is an upgraded version of K5, and maintains compatibility to facilitate maintenance and upgrades for users. The CPU and expansion modules of K6 and K5 can be used in any combination, that is, the CPU of K6 can be connected to the expansion module of K5, and the CPU of K5 can be connected to the expansion module of K6.

The expansion modules of K6/K5 are connected by special flat flexible cables, as shown in the figure below:



- **KS、KW series and K209M**

The CAN1 port of KS, KW series and K209M can be used as an expansion bus interface to connect to KS series expansion modules.

The KS expansion module adopts the RJ45 interface form, and the user can use a shielded Category 5e twisted-pair straight-connected network cable to connect the expansion module. The expansion module provides two interfaces, one in and one out, which is convenient for users to use when connecting multiple expansion modules. The specific connection can refer to the figure below:

- **MK series all-in-one machine**

Limited by cost and volume requirements, the MK series all-in-one provides CAN1 and two RS485 communication ports in a DB9 connector, so when the MK body is connected to the first KS expansion module, the user needs to make a communication cable by himself.

The pin definition and connection relationship of the CAN1 interface are shown in the table below. Note that the CAN_H, CAN_L, and CAN_GND pins need to be connected one by one.

MK body DB9	KS expansion module RJ45	Definition
Pin 7	Pin1	CAN_H
Pin2	Pin2	CAN_L
Pin5	Pin3	CAN_GND

Appendix G Use of Extended BD Board

The expansion BD board (Basic Unit Expansion Board) is a circuit board that provides a small number of IO channels or communication ports. It can be inserted into the BD board slot of the CPU module to increase the number of IO channels or communication ports for the system. The BD board can further enrich the functions of the CPU module, and compared with the expansion module, the price of the BD board is lower, thus providing users with a more cost-effective expansion method.

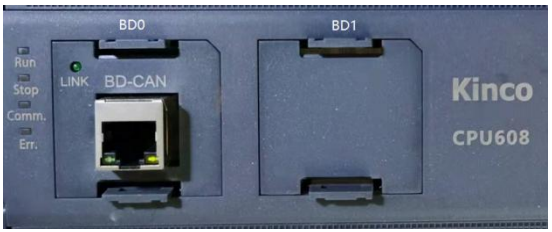
1. Overview

KPLC provides various types of BD boards. From the overall function, these BD boards can be divided into the following two categories:

- Communication type: Provides serial communication (RS232/RS485) or CAN communication ports.
- IO class: Only a certain number of IO channels such as DI, DO, AI, or AO are provided.

Communication type: serial communication (RS232/RS485) or CAN and other communication ports are provided.

IO type: only a certain number of IO channels such as DI, DO, AI or AO are provided.



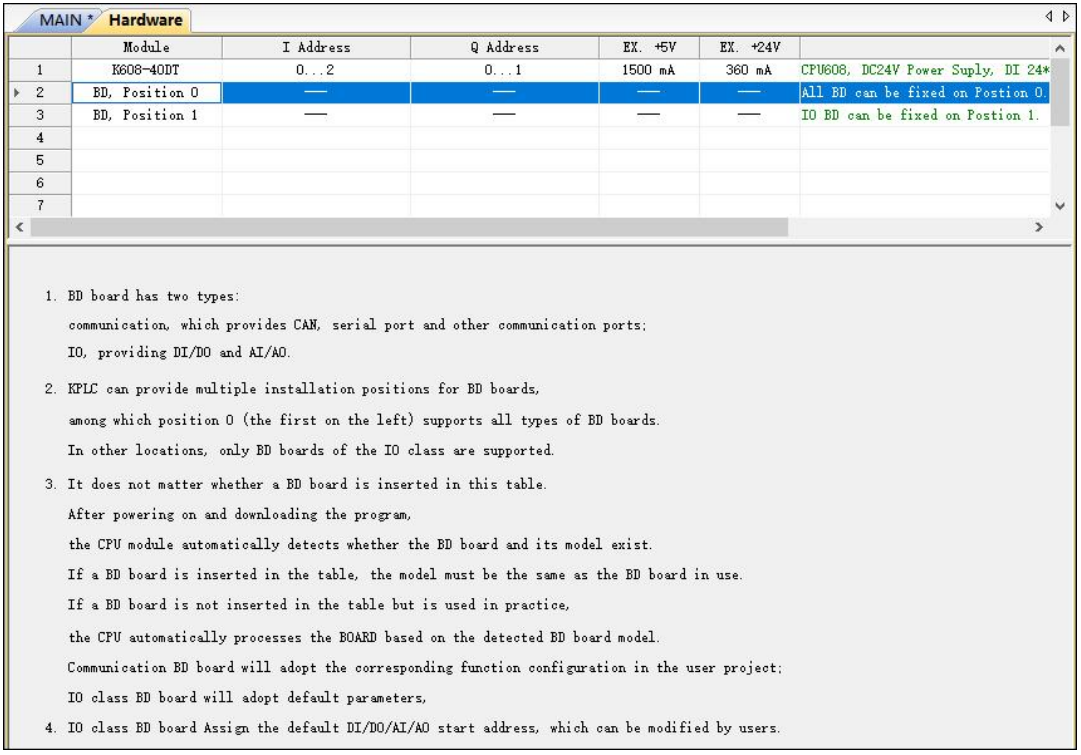
Each BD board slot has a unique position number, the rule is: starting from the left, the first slot number is BD0, the second slot number is BD1, and so on.

Note: 1) BD boards of communication type can only be installed in BD0; BD boards of IO type can be installed in all slots.

2) The user must first power off the CPU module before installing or removing the BD board, live operation is prohibited!

2. Use of BD board

The [Hardware Configuration] of the Kincobuilder programming software provides the function of adding and configuring a BD board, and when the user clicks a row of a BD board position in the table, the relevant description information will be displayed in the lower window. If the bank has not yet joined the BD board, it will display the overall instructions for using the BD board; if the bank has joined the BD board, it will display the instructions for using the corresponding model of the BD board. As shown below.



The CPU module will automatically detect whether a BD board and its model are installed in each BD slot after it is powered on and downloaded.

Therefore, in practical applications, it doesn't matter whether the user adds the BD board in the [Hardware Configuration] form of the program. If the user adds a BD board to the [Hardware Configuration] table of the program, the model configured in the table must be consistent with the actual installed BD model, otherwise the CPU module will report an error and enter the STOP state. If the user does not add the BD board in the [Hardware Configuration] table, but actually installs it on the CPU module, the CPU module will automatically process it according to the detected BD board model. For communication-type BD boards, the CPU module will adopt the parameters and function configuration of the corresponding communication port in the user project; for IO-type BD boards, the CPU module will adopt the default address and signal type, etc.

2.1 Function and parameter description of various BD boards

1)Serial communication port model KB6-2COM

This type of BD board provides 1 RS232 communication port and 1 RS485 communication port. The RS232 port number is PORT0, the RS485 port number is PORT3, and the communication parameters are all configured in the [Communication Settings] of the CPU in the [Hardware Configuration].

PORT0 (RS232 port) supports programming protocol, Modbus RTU slave station and free communication function. PORT3 (RS485 port) supports Modbus RTU master station, Modbus RTU slave station and free communication function.

2)CAN bus communication port model KB6-CAN

This type of BD board provides a CAN bus communication port, numbered as CAN2.

CAN2 supports Kinco motion control, CANOpen master station and CAN free communication function, among them, CANOpen master station and CAN free communication function can be used at the same time. For the use and configuration methods of various CAN communication functions, please refer to the previous article 10.5 Use of CAN bus.

3)Various IO types

KB6-4DI provides 4 transistor-type DI channels with a rated input voltage of DC24V, which can be used

as source or sink input at the same time.

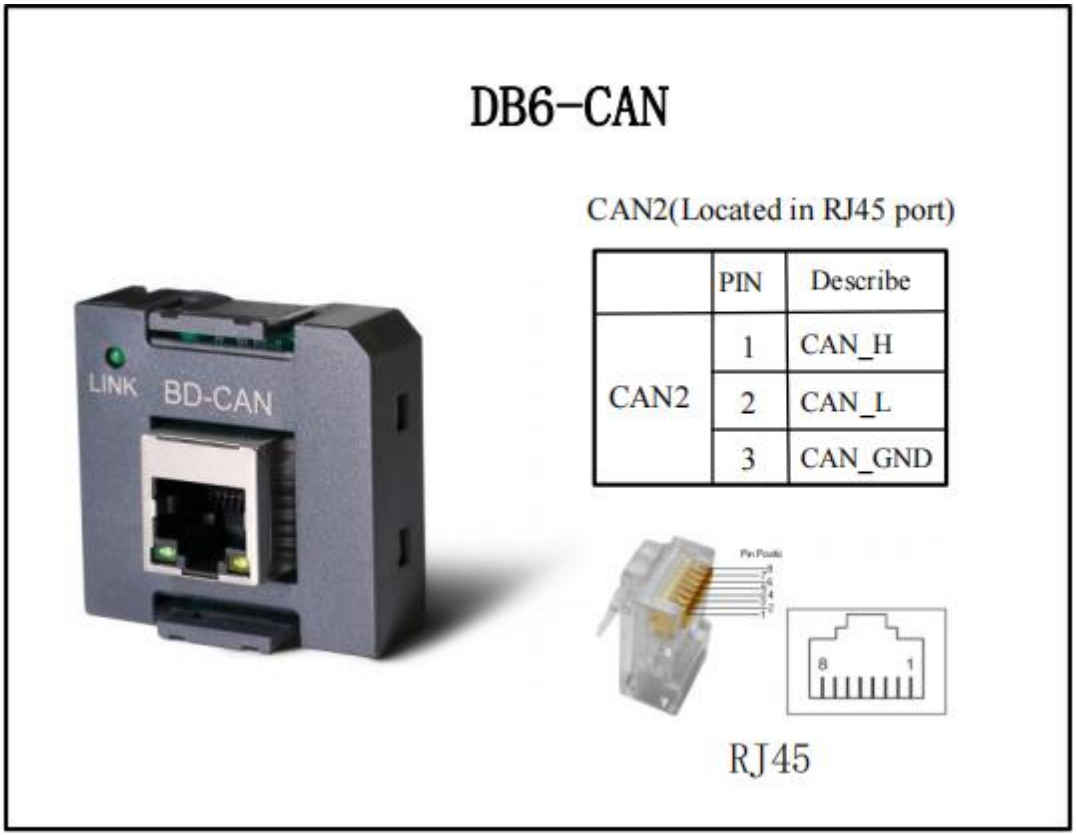
KB6-4DO provides 4 photorelay DO channels with a rated output voltage of DC24V, which can be used as source or sink output at the same time. The rated output current of each channel is 300mA, the maximum output current is 500mA, and it has the channel short-circuit protection function.

The default memory addresses and other channel information occupied by various IO channels on the BD board at different slot positions are shown in the table below. If a certain type of BD board is added to the [Hardware Configuration] table of the program, the user can modify its memory address and channel information.

Type	**BD board channel default memory address**		**Others**
	BDO	**BD1**	
DI	IB30	IB31	
DO	QB30	QB31	The shutdown hold value of all channels is 0
AI	AIW120	AIW124	The default signal form is 0-20mA, without filtering.
AO	AQW120	AQW124	The signal form defaults to 4-20mA, no shutdown hold.

For example, if the user installs KB-4DO at BD0, the default memory addresses of each channel are Q30.0, Q30.1, Q30.2 and Q30.3 in turn. If the user installs KB-4DI at BD1, the default memory addresses of each channel are I31.0, I31.1, I31.2 and I31.3.

2.2 BD board wiring diagram

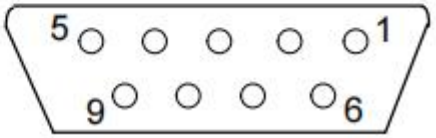


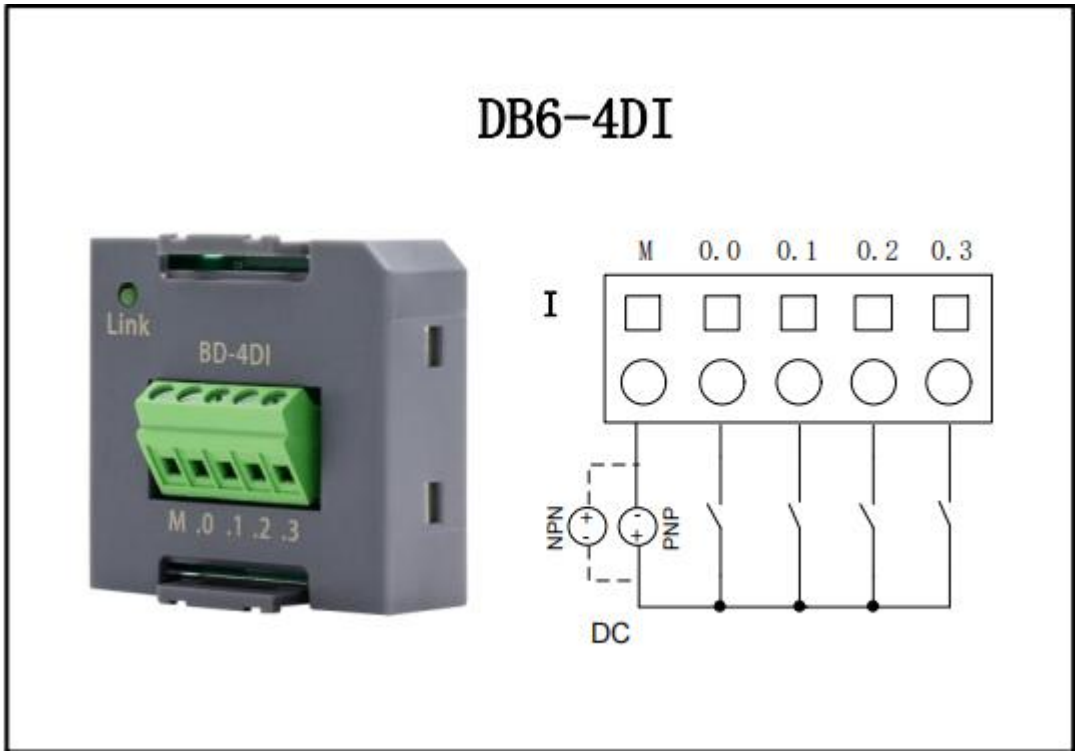
	PIN	Describe
CAN2	1	CAN_H
	2	CAN_L
	3	CAN_GND

DB6-2COM

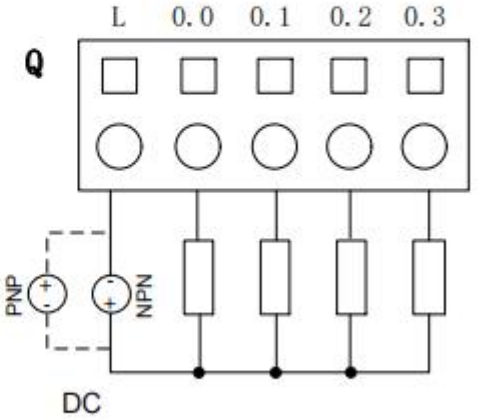
	No.	Defintion	Describe
PORT0 (RS232)	2	RxD	Receive
	3	TxD	Trasmit
	5	GND	Ground
PORT3 (RS485)	1	B	RS485-
	6	A	RS485+

PORT0/PORT3




DB6-4DI
Link
BD-4DI
M .0 .1 .2 .3
M 0.0 0.1 0.2 0.3
I
NPN
PNP
DC

DB6-4DO



Appendix H Usage of KS expansion module as ModBus RTU slave station

KS series expansion modules all provide a standard RS485 interface, and can be used as ModBus RTU slave stations for access by the master station. (You can check whether the middle 5 digits of the serial number ****19150**** are after this, and only after this is supported).

- KS expansion module factory default communication parameters:

Station No	1
baud rate	38400
Parity	None
data bit	8
stop bit	1

- Modify communication parameters:

UseKinco_Modbus_Module_Config_V10 (Please ask the manufacturer to obtain) Modify through RS485 port.

Note: After changing, the factory settings cannot be restored with one key, so it is required to remember the changed parameters!

Instructions:

Open the Kinco Remote Module Config (as shown in Figure 1), select the PC COM Port on the left and set the communication parameters to the same parameters as the currently connected module (for example, the default: station number is 1, baud rate is 38400, no parity). Select read on the right, prompting that the read is successful. The communication parameters of the currently connected modules are displayed in the right frame, change the station number (the maximum station number can be set to 128) or the baud rate or the verification method in the right frame. Click Modify, and it will prompt that the modification is successful, that is, the modification of the RS485 communication parameters of the currently connected module is completed.。

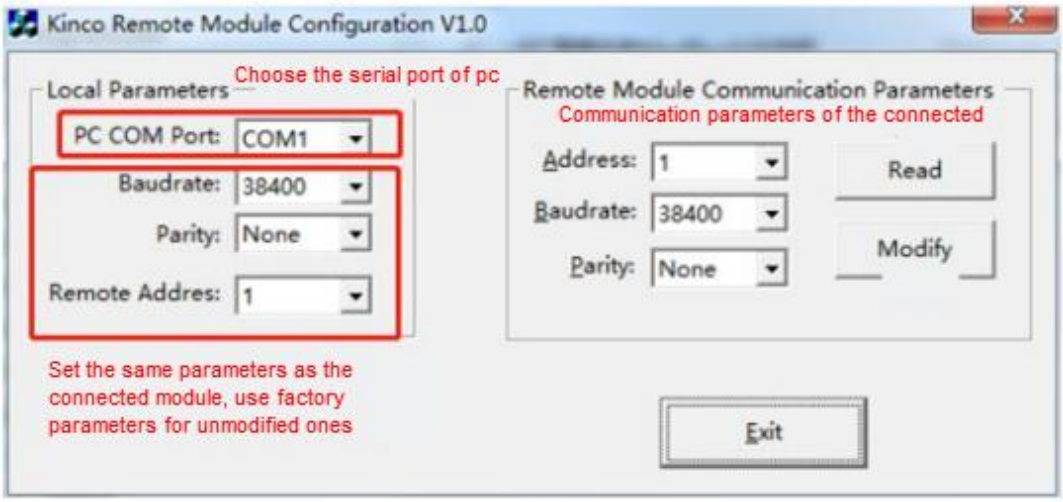


Figure 1: Kinco Remote Module

- Expansion module ModBus register memory area comparison:

Module Type	**Memory Area**	**Range**	**Type**	**Modbus Function code**	**Corresponding Modbus register number (decimal)**
KS121-16DX	I	I0.0-I1.7	DI	02	0-15
KS122-12XR	Q	Q0.0-Q1.3	DO	01, 05, 15	0-11
KS122-14DT	Q	Q0.0-Q1.5	DO	01, 05, 15	0-13
KS123-14DR	I	I0.0-I0.7	DI	02	0-7
	Q	Q0.0-Q0.5	DO	01, 05, 15	0-5
KS131-04RD	AI	AIW0-AIW6	AI	04	0-3
KS133-06IV	AI	AIW0-AIW6	AI	04	0-3
	AQ	AQW0-AQW2	AQ	03, 06, 16	0-1

- Simple application example:

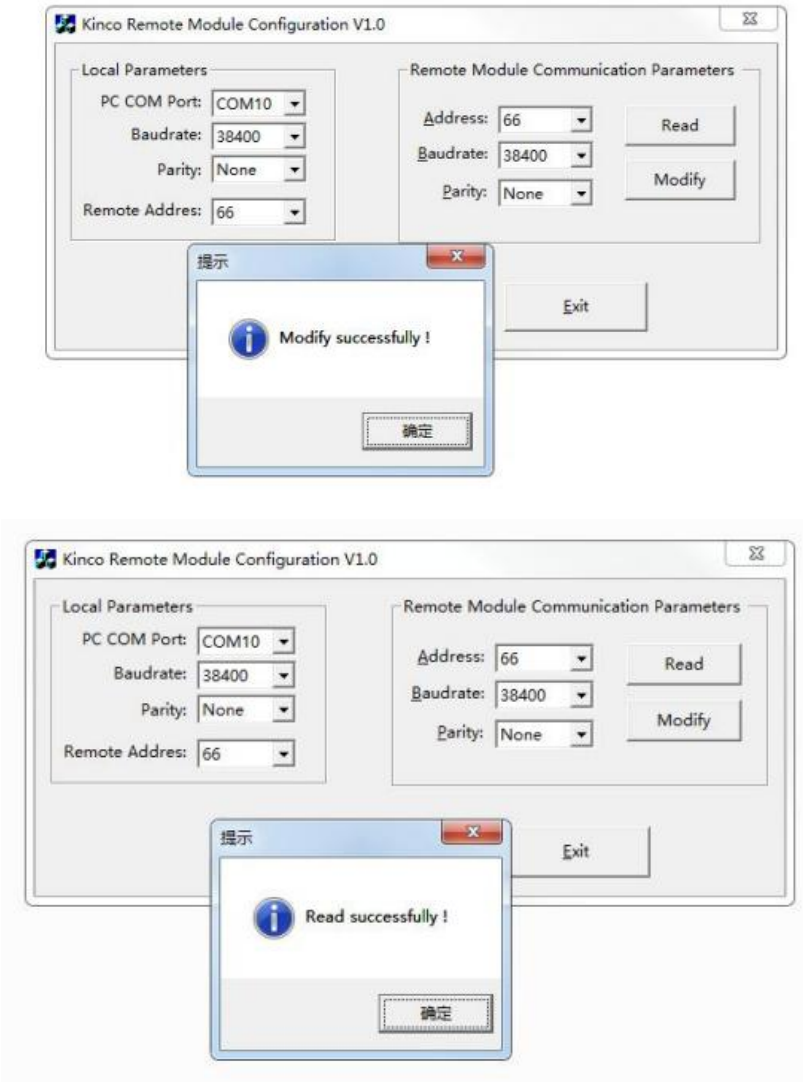
Introduction :

ModBus RTU master (K209M-56DT) accesses 3 expansion modules (KS121-16DX, KS122-14DT, KS133-06IV) via RS485.

Configuration:

K209M-56DT Port1 (parameters: baud rate 38400, no parity, 8 data bits, 1 stop bit)+1*KS122-14DT (station number is 1)+1*KS133-06IV (station number is 66);

Port2 (parameters: baud rate 9600, no parity, 8 data bits, 1 stop bit) + 1*KS121-16DX (station number is 128);



Example in Figure 2: Changing the station number of the module to 66

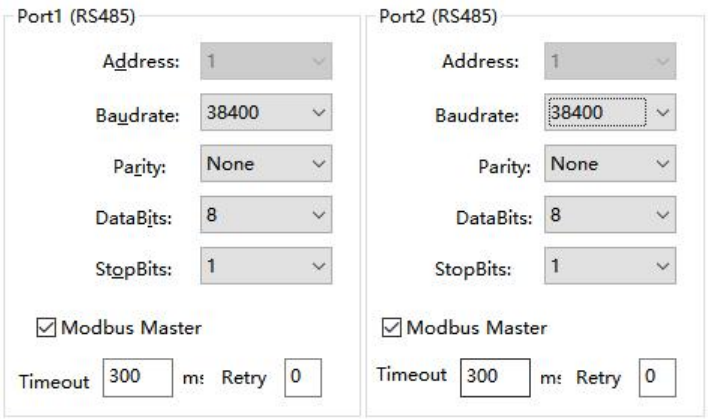


Figure 3: ModBus RTU master station communication parameter configuration

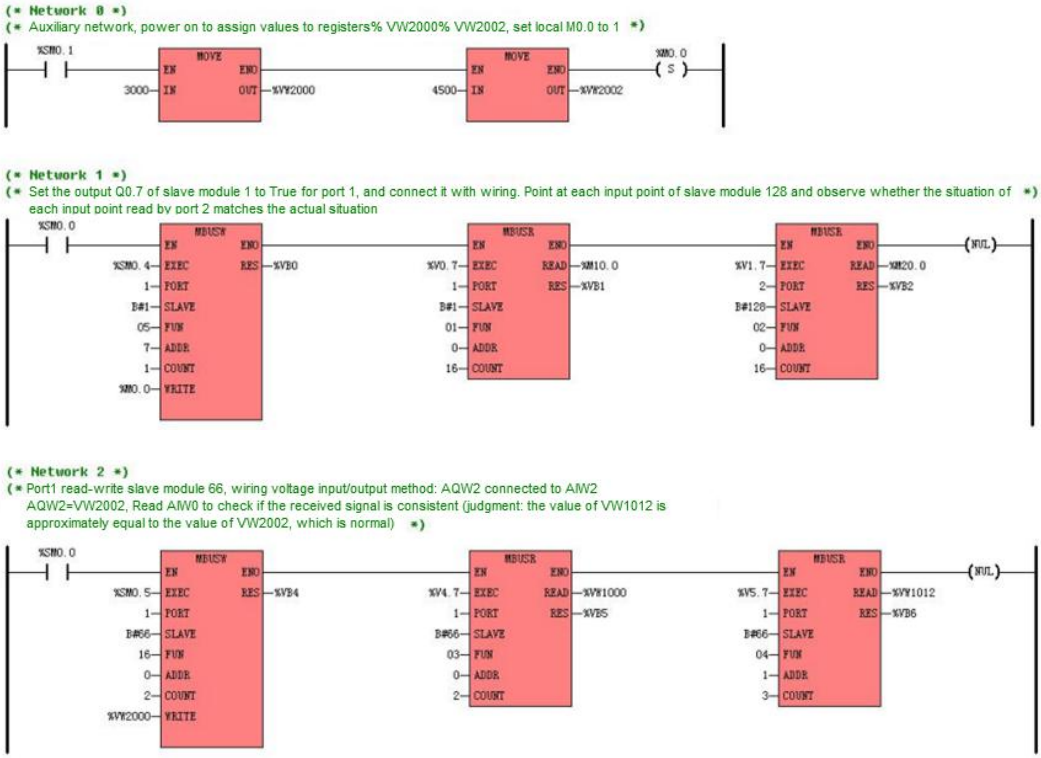


Figure 4: ModBus RTU master program configuration

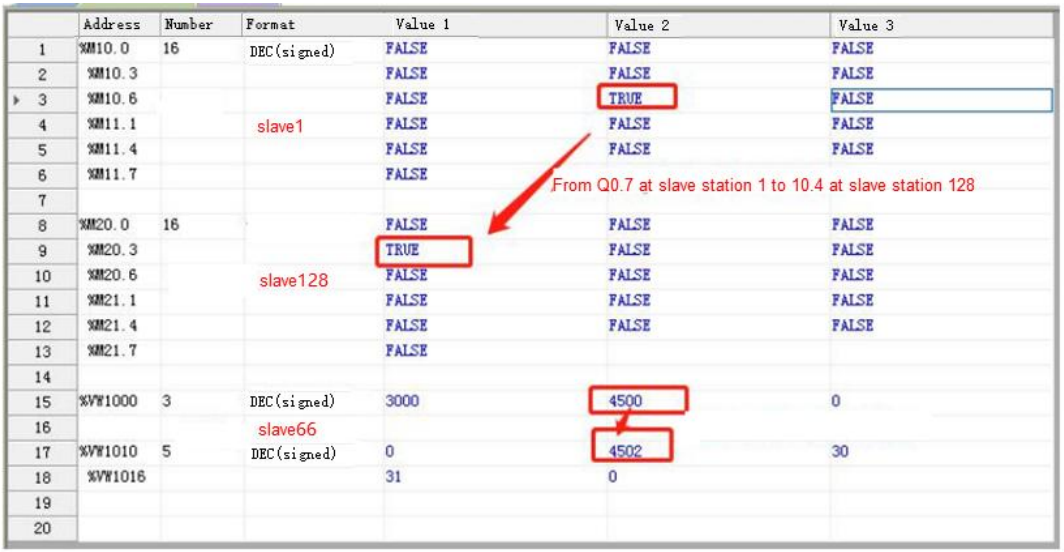
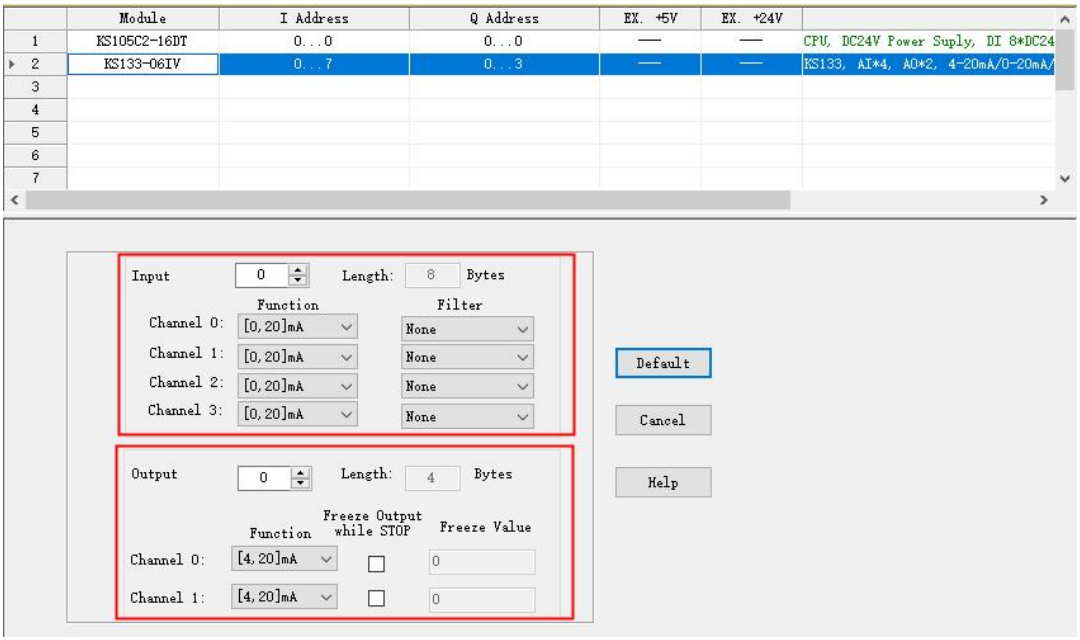


Figure 5: Monitoring Status

Note: When the KS expansion module is not connected to the CPU through the expansion port, the RUN light (green) flashes after power-on, which is normal.

In addition, because Kinco_Modbus_Module_Config_V10 can only modify communication parameters, other parameters of the module cannot be set, such as the signal form and filtering method of the analog module. Therefore, it is necessary to connect the module through the expansion bus through the CPU module, set the relevant parameters in advance and save them through the EX_ADDR command, and then connect through RS485.

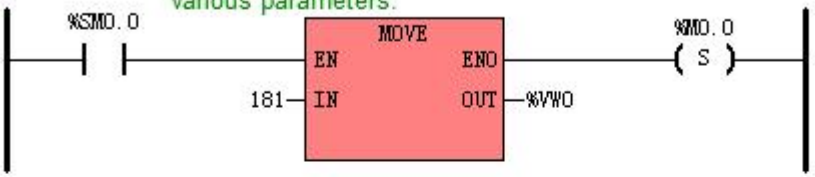
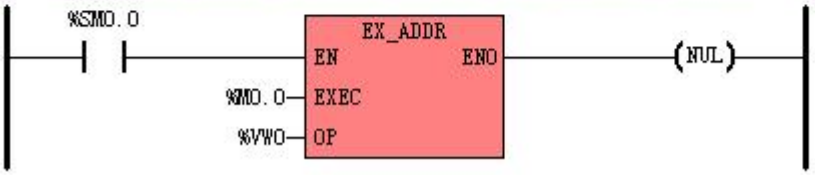
The hardware configuration is described by taking KS105C2-16DT with module KS133-06IV as an example:



As shown in the figure, assuming that the signal form and filtering method of the module need to be modified as above, write a program to save the parameters.

The procedure is as follows:

(* Network 0 *)
(* 181 --- Command all extension modules to save their own IDs and various parameters. *)



(* Network 1 *)
(* Execute the EX-ADDR command to save module parameters *)