

xLadder Programming guide

RIEV/TECH
EXPERT FOR MICRO-AUTOMATION SOLUTIONS

MICRO PLC User Manual

Applied to SR & PR series

INTRODUCTION
GETTING STARTED
INSTALLATION AND WIRING
PROGRAMMING PLC
CONFIGURING & SOFTWARE
APPLICATIONS
TECHNICAL DATA



Ver. 2.0.0.7

Document Revisions

| Date | Version Number | Document Changes |
|------------|----------------|--|
| 25/12/2020 | 2.0.0.3 | Add new instructions description for Ethernet PLC (Chapter 10.12~10.16) |
| 26/05/2021 | 2.0.0.4 | Modify the expansion module I/O, AI, AQ address addressing (Chapter 10.3) |
| 01/03/2022 | 2.0.0.5 | SMB34-SMB35 Time Interval Registers for Timed Interrupts (Chapter 4.1) |
| 24/03/2022 | 2.0.0.6 | NEW Modbus Address Assignment Table (Page 304) |
| 24/12/2024 | 2.0.0.7 | xladder also supports SR-12 series and adds related content. Add some new content in Chapter 10. |
| | | |
| | | |
| | | |
| | | |

| | |
|--|----|
| Introduction | 10 |
| Valid range of this manual | 10 |
| Safety Guideline..... | 10 |
| Qualified Personnel..... | 11 |
| Prescribed Usage..... | 12 |
| Warning..... | 12 |
| Trademarks | 12 |
| Copyright Rievtech 2016 all rights reserved..... | 12 |
| Disclaim of Liability | 12 |
| Additional support | 12 |
| 1. What is Rievtech Micro PLC? | 13 |
| 1.1 Overview | 13 |
| 1.2 Highlight feature | 13 |
| 1.3 Some of the things Rievtech Micro PLC can do for you?..... | 14 |
| 1.4 Rievtech devices: | 14 |
| Rievtech Basic is available in two voltage classes..... | 14 |
| Expansion modules | 14 |
| Communication cable and module..... | 15 |
| 2. Hardware introduction..... | 16 |
| 2.1 Naming Rules of PR Series PLC..... | 16 |
| 2.2 Hardware model selection..... | 18 |
| 2.3 Structure & dimension..... | 1 |
| 3 .Installing/removing Rievtech Micro PLC | 4 |
| Dimensions | 4 |
| 3.1 DIN rail mounting | 5 |
| 3.2 Wall-mounting..... | 6 |
| 3.3 wiring Rievtech Micro PLC | 7 |
| 3.4 Connecting the power supply..... | 7 |
| 3.4.1 Connecting Rievtech micro PLC inputs | 8 |
| 3.4.2 Connecting Outputs | 10 |
| 3.4.3 Communication port instructions: | 12 |
| 4.Quick reference manual | 17 |
| 4.1 Special memory-SM area:..... | 17 |
| 4.2Interrupt Events: | 18 |
| 4.3 High speed counter: | 18 |
| XLadder direction for use..... | 20 |
| 5.The detailed annotation of operation interface..... | 20 |
| 5.1The main menu | 20 |
| 5.1.1 File..... | 20 |
| 5.1.2 Edit | 21 |
| 5.1.3 View | 22 |
| 5.1.4 PLC..... | 23 |
| 5.1.5 Debug..... | 24 |
| 5.1.6 SMS..... | 24 |

| | |
|---|----|
| 5.1.7 Help | 26 |
| 5.2 Toolbar | 27 |
| 5.3 Instruction tree | 30 |
| 5.3.1 Project | 31 |
| 5.3.2 Data block | 31 |
| 5.3.3 System block | 32 |
| 5.3.4 Program block | 38 |
| 5.3.5 Function symbol | 39 |
| 5.3.6 Variable symbol | 39 |
| 5.3.7 Status chart | 39 |
| 5.3.8 Cross reference | 40 |
| 5.3.9 Communication | 40 |
| 5.3.10 Instructions | 42 |
| 5.3.11 Libraries | 42 |
| 5.3.12 The program editor | 43 |
| 5.3.13 Status chart, information output | 44 |
| 5.4 Programming concepts | 45 |
| 5.4.1 How the program works | 45 |
| 5.4.2 Addressing overview | 45 |
| 5.4.3 How to organize the program | 47 |
| 5.5 How to enter the ladder logic program | 49 |
| 5.5.1 How to build a new project | 49 |
| 5.5.2 Ladder logic element and its working principle | 50 |
| 5.5.3 Network rules for series and parallel in LAD | 51 |
| 5.5.4 How to input commands in LAD | 51 |
| 5.5.5 How to enter the address in LAD | 53 |
| 5.5.6 How to edit program elements in LAD | 53 |
| 5.5.7 How to use find / replace | 55 |
| 5.5.8 How to display errors in LAD in the program editor | 56 |
| 5.5.9 How to compile in LAD | 57 |
| 5.5.10 How to save the project | 57 |
| 5.6 How to set up a communication and download program | 58 |
| 5.6.1 Communication settings | 58 |
| 5.6.2 Download program | 59 |
| 5.6.3 How to correct compilation errors and download errors | 61 |
| 5.7 How to monitor the program | 62 |
| 5.8 PLC operation and options | 65 |
| 6. xLadder instructions descriptions | 67 |
| 6.1 Bit logic | 67 |
| 6.1.1 Normally open and normally closed | 67 |
| 6.1.2 Normally open immediate and normally closed immediate | 68 |
| 6.1.3 NOT Reverse instruction | 69 |
| 6.1.4 Rising edge and falling edge | 69 |
| 6.1.5 Output | 70 |

| | |
|---|-----|
| 6.1.6 Output immediate ----- | 71 |
| 6.1.7 Set and reset ----- | 71 |
| 6.1.8 Set immediate and reset immediate ----- | 72 |
| 6.1.9 SR instruction ----- | 72 |
| 6.1.10 RS instruction ----- | 73 |
| 6.1.11 NOP instruction ----- | 74 |
| 6.2 Clock instruction ----- | 75 |
| 6.2.1 Read and set the real time clock ----- | 75 |
| 6.3 Communication ----- | 76 |
| 6.3.1 Get port address ----- | 76 |
| 6.3.2 Set port address ----- | 76 |
| 6.4 Compare ----- | 77 |
| 6.4.1 Byte compare ----- | 78 |
| 6.4.2 Integer comparison ----- | 79 |
| 6.4.3 Double integer comparison ----- | 80 |
| 6.4.4 Real number comparison ----- | 81 |
| 6.4.5 String comparison ----- | 82 |
| 6.5 Convert ----- | 83 |
| 6.5.1 Byte to integer ----- | 83 |
| 6.5.2 Integer to byte ----- | 84 |
| 6.5.3 Integer to double integer ----- | 84 |
| 6.5.4 Integer to string ----- | 84 |
| 6.5.5 Double integer to integer ----- | 86 |
| 6.5.6 Double integer to real number ----- | 86 |
| 6.5.7 Double integer to string ----- | 87 |
| 6.5.8 BCD to integer, integer to BCD conversion ----- | 88 |
| 6.5.9 ROUND ----- | 89 |
| 6.5.10 TRUNC ----- | 90 |
| 6.5.11 Real number to string ----- | 90 |
| 6.5.12 Integer to ASCII code ----- | 92 |
| 6.5.13 Double integer to ASCII code ----- | 94 |
| 6.5.14 Real number to ASCII code ----- | 95 |
| 6.5.15 ATH&HTA ----- | 97 |
| 6.5.16 String to integer ----- | 98 |
| 6.5.17 String to double integer ----- | 100 |
| 6.5.18 String to real number ----- | 102 |
| 6.5.19 DECO ----- | 103 |
| 6.5.20 ENCO ----- | 104 |
| 6.5.21 Seven segment code ----- | 105 |
| 6.6 Counter ----- | 106 |
| 6.6.1 CTU ----- | 106 |
| 6.6.2 CTD ----- | 107 |
| 6.6.3 CTUD ----- | 107 |
| 6.7 Floating point calculation ----- | 108 |

| | |
|---------------------------------------|-----|
| 6.7.1 ADD-R&SUB-R----- | 109 |
| 6.7.2 MUL - R&DIV - R----- | 110 |
| 6.7.3 SQRT----- | 111 |
| 6.7.4 SIN ----- | 112 |
| 6.7.5 COS----- | 113 |
| 6.7.6 TAN ----- | 113 |
| 6.7.7 LN ----- | 114 |
| 6.7.8 EXP----- | 115 |
| 6.7.9 PID ----- | 116 |
| 6.8 Integer operations ----- | 119 |
| 6.8.1 ADD-I&SUB-I ----- | 119 |
| 6.8.2 ADD- DI & SUB- DI----- | 120 |
| 6.8.3 MUL & DIV----- | 121 |
| 6.8.4 MUL -I & DIV-I----- | 122 |
| 6.8.5 MUL -DI & DIV -DI----- | 124 |
| 6.8.6 INC-B & DEC-B ----- | 125 |
| 6.8.7 INC-W & DEC-W ----- | 126 |
| 6.8.8 INC -DW & DEC -DW----- | 127 |
| 6.9 Interrupt----- | 128 |
| 6.9.1 ENI & DISI ----- | 128 |
| 6.9.2 RETI instruction----- | 129 |
| 6.9.3 ATCH----- | 130 |
| 6.9.4 DTCH ----- | 133 |
| 6.9.5 Clear interrupt event ----- | 134 |
| 6.10 Logic operation ----- | 135 |
| 6.10.1 INV -B----- | 135 |
| 6.10.2 INV -W ----- | 136 |
| 6.10.3 INV -DW----- | 137 |
| 6.10.4 WAND-B、WOR -B、WXOR -B----- | 138 |
| 6.10.5 WAND-W、WOR -W、WXOR -W ----- | 139 |
| 6.10.6 WAND- DW、WOR -DW、WXOR -DW----- | 140 |
| 6.11 Move ----- | 141 |
| 6.11.1 Byte move----- | 141 |
| 6.11.2 Word move----- | 142 |
| 6.11.3 Double word move----- | 143 |
| 6.11.4 Real number move ----- | 144 |
| 6.11.5 BLKMOV -B----- | 145 |
| 6.11.6 BLKMOV -W----- | 146 |
| 6.11.7 BLKMOV -D----- | 147 |
| 6.11.8 SWAP ----- | 148 |
| 6.11.9 MOV -BIR ----- | 149 |
| 6.11.10 MOV -BIW ----- | 149 |
| 6.12 Program control----- | 150 |
| 6.12.1 FOR、NEXT----- | 150 |

| | |
|---|-----|
| 6.12.2 Jump to label and label ----- | 152 |
| 6.12.3 Sequence control relay----- | 153 |
| 6.12.4 Return from subroutine----- | 155 |
| 6.12.5 Conditional end----- | 156 |
| 6.12.6 STOP ----- | 157 |
| 6.12.7 Watchdog Reset ----- | 158 |
| 6.12.8 Diagnosis LED----- | 159 |
| 6.13 Shift cycle----- | 160 |
| 6.13.1 SHR -B & SHL -B ----- | 160 |
| 6.13.2 SHR -W & SHL -W ----- | 162 |
| 6.13.3 SHR -DW & SHL -DW ----- | 163 |
| 6.13.4 ROR -B & ROL -B ----- | 164 |
| 6.13.5 ROR -W & ROL -W ----- | 165 |
| 6.13.6 ROR -DW & ROL -DW ----- | 166 |
| 6.13.7 SHRB----- | 167 |
| 6.14 Character string----- | 168 |
| 6.14.1 String length ----- | 168 |
| 6.14.2 Copy string ----- | 169 |
| 6.14.3 SSTR-CPY----- | 170 |
| 6.14.4 String catenate ----- | 171 |
| 6.14.5 STR -FIND----- | 172 |
| 6.14.6 Look for the first character in the string ----- | 173 |
| 6.15 Table----- | 174 |
| 6.15.1 Last in first out ----- | 174 |
| 6.15.2 FIFO ----- | 176 |
| 6.15.3 Add to table----- | 178 |
| 6.15.4 Memory fill----- | 180 |
| 6.15.5 Table Find----- | 181 |
| 6.16 Timer ----- | 183 |
| 6.16.1 Switch on delay timer ----- | 183 |
| 6.16.2 TONR ----- | 185 |
| 6.16.3 Disconnect delay timer----- | 186 |
| 6.16.4 Start time interval ----- | 187 |
| 6.16.5 Calculation interval time----- | 188 |
| 6.17 Pulse train output ----- | 189 |
| 6.17.1 Pulse output ----- | 189 |
| 6.17.2 Pulse width modulation----- | 190 |
| 6.18 Subroutine ----- | 191 |
| 6.18.1 Using subroutine ----- | 191 |
| 6.18.2 Using parameters to call subroutine ----- | 192 |
| 6.18.3 How to set up a subroutine ----- | 193 |
| 6.18.4 How to call a subroutine----- | 194 |
| 7.PLC storage area----- | 196 |
| 7.1 Storage area types and properties ----- | 196 |

| | |
|---|-----|
| 7.2 Direct and indirect addressing ----- | 197 |
| 7.3 Bit, byte, word and double word access----- | 200 |
| 7.4 Memory address range----- | 200 |
| 7.5 Data type----- | 201 |
| 7.6 Constant----- | 202 |
| 8.Assignment and function of SM special storage area ----- | 203 |
| 9.xladder communication ----- | 205 |
| 9.1 PLC basic introduction of network communication ----- | 205 |
| 9.2 Rievtech PLC communication ----- | 207 |
| 9.3 Optimize network performance ----- | 214 |
| 10.Additional chapter ----- | 214 |
| 10.1 How to switch PLC mode ----- | 214 |
| 10.2 How to obtain the firmware version number of the PLC----- | 221 |
| 10.3 Value range of analog quantity:----- | 223 |
| 10.4 Extension module address ----- | 224 |
| 10.4.1 Digital Input addressing----- | 224 |
| 10.4.2 Digital Output addressing ----- | 225 |
| 10.4.3 Analog Input addressing ----- | 227 |
| 10.4.4 Analog Output addressing----- | 228 |
| 10.5 Set extension module address with a DIP switch ----- | 230 |
| 10.6 Set the analog sensor type for AI and AQ on xLadder ----- | 231 |
| 10.6.1 AI: Set to voltage, current, PT100, PT1000 mode ----- | 231 |
| 10.6.2 AQ: Set to voltage or current mode ----- | 232 |
| 10.7 Additional instructions ----- | 234 |
| 10.7.1 LCD related instructions----- | 234 |
| 10.7.2 CAN, serial port initialization instructions ----- | 242 |
| 10.8 Example of serial port free port communication ----- | 244 |
| 10.9 Example of CAN free port ----- | 246 |
| 10.10 MODBUS communication master program ----- | 248 |
| 10.11 The example of using PID instruction ----- | 249 |
| 10.12 Read / Write Program via Ethernet ----- | 252 |
| 10.12.1 PR Ethernet Series ----- | 253 |
| 10.12.2 SR-12 Series ----- | 260 |
| 10.13 Modbus block (New block for Ethernet series PLC)----- | 265 |
| 10.14 MQTT (New block for Ethernet series PLC) ----- | 268 |
| 10.14.1 Connect to MQTT Server ----- | 268 |
| 10.14.2 Publish with MQTT ----- | 271 |
| 10.14.3 Subscribe with MQTT ----- | 276 |
| 10.15 Memory Read & Memory Write (New block for Ethernet series PLC)----- | 278 |
| 10.15.1 Memory Write ----- | 278 |
| 10.15.2 Memory Read ----- | 281 |
| 10.16 GPRS_CONN & GPRS_SEND (New block for Ethernet series PLC) ----- | 284 |
| 10.16.1 GPRS_CONN ----- | 284 |
| 10.16.2 GPRS_SEND ----- | 285 |

| | |
|--|-----|
| 10.17 SMS_IN & SMS_OUT (New block for Ethernet series PLC) ----- | 288 |
| 10.17.1 SMS_IN ----- | 289 |
| 10.17.2 SMS_OUT ----- | 292 |
| 10.18 EMAIL_POST (New block for Ethernet series PLC) ----- | 295 |
| 10.18.1 Set email params ----- | 295 |
| 10.18.2 EMAIL_POST ----- | 296 |
| 10.19 Bacnet Protocol ----- | 298 |
| 10.20 Libraries files function ----- | 305 |
| 10.21 Offline simulation ----- | 307 |
| 10.22 C language subroutine ----- | 310 |
| MODBUS ADDRESS ----- | 314 |

Introduction

Congratulations with your Rievtech Micro PLC provided by Rievtech Electronic Co., Ltd.

The Rievtech Micro PLC is a compact and expandable CPU replacing mini-PLCs, multiple timers, relays and counters.

The Rievtech Micro PLC perfectly fits in the space between timing relays and low-end PLCs. Each CPU incorporates not only a real-time clock and calendar, but also provides support for optional expansion I/O modules to enhance control and monitoring applications. Data adjustments can easily be performed via the keypad, the LCD display, or through the xLadder soft. DIN-rail and panel-mounted options are both available, offering full flexibility to the various installation needs of your application.

The Rievtech Micro PLC is available in 110V-240V AC or 12V-24V DC versions, making it the ideal solution for relay replacement, or simple control applications as building and parking lot lighting, managing automatic lighting, access control, watering systems, pump control, ventilation systems, home automation and a wide field of other applications demanding low cost to be a primary design issue.

We strongly recommended taking the time to read this manual, before putting the Rievtech Micro PLC to work. Installation, programming and use of the unit are detailed in this manual. The feature-rich Rievtech Micro PLC provides a for off-line operation mode, allowing full configuration and testing prior to in-field service commissioning. In reviewing this manual you will discover many additional advantageous product properties, it will greatly simplify and optimize the use of your Rievtech Micro PLC.

Valid range of this manual

The manual applies to devices of PR and SR series PLC.

Safety Guideline

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol; notices referring to property damage only have no safety alert symbol. The notices shown below are graded according to the degree of danger.

Caution

Indicates that death or severe personal injury may result if proper precautions are not taken

Caution

With a safety alert symbol indicates that minor personal injury can result if proper precautions are not taken.

Caution

Without a safety alert symbol indicates that property damage can result if proper precautions are not taken.

Attention

Indicate that an unintended result or situation can occur if the corresponding notice is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by qualified personnel. Within the context of the safety notices in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards. Please read the complete operating instructions before installation and commissioning.

Rievtech does not accept any liability for possible damage to persons, buildings or machines, which occur due to incorrect use or from not following the details.

Prescribed Usage

Note the following:

Warning

This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Rievtech. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

All names identified by Rievtech are registered trademarks of the Rievtech. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Copyright Rievtech 2016 all rights reserved

The distribution and duplication of this document or the utilization and transmission of its contents are not permitted without express written permission. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Disclaim of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Additional support

We take pride in answering your question as soon as we can:

Please consult our website at www.rievtech.com for your closest point of contact or email us at sales@rieitech.com

1. What is Rievtech Micro PLC?

1.1 Overview

Rievtech Micro PLC is a universal logic module made by Rievtech.

Rievtech Micro PLC, a compact, expandable CPU that can replace mini-PLC, multiple timers, relays and counters, Splitting the difference between a timing relay and a low-end PLC, Each CPU has a real-time clock and calendar, and supports optional expansion I/O modules to enhance your control and monitoring applications. Data adjustments can be done via the on-board keypad and LCD display, or with xLadder soft. It can be either DIN-rail or panel mounted, depending upon the needs of your application, and it is available in 120V-240V ac as well as 12V-24V dc versions, and it is the ideal solution for relay replacement applications, simple control applications such as building and parking lot lighting, managing automatic lighting, access control, watering systems, pump control, or ventilation systems in factory, and home automation and applications in which cost is a primary design issue.

1.2 Highlight feature

- Multiple value display and input via keypad and LCD display.
- Unique ladder diagram, improves your programming efficiency.
- Standard Modbus RTU/ASCII/TCP communication protocol supported.
- It's optional for Rievtech Micro PLC to act as slave or master in certain Modbus communication network. (Easy connect to other factory touch screen by RS232 cable, RS485 module)
- Support MODBUS RTU/TCP, Bacnet, MQTT, Free protocol
- Expandable up to 16 linked IO expansion modules reaching 282 I/O points in maximum
- Optional RS232, RS485 connectivity
- Multiple channels analog inputs available with DC 0-10V signal, PT100-PT1000 signal & 0/4... 20mA.
- Default Real Time Clock (RTC)
- Backup at Real Time Clock (RTC) at 25 °C:20 days
- 4 channels high-speed counting
- Pre-configured standard functions, e.g. on/ off-delays, pulse relay and softkey
- 2 PWM channels (10KHz in maximum)
- Retentive memory capability
- RS232 and USB communication download cable with photo-electricity isolation
- Support ladder diagram programming

- Mounting via modular 35mm DIN rail or screw fixed mounting plate
- On-line monitor capability (Free charge SCADA for all series)
- Datalogging
- Kinds of analog signals process capacity (DC 0... 10V ,0/4... 20 mA, PT100-PT1000 probe inputs and DC 0... 10V and 0/4... 20mA outputs)
- Low cost

1.3 Some of the things Rievtech Micro PLC can do for you?

The Rievtech Micro PLC provides solutions for commercial, industrial, building and domestic applications such as lighting, pumping, ventilation, shutter operations or in switching cabinets. The application field is widespread and these are just a few to mention.

Using the RS485 bus and Ethernet connectivity allows the user to realize various extensive (real-time) monitoring and control applications.

Special versions without operator panel and display unit are available for series production applications in small machine, installation and cabinet building environments to further slash cost.

1.4 Rievtech devices:

Rievtech Basic is available in two voltage classes :

*Classes 1: DC12-24V: i.e.: PR-6DC Series, SR-12DC series, PR-12DC series, PR-18DC series, PR-24DC series, and PR-DC ethernet series.

*Classes2: AC110-240V: i.e.: PR-6AC Series, SR-12AC series, PR-12AC series, PR-18AC series, PR-24AC series and PR-AC ethernet series.

Expansion modules :

PR-E/SR-E

* Digital expansion modules are available for operation with 12...24V DC, and 110.. .240 V AC, and are equipped with eight inputs and eight outputs.

* Analog expansion modules are available for operation with 12...24 V DC and are equipped with 6 digital and 4 analog inputs.

Communication cable and module :

- **RS232 communication cable (Model: RS232 Cable)**

It is kind of universal cable with photoelectricity isolation which can be directly connected to standard 9-pin port of PC, also kind of interface module which can enable user's program to be downloaded into the PLC through xLadder-soft for running. It also is the connection cable between CPU and third-party device with the RS232 port (just like HMI) in modbus communication system.

- **USB communication cable (Model: USB Cable).**

It is kind of communication cable with photoelectricity isolation through which PC with USB port only can be connected to the PLC, moreover, it has same features as RS232 Cable, so it is quite convenient for user whose computer has no standard serial port.

- **PRO-RS485 cable (Model: PRO-RS485).**

It is kind of converter cable with photoelectricity isolation to make the program port serves as RS485 port.

- **RS485 module (Model: PR-RS485 / SR-RS485)**

RS485 expansion module, connected to the expansion port of PLC CPU. Adds one RS485 channel to PLC CPU.

Communication / Network

Rievtech PLC offers different ways to communicate within the system.

RS485 port

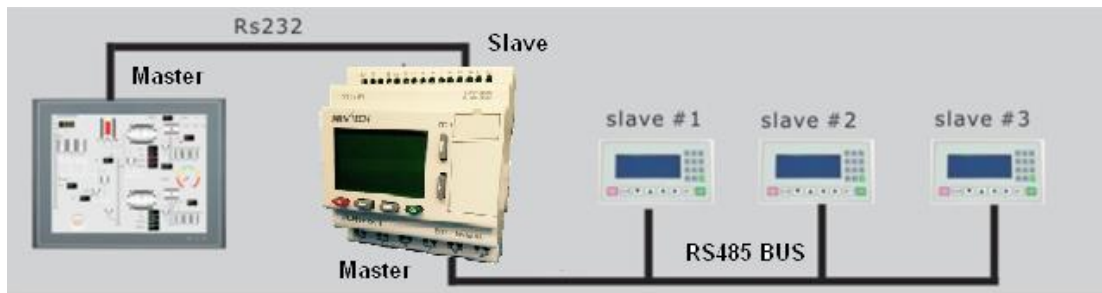
The RS485 port is used for communication between the CPU and various device or equipment which have the standard RS485 port. Communicate using Modbus RTU/ASCII protocol.



Note: PR-RS485/SR-RS485 module is required to connect the CPU to RS485 BUS.

RS232 or USB port (RS232 Cable / USB Cable needed)

If there is no network required and only one main module with some expansion modules is needed for the application, the down- and upload of the project to and from the main module happens over the standard RS232 or USB port. It allows system maintenance like monitoring too.



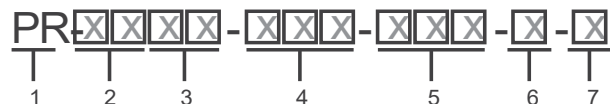
Note: PR-RS485/SR-RS485 module is required to connect the CPU to RS485 BUS.

Note

Rievtech PLC may be equipped with expansion modules of the different voltage class, but expansion module must be supplied the correct power corresponding to its type.

2. Hardware introduction

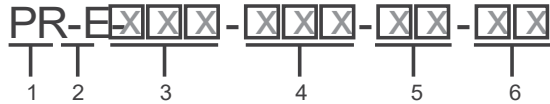
2.1 Naming Rules of PR Series PLC



1. Series name – PR Series or SR-Series.
2. Total number of inputs and outputs for the device.
3. Voltage Supply
 - a. AC
 - b. DC
 - c. AD – DC or AC 24V
4. Types of Inputs:
 - a. D – Digital only
 - b. DA – Digital and Analogue Voltage
 - c. DAI – Digital and Analogue Voltage and Current
 - d. PTDAI – PT1000, Digital and Analogue Voltage and Current
5. Type of Outputs:
 - a. R – Relay
 - b. TN – Transistor PNP
 - c. RT – Relay and Transistor PNP

d.RTA – Relay, Transistor PNP and Analogue Voltage and Current

6. E = Economic model –Non-expandable.
7. N = Ethernet built-in (Sometimes omitted)
8. 2G = Built-in 2G communication module
9. 4G = Built-in 4G communication module
10. WIFI = Built-in wifi communication module
11. 4GWIFI = Built-in 4G & wifi communication module



1. Series name – PR or SR.
2. Indicates Expansion module.
3. Type of input and output points
4. A number – Total number of inputs and outputs
 - a. AI – Analogue input only
 - b. AQ – Analogue output only
 - c. PT100 – Temperature input only PT100
5. PT100/PT1000 - Temperature input PT100 or PT1000
 - a. RS485 – Isolated RS485 communication
6. Voltage Supply:
 - a. AC
 - b. DC
7. Types of Inputs:
 - a. D – Digital only
 - b. DA – Digital and Analogue
 - c. V/I - Voltage or current
 - d. 16IN - 16 inputs
8. Type of Outputs:
 - a. R – Relay
 - b. TN – Transistor PNP
 - c. V/I - Voltage or current
 - d. 16DO - 16 outputs

2.2 Hardware model selection

Table 1 - PR-12 Series

| Model | Expansion | Supply | Inputs | Outputs | High Speed Count | PWM | HMI | RTC |
|-----------------|-----------|-------------|----------------|----------------|------------------|---------|-----|-----|
| PR-12AC-R * | No | 110-240 VAC | 8 Digital | 4 Relays (10A) | No | No | Yes | Yes |
| PR-12DC-DA-R * | No | 12-24 VDC | 8D (4DA 0-10V) | 4 Relays (10A) | 4x60kHz | No | Yes | Yes |
| PR-12DC-DA-TN * | No | 12-24 VDC | 8D (4DA 0-10V) | 4 T (PNP) | 4x60kHz | 4x10kHz | Yes | Yes |

*Discontinued production

Table 2 - PR-14 Series

| Model | Expansion | Supply | Inputs | Outputs | High Speed Count | PWM | HMI | RTC |
|----------------|-----------|-------------|----------------|----------------|------------------|-----|-----|-----|
| PR-14AC-R * | Yes | 110-240 VAC | 10 Digital | 4 Relays (10A) | No | No | Yes | Yes |
| PR-14DC-DA-R * | Yes | 12-24 VDC | 10D(6DA 0-10V) | 4 Relays (10A) | 4x60kHz | No | Yes | Yes |

*Discontinued production

Table 3 - PR-18 Series

| Model | Expansion | Supply | Inputs | Outputs | High Speed Count | PWM | HMI | RTC |
|-----------------|-----------|-------------|-----------------|----------------|------------------|---------|-----|-----|
| PR-18AC-R | Yes | 110-240 VAC | 12 Digital | 6 Relays (10A) | No | No | Yes | Yes |
| PR-18DC-DA-R | Yes | 12-24 VDC | 12D (6DA 0-10V) | 6 Relays (10A) | 4x60kHz | No | Yes | Yes |
| PR-18DC-DA-RT * | Yes | 12-24 VDC | 12D (6DA 0-10V) | 4R + 2T (PNP) | 4x60kHz | 2x10kHz | Yes | Yes |

*Discontinued production

Table 4 - PR-24 Series CPU Range – Built-In RS485 Port

| Model | Expansion | Supply | Inputs | Outputs | High Speed Count | PWM | HMI | RTC |
|-------------------|-----------|-------------|----------------------------------|---|------------------|---------|-----|-----|
| PR-24AC-R | Yes | 110-240 VAC | 14 Digital | 6 Relays (10A) | No | No | Yes | Yes |
| PR-24DC-DA-R | Yes | 12-24 VDC | 14D (6DA 0-10V) | 6 Relays (10A) | 4x60kHz | No | Yes | Yes |
| PR-24DC-DAI-RTA * | Yes | 12-24 VDC | 12D (6DA 0-10V) + 2A (0-20mA) | 6R + 2T (PNP) + 1A (0-10V/0-20mA) | 4x60kHz | 2x10kHz | Yes | Yes |

*Discontinued production

Table 5 - Ethernet Remote SR Series CPU Units

| Model | Expansion | Supply | Inputs | Outputs | High Speed Count | PWM | HMI | RTC |
|--------------|-----------|--------------------|----------------|----------------|------------------|-----|-----|-----|
| SR-12AC-R | Yes | 110-240 VAC | 8 Digital | 4 Relays (10A) | No | No | Yes | Yes |
| SR-12DC-DA-R | Yes | DC12-24V /AC24V | 8D (4DA 0-10V) | 4 Relays (10A) | No | No | Yes | Yes |

Table 6 - Ethernet Remote PR Series CPU Units – Built-In RS485 Port and Web Server

| Model | Expansion | Supply | Inputs | Outputs | High Speed Count | PWM | HMI | RTC |
|--|-----------|-------------|---|----------------------------------|------------------|---------|-----|-----|
| PR-12AC-R-N | Yes | 110-240 VAC | 8 Digital | 4 Relays (10A) | No | No | Yes | Yes |
| PR-12DC-DA-R-N | Yes | 24 VDC | 8D (4DA 0-10V) | 4 Relays (10A) | 4x60kHz | No | Yes | Yes |
| PR-18AC-R-N (V1-V3 No Built-In RS485) | Yes | 110-240 VAC | 12 Digital | 6 Relays (10A) | No | No | Yes | Yes |
| PR-18DC-DAI-R-N | Yes | 24 VDC | 12D (6DA 0-10V + 2DA(0-20mA) | 6 Relays (10A) | 4x60kHz | No | Yes | Yes |
| PR-18DC-DAI-TN-N | Yes | 12-24 VDC | 12D (6DA 0-10V) + 2A (0-20mA) | 6T (PNP) | 4x60kHz | 2x10kHz | Yes | Yes |
| PR-23DC-PTDAI-RT-N * | Yes | 24 VDC | 3xPT100 + 10D (2DA 0-10V + 4DA 0-10V or 0-20mA) | 10R + 2T(PNP) | 4x60kHz | 2x10kHz | Yes | Yes |
| PR-23DC-PTDAI-RT-4G * | Yes | 24 VDC | 3xPT100 + 10D (2DA 0-10V + 4DA 0-10V or 0-20mA) | 10R + 2T(PNP) | 4x60kHz | 2x10kHz | Yes | Yes |
| PR-26AC-R-N | Yes | 110-240 VAC | 16 Digital | 10 Relays (10/5A) | No | No | Yes | Yes |
| PR-26DC-DAI-RA-N | Yes | 24 VDC | 16D (8DA 0-10V + 4DA 0-20mA) | 8R + 2A (1 x 0-10V + 1 x 0-20mA) | 4x60kHz | No | Yes | Yes |
| PR-26DC-DAI-RT-N | Yes | 24 VDC | 16D (8DA 0-10V + 4DA 0-20mA) | 8R + 2T (PNP) | 4x60kHz | 2x10kHz | Yes | Yes |
| PR-26DC-DAI-RT-2G | Yes | 24 VDC | 16D (8DA 0-10V + 4DA 0-20mA) | 8R + 2T (PNP) | 4x60kHz | 2x10kHz | Yes | Yes |
| PR-26DC-DAI-RT-4G | Yes | 24 VDC | 16D (8DA 0-10V + 4DA 0-20mA) | 8R + 2T (PNP) | 4x60kHz | 2x10kHz | Yes | Yes |
| PR-26DC-DAI-RT-WIFI | Yes | 24 VDC | 16D (8DA 0-10V + 4DA 0-20mA) | 8R + 2T (PNP) | 4x60kHz | 2x10kHz | Yes | Yes |
| PR-26DC-DA-RT-4GWIFI | Yes | 24 VDC | 16D (8DA 0-10V + 4DA 0-20mA) | 8R + 2T (PNP) | 4x60kHz | 2x10kHz | Yes | Yes |

*Discontinued production

Table 7 - Expansion Modules for PR-14, PR-18, PR-24, PR-26 and Remote PLC CPUs (Max. 16 Units)

| Model | Supply | Inputs | Outputs |
|-------------------|-------------|--------------------------------|---------------------------------|
| PR-E-16AC-R | 110-240 VAC | 8 Digital | 4 Relays (10A) + 4 Relays (3A) |
| PR-E-16DC-DA-R | 12-24 VDC | 8Digital (4DA 0-10V) | 4 Relays (10A) + 4 Relays (3A) |
| PR-E-16DC-DA-TN | 12-24 VDC | 8Digital (4DA 0-10V) | 8 Transistor (0.3A) |
| PR-E-DC-16IN | 12-24 VDC | 16Digital (4DA 0-10V) | - |
| PR-E-DC-16DO | 12-24 VDC | - | 15 Relays (3A) + 1 Relays (10A) |
| PR-E-AI-I ** | 12-24 VDC | 4x0/4-20mA | - |
| PR-E-PT100 ** | 12-24 VDC | 3 x Thermistor PT100 | - |
| PR-E-PT100/PT1000 | 12-24 VDC | 4 x Thermistor PT100/PT1000 | - |
| PR-E-AQ-VI ** | 12-24 VDC | - | 2 x 0-10 V/0-20 mA |
| PR-E-AI-V/I | 12-24 VDC | 4 x 0-10 V/0-20 mA(14bit) | - |
| PR-RS485 | 12-24 VDC | Additional Isolated RS485 Port | |
| PR-E-4AI-V/I * | 12-24 VDC | 4 x 0/4-20mA | - |
| PR-E-2AQ-V/I * | 12-24 VDC | - | 2 x 0-10 V / 0-20 mA |

*Adopts the SR-E series housing, which is smaller in size.

** Discontinued production

Table 8 - Expansion Modules for SR PLC CPUs (Max. 3 Units for SR-12)

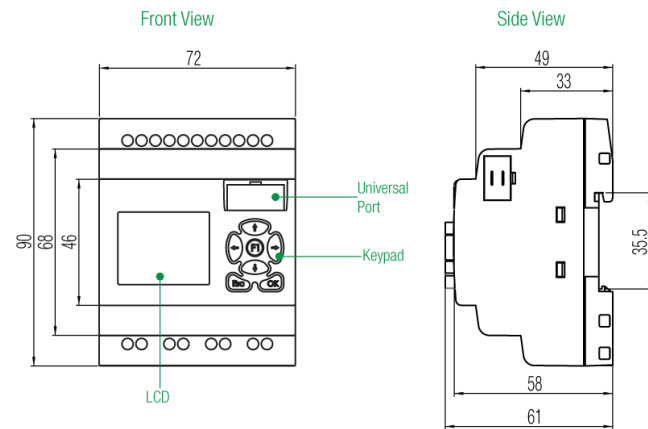
| Model | Supply | Inputs | Outputs |
|----------------------|-------------|--------------------------------|--------------------------------|
| SR-E-8AC-R | 110-240 VAC | 4 Digital | 2 Relays (10A) + 2 Relays (3A) |
| SR-E-8DC-DA-R | 12-24 VDC | 4Digital (4DA 0-10V) | 2 Relays (10A) + 2 Relays (3A) |
| SR-E-1AQ-VI | 12-24 VDC | - | 1 x 0-10 V/0-20 mA |
| SR-E-2AI-VI | 12-24 VDC | 2 x 0-10 V/0-20 mA | - |
| SR-E-2PT00 | 12-24 VDC | 2 x Thermistor PT100 | - |
| SR-RS485 | 12-24 VDC | Additional Isolated RS485 Port | |

Table 1 - Rievtech Accessories

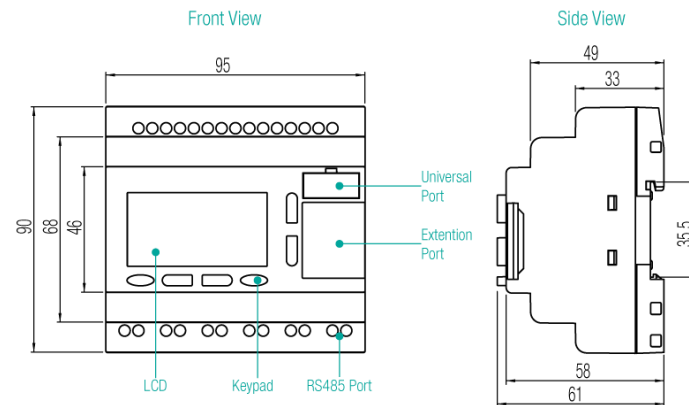
| Model | Description |
|--------------------|---|
| RS232 Cable | RS232 Programming and communication lead between CPU and PC |
| USB Cable | USB Programming lead between CPU and PC. Needed for firmware updates. |
| ELC-COPIER | A device used to copy a program from one CPU to another same model CPU |
| ELC-MEMORY | Real-time data logging module, which can log IO and program data with time and date stamp |
| PRO-RS485 | Optically isolated RS232 to RS485 converter cable |
| ELC-BATTERY | A standard number of days for RTC retentivity is 20-days. This unit extends that to 1-year. |
| PR-FLAT | Expansion PR-E Ribbon Cable 1-metre |

2.3 Structure & dimension

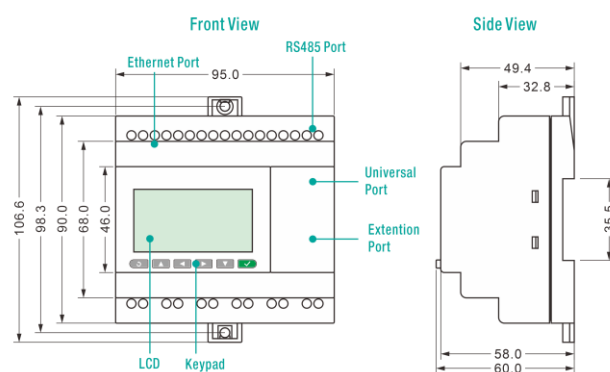
1. PR-12 series with LCD:



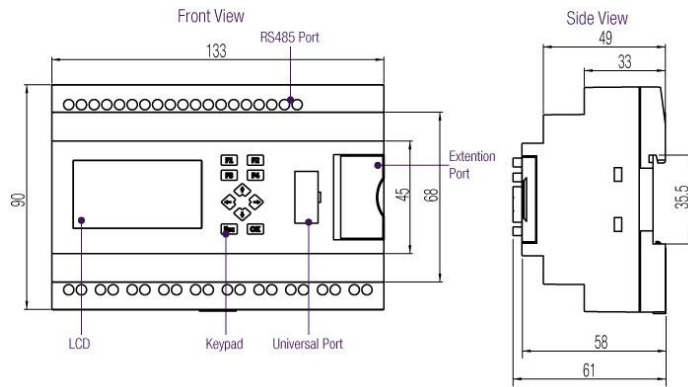
2. PR-14 and PR-18 Series



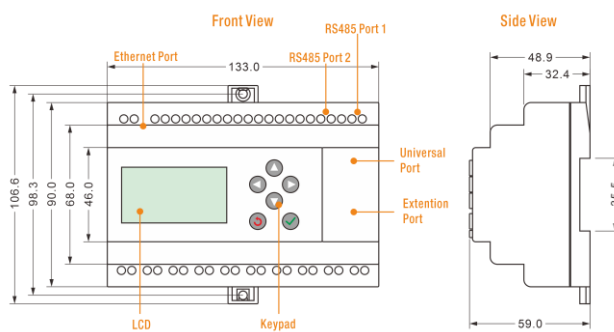
3. PR-18(V2) Series



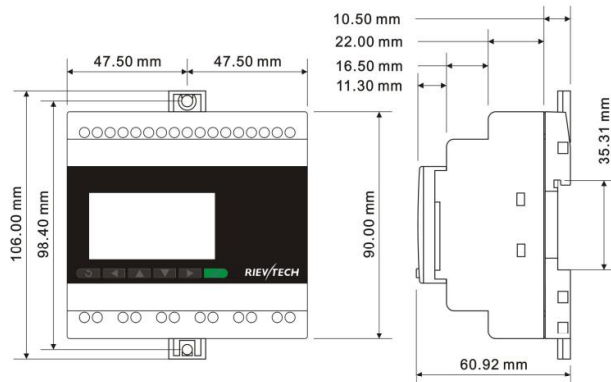
4. PR-24 Series



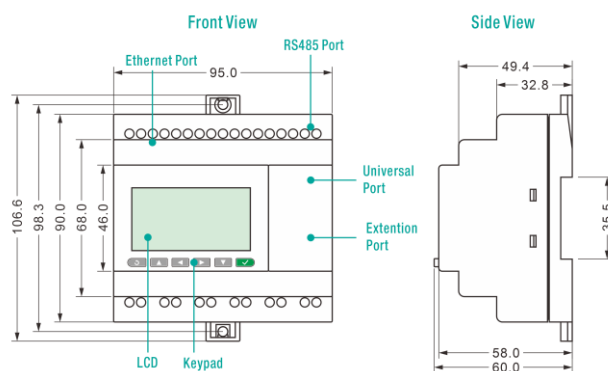
5. PR-24(V2) Series



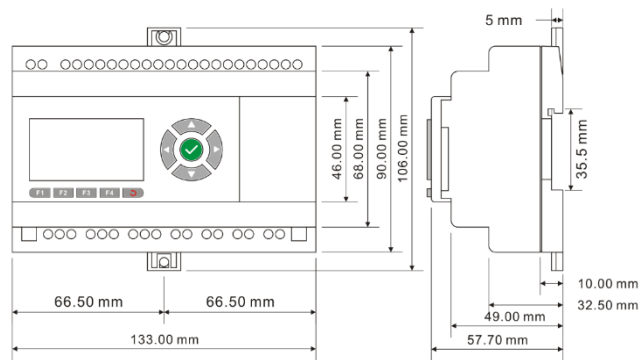
6. PR-12N, 18N Series



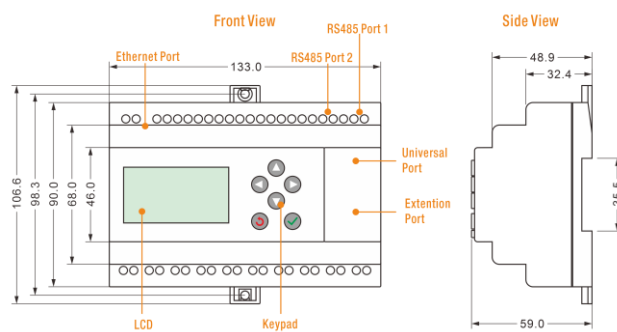
7. PR-12N, 18N(V4) Series



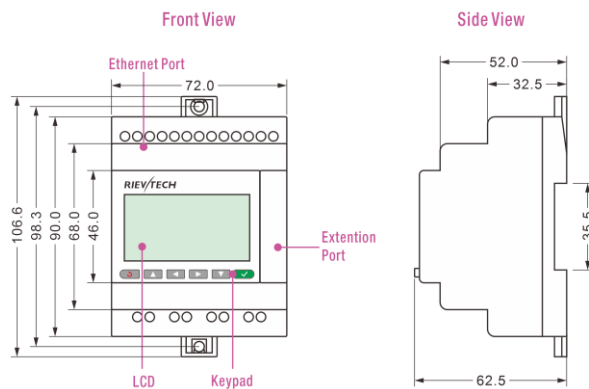
8. PR-26 Series



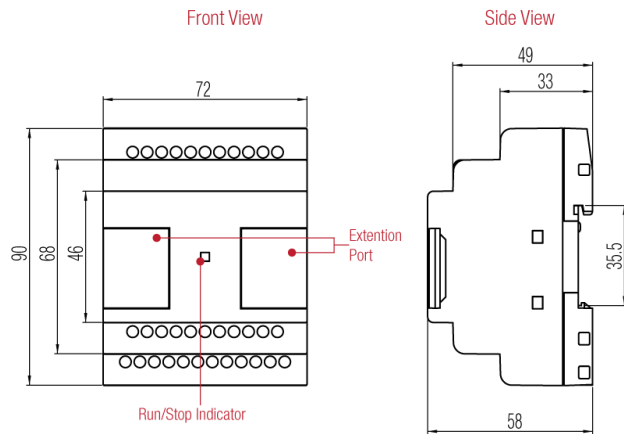
9. PR-26(V4) Series



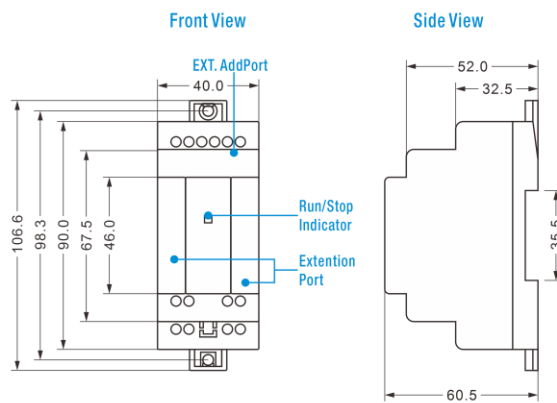
10. SR-12 Series



11. PR-E Expansion Modules



12. SR-E Expansion Modules



3 .Installing/removing Rievtech Micro PLC

Dimensions

The dimensions of the Rievtech hardware is compliant with DIN 43880. Rievtech can be “snap-mounted” to 35mm (EN 50022) DIN rail or wall mounted.

Rievtech Micro PLC width:

| Model | Width (mm) |
|------------|------------|
| PR-12 CPU | 72 |
| PR-14 CPU | 95 |
| PR-12N CPU | 95 |
| PR-18 CPU | 95 |
| PR-18N CPU | 95 |
| ELC-22 CPU | 133 |
| PR-24 CPU | 133 |
| PR-26 CPU | 133 |

| | |
|------------------------------|----|
| SR-12 CPU | 72 |
| PR-E Expansion module | 72 |
| SR-E Expansion module | 40 |



Warning

Always switch off power before you “remove” and “insert” an expansion module.

3.1 DIN rail mounting

Mounting

How to mount a Rievtech module and an expansion module onto a DIN rail:

1. Hook the Rievtech Basic module onto the rail.
2. Push down the lower end to snap it on. The mounting interlock at the rear must engage.
3. Hook the Rievtech expansion module onto the rail
4. Slide the module towards the left until it touches the Rievtech CPU.
5. Push down the lower end to snap it on. The mounting interlock at the rear must engage.
6. Remove the plastic cover in the expansion port of CPU and expansion module.
7. Plug the connector on the flat cable to CPU



Repeat the expansion module steps to mount further expansion modules.

Note: If you need install the expansion and CPU on different rows, you need order the longer flat connection which is used to connected with CPU, the longest distance can be 200meters between the CPU and the end expansion module.

Removal

To remove Rievtech Micro PLC:

if you have installed only one Rievtech Basic:

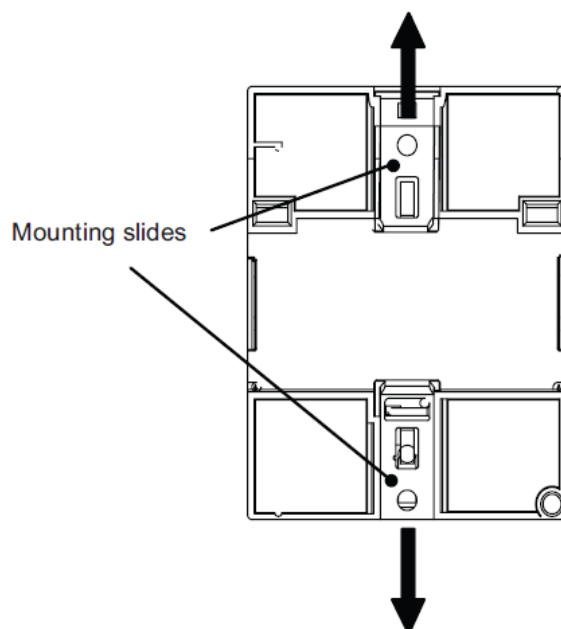
1. Insert a screwdriver into the eyelet at the bottom of the slide interlock and move the latch downward.
2. Swing the Rievtech Basic off the DIN rail.

if you have connected at least one expansion module to Rievtech Basic:

1. Remove the connector on the flat cable
 2. Slide the expansion module off towards the right.
 3. Insert a screwdriver into the eyelet at the bottom of the slide interlock and lever it downward.
 4. Swing the expansion module off the profile rail.
- Repeat steps 1 to 4 for all other expansion modules.

3.2 Wall-mounting

For wall-mounting, first slide the mounting slides on the rear side of the devices towards the outside. You can now wall-mount Rievtech Micro PLC by means of two mounting slides and two ØM4 screws (tightening torque 0.8 to 1.2 Nm).



Drilling template for wall-mounting

Before you can wall-mount Rievtech Micro PLC, you need to drill holes using the template shown below.

All dimensions in mm

Bore hole for Ø M4 screw, tightening torque 0.8 to 1.2 Nm

3.3 wiring Rievtech Micro PLC

Wire the Rievtech Micro PLC by using a screwdriver with a 3-mm blade.

You do not need wire ferrules for the terminals. You can use conductors with cross-sections of up to the following thicknesses:

1 x 2.5 mm²

2 x 1.5 mm² for each second terminal chamber

Tightening torque: 0.4.. .0.5 N/m or 3. ..4 lbs/in

Note

Always cover the terminals after you have completed the installation. To protect Rievtech Micro PLC adequately from impermissible contact to live parts, local standards must be complied with.

3.4 Connecting the power supply

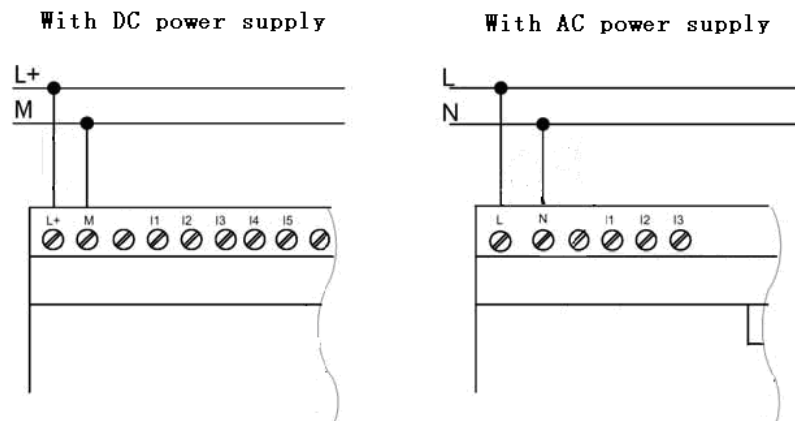
The SR-12AC, PR-12AC, PR-18AC, PR-24AC, PR-12AC-N, PR-18AC-N, PR-26AC versions of Rievtech Micro PLC are suitable for operation with rated voltages of 110 V AC and 240 V AC. The SR-12DC, PR-12DC, PR-18DC, PR-24DC, PR-12DC-N, PR-18DC-N, PR-26DC versions can be operated with a 12 or 24 VDC power supply.

Note

A power failure may cause an additional edge triggering signal.

Data of the last uninterrupted cycle are stored in Rievtech Micro PLC

The power supply:



3.4.1 Connecting Rievtech micro PLC inputs

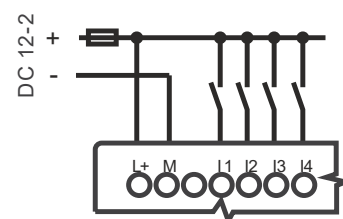
1. Requirements

Inputs can be connected to various sensor elements such as momentary switches, switches, light barriers, daylight control switches, etc.

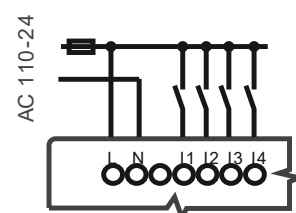
| Signal state | AC Type | DC Type |
|----------------|-----------------|----------------|
| OFF/0 | < 40 VAC | < 5 VDC |
| Input Current | Typical 0.05 mA | Typical 1 mA |
| ON/1 | > 79 VAC | > 8 VDC |
| Input Current | Typical 0.1 mA | Typical 1.7 mA |
| Analogue Input | NOT APPLICABLE | 0 to 10 VDC |

2. Connecting Rievtech is shown as in the following figures:

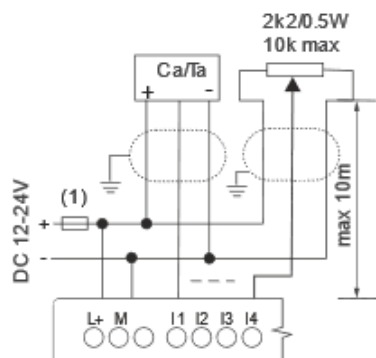
(1) DC Type Digital Input Connections



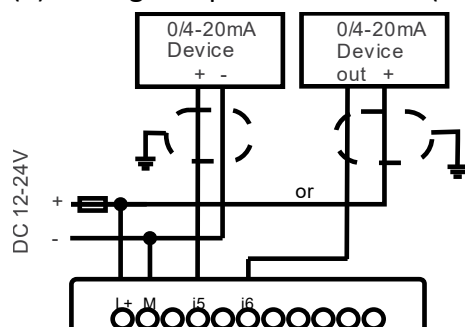
(2) AC Type Digital Input Connections



(3) Analogue Input Connections (0 to 10 VDC)

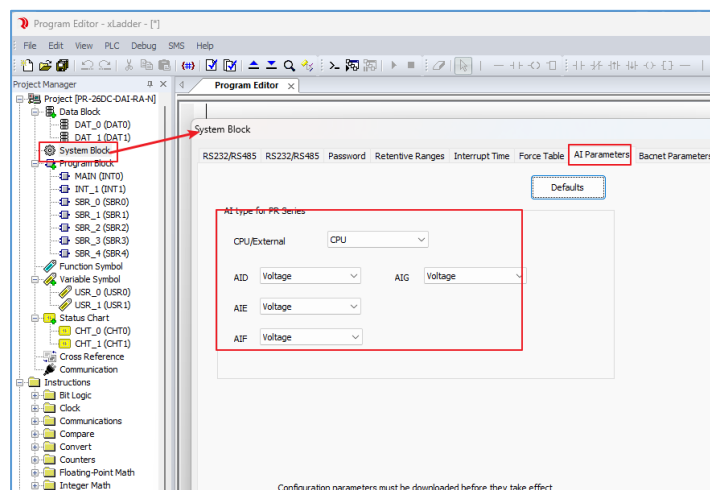


(4) Analogue Input Connections (0 to 20 mA)



NOTICE:

For (0~20mA /0~10V DC) input, the sensor type needs to be set during programming. For example, ID~IG of PR-26DC. Refer to the following figure:

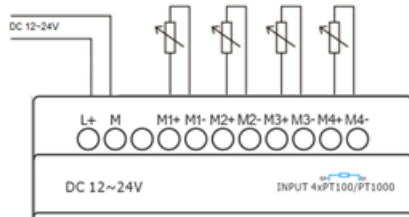
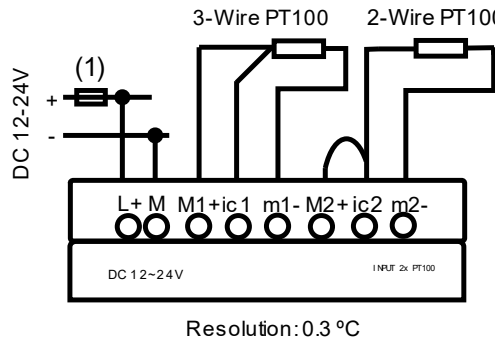


(5) Analogue Input Connections (PT100/PT1000)

Either 2 or 3-wire PT100 / 2-wire PT1000 sensors can be connected.

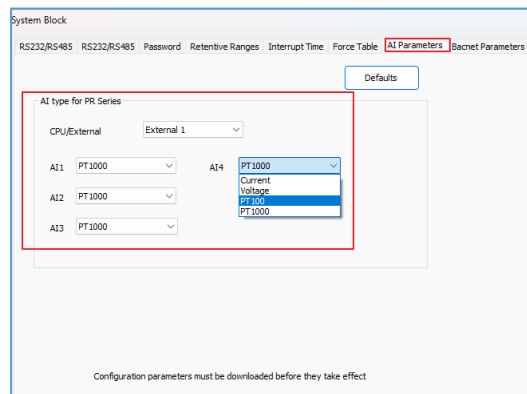
With a 2-wire sensor, connect to terminals M1+ and M1-. Short between M1+ and IC1. There is no compensation for any impedance in the wire. A measurement error of 1 Ω is equivalent to +2.5 $^{\circ}\text{C}$.

Using a 3-wire sensor can inhibit any influence caused by cable length.



NOTICE:

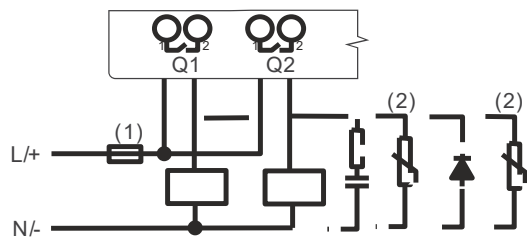
For (PT100 /PT1000) input, the sensor type needs to be set during programming.



3.4.2 Connecting Outputs

1. Requirement for the relay output

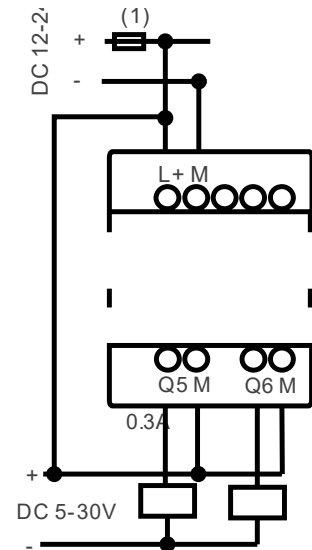
Various loads such as a lamp, fluorescent tube, motor, contactor, etc., can be connected to the relay outputs of the Rievtech hardware. The maximum output current that can be handled by the relay is 10A for the resistance load and 3A for an inductive load. The relay output connection should be wired as per the following figure:



2. Requirement for the transistor output:

The load connected to a transistor output must have the following characteristics:

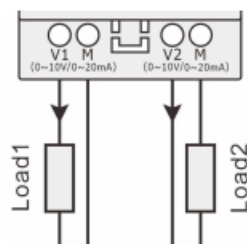
- * The maximum switching current cannot exceed 0.3A.
- * When the switch is ON (Q=1), the maximum current is 0.3A.



i NOTICE:

- * The load connecting voltage must be ≤ 30 VDC and it must be DC.
- * The "+" terminal of the output wiring must be connected to the DC positive voltage, and it must be connected with the "L+" terminal of the CPU power terminal. The load must be connected with the "-ve" terminal of the DC negative voltage supply.

3. Requirement for the Analogue output:

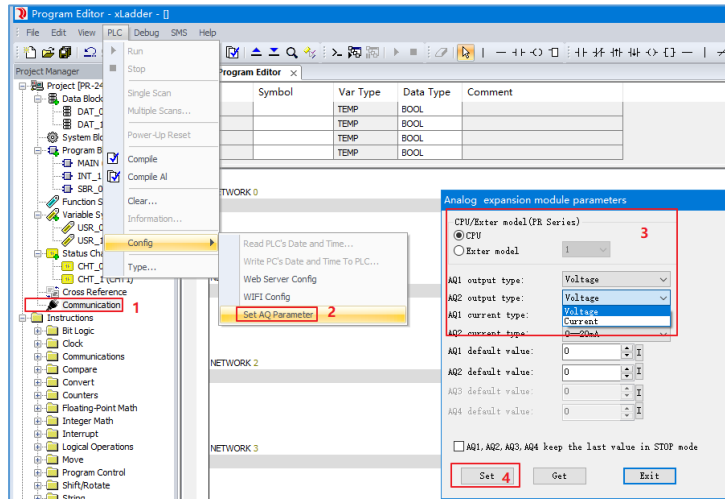


0~10V: $R \geq 650\Omega$ / 0~20mA: $R \leq 500\Omega$

i NOTICE:

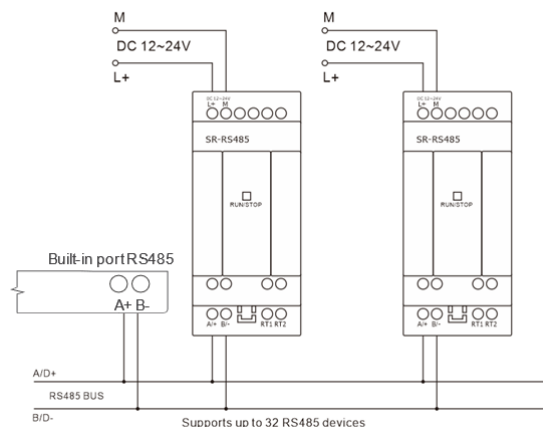
Set the output signal (voltage/current) of AQ1, AQ2 on the programming software.

Menu Path: 'Tools – Transfer – Set AQ Parameter'



PR-RS485/SR-RS485

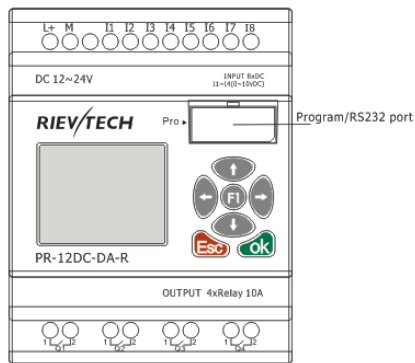
The RS485 expansions are just converters with photo isolation providing 3-sets of wiring terminals (although there are 3-sets of terminals, they are connected within the unit, so only one channel RS485 is available) of expansion port of PLC CPU for easy connection with other devices.



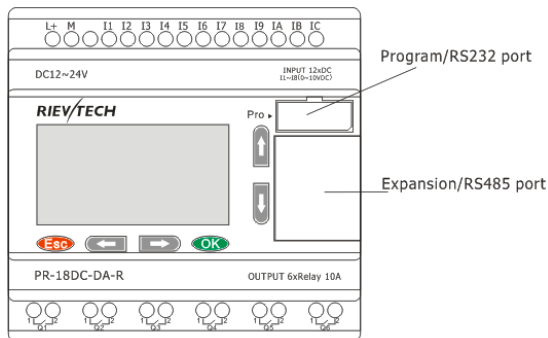
If “RT1”, RT2” terminals are short connected, one 120R resistor will be connected between A/+ and B/- providing a termination resistor for end-of-line purposes.

3.4.3 Communication port instructions:

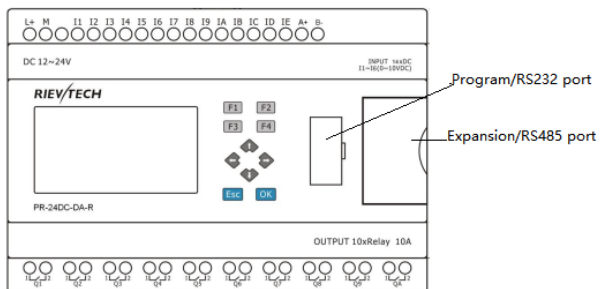
PR-12 CPUs



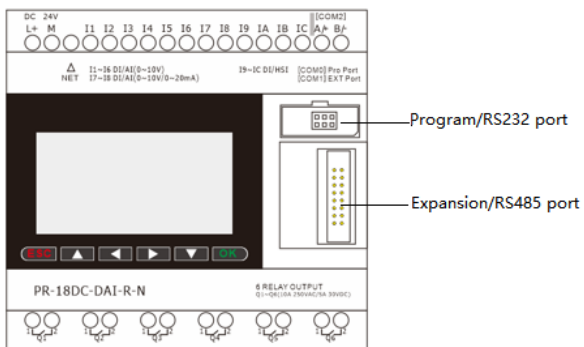
PR-14, PR-18 CPUs



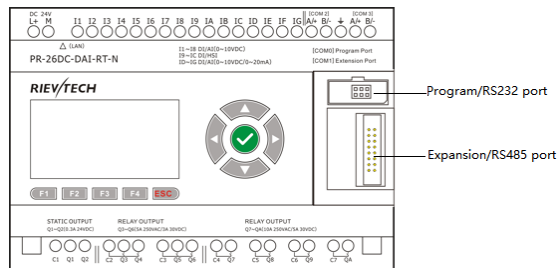
ELC-22, PR-24 CPUs



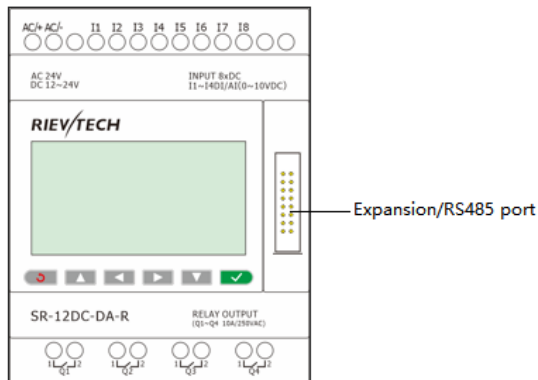
PR-12N, PR-18N CPUs



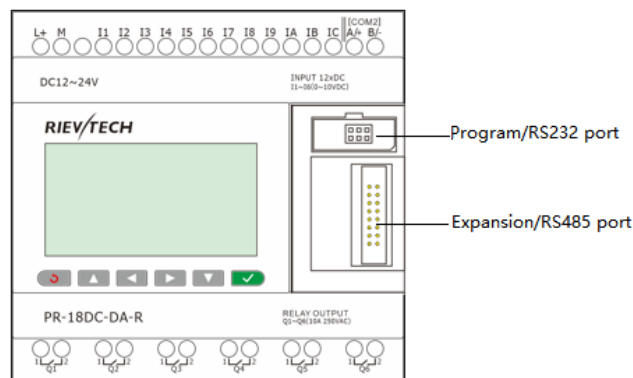
PR26 CPUs



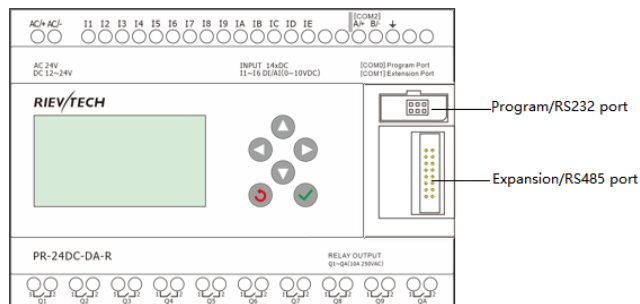
SR-12 CPUs



NEW PR-18(V2), PR-12N(V4), PR-18N(V4) CPUs



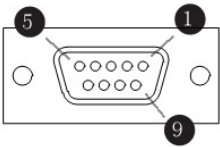
NEW PR-24(V2), PR-26(V4) CPUs



The programming port/RS232 port can be used with any of the following Rievtech devices. RS232 Cable, USB Cable, ELC-Copier, ELC-MEMORY, ELC-BATTERY, and PRO-RS485. This is COM 0.

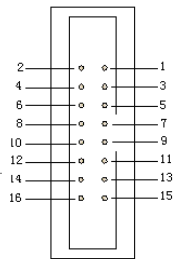
When the programming port is being used as the standard RS232 port (D-shape 9 pin header). The RS232 Cable is needed. Below shows you the

pinouts for the device:



RS232 Pinouts

| PIN | Function |
|--------|----------|
| 2 | RXD |
| 3 | TXD |
| 5 | GND |
| others | NULL |



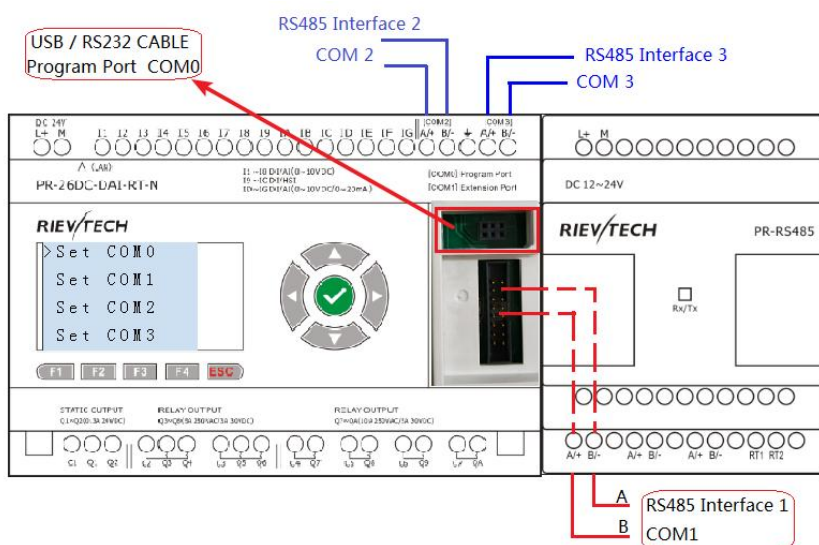
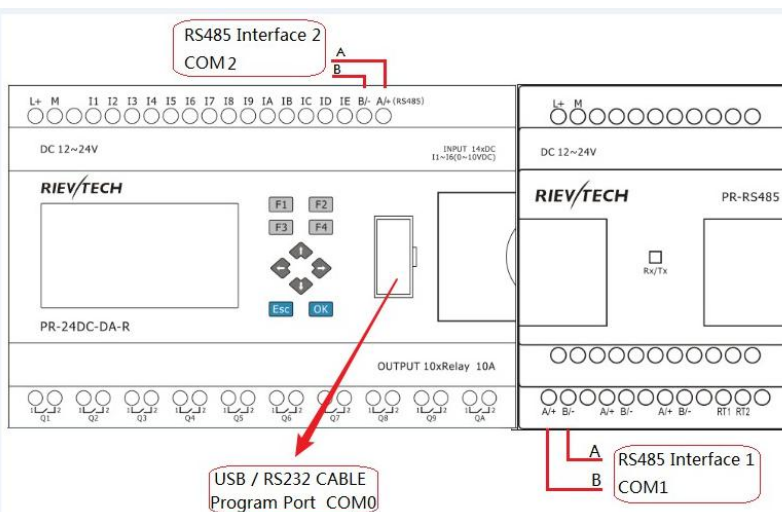
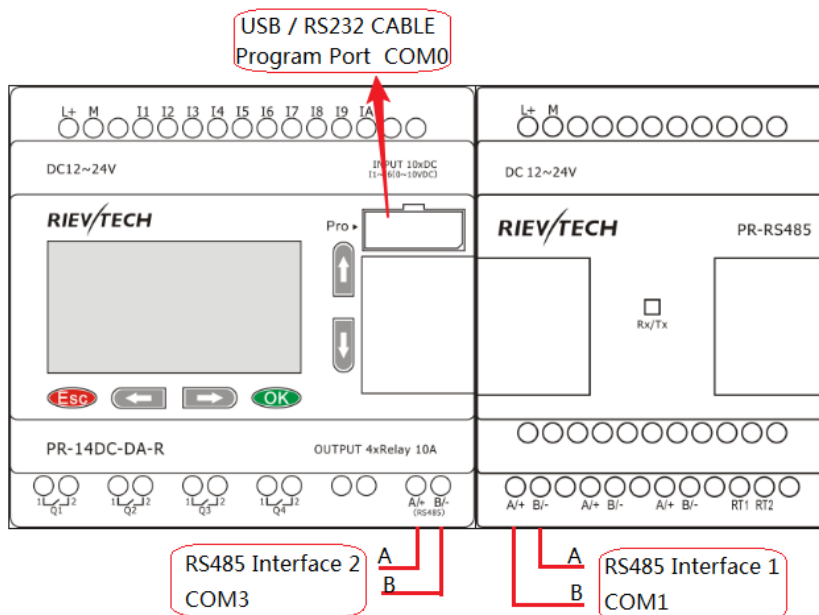
RS485 Pinouts

| | |
|---------------|------------|
| 3-----RS485 A | 7-----CANL |
| 5-----RS485 B | 9-----CANH |
| 4-----GND | 15-----+5V |
| 6-----GND | 16-----+5V |

Communication between CPU and expansion module will use pins 4, 7, 9 and 15.

PR-RS485/SR-RS485 module is required when either the PR-14/ PR-18/ PR-24/ PR12N/ PR-18N/ PR-26 CPUS are required to communicate with a third-party device via RS485. Note the PR-24/ PR12N/ PR-18N/ PR-26 have built-in RS485.

COM port diagram:



4.Quick reference manual

4.1 Special memory-SM area:

SMB0

| | | |
|---------------|-------|----------------------------------|
| Always_On | SM0.0 | Always ON |
| First_Scan_On | SM0.1 | ON for the first scan cycle only |

SMB1

| | | |
|------------------|-------|---|
| Result_0 | SM1.0 | Set to 1 by the execution of certain instructions when the operation result = 0 |
| Overflow_Illegal | SM1.1 | Set to 1 by exec. of certain instructions on overflow or illegal numeric value. |
| Neg_Result | SM1.2 | Set to 1 when a math operation produces a negative result |
| Divide_By_0 | SM1.3 | Set to 1 when an attempt is made to divide by zero |
| Table_Overflow | SM1.4 | Set to 1 when the Add to Table instruction attempts to overfill the table |
| Table_Empty | SM1.5 | Set to 1 when a LIFO or FIFO instruction attempts to read from an empty table |
| Not_BCD | SM1.6 | Set to 1 when an attempt is made to convert a non-BCD value to a binary value |
| Not_Hex | SM1.7 | Set to 1 when an ASCII value cannot be converted to a valid hexadecimal value |

SMB34-SMB35 Time Interval Registers for Timed Interrupts

| | | |
|--------------|-------|--|
| Time_0_Intrl | SMB34 | Timed Interrupt 0: Time interval value (in 1 ms increments from 1 ms to 255 ms). |
| Time_1_Intrl | SMB35 | Timed Interrupt 1: Time interval value (in 1 ms increments from 1 ms to 255 ms). |

Newly added special registers and their functions

| | |
|----------------|---|
| SMB10 | Adjust the contrast of LCD display. (value range: 0~25) |
| SMB192 | Control the LCD backlight on or off. (0: off, 1: always on) |
| SMB12 SMB13 | Used to indicate the connection status of the expansion module. SM12.0 - SM12.7: The status of the bit corresponds to the connection status of the expansion modules with station address 1 to 8. |

| | |
|-------|---|
| | <p>SM13.0 - SM13.7: The status of the bit corresponds to the connection status of the expansion modules with station address 9 to 16.</p> <p>‘0’ means the expansion is not connected/communicating with the CPU, ‘1’ means the expansion is connected/communicating with the CPU normally.</p> |
| SMB14 | <p>Working status of built-in SD:</p> <p>0: There are no errors.</p> <p>1: Failed to open file.</p> <p>2: Update file new content error, write failure.</p> <p>3: Exceeds the maximum allowed file size</p> <p>4: Insufficient memory, write failed</p> <p>60: No SD card detected</p> |

* Note that the functions in this table have requirements on the PLC firmware version.

4.2 Interrupt Events:

| | | | |
|-------------------|-------------------|----|------------------|
| I1.2 Rising edge | PLC_EVENT_INPUTP0 | 0 | Highest priority |
| I1.4 Rising edge | PLC_EVENT_INPUTP1 | 1 | High priority |
| Timer interrupt 0 | PLC_EVENT_TIMER0 | 10 | Low priority |
| Timer interrupt 1 | PLC_EVENT_TIMER1 | 11 | Lowest priority |

4.3 High speed counter:

Correspondence between HC and input

| Model | HC0 | HC1 | HC2 | HC3 |
|---------------|-----|-----|-----|-----|
| PR-12DC-DA-R | I5 | I6 | I7 | I8 |
| PR-12DC-DA-TN | I5 | I6 | I7 | I8 |
| PR-14DC-DA-R | I7 | I8 | I9 | IA |
| PR-18DC-DA-R | I9 | IA | IB | IC |

| | | | | |
|----------------------|----|----|----|----|
| PR-18DC-DA-RT | I9 | IA | IB | IC |
| PR-24DC-DA-R | I9 | IA | IB | IC |
| PR-24DC-DAI-RTA | I9 | IA | IB | IC |
| PR-12DC-DA-R-N | I5 | I6 | I7 | I8 |
| PR-18DC-DAI-R-N | I9 | IA | IB | IC |
| PR-18DC-DAI-TN-N | I9 | IA | IB | IC |
| PR-23DC-PTDAI-RT-N | I9 | IA | IB | IC |
| PR-23DC-PTDAI-RT-4G | I9 | IA | IB | IC |
| PR-26DC-DAI-RA-N | I9 | IA | IB | IC |
| PR-26DC-DAI-RT-N | I9 | IA | IB | IC |
| PR-26DC-DAI-RT-4G | I9 | IA | IB | IC |
| PR-26DC-DAI-RT-WIFI | I9 | IA | IB | IC |
| PR-26DC-DA-RT-4GWIFI | I9 | IA | IB | IC |

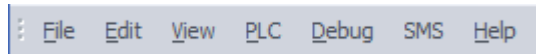
The value in HC can be directly read to obtain the high-speed counting value.

Each high-speed counter occupies an input point for receiving the pulse. No reset, adjust the direction, start and other functions. The high-speed counter can only be used for recording number in the PLC.

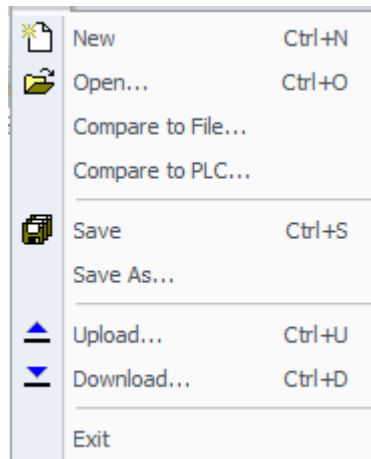
XLadder direction for use

5.The detailed annotation of operation interface

5.1The main menu



5.1.1 File



New: New command is used to create a new project, you can use the shortcut CTRL + N to create a new project.

Open: Open an existing project, VCW format files, which can be opened by the shortcut CTRL + O.

Compare to File: Compare the currently editable program opened by xLadder with the program saved in the PC path.

Compare to PLC: Compare the currently editable program opened by xLadder with the program currently running in the PLC.

Save: Save the current edit program, you can use the shortcut CTRL + S to save.

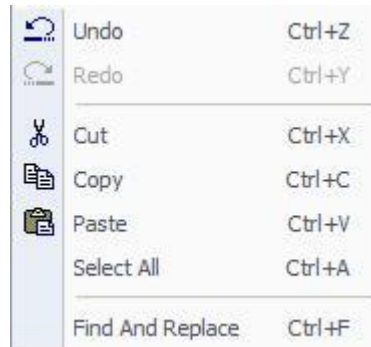
Save as: You can save the project and you can save the project with other names which have been saved.






Upload: Read the program from the PLC, ensure that communication is normal. Program is read to a new project, can be saved and named. You can use the shortcut CTRL + U to read.

Download: Write program to the PLC, Compiler success and then click the download or use the shortcut CTRL + D to download the program.

Exit: Close the XLadder software.

5.1.2 Edit



| | | |
|---|------------------|--------|
|  | Undo | Ctrl+Z |
|  | Redo | Ctrl+Y |
| <hr/> | | |
|  | Cut | Ctrl+X |
|  | Copy | Ctrl+C |
|  | Paste | Ctrl+V |
| | Select All | Ctrl+A |
| <hr/> | | |
| | Find And Replace | Ctrl+F |

Undo: Returns to the last operation, you can send multiple "undo" command. If you send out "open" and "off" or "save" or "compile" command, "undo" buffer is cleared. Your next action is recorded as the beginning of a new "undo" order. Can use the shortcut CTRL + Z to undo operation.

Redo: Contrary to the function of the undo.

Cut: Select an object, cut it, and place it in the WINDOWS clipboard buffer.

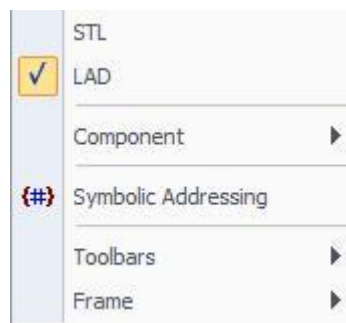
Copy: Select the object, copy the operation, and the copy of the object is placed in the WINDOWS clipboard buffer.

Paste: Stick the cut or copied object in the selected area.

Select all: Select all the text of the current cursor position, you can use the shortcut key CTRL+A.

Find and replace: Is used to perform the "find", "replace" and "go" operation on the program, local variable table, data block, symbol table or state table, and can use the shortcut key CTRL+F.

5.1.3 View



STL: Display the program to STL instruction.

LAD: Display the program to LAD instruction.

Component: Components include data blocks, system blocks, program editing, function symbols, variable symbols, cross reference and communication settings, will be detailed description of the function of the module in the [instruction tree](#).

Symbolic addressing: After the start symbol editing function, it will display the symbol of the annotation.

Program Editor Variable Symbol

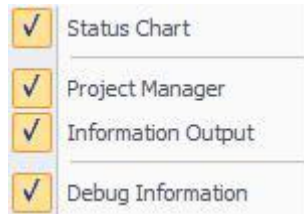
| | Symbol | Address | Data Type | Comment |
|---|--------|---------|-----------|---------|
| ✓ | Start | M0.0 | BOOL | |
| ✓ | Stop | M0.2 | BOOL | |
| ✓ | Motor | M0.3 | BOOL | |
| | | | BOOL | |

Toolbar: The toolbar of the contents are as follows



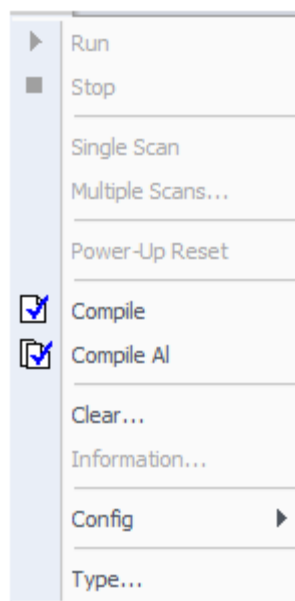
You can choose to use the tools, the default is full.

Floating window: Window that can be moved in the operating interface. Its contents are as follows:



Status table, project management, information output and debugging information window can be moved.

5.1.4 PLC



RUN: Make RUN in PLC mode, running the program in PLC, if you use the software to open RUN mode, you need to ensure the normal communication between software and PLC.

Stop: Make STOP in PLC mode, to stop the program in the PLC. If you use the software to make STOP into the PLC mode, you need to ensure the normal communication between software and PLC.

Compile: Compile the program of the current page.

Compile all: Compile all project components (program block, data block and system block).

Clear: Clear all the data in the PLC, only offline can clear the data from the PLC.

Information: The information view of PLC, we can only see in the connection state.

Clear: Clear all the PLC variable.

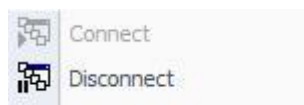
Information: After clicking the 'Connect' button in the toolbar, this menu is valid when xLadder is

in online monitoring state. Read the firmware version and operating status information of the PLC.

Config: xLadder communicates with PLC, reads or writes some information: Read PLC' Date and Time, Write PLC' Date and Time, Web Server Config, WIFI Config, Set AQ Parameter.

Type: Can choose the type of PLC.

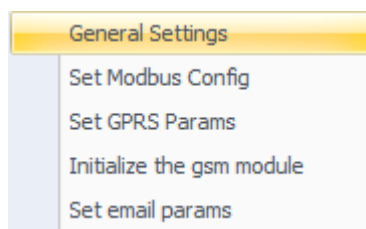
5.1.5 Debug



Connect: Display PLC data status in the program editor window.

Disconnect: No longer monitor the current value of the PLC data. In offline state does not represent the state of STOP in PLC, if you want to make STOP in PLC state, you can modify the state of PLC in the connection state, in offline state can't be modified.

5.1.6 SMS



General Settings

Info

Setting

General Settings

Enter PIN 1

Provider search ☒ Automatic ☐ Manual

GSM Service Provider

GSM Service Center

description

Write Read

OK Cancel Help

Set Modbus Config

Config

Select Language

Language

Data Register Index of Modbus

☐ High low ☒ Low High

Set SMS Config

☐ Sms Model

File Optimization Config

☒ File optimization

Set Extended module

☐ Extended module

Set extension address

Address

Set PLC Source Type

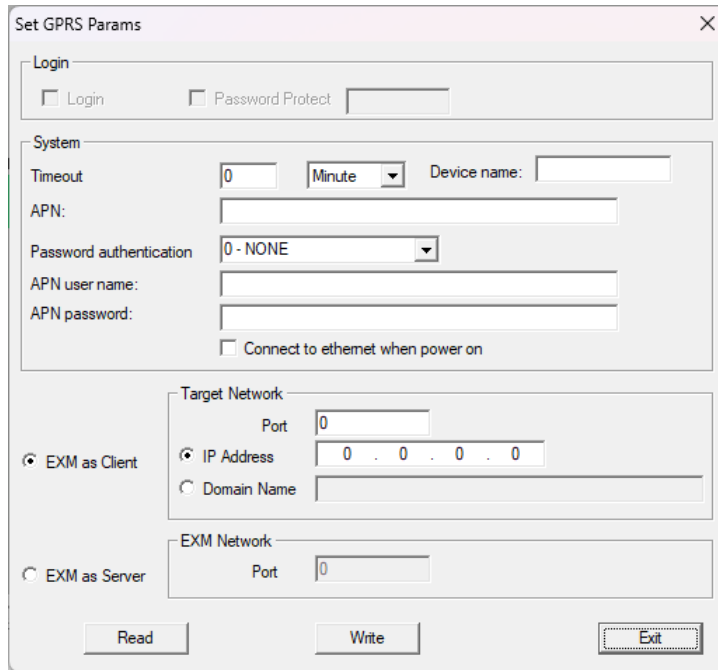
☐ AC ☒ DC

Allow entity's output pin empty

☐ Empty

OK Cancel

Set GPRS Params



Set GPRS Params

Login

☐ Login ☐ Password Protect

System

Timeout: Minute Device name:

APN:

Password authentication:

APN user name:

APN password:

☐ Connect to ethernet when power on

Target Network

☒ EXM as Client

Port:

☒ IP Address: . . .

☐ Domain Name:

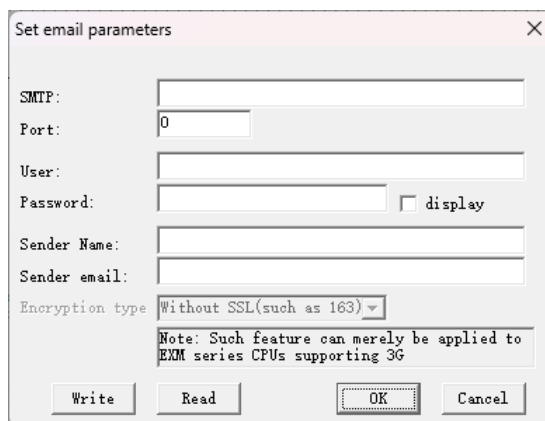
EXM Network

☐ EXM as Server

Port:

Initialize the gsm module

Set email params



Set email parameters

SMTP:

Port:

User:

Password: ☐ display

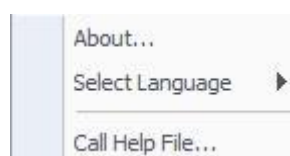
Sender Name:

Sender email:

Encryption type:

Note: Such feature can merely be applied to EXM series CPUs supporting 3G

5.1.7 Help



About...

Select Language ▶

Call Help File...

About: the information of the software. The version number of xLadder can be found

here.

Choose Language: Can choose Chinese, English or Russian

Call help file: Can call help file.

5.2 Toolbar



New: New command is used to create a new project; you can use the shortcut CTRL + N to create a new project.

Open: Open an existing project, VCW format files, which can be opened by the shortcut CTRL + O.

Save: Save the current edit program, you can use the shortcut CTRL + S to save.

Undo: Returns to the last operation, you can send multiple "undo" command. If you send out "open" and "off" or "save" or "compile" command, "undo" buffer is cleared. Your next action is recorded as the beginning of a new "undo" order. Can use the shortcut CTRL + Z to undo operation.

Redo: Contrary to the function of the undo.

Cut: Select an object, cut it, and place it in the WINDOWS clipboard buffer.

Copy: Select the object, copy the operation, and the copy of the object is placed in the WINDOWS clipboard buffer.

Paste: Stick the cut or copied object in the selected area.

Symbolic addressing: After the start symbol editing function, it will display the symbol of the annotation.

Compile: Compile the program of the current page.

Compile all: Compile all project components (program block, data block and system block).

Upload: Read the program from the PLC, ensure that communication is normal. Program is read to a new project, can be saved and named. You can use the shortcut CTRL + U to read.

Download: Write program to the PLC, Compiler success and then click the download or use the shortcut CTRL + D to download the program.

Network Search: Search for the PR Ethernet PLC in the same LAN as the PC. This tool can be used to configure the Ethernet parameters of the Ethernet PLC.

FBD Blocks: Function is under development.

Simulate: Offline simulation function of xLadder.

Connect: Display PLC data status in the program editor window.

Disconnect: No longer monitor the current value of the PLC data. In offline state does not represent the state of STOP in PLC, if you want to make STOP in PLC state, you can modify the state of PLC in the connection state, in offline state can not be modified.

Run: Run instructions in the main menu.

Stop: Stop instructions in the main menu

Erase: Select the instruction that has been written, click erase, delete instruction.

Select: When the selection is lit, it indicates that the current location area can be selected, copied, cut and pasted. Selection is gray, the current location of the selected operation can not be carried out.

Vert Line: Place a vertical connecting line.

Horz Line: Place a horizontal connecting line.

Horz Contact: Place a contact. After selecting this tool, click on a blank area in the program editing area to pop up all contacts for selection.

Horz Coil: Place a coil. After selecting this tool, click in a blank area of the program editing area to pop up all coils for selection.

Horz Block: Place a function block. After selecting this tool, click in a blank area of the program editing area to pop up all function blocks for selection.

Normally open contacts: Click to select the normally open contact, in the program editing area will appear normally open contact which is undefined, you can click the mark to enter the address.

Normally closed contacts: Click to select the normally closed contact, in the program editing area will appear normally closed contact which is undefined, you can click the

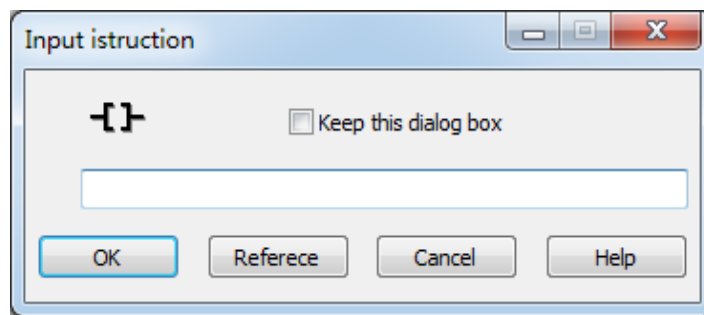
mark to enter the address.

Rising edge contact: Click to select the rising edge of the contact, enable input will lead to a scan cycle.

Falling edge contact: Click to select the falling edge contact, when the enable input is disconnected, it will lead to a scan cycle.

Output coil: The output coil must be at the end of each line. Write the new value of the output bit to the output image register.

Function block: Click the function block, the interface is as follows.



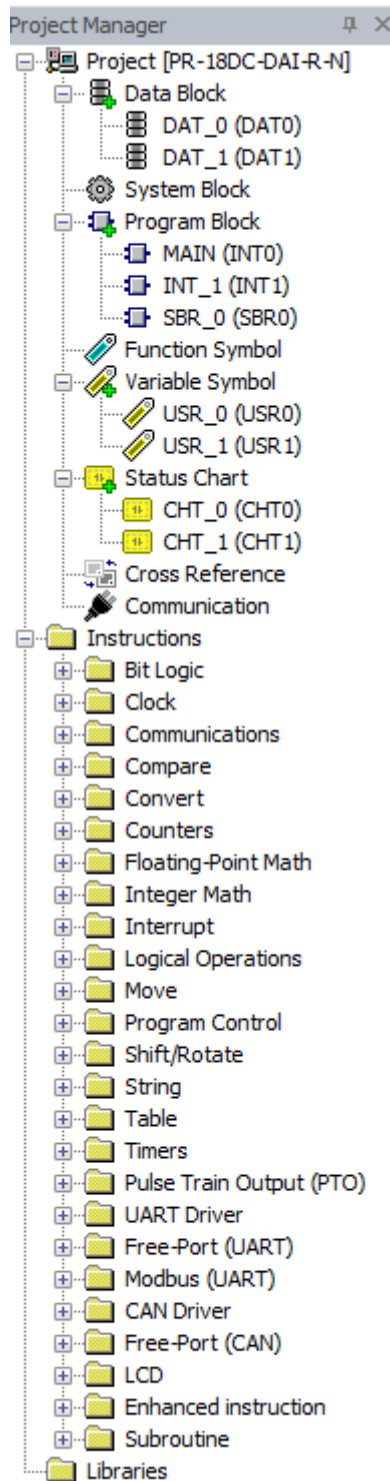
Enter the required function block instruction in the dialog box, letters are capital.

Horz Line: Place a horizontal connector line at the location selected by the mouse.

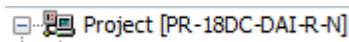
Vert Line: Place a vertical connecting line at the mouse selected location.

Invert Line: Place a horizontal connecting line with the NOT function at the location selected by the mouse.

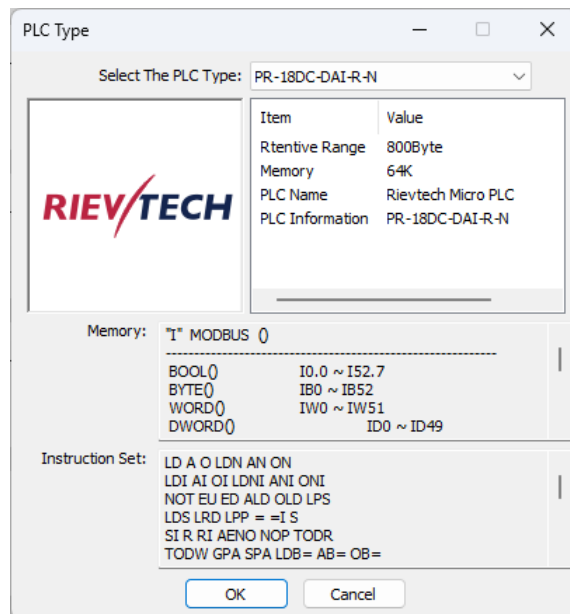
5.3 Instruction tree



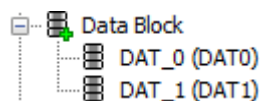
5.3.1 Project



The PLC model selected by the current program is displayed here. Double-click the PLC model here to pop up the 'PLC Type' window.



5.3.2 Data block

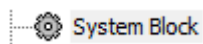


Data blocks contain DAT-0 and DAT-1, you can right-click to insert new data blocks for programmers to use. The contents of the data block are as follows:

| | Adress | Data Type | Value | Comment |
|--|--------|-----------|-------|---------|
| | | BOOL | | |
| | | BOOL | | |
| | | BOOL | | |
| | | BOOL | | |

In the data block, you can set the address, data type, data value, and annotation. The contents of the data blocks are written to PLC after a permanent save, unless a new program is written in PLC. The content in the block of data is written to the PLC and it will be permanently preserved, unless a new program is written in PLC.

5.3.3 System block



Double click the system block, pop the following interface:

The 'System Block' configuration window is shown with the 'RS232/RS485' tab selected. It contains settings for COM 0 and COM 1. A 'Defaults' button is located at the top right of the configuration area. The settings for both COM 0 and COM 1 are identical: Protocol is 'Modbus', Station number is '1' (range 0... 255), Baud rate is '9600 bps', Data bits is '8 (RTU)', Parity is 'NONE', Stop bits is '1 Bit', Response timeout (100ms) is '10' (range 0... 255), and Interval frame delay (B) is '10' (range 0... 255). A note at the bottom states: 'Configuration parameters must be downloaded before they take effect'. The window has 'OK' and 'Cancel' buttons at the bottom right.

| | COM 0 | COM 1 |
|--------------------------|----------|---------------------|
| Protocol: | Modbus | Modbus |
| Station number: | 1 | 1 (range 0... 255) |
| Baud rate: | 9600 bps | 9600 bps |
| Data bits: | 8 (RTU) | 8 (RTU) |
| Parity: | NONE | NONE |
| Stop bits: | 1 Bit | 1 Bit |
| Response timeout (100ms) | 10 | 10 (range 0... 255) |
| Interval frame delay (B) | 10 | 10 (range 0... 255) |

Configuration parameters must be downloaded before they take effect

RS232 / RS485 interface: All ports are using MODBUS communication protocol. You can set four ports, they are: COM 0, COM 1, COM 2 and COM 3.

COM0 corresponds to the programming port. COM1 corresponds to the RS485 port of PR-RS485/SR-RS485. COM2 and COM3 correspond to the PLC's built-in RS485 port. You can set the station number, baud rate, data bit, stop bit, parity, timeout and frame interval time.

Password interface:

System Block

RS232/RS485 RS232/RS485 Password Retentive Ranges Interrupt Time Force Table AI Parameters Bacnet Parameters

Defaults

Privileges

☒ Level 1 - Full

☐ Level 3 - Minimum

☐ Level 4 - Disallow Upload

Password

Verify

Full Privileges: All PLC functions are available without restriction.

Configuration parameters must be downloaded before they take effect

OK Cancel

Password has 3 levels.

Level 1-Full: All PLC functions are available without restriction.

Level 2-Minimum: You have to enter a password before using each function of PLC.

Level 3-Disallow Upload: You can't upload the PLC program. Then you have to enter a password before using each function of PLC.

The length of the password is 1 to 16 bits.

If you forget the password in the PLC, you can switch the PLC to FBD mode (for PR Ethernet series PLCs, use Rievtech to download an FBD program). Then switch back to ladder mode and re-download a new xLadder program.

Retentive Ranges interface

| Range | Data Area | Offset | Number of Elements | Clear |
|---------|-----------|--------|--------------------|-------|
| Range 0 | VB | 0 | 12 | Clear |
| Range 1 | VD | 12 | 10 | Clear |
| Range 2 | VW | 52 | 296 | Clear |
| Range 3 | T | 0 | 32 | Clear |
| Range 4 | C | 0 | 16 | Clear |
| Range 5 | MW | 0 | 30 | Clear |

☒ Clear Memory ☒ Clear EEPROM

Configuration parameters must be downloaded before they take effect

OK Cancel

By default, all M, T, V, and C storage areas are set to remain. You can redefine the scope and set some storage areas to non - hold. You can define the six holding range, select the storage area you want to keep. You can define the address holding range in the following storage areas. As the following: V, M, C, and T. For timers, only the memory timer (TONR) can be kept, and only the current value of the timer and the counter can be kept. Timer and counter bits are cleared

All the variables in the retentive ranges are saved permanently. PLC can hold up to 800 bytes.

| Range | Data Area | Offset | Number of Elements | Clear |
|---------|-----------|--------|--------------------|-------|
| Range 0 | VW | 100 | 400 * 2 | Clear |
| Range 1 | VB | 0 | 0 * 1 | Clear |
| Range 2 | T | 0 | 0 * 2 | Clear |
| Range 3 | T | 64 | 0 * 2 | Clear |
| Range 4 | C | 0 | 0 * 2 | Clear |
| Range 5 | MB | 14 | 0 * 1 | Clear |

☒ Clear Memory ☒ Clear EEPROM

Max:800

OK Cancel

Interrupt time parameter setting interface:

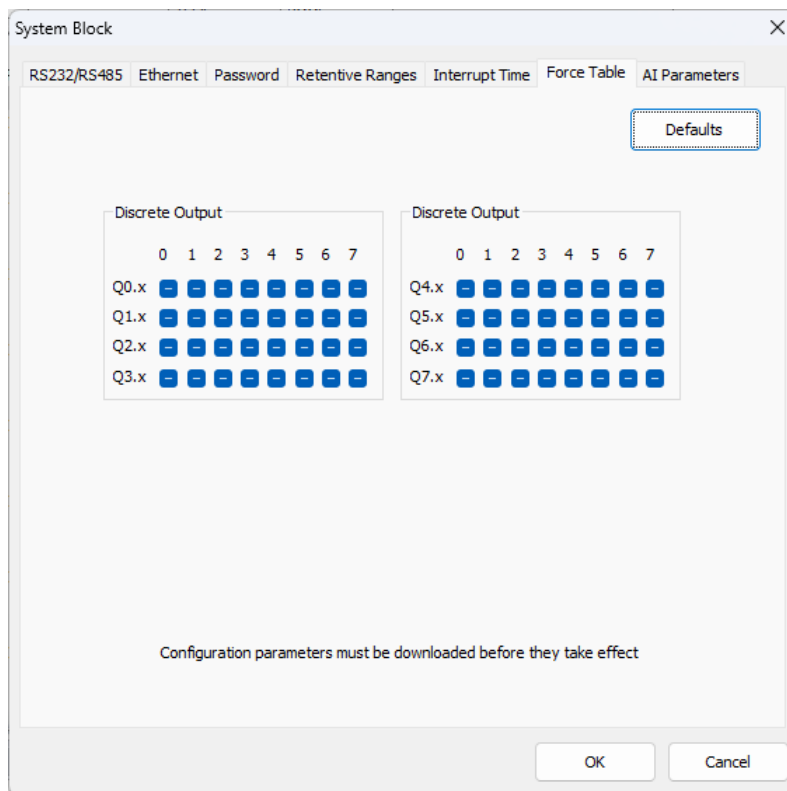
The screenshot shows a software window titled "System Block" with a close button (X) in the top right corner. The window contains a tabbed interface with the following tabs: RS232/RS485, Ethernet, Password, Retentive Ranges, Interrupt Time (selected), Force Table, and AI Parameters. A "Defaults" button is located in the top right area of the main content pane. The main content area displays two configuration items:

| Parameter | Value | Range |
|------------------------|-------|-------------------|
| Time Interrupt 0 (1ms) | 200 | (Range 1 ... 255) |
| Time Interrupt 1 (1ms) | 200 | (Range 1 ... 255) |

At the bottom of the window, there is a message: "Configuration parameters must be downloaded before they take effect". Below this message are two buttons: "OK" and "Cancel".

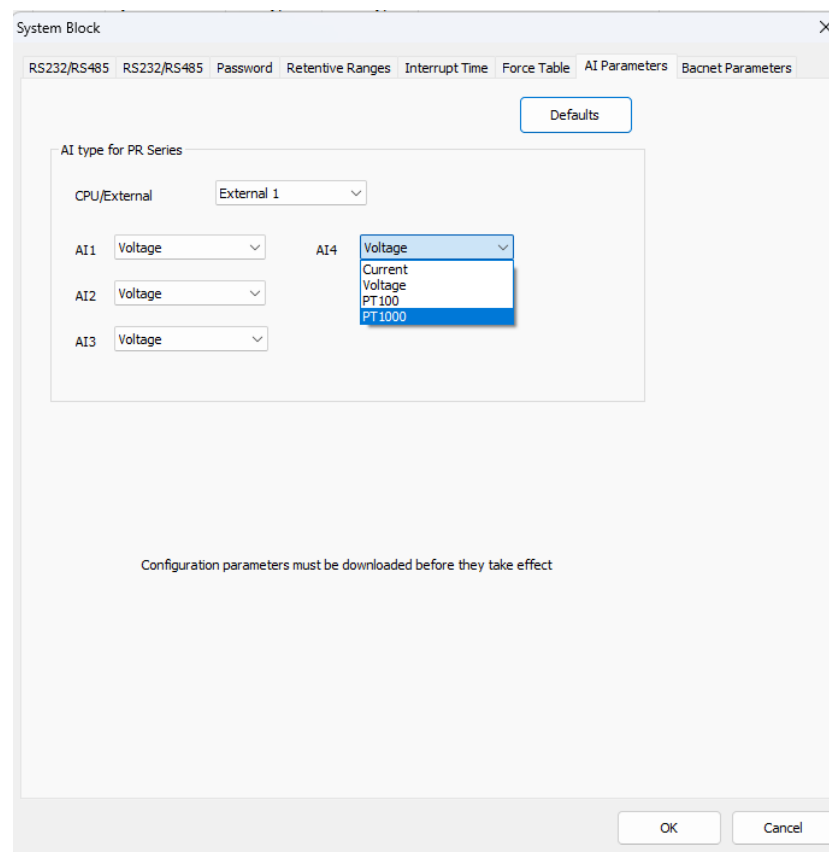
There are 2 time-interrupt events. Respectively, the time of the interrupt event 1 and the time of the interrupt event 0. The interrupt time you can set is 1 to 255 milliseconds.

Force table interface:



When PLC is converted from RUN mode to STOP mode, the selected output points will be 1.

AI Parameters:



CPU module:

PR-18N (I7 -- I8), PR-26N (ID -- IG)

Expansion module:

PR-E-AI-V/I (AI1 – AI4)

PR-E-PT100/PT1000

SR-E-AI-VI

Set the analog channel working mode (voltage or current, PT100 or PT1000) on the software. After setting, you need to download the program to the PLC for the setting to take effect.

Bacnet Parameters:

System Block

RS232/RS485 RS232/RS485 Password Retentive Ranges Interrupt Time Force Table AI Parameters **Bacnet Parameters**

☒ Enable Defaults

Device Information Comm Object(BV) Object(AV)

Description xLadder_bacnet_test

Location rievtech

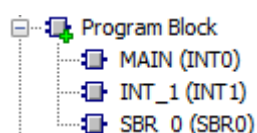
My Object Name: RIEVTECH_PLC

Configuration parameters must be downloaded before they take effect

OK Cancel

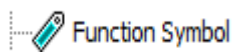
The PR Ethernet series supports the Bacnet protocol and has requirements for the firmware version number. For specific usage, refer to Chapter [Bacnet protocol function](#).

5.3.4 Program block



The program block contains three parts, namely, MAIN (main program), INT-1 (interrupt routine) and SBR-0 (subroutine). Check the interrupt program, right click to add or delete interrupt program. Check the subroutine, right click to add or delete subroutine. The main program can't be added or deleted.

5.3.5 Function symbol



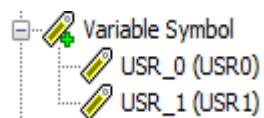
Double click the function symbol, the pop-up interface is as follows:

| | Symbol | Address | Comment |
|---|--------|---------|---------|
| ✓ | MAIN | INT0 | |
| ✓ | INT_1 | INT1 | |
| ✓ | SBR_0 | SBR0 | |

You can modify the symbols, addresses, and comments.

| | Symbol | Address | Comment |
|---|--------|---------|---------------------|
| ✓ | zcx | INT0 | zhu cheng xu |
| ✓ | zdcx | INT1 | zhong duan cheng xu |
| ✓ | zcx | SBR0 | zi cheng xu |

5.3.6 Variable symbol



Double click the variable symbol, the pop-up interface is as follows:

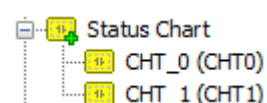
| | Symbol | Address | Data Type | Comment |
|--|--------|---------|-----------|---------|
| | | | BOOL | |
| | | | BOOL | |
| | | | BOOL | |
| | | | BOOL | |

Symbol, address, data type, and comment can be set in variable symbol.

| | Symbol | Address | Data Type | Comment |
|---|--------|---------|-----------|----------|
| ✓ | start | I0.0 | BOOL | qi dong |
| ✓ | stop | I0.1 | BOOL | ji ting |
| | alarm | Q0.0 | BYTE | bao jing |
| | | | BOOL | |

When the address and data types do not match, the address is red.

5.3.7 Status chart



Double click the status chart, the pop-up interface is as follows:

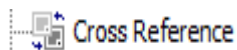
Status Chart

| | Address | Data Type | Value | Forced | Address | Data Type | Value | Forced |
|--|---------|-----------|-------|--------|---------|-----------|-------|--------|
| | SMB10 | SINT | 0 | | V700.0 | BOOL | 1 | |
| | VW52 | INT | 296 | | C15 | INT | 28271 | |
| | VW100 | INT | 296 | | | | | |
| | VW450 | INT | 296 | | | | | |
| | VW452 | INT | 296 | | | | | |
| | VW640 | INT | 296 | | | | | |
| | VW642 | INT | 296 | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | T0 | INT | 27757 | | | | | |
| | T30 | INT | 27757 | | | | | |
| | T31 | INT | 27757 | | | | | |
| | C0 | INT | 28271 | | | | | |
| | C14 | INT | 28271 | | | | | |

CHT_0 (CHT0)CHT_1 (CHT1)

In the status chart, you can set the address, data type, value, and forced.

5.3.8 Cross reference

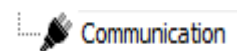


The cross reference displays the address, symbol, location, and context, the interface is as follows:

| Program Editor | | Cross Reference | |
|----------------|--------|------------------------------------|------------|
| Address | Symbol | Location | Context |
| I0.0 | | MAIN (INT0) NETWORK 0 Col 0 Row 0 | - - |
| I0.1 | | MAIN (INT0) NETWORK 0 Col 0 Row 1 | - - |
| I0.2 | | MAIN (INT0) NETWORK 0 Col 0 Row 2 | - - |
| I0.3 | | MAIN (INT0) NETWORK 0 Col 0 Row 3 | - - |
| I0.4 | | MAIN (INT0) NETWORK 0 Col 0 Row 4 | - - |
| I0.5 | | MAIN (INT0) NETWORK 0 Col 0 Row 5 | - - |
| I0.6 | | MAIN (INT0) NETWORK 0 Col 0 Row 6 | - - |
| I0.7 | | MAIN (INT0) NETWORK 0 Col 0 Row 7 | - - |
| Q0.0 | | MAIN (INT0) NETWORK 0 Col 1 Row 0 | -() |
| Q0.1 | | MAIN (INT0) NETWORK 0 Col 1 Row 2 | -() |
| Q0.2 | | MAIN (INT0) NETWORK 0 Col 1 Row 4 | -() |
| Q0.3 | | MAIN (INT0) NETWORK 0 Col 1 Row 6 | -() |
| M10.0 | | SBR_0 (SBR0) NETWORK 0 Col 2 Row 0 | -[LCD_KEY] |
| MW0 | | SBR_1 (SBR1) NETWORK 0 Col 2 Row 8 | -[FILL_N] |

Using cross reference does not require compilation.

5.3.9 Communication



Set the PLC communication, the setting interface is as follows:

Communication

Serial Port Modbus TCP/IP

Search Default

Station: 0

Port: MOXA Communication Port 2 (COM3)

Bus Parameters

Baud Rate: 9600 bps

Parity: NONE

Stop Bit: 1 Bit

OK Cancel

xLadder can be connected to PLC via serial port and Ethernet.

Serial Port: Select the COM port on the PC that RS232 Cable, USB Cable or RS485 is used for. Then set the same communication parameters as the PLC side: baud rate, checksum, stop bit, station number.

Modbus TCP/IP: xLadder is used as client. Fill in the PLC IP and server port here. If the PC has multiple network cards or network segments, you need to select the network adapter that is in the same network segment as the PLC IP.

And fill in the PLC station number.

Communication

Serial Port Modbus TCP/IP

Default

Station: 0

Adapter: 192.168.0.28 (Intel(R) Ethernet Connection (14) I219-V)

IP Address: 192.168.0.223

Port: 8008

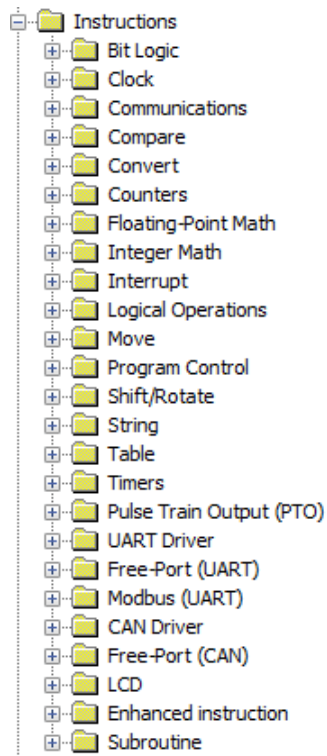
OK Cancel

If you don't remember the PLC station number, you can use 0.

The new version of xLadder can save the last set communication parameters and use

them directly the next time you open it.

5.3.10 Instructions



Instructions will be explained in detail in the [instructions section](#).

5.3.11 Libraries



The new version of xLadder supports exporting subroutines as library files, and supports importing saved library files into xLadder.

For details, refer to Chapter [library file function](#).

5.3.12 The program editor

The screenshot displays the 'Program Editor' window. At the top, there is a table with the following columns: 'Symbol', 'Var Type', 'Data Type', and 'Comment'. The table contains three rows of data:

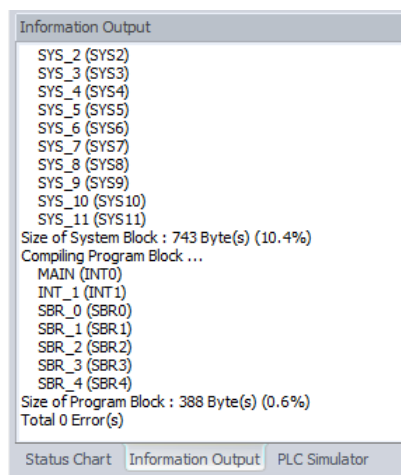
| | Symbol | Var Type | Data Type | Comment |
|-------|--------|----------|-----------|---------|
| ✓ LW0 | IN1 | IN | WORD | |
| ✓ LW2 | IN2 | IN | WORD | |
| ✓ LW4 | OUT1 | OUT | WORD | |

Below the table, the main editing area shows 'NETWORK 0'. It contains a ladder logic diagram with a normally open contact labeled 'SM0.0' connected to the 'EN' (Enable) input of a function block labeled 'ADD_I'. The function block has two inputs: 'IN1' and 'IN2'. 'IN1' is connected to a variable labeled '#IN1 (LW0)' and 'IN2' is connected to a variable labeled '#IN2 (LW2)'. The output of the function block is labeled 'OUT' and is connected to a variable labeled '#OUT1 (LW4)'. Below 'NETWORK 0' is 'NETWORK 1', which is currently empty. At the bottom of the window, there is a tabbed interface with the following tabs: 'MAIN (INT0)', 'INT_1 (INT1)', 'SBR_0 (SBR0)', 'SBR_1 (SBR1)', 'SBR_2 (SBR2)', 'SBR_3 (SBR3)', and 'SBR_4 (SBR4)'. The 'MAIN (INT0)' tab is currently selected.

Local Variable table: MAIN, each INT and SBR subroutine has its own local variable table. You can define local variables here. Local variables can be used to pass parameters to subroutines, which enhances the portability and reusability of subroutines. The scope of a local variable is only within this subroutine.

Program editing area: In the program editing area, the main program, interrupt program and subroutine can be edited.

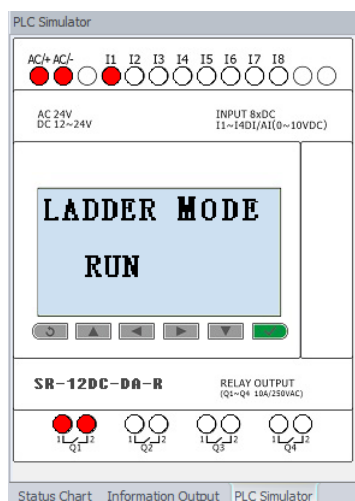
5.3.13 Status chart, information output



Status Chart: [5.3.7 chapter](#).

Information Output: Displays information about the compiled program. The output information window keeps a list of errors generated during compilation. When program modification is completed, compile the program again.

PLC Simulator: During offline simulation, a picture of the PLC is displayed here. The buttons on the LCD panel in the picture are available, the input can be used to control and indicate the current state, and the output can be used to indicate the state change of the output. For detailed information, please refer to Chapter [Offline simulation](#).



5.4 Programming concepts

5.4.1 How the program works

The program is run by the loop, PLC reads and writes data continuously. When you download the program to PLC and make PLC in the run mode, the PLC' s central processing unit (CPU) executes the program in the following order:

A: PLC read input status.

B: The PLC program uses input values for logic control.

C: When the program is running and writes the results to the output image register.

D: At the end of the program, the output value of the output image register.

E: repeat the above steps.

PLC performs a series of tasks repeatedly. The cycle execution task is called the scan cycle. PLC performs most or all of the following tasks during the scan cycle:

A: PLC read input status.

B: PLC executes the program's instructions, and stores the data in different memory areas.

C: performs all communication requests.

D: PLC performs CPU self-test diagnostic program. PLC ensure that the hardware, program memory and all expansion modules are normal operation.

E: The values stored in the output image register are written to the actual output.

Attention:

(1) The execution of the scan cycle depends on the PLC that is set in the STOP (stop) mode or the RUN (run) mode. In the RUN (run) mode, the program is executed; in the STOP (stop) mode, the program is not executed.

(2) The execution cycle time of the PLC is related to the size of the program. When the program is large, the running cycle will be longer.

5.4.2 Addressing overview

Identifying absolute and symbolic address

You can use absolute or symbol to identify the instructions in the program. Absolute reference use memory area and bit or byte location to identify the address. Symbolic

reference uses letters, numbers, and characters to identify addresses or values.

How to display the address of the program editor:

I0.0

The absolute address is made up of the memory area and the number of addresses.

#INPUT1

symbols in a local variable before

INPUT1

Global symbol name

??.? or ????

A question mark indicates an undefined address (which must be defined before the program is compiled).

Global scope and local scope

The symbol value in the symbol table has a global scope, and the symbol value in the local variable table has a local scope.

Global symbol

Global symbols can be used in the xLadder program editor.

In the xLadder program, you can use the global variable table to assign the global symbol.

local variable

Local variables can be used in the xLadder program editor.

Local variables are assigned in the local variable table of the respective POU, and the scope is limited to the POU of the local variable. Each POU has a separate local variable table.

Attention: If you use the same address name in the local and global variables table, local variables are preferred.

Local variables use temporary L memory, and do not require the PLC program memory space. The subroutines that use only the local variable parameters (or don't use the parameters) are mobile subroutines. They can be used in more than one

program. If you want to use a parameter in a plurality of POU, it is best to define it as a global symbol in the global variable table, and do not define it as a local variable, or you must assign each POU 's local variable table separately. Because local variables use temporary memory, every time POU is called, be sure to initialize local variables in POU. Global symbol table supports global symbol constant. Local variable table does not support symbolic constants.

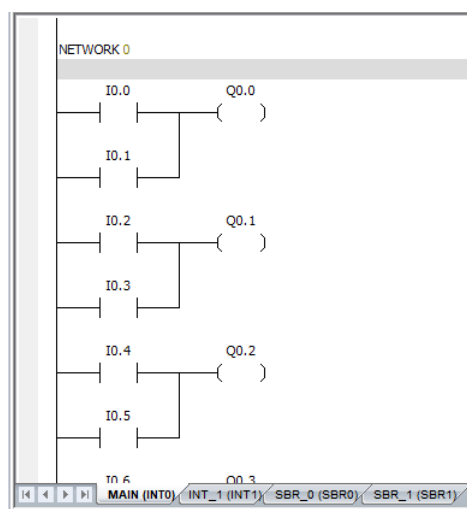
5.4.3 How to organize the program

Basic elements of a control program

The program consists of the following program types:

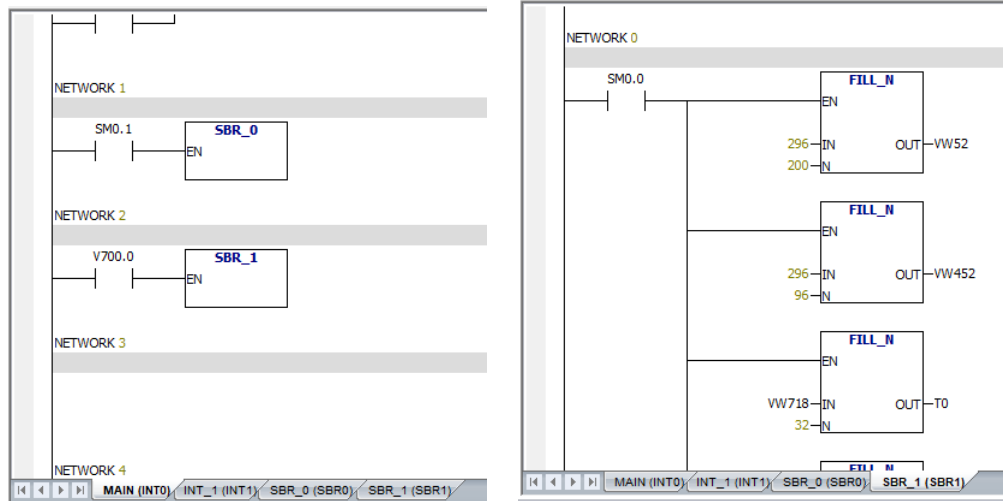
MAIN program

The main body of the program is where you place the control application instructions. The instructions in the main program are executed in sequence, and each scan cycle is executed once.



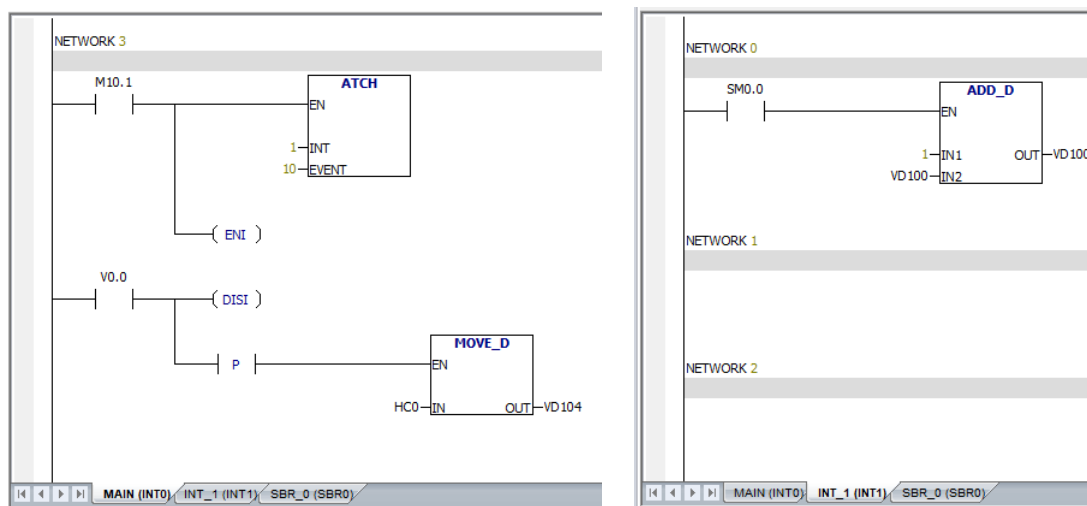
Subroutine

Subroutine stored in a separate block, when the main program, interrupt routine or another subroutine call subroutine, the subroutine will be executed.



Interrupt routine

The interrupt routine is stored in a separate block, which is executed only when the interrupt event occurs.



How to terminate POU

The compiler uses unconditional END, MEND, RET, or RETI to terminate each POU. If you put the unconditional END, MEND, RET, or RETI into the program, the compiler will return an error message.

Subroutine

Subroutine is particularly useful when you want to perform a function repeatedly. You just need to write a logic in the subroutine, then you can call the subroutine

every time when you need it in the main program.

Advantages:

1. Your program size becomes smaller.
2. Because you remove the code from the main program, the scan time will be reduced.

Subroutine can be scanned only when it is called. The main program is constantly scanned.

3. Subroutine is easy to be moved; You can select a function and copy it to another program. You don't need or need a little repetitive operation.

Attention: V memory usage limits the portability of the subroutine. Because a program's V memory address assignment may be in conflict with the assignment in another program. Instead, the subroutine which only use local variables is easy to move, because there is no need to worry about addressing conflicts.

Interrupt routine

You can write an interrupt routine to handle some predefined interrupt events: The interrupt routine is not called by the main program. When the interrupt event occurs, it is called by the PLC operating system. Interrupt routine is best to use local variables. You can use a local variable table to ensure that your interrupt routine uses only temporary memory.

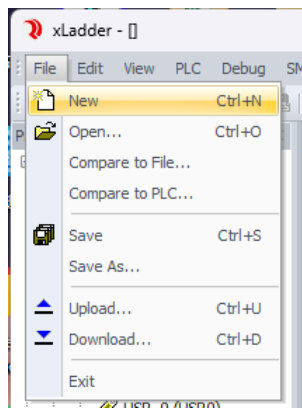
5.5 How to enter the ladder logic program

5.5.1 How to build a new project

Click Click the 'New' con in the toolbar. Or use the 'New' menu in the 'File' menu.

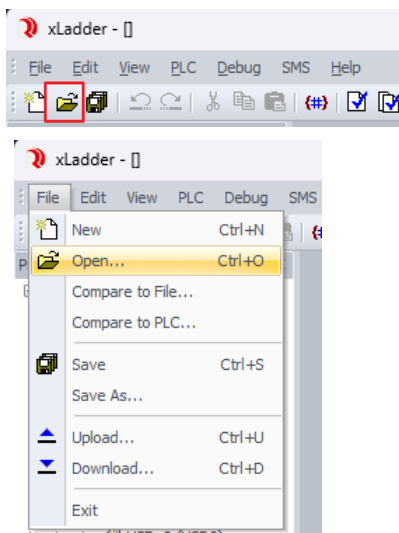
Create a new project.






Open an existing project

Click the 'Open' icon in the toolbar. Or use the 'Open' menu in the 'File' menu.




5.5.2 Ladder logic element and its working principle

Ladder logic (LAD) is a graphical language which is similar to the electrical relay diagram. When you write a program in LAD, you use graphical components and arrange them into a logical network. The following component types are available for use when you build a program:

Contact  the switch which power supply can pass through. When the normally open contact logic is 1 and the normally closed contact logic is 0, the power supply can pass through these contacts.

Coil  The coil represents the output.

Block  Each block represents a function.

The network is composed of the above elements. The power supply from the left side of the power rod flows through the closed contact to charge the coil or the block.

5.5.3 Network rules for series and parallel in LAD

Rules for placing contacts

Each network must begin with a contact.

The network cannot be terminated by contact.

Rules for placing coils

The network can't start with the coil. The coil is used to terminate the logical network. A network may have a number of coils, and the coils are located on a parallel branch of the network. Could not be connected in series with more than one coil in the network

Rules for placing blocks

If the block has ENO, the enable bit can be extended to the out of block. This means that you can place more instructions behind the block. In the network, you can connect in series with a number of boxes with ENO. If there is no ENO in the box, no instruction can be placed on the following.

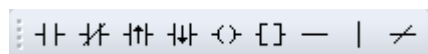
Network size limit

Cell is the area which is placed instruction. In the network, a single network can extend 32 cells Vertically or 32 cells horizontally.

5.5.4 How to input commands in LAD

Line

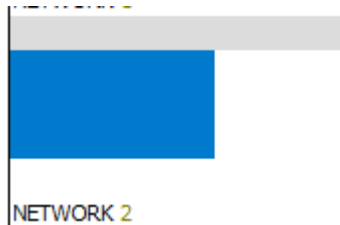
You can use horizontal and vertical lines to connect elements to finish the network.



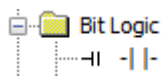
Double click the instruction tree

1.Place the cursor in the position you want to edit in the program editor window.

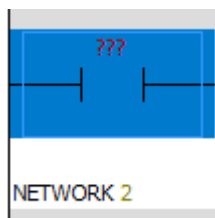
Click the mouse, there will be a selection box.



2.Select the required instruction, double click it.



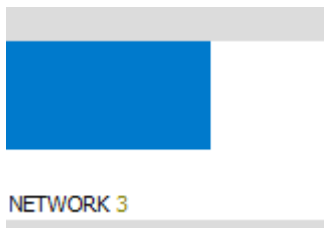
Instruction will appear in the selected editing area.



Use the toolbar button or function key

1.Place the cursor in the position you want to edit in the program editor window.

Click the mouse, there will be a selection box.

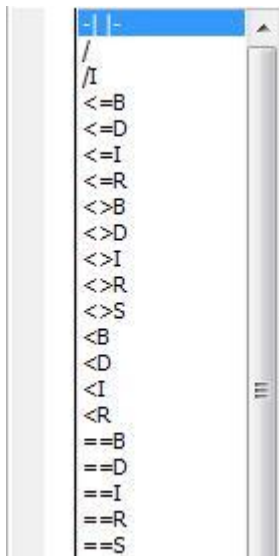


1.Select the required button in the toolbar



Or use the functional keys (F4= contacts, F6= coil, F9= box).

2.The second step is over, there will be a drop-down list. Find the needed instructions in the list. Double click the instruction or use the ENTER key to enter the instruction.



5.5.5 How to enter the address in LAD

When you enter a command in the LAD, the instruction contains question marks. The question mark indicates that the parameter is not assigned. You can assign values to the parameters of the element when you enter the element. If the parameter is not assigned, the program will not be properly compiled.

To specify a symbolic address, you must perform the following simple steps:

1. Enter a symbol or variable name in the address area of the instruction.
2. If it is a global symbol, the symbol table / global variable table is used for Specifying a symbol name to the memory address.

Attention: You can use local variable table at the top of the program editor window. Input symbol name in the "symbol" column. Because the compiler will automatically specify the L memory address, you do not have to enter the address for the local variable. You can drag the edge of the table to minimize the size of the local variable table.

5.5.6 How to edit program elements in LAD

Cut, copy, paste, or delete multiple networks

By dragging the mouse or holding down the shift key with the mouse to select the adjacent networks, you can choose a number of adjacent networks for cutting,

copying, pasting or deleting options.

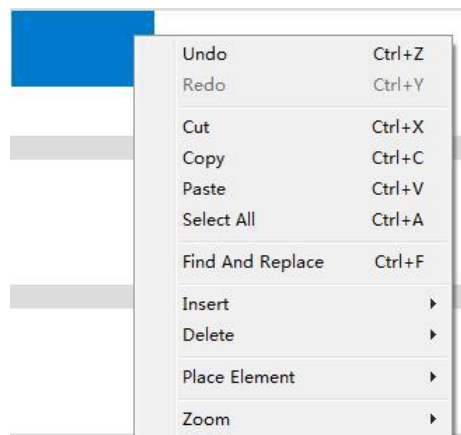
First of all, you should select a project, and then you can use the copy function. The contents of the copy are placed in the Windows clipboard buffer.

You can choose the following objectives in the project:

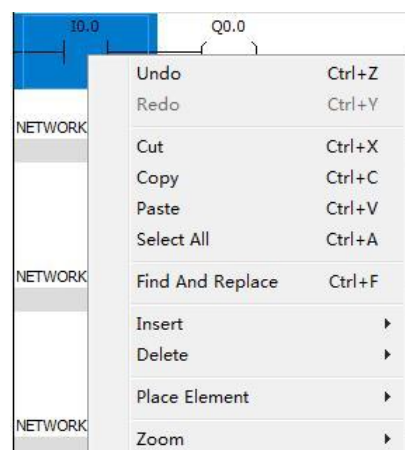
1. Program text or data domain
2. Instructions in the LAD, STL editors
3. Single network
4. Multiple adjacent networks
5. All networks
6. Symbol table, row and column of the symbol table
7. State table, row and column of the state table

Edit cells, instructions, addresses, and networks

1. Select an empty cell, you can use the right key to select the operations as follows:

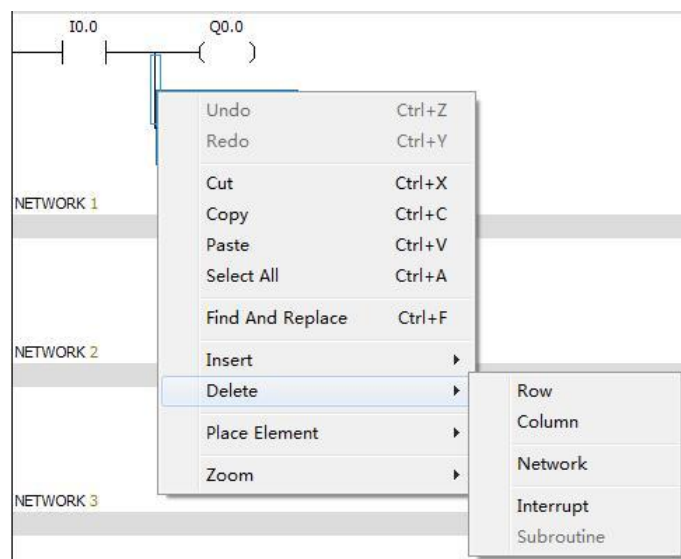


2. Select an instruction, you can use the right key to select the operations as follows:



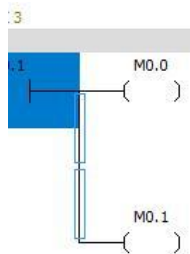
3.You can cut and paste elements and rows, delete rows or columns.

Delete element:



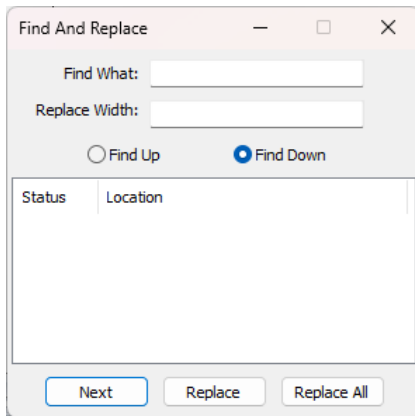
You can use the DELETE or BACKSPACE key to delete the cells. You can select the elements that need to be deleted, use the right key to select the “delete” function component.

Attention: In order to select the vertical line that needs to be deleted, you should use the cursor to select the vertical line.



5.5.7 How to use find / replace

- 1.Select Edit > Find, Edit > replace
2. Use the shortcut key CTRL+F to start the search function.



How to use search and replace function

Search function

1. Enter the string you want to search in the "Find What" field.
2. You can use the "Find up" and "Find down" functions.

Replacement function

1. Enter the string you want to search in the "Find What" field.
2. Enter the string you want to replace in the "Replace With" field.
3. To find the next string, click the "Next" button.
4. If you want to replace the string, click "Replace". If you want to replace all of the characters, click "Replace All".

Where to use

You can use the "find" and "replace" in the program editor window.

How them works

1. The "find" function allows you to search strings, such as the operation of network number, title, instruction mnemonic or register name. ("Find" function does not search network comments, It just search the network title.)
2. "Replace" function allows you to replace the specified string.

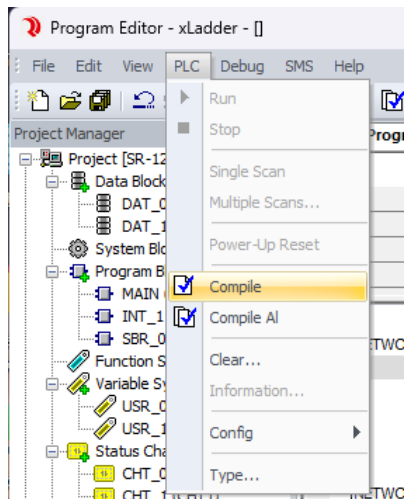
5.5.8 How to display errors in LAD in the program editor


Red words display errors.


Attention: When you replace the invalid value or symbol with a valid value, the font is automatically changed to the default font color.

5.5.9 How to compile in LAD

You can use the toolbar button or the "PLC" menu to compile.



"Compile"  Allows you to compile a single element of the project. When you select "compile", the current window is compiled and the other windows are not compiled.

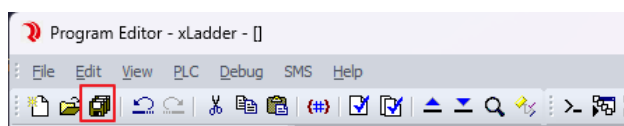
"All compile"  Compiles the program editor, system block, and data block. When you use the "All compile" command, all windows are compiled.

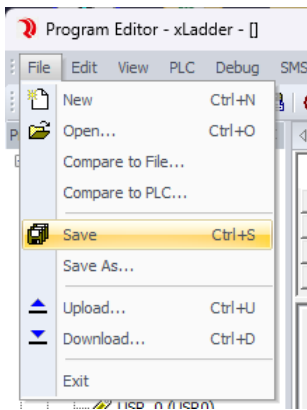
Use the output window to resolve the error

When you compile a program, the output window lists all the errors about the program. Errors include location (network, row and column) and error types.

5.5.10 How to save the project

You can use the 'Save' button in the toolbar or the 'Save' menu under the 'File' menu in the menu bar to save the program. Its shortcut key is 'CTRL+S'.





"Save" allows you to save all changes quickly in your project.

"Save as" allows you to change the name of the current project and the location of the directory.

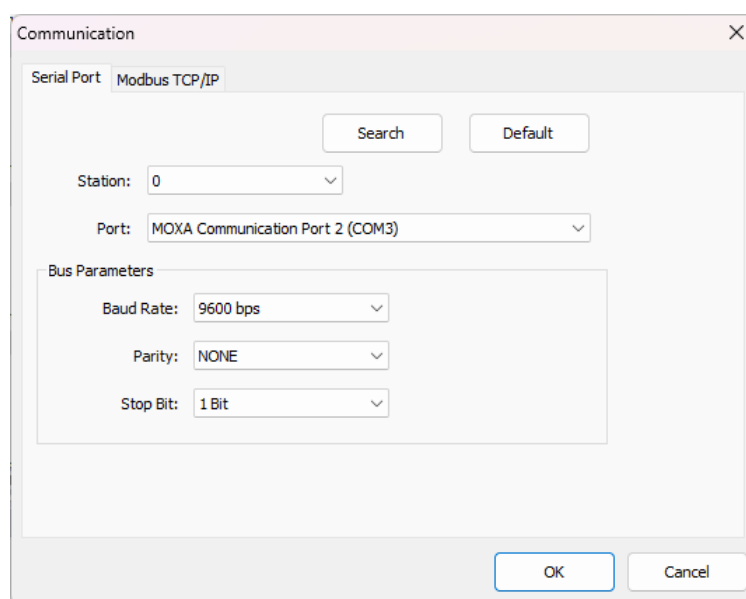
5.6 How to set up a communication and download program

5.6.1 Communication settings

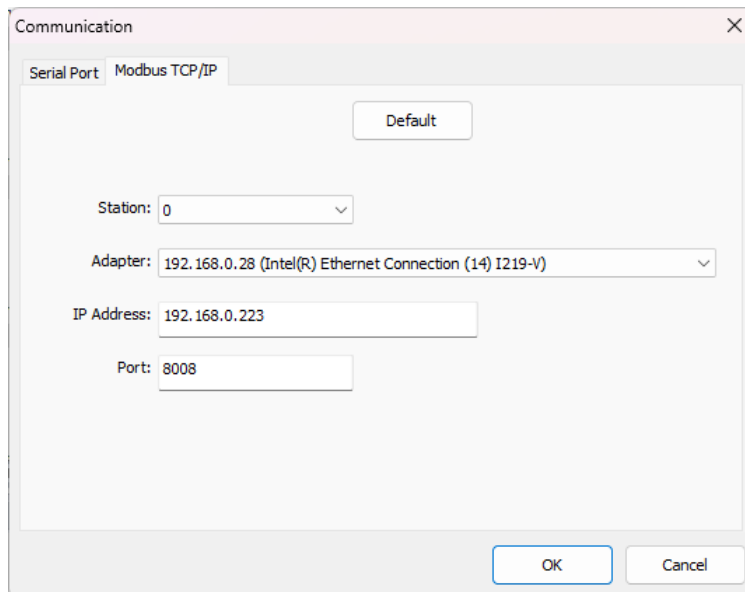
1. Select the communication method. Some PLC models only support serial port, while some PLC models support both serial port and Ethernet.

Set the communication parameters in 'Communication'. For related content, please refer to [here](#).

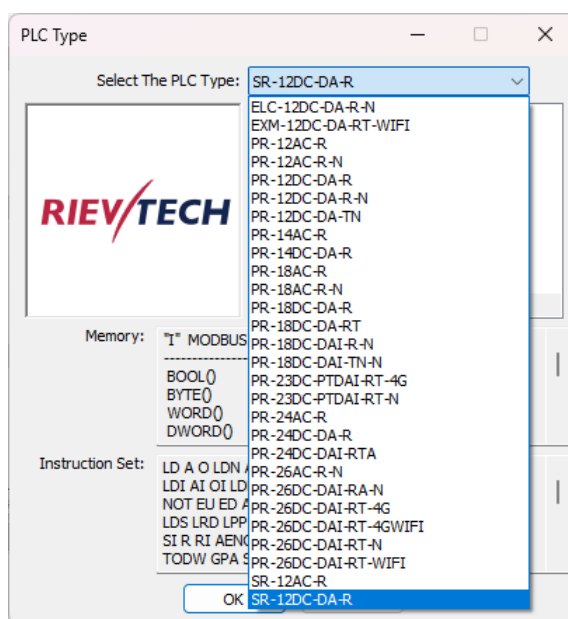
Serial port:



Ethernet:



2. Select the PLC model: Ensure that the PLC model in the software is consistent with the actual PLC model.




5.6.2 Download program

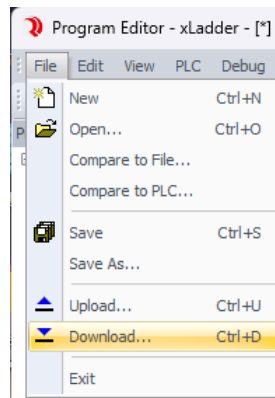
If xLadder and PLC communicate successfully, you can download the program to the PLC. Steps are as follows:

Attention: The new program will cover the old program.

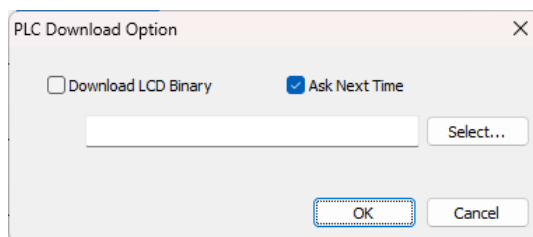
1. Before the program is downloaded to PLC, the program needs to be compiled



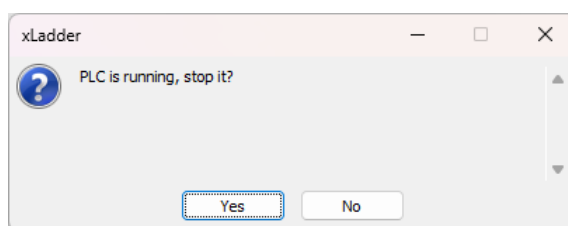
2. After the success of the compiler, click the "download"  button in the toolbar, or select File > download.



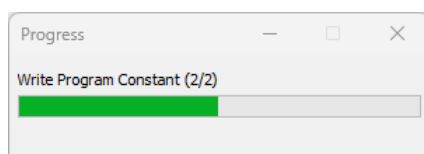
The interface is as follows:



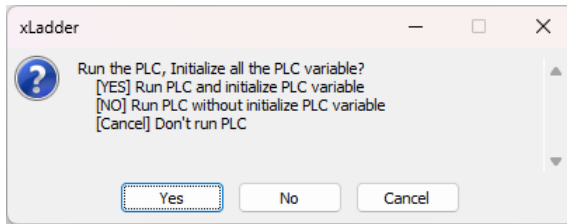
Click OK.






Click Yes, the software will automatically download the program block, the data block and the CPU configurations to the PLC.

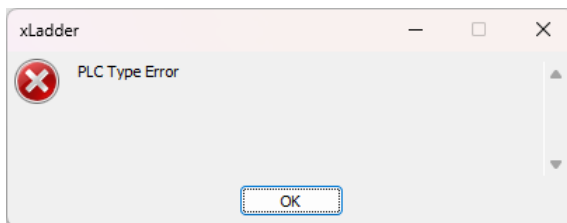


3. When the program is downloaded successfully, the interface is as follows:

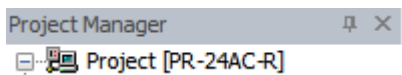


There are three options, you can choose one of them.

4. When you choose “yes” or “no”, you can click on the “connection”  to monitor the program. When you choose “cancel”, PLC is stopped. You can click on the “run” button , then click on the “connection”  to monitor the program.
5. If the type of PLC set in the software is not consistent with the PLC type of the actual connection, the software will display a warning message.



6. You can double click on the project of the project manager to modify the PLC model.

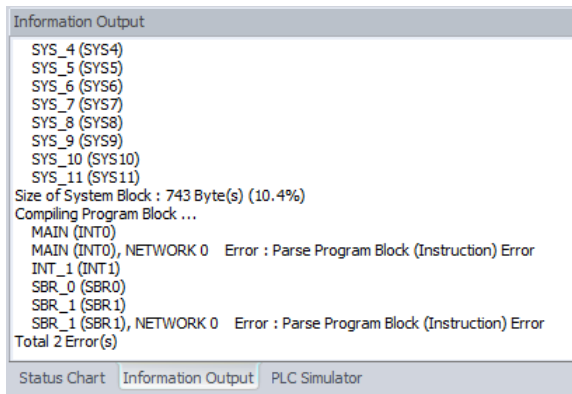


7. Click the “download” button to download the program again.
8. If the program downloads successfully, you can convert the PLC from the STOP mode to the RUN mode to run the program.

5.6.3 How to correct compilation errors and download errors

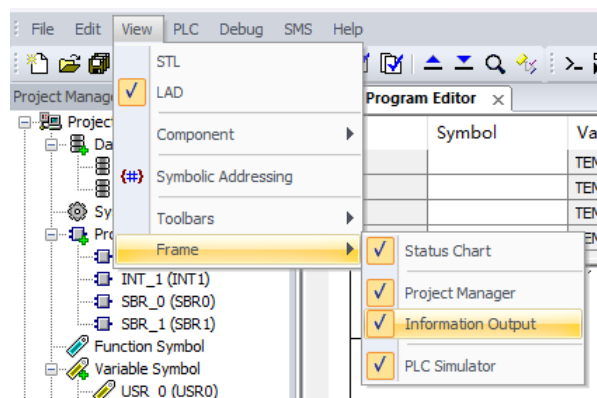
The output window automatically displays program information and error messages at any time when you compile a program or download a program.

The information usually includes the error of the network, the column and row position and the error code and instructions.



Follow the prompts to correct the errors.

If you have closed the output window, select View > Frame > Information output from the menu bar to display the output window again.



5.7 How to monitor the program

After the program is created, you can use the offline simulation button in "debug" to simulate the program. You can also download the program to the PLC and use the online monitoring button to observe the program running in the PLC.

Debug Toolbar:

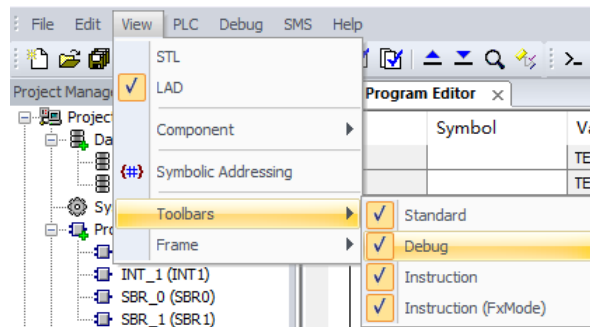
Simulate:

Online monitoring:

Run / Stop:

This chapter does not describe the simulation content of the program, please refer to [here](#) for related content.

If this toolbar is closed, you can open it again using the method shown in the figure below.



What is "state monitoring"?

State monitoring shows the current value of the PLC data and the information of the current state. You can monitor, read, write, and enforce PLC data values by using the status table. When the program runs, there are two ways to view the PLC data dynamics.

Status table monitoring

Displays the data status in the table: you can specify address, data type, value, and forced.

Program status monitoring

Displays data status in the program editor window: The current PLC data value is displayed on the STL statement or LAD graph.

Program status monitor window and status table monitor window can be run simultaneously:

PLC data written or forced in the state table window will be applied to the program status monitor window.

The conditions of Viewing data status

1. xLadder and PLC communicate successfully.
2. Download the program to the PLC successfully.


3. To view the continuous changes of the PLC data state, the PLC must be located in the RUN mode.

4. If the program that you monitor is not implemented, there will not be a state display.

Attention:

If the program downloads successfully, you have to convert the PLC from the STOP mode to the RUN mode to run the program. Because in STOP mode, you will not be able to see the expected results of the program logic operation.

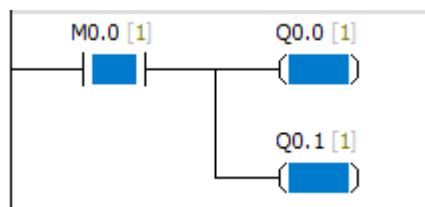
How to view data status

In RUN mode, click on the connection  to monitor the program. Write the address of the data that you want to view in the state table, the status table will show its current value.

The color of execution status:

Contact: When the contact is switched on, the instruction will change the color.

Coil: When the output is switched on, the instruction will change the color.



PLC RUN / STOP mode

Use the following methods to change the PLC operation mode:

1. Click the "run" button to execute the RUN mode. Or click the "stop" button to execute the STOP mode.
2. Select the PLC > run menu command to execute the RUN mode, Or select PLC > stop menu command to execute the STOP mode.
3. Insert a STOP instruction in the program.

Mandatory and cancel the mandatory

Forced

Enter the address and its value that you want to force in the state table. Then select the mandatory function. Before canceling the mandatory, the mandatory function has been effective.

| Status Chart | | | | |
|--------------|---------|-----------|-------|------------|
| | Address | Data Type | Value | Forced |
| | I0.0 | BOOL ▼ | 0 | Unforced ▼ |
| | IB1 | SINT | 0 | |
| | Q0.0 | BOOL | 1 | Forced |
| | | | | |

"Mandatory" function covers "read immediately " and "write immediately" functions.

I/O points can be forced, and other storage areas can't be forced.

Cancel the mandatory

Select "unforced" in the status table to cancel mandatory.

5.8 PLC operation and options

Elements of the control program

Ladder Program

In the LAD program, the basic elements of the logic are represented by contacts, coils, and boxes.

The input is represented by a symbol called a contact. Contact is divided into normally open contact and normally closed contact.

Normally open contact: a contact that is open in nature.

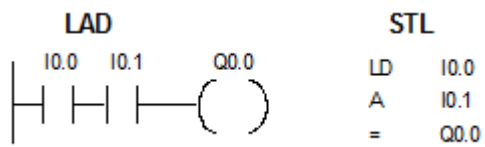
Normally closed contact: a contact that is closed in nature.

The output is represented by a symbol called a coil.

The blocks are function blocks with various functions. The blocks can make programming easier.

STL program

The STL program elements are represented by instructions. Ladder diagram and instructions are as follows:

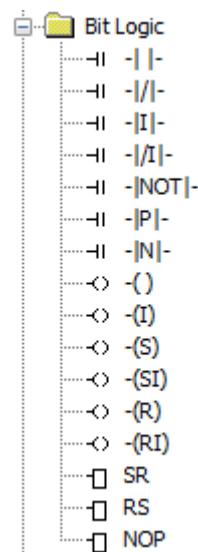


System blocks configuration

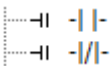
Instructions: The detailed annotation of operation interface--->System blocks

6. xLadder instructions descriptions

6.1 Bit logic



6.1.1 Normally open and normally closed



| Input / output | Operand | Data type |
|----------------|----------------------------|-----------|
| Bit (LAD、STL) | I, Q, M, SM, T, C, V, S, L | Boolean |

Normally open and normally closed

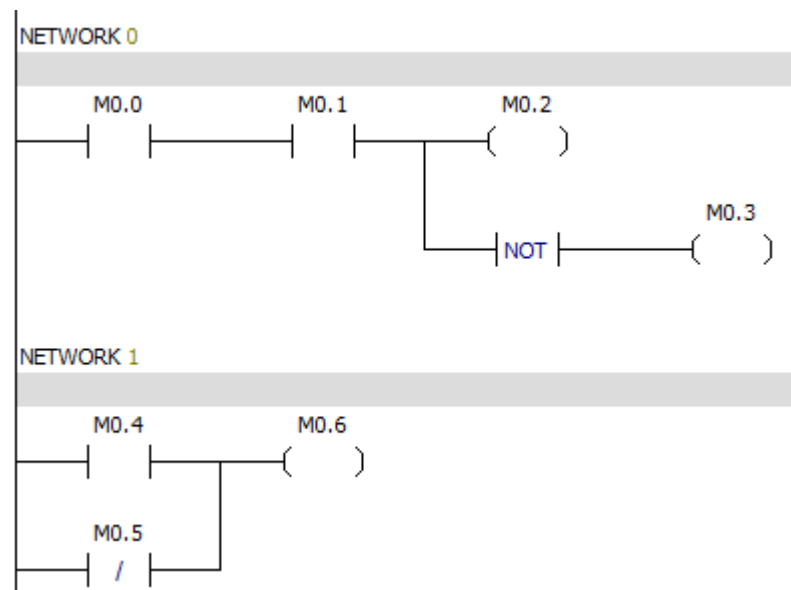
When the bit is equal to 1, the normally open contact is closed, and the normally closed contact is disconnected.

When the bit is equal to 0, the normally open contact is disconnected, and the normally closed contact is closed.

In STL, the normally open contact is represented by "LD", "And" and "Or" instructions.

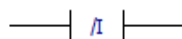
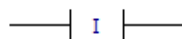
In STL, normally closed contacts are represented by "NOT", "NOT AND" and "NOT OR" instructions.

Example:



6.1.2 Normally open immediate and normally closed immediate.

Normally open immediate and normally closed immediate



When PLC executes the instruction, the immediate instruction obtains the actual input value, but the PLC does not update the process image register.

The immediate contact update does not depend on the PLC scan cycle; it will be updated immediately.

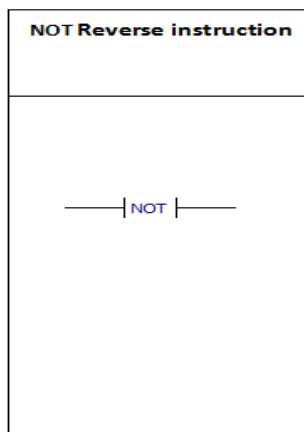
When the actual input point is 1, normally open immediate is closed.

When the actual input point is 0, normally closed immediate is closed.

In LAD, normally open immediate and normally closed immediate instructions are represented by contacts.

Forcing function can't be used for immediate input instructions.

6.1.3 NOT Reverse instruction



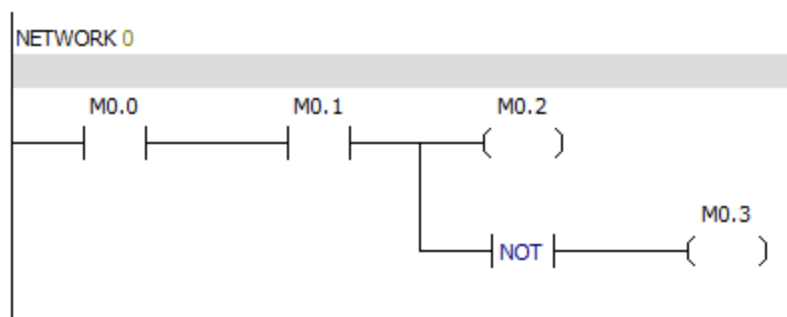
The functions of "NOT instruction" are as follows:

When the input is 0, the output is 1.

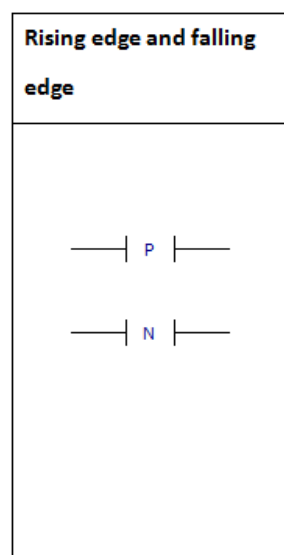
When the input is 1, the output is 0.

In LAD, the NOT instruction is represented by a contact.

Example:



6.1.4 Rising edge and falling edge

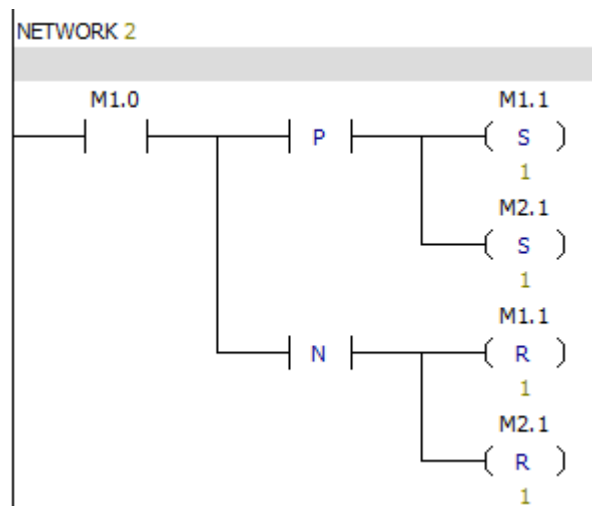


When left logic is converted from 0 to 1, Rising edge contact conduction time is a scan cycle.

When left logic is converted from 1 to 0, Falling edge contact conduction time is a scan cycle.

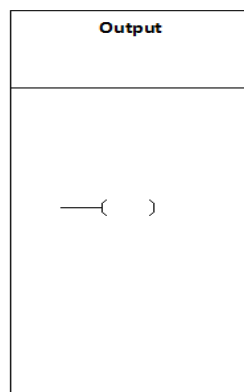
In LAD, the rising edge and the falling edge are represented by the contacts.

Example:



6.1.5 Output

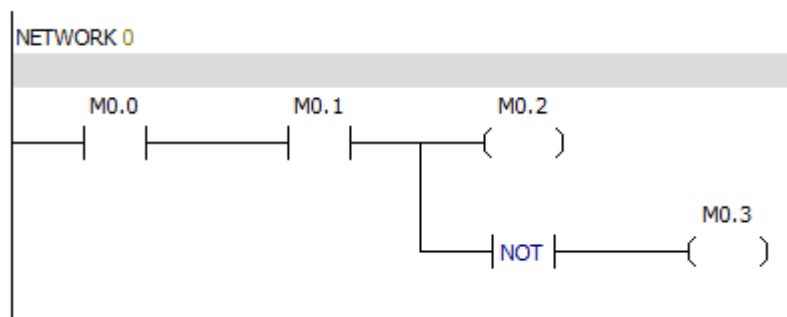
| Input / output | Operand | Data type |
|----------------|----------------------------|-----------|
| Bit | I, Q, M, SM, T, C, V, S, L | Boolean |
| Input (LAD) | Enable bit | Boolean |



The output instruction writes the new value of output bit to process image register.

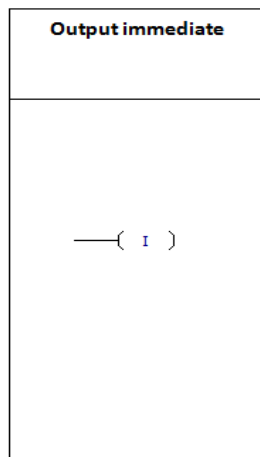
In LAD, when the output instruction is executed, the PLC will open or close the output bit in the process image register.

Example:



6.1.6 Output immediate

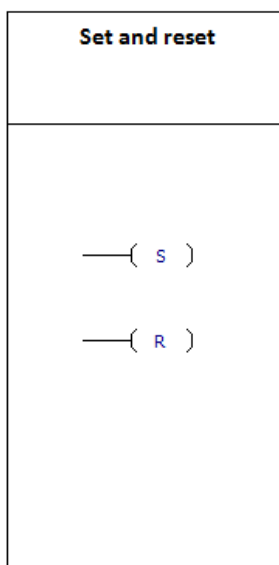
| Input / output | Operand | Data type |
|----------------|------------|-----------|
| Bit | Q | Boolean |
| Input (LAD) | Enable bit | Boolean |



The new value generated by executing the immediate output instruction is written to the actual output and the corresponding process image register.

6.1.7 Set and reset

| Input / output | Operand | Data type |
|----------------|--|-----------|
| Bit | I, Q, M, SM, T, C, V, S, L | Boolean |
| N | VB, IB, QB, MB, SMB, SB, LB, AC, constant, *VD, *AC, *LD | Byte |



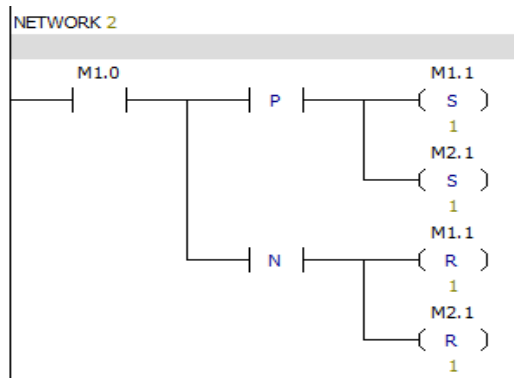
"Set" instruction can make a bit or a series of bits be 1.

"Reset" instruction can make a bit or a series of bits be 0.

The value of N is between 1 and 255.

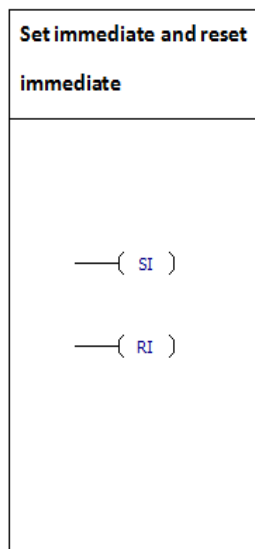
The "reset" instruction can reset the bits of the timer and counter, and can clear the current value of the timer and counter.

Example:



6.1.8 Set immediate and reset immediate

| Input / output | Operand | Data type |
|----------------|--|-----------|
| Bit | Q | Boolean |
| N | VB, IB, QB, MB, SMB, SB, LB, AC, constant, *VD, *AC, *LD | Byte |



“Set immediate” can set many of points immediately.

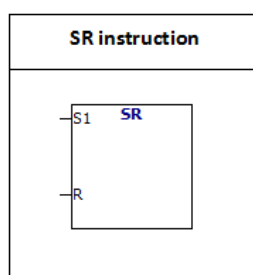
“Reset immediate” can reset many of points immediately.

The value of N is between 1 and 128.

"I" means "reference immediately". The new value generated by executing the instruction is written to the actual output and the corresponding process image register.

6.1.9 SR instruction

| Input / output | Operand | Data type |
|----------------|---------------|-----------|
| S1, R | Enable bit | Boolean |
| OUT | Enable bit | Boolean |
| xxx | I, Q, M, V, S | Boolean |



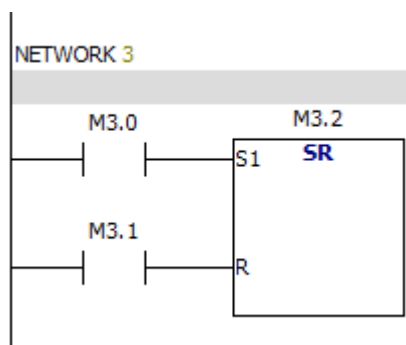
Bistable trigger is a latch. When both R and S1 are equal to 1, the

output is 1.

The truth table of the "SR" instruction is as follows:

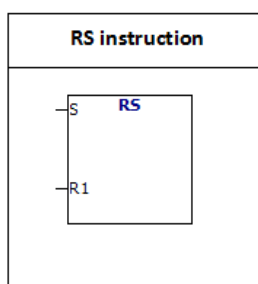
| Instruction | S1 | R | OUT |
|-------------|----|---|----------------|
| SR | 0 | 0 | Previous state |
| | 0 | 1 | 0 |
| | 1 | 0 | 1 |
| | 1 | 1 | 1 |

Example:



6.1.10 RS instruction

| Input / output | Operand | Data type |
|----------------|---------------|-----------|
| S, R1 | Enable bit | Boolean |
| OUT | Enable bit | Boolean |
| xxx | I, Q, M, V, S | Boolean |



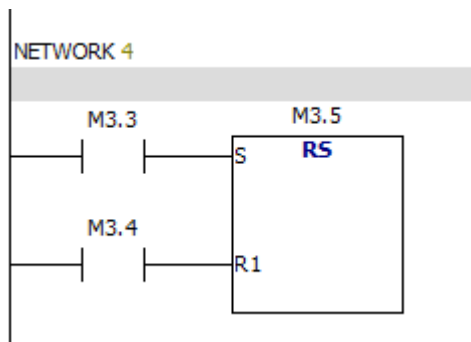
Bistable trigger is a latch. When both R1 and S are equal to 1, the output is 0.

The truth table of the "RS" instruction is as follows:

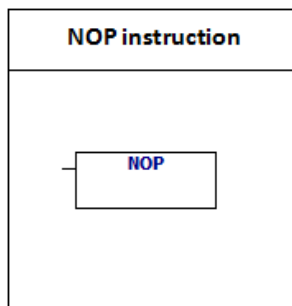
| Instruction | S | R1 | OUT |
|-------------|---|----|----------------|
| RS | 0 | 0 | Previous state |
| | 0 | 1 | 0 |
| | 1 | 0 | 1 |

| | | | |
|--|---|---|---|
| | 1 | 1 | 0 |
|--|---|---|---|

Example:

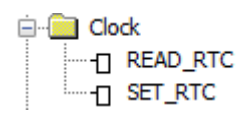


6.1.11 NOP instruction



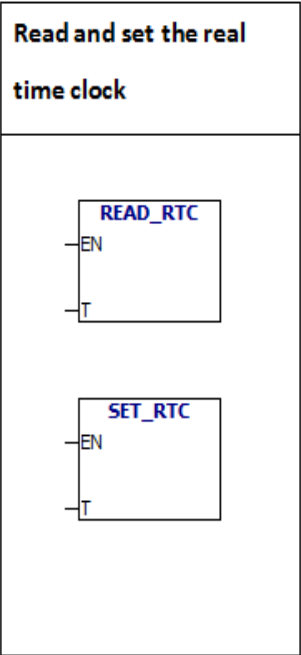
NOP instruction is invalid for user program execution. The value of N is between 0 and 255.

6.2 Clock instruction



6.2.1 Read and set the real time clock

| Input / output | Operand | Data type |
|----------------|--|-----------|
| T | VB, IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD | Byte |



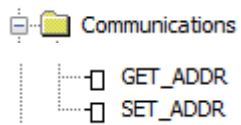
TODR instruction reads the current time and date from the hardware clock and load it into the time buffer of 7bytes starting at the address T.

The TODW instruction writes the current time and date to the hardware clock that is specified by the 7 bytes time buffer at the beginning of the T.

All date and time values must be encoded in USINT format. Please refer to the following table.

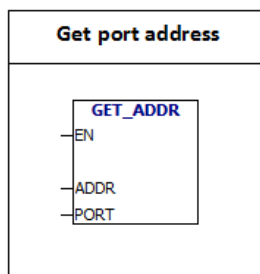
| T byte | direction | byte data type |
|--------|-----------|----------------|
| 0 | second | USINT |
| 1 | minute | USINT |
| 2 | hour | USINT |
| 3 | date | USINT |
| 4 | week | USINT |
| 5 | month | USINT |
| 6 | year | USINT |

6.3 Communication



6.3.1 Get port address

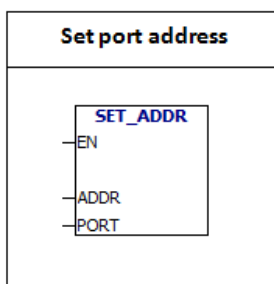
| Input / output | Operand | Data type |
|----------------|--|-----------|
| ADDR | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC | byte |
| PORT | Constant (0 or 1) | byte |



The GET -ADDR instruction reads the PLC port site from the PORT, and put the value in the address specified in the ADDR.

6.3.2 Set port address

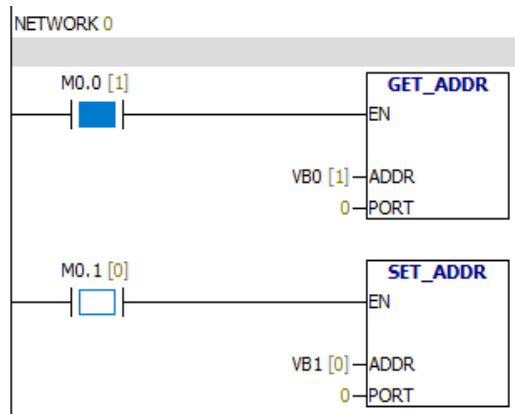
| Input / output | Operand | Data type |
|----------------|--|-----------|
| ADDR | VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC | byte |
| PORT | Constant (0 or 1) | byte |



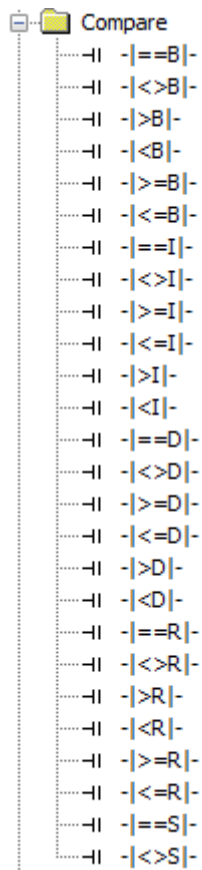
SET- ADDR instruction set the PORT site to the value specified in the ADDR.

The new address is not permanently saved.

Example:



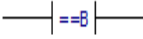

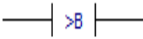
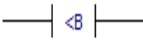
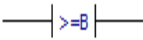
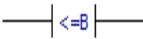
6.4 Compare



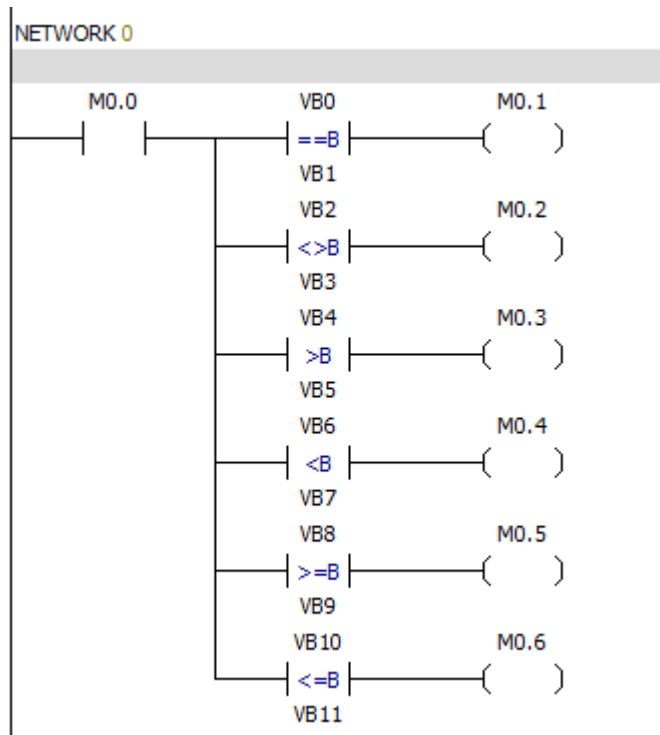
6.4.1 Byte compare

Input / output Operand Data type

Input IB, QB, MB, SMB, VB, SB, LB, AC, constant, *VD, *LD, *AC byte

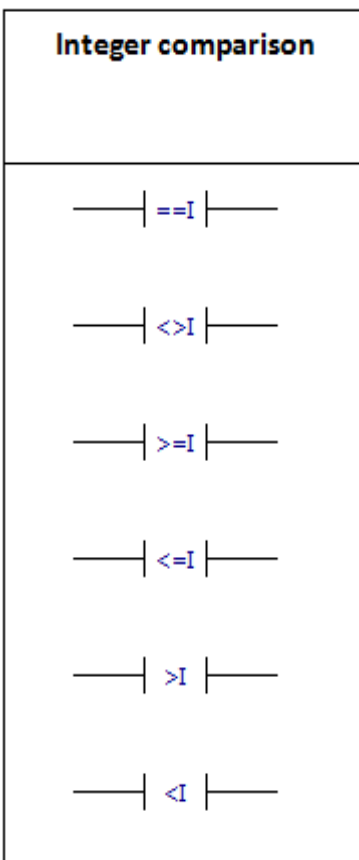
| Byte compare | Byte comparison instructions are used for comparing two values: IN1 and IN2. |
|---|--|
|  | Comparison includes: IN2、IN1 >= IN2、IN1 <= IN2、IN1 > IN2、IN1 < IN2 or IN1 <> IN2. Byte comparison without symbol. |
|  | In LAD, the contact is open when the result is 1. |
|  | Attention: |
|  | The following conditions are serious errors. These errors will cause the PLC to immediately stop the execution of the program: |
|  | 1.Enter illegal indirect address. |
|  | 2.Enter the illegal real number |

Example:



6.4.2 Integer comparison

| Input / output | Operand | Data type |
|----------------|---|-----------|
| Input | IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, constant, *VD, *LD, *AC Integer | |



Comparison instructions are used for comparing two values: IN1 and IN2.

Comparison includes: IN1 = IN2、IN1 >= IN2、IN1 <= IN2、IN1 > IN2、IN1 < IN2 or IN1 <> IN2.

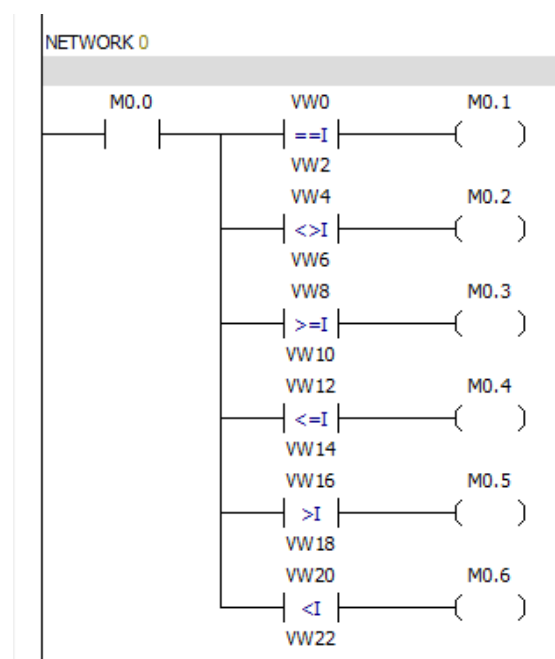
Integer comparison with symbol (16#7FFF > 16#8000).

In LAD, when the comparison result is true, the contact will be open.

Attention: The following conditions are serious errors. These errors will cause the PLC to immediately stop the execution of the program:

- 1.Enter illegal indirect address.
- 2.Enter the illegal real number.

Example:

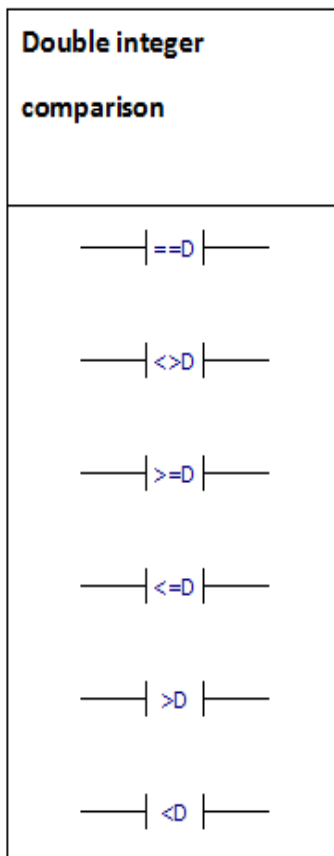


6.4.3 Double integer comparison

Input / output Operand

Data type

Input ID, QD, MD, SD, SMD, VD, LD, HC, AC, constant, *VD, *LD, *AC Double integer



Comparison double integer instructions are used for comparing two values: IN1 and IN2.

Comparison includes: IN1 = IN2、IN1 >= IN2、IN1 <= IN2、IN1 > IN2、IN1 < IN2 or IN1 <> IN2.

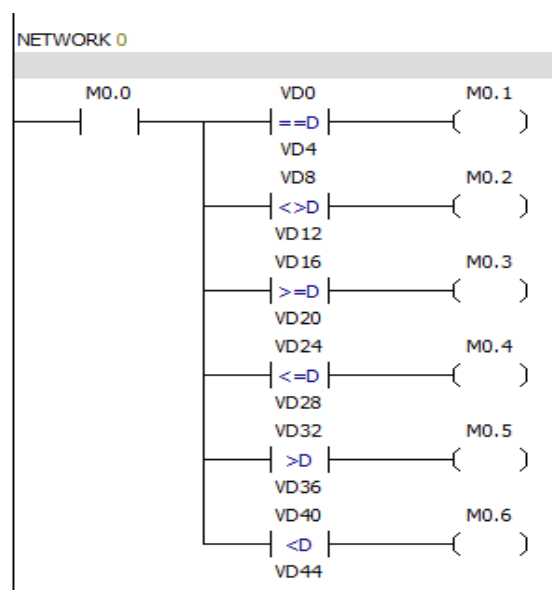
Double integer comparison with symbol (16#7FFFFFFF > 16#80000000).

In LAD, when the comparison result is true, the contact will be open.

Attention: The following conditions are serious errors. These errors will cause the PLC to immediately stop the execution of the program:

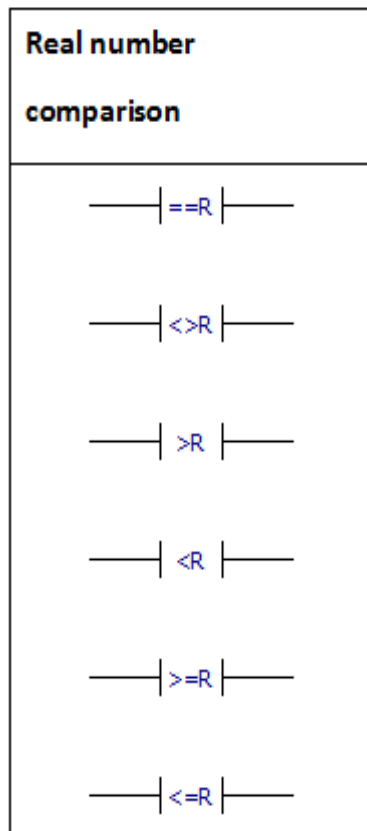
1. Enter illegal indirect address.
2. Enter the illegal real number.

Example:



6.4.4 Real number comparison

| Input / output | Operand | Data type |
|----------------|--|-------------|
| Input | ID, QD, MD, SD, SMD, VD, LD, AC, constant, *VD, *LD, *AC | Real number |



Comparison real number instructions are used for comparing two values: IN1 and IN2.

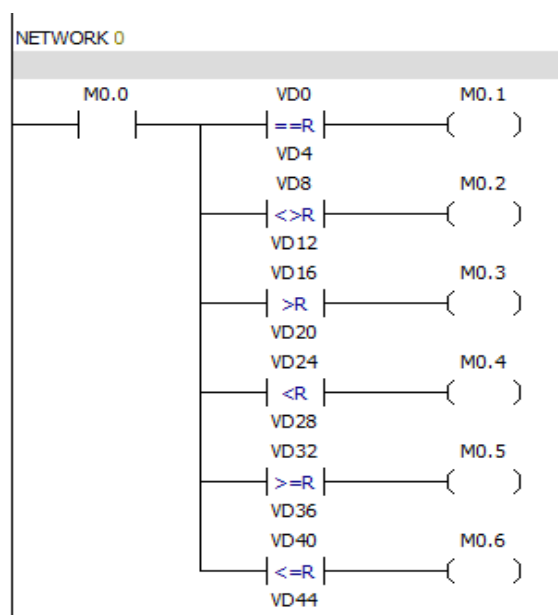
Comparison includes: IN1 = IN2、IN1 >= IN2、IN1 <= IN2、IN1 > IN2、IN1 < IN2 or IN1 <> IN2. Real number comparison with symbol.

In LAD, when the comparison result is true, the contact will be open.

Attention: The following conditions are serious errors. These errors will cause the PLC to immediately stop the execution of the program:

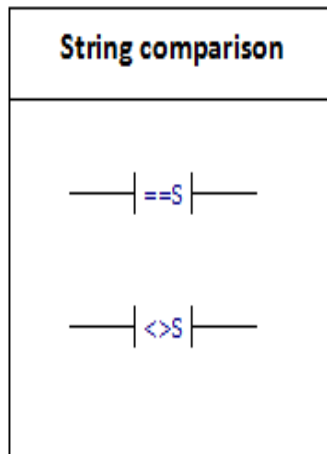
1. Enter illegal indirect address.
2. Enter the illegal real number.

Example:



6.4.5 String comparison

| Input / output | Operand | Data type |
|----------------|--|-----------|
| IN1 | VB, Constant string, LB, *VD, *LD, *AC | String |
| IN2 | VB, LB, *VD, *LD, *AC | String |



Comparison string instructions are used for comparing two ASCII strings: IN1=IN2, IN1<>IN2

In LAD, when the comparison result is true, the comparison contact will be turned on.

The maximum length of a single constant string is 126 bytes. The maximum combined length of the two constant string is 242 bytes.

Attention: The following conditions are serious errors.

These errors will cause the PLC to immediately stop the execution of the program:

1. Enter illegal indirect address.
2. Enter a string of more than 254 characters in length.
3. The start address and length of the string cannot be put into a specified memory area.

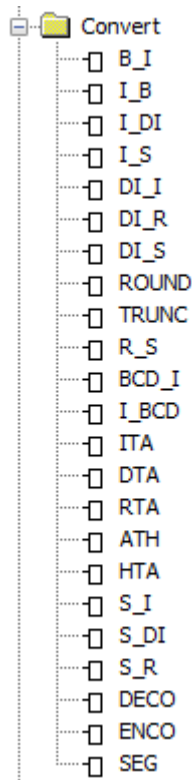
ASCII constant string data type format:

String is a series of characters and the corresponding memory address, each character is stored in a byte. The value of the first byte of a string is the length of the string. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

The following memory map shows the format of the string data type. The length of a string can be between 0 and 254 characters. The maximum length of the string is 255 bytes.

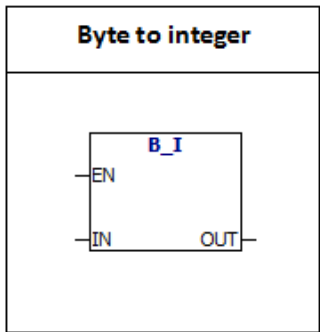
| | | | | | | | |
|---------------|-------------|-------------|-------------|-------------|-------------|-------|---------------|
| String length | Character 1 | Character 2 | Character 3 | Character 4 | Character 5 | | Character 254 |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | | Byte 254 |

6.5 Convert



6.5.1 Byte to integer

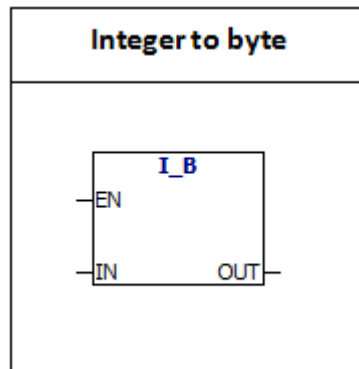
| Input / output Operand | | Data type |
|------------------------|---|-----------|
| IN | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *AC, *VD, *LD | Byte |
| OUT | VW, IW, QW, MW, SW, SMW, LW, AQW, T, C, AC, *VD, *LD, *AC | Integer |



Byte to integer: The B-I instruction converts the byte value to the integer value, and the result is inserted into the variable specified by the OUT. Because the byte does not have a symbol, the result does not have extension of the symbol.

6.5.2 Integer to byte

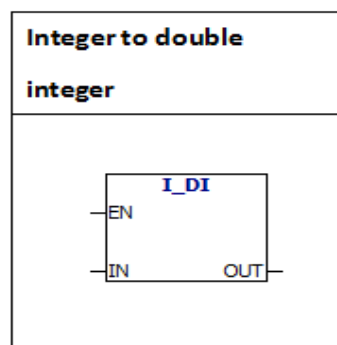
| Input / output | Operand | Data type |
|----------------|---|-----------|
| IN | VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, constant, *VD, *LD, *AC | Integer |
| OUT | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD | Byte |



Integer to byte: I-B Instruction converts the value of a word to a byte value, and the result is inserted into the variable specified by the OUT. The numerical range is 0 to 255. Other values will result in overflow and the output will not be affected.

6.5.3 Integer to double integer

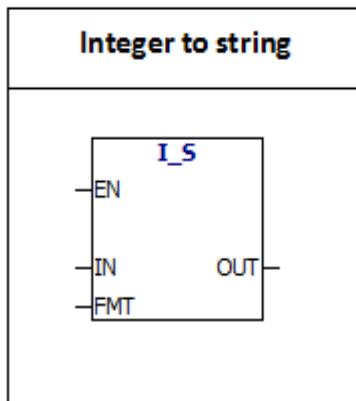
| Input / output | Operand | Data type |
|----------------|---|----------------|
| IN | VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, constant, *VD, *LD, *AC | Integer |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Double-integer |



Integer to double integer: I- DI instruction converts the value of integer to a double integer value, and the result is inserted into the variable specified by the OUT. Sign is extended.

6.5.4 Integer to string

| Input / output | Operand | Data type |
|----------------|---|-----------|
| IN | VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, constant, AC, *VD, *LD, *AC | Integer |
| FMT | VB, IB, QB, MB, SB, SMB, LB, constant, AC, *VD, *LD, *AC | Byte |
| OUT | VB, *VD, LB, *AC, *LD | String |



I-s instruction: the instruction converts the integer word to a ASCII string of 8 characters in length. Format (FMT) specifies the number of digits to the right of the decimal point. The result string is written in 9 consecutive bytes from the OUT.

Illegal format (nnn > 5)

ASCII constant string data type format:

String is a series of characters, each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

The following memory map shows the format of the string data type. The length of a string can be between 0 and 254 characters. The maximum length of the string is 255 bytes.

| | | | | | | | |
|---------------|-------------|-------------|-------------|-------------|-------------|-------|---------------|
| String length | Character 1 | Character 2 | Character 3 | Character 4 | Character 5 | | Character 254 |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | | Byte 254 |

The following is the definition of Operation number of ITS format:

| | | | | | | | |
|-----|---|---|---|-----|---|---|---|
| MSB | | | | LSB | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | c | n | n | n |

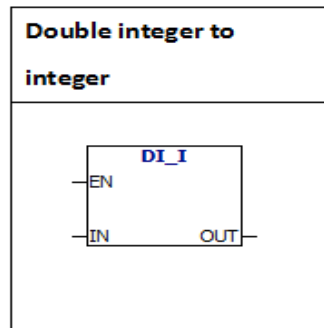
C = comma (1) or decimal point (0)

nnn = The number of digits on the right side of the decimal point

The length of the output string is always 8 characters. nnn valid values are from 0 to 5. If nnn=0, the value will be displayed without a decimal point. When the value of NNN is greater than 5, the output is displayed as a string of 8 ASCII space characters. C decides to use a comma or a decimal point between integer and decimal. The 4 bits above the top of the format must be zero.

6.5.5 Double integer to integer

| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, HC, AC, constant, *VD, *LD, *AC | Double integer |
| OUT | VW, IW, QW, MW, SW, SMW, LW, AQW, T, C, AC, *VD, *LD, *AC | Integer |

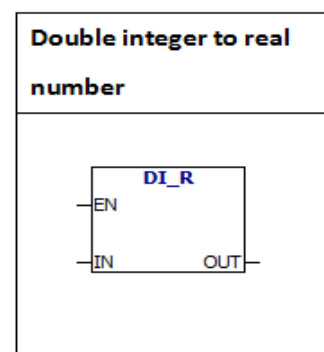


Double integer to integer: DI-I instruction converts the value of double integer to an integer value, and the result is inserted into the variable specified by the OUT.

Large value will result in overflow and the output will not be affected.

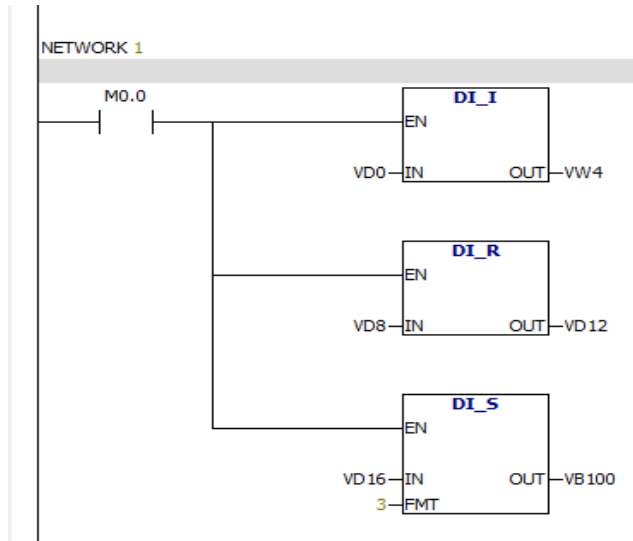
6.5.6 Double integer to real number

| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, HC, AC, constant, *VD, *AC, *LD | Double integer |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Real number |



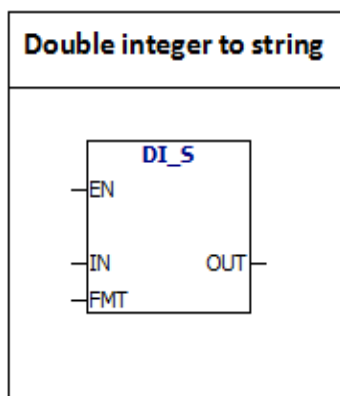
Double integer to real number: Instruction converts 32-bit signed integer to 32 bit real number, and the result is inserted into the variable specified by the OUT.

Example:



6.5.7 Double integer to string

| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, HC, constant, AC, *VD, *AC, *LD | Double integer |
| FMT | VB, IB, QB, MB, SB, SMB, LB, constant, AC, *VD, *LD, *AC | Byte |
| OUT | VB, *VD, LB, *AC, *LD | String |



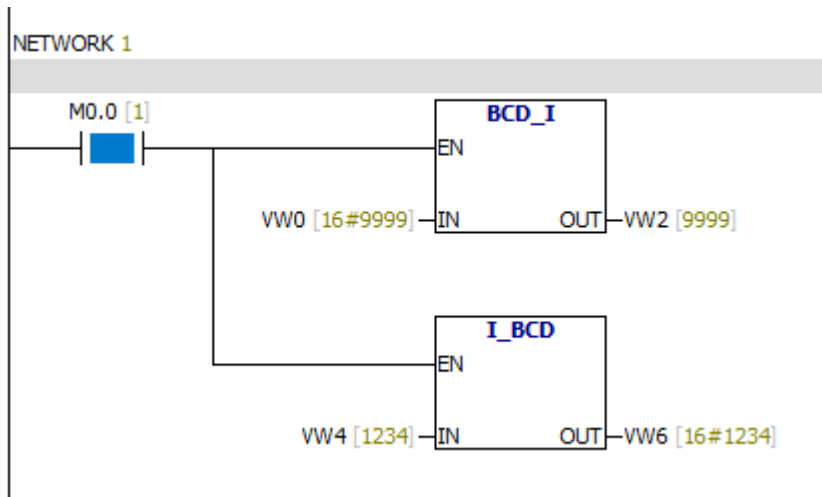
Double integer to string: DI-s instruction: the instruction converts the double integer to a ASCII string of 12 characters in length. Format (FMT) specifies the number of digits to the right of the decimal point. The output string is written in 13 consecutive bytes from the OUT.

Illegal format (nnn > 5)

ASCII constant string data type format:

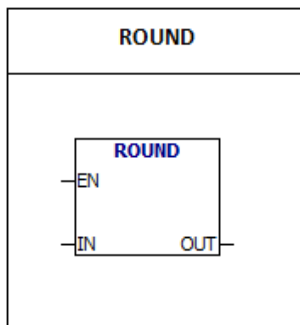
String is a series of characters, each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

The following memory map shows the format of the string data type. The length of a string can be between 0 and 254 characters. The maximum length of the string is 255 bytes.



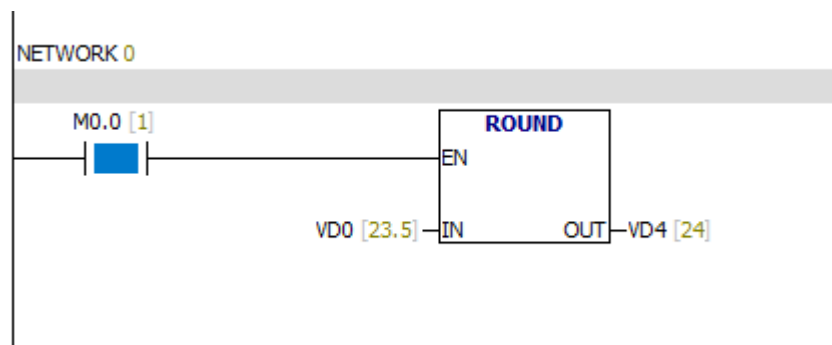
6.5.9 ROUND

| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Double integer |



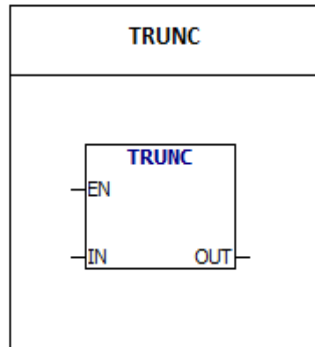
The ROUND instruction converts the real number value to a double integer value and the result is inserted into the variable specified by the OUT. If the fractional part is equal to or greater than 0.5, the integer part will be added to 1.

Example:



6.5.10 TRUNC

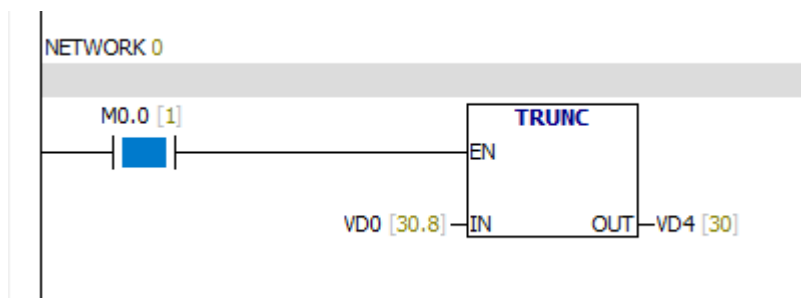
| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD | Double integer |



TRUNC: Instruction converts 32 bits of real number to 32 bits integer, and the result is inserted into the variable specified by the OUT.

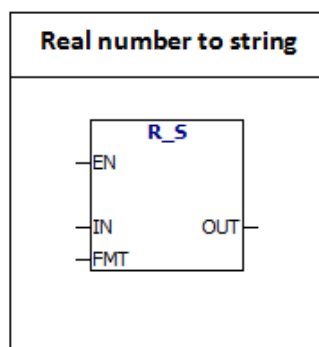
Only the integer part of the real number is converted, and the fractional part is discarded.

Example:



6.5.11 Real number to string

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, constant, AC, *VD, *LD, *AC | Real number |
| FMT | VB, IB, QB, MB, SB, SMB, LB, constant, AC, *VD, *LD, *AC | Byte |
| OUT | VB, LB, *VD, *AC, *LD | String |



R-S: Instruction converts the real number value to a ASCII string. (FMT) format specifies the conversion accuracy of the right of the decimal point.

The conversion result is placed in a string starting with OUT.

The output string length specified in the format can be 3 to 15 characters. The format of real numbers used in PLC is at

most 7 digits.

Illegal format:

$$nnn > 5$$
$$SSSS < 3$$

ssss < Required number of characters

ASCII constant string data type format:

String is a series of characters, each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

The following memory map shows the format of the string data type. The length of a string can be between 0 and 254 characters. The maximum length of the string is 255 bytes.

| | | | | | | | |
|---------------|-------------|-------------|-------------|-------------|-------------|-------|---------------|
| String length | Character 1 | Character 2 | Character 3 | Character 4 | Character 5 | | Character 254 |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | | Byte 254 |

The following is the RTS instruction format (FMT) operand definition:

MSB LSB

7 6 5 4 3 2 1 0

S S S S C n n n

ssss = The length of the output string

c = Comma (1) or decimal point (0)

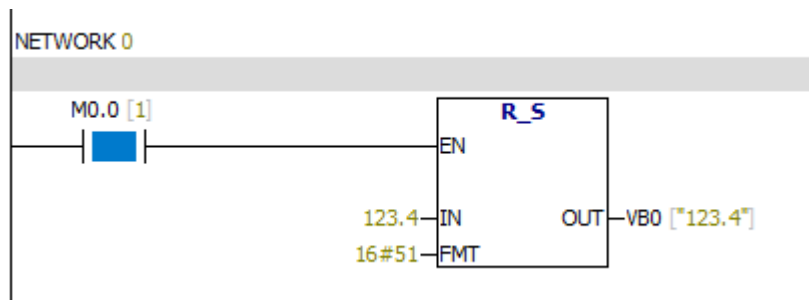
nnn = The number of characters of the right of the decimal point.

The length of the output string is specified by the SSSS field. 0, 1, or 2 bytes are not valid. The effective range of the NNN is from 0 to 5. NNN is equal to 0, the output shows no decimal point. When the NNN value is greater than 5 or when the specified output string length is too small to store the conversion value, the output string is filled with ASCII space characters. The C bit specifies using a comma (C = 1) or a decimal point (C = 0).

Prompt: output string according to the following rules

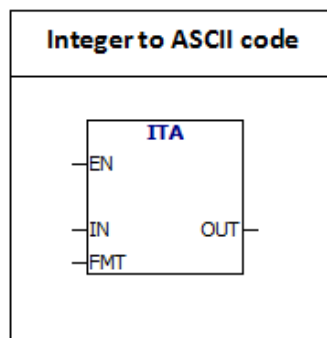
- 1.Positive number is written to output buffer without a sign.
- 2.Negative number is written to the output buffer with “-” .
- 3.The starting zero on the left of the decimal point is compressed.
- 4.The size of the output string must be 3 bytes larger than “nnn”
- 5.The value in the output string must be aligned to the right.

Example:



6.5.12 Integer to ASCII code

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN | VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, constant, *VD, *LD, *AC | Integer |
| FMT | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | byte |
| OUT | VB, IB, QB, MB, SB, SMB, LB, *VD, *LD, *AC | byte |



ITA: The instruction converts the integer word to ASCII characters. (FMT) format specifies the conversion accuracy of the right of the decimal point.

The conversion result is placed in the 8 successive bytes from the OUT.ASCII character number is always 8 characters.

Error condition:

nnn > 5

The following is the ITA instruction format (FMT) operand definition:



The size of the output buffer is always 8 bytes. nnn = The number of characters of the

right of the decimal point. The effective range of the NNN is from 0 to 5. NNN is equal to 0, the output shows no decimal point. When the NNN value is greater than 5, the output string is filled with ASCII space characters. The C bit specifies using a comma (C = 1) or a decimal point (C = 0). High 4 bits must be 0.

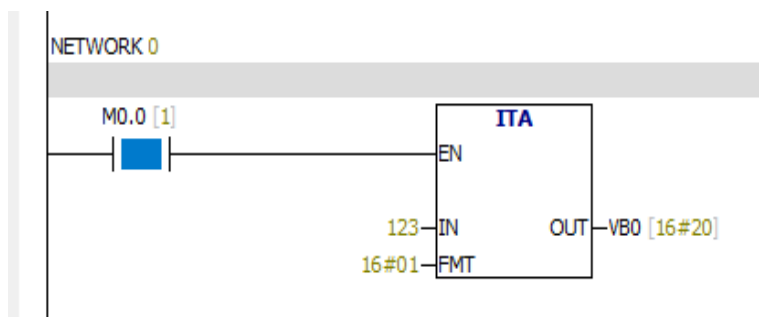
Prompt: The output according to the following rules

1. Positive number is written to output buffer without a sign.
2. Negative number is written to the output buffer with “-” .
3. The starting zero on the left of the decimal point is compressed.
4. The value in the output string must be aligned to the right.

Example:

| | OUT | OUT | OUT | OUT | OUT | OUT | OUT | OUT |
|-------------|-----|-----|-----|-----|-----|-----|-----|-----|
| | +1 | +2 | +3 | +4 | +5 | +6 | +7 | |
| in = 12 | | | 0 | . | 0 | 1 | 2 | |
| in = -123 | | - | 0 | . | 1 | 2 | 3 | |
| in = 1234 | | | 1 | . | 2 | 3 | 4 | |
| in = -12345 | - | 1 | 2 | . | 3 | 4 | 5 | |

Example:



| | Address | Data Type | Value |
|--|---------|-----------|-------|
| | M0.0 | BOOL | 1 |
| | VB0 | BYTE | 16#20 |
| | VB1 | BYTE | 16#20 |
| | VB2 | BYTE | 16#20 |
| | VB3 | BYTE | 16#20 |
| | VB4 | BYTE | 16#31 |
| | VB5 | BYTE | 16#32 |
| | VB6 | BYTE | 16#2E |
| | VB7 | BYTE | 16#33 |

As shown in Figure: The integer input is 123; nnn=1

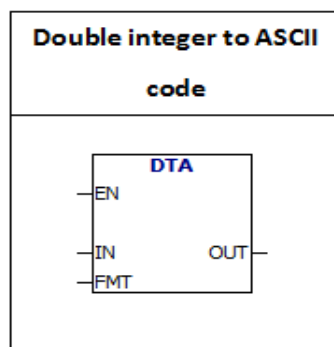
The output value is as follows:

VB7 16#33 3

| | | |
|-----|-------|-------|
| VB6 | 16#2E | . |
| VB5 | 16#32 | 2 |
| VB4 | 16#31 | 1 |
| VB3 | 16#20 | Space |
| VB2 | 16#20 | Space |
| VB1 | 16#20 | Space |
| VB0 | 16#20 | Space |

6.5.13 Double integer to ASCII code

| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, HC, constant, AC, *VD, *AC, *LD | Double integer |
| FMT | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | Byte |
| OUT | VB, IB, QB, MB, SB, SMB, LB, *VD, *LD, *AC | Byte |



DTA: The instruction converts the double integer to ASCII characters. (FMT) format specifies the conversion accuracy of the right of the decimal point.

The conversion result is placed in the 12 successive bytes from the OUT.

Error conditions:

FMT high four bits value is greater than 0

nnn > 5

The following is the DTA instruction format (FMT) operand definition:



The size of the output buffer is always 12 bytes. nnn = The number of characters of the right of the decimal point. The effective range of the NNN is from 0 to 5. NNN is equal to 0, the output shows no decimal point. When the NNN value is greater than 5, the output string is filled with ASCII space characters. The C bit specifies using a

comma (C = 1) or a decimal point (C = 0). High 4 bits must be 0.

Prompt: The output according to the following rules

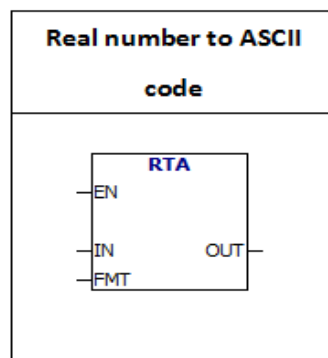
- 1.Positive number is written to output buffer without a sign.
- 2.Negative number is written to the output buffer with “-” .
- 3.The starting zero on the left of the decimal point is compressed.
- 4.The value in the output string must be aligned to the right.

Example:

| OUT character | 0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 | +8 | +9 | +10 | +11 |
|---------------|---|----|----|----|----|----|----|----|----|----|-----|-----|
| in = -12 | | | | | | - | 0 | . | 0 | 0 | 1 | 2 |
| in = 1234567 | | | | | 1 | 2 | 3 | . | 4 | 5 | 6 | 7 |

6.5.14 Real number to ASCII code

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, AC, constant, *VD, *LD, *AC | Real number |
| FMT | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | Byte |
| OUT | VB, IB, QB, MB, SB, SMB, LB, *VD, *LD, *AC | Byte |



RTA: The instruction converts the real number to ASCII characters. (FMT) format specifies the conversion accuracy of the right of the decimal point.

The conversion result is placed in the output buffer from the OUT. The length of the output buffer is 3 to 15 characters.

Error conditions:

nnn > 5

ssss < 3

ssss < Number of characters in OUT

The following is the RTA instruction format (FMT) operand definition:



The length of the output string is specified by the SSSS field. 0, 1, or 2 bytes are not

valid. The effective range of the NNN is from 0 to 5. NNN is equal to 0, the output shows no decimal point. When the NNN value is greater than 5 or when the specified output string length is too small to store the conversion value, the output string is filled with ASCII space characters. The C bit specifies using a comma (C = 1) or a decimal point (C = 0).

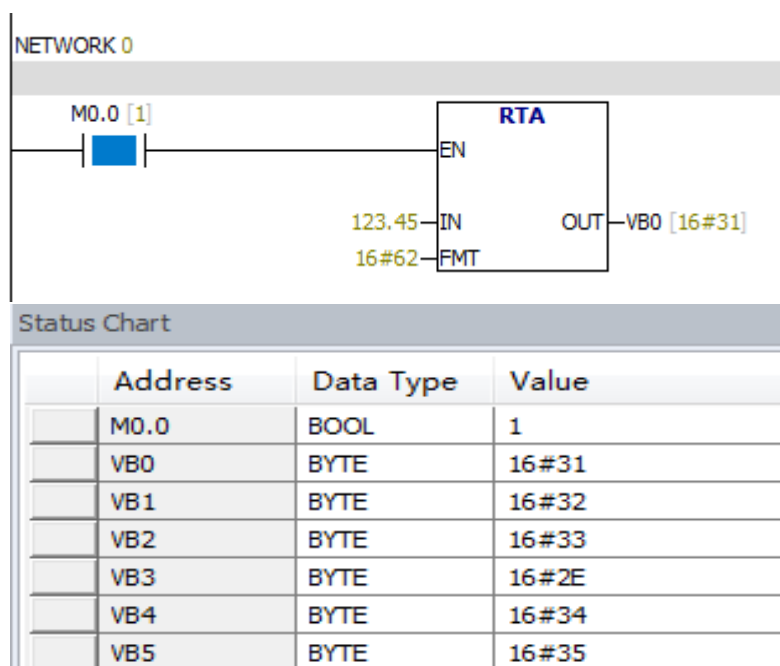
The output according to the following rules:

1. Positive number is written to output buffer without a sign.
2. Negative number is written to the output buffer with “-”.
3. The starting zero on the left of the decimal point is compressed.
4. The number of characters of the right of the decimal point is equal to the value of “nnn”.
5. The size of the output string must be 3 bytes larger than “nnn”.
6. The value in the output string must be aligned to the right.

Example:

| | OUT | OUT | OUT | OUT | OUT | OUT |
|---------------|-----|-----|-----|-----|-----|-----|
| | | +1 | +2 | +3 | +4 | +5 |
| in = 1234.5 | 1 | 2 | 3 | 4 | . | 5 |
| in = -0.0004 | | | | 0 | . | 0 |
| in = -3.67526 | | | - | 3 | . | 7 |
| in = 1.95 | | | | 2 | . | 0 |

Example:



Convert the real number 123.45 into ASCII code. The output is 6 bytes.

The output:

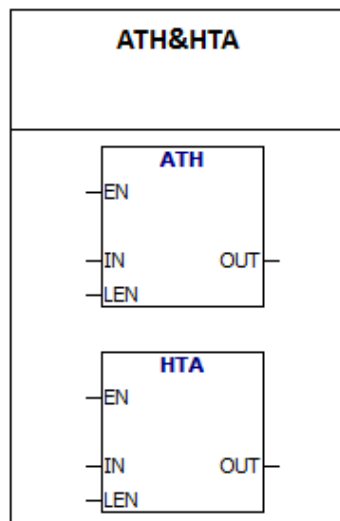
| VB0 | VB1 | VB2 | VB3 | VB4 | VB5 |
|-------|-------|-------|-------|-------|-------|
| 16#31 | 16#32 | 16#33 | 16#2E | 16#34 | 16#35 |
| 1 | 2 | 3 | . | 4 | 5 |

6.5.15 ATH&HTA

Input/output Operand

Data type

| | | |
|--------|---|------|
| IN,OUT | VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD | Byte |
| LEN | VB, IB, QB, MB, SB, SMB, LB, AC,constant, *VD, *LD, *AC | Byte |



ASCII to HEX Instruction converts the ASCII characters starting with “IN” to the hexadecimal digits starting with the “out”. The maximum length of the ASCII string is 255 characters.

HEX to ASCII Instruction converts the hexadecimal digits starting with “IN” to the ASCII characters starting with the “out”.

The length of conversion hexadecimal digits is specified by the LEN. The maximum length is 255.

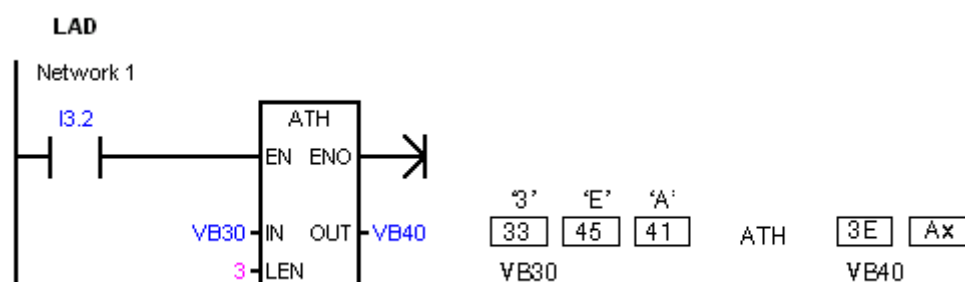
Valid ASCII input character:

Numbers 0 to 9 and capital letters A to F.

ASCII Codes: 30 to 39 and 41 to 46.

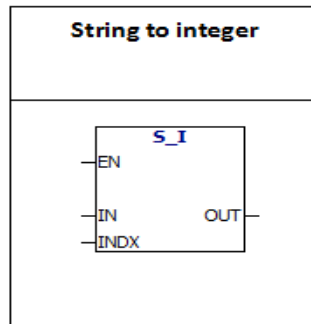
Error condition: Illegal ASCII code

Example:



6.5.16 String to integer

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN | VB, constant string, LB, *VD, *LD, *AC | String |
| INDX | VB, IB, QB, MB, SB, SMB, LB, constant, AC, *VD, *LD, *AC | Byte |
| OUT | VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, AC, *VD, *LD, *AC | Integer |



S-I: The instruction converts the string value “IN” to the integer value stored in the OUT, starting with the offset INDX location.

ASCII constant string data type format:

String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

The following memory map shows the format of the string data type. The length of a string can be between 0 and 254 characters. The maximum length of the string is 255 bytes.

| | | | | | | | |
|---------------|-------------|-------------|-------------|-------------|-------------|-------|---------------|
| String length | Character 1 | Character 2 | Character 3 | Character 4 | Character 5 | | Character 254 |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | | Byte 254 |

INDX value is typically set to 1, starting conversion from the first character of the string. INDX value can be set to other values. This method can be used when the input string contains characters that are not required to be converted. For example, if the input string is "Temperature: 77.8", you can set INDX value 13 to skip the characters "Temperature:".

When the end of the string is reached or when the first invalid character is found, the conversion is terminated. Invalid character is any character other than number (0-9).

The following table shows examples of valid and invalid integer input strings:

Valid Input Strings for String to Integer/Double Integer

| Input String | Output Integer |
|-----------------|----------------|
| "123" | 123 |
| "-00456" | -456 |
| "123.45" | 123 |
| " +2345" | 2345 |
| "000000123ABCD" | 123 |

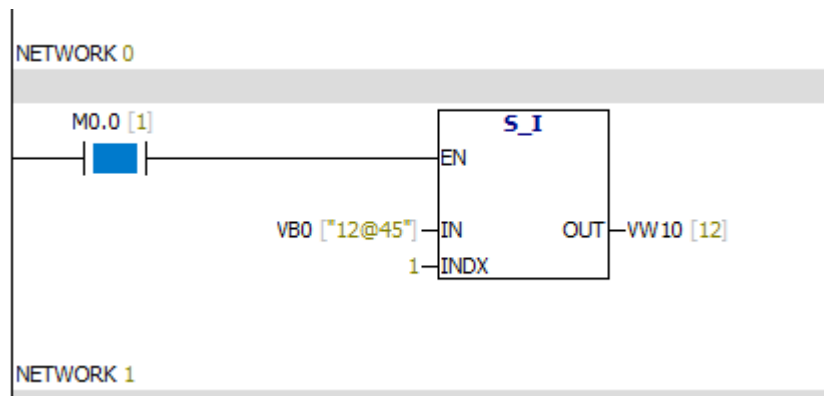
Valid Input Strings for String to Real

| Input String | Output Real |
|--------------|-------------|
| "123" | 123.0 |
| "-00456" | -456.0 |
| "123.45" | 123.45 |
| " +2345" | 2345.0 |
| "000000123" | .000000123 |

Invalid Input Strings

| Input String |
|--------------|
| "A123" |
| " " |
| "++123" |
| "-123" |
| " +123" |

Example:



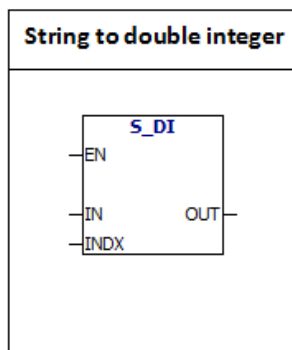
Status Chart

| | Address | Data Type | Value |
|--|---------|-----------|-------|
| | M0.0 | BOOL | 1 |
| | VB0 | BYTE | 16#05 |
| | VB1 | BYTE | 16#31 |
| | VB2 | BYTE | 16#32 |
| | VB3 | BYTE | 16#40 |
| | VB4 | BYTE | 16#34 |
| | VB5 | BYTE | 16#35 |

Enter the string "12@45". The S-I instruction converts the string from the first character, and the result is an integer 12.

6.5.17 String to double integer

| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN | VB, constant string, LB, *VD, *LD, *AC | String |
| INDX | VB, IB, QB, MB, SB, SMB, LB, constant, AC, *VD, *LD, *AC | Byte |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Double integer |



S- DI: The instruction converts the string value “IN” to the double integer value stored in the “OUT” , starting with the offset INDX location.

ASCII constant string data type format:

String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

The following memory map shows the format of the string data type. The length of a string can be between 0 and 254 characters. The maximum length of the string is 255 bytes.

| | | | | | | | |
|---------------|-------------|-------------|-------------|-------------|-------------|-------|---------------|
| String length | Character 1 | Character 2 | Character 3 | Character 4 | Character 5 | | Character 254 |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | | Byte 254 |

INDX value is typically set to 1, starting conversion from the first character of the string. INDX value can be set to other values. This method can be used when the input string contains characters that are not required to be converted. For example, if the input string is "Temperature: 77.8", you can set INDX value 13 to skip the characters "Temperature:".

When the end of the string is reached or when the first invalid character is found, the conversion is terminated. Invalid character is any character other than number (0-9).

The following table shows examples of valid and invalid integer input strings:

Valid Input Strings for String to Integer/Double Integer

| Input String | Output Integer |
|-----------------|----------------|
| "123" | 123 |
| "-00456" | -456 |
| "123.45" | 123 |
| " +2345" | 2345 |
| "000000123ABCD" | 123 |

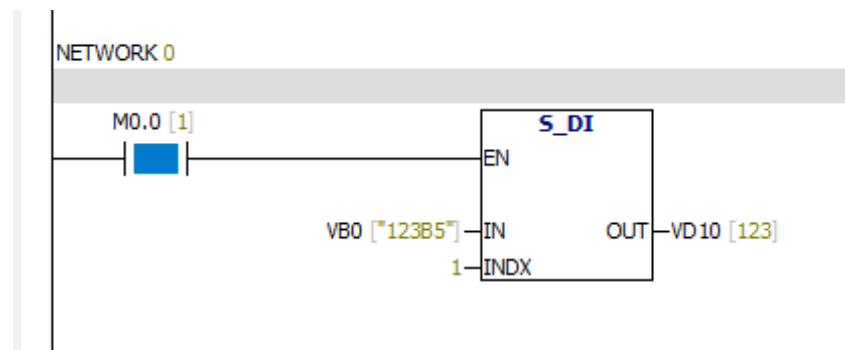
Valid Input Strings for String to Real

| Input String | Output Real |
|--------------|-------------|
| "123" | 123.0 |
| "-00456" | -456.0 |
| "123.45" | 123.45 |
| " +2345" | 2345.0 |
| "000000123" | .000000123 |

Invalid Input Strings

| Input String |
|--------------|
| "A123" |
| " " |
| "++123" |
| " +123" |
| " +123" |

Example:



Status Chart

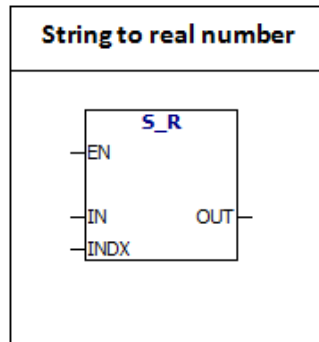
| | Address | Data Type | Value |
|--|---------|-----------|-------|
| | M0.0 | BOOL | 1 |
| | VB0 | BYTE | 16#05 |
| | VB1 | BYTE | 16#31 |
| | VB2 | BYTE | 16#32 |
| | VB3 | BYTE | 16#33 |
| | VB4 | BYTE | 16#42 |
| | VB5 | BYTE | 16#35 |

Enter the string "123B5". The S- DI instruction converts the string from the first character, and the result is a double integer 123.

Because B is an invalid character, the characters after B are no longer converted.

6.5.18 String to real number

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VB, constant string, LB, *VD, *LD, *AC | String |
| INDX | VB, IB, QB, MB, SB, SMB, LB, Constant, AC, *VD, *LD, *AC | Byte |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Real number |



S-R: The instruction converts the string value “IN” to the real number value stored in the “OUT”, starting with the offset INDX location.

ASCII constant string data type format:

String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

The following memory map shows the format of the string data type. The length of a string can be between 0 and 254 characters. The maximum length of the string is 255 bytes.

| | | | | | | | |
|---------------|-------------|-------------|-------------|-------------|-------------|-------|---------------|
| String length | Character 1 | Character 2 | Character 3 | Character 4 | Character 5 | | Character 254 |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | | Byte 254 |

INDX value is typically set to 1, starting conversion from the first character of the string. INDX value can be set to other values. This method can be used when the input string contains characters that are not required to be converted. For example, if the input string is “Temperature: 77.8”, you can set INDX value 13 to skip the characters "Temperature:".

When the end of the string is reached or when the first invalid character is found, the conversion is terminated. Invalid character is any character other than number (0-9).

This instruction does not generate overflow errors, but only converts the string to

real number and then terminates the conversion.

For example, the string "1.234E6" will be converted to a real number value "1.234" without generating an error message.

The following table shows examples of valid and invalid integer input strings:

Valid Input Strings for String to Integer/Double Integer

| Input String | Output Integer |
|-----------------|----------------|
| "123" | 123 |
| "-00456" | -456 |
| "123.45" | 123 |
| " +2345" | 2345 |
| "000000123ABCD" | 123 |

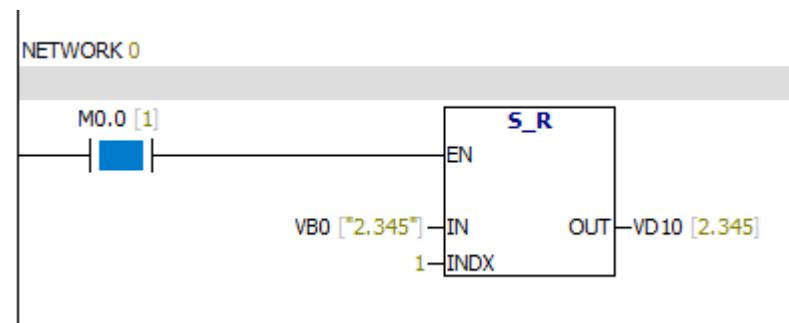
Valid Input Strings for String to Real

| Input String | Output Real |
|--------------|-------------|
| "123" | 123.0 |
| "-00456" | -456.0 |
| "123.45" | 123.45 |
| " +2345" | 2345.0 |
| "000000123" | .000000123 |

Invalid Input Strings

| Input String |
|--------------|
| "A123" |
| " " |
| "++123" |
| " +123" |
| " +123" |

Example:



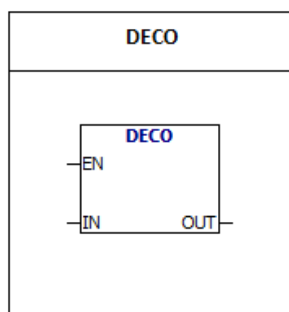
Status Chart

| | Address | Data Type | Value |
|--|---------|-----------|-------|
| | M0.0 | BOOL | 1 |
| | VB0 | BYTE | 16#05 |
| | VB1 | BYTE | 16#32 |
| | VB2 | BYTE | 16#2E |
| | VB3 | BYTE | 16#33 |
| | VB4 | BYTE | 16#34 |
| | VB5 | BYTE | 16#35 |

Input string "2.345" and output the real number 2.345

6.5.19 DECO

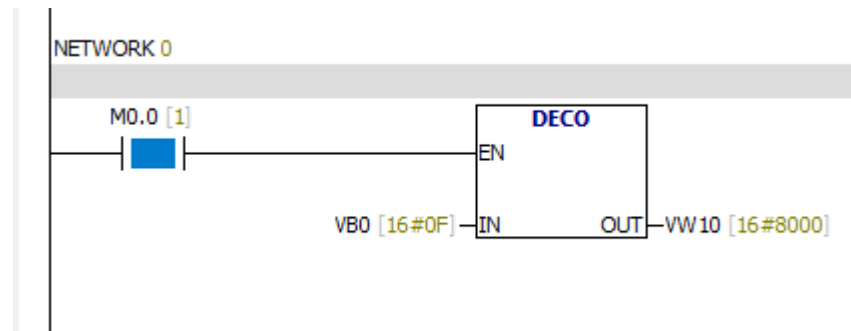
| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN | VB, IB, QB, MB, SMB, LB, SB, AC, constant, *VD, *LD, *AC | Byte |
| OUT | VW, IW, QW, MW, SMW, LW, SW, AQW, T, C, AC, *VD, *AC, *LD | word |



The low four bits value of input byte is n, the nth bit of the output word is equal to 1.

The other bits of the output word are set to 0.

Example:

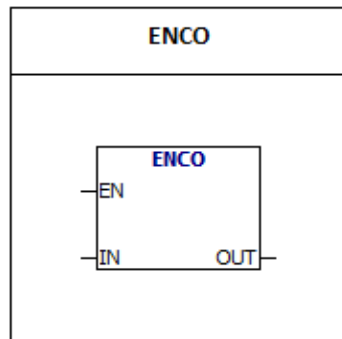


The low four bits value of VB0 is 15, the 15th bit of the VW10 is equal to 1.

The other bits of the VW10 are set to 0.

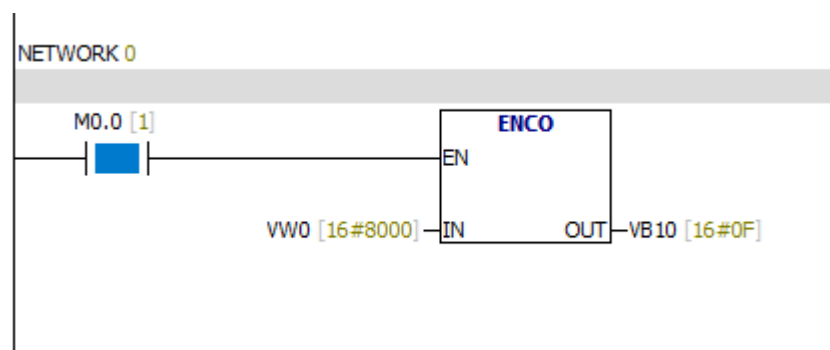
6.5.20 ENCO

| Input/output Operand | Data type |
|--|-----------|
| IN VW, IW, QW, MW, SMW, LW, SW, AIW, T, C, AC, constant, *VD, *AC, *LD | Word |
| OUT VB, IB, QB, MB, SMB, LB, SB, AC, *VD, *LD, *AC | Byte |



ENCO: The nth bit of the input word is equal to 1. The low four bits value of output byte is n.

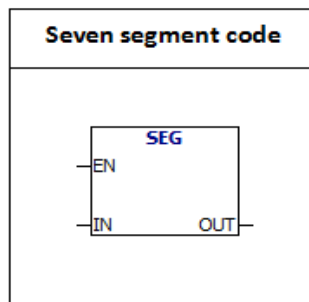
Example:



As shown in the above figure: The 15th bit of the input word vw0 is equal to 1. The low four bits value of output byte vb10 is 15.

6.5.21 Seven segment code

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *AC, *LD | Byte |
| OUT | VB, IB, QB, MB, SMB, LB, AC, *VD, *AC, SB, *LD | Byte |



SEG: The instruction generates the bits of the seven segments.

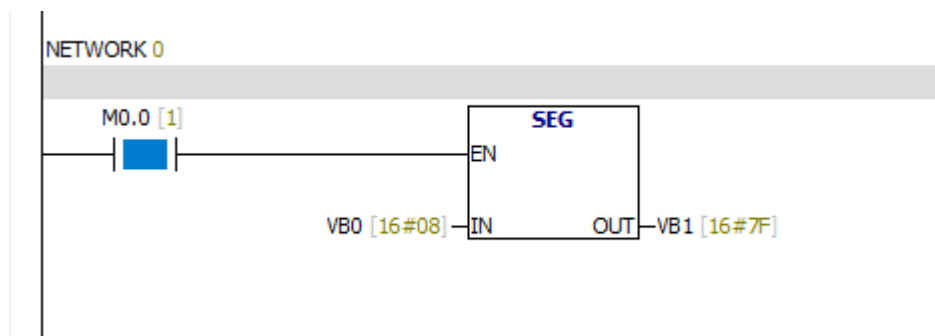
The low four bits value of input byte is converted.

Seven segment code table:

| (IN) LSD | Segment Display | (OUT) - g f e d c b a |
|-------------|--------------------|--------------------------|
| 0 | | 0 0 1 1 1 1 1 1 |
| 1 | | 0 0 0 0 0 1 1 0 |
| 2 | | 0 1 0 1 1 0 1 1 |
| 3 | | 0 1 0 0 1 1 1 1 |
| 4 | | 0 1 1 0 0 1 1 0 |
| 5 | | 0 1 1 0 1 1 0 1 |
| 6 | | 0 1 1 1 1 1 0 1 |
| 7 | | 0 0 0 0 0 1 1 1 |

| (IN) LSD | Segment Display | (OUT) - g f e d c b a |
|-------------|--------------------|--------------------------|
| 8 | | 0 1 1 1 1 1 1 1 |
| 9 | | 0 1 1 0 0 1 1 1 |
| A | | 0 1 1 1 0 1 1 1 |
| B | | 0 1 1 1 1 1 0 0 |
| C | | 0 0 1 1 1 0 0 1 |
| D | | 0 1 0 1 1 1 1 0 |
| E | | 0 1 1 1 1 0 0 1 |
| F | | 0 1 1 1 0 0 0 1 |

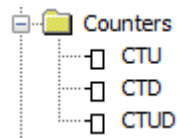
Example:



Analysis:

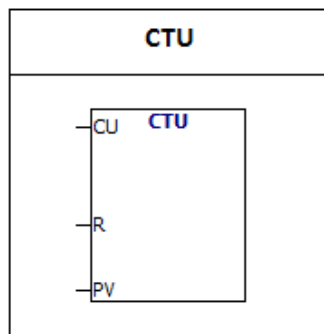
The low four bits value of VB0 is 8. The value of the output byte VB1 is 16#7F. The result of converting VB1 to binary is 2# 0111 1111.

6.6 Counter



6.6.1 CTU

| Input/output | Operand | Data type |
|--------------|--|-----------|
| C xxx | Constant(C0—C255) | Word |
| CU | Enable bit | Boolean |
| R | Enable bit | Boolean |
| PV | VW, IW, QW, MW, SMW, LW, AIW, AC, T, C,constant, *VD, *AC, *LD, SW Integer | Integer |

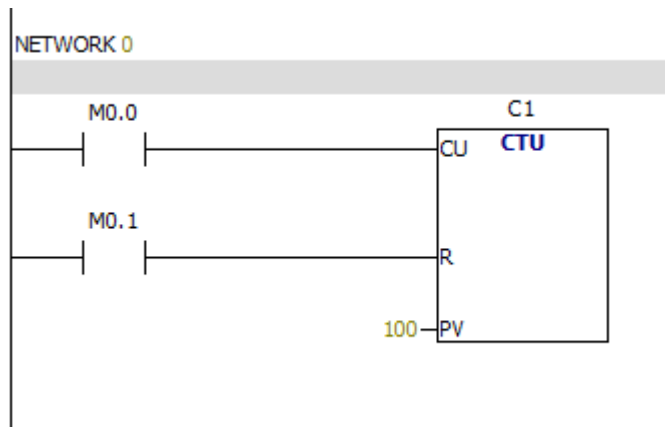


CU bit gets a high level and the current value of the counter plus 1. When the current value is greater than or equal to the present value, the counter bit opens. When R gets a high level, the counter is restored. The maximum value of the counter is 32767.

Counter range: C xxx=C0 ~ C255

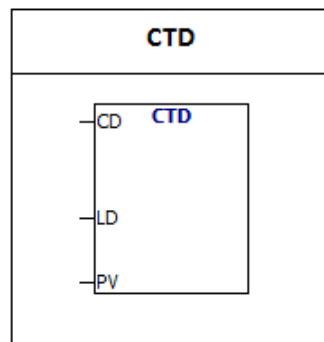
The counter number of each counter is different.

Example:



6.6.2 CTD

| Input/output | Operand | Data type |
|--------------|--|-----------|
| Cxxx | Constant(C0—C255) | Word |
| CD | Enable bit | Boolean |
| LD | Enable bit | Boolean |
| PV | VW, IW, QW, MW, LW, SMW, AC, T, C, AIW,constant, *VD, *AC, *LD, SW | Integer |

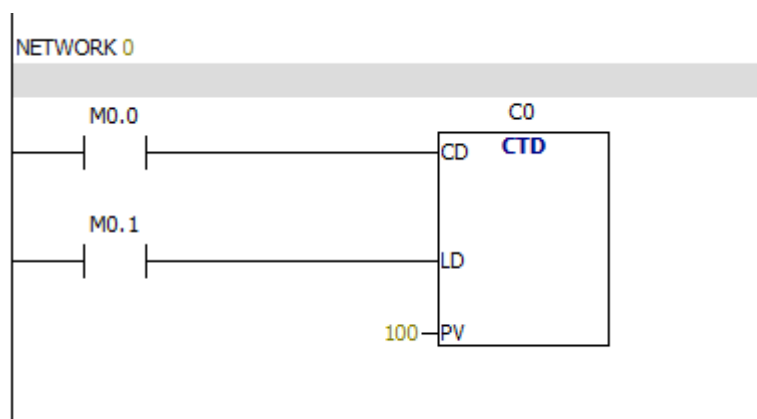


The bit of CD is converted from 0 to 1 and the current value minus 1. When the current value is equal to 0, the counter is opened and counter stops count. When the LD bit is equal to 1, counter bit is restored and the preset value is loaded into the current value.

Counter range: C xxx=C0~C255

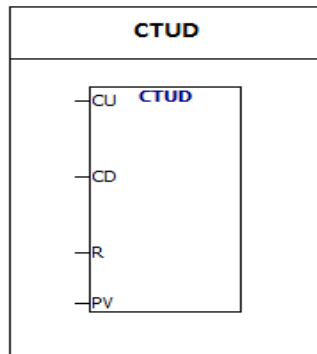
Attention: The counter number of each counter is different.

Example:



6.6.3 CTUD

| Input/output | Operand | Data type |
|--------------|--|-----------|
| C xxx | Constant(C0—C255) | word |
| CU, CD | Enable bit | Boolean |
| R | Enable bit | Boolean |
| PV | VW, IW, QW, MW, LW, SMW, AC, T, C, AIW,constant, *VD, *AC, *LD, SW | Integer |



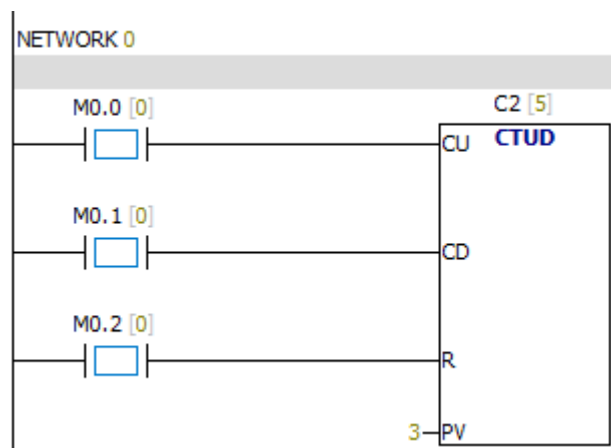
CU bit gets a high level and the current value of the counter plus 1. The bit of CD is converted from 0 to 1 and the current value minus 1. When the current value is greater than or equal to the present value, the counter bit opens. The maximum value of the counter is 32767, and the minimum value is -32768. When R gets a high level, the

counter is restored.

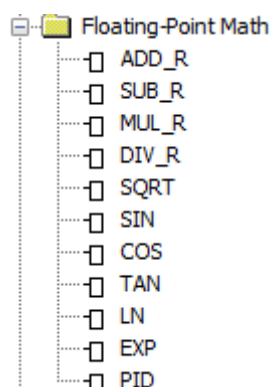
Counter range: C xxx=C0~C255

Attention: The counter number of each counter is different.

Example:



6.7 Floating point calculation



6.7.1 ADD-R&SUB-R

Input/output Operand

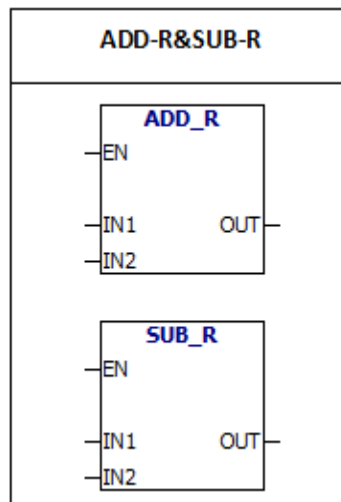
Data type

IN1, IN2 VD, ID, QD, MD, SD, SMD, LD, AC, constant, *VD, *LD, *AC

Real number

OUT VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

Real number



ADD-R: Adding N1 and N2, the result is put into the output buffer.

SUB-R: N1 minus N2, the result is put into the output buffer.

N1, N2, and OUT are 32 bits of real numbers.

$$IN1 + IN2 = OUT$$

$$IN1 - IN2 = OUT$$

Special memory bit:

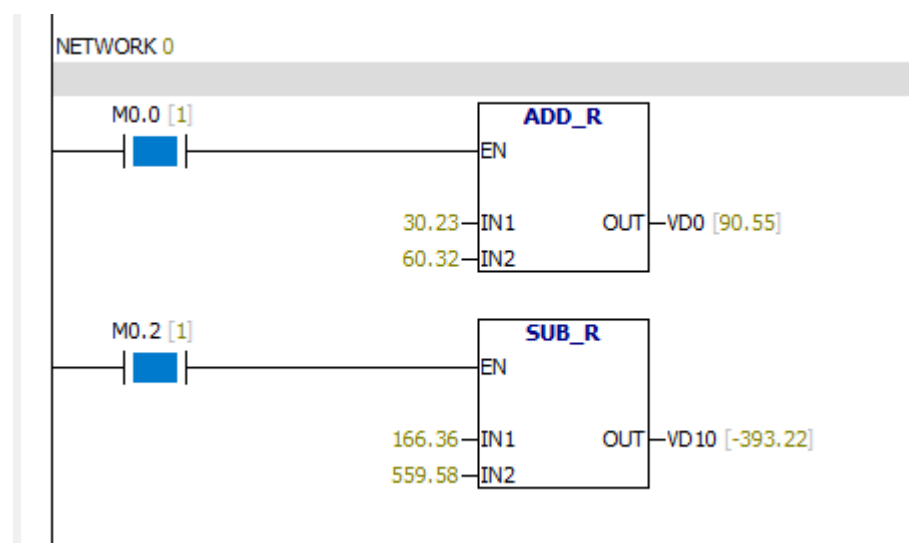
SM1.0 Zero result

SM1.1 Overflow

SM1.2 Negative result

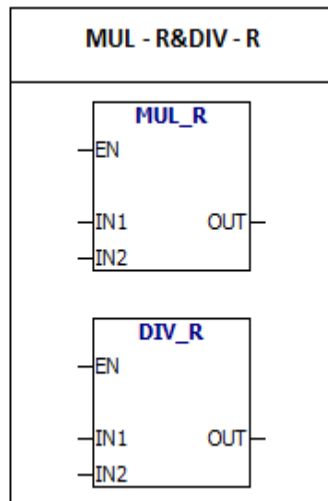
SM1.1 is used to indicate overflow errors and illegal values.

Example:



6.7.2 MUL - R&DIV - R

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN1, IN2 | VD, ID, QD, MD, SMD, SD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Real number |



MUL - R: IN1 multiplied by IN2, the result is put into the output buffer.

DIV - R: IN1 divided by IN2, the result is put into the output buffer.

IN1, IN2, and OUT are 32 bits of real numbers.

$$IN1 * IN2 = OUT$$

$$IN1 / IN2 = OUT$$

Error conditions:

SM1.1 Overflow

SM1.3 The divisor is 0

Special memory bit:

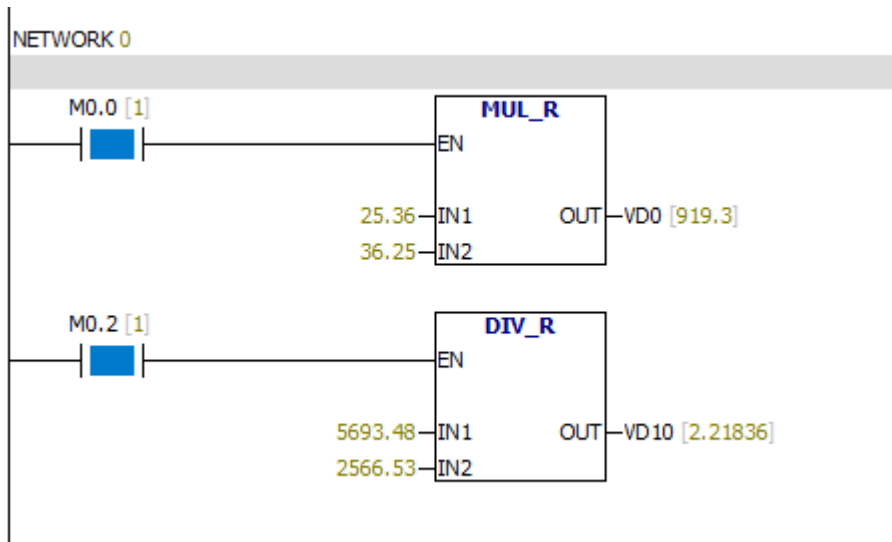
SM1.0 Zero result

SM1.1 Overflow

SM1.2 Negative result

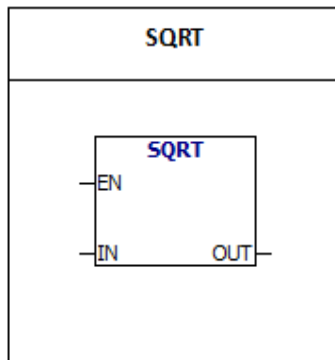
SM1.3 The divisor is 0

Example:



6.7.3 SQRT

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SMD, SD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Real number |



SQRT: Enter a 32 bits real number (IN). Take “IN” square root and output 32 bits real number.

Formula:

$$\sqrt{IN} = OUT$$

Error conditions:

SM1.1 Overflow

Special memory bits:

SM1.0 Zero result

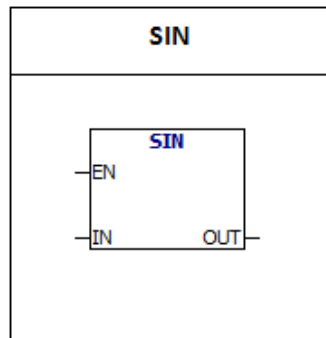
SM1.1 Overflow

SM1.2 Negative result

SM1.1 is used for indicating overflow errors and illegal values.

6.7.4 SIN

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SMD, SD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Real number |



SIN: Perform trigonometric operations on the input radian value and put the result into OUT. You can use the angle value multiplied by 1.745329E-2 to get the value of the radian. The value of the input "IN" is radian.

SM1.1 is used for indicating overflow errors and illegal values.

Error conditions:

SM1.1 Overflow

Special memory bit:

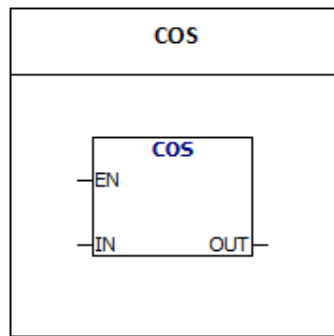
SM1.0 Zero result

SM1.1 Overflow

SM1.2 Negative result

6.7.5 COS

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SMD, SD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Real number |



COS: Perform trigonometric operations on the input radian value and put the result into OUT. You can use the angle value multiplied by 1.745329E-2 to get the value of the radian. The value of the input “IN” is radian.

SM1.1 is used for indicating overflow errors and illegal values.

Error conditions:

SM1.1 Overflow

Special memory bit:

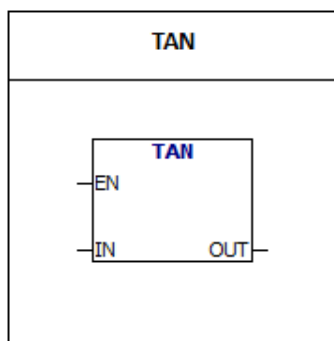
SM1.0 Zero result

SM1.1 Overflow

SM1.2 Negative result

6.7.6 TAN

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SMD, SD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Real number |



TAN: Perform trigonometric operations on the input radian value and put the result into OUT. You can use the angle value multiplied by 1.745329E-2 to get the value of the radian. The value of the input “IN” is radian.

SM1.1 is used for indicating overflow errors and illegal values.

error conditions:

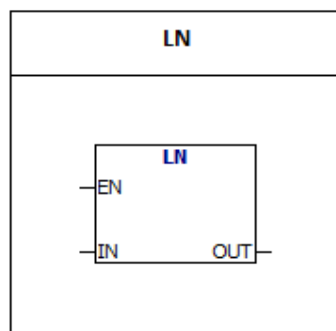
SM1.1 Overflow

Special memory bit:

- SM1.0 Zero result
- SM1.1 Overflow
- SM1.2 Negative result

6.7.7 LN

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SMD, SD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Real number |



LN: Use the input value to perform natural logarithm calculation and put the result in OUT.

The output value $\times 2.302585 \approx$ Natural logarithm of 10

SM1.1 is used for indicating overflow errors and illegal values.

error conditions:

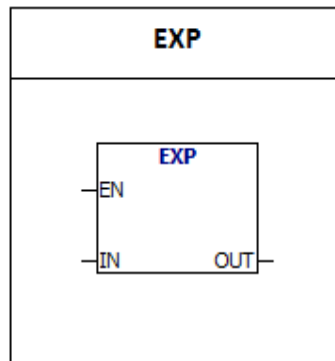
- SM1.1 Overflow

Special memory bit:

- SM1.0 Zero result
- SM1.1 Overflow
- SM1.2 Negative result

6.7.8 EXP

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SMD, SD, LD, AC, constant, *VD, *LD, *AC | Real number |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Real number |



EXP: Input value is N and output value is e^N .

N is a real number.

SM1.1 is used for indicating overflow errors and illegal values.

Example:

$$5 \text{ cube} = 5^3 = \text{EXP}(3 * \text{LN}(5)) = 125$$

$$\text{The cube root of } 125 = 125^{(1/3)} = \text{EXP}(1/3 * \text{LN}(125)) = 5$$

$$5 \text{ cubic square root} = 5^{(3/2)} = \text{EXP}(3/2 * \text{LN}(5)) = 11.18034$$

Error condition:

0006 Indirect address

SM1.1 Overflow

Special memory bit:

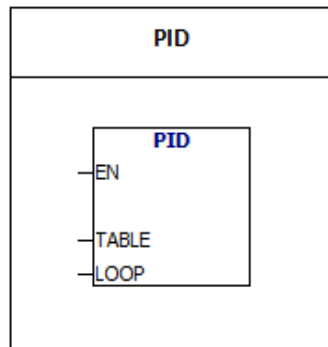
SM1.0 Zero result

SM1.1 Overflow

SM1.2 Negative result

6.7.9 PID

| Input/output | Operand | Data type |
|--------------|------------------|-----------|
| TBL | VB | Byte |
| LOOP | Constant(0 to 7) | Byte |



According to the parameters in the TBL, PID instruction performs the PID operation. Up to 8 PID instructions can be used in the program, the value of LOOP is the loop number of PID. PID loop number can't be the same, otherwise it will cause interference. Parameters in the TBL parameter table includes: Process, set value, output,

gain, sampling time, integration time, differential time, the last time integral term, the last time the amount of the process.

The parameter table contains 36 bytes:

| Offset | Meaning | Format | Type | Explain |
|--------|-------------------------------------|--------|--------------|--------------------------|
| 0 | PV_n Process quantity | DINT | Input | 0.0~1.0 |
| 4 | SP_n Set point | DINT | Input | 0.0~1.0 |
| 8 | Mn Output value | DINT | Input/Output | 0.0~1.0 |
| 12 | Kc gain | | Input | Ratio constant |
| 16 | Ts Sampling time | DINT | Input | Ms Positive |
| 20 | T_I Integral time | DINT | Input | S Positive |
| 24 | T_D Differential time | DINT | Input | S Positive |
| 28 | MI_{n-1} Last time integral value | DINT | Input/Output | Last time integral value |
| 32 | PV_{n-1} Last time process | DINT | Input/Output | Last time process |

Mathematical formula of PID loop instruction:

$$M_n = MP_n + MI_n + MD_n$$

- M_n : Output value
- MP_n : Proportion term
- MI_n : Integral term
- MD_n : Differential term

Proportion term

$$MP_n = K_c * (SP_n - PV_n)$$

- MP_n : Proportion term
- K_c : gain
- SP_n : Set point
- PV_n : Process quantity

Integral term:

$$MI_n = K_c * T_s / T_I * (SP_n - PV_n) + MI_{n-1}$$

- MI_n : Integral term
- K_c : gain
- T_s : Sampling time
- T_I : Integral time
- SP_n : Set point
- PV_n : Process quantity
- MI_{n-1} : Last time integral term

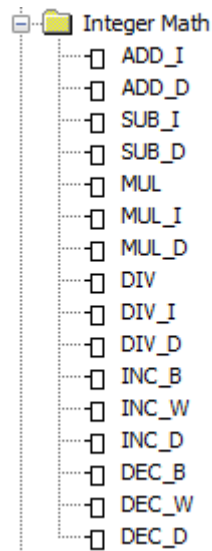
Differential term:

$$MD_n = K_c * T_D / T_s * (PV_{n-1} - PV_n)$$

- MD_n : Differential term
- K_c : gain

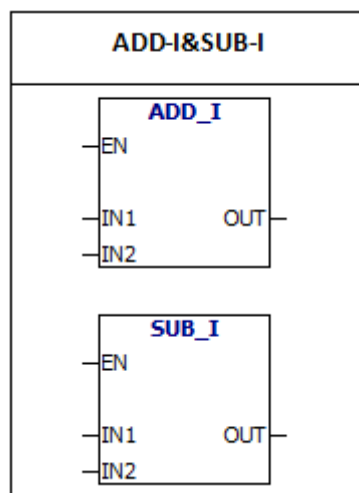
- T_D : Differential time
- T_s : Sampling time
- PV_{n-1} : Last time process variable
- PV_n : Process variable

6.8 Integer operations



6.8.1 ADD-I&SUB-I

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN1, IN2 | VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, constant, *VD, *LD, *AC | Integer |
| OUT | VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *LD, *AC | Integer |



ADD-I: $IN1 + IN2 = OUT$ Both input and output are 16 bits integers.

SUB-I: $IN1 - IN2 = OUT$ Both input and output are 16 bits integers.

$IN1 + IN2 = OUT$

$IN1 - IN2 = OUT$

SM1.1 is used for indicating overflow errors and illegal values.

Error conditions:

0006 Indirect address

SM1.1 overflow

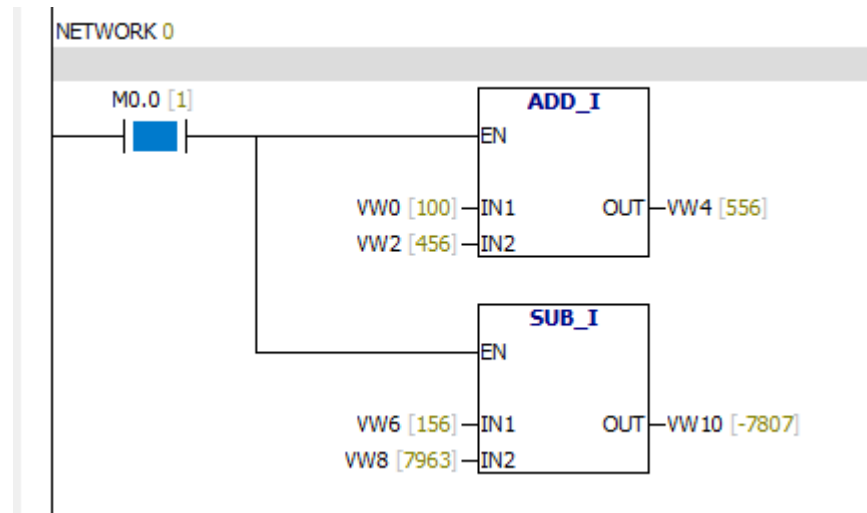
Special memory bit:

SM1.0 Zero result

SM1.1 overflow

SM1.2 Negative result

Example:



6.8.2 ADD- DI & SUB- DI

Input/output Operand

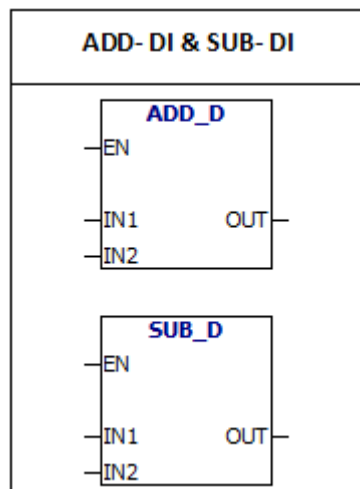
Data type

IN1, IN2 VD, ID, QD, MD, SMD, SD, LD, AC, HC, Constant, *VD, *LD, *AC

Double integer

OUT VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Double integer



ADD- DI: $IN1 + IN2 = OUT$ Both input and output are 32 bits integers.

SUB- DI: $IN1 - IN2 = OUT$ Both input and output are 32 bits integers.

$IN1 + IN2 = OUT$

$IN1 - IN2 = OUT$

SM1.1 is used for indicating overflow errors and illegal values.

Error conditions:

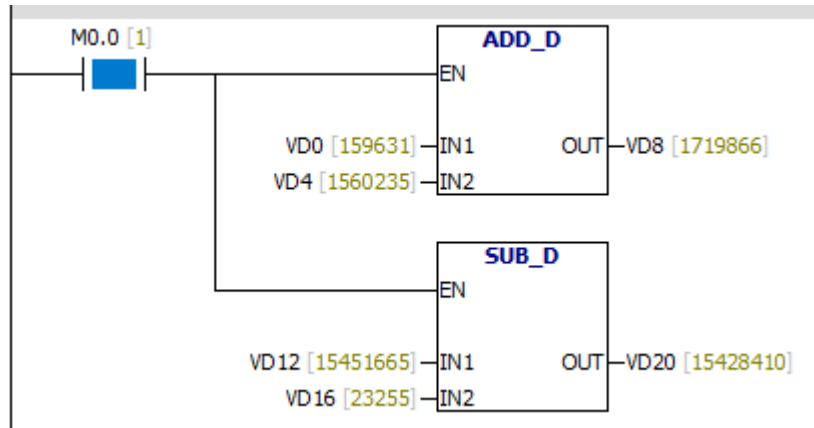
0006 Indirect address

SM1.1 overflow

Special memory bit:

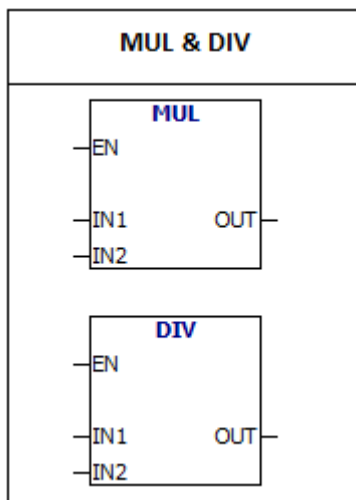
- SM1.0 Zero result
- SM1.1 overflow
- SM1.2 Negative result

Example:



6.8.3 MUL & DIV

| Input/output | Operand | Data type |
|--------------|---|----------------|
| IN1, IN2 | VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, constant, *VD, *LD, *AC | Integer |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Double Integer |



MUL: $IN1 \times IN2 = OUT$ Input 16 bits integers and output 32 bits integer.

DIV: $IN1 / IN2 = OUT$ Input 16 bits integers and the output result is 32 bits. The result includes a 16 bits remainder (high) and a 16 bits quotient (low).

$$IN1 * IN2 = OUT$$

$$IN1 / IN2 = OUT$$

SM1.1 is used for indicating overflow errors and illegal values.

error conditions:

- 0006 Indirect address
- SM1.1 overflow

SM1.3 The divisor is 0

Special memory bit:

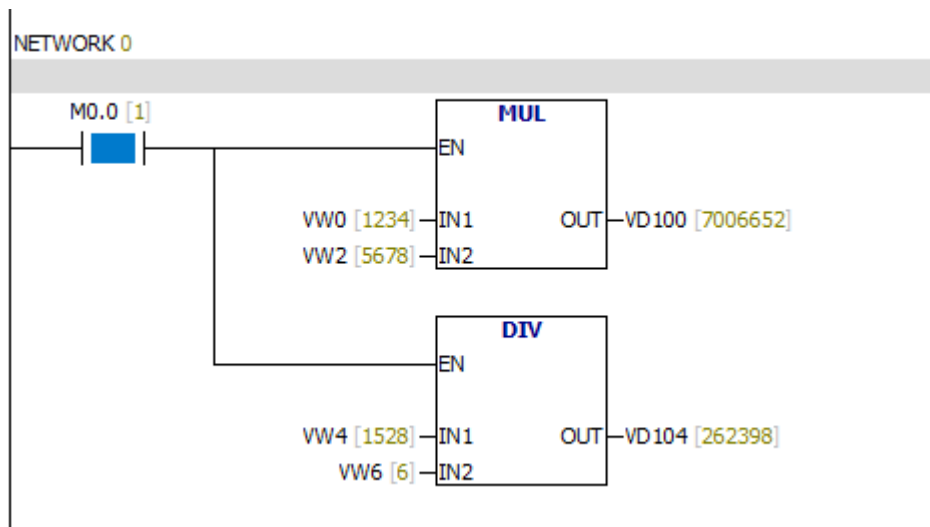
SM1.0 Zero result

SM1.1 overflow

SM1.2 Negative result

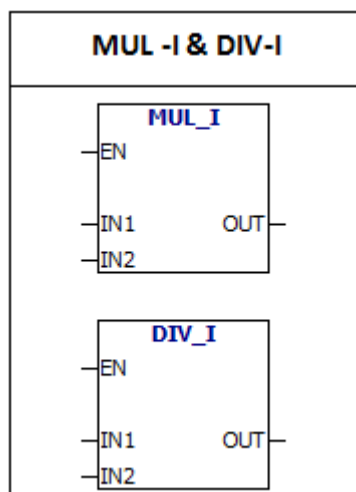
SM1.3 The divisor is 0

Example:



6.8.4 MUL -I & DIV-I

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN1, IN2 | VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, constant, *VD, *LD, *AC | Integer |
| OUT | VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC | Integer |



MUL -I: $IN1 * IN2 = OUT$ Both input and output are 16 bits integers.

DIV-I: $IN1 / IN2 = OUT$ Both input and output are 16 bits integers. Output is quotient. There is no remainder.

If the output is larger than a word, then set overflow bit.

$$IN1 * IN2 = OUT$$

$$IN1 / IN2 = OUT$$

SM1.1 is used for indicating overflow errors and illegal values.

Error conditions:

0006 Indirect address

SM1.1 overflow

SM1.3 The divisor is 0

Special memory bit:

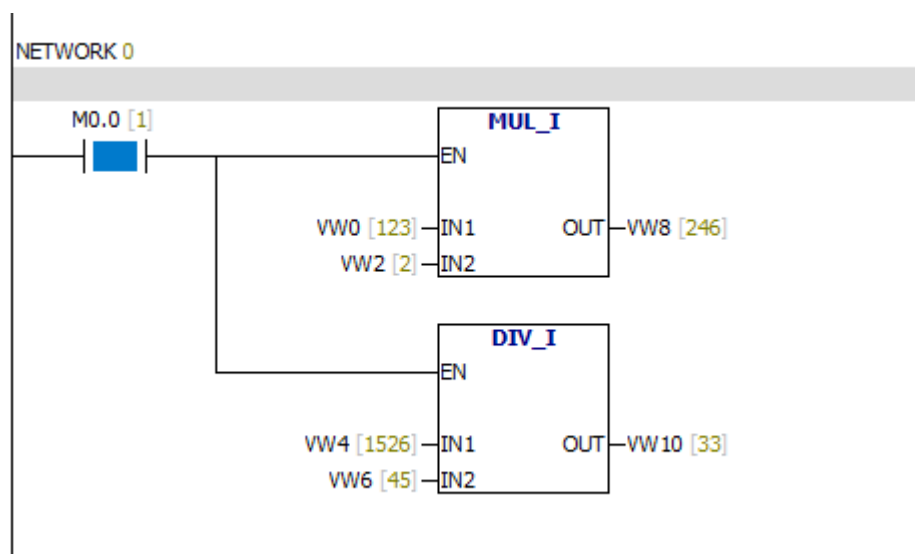
SM1.0 Zero result

SM1.1 overflow

SM1.2 Negative result

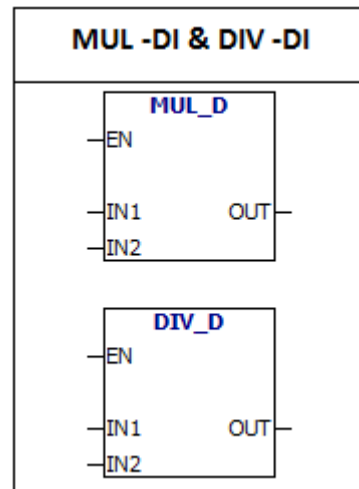
SM1.3 The divisor is 0

Example:



6.8.5 MUL -DI & DIV -DI

| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN1, IN2 | VD, ID, QD, MD, SMD, SD, LD, HC, AC, constant, *VD, *LD, *AC | Double integer |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Double integer |



MUL -DI: $IN1 * IN2 = OUT$ Both input and output are 32 bits integers.

DIV -DI: $IN1 / IN2 = OUT$ Both input and output are 32 bits integers.

Output is quotient. There is no remainder.

$IN1 * IN2 = OUT$

$IN1 / IN2 = OUT$

SM1.1 is used for indicating overflow errors and illegal values.

Error conditions:

0006 Indirect address

SM1.1 overflow

SM1.3 The divisor is 0

Special memory bit:

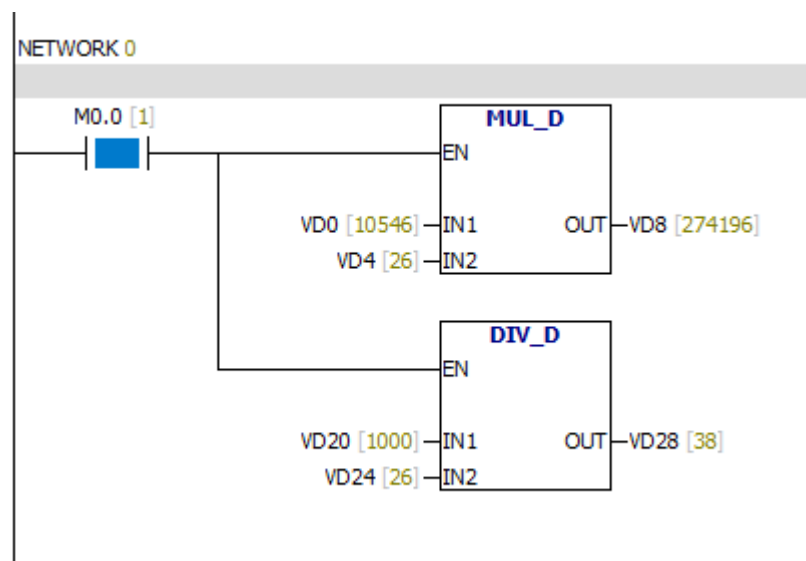
SM1.0 Zero result

SM1.1 overflow

SM1.2 Negative result

SM1.3 The divisor is 0

Example:



6.8.6 INC-B & DEC-B

Input/output Operand

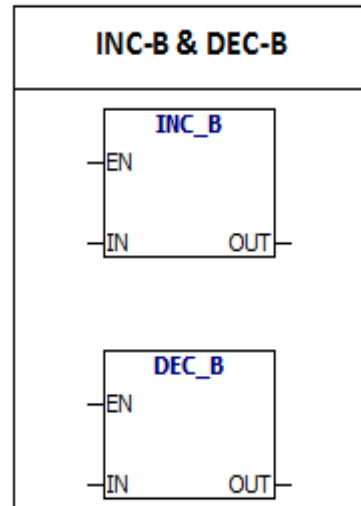
Data type

IN VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC

Byte

OUT VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

Byte



INC-B: $IN + 1 = OUT$ Both input and output are 8 bits integers.

DEC-B: $IN - 1 = OUT$ Both input and output are 8 bits integers.

The two instructions operations do not take symbols.

$IN + 1 = OUT$

$IN - 1 = OUT$

Error conditions:

0006 Indirect address

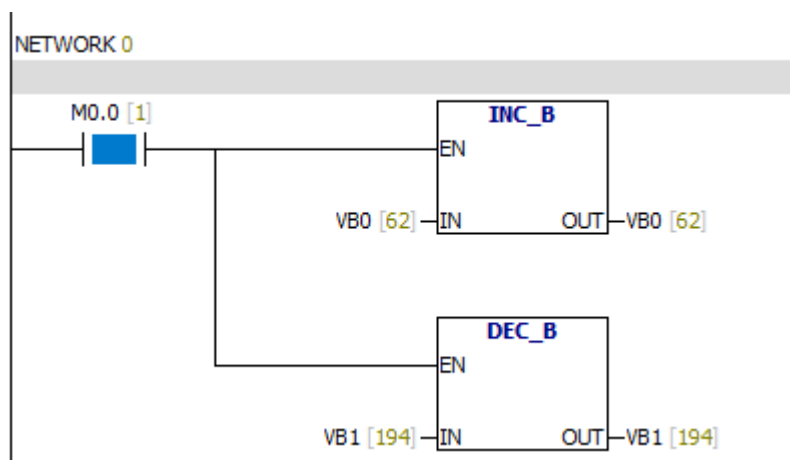
SM1.1 overflow

Special memory bit:

SM1.0 Zero result

SM1.1 overflow

Example:



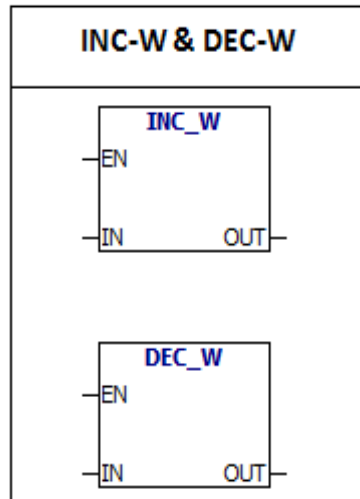
6.8.7 INC-W & DEC-W

Input/output Operand

Data type

IN VW, IW, QW, MW, SW, SMW, AC, AIW, LW, T, C, constant, *VD, *LD, *AC Integer

OUT VW, IW, QW, MW, SW, SMW, LW, AC, T, C, *VD, *LD, *AC Integer



INC-W: $IN + 1 = OUT$ Both input and output are 16 bits integers.

DEC-W: $IN - 1 = OUT$ Both input and output are 16 bits integers.

The two instructions operations take with symbols (16#7FFF > 16#8000).

$IN + 1 = OUT$

$IN - 1 = OUT$

Error conditions:

0006 Indirect address

SM1.1 overflow

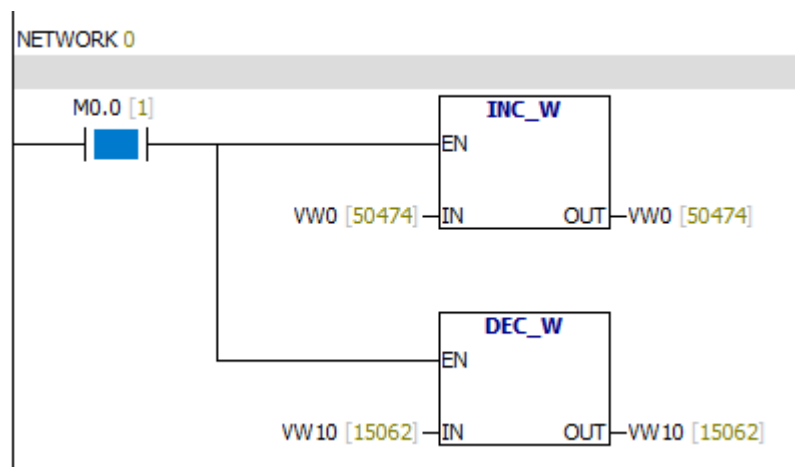
Special memory bit:

SM1.0 Zero result

SM1.1 overflow

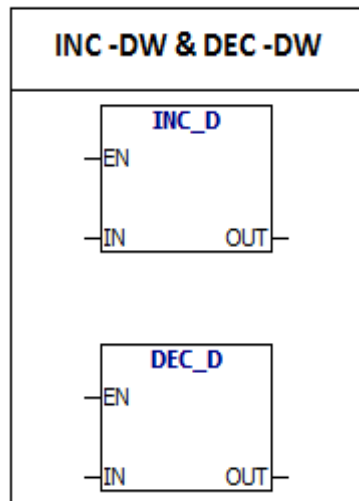
SM1.2 Negative result

Example:



6.8.8 INC -DW & DEC -DW

| Input/output | Operand | Data type |
|--------------|--|----------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, AC, HC, constant, *VD, *LD, *AC | Double integer |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Double integer |



INC -DW: $IN + 1 = OUT$ Both input and output are 32 bits double integers.

DEC -DW: $IN - 1 = OUT$ Both input and output are 32 bits double integers.

$IN + 1 = OUT$

$IN - 1 = OUT$

The two instructions operations take with symbols. (16#7FFFFFFF > 16#80000000).

Error conditions:

0006 Indirect address

SM1.1 overflow

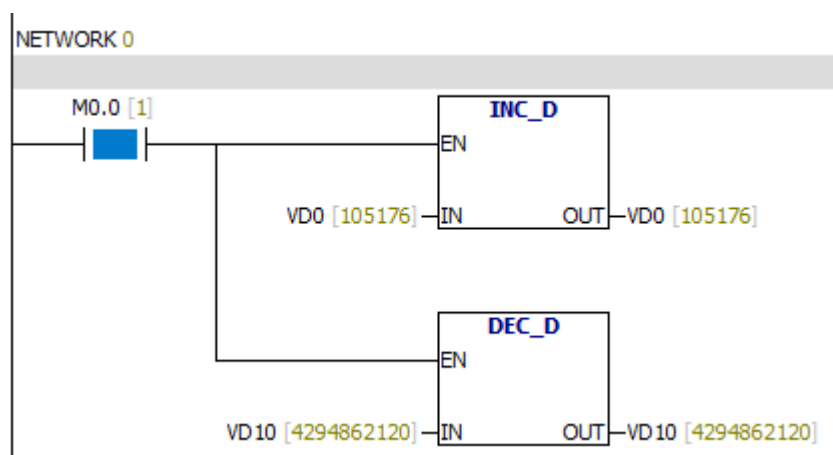
Special memory bit:

SM1.0 Zero result

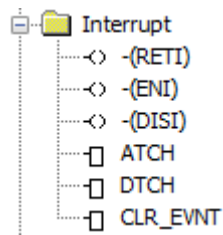
SM1.1 overflow

SM1.2 Negative result

Example:

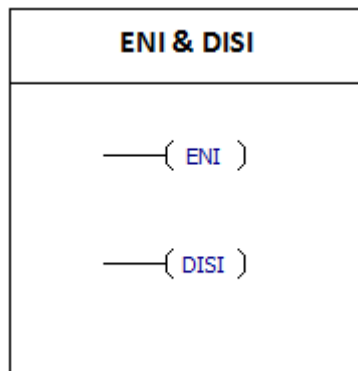


6.9 Interrupt



6.9.1 ENI & DISI

| Operand | Data type |
|---------|-----------|
| Nothing | Nothing |



Interrupt enable (ENI): If the instruction is activated, all interrupts can be used.

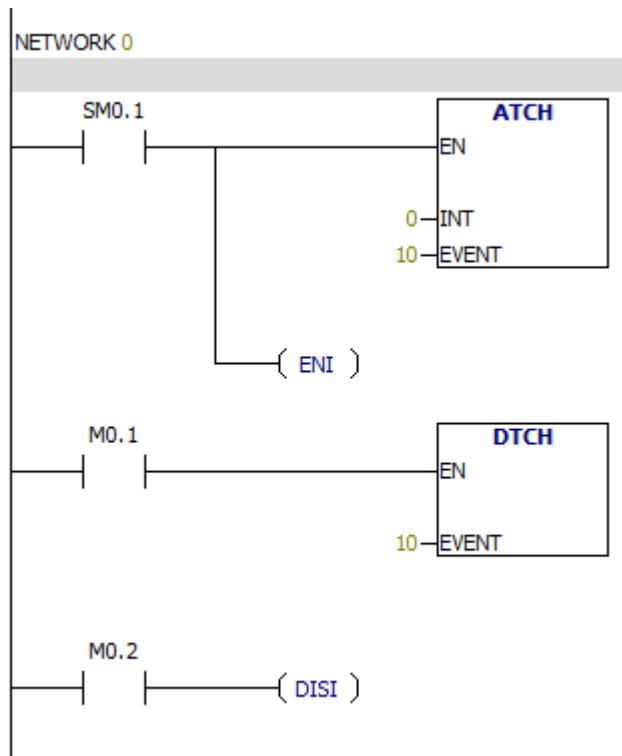
Interrupt disable (DISI): If the instruction is activated, all interrupts can't be used.

When the DISI instruction is used, the interrupt events will be queued.

Interrupt Events:

| | | | |
|-------------------|-------------------|----|------------------|
| I1.2 Rising edge | PLC_EVENT_INPUTP0 | 0 | Highest priority |
| I1.4 Rising edge | PLC_EVENT_INPUTP1 | 1 | High priority |
| Timer interrupt 0 | PLC_EVENT_TIMER0 | 10 | Low priority |
| Timer interrupt 1 | PLC_EVENT_TIMER1 | 11 | Lowest priority |

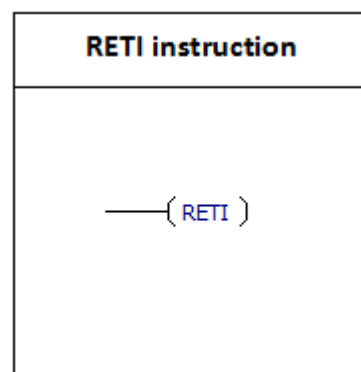
Example:



6.9.2 RETI instruction

Operand Data type

Nothing Nothing



RETI: When the Logic in front of the RETI instruction is 1, PLC execution returns from interrupt.

Interrupt Events:

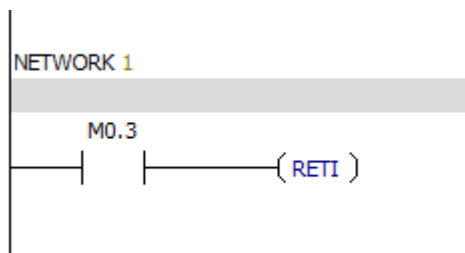
| | | | |
|-------------------|-------------------|----|------------------|
| I1.2 Rising edge | PLC_EVENT_INPUTP0 | 0 | Highest priority |
| I1.4 Rising edge | PLC_EVENT_INPUTP1 | 1 | High priority |
| Timer interrupt 0 | PLC_EVENT_TIMER0 | 10 | Low priority |
| Timer interrupt 1 | PLC_EVENT_TIMER1 | 11 | Lowest priority |

Interrupt use guide

Interrupt routine offers a quick response to a particular internal or external event. Interrupt routine should be concise and efficient, so it can accelerate the speed of execution.

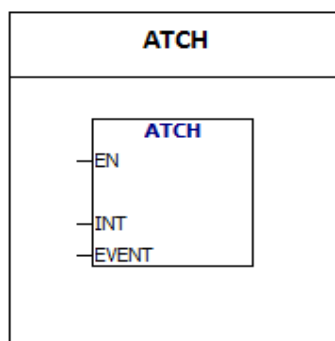
Limit: DISI, ENI, HDEF, LSCR, and END instructions can not be used in the interrupt routine.

Example:



6.9.3 ATCH

| Input/output | Operand | Data type |
|--------------|----------------|-----------|
| INT | Constant 0-127 | Byte |
| EVNT | Constant 0-33 | Byte |



ATCH: The interrupt event (EVNT) is connected to the interrupt routine number (INT) by the “ATCH” instruction, and then activates the interrupt event.

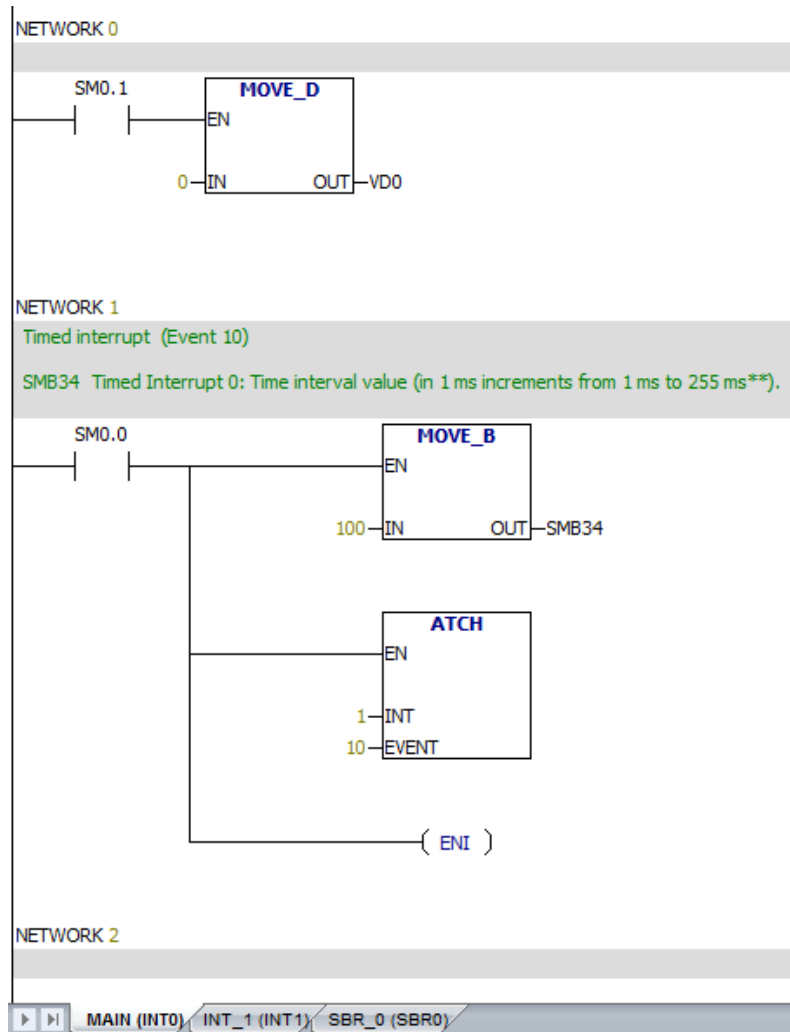
You can attach more than one interrupt events to an interrupt routine. However, an interrupt event can't be attached to the multiple interrupt routines. When you attach an interrupt event to an interrupt routine, the interrupt is automatically enabled. When the DISI instruction is used, the interrupt events will be queued.

If you want to disable a single interrupt event, you can use the "DTCH" instruction.

Interrupt Events:

| | | | |
|-------------------|-------------------|----|------------------|
| I1.2 Rising edge | PLC_EVENT_INPUTP0 | 0 | Highest priority |
| I1.4 Rising edge | PLC_EVENT_INPUTP1 | 1 | High priority |
| Timer interrupt 0 | PLC_EVENT_TIMER0 | 10 | Low priority |
| Timer interrupt 1 | PLC_EVENT_TIMER1 | 11 | Lowest priority |

Example: VD0 increases by 1 every 100ms



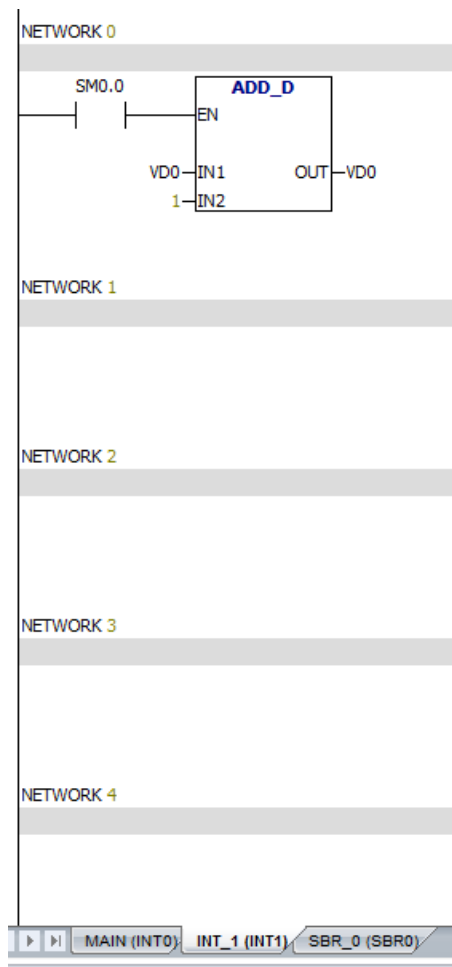
Notes:

Timed interrupt (Event 10)

SMB34 Timed Interrupt 0: Time interval value (in 1 ms increments from 1 ms to 255 ms**).

Timed interrupt (Event 11)

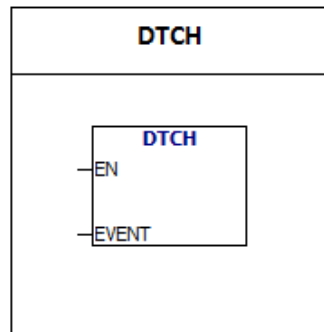
SMB35 Timed Interrupt 1: Time interval value (in 1 ms increments from 1 ms to 255 ms**).



“ATCH” instruction only needs to be connected once.

6.9.4 DTCH

| Input/output | Operand | Data type |
|--------------|-----------------|-----------|
| EVNT | Constant (0-33) | Byte |



Interrupt separation (DTCH) instruction cancels the association between interrupt event (EVNT) and interrupt routine, and disables the interrupt event.

The interrupt event (EVNT) is connected to the interrupt routine number (INT) by the "ATCH" instruction, and then activates the interrupt event.

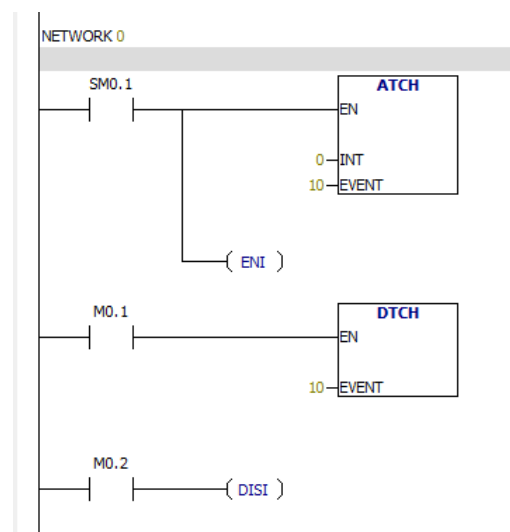
You can attach more than one interrupt events to an interrupt routine. However, an interrupt event can't be attached to the multiple interrupt routines. When you attach an interrupt event to an interrupt routine, the interrupt is automatically enabled. When the DISI instruction is used, the interrupt events will be queued.

If you want to disable a single interrupt event, you can use the "DTCH" instruction.

Interrupt Events:

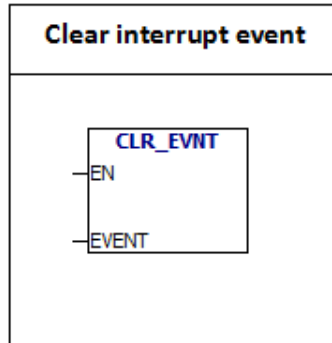
| | | | |
|-------------------|-------------------|----|------------------|
| I1.2 Rising edge | PLC_EVENT_INPUTP0 | 0 | Highest priority |
| I1.4 Rising edge | PLC_EVENT_INPUTP1 | 1 | High priority |
| Timer interrupt 0 | PLC_EVENT_TIMER0 | 10 | Low priority |
| Timer interrupt 1 | PLC_EVENT_TIMER1 | 11 | Lowest priority |

Example:



6.9.5 Clear interrupt event

| | | |
|--------------|----------|-----------|
| Input/output | Operand | Data type |
| EVNT | Constant | Byte |

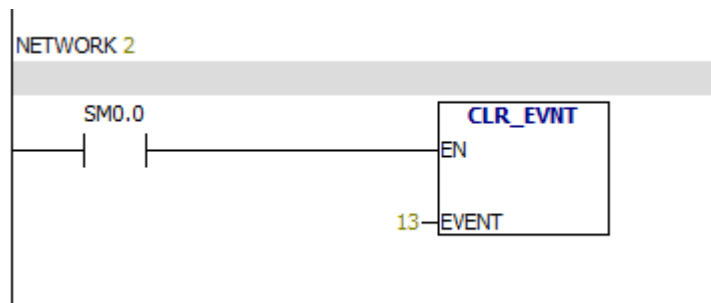


CLR - EVNT instruction will remove all types of EVNT interrupt events in interrupt queue. This instruction is used for removing unnecessary interrupts.

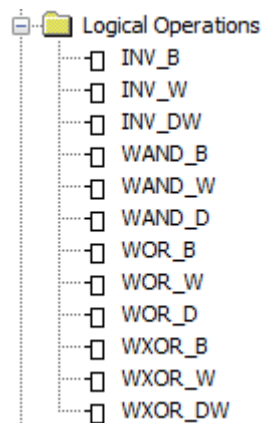
Interrupt Events:

| | | | |
|-------------------|-------------------|----|------------------|
| I1.2 Rising edge | PLC_EVENT_INPUTP0 | 0 | Highest priority |
| I1.4 Rising edge | PLC_EVENT_INPUTP1 | 1 | High priority |
| Timer interrupt 0 | PLC_EVENT_TIMER0 | 10 | Low priority |
| Timer interrupt 1 | PLC_EVENT_TIMER1 | 11 | Lowest priority |

Example:

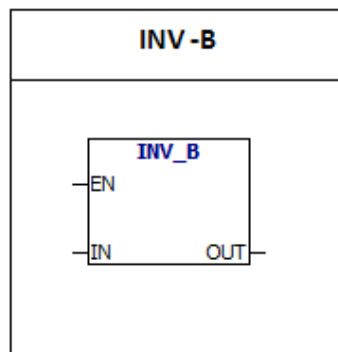


6.10 Logic operation



6.10.1 INV -B

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *AC, *LD | Byte |
| OUT | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD | Byte |



INV -B: The instruction performs the complement operation to the input byte and puts the result in OUT.

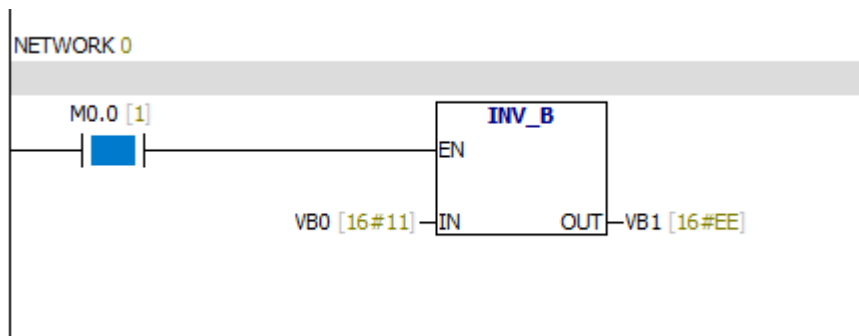
Error condition:

0006 Indirect address

Special memory bit:

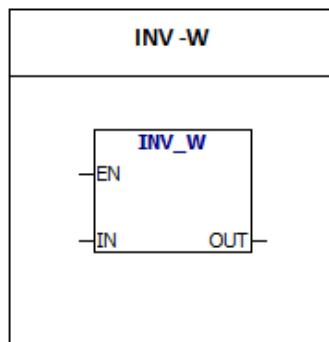
SM1.0 Zero result

Example:



6.10.2 INV -W

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN | VW, IW, QW, MW, SW, SMW, T, C, AIW, LW, AC, constant, *VD, *AC, *LD | word |
| OUT | VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *AC, *LD | word |



INV -W: The instruction performs the complement operation to the input word and puts the result in OUT.

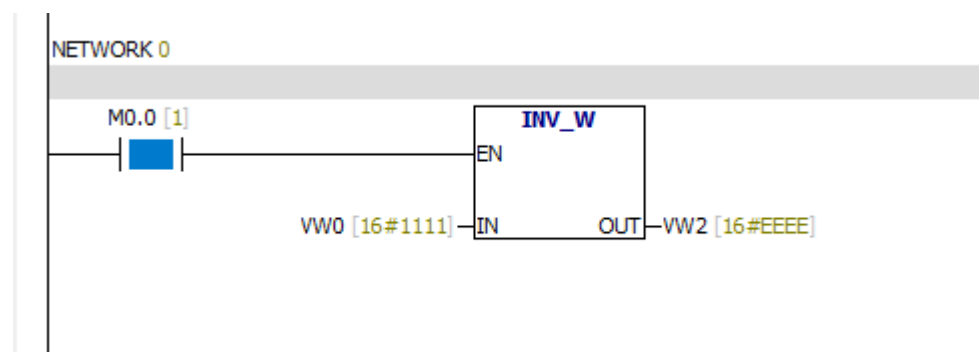
Error condition:

0006 Indirect address

Special memory bit:

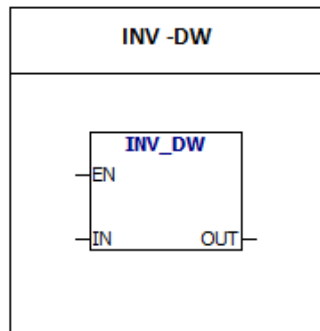
SM1.0 Zero result

Example:



6.10.3 INV -DW

| Input/output Operand | | Data type |
|----------------------|--|-------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, HC, AC, constant, *VD, *AC, *LD | Double word |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD | Double word |



INV -DW: The instruction performs the complement operation to the input word and puts the result in OUT.

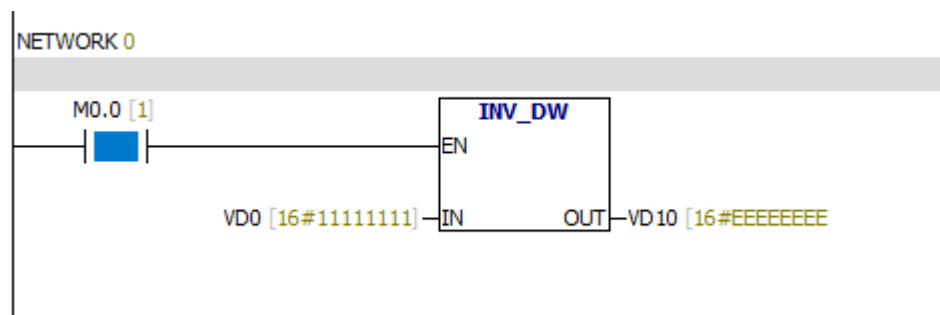
Error condition:

0006 Indirect address

Special memory bit:

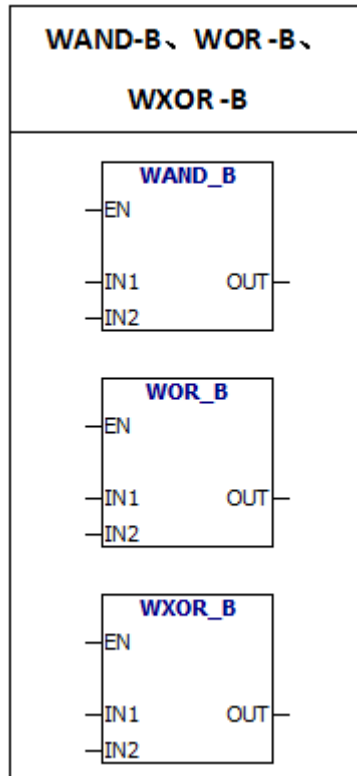
SM1.0 Zero result

Example:



6.10.4 WAND-B、WOR-B、WXOR-B

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN1, IN2 | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *AC, *LD | Byte |
| OUT | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD | Byte |



WAND -B: The instruction performs “And calculation” on IN1 and IN2. Then puts the result in out.

WOR -B: The instruction performs “OR calculation” on IN1 and IN2. Then puts the result in out.

WXOR -B: The instruction performs “XOR calculation” on IN1 and IN2. Then puts the result in out.

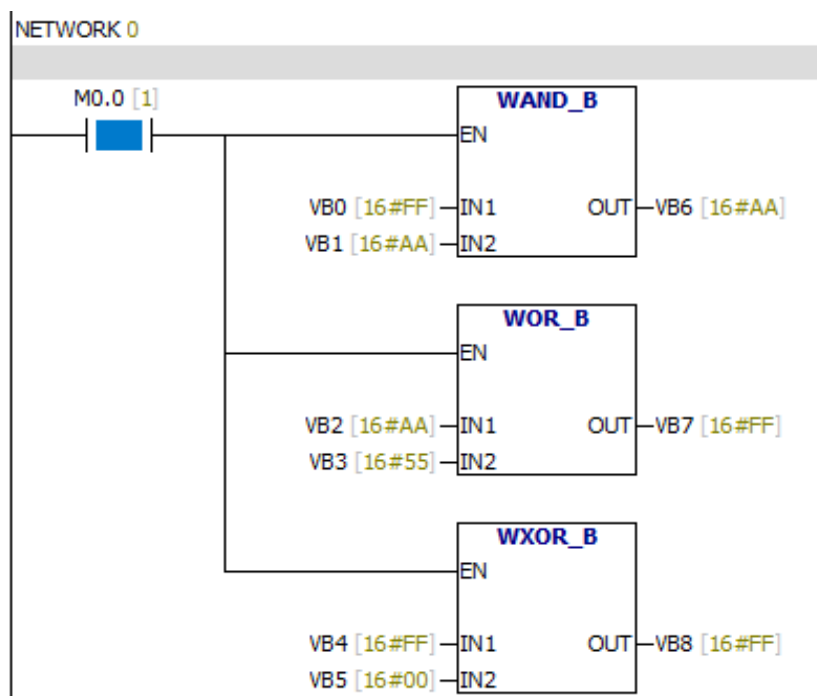
Error condition:

0006 Indirect address

Special memory bit:

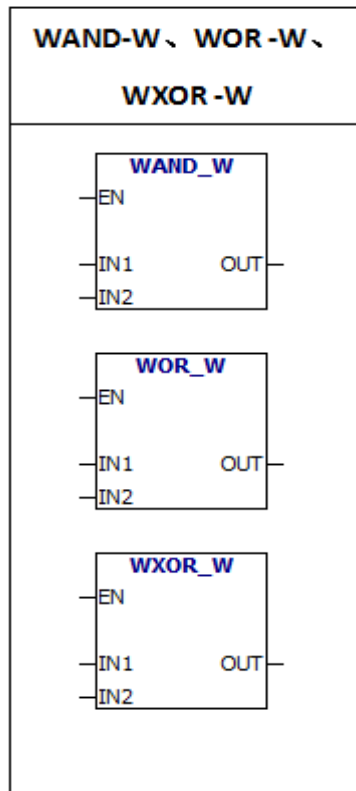
SM1.0 Zero result

Example:



6.10.5 WAND-W、WOR -W、WXOR -W

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN1, IN2 | VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, constant, *VD, *AC, *LD | word |
| OUT | VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *AC, *LD | word |



WAND -W: The instruction performs “And calculation” on IN1 and IN2. Then puts the result in out.

WOR -W: The instruction performs “OR calculation” on IN1 and IN2. Then puts the result in out.

WXOR -W: The instruction performs “XOR calculation” on IN1 and IN2. Then puts the result in out.

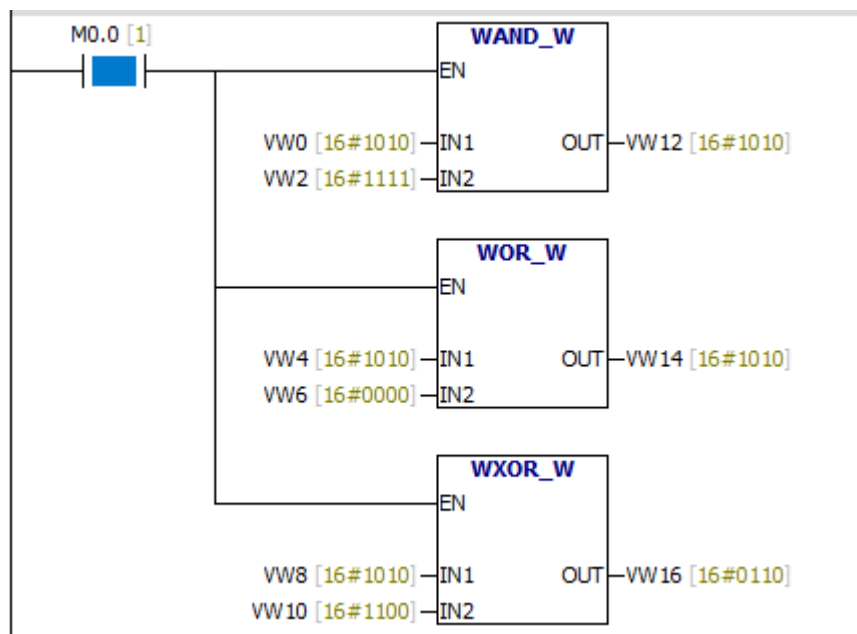
Error condition:

0006 Indirect address

Special memory bit:

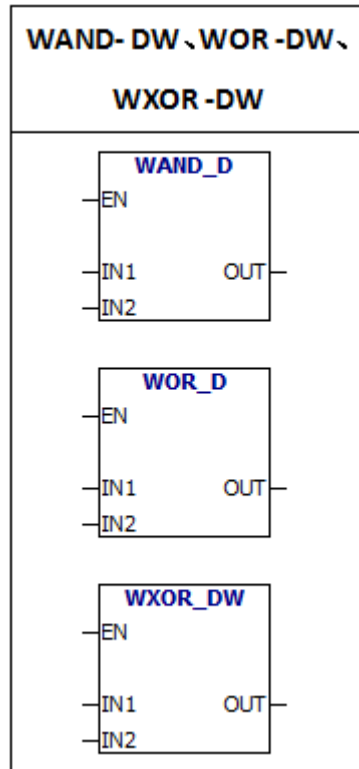
SM1.0 Zero result

Example:



6.10.6 WAND- DW、 WOR -DW、 WXOR -DW

| Input/output Operand | | Data type |
|----------------------|--|-------------|
| IN1, IN2 | VD, ID, QD, MD, SMD, AC, LD, HC, constant, *VD, *AC, SD, *LD | Double word |
| OUT | VD, ID, QD, MD, SMD, LD, AC, *VD, *AC, SD, *LD | Double word |



WAND -DW: The instruction performs “And calculation” on IN1 and IN2. Then puts the result in out.

WOR -DW: The instruction performs “OR calculation” on IN1 and IN2. Then puts the result in out.

WXOR -DW: The instruction performs “XOR calculation” on IN1 and IN2. Then puts the result in out.

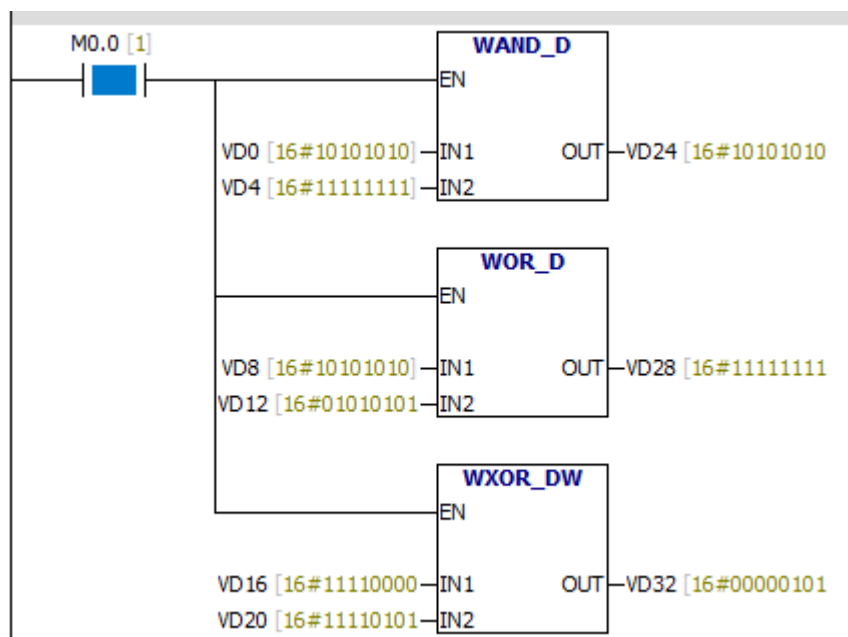
Error condition:

0006 Indirect address

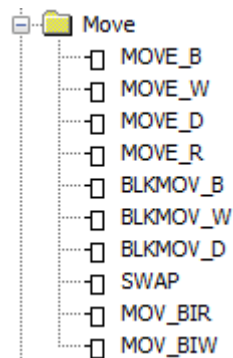
Special memory bit:

SM1.0 Zero result

Example:

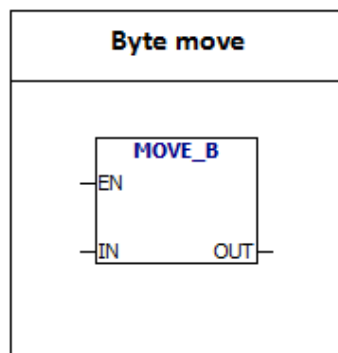


6.11 Move



6.11.1 Byte move

| Input/output Operand | Data type |
|---|-----------|
| IN VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | Byte |
| OUT VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC | Byte |

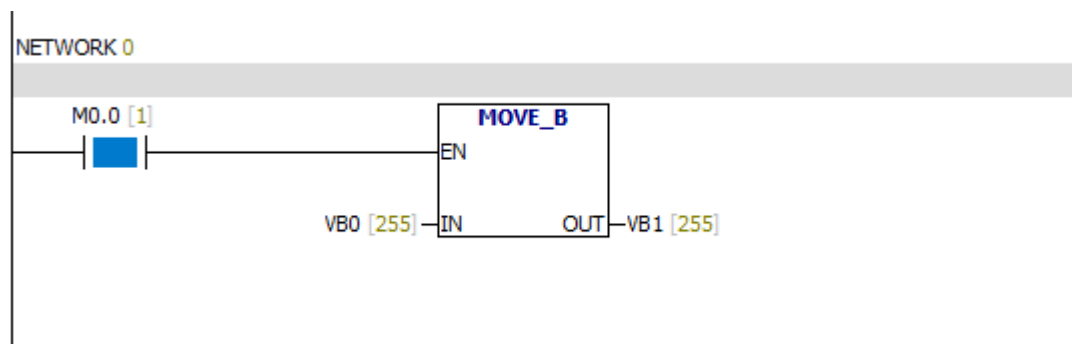


MOV -B: The instruction moves the input byte (IN) to the output byte (OUT), which does not change the original value.

Error condition:

0006 Indirect address

Example:



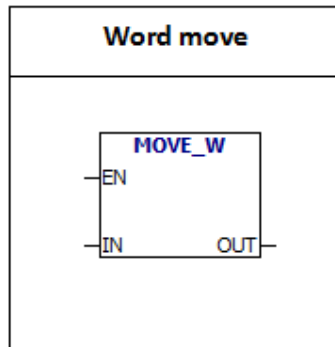
6.11.2 Word move

Input/output Operand

Data type

IN VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, constant, AC, *VD, *AC, *LD word, integer

OUT VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD word, integer

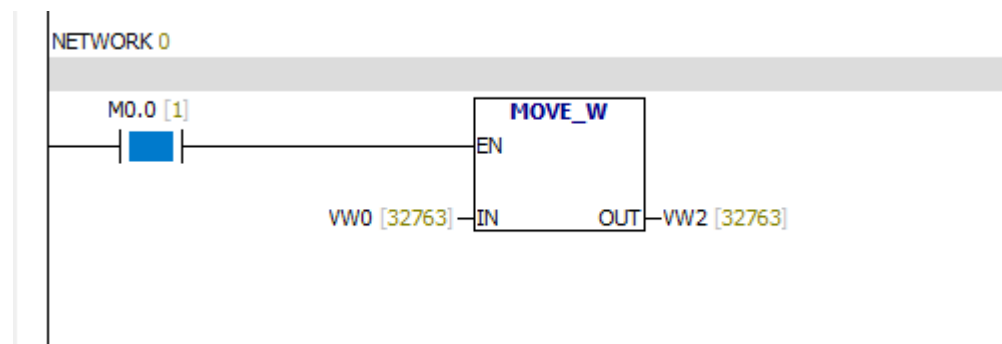


MOV -W: The instruction moves the input word (IN) to the output word (OUT), which does not change the original value.

Error condition:

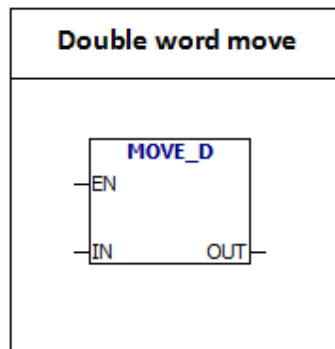
0006 Indirect address

Example:



6.11.3 Double word move

| Input/output | Operand | Data type |
|--------------|--|-----------------------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW AC, constant, *VD, *LD, *AC | Double word, double integer |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Double word, double integer |



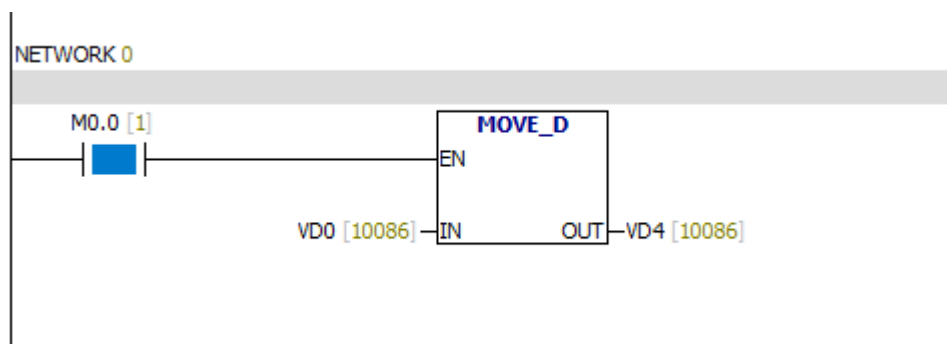
MOV -DW: The instruction moves the input double word (IN) to the output double word (OUT), which does not change the original value.

You can use the "MOVE-D" instruction to create a pointer.

Error condition:

0006 Indirect address

Example:



6.11.4 Real number move

Input/output Operand

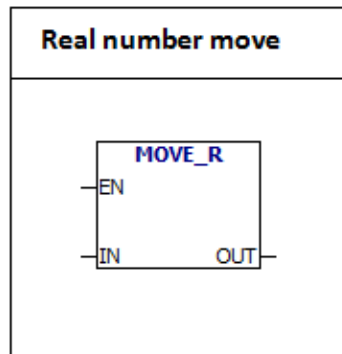
Data type

IN VD, ID, QD, MD, SD, SMD, LD, AC, constant, *VD, *LD, *AC

Real number

OUT VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

Real number

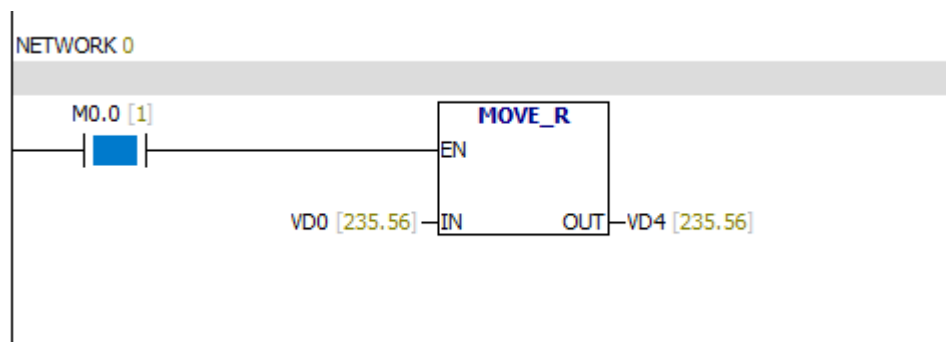


MOV -R: The instruction moves the input real number (IN) to the output real number (OUT), which does not change the original value.

Error condition:

0006 Indirect address

Example:



6.11.5 BLKMOV -B

Input/output Operand

Data type

IN VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD

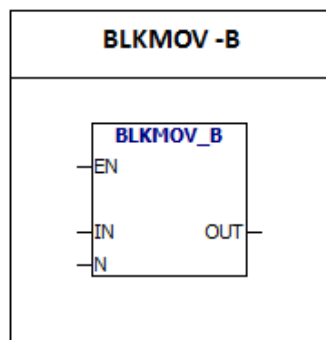
Byte

N VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *AC, *LD

Byte

OUT VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD

Byte



These successive “N” bytes which start with “IN” are moved to OUT.

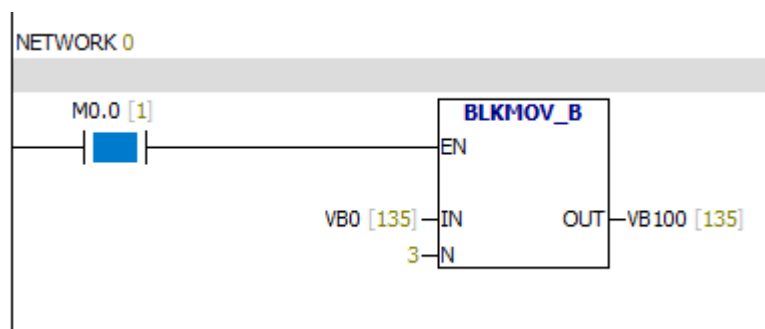
The range of N is from 1 to 255.

Error conditions:

0006 Indirect address

0091 Operating number is out of range

Example:

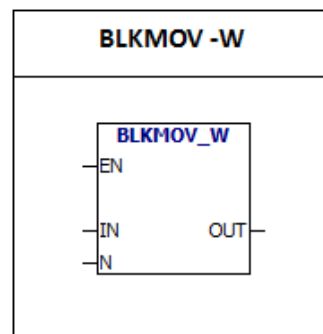


Status Chart

| | Address | Data Type | Value |
|--|---------|-----------|-------|
| | M0.0 | BOOL | 1 |
| | VB0 | USINT | 135 |
| | VB1 | USINT | 153 |
| | VB2 | USINT | 255 |
| | VB100 | USINT | 135 |
| | VB101 | USINT | 153 |
| | VB102 | USINT | 255 |

6.11.6 BLKMOV -W

| Input/output Operand | | Data type |
|----------------------|--|-----------|
| IN | VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, *VD, *LD, *AC | word |
| N | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | byte |
| OUT | VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, *VD, *LD, *AC | word |



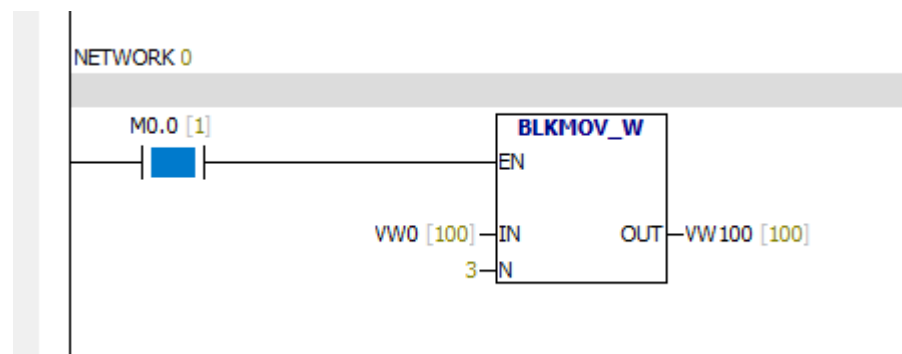
BLKMOV -W: These successive “N” words which start with “IN” are moved to OUT.

The range of N is from 1 to 255.

Error conditions:

- 0006 Indirect address
- 0091 Operating number is out of range

Example:



| Status Chart | | | |
|--------------|---------|-----------|-------|
| | Address | Data Type | Value |
| | M0.0 | BOOL | 1 |
| | VW0 | INT | 100 |
| | VW2 | INT | 101 |
| | VW4 | INT | 102 |
| | VW100 | INT | 100 |
| | VW102 | INT | 101 |
| | VW104 | INT | 102 |

6.11.7 BLKMOV -D

Input/output Operand

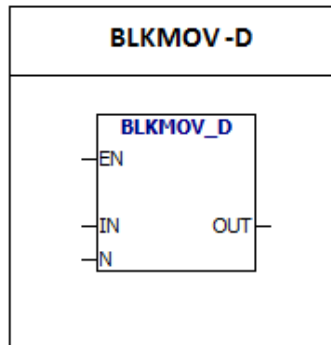
Data type

IN, OUT VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD

Double word

N VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *AC, *LD

Byte



BLKMOV - D: These successive “N” double words which start with “IN” are moved to OUT.

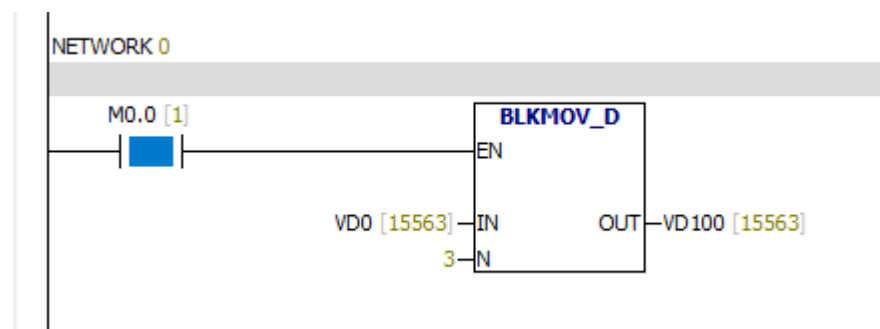
The range of N is from 1 to 255.

Error conditions:

0006 Indirect address

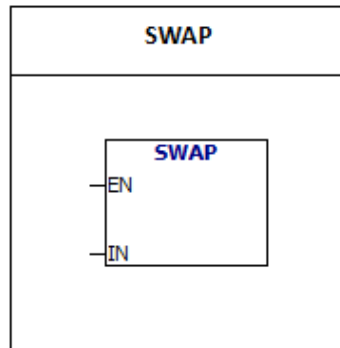
0091 Operating number is out of range

Example:



6.11.8 SWAP

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN | VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *AC, *LD | word |

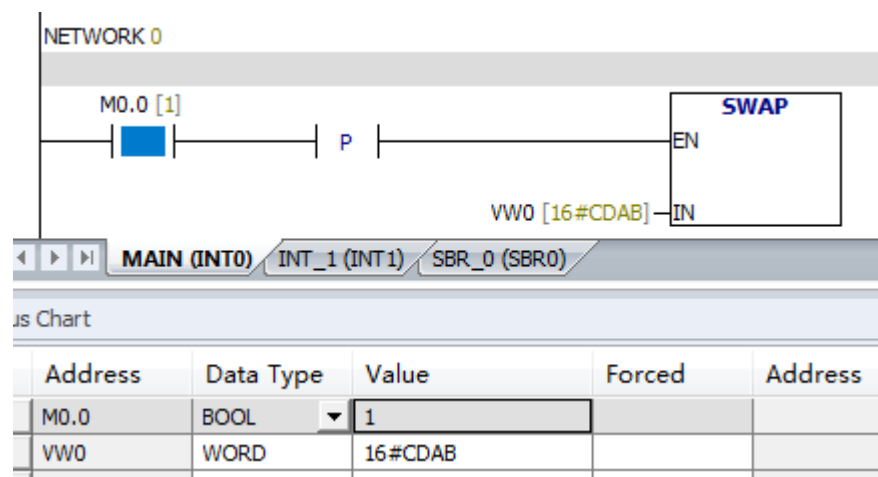
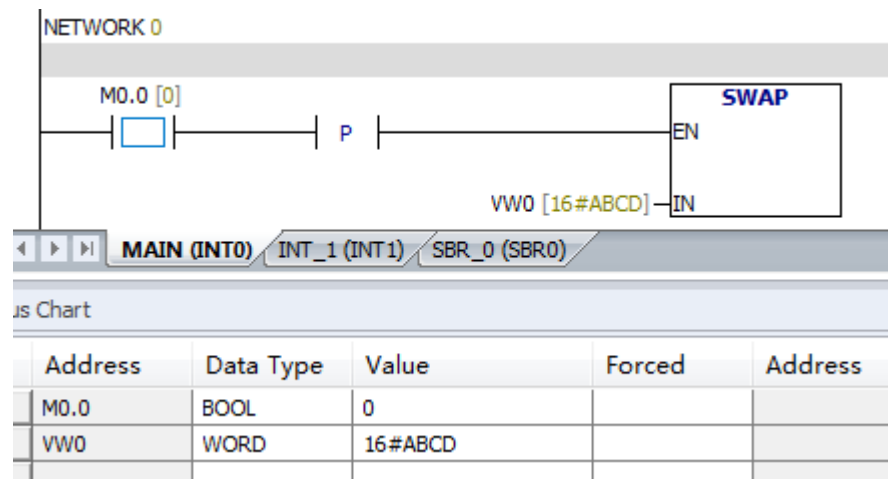


SWAP: The instruction interchanges high byte and low byte of the input word.

Error conditions:

0006 Indirect address

Example:

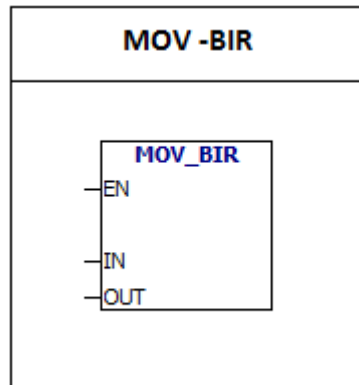


6.11.9 MOV -BIR

| Input/output | Operand | Data type |
|--------------|---------|-----------|
|--------------|---------|-----------|

| | | |
|----|-------------------|------|
| IN | IB, *VD, *LD, *AC | Byte |
|----|-------------------|------|

| | | |
|-----|--|------|
| OUT | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD | byte |
|-----|--|------|



MOV -BIR: Instruction reads the actual input value(byte), then writes the value to OUT. The process image register is not updated.

Error conditions:

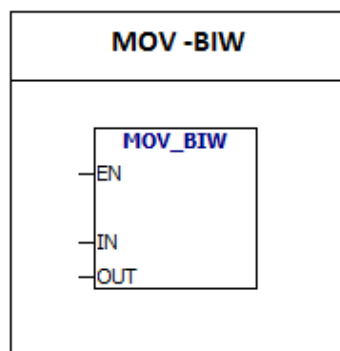
0006 Indirect address

6.11.10 MOV -BIW

| Input/output | Operand | Data type |
|--------------|---------|-----------|
|--------------|---------|-----------|

| | | |
|----|--|------|
| IN | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *AC, *LD | byte |
|----|--|------|

| | | |
|-----|-------------------|------|
| OUT | QB, *VD, *LD, *AC | byte |
|-----|-------------------|------|

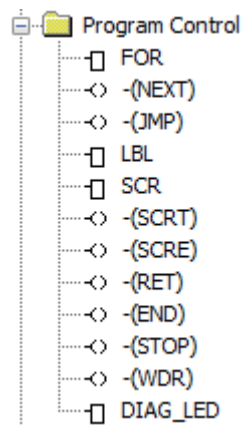


MOV -BIW: The instruction writes the input value (IN) to the actual input (OUT) and update the corresponding process image register.

Error conditions:

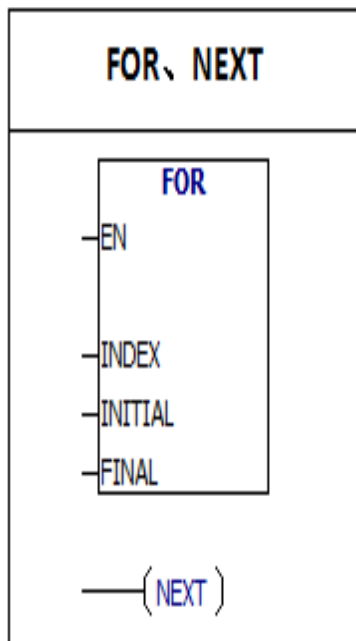
0006 Indirect address

6.12 Program control



6.12.1 FOR、NEXT

| Input/output | Operand | Data type |
|--------------|---|-----------|
| INDX | VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC | Integer |
| INIT | VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, constant, *VD, *LD, *AC | Integer |
| FINAL | VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AIW, constant, *VD, *LD, *AC | Integer |

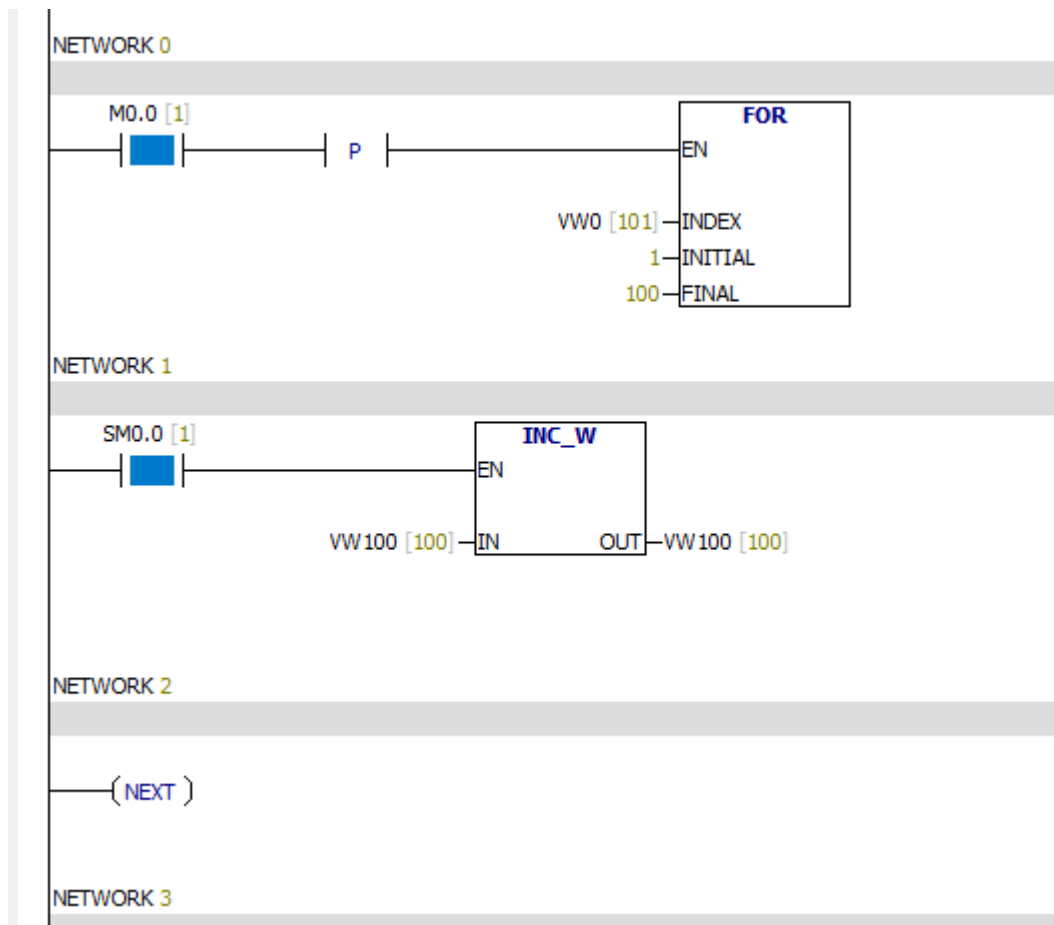


FOR instruction executes instructions between FOR and NEXT. You have to specify the current cycle count (INDX), start value (INIT), and end value (FINAL). NEXT (NEXT) instruction marks the end of the FOR loop, and the top value of the stack is set to 1. Use FOR/NEXT to set the number of loops. Each FOR instruction requires a NEXT instruction. FOR/NEXT loops can be nested with 8 FOR/NEXT loops. After each execution of the FOR and NEXT instructions, the INDX value is increased, and the result is compared with the end value. If the INDX is greater than the end value, the loop terminates.

Error condition:

0006 indirect address

Example:



Notes:

Cycle times are set to 100 times. At the end of the cycle, the value of VW100 is 100.

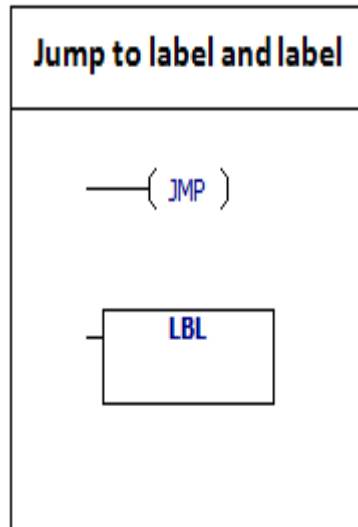
6.12.2 Jump to label and label

Input/output

Data type

n:constant (0~255)

word



JMP instruction performs the branch operation to the program in the specified tag (n). When the jump is accepted, the top value of the stack is 1.

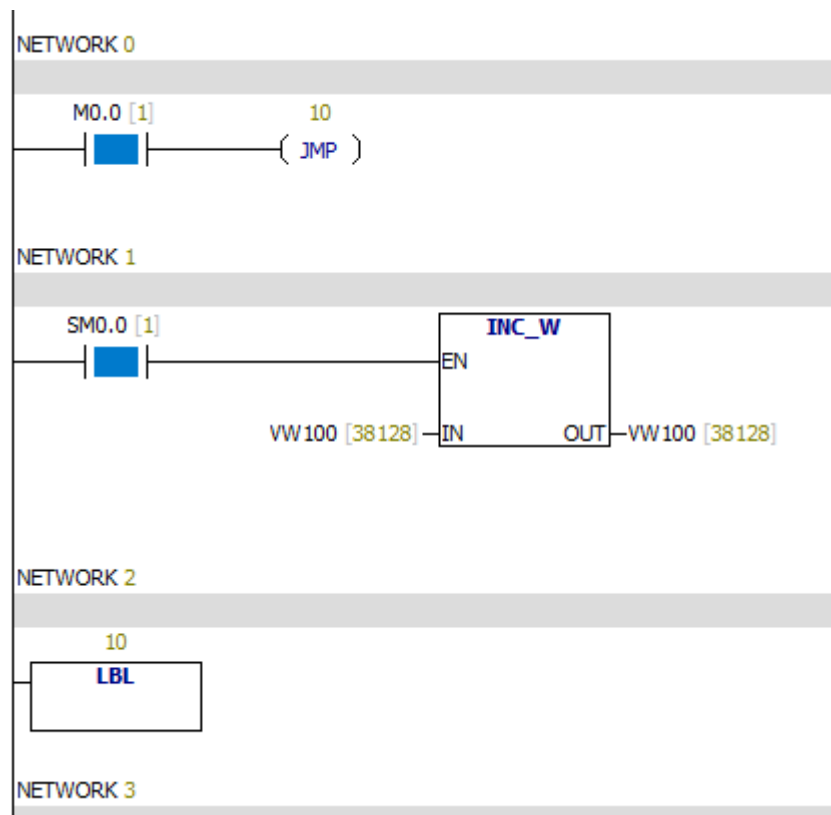
LBL instruction signs the location of n.

You can use the "jump" instruction in the main program, subroutine, or interrupt routine.

You can't jump from the main program to a subroutine or an interrupt routine. You can use the "jump" instruction in the SCR segment, but the corresponding

"label" instruction must be located within the same SCR segment.

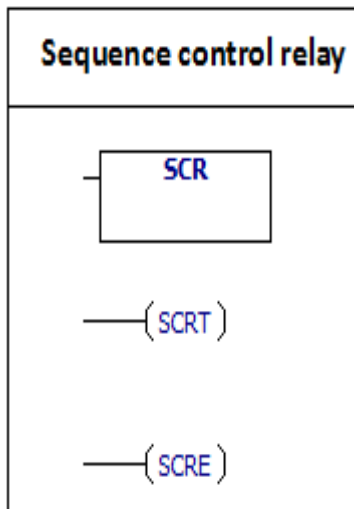
Example:



When the M0.0 bit is 1, the value of VW100 is no longer increased.

6.12.3 Sequence control relay

| Input/output | Operand | Data type |
|--------------|---------|-----------|
| n | S | Boolean |



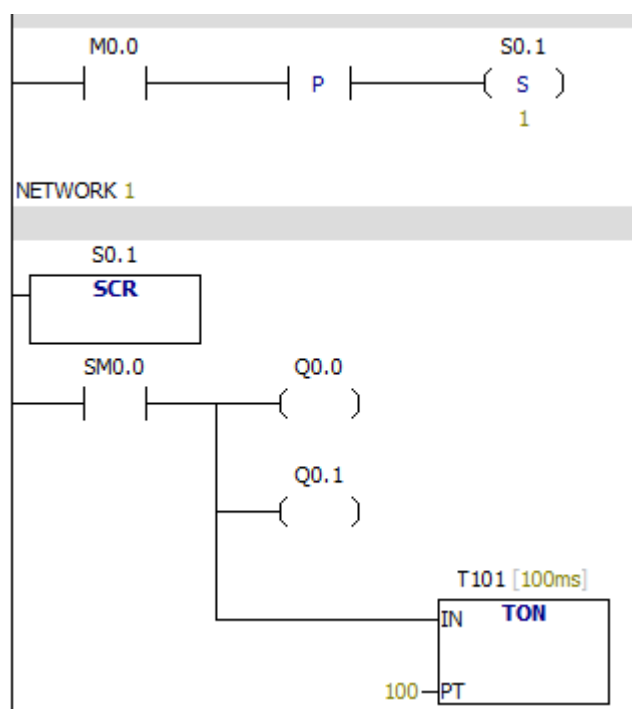
SCR instruction is good at dealing with repetitive operations.

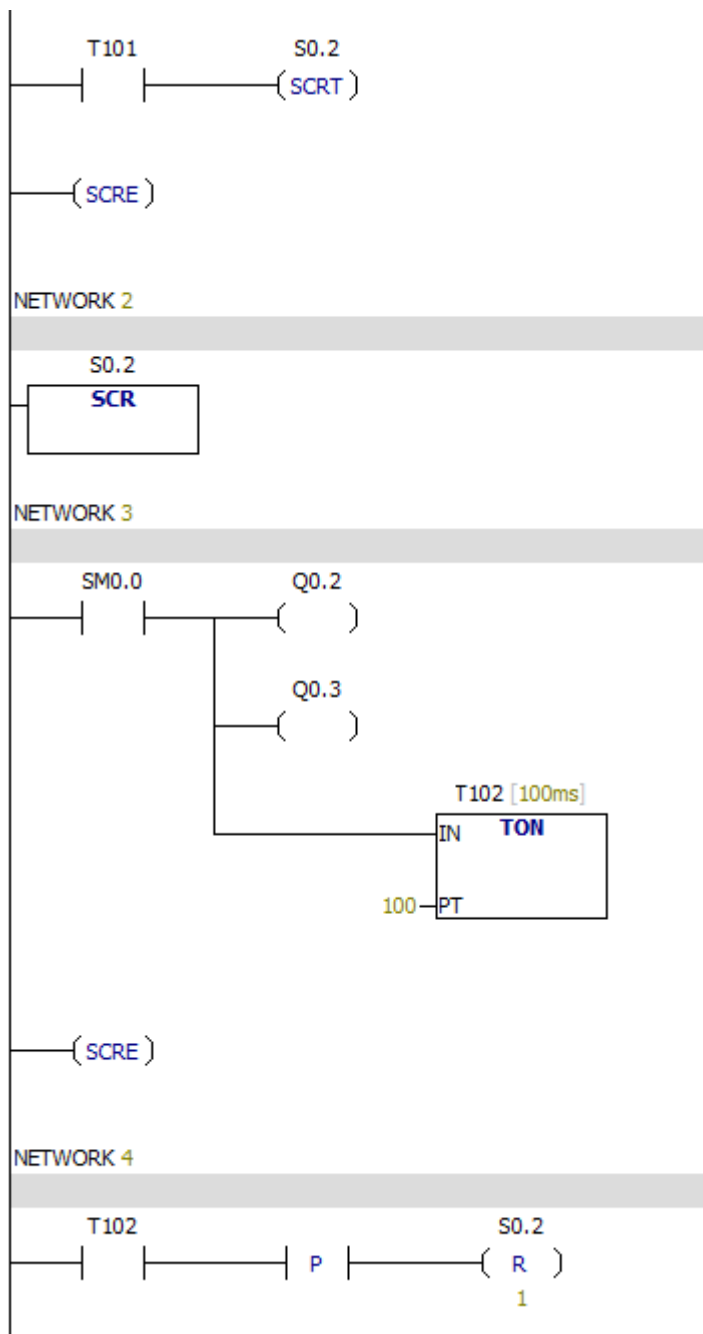
SCR: load the SCR section, you can use the SET instruction.

SCRT: Jump to another SCR segment and close the current SCR segment.

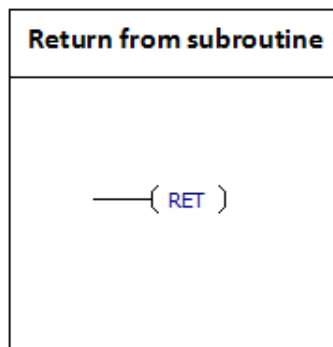
SCRE: The instruction signs the end of SCR segment.

Example:





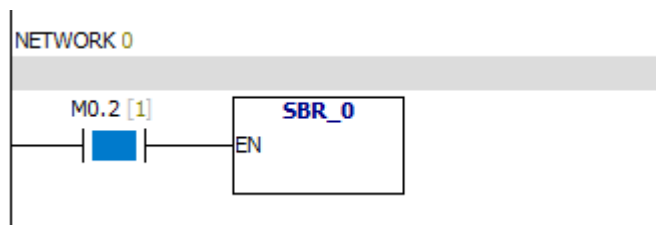
6.12.4 Return from subroutine



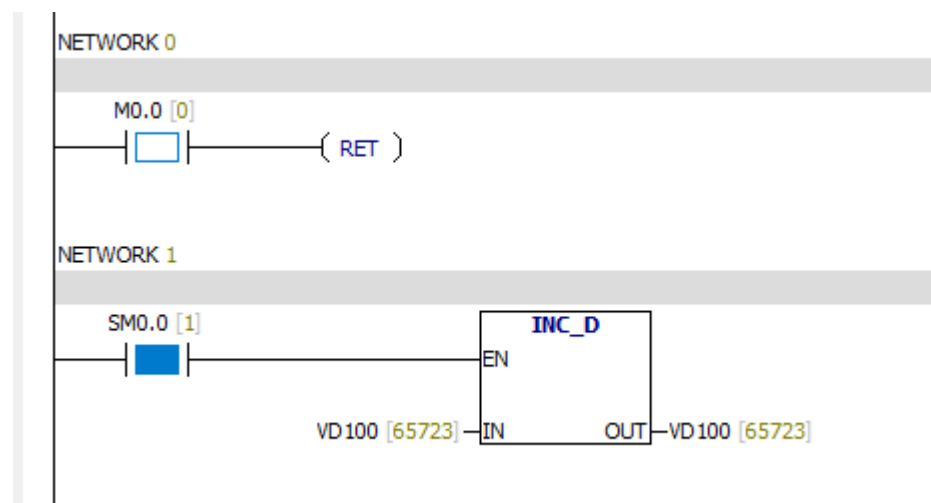
RET: Return from the subroutine to the main program.

Example:

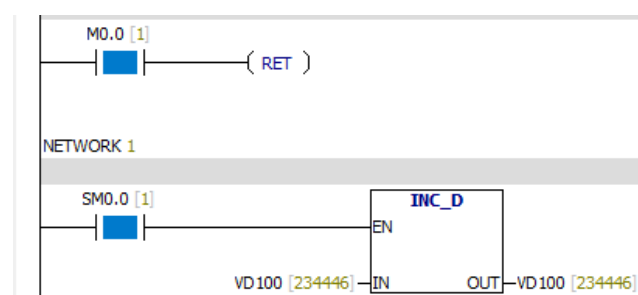
Main program:



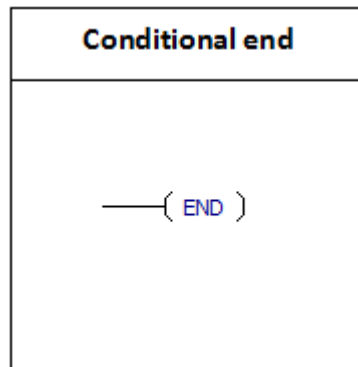
Subroutine:



When the M0.0 bit is 1, return from the subroutine, the following program will no longer be scanned.



6.12.5 Conditional end

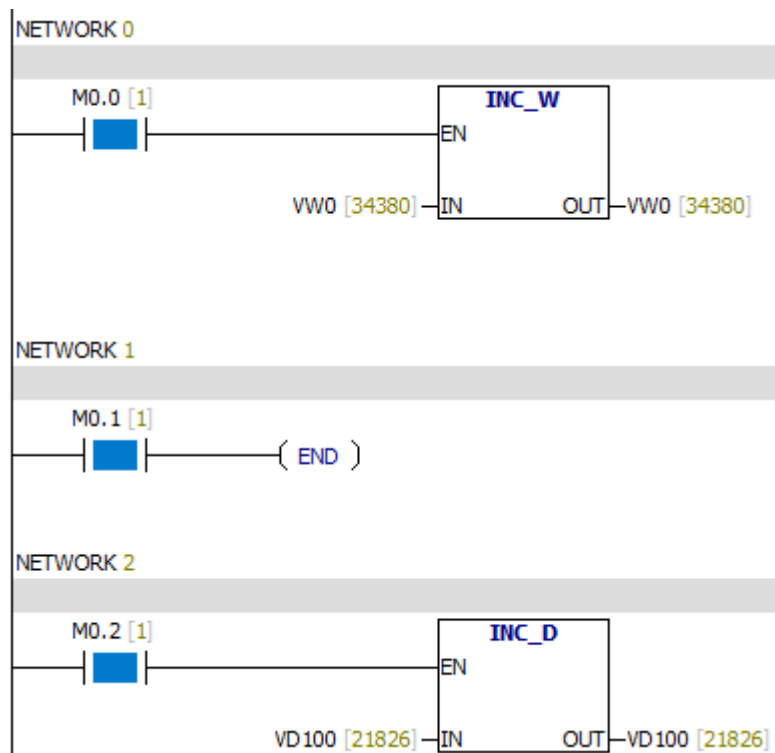


The **END** instruction terminates the user program.

Notes:

You can use the "END" instruction in the main program, but can't be used in subroutine or interrupt routine.

Example:

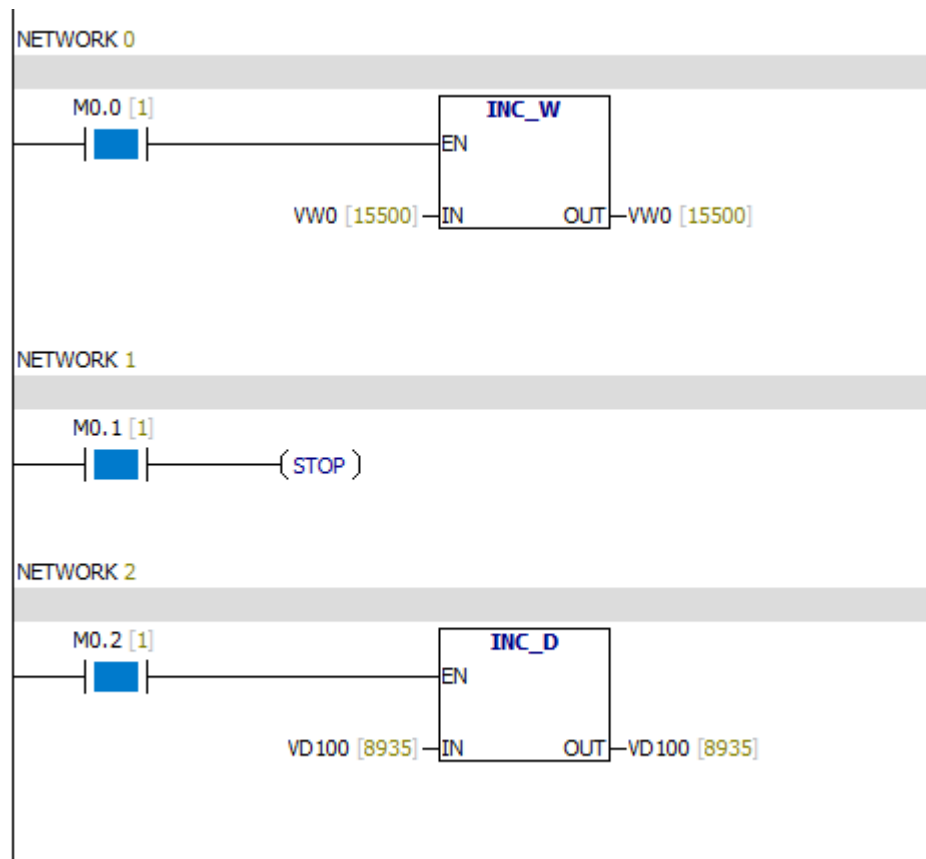


When the M0.1 bit is 1, the program will not be scanned.

6.12.6 STOP

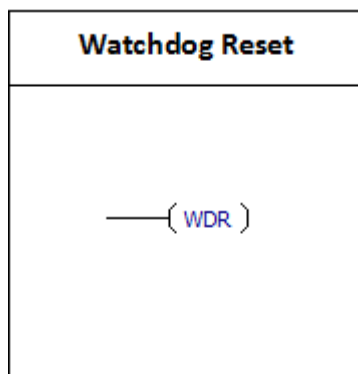
STOP instruction: STOP

Example:



When the M0.1 bit is 1, PLC converts to the STOP mode, all the programs stop running.

6.12.7 Watchdog Reset



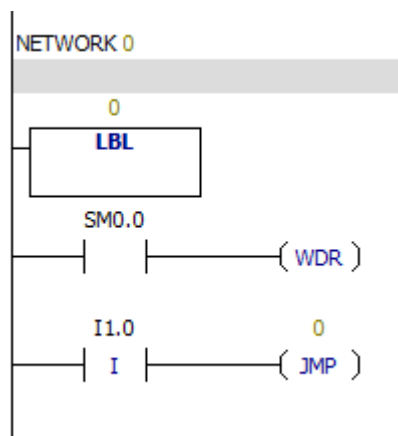
WDR clear watchdog time. When the scan cycle is greater than the watchdog time, the WDR makes the watchdog not issue a warning.

Using “WDR” instruction should be careful. The following programs can be performed after the scan cycle is completed.

- 1.Communication
- 2.I/O update (except for immediate I/O)
- 3.Forced update
- 4.SM bits update
- 5.Run time diagnostic program
- 6.STOP (stop) instruction for interrupt routine

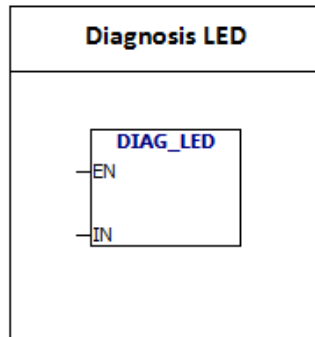
Attention: If you expect scan time will be more than 500 ms, you should use the WDR instruction to re trigger the watchdog timer.

Example:



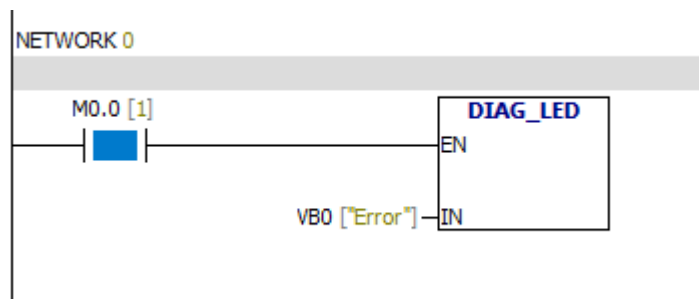
6.12.8 Diagnosis LED

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | String |



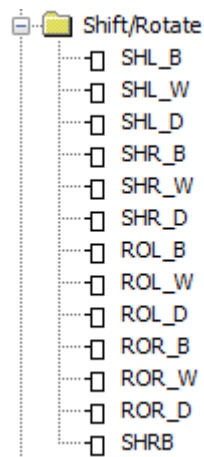
If the value of "EN" is 1, then the LCD will display the string from "IN".

Example:



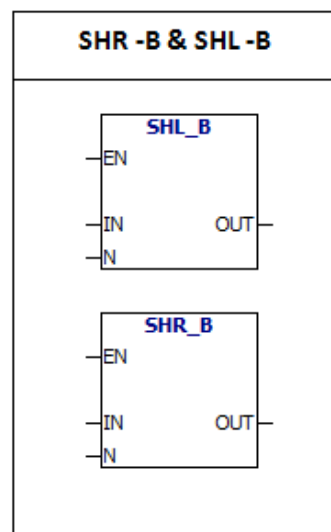
When the value of M0.0 is equal to 1, the LCD will display "Error".

6.13 Shift cycle



6.13.1 SHR -B & SHL -B

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | Byte |
| N | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | Byte |
| OUT | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC | Byte |



SHR -B: Input byte "IN" and move N bits towards the right.

Then place the results in OUT.

SHL -B: Input byte "IN" and move N bits towards the left.

Then place the results in OUT.

The moved-out bits are filled with zero. If N is greater than or equal to 8, you can move up to 8 bits.

SHR -B & SHL -B operations are not signed.

Error conditions:

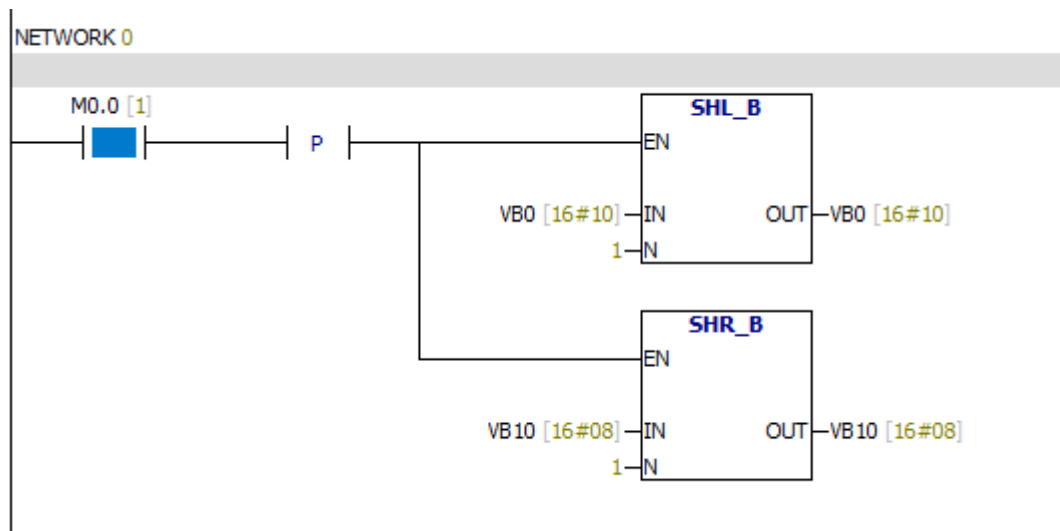
0006 Indirect address

Special memory bit:

SM1.0 Zero Result

SM1.1 Overflow

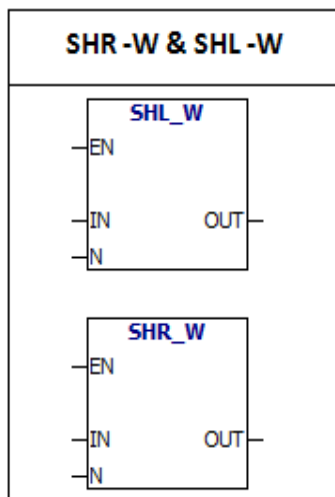
Example:



When the value of M0.0 is 1, VB0 moves a bit towards the left and VB10 moves a bit towards the right.

6.13.2 SHR -W & SHL -W

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN | VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, constant, *VD, *LD, *AC | word |
| N | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | byte |
| OUT | VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC | word |



SHR -W: Input word "IN" and move N bits towards the right. Then place the results in OUT.

SHL -W: Input word "IN" and move N bits towards the left. Then place the results in OUT.

The moved-out bits are filled with zero. If N is greater than or equal to 16, you can move up to 16 bits.

SHR -W & SHL -W operations are signed. Symbol bit can be moved.

Error conditions:

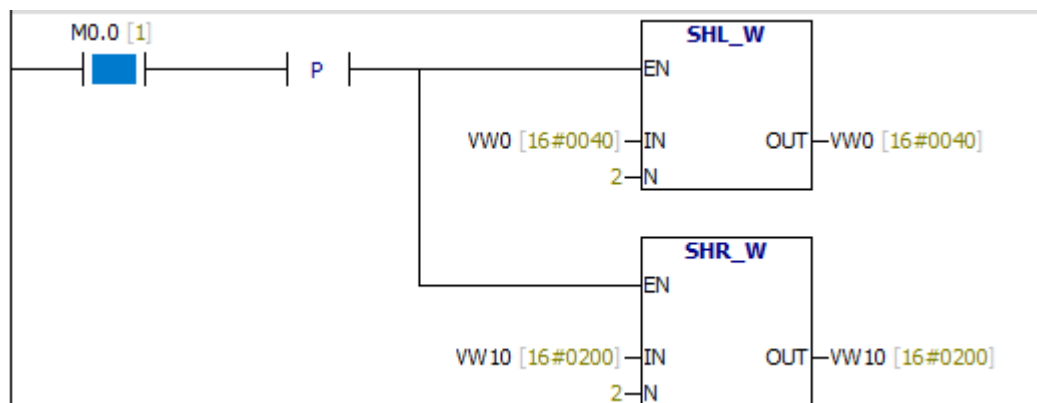
0006 Indirect address

Special memory bit:

SM1.0 Zero Result

SM1.1 Overflow

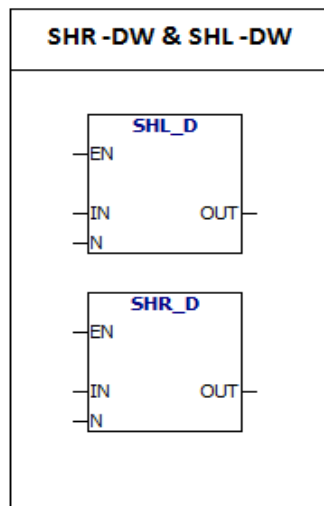
Example:



When the value of M0.0 is 1, VW0 moves two bits towards the left and VW10 moves two bits towards the right.

6.13.3 SHR -DW & SHL -DW

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, AC, HC, constant, *VD, *LD, *AC | Double word |
| N | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | Double word |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Double word |



SHR -DW: Input double word "IN" and move N bits towards the right. Then place the results in OUT.

SHL -DW: Input double word "IN" and move N bits towards the left. Then place the results in OUT.

The moved-out bits are filled with zero. If N is greater than or equal to 32, you can move up to 32 bits.

SHR -DW & SHL -DW operations are signed. Symbol bit can be moved.

Error conditions:

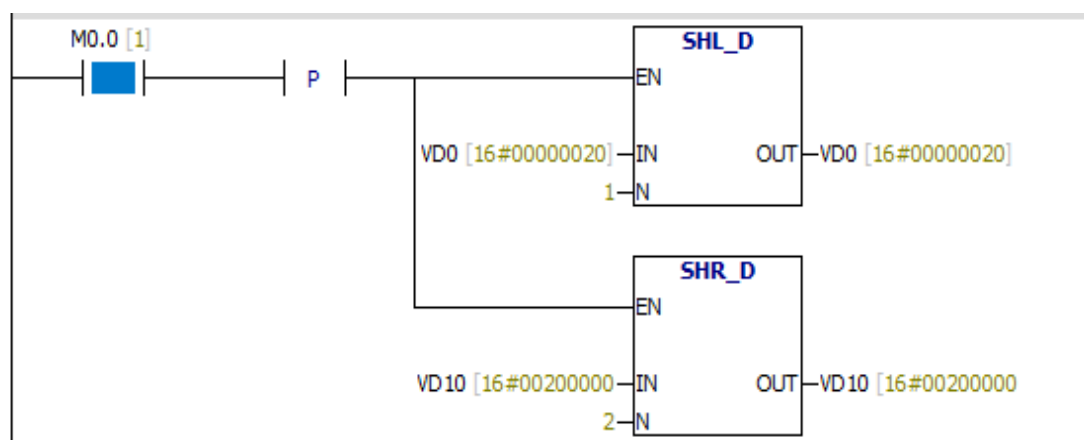
0006 Indirect address

Special memory bit:

SM1.0 Zero Result

SM1.1 Overflow

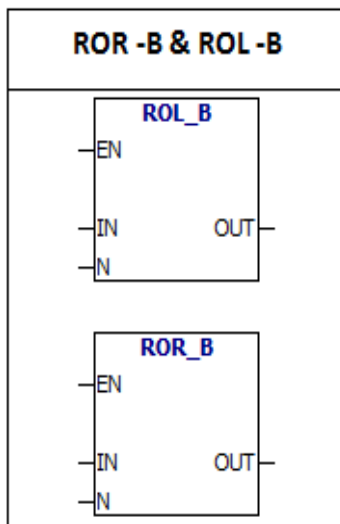
Example:



When the value of M0.0 is 1, VD0 moves a bit towards the left and VD10 moves two bits towards the right.

6.13.4 ROR -B & ROL -B

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN | VB, IB, QB, MB, SMB, SB, LB, AC, constant, *VD, *LD, *AC | Byte |
| N | VB, IB, QB, MB, SMB, SB, LB, AC, constant, *VD, *LD, *AC | Byte |
| OUT | VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC | Byte |



ROR -B & ROL -B: Instruction rotates the input byte to the right or to the left n bits and puts the result in the output byte (OUT). Rotation is cyclic.

If N is greater than or equal to 8, the remainder of N/8 is the number of rotation bits. If remainder is equal to 0, Rotation operation is not performed and the value of SM1.0 is 1. If the rotation operation is performed, the final rotation bit is copied to overflow bit (SM1.1).

ROR -B & ROL -B operations are not signed.

Error conditions:

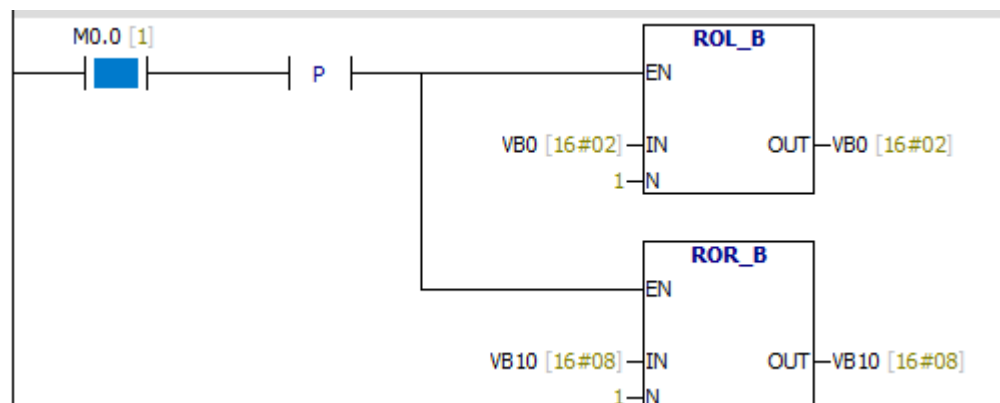
0006 Indirect address

Special memory bit:

SM1.0 When the value of the loop is zero, SM1.0 is set to 1.

SM1.1 Overflow bit

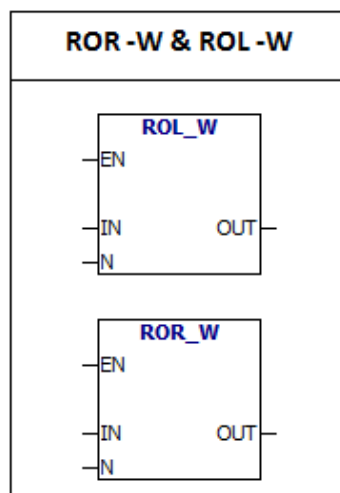
Example:



When the value of M0.0 is 1, VB0 moves a bit towards the left and VB10 moves a bit towards the right circularly.

6.13.5 ROR -W & ROL -W

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN | VW, T, C, IW, QW, MW, SW, SMW, LW, AC, AIW, constant, *VD, *LD, *AC | word |
| N | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | byte |
| OUT | VW, T, C, IW, QW, MW, SW, SMW, LW, AC, *VD, *LD, *AC | word |



ROR -W & ROL -W: Instruction rotates the input word to the right or to the left n bits and puts the result in the output word (OUT). Rotation is cyclic.

If N is greater than or equal to 16, the remainder of N/16 is the number of rotation bits. If remainder is equal to 0, Rotation operation is not performed and the value of SM1.0 is 1. If the rotation operation is performed, the final rotation bit is copied to overflow bit (SM1.1).

ROR -W & ROL -W operations are not signed.

Error conditions:

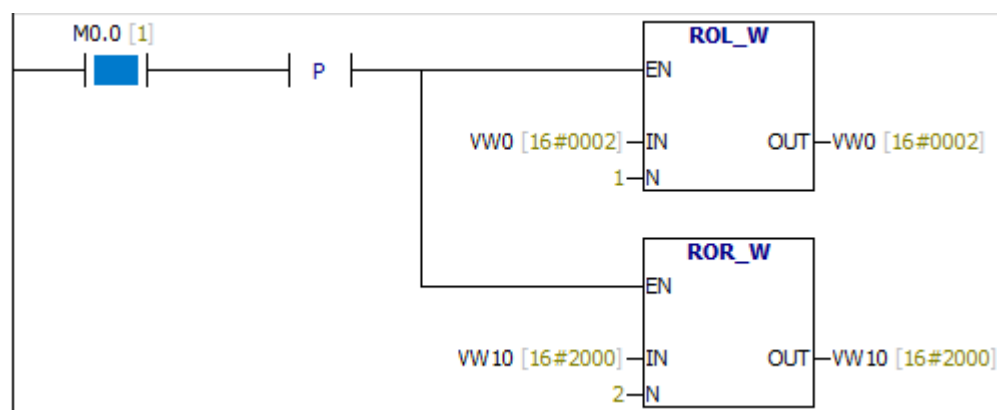
0006 Indirect address

Special memory bit:

SM1.0 When the value of the loop is zero, SM1.0 is set to 1.

SM1.1 Overflow bit

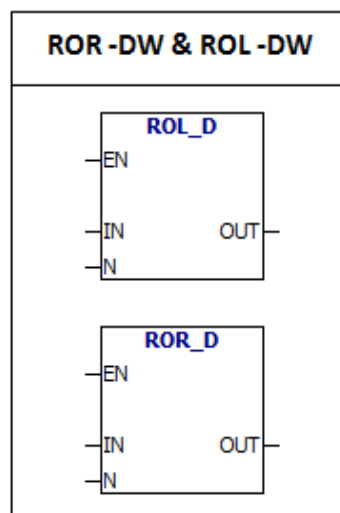
Example:



When the value of M0.0 is 1, VW0 moves a bit towards the left and VW10 moves two bits towards the right circularly.

6.13.6 ROR -DW & ROL -DW

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SD, SMD, LD, AC, HC, constant, *VD, *LD, *AC | Double word |
| N | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | Byte |
| OUT | VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC | Double word |



ROR -DW & ROL -DW: Instruction rotates the input double word to the right or to the left n bits and puts the result in the output double word (OUT). Rotation is cyclic. If N is greater than or equal to 32, the remainder of N/32 is the number of rotation bits. If remainder is equal to 0, Rotation operation is not performed and the value of SM1.0 is 1. If the rotation operation is performed, the final rotation bit is copied to overflow bit (SM1.1). ROR -DW& ROL -DW operations are not signed.

Error conditions:

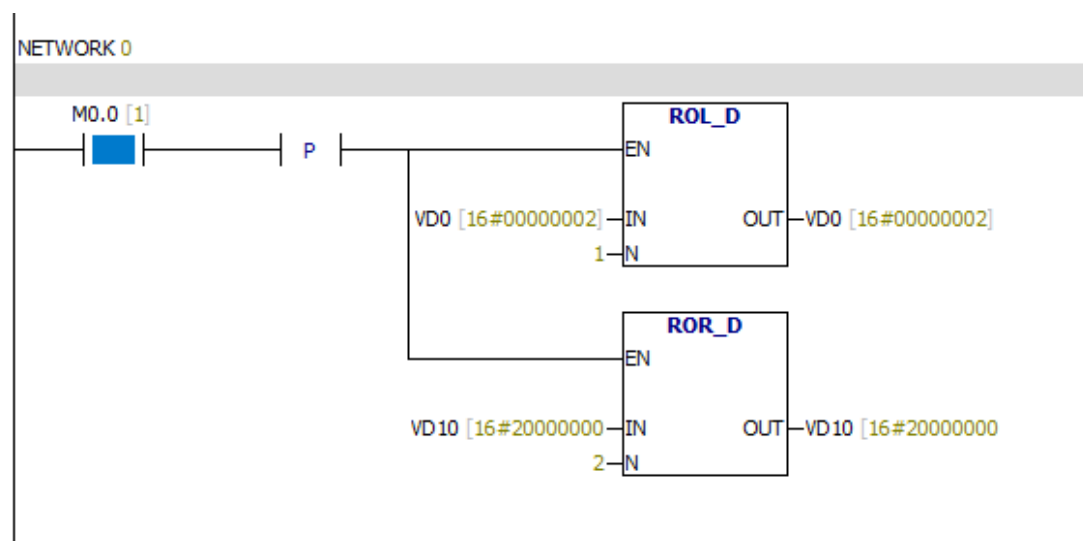
0006 Indirect address

Special memory bit:

SM1.0 When the value of the loop is zero, SM1.0 is set to 1.

SM1.1 Overflow bit

Example:



6.13.7 SHRB

Input/output Operand

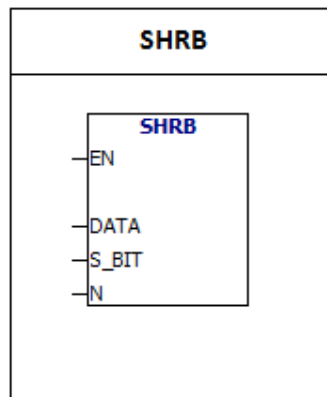
Data type

DATA, S_BIT I, Q, M, SM, T, C, V, S, L

Boolean

N VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC

Boolean



SHRB instruction moves the DATA value to the shift register. S_BIT specifies the lowest bit of the shift register. N specifies the length of the shift register and the shift direction (shift plus = N, shift minus = -N). The moved-out bit is placed in the overflow memory bit (SM1.1). The instruction is defined by S_BIT and N.

Error conditions:

0006 Indirect address

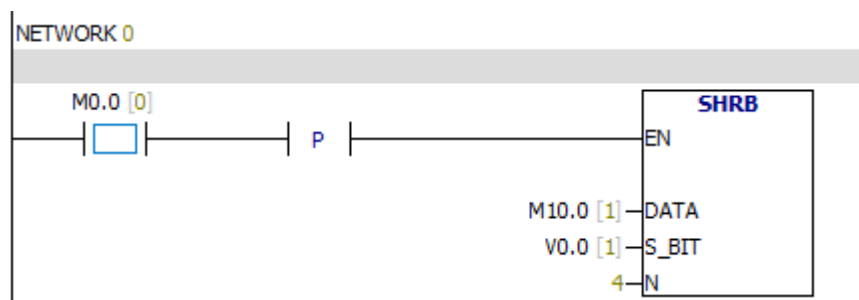
0091 Operating number is out of range

0092 Count field error

Special memory bit:

SM1.1 Overflow bit

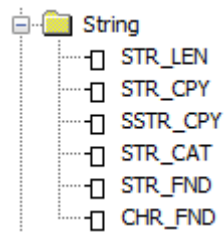
Example:



When the value of M0.0 is 1, the value of M10.0 is moved to V0.0, the value of V0.0 is moved to V0.1, the value of V0.1 is moved to V0.2, the value of V0.2 is moved to V0.3, the value of V0.3 is moved to SM1.1.

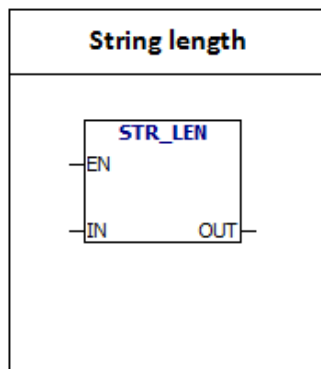
If N is negative, the shift direction is opposite.

6.14 Character string



6.14.1 String length

| Input/output | Operand | Data type |
|--------------|--|------------------|
| IN | VB, Constant string, LB, *VD, *LD, *AC | Character string |
| OUT | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC | Byte |



STR-LEN: Instruction output "IN" string length.

The longest constant string is 126 bytes.

Error conditions:

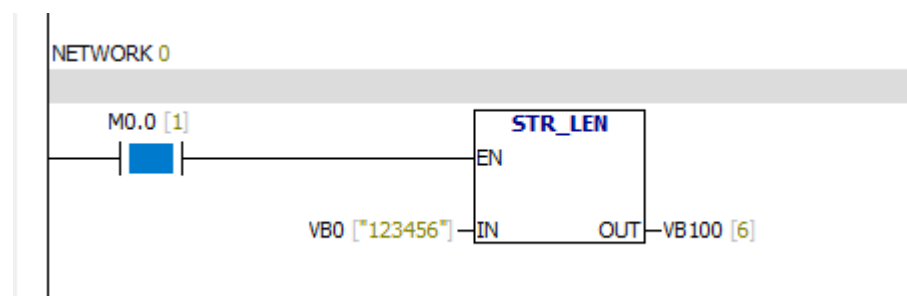
0006 Indirect address

0091 Operand range

ASCII constant string data type format:

String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

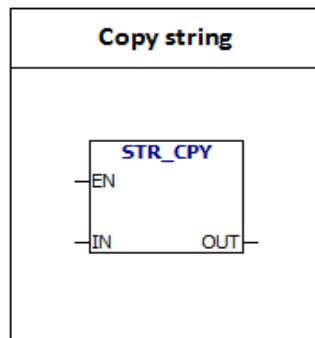
Example:



When the string is "123456", the length of the string is 6.

6.14.2 Copy string

| Input/output | Operand | Data type |
|--------------|--|------------------|
| IN | VB, Constant string, LB, *VD, *LD, *AC | Character string |
| OUT | VB, *VD, LB, *LD, *AC | Character string |



STR-CPY: Instruction copies the “IN” string to the “OUT” string.

The longest constant string is 126 bytes.

Error conditions:

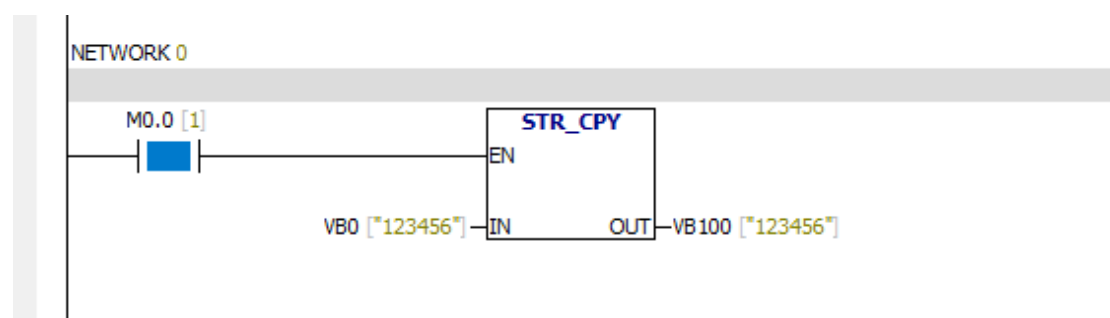
0006 Indirect address

0091 Operand range

ASCII constant string data type format:

String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

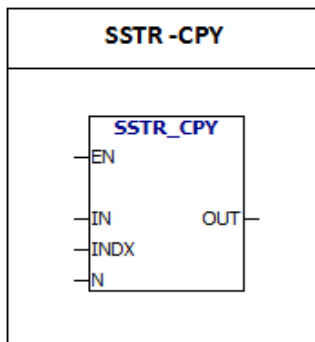
Example:



When M0.0 is 1, string starting with VB0 is copied to the string starting with VB100. VB100 storage is an integer 6, VB101 storage is character "1", VB102 storage is character "2", VB103 storage is "3", VB104 storage is "4", VB105 storage is "5", VB106 storage is "6".

6.14.3 SSTR-CPY

| Input/output | Operand | Data type |
|--------------|--|-----------|
| Iput | VB, Constant string, LB, *VD, *LD, *AC | string |
| INDX, N | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | byte |
| OUT | VB, *VD, LB, *LD, *AC | string |



SSTR-CPY: Copy a portion of the input string to the OUT string. If the value of INDX is X, copy the string starting from the xth character. The length of the copy string is N. The longest constant string is 126 bytes.

Error conditions:

0006 Indirect address

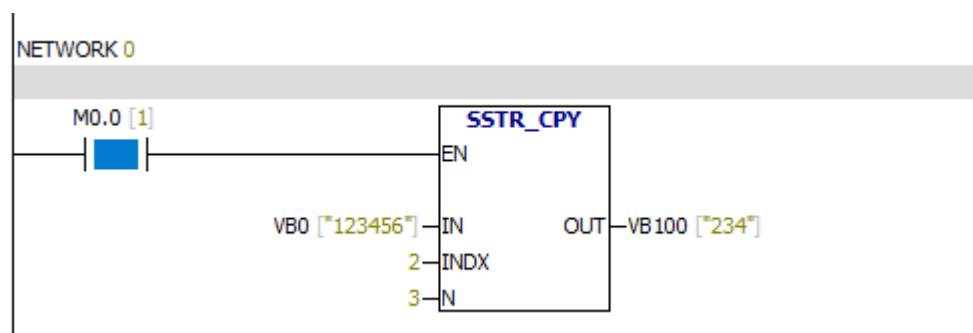
0091 Operand range

009B Illegal index

ASCII constant string data type format:

String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

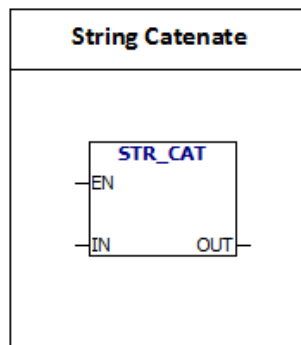
Example:



Copy VB0 string. Copy the string starting from the second character. The length of the copy string is 3. The result is placed VB100.

6.14.4 String concatenate

| Input/output Operand | | Data type |
|----------------------|--|-----------|
| Input | VB, Constant string, LB, *VD, *LD, *AC | String |
| OUT | VB, LB, *VD, *LD, *AC | String |



STR -CAT: Add the string specified by the IN to the string specified by the OUT.

The longest constant string is 126 bytes.

Error conditions:

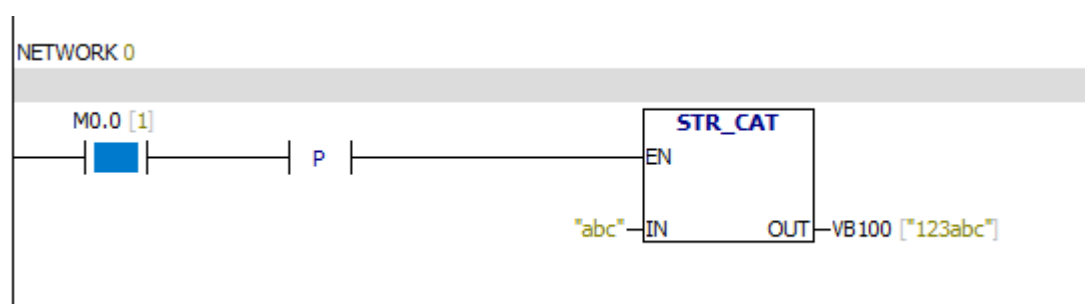
0006 Indirect address

0091 Operand range

ASCII constant string data type format:

String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

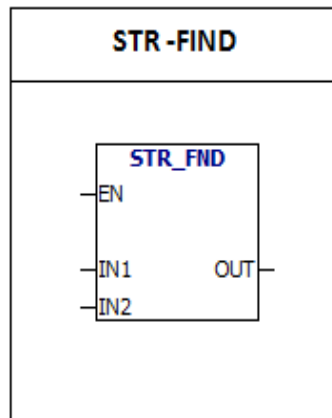
Example:



VB100 string is "123". After using the STR -CAT instruction, the VB100 string is "123abc".

6.14.5 STR -FIND

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN1, IN2 | VB, constant string, LB, *VD, *LD, *AC | string |
| OUT | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC | byte |



STR -FIND: The instruction searches for the string IN2 in the string IN1. Search starts from the OUT-start position. If you find a string that is the same as the string IN2, the first character position of the string is written to the OUT. If you do not find IN2 in IN1, OUT is set to 0. The longest length of a single constant string is 126 bytes. The longest comprehensive length of two constant strings is 240 bytes.

bytes.

Error conditions:

0006 Indirect address

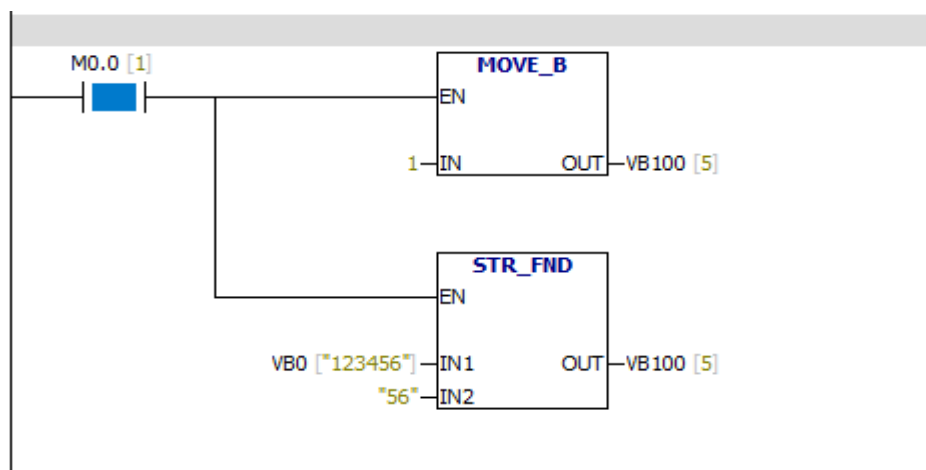
0091 Operand range

009B Illegal index

ASCII constant string data type format:

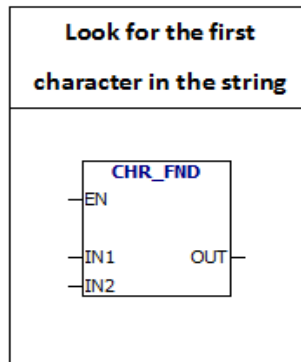
String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

Example:



6.14.6 Look for the first character in the string

| Input/output | Operand | Data type |
|--------------|--|-----------|
| IN1, IN2 | VB, constant string, LB, *VD, *LD, *AC | string |
| OUT | VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC | byte |



CHR -FIND: The instruction searches for the same character as the string IN2 in the string IN1. Search starts from the OUT-start position. If a match character is found, the character position is written to OUT. If a match character is not found, the OUT is set to 0.

The longest length of a single constant string is 126 bytes.

The longest comprehensive length of two constant strings is 240 bytes.

Error conditions:

0006 Indirect address

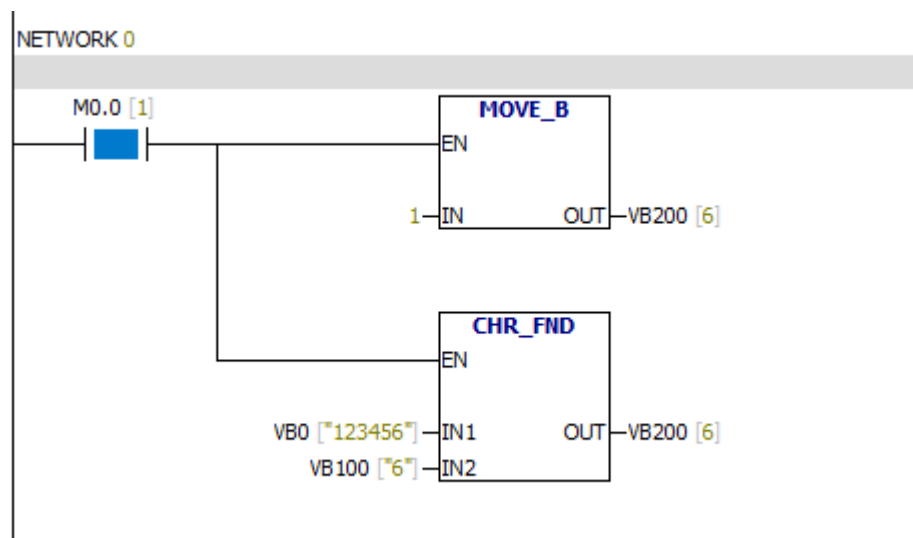
0091 Operand range

009B Illegal index

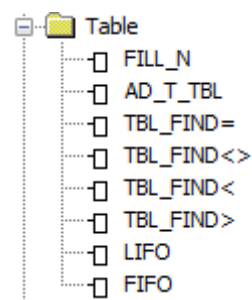
ASCII constant string data type format:

String is a series of characters; each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks ("string constant").

Example:

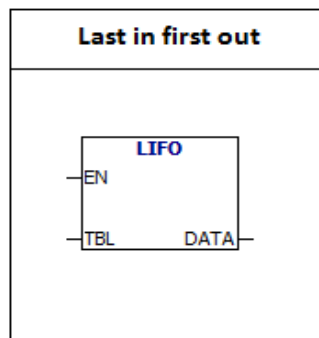


6.15 Table



6.15.1 Last in first out

| Input/output Operand | | Data type |
|----------------------|---|-----------|
| TBL | VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, *LD, *AC | word |
| DATA | VW, IW, QW, MW, SW, SMW, LW, AC, T, C, AQW, *VD, *LD, *AC | integer |



LIFO: Instruction moves the latest (or last) entry in the table to the output memory address. Remove the last entry in the table (TBL) and move the value to the location specified by DATA. Each time the instruction is executed, the number of entries in the table reduces 1.

Error conditions:

0006 Indirect address

0091 Operand range

SM1.5 Empty list

Special memory bit:

SM1.5 Empty list

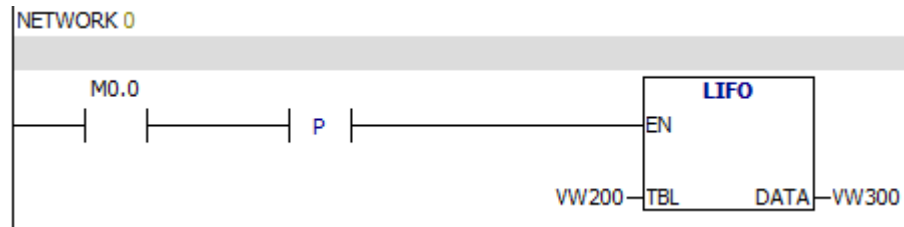
Table Format:

| | |
|-------|-------------------------------|
| VW200 | The maximum number of entries |
| VW202 | Entry count |
| W204 | Data 0 |
| VW206 | Data 1 |
| VW208 | Data 2 |

| | |
|-------|-------|
| | |
|-------|-------|

For example:

PLC program:

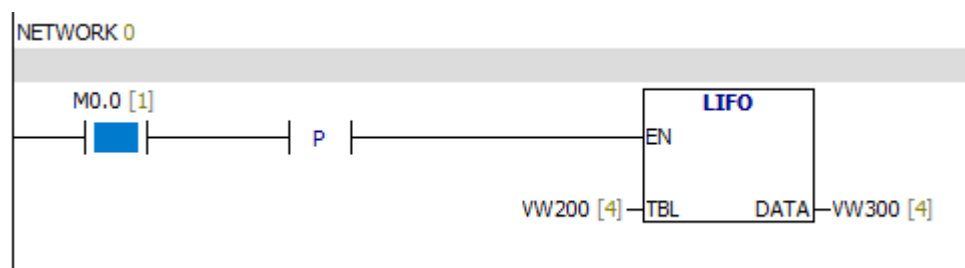


Data block:

| | Adress | Data Type | Val |
|---|--------|-----------|-----|
| ✓ | VW200 | INT | 4 |
| ✓ | VW202 | INT | 4 |
| ✓ | VW204 | INT | 1 |
| ✓ | VW206 | INT | 2 |
| ✓ | VW208 | INT | 3 |
| ✓ | VW210 | INT | 4 |

Analysis:

When the value of M0.0 is equal to 1, the last entry of the table will be deleted and the value of the last entry of the table will be moved to "VW300".



| | | |
|-------|-----|---|
| VW200 | INT | 4 |
| VW202 | INT | 3 |
| VW204 | INT | 1 |
| VW206 | INT | 2 |
| VW208 | INT | 3 |
| VW210 | INT | 4 |

When the value of M0.0 is equal to 1:

VW202=3

VW210 is invalid

VW300=4

6.15.2 FIFO

Input/output Operand

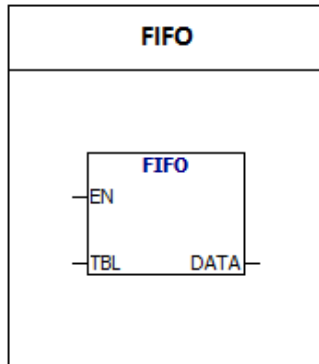
Data type

TBL VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, *LD, *AC

word

DATA VW, IW, QW, MW, SW, SMW, LW, AC, T, C, AQW, *VD, *LD, *AC

integer



FIFO: Remove the first entry in the table (TBL) and move the value to the location specified by DATA. All other entries in the table move a location upward. Each time the instruction is executed, the number of entries in the table reduces 1.

Error conditions:

0006 Indirect address

0091 Operand range

SM1.5 Empty list

Special memory bit:

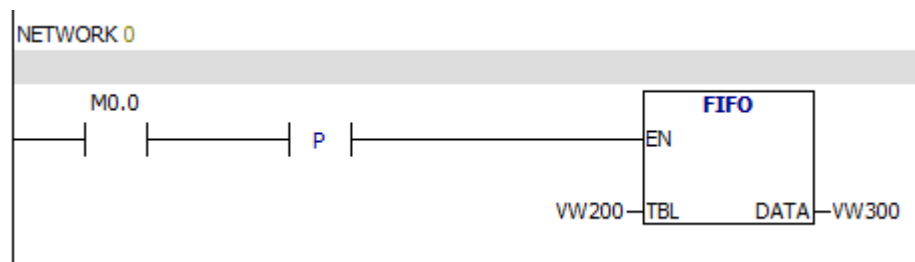
SM1.5 Empty list

Table Format:

| | |
|-------|-------------------------------|
| VW200 | The maximum number of entries |
| VW202 | Entry count |
| W204 | Data 0 |
| VW206 | Data 1 |
| VW208 | Data 2 |
| | |

For example:

PLC program:

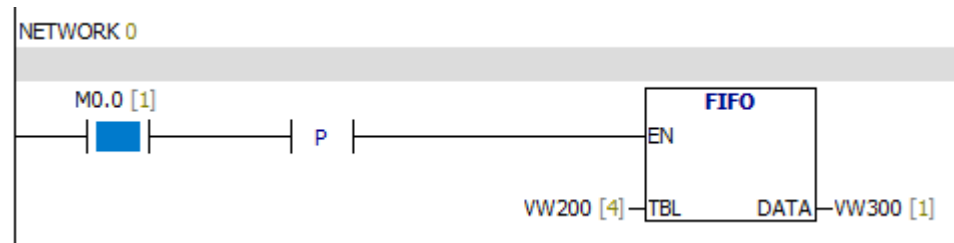


Data block:

| | Adress | Data Type | Value |
|---|--------|-----------|-------|
| ✓ | VW200 | INT | 4 |
| ✓ | VW202 | INT | 4 |
| ✓ | VW204 | INT | 1 |
| ✓ | VW206 | INT | 2 |
| ✓ | VW208 | INT | 3 |
| ✓ | VW210 | INT | 4 |

Analysis:

When the value of M0.0 is equal to 1, the first entry of the table will be deleted and the value of the first entry of the table will be moved to “VW300”.



| | | | |
|--|-------|-----|---|
| | VW200 | INT | 4 |
| | VW202 | INT | 3 |
| | VW204 | INT | 2 |
| | VW206 | INT | 3 |
| | VW208 | INT | 4 |
| | VW210 | INT | 4 |

When the value of M0.0 is equal to 1:

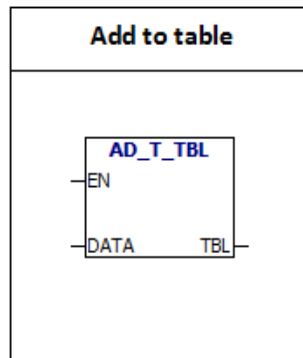
VW202=3

VW210 is invalid

VW300=1

6.15.3 Add to table

| Input/output | Operand | Data type |
|--------------|---|-----------|
| DATA | VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, constant, *VD, *LD, *AC | integer |
| TBL | VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, , *LD *AC | word |



AD -T- TBL: The instruction adds the word (DATA) to the table (TBL). The first value in the table is the maximum length of the table. The second value is the entry count (EC), it specifies the number of entries in the table. Each time you add new data to the table, the number of entries adds 1. Table can contain up to 100 entries, not including the first entry and the second entry.

entry and the second entry.

Error conditions:

0006 Indirect address

0091 Operand range

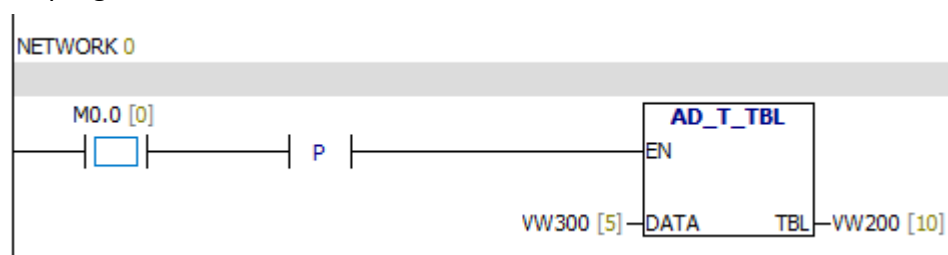
SM1.4 Table overflow

Special memory bit:

SM1.4 Table overflow

For example:

PLC program:



Data block:

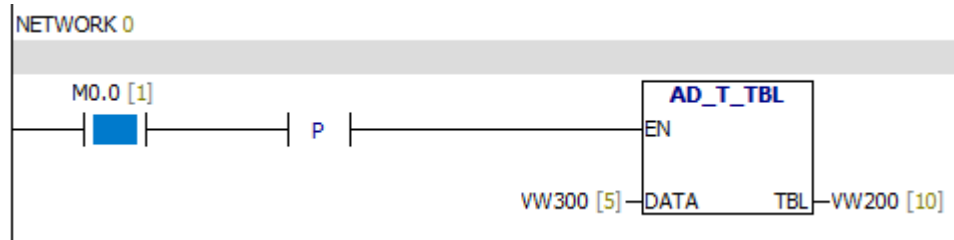
| | Address | Data Type | Value |
|---|---------|-----------|-------|
| ✓ | VW200 | INT | 10 |
| ✓ | VW202 | INT | 4 |
| ✓ | VW204 | INT | 1 |
| ✓ | VW206 | INT | 2 |
| ✓ | VW208 | INT | 3 |
| ✓ | VW210 | INT | 4 |

When the value of M0.0 is equal to 1:

The value of VW202 + 1

The Table will have a new entry

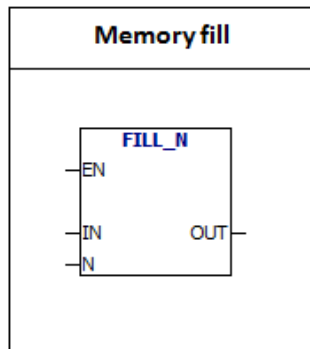
The value of the new entry is equal to the value of VW300.



| | | |
|-------|-----|----|
| VW200 | INT | 10 |
| VW202 | INT | 5 |
| VW204 | INT | 1 |
| VW206 | INT | 2 |
| VW208 | INT | 3 |
| VW210 | INT | 4 |
| VW212 | INT | 5 |

6.15.4 Memory fill

| Input/output | Operand | Data type |
|--------------|---|-----------|
| IN | VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, constant, *VD, *LD, *AC | integer |
| N | VB, IB, QB, MB, SB, SMB, LB, AC, constant, *VD, *LD, *AC | byte |
| OUT | VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, *VD, *LD, *AC | integer |



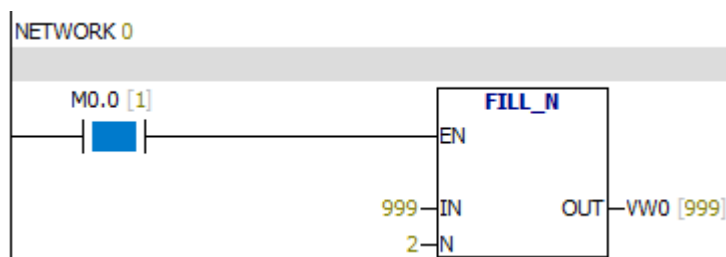
FILL-N: The input value of “IN” is written to the “OUT” N continuous words.

The range of N is from 1 to 255.

Error conditions:

- 0006 Indirect address
- 0091 Operand range

Example:



Status Chart

| | Address | Data Type | Value |
|--|---------|-----------|-------|
| | M0.0 | BOOL | 1 |
| | VW0 | INT | 999 |
| | VW2 | INT | 999 |

6.15.5 Table Find

| Input/output | Operand | Data type |
|--------------|---|-----------|
| TBL | VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, *LD, *AC | word |
| PTN | VW, IW, QW, MW, SW, SMW, AIW, LW, T, C, AC, constant, *VD, *LD, *AC | integer |
| INDX | VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC | word |

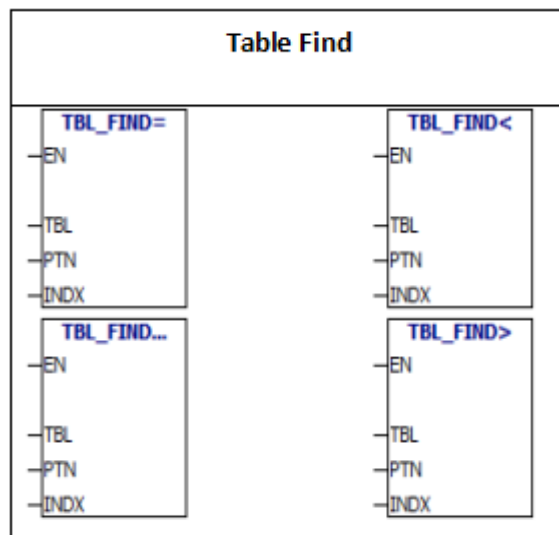
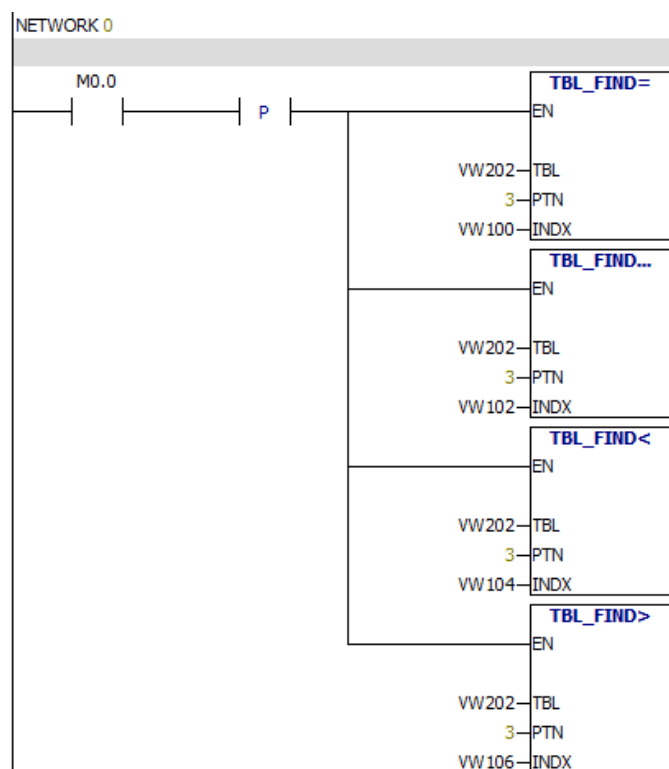


Table Find instruction: The instruction searches the same data as “PTN” in the table. “Table Find” starts from the entry specified by INDX. If a matching entry is found, the INDX points to the entry in the table. To find the next matching entry, you must add 1 to the INDX before using the “Table Find” instruction. If matching entry is not

found, the value of INDX is equal to the number of entries.

For example:

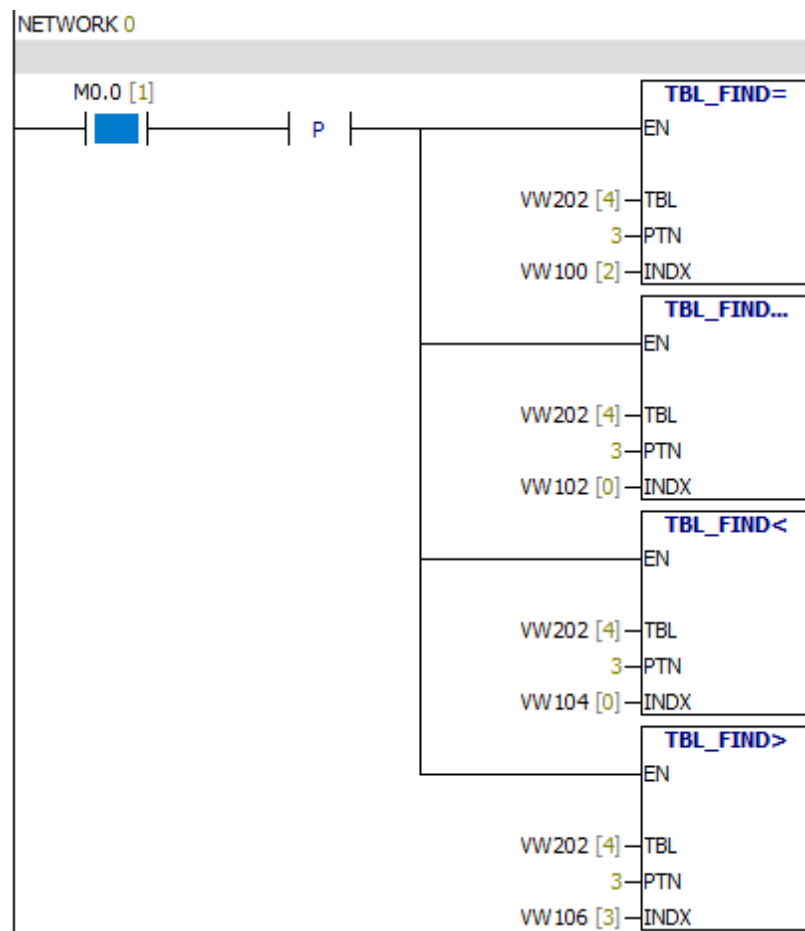
PLC program:



Data block:

| | Adress | Data Type | Value |
|---|--------|-----------|-------|
| ✓ | VW200 | INT | 10 |
| ✓ | VW202 | INT | 4 |
| ✓ | VW204 | INT | 1 |
| ✓ | VW206 | INT | 2 |
| ✓ | VW208 | INT | 3 |
| ✓ | VW210 | INT | 4 |

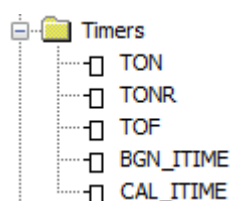
When the value of M0.0 is equal to 1:



The table format of the “Table-Find” begins with the entry count. It doesn’t have the “maximum number of entries”:

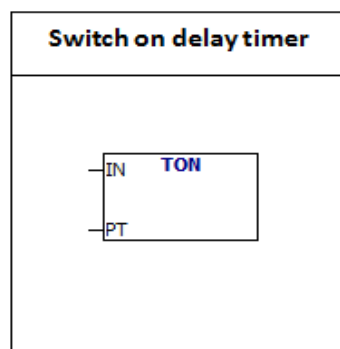
| | |
|-------|-------------|
| VW202 | Entry Count |
| VW204 | Data 0 |
| VW206 | Data 1 |
| VW208 | Data 2 |
| VW210 | Data 3 |

6.16 Timer



6.16.1 Switch on delay timer

| Input/output | Operand | Data type |
|--------------|---|-----------|
| Txxx | constant(T0 -T255) | word |
| IN | Enable bit | Boolean |
| PT | VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, constant, *VD, *LD, *AC | integer |



TON: When the value of the input “IN” is equal to 1, timer starts time. Timer current value of Txxx is the current time (a multiple of the time base). When the current value of the timer is equal to the present time (PT), the value of the timer bit is 1. When the value of the input “IN” is equal to 0, timer current value is cleared.

TON, TONR and TOF timers have three kinds of resolutions. Each current value is a multiple of the time base. For example, the number 50 in the 10-millisecond timer is 500 milliseconds.

Timer range:

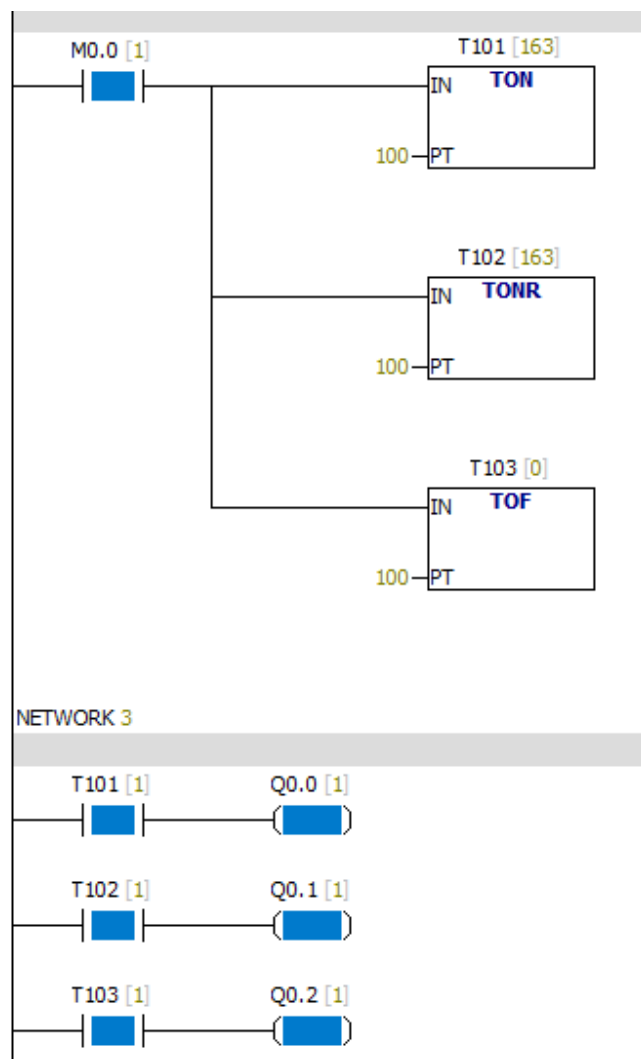
| Timer number | Time base (ms) | Time range (s) |
|--------------|----------------|----------------|
| T0 | 1 | 65.535 |
| T1~T4 | 10 | 655.35 |
| T5~T31 | 100 | 6553.5 |
| T32 | 1 | 65.535 |
| T33~T36 | 10 | 655.35 |
| T37~T63 | 100 | 6553.5 |
| T64 | 1 | 65.535 |
| T65~T68 | 10 | 655.35 |
| T69~T95 | 100 | 6553.5 |
| T96 | 1 | 65.535 |
| T97~T100 | 10 | 655.35 |

| | | |
|-----------|-----|--------|
| T101~T127 | 100 | 6553.5 |
|-----------|-----|--------|

Attention:

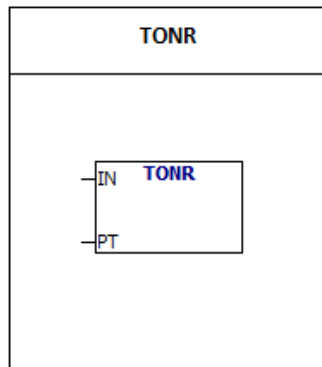
- 1.The value of each timer TXXX is different.
- 2.The resolution of the timer depends on the time base. For example, the error range of the 10-millisecond timer is 10 milliseconds.

Example:



6.16.2 TONR

| Input/output Operand | | Data type |
|----------------------|---|-----------|
| Txxx | Constant(T0—T255) | word |
| IN | Enable bit | Boolean |
| PT | VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, constant, *VD, *LD, *AC | Integer |



TONR: When the value of the input “IN” is equal to 1, timer starts time. Timer current value of Txxx is the current time (a multiple of the time base). When the current value of the timer is equal to the present time (PT), the value of the timer bit is 1. When the value of the input “IN” is equal to 0, if the current value of the timer is less

than the present value, timer current value is retained. Otherwise, the current value of the timer is cleared.

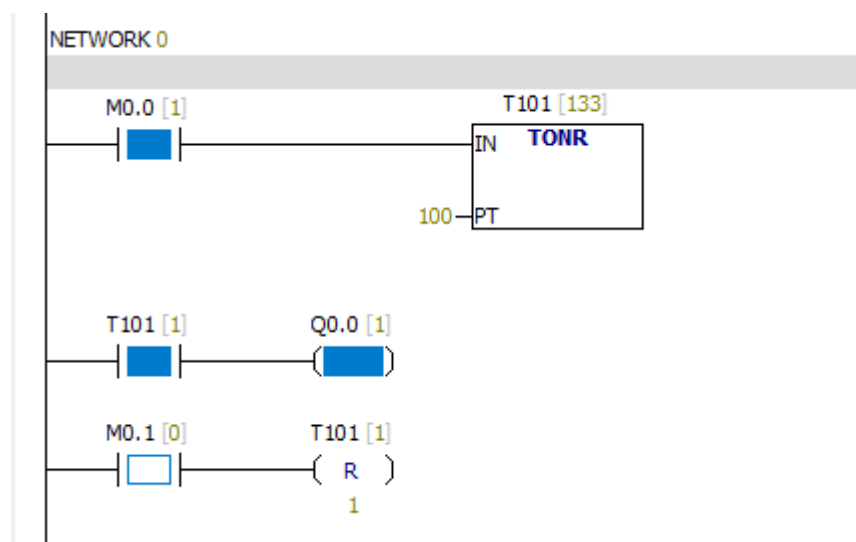
Notes:

You can use TONR to accumulate multiple time intervals.

You can use the "recovery" (R) instruction to recover any timer.

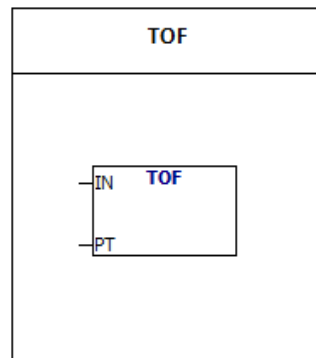
You can only use the "recovery" instruction to recover the TONR timer.

Example:



6.16.3 Disconnect delay timer

| Input/output Operand | | Data type |
|----------------------|---|-----------|
| Txxx | constant(T0—T255) | word |
| IN | Enable bit | Boolean |
| PT | VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, constant, *VD, *LD, *AC | Integer |



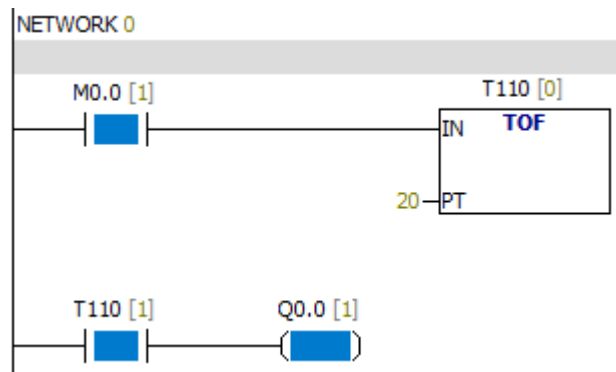
TOF: When the input is closed, the output will be closed for a period of time. When the value of IN is 1, the bit of the timer is 1 Immediately and timer current value is set to 0. When the value of IN is 0, the timer starts time. When the current value is equal to the present value, the bit of the timer is 0.

Notes:

The value of each timer TXXX is different.

You can use the "recovery" (R) instruction to recover TOF timer.

Example:



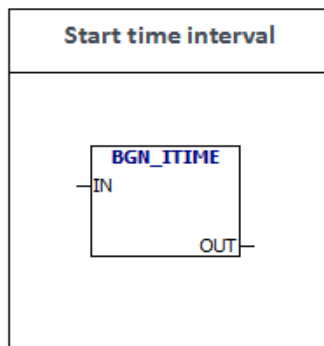
6.16.4 Start time interval

Input/output Operand

Data type

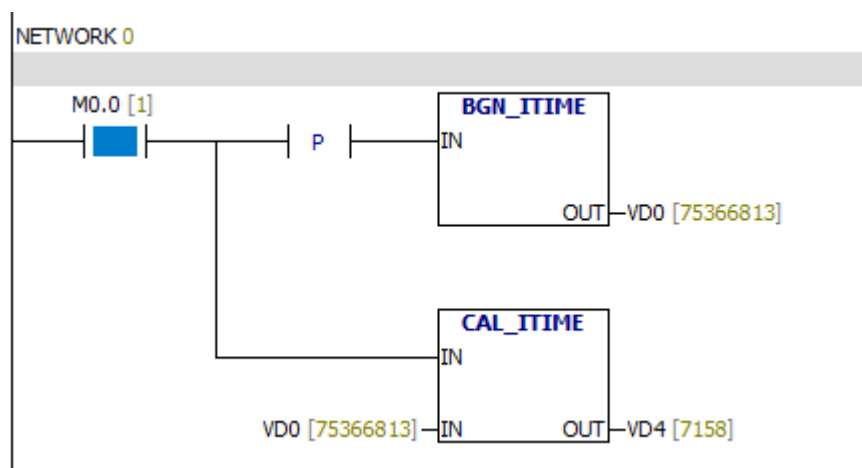
OUT VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Double word



Reads the current value of the built-in 1 ms counter and stores it in the OUT.

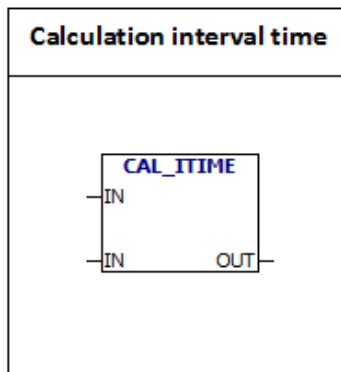
Example:



The value of VD4 is the conduction time of M0.0

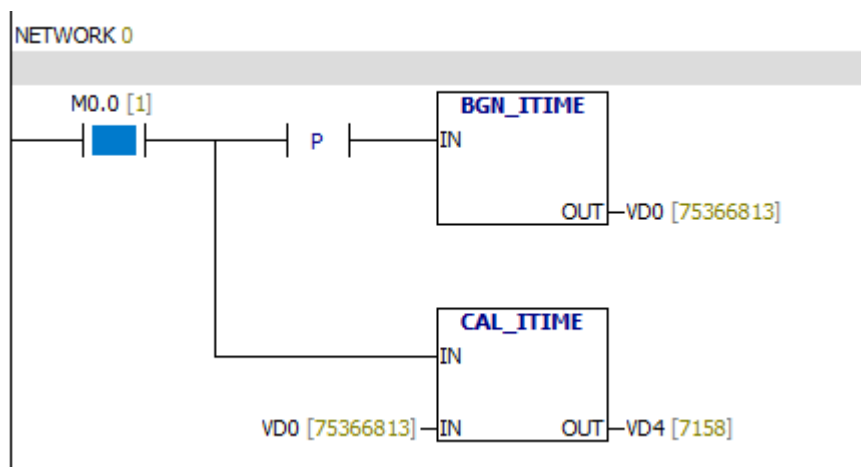
6.16.5 Calculation interval time

| Input/output | Operand | Data type |
|--------------|--|-------------|
| IN | VD, ID, QD, MD, SMD, SD, LD, HC, AC, *VD, *LD, *AC | Double word |
| OUT | VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC | Double word |



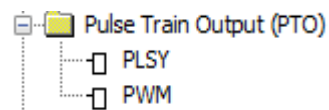
Calculates the time difference between the current time and the time provided by the IN, and stores the time difference in the OUT.

Example:



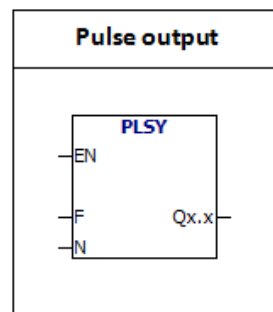
The value of VD4 is the conduction time of M0.0

6.17 Pulse train output



6.17.1 Pulse output

| Input/output | Operand | Data type |
|--------------|--|----------------|
| F | ID, QD, AID, AQD, MD, VD, HC, SMD, LD, *MD, *VD, *LD | Double integer |
| N | ID, QD, AID, AQD, MD, VD, HC, SMD, LD, *MD, *VD, *LD | Double integer |
| OUT | QX.X | Bit |



PLSY: When the value of the enable bit is 1, Instruction issues N pulses. The pulse frequency is F.

PLSY instructions:

1.The frequency range of F is 10 ~ 40K (Hz). Different models have different frequency ranges. Please set the frequency according to the specific model. Frequency F can be changed in the process of pulse transmission, the sending pulse frequency is also changed.

2.The range of N is 0 ~ 2147483647. If N is 0, the number of pulses is ignored. When n is equal to 0 and the enable bit is 1, the PLSY instruction will send pulses ceaselessly.

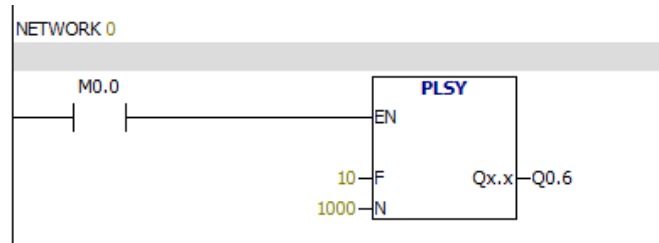
When the pulse is sending, changing the value of N does not work. N changes will be in effect after the next pulse.

3.If the value of the enable bit is 0, the pulse will stop sending. When the enable bit is changed from 0 to 1, PLSY instruction sends new pulses and ignores the interrupted pulses before.

4.The duty ratio of pulse transmission is 50%ON, 50%OFF.The transmission of the pulse is completely processed by the hardware interrupt, which is not

affected by the scan period.

For example:



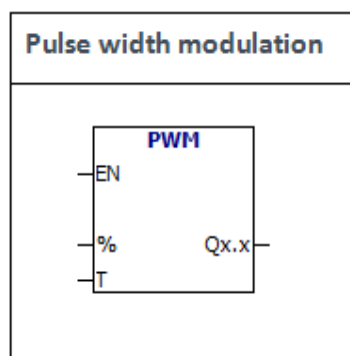
Attention:

Output point must be high speed output point.

For different PLC, the addresses of high-speed output points may be different.

6.17.2 Pulse width modulation

| Input/output | Operand | Data type |
|--------------|--|-------------|
| % | IW, QW, AIW, AQW, MW, VW, T, C, SMW, LW, *MD, *VD, *LD | Double word |
| T | IW, QW, AIW, AQW, MW, VW, T, C, SMW, LW, *MD, *VD, *LD | Double word |
| OUT | Q | Bit |



Pulse width modulation (PWM) instruction initializes the PWM hardware and sends high speed pulses.

The input value of "%" =conduction time/period.

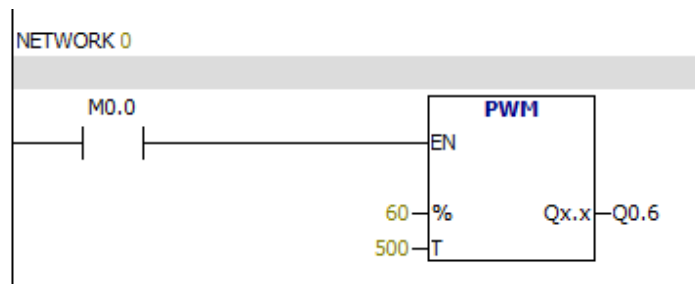
The input value of "T" is the period of the pulse

PWM description:

- 1.The unit of T is 1ms.
- 2.If the input value of "%" is 0, then the instruction does not output the pulse. If the input value of "%" is equal to 100, the value of output pulse is always 1.
- 3.When the pulse is sending, you can change the value of "%"and period of the pulse. Then the value of "%" and period of pulse will change.
- 4.If the value of enable bit is 0, pulse sending will stop. When the enable bit is

changed from 0 to 1, PWM instruction restarts sending pulse.

For example:



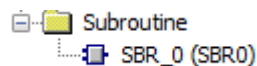
The pulse period is 500 ms, the conduction time is 300 ms.

Attention:

Output point must be high speed output point.

For different PLC, the addresses of high-speed output points may be different.

6.18 Subroutine



6.18.1 Using subroutine

Subroutine is used for program partitioning. When the main program calls subroutine and performs the subroutine, subroutine executes all instructions to the end. Then, the system returns to the main program.

Subroutine is used for program partitioning. It helps to read and manage programs. It also helps to debug and maintain programs. You can use PLC more effectively by using subroutine. Because all of the subroutine blocks are not scanned when they are not called.

If the subroutine only references parameters and local memory, then the subroutine can be moved. In order to move the subroutine, you can't use any global variables / symbols (I, Q, M, SM, AI, AQ, V, T, C, S, AC absolute address). If the subroutine does not call parameters (IN, OUT, or IN_OUT) or only uses local variables, you can export the subroutine and import it into another project.

Conditions of using subroutine:

- 1.Create a subroutine
- 2.Define parameters in the local variable table.
- 3.Call subroutine from the appropriate POU (from the main program or another subroutine)

Using subroutine does not save or restore the accumulator.

6.18.2 Using parameters to call subroutine

Subroutine may contain the transfer parameters. The parameter is defined in the local variable table of the subroutine. Parameters must have a symbol name (up to 23 characters), a variable type, and a data type. Each subroutine can be set up to 16 IN/OUT parameters.

Local variable table has 4 types of variables. They are IN, IN-OUT, OUT and TEMP.

| | | Symbol | Var Type | Data Ty... | Comment |
|--|--------|--------|----------|------------|---------|
| | ✓ L0.0 | ss | IN | BOOL | |
| | | | IN | BOOL | |
| | | | IN_OUT | BOOL | |
| | | | IN_OUT | BOOL | |
| | | | IN_OUT | BOOL | |
| | | | OUT | BOOL | |
| | | | TEMP | BOOL | |

Parameter type and description

IN Parameters are transferred to the subroutine. If the parameter is a direct address (e.g. VB10), the specified location value is transferred to the subroutine. If the parameter is an indirect address (such as *AC1), the specified location value is transferred to the subroutine. If the parameter is the data constant (16#1234) or address (&VB100), constants or addresses are transferred to the subroutine.

IN_OUT The specified location value is transferred to the subroutine. The result of subroutine operation is transferred to the specified same location. This parameter does not allow to use constants (such as 16#1234) and addresses (e.g.&VB100).

OUT The result of subroutine operation is transferred to the specified location. Constants (such as 16#1234) and addresses (e.g. &VB100) are not allowed to be used as output.

TEMP Any local memory which is not used as a transfer parameter can't be used for temporary storage in subroutine.

| Parameter data type | Illustration |
|-------------------------|---|
| Boolean | It is used for unit input and output. |
| Byte, word, double word | Input or output parameters without symbols. |
| Integer, double integer | Input or output parameters with symbols. |
| Real number | It identifies single precision floating point values. |
| String | This data type is used as a four-byte pointer to the string. |
| Enable bit | Boolean enable bit can be used only for bit. It can be used as input. |

6.18.3 How to set up a subroutine

The following methods can be used to establish a subroutine:

- 1.Project manager→program block→Right click program block→Insert→subroutine
 - 2.Project manager→program block→SBR-0→Right click SBR-0→Insert→subroutine
- You can use the local variable table to define the parameters of the subroutine.

Notes:

- 1.Please remember that each POU in the program has an independent local variable table. In the A subroutine, you can only use the A local variable table to define variables.
- 2.Each subroutine can be set up to 16 IN/OUT parameters. If the number of parameters is greater than 16, the program will generate errors.
- 3.You can write a subroutine in the program edit window.
- 4.Click on the label of POU that you want to edit. SO, you can edit the POU in the program edit window.

Editor inserts POU termination instruction automatically. (END for main, RET for SBR, RETI for INT).

6.18.4 How to call a subroutine

You can call a subroutine from the main program, another subroutine or an interrupt routine. You can't call the subroutine from the subroutine itself.

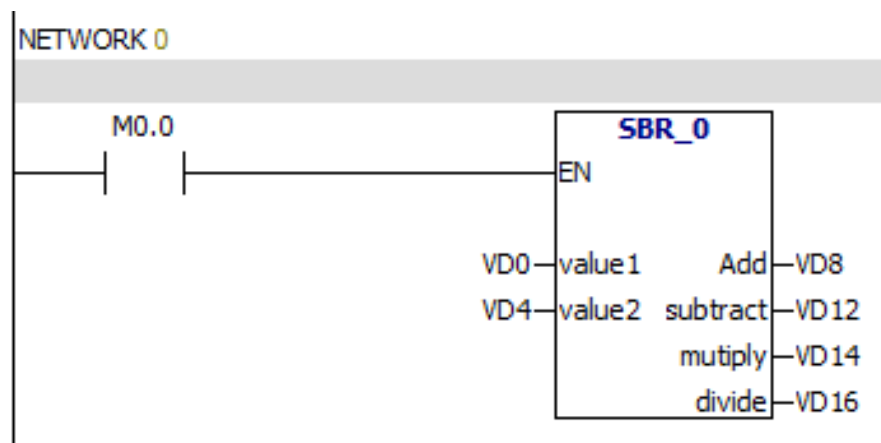
In LAD, the subroutine generates a block instruction. You can call the block instruction to call the subroutine.

Steps to call a subroutine:

1. In program edit window, place the cursor on the position where you want to place the subroutine.
2. Instructions→Subroutine, then select the subroutine that you need. Double click on it.

Example: Four arithmetic operations

Main program:



Subroutine:

Program Editor

| | | Symbol | Var Type | Data Type | Comment |
|---|------|----------|----------|-----------|---------|
| ✓ | LD0 | value1 | IN | DINT | |
| ✓ | LD4 | value2 | IN | DINT | |
| | | | IN | BOOL | |
| | | | IN_OUT | BOOL | |
| ✓ | LD8 | Add | OUT | DINT | |
| ✓ | LD12 | subtract | OUT | DINT | |
| ✓ | LD16 | multiply | OUT | DINT | |
| ✓ | LD20 | divide | OUT | DINT | |
| | | | OUT | BOOL | |
| | | | TEMP | BOOL | |

```

graph LR
    SM00[SM0.0] --- EN_ADD[EN]
    SM00 --- EN_SUB[EN]
    SM00 --- EN_MUL[EN]
    SM00 --- EN_DIV[EN]
    LD0 --- IN1_ADD[IN1]
    LD0 --- IN1_SUB[IN1]
    LD0 --- IN1_MUL[IN1]
    LD0 --- IN1_DIV[IN1]
    LD4 --- IN2_ADD[IN2]
    LD4 --- IN2_SUB[IN2]
    LD4 --- IN2_MUL[IN2]
    LD4 --- IN2_DIV[IN2]
    OUT_ADD[OUT] --- LD8
    OUT_SUB[OUT] --- LD12
    OUT_MUL[OUT] --- LD16
    OUT_DIV[OUT] --- LD20
    
```

MAIN (INT0)

INT_1 (INT1)

SBR_0 (SBR0)

SBR_1 (SBR1)

7.PLC storage area

7.1 Storage area types and properties

| Region | Illustration | Bit | Byte | Word | Double Word | Retain | Force |
|--------|--|------------------------|--------------|----------------------------------|--------------|---------------------|----------|
| I | Discrete input and image register | Read / write | Read / write | Read / write | Read / write | NO | YES |
| Q | Discrete output and image register | Read / write | Read / write | Read / write | Read / write | NO | YES |
| M | Internal memory bit | Read / write | Read / write | Read / write | Read / write | YES | YES |
| SM | Special memory bit (SM0 - SM29 Read-only) | Read / write | Read / write | Read / write | Read / write | NO | NO |
| V | Variable memory | Read / write | Read / write | Read / write | Read / write | YES | YES |
| T | Timer current value and timer bit | Read / write (T bit) | NO | Read / write (T Current value) | NO | T current value YES | T bit NO |
| C | Counter current value and counter bit | Read / write (C bit) | NO | Read / write (C current value) | NO | C current value YES | C bit NO |
| HC | Current value of high speed counter | NO | NO | NO | read-only | NO | NO |
| AI | Analog input | NO | NO | Read-only | NO | NO | YES |
| AQ | Analog output | NO | NO | write only | NO | NO | YES |
| AC | Accumulator register | NO | Read / write | Read / write | Read / write | NO | NO |
| L | Local variable memory | Read / write | Read / write | Read / write | Read / write | NO | NO |

| | | | | | | | | | | | |
|---|-----|---------------|---|---------------|---|---------------|---|---------------|---|----|----|
| S | SCR | Read write | / | Read write | / | Read write | / | Read write | / | NO | NO |
|---|-----|---------------|---|---------------|---|---------------|---|---------------|---|----|----|

7.2 Direct and indirect addressing

When you write the program, you can use the three ways to address instruction:

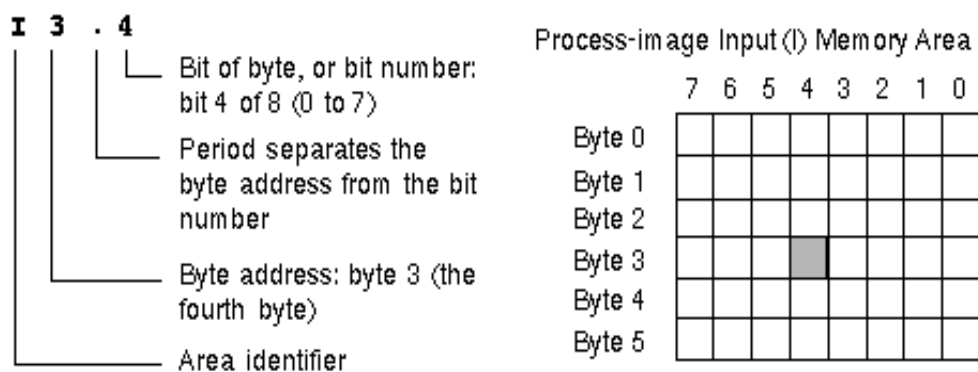
- 1.Direct addressing
- 2.Symbol addressing
- 3.indirect addressing

Direct addressing

PLC can directly specify the memory area, size, and location;

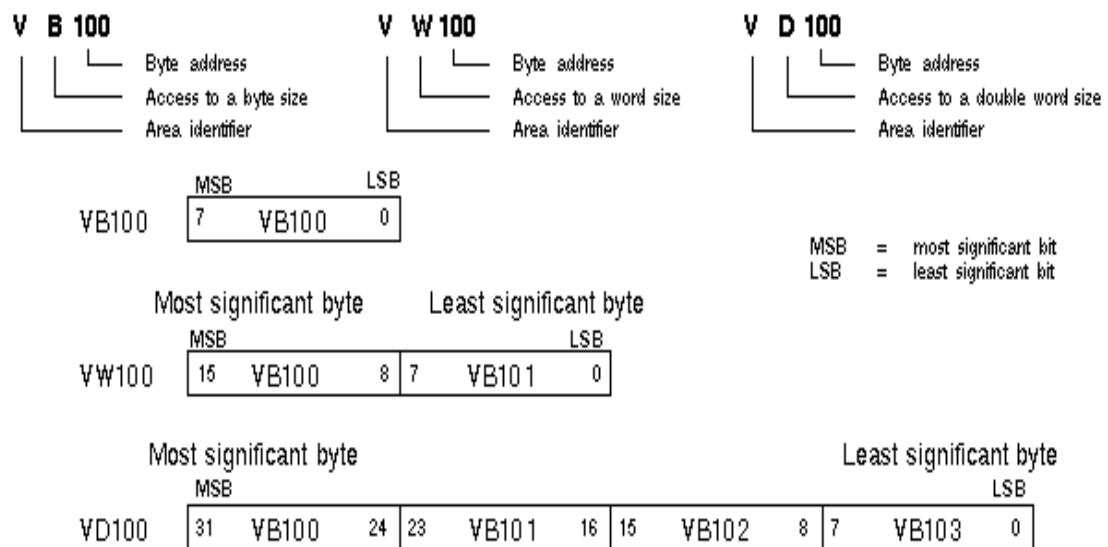
In order to read/write a bit in the memory area, you need to specify the address. The address Includes memory area identifier, byte address, a period and number.

Example:



Specifying byte、word and double word addresses are similar to specifying bit address.

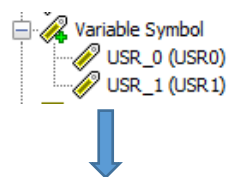
Example:



Symbolic addressing

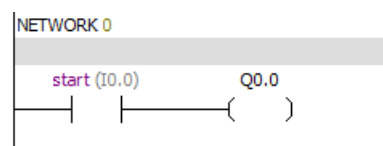
Symbol addressing consists of letters, numbers and characters.

You can set the symbol of address by the following steps:



| | Symbol | Adress | Data Ty... | Comment |
|---|--------|--------|------------|---------|
| ✓ | start | I0.0 | BOOL | start |
| | | | BOOL | |
| | | | BOOL | |
| | | | BOOL | |

You can enter "start" as the address of I0.0



Indirect addressing

Indirect addressing uses pointer to access the data of memory. Pointer is a double word. It contains the address of another memory location. Only V memory location, L memory location or register accumulator (AC1, AC2, AC3) can be used as pointers.

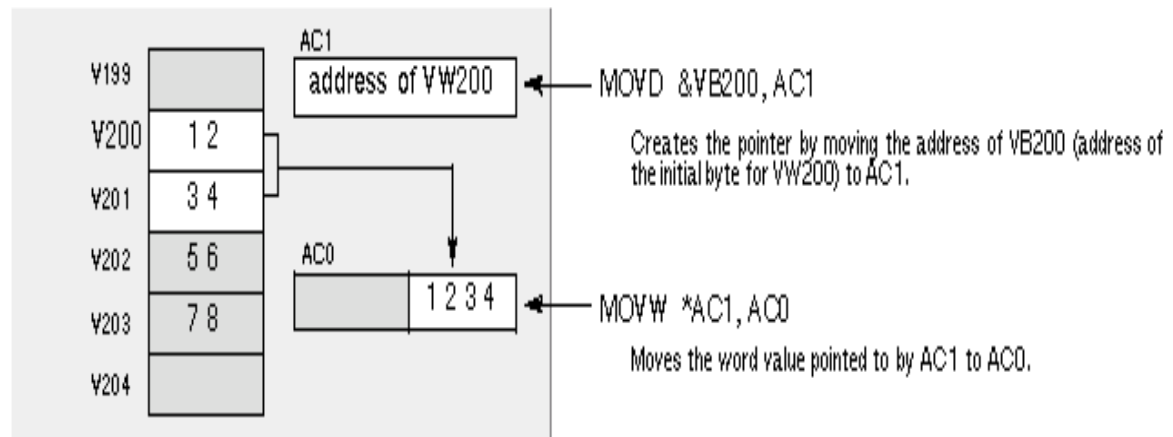
PLC allows the pointer to access the following memory area: I, Q, V, M, S, T, C, T and

C can only use the current value.

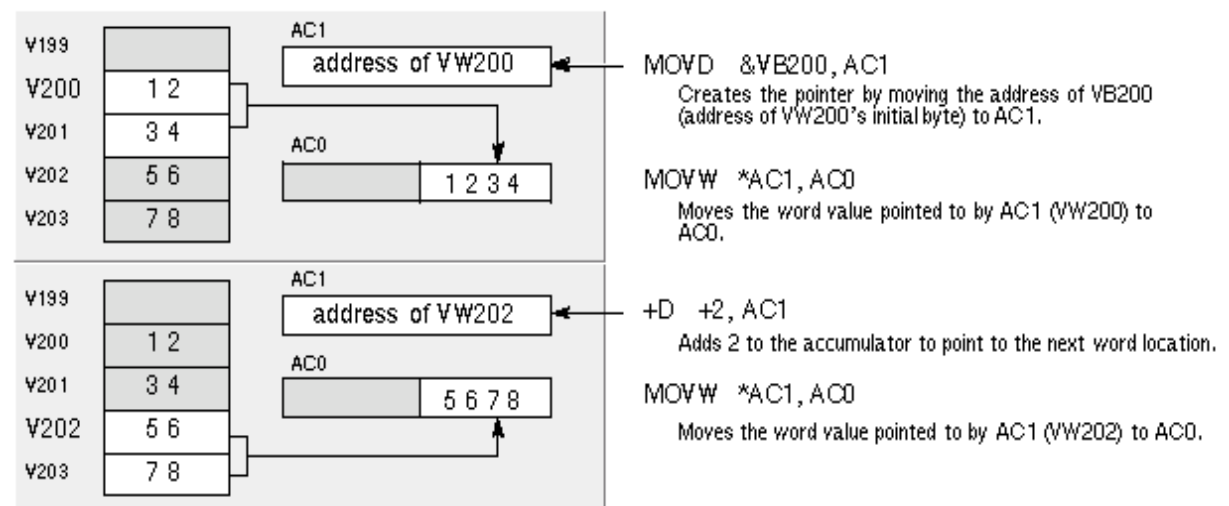
Pointer consists of memory location and symbol "&".

To specify the operand be a pointer, you should input an asterisk (*) in front of the operand.

Example: The values stored in the VB200 and VB201 are moved to AC0.



As shown in the figure below, you can change the pointer value. Because the pointer is a 32-bit value, you should use the double word instruction to modify the pointer value.



Prompt:

If you use the pointer to execute the byte operation, the minimum pointer interval is

1.

If you use the pointer to execute the word operation, the minimum pointer interval is

2.

If you use the pointer to execute the double word operation, the minimum pointer

interval is 4.

If the value of the pointer is greater than the maximum value of the V memory, program will generate errors.

The current value of the timer and counter is 16 bits, so the minimum pointer interval is 2.

7.3 Bit, byte, word and double word access

Bit access

If you want to access a bit, you need to specify the address of the bit. Address contains region identifier and byte number. Zero is the first address of all data areas.

The decimal point is used to separate the number of bytes and the number of bits.

The range of the number of the bits is 0~7. For example: M0.0

Byte, word and double word access

If you want to access byte, word or double word, you need to specify the address.

Address contains a region identifier, a letter and an address number.

For example:

VB100 Access V memory address byte 100

VW100 Access V memory address bytes 100 and 101

VD100 Access V memory address bytes 100, 101, 102, and 103

7.4 Memory address range

| Bit | | Byte | | Word | | Double Word | |
|----------|-----------------|-----------|--------------|-----------|--------------|-------------|--------------|
| I | I0.0~I31.7 | IB | IB0~IB31 | IW | IW0~IW3 0 | ID | ID0~ID28 |
| Q | Q0.0~Q31. .7 | QB | QB0~QB3 1 | QW | QW0~QW 30 | QD | QD0~QD 28 |
| M | M0.0~31. | MB | MB0~MB | MW | MW0~M | MD | MD0~MD |

| | | | | | | | |
|-----------|---------------|------------|-------------|------------|-------------|------------|-------------|
| | 7 | | 31 | | W30 | | 28 |
| S | S0.0~S31.7 | SB | SB0~SB31 | SW | SW0~SW30 | SD | SD0~SD28 |
| SM | SM0.0~SM551.7 | SMB | SMB0~SMB551 | SMW | SMW0~SMW550 | SMD | SMD0~SMD548 |
| T | T0~T255 | | | T | T0~T255 | | |
| C | C0~C255 | | | C | C0~C255 | | |
| V | V0.0~V8191.7 | VB | VB0~VB8191 | VW | VW0~VW8190 | VD | VD0~VD8188 |
| L | L0.0~L63.7 | LB | LB0~LB63 | LW | LW0~LW62 | LD | LD0~LD60 |
| | | AC | AC0~AC3 | AC | AC0~AC3 | AC | AC0~AC3 |
| | | | | | | HC | HC0~HC15 |
| | | | | | | | |
| | | | | | | | |

7.5 Data type

| Data Type | Data width | Range |
|--------------|------------|--------------------------------|
| BOOL | 1 | 0~1 |
| BYTE | 8 | 16#00~16#FF |
| WORD | 16 | 16#0000~16#FFFF |
| DWORD | 32 | 16#00000000~16#FFFFFFFF |
| SINT | 8 | -128~127 |

| | | |
|-------|----|------------------------|
| INT | 16 | -32768~32767 |
| DINT | 32 | -2147483648~2147483647 |
| USINT | 8 | 0~255 |
| UINT | 16 | 0~65535 |
| UDINT | 32 | 0~4294967295 |

7.6 Constant

| Unsigned integer range | | | Signed integer range | |
|------------------------|-----------------------------|--------------------|-----------------------------|-------------------------|
| Data size: | Decimal digit: | Hexadecimal digit: | Decimal digit: | Hexadecimal digit: |
| B (byte) | 0~255 | 0~FF | -128 ~+127 | 80~7F |
| W (word) | 0~65535 | 0~FFFF | -32768~+32767 | 8000~7FFF |
| D (double word) | 0~4294967295 | 0~FFFF FFFF | -2147483648 ~+2147483647 | 8000 0000~ 7FFF FFFF |
| Data size: | Decimal numbers (positive) | | Decimal number (negative) | |
| D (double word) | +1.175495E-38~+3.402823E+38 | | -1.175495E-38~-3.402823E+38 | |

8. Assignment and function of SM special storage area

SMB0

| | | |
|---------------|-------|----------------------------------|
| Always_On | SM0.0 | Always ON |
| First_Scan_On | SM0.1 | ON for the first scan cycle only |
| Clock_60s | SM0.4 | 30 seconds OFF, 30 seconds ON |
| Clock_1s | SM0.5 | 0.5 second OFF, 0.5 second ON |

SMB1

| | | |
|------------------|-------|---|
| Result_0 | SM1.0 | Set to 1 by the execution of certain instructions when the operation result = 0 |
| Overflow_Illegal | SM1.1 | Set to 1 by exec. of certain instructions on overflow or illegal numeric value. |
| Neg_Result | SM1.2 | Set to 1 when a math operation produces a negative result |
| Divide_By_0 | SM1.3 | Set to 1 when an attempt is made to divide by zero |
| Table_Overflow | SM1.4 | Set to 1 when the Add to Table instruction attempts to overfill the table |
| Table_Empty | SM1.5 | Set to 1 when a LIFO or FIFO instruction attempts to read from an empty table |
| Not_BCD | SM1.6 | Set to 1 when an attempt is made to convert a non-BCD value to a binary value |
| Not_Hex | SM1.7 | Set to 1 when an ASCII value cannot be converted to a valid hexadecimal value |

The PLC variables address of LCD keys:

F1 → SM191.0
 F2 → SM191.1
 F3 → SM191.2
 F4 → SM191.3

ESC → SM190.0
 OK → SM190.1
 UP → SM190.2
 DOWN → SM190.3
 LEFT → SM190.4
 RIGHT → SM190.5

When the value of SM192.0 is equal to 1, LCD will be bright.

When the value of SM192.0 is equal to 0, LCD will be dark.

SMW22-SMW26 Scan time (millisecond)

SMW22 Scan time of the last scan.

SMW24 Minimum scan time

SMW26 Maximum scan time

Newly added special registers and their functions

| | |
|----------------|---|
| SMB10 | Adjust the contrast of LCD display. (value range: 0~25) |
| SMB192 | Control the LCD backlight on or off. (0: off, 1: always on) |
| SMB12 SMB13 | <p>Used to indicate the connection status of the expansion module.</p> <p>SM12.0 - SM12.7: The status of the bit corresponds to the connection status of the expansion modules with station address 1 to 8.</p> <p>SM13.0 - SM13.7: The status of the bit corresponds to the connection status of the expansion modules with station address 9 to 16.</p> <p>‘0’ means the expansion is not connected/communicating with the CPU, ‘1’ means the expansion is connected/communicating with the CPU normally.</p> |
| SMB14 | <p>Working status of built-in SD:</p> <p>0: There are no errors.</p> <p>1: Failed to open file.</p> <p>2: Update file new content error, write failure.</p> <p>3: Exceeds the maximum allowed file size</p> <p>4: Insufficient memory, write failed</p> <p>60: No SD card detected</p> |

* Note that the functions in this table have requirements on the PLC firmware version.

9.xladder communication

9.1 PLC basic introduction of network communication

Rievtech PLC is designed to solve your communications and networking needs. It supports both simple networks and complex networks. xladder makes it simple to set up and configure your network.

Master slave network definition

Rievtech PLC supports master slave network. It can be used as the master station or the slave station in the network. xladder is always used as the master station.

Master station: The master station can send a request to another device in the network. The master station can also respond to requests from other master stations in the network.

Slave station: The device which is configured to be the slave station can only respond to requests from a master station. The slave station will not take the initiative to issue a request.

The concept of baud rate and network address

The speed of transmission of data in the network is called the baud rate. Units are kbaud and Mbaud. For example, 19.2 kbaud indicates that 19200 bits are transmitted per second.

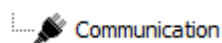
Each device must be the same baud rate in the network. So, the communication speed of the network is decided by the minimum baud rate of equipment.

The range of Rievtech PLC baud rate is 1200 bps~115200 bps. The default value is 9600bps.

Network address is a unique number that you specify for each device on the network. Network address ensures that data is delivered to the correct device. The range of PLC network addresses is 0~255.

Set the baud rate and network address of xLadder


Open the communication in project management:



The 'Communication' dialog box has a 'Modbus TCP/IP' tab. It includes a 'Search' button and a 'Default' button. The 'Station' dropdown is set to '0'. The 'Port' dropdown is set to 'MOXA Communication Port 2 (COM3)'. A 'Bus Parameters' section contains three dropdowns: 'Baud Rate' set to '9600 bps', 'Parity' set to 'NONE', and 'Stop Bit' set to '1 Bit'. At the bottom are 'OK' and 'Cancel' buttons.

You can set the station number, port, baud rate, parity and stop bit of xladder. The default station number is 0. The default baud rate is 9600 bps.

Set the baud rate and network address of Rievtech PLC

Open system block in project management  **System Block**

The 'System Block' dialog box has multiple tabs: 'RS232/RS485', 'RS232/RS485', 'Password', 'Retentive Ranges', 'Interrupt Time', 'Force Table', 'AI Parameters', and 'Bacnet Parameters'. The 'Defaults' button is at the top right. Below are two columns for 'COM 0' and 'COM 1'. Each column has a 'Protocol' dropdown set to 'Modbus'. The 'Station number' is set to '1' for both, with a range of '(range 0... 255)'. The 'Baud rate' is set to '9600 bps' for both. The 'Data bits' is set to '8 (RTU)' for both. The 'Parity' is set to 'NONE' for both. The 'Stop bits' is set to '1 Bit' for both. The 'Response timeout (100ms)' is set to '10' for both, with a range of '(range 0... 255)'. The 'Interval frame delay (B)' is set to '10' for both, with a range of '(range 0... 255)'. A note at the bottom states: 'Configuration parameters must be downloaded before they take effect'. At the bottom right are 'OK' and 'Cancel' buttons.

You can set the station number, baud rate, data bits, parity bit and stop bit of PLC.

Attention: Only when xLadder software station number is equal to 0 or PLC station number, you can download the program to PLC.

9.2 Rievtech PLC communication

Rievtech PLC support free port communication, MODBUS communication.

Free port communication:

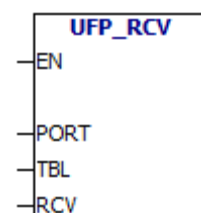
Free port communication is a half-duplex communication based on RS-485 communication. Users can make their own communication protocol in free port communication. The third-party devices mostly support RS-485 communication.

The core of the free port communication is receiving and sending instructions. RS-485 communication can't receive and send data at the same time. The RS-485 communication format includes a start bit, 7 or 8 bit characters, a parity bit, and a stop bit.

Free port communication baud rate can be set to 1200, 2400, 4800, 9600, 19200, 38400, 57600 or 115200. Devices that meet the above conditions can communicate with PLC. Free port mode has great flexibility.

Free port instructions:

UFP_RCV



UFP_RCV: Receive data instruction

- PORT: Communication port.
- TBL: Configuration table, If the input is MB200.

MB200 is the configuration byte:

(Instruction output) M200.0 Communication preparation

(Instruction output) M200.1 Communication completion

(Instruction output) M200.2 Communication error

(Instruction input) M200.3 Send CRC check

(Instruction input) M200.4 Send CRC check

(Instruction input) M200.5 Receive CRC check

(Instruction input) M200.6 Receive CRC check

(Instruction output) MB201 Error number: 0 indicates no error.

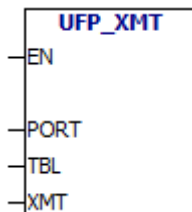
- RCV: receive data, If the input is MB400:

(Instruction input) MW400: Receive data FIFO buffer size (byte unit)

(Instruction output) MW402: The size of the received data (in bytes)

(Instruction output) MB404 ~ ... receive data.

UFP_XMT



UFP_XMT: Send data instruction

- PORT Communication port.
- TBL: Configuration table, If the input is MB200.

MB200 is the configuration byte:

(Instruction output) M200.0 Communication preparation

(Instruction output) M200.1 Communication completion

(Instruction output) M200.2 Communication error

(Instruction input) M200.3 Send CRC check

(Instruction input) M200.4 Send CRC check

(Instruction input) M200.5 Receive CRC check

(Instruction input) M200.6 Receive CRC check

(Instruction output) MB201 Error number: 0 indicates no error.

- XMT Send data FIFO, If the input is MB400:

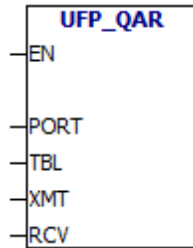
(Instruction input) MW400 Sending data FIFO buffer size

(byte unit)

(Instruction input) MW402 Sending data size (byte)

(Instruction input) MB404 ~ ... Send data.

UFP_QAR



UFP_QAR: Sending and receiving data instruction

- PORT Communication port.
- TBL: Configuration table, If the input is MB200.

MB200 is the configuration byte:

(Instruction output) M200.0 Communication preparation

(Instruction output) M200.1 Communication completion

(Instruction output) M200.2 Communication error

(Instruction input) M200.3 Send CRC check

(Instruction input) M200.4 Send CRC check

(Instruction input) M200.5 Receive CRC check

(Instruction input) M200.6 Receive CRC check

(Instruction output) MB201 Error number: 0 indicates no error.

- XMT: Send data FIFO, If the input is MB300:

(Instruction input) MW300 Sending data FIFO buffer size
(byte unit)

(Instruction input) MW302 Sending data size (byte)

(Instruction input) MB304 ~ ... Send data.

- RCV receive data FIFO, If the input is MB400:

(Instruction input) MW400 Receive data FIFO buffer size
(byte unit)

(Instruction output) MW402 The size of the received data
(byte unit)

(Instruction output) MB404 ~ ... Receive data.

UFP_RCV、UFP_XMT、UFP_QAR error numbers:

- 1 Port doesn't exist
- 2 Port isn't enabled
- 3 Communication task queue is full
- 4 Table error
- 5 Sent data error
- 6 Timeout
- 7 Received data error
- 8 Receive data check error

The use of free port communication instructions will be illustrated with examples in the additional chapter.

MODBUS communication protocol

MODBUS protocol is a common language used in electronic controllers. Different devices can communicate with each other by using the MODBUS communication protocol. It has become a general industrial standard. You can use it to connect different devices.

This protocol defines a message structure, no matter what network they use to communicate. It describes the process of controller requesting to access other devices. It has formulated message domain structure and the common format of the content.

MODBUS network protocol determines that each controller should know its address. It identifies the messages sent from different addresses and decides what action to take. The controller generates feedback information, the format of the information is the information format of MODBUS. It is issued through the MODBUS protocol.

MODBUS address usually contains data type and offset. MODBUS address contains a total of 5 characters. The first character represents the data type and the other four characters represent the correct values in the data type.

MODBUS addressing:

0XXXX are discrete outputs

1XXXX are discrete inputs

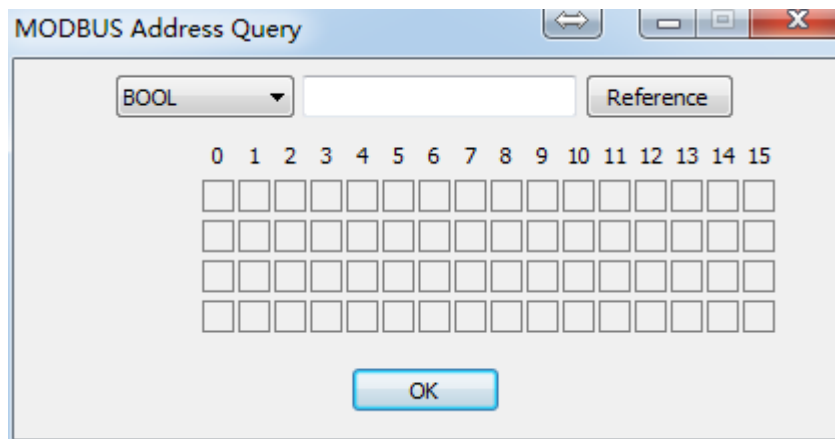
3XXXX are analog inputs

4XXXX are hold registers

You can use “MODBUS address query” to query the MODBUS address of the variable,

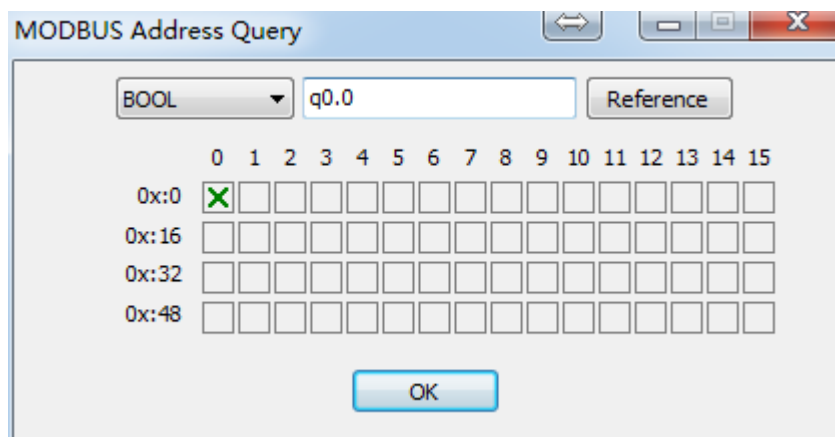
Steps are as follows:

Menu bar→PLC→MODBUS address query



The dialog box titled "MODBUS Address Query" has a dropdown menu set to "BOOL" and an empty text field. To the right is a "Reference" button. Below these is a grid of 16 columns (0-15) and 4 rows of checkboxes. An "OK" button is at the bottom.

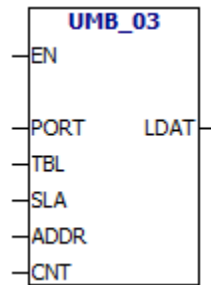
For example: Query the MODBUS address of Q0.0:



The dialog box titled "MODBUS Address Query" has a dropdown menu set to "BOOL" and a text field containing "q0.0". To the right is a "Reference" button. Below these is a grid of 16 columns (0-15) and 4 rows of checkboxes. The first checkbox in the first row (0x:0, column 0) is checked with a green 'X'. An "OK" button is at the bottom.

MODBUS instructions:

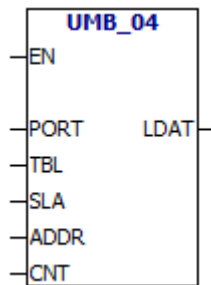
UMB03



Read more than one hold registers

- EN: enable or not enable
- TBL: Configuration table, If the input is MB200:
MB200 is the configuration word
(Instruction Output) M200.0 Communications have been queued
(Instruction Output) M200.1 Communication completion
(Instruction Output) M200.2 Communication error
(Instruction Output) MB201 is error number.0 indicates no error.
- SLA: MODBUS slave address
- ADDR: The offset of hold register (The offset of 4X)
- CNT: Number of hold register
- LDAT: Store the data which was written from slave station

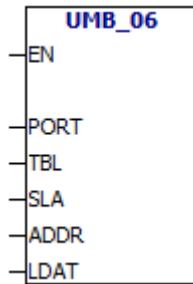
UMB04



Read the input register

- EN: enable or not enable
- TBL: Configuration table, If the input is MB200:
MB200 is the configuration word
(Instruction Output) M200.0 Communications have been queued
(Instruction Output) M200.1 Communication completion
(Instruction Output) M200.2 Communication error
(Instruction Output) MB201 is error number.0 indicates no error.
- SLA: MODBUS slave address
- ADDR: The offset of input register. (The offset of 3x)
- CNT: Number of input registers.
- LDAT: Store the data which was written from slave station

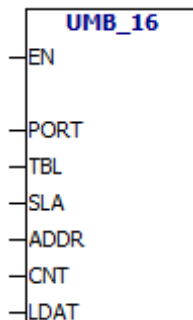
UMB06



Write a single hold register

- EN: enable or not enable
- TBL: Configuration table, If the input is MB200:
MB200 is the configuration word
(Instruction Output) M200.0 Communications have been queued
(Instruction Output) M200.1 Communication completion
(Instruction Output) M200.2 Communication error
(Instruction Output) MB201 is error number.0 indicates no error.
- SLA: MODBUS slave address
- ADDR: The offset of hold register (The offset of 4X)
- LDAT: Store the data which will be written to the slave station.

UMB16



Write more than one hold registers

- EN: enable or not enable
- TBL: Configuration table, If the input is MB200:
MB200 is the configuration word
(Instruction Output) M200.0 Communications have been queued
(Instruction Output) M200.1 Communication completion
(Instruction Output) M200.2 Communication error
(Instruction Output) MB201 is error number.0 indicates no error.
- SLA: MODBUS slave address
- ADDR: The offset of hold register (The offset of 4X)
- CNT: Number of hold register
- LDAT: Store the data which will be written to the slave station.

The use of MODBUS communication instructions will be illustrated with examples in the additional chapter.

9.3 Optimize network performance

The following factors will affect the performance of the network (The baud rate and the master station produce the greatest impact to the performance of the network):

Baud rate: It determines the speed of network communication.

Number of master stations on the network: To enhance network performance, you can reduce the number of main stations on the network. Each station on the network will increase the additional requirements of the network.

Select master station and slave station addresses: The master station address should be continuous. When there is a spacing address between the master stations, the master station will check the spacing address ceaselessly and see if there is another master station waiting to be on line. So the master station spacing address will increase the additional requirements of the network. You can set the slave address to any value. But slave station address can't be placed between the master station address. Or it will increase the additional requirements of the network.

10. Additional chapter

10.1 How to switch PLC mode

How to switch between FBD and LADDER

1, FBD → Ladder

The default factory system of RIEVTECH-PLC is FBD.

As the CPU chips used by each series of products are different, there will be three methods to switch from FBD mode to ladder mode.

There are 3 situations:

| | Applicable Model |
|-----|--|
| (1) | PR-12DC-DA-R, PR-12AC-R, SR-12AC-R, SR-12DC-DA-R |
| (2) | PR-14, 18, 24 series, EXM-12DC-DA-RT-WIFI |
| (3) | PR-12N, 18N, 26N series |


(1) PR-12DC-DA-R, PR-12AC-R, SR-12AC-R, SR-12DC-DA-R

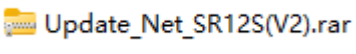
These 4 types of PLC need to use firmware update package to burn the firmware of the corresponding mode.

1. Download the firmware update package for the corresponding series.

The firmware update package on the REVTECH website:

<https://www.rievtech.com/download.html>

PR-12 Series: 

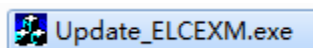
SR-12 Series: 

Note: The suffix date of the update package file name will change accordingly.

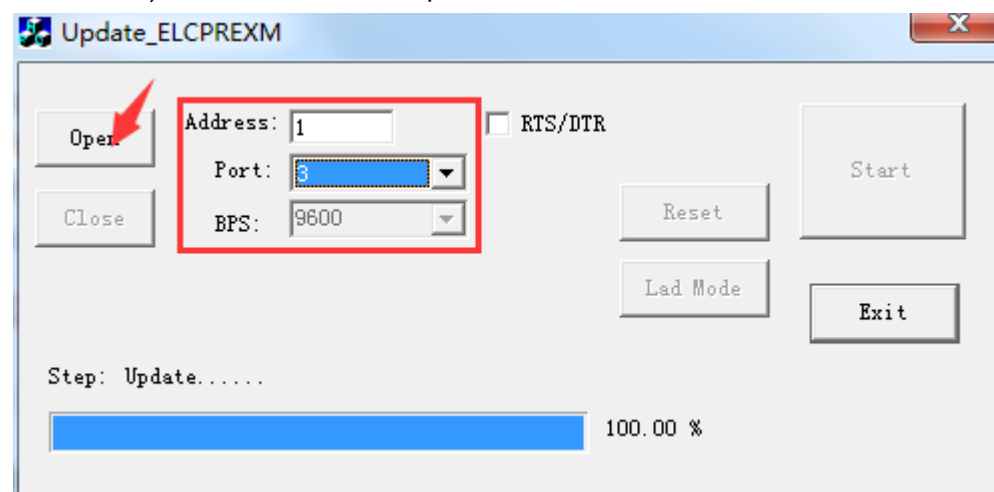
2. After the download is complete, press it into a folder.

The following takes the PR-12 series as an example. The PR-12 series updates the firmware via the serial port, and the SR-12 series updates the firmware via Ethernet.

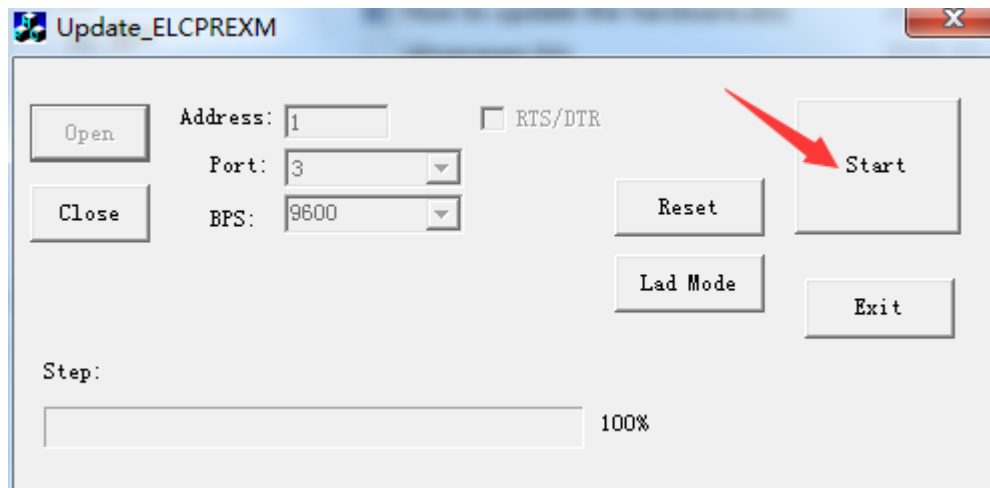
Execute the exe file in the folder.



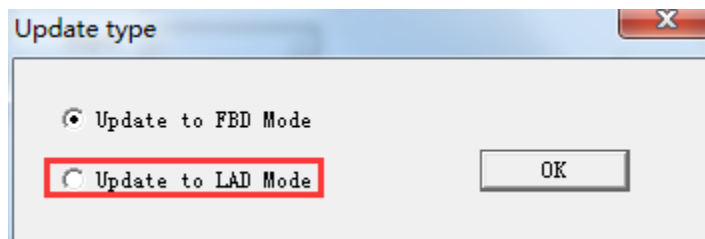
3. Select the com port of the programming cable (you need to set the BPS of PLC com0 to 9600, RTU), and then click the 'Open' button.



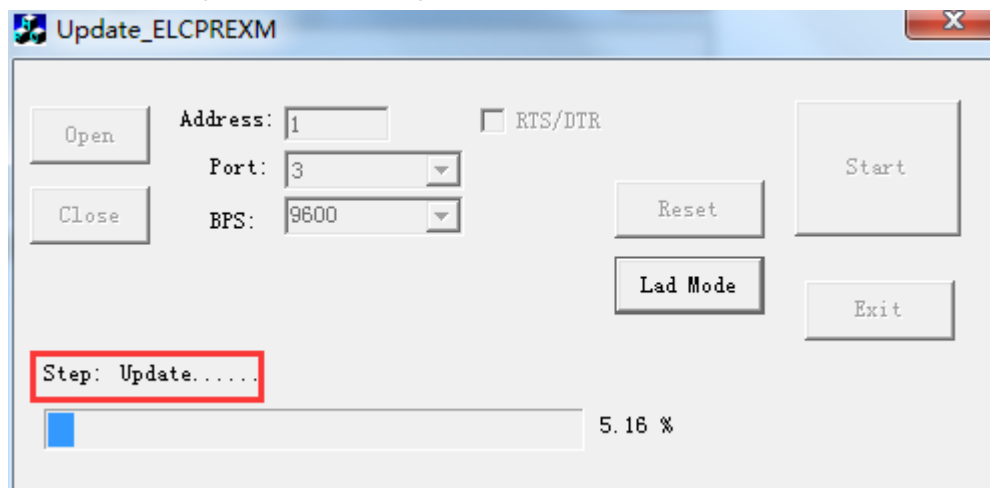
See below:



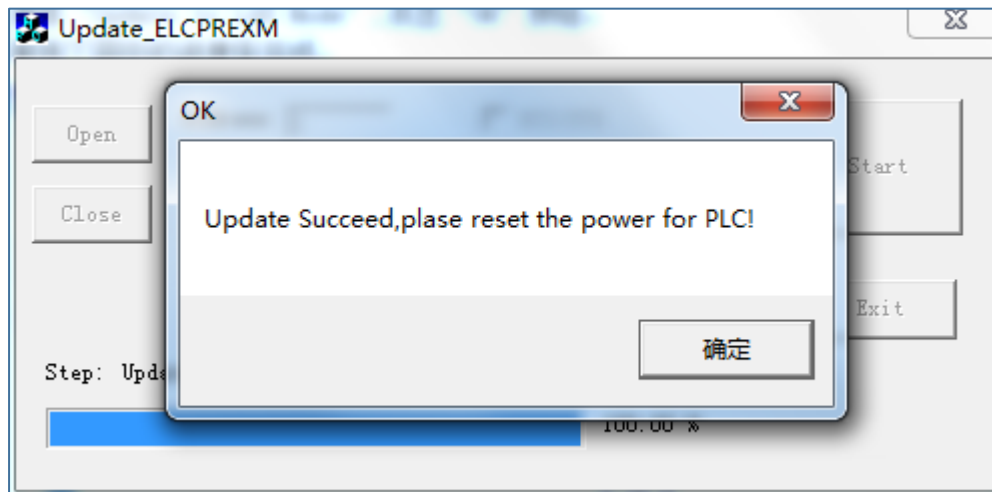
4. Then click the 'Start' button, and the following prompt will pop up:



5. Select 'Update to LAD Mode' and click the 'OK' button. Wait for the progress bar in the figure below to complete.



6. After the update is completed, the following prompt will appear.



7, Power off the PR-12DC-DA-R and then power on and restart, and the LCD will display

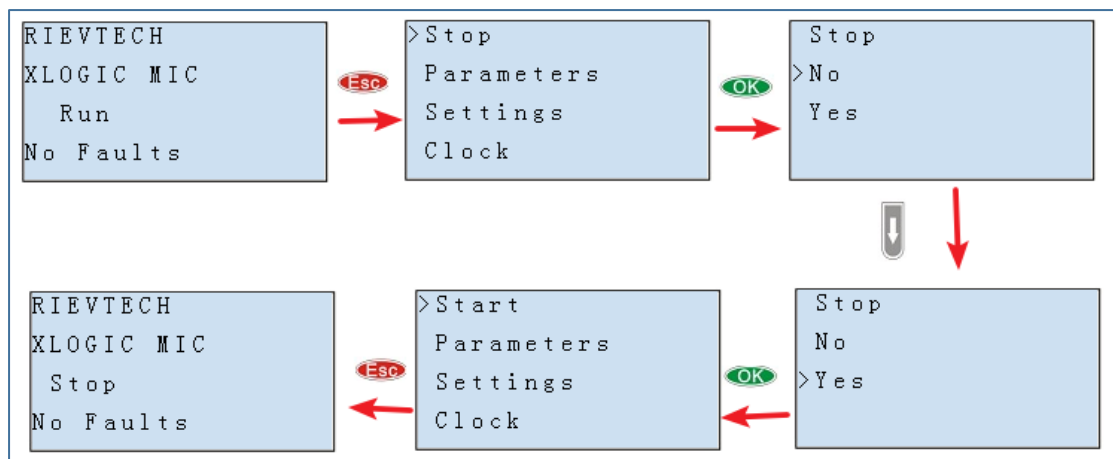
LADDER MODE

RUN

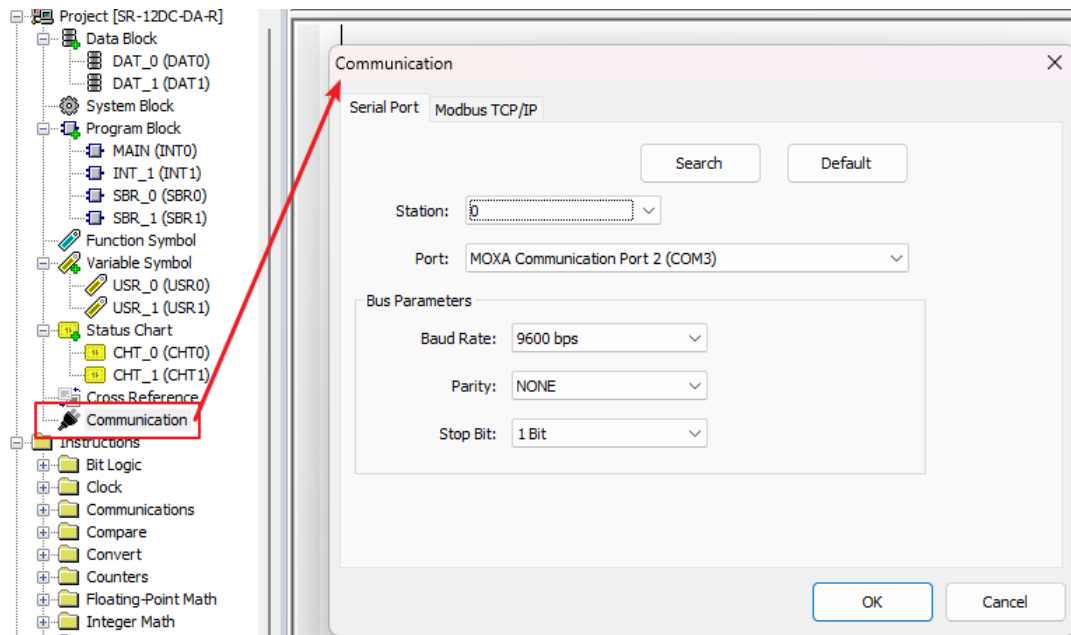
Now this PLC works in ladder mode.

(2) PR-14, 18, 24 series, EXM-12DC-DA-RT-WIFI

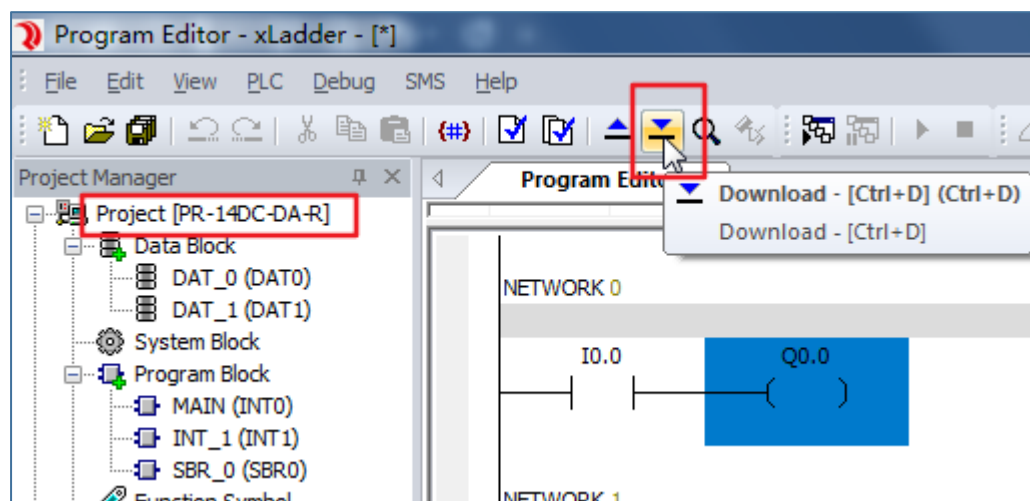
1. Set the PLC to the STOP state on the LCD panel, the steps are as follows:



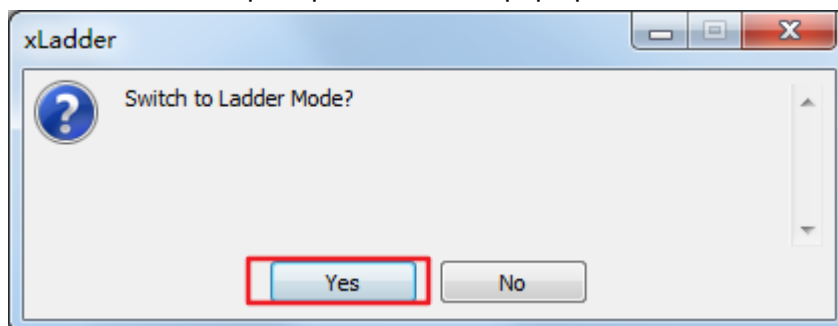
2. Open the xLadder software and select the com port of the programming cable, as shown below:

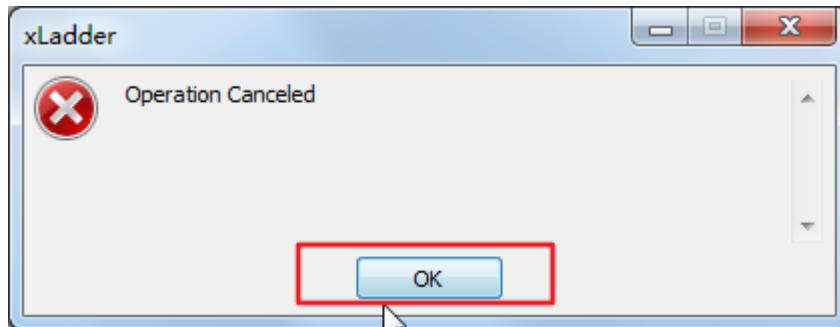
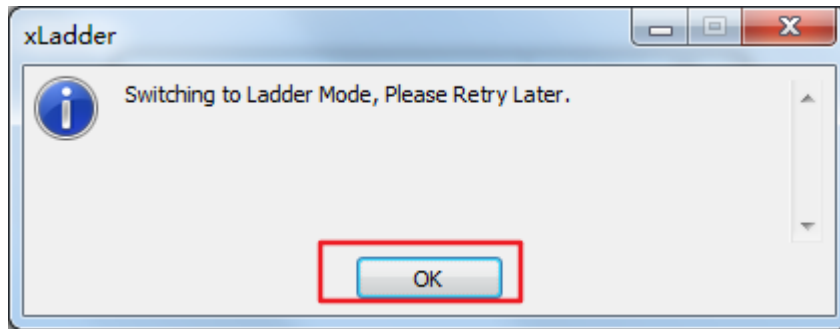


3. Choose the correct PLC model, such as PR-14DC-DA-R. Then click the 'Download' button in the picture below.



4. Then a series of prompt windows will pop up as follows.





5. Then, the following characters will be displayed on the LCD:

LADDER MODE
RUN

This PLC has successfully switched to ladder mode.

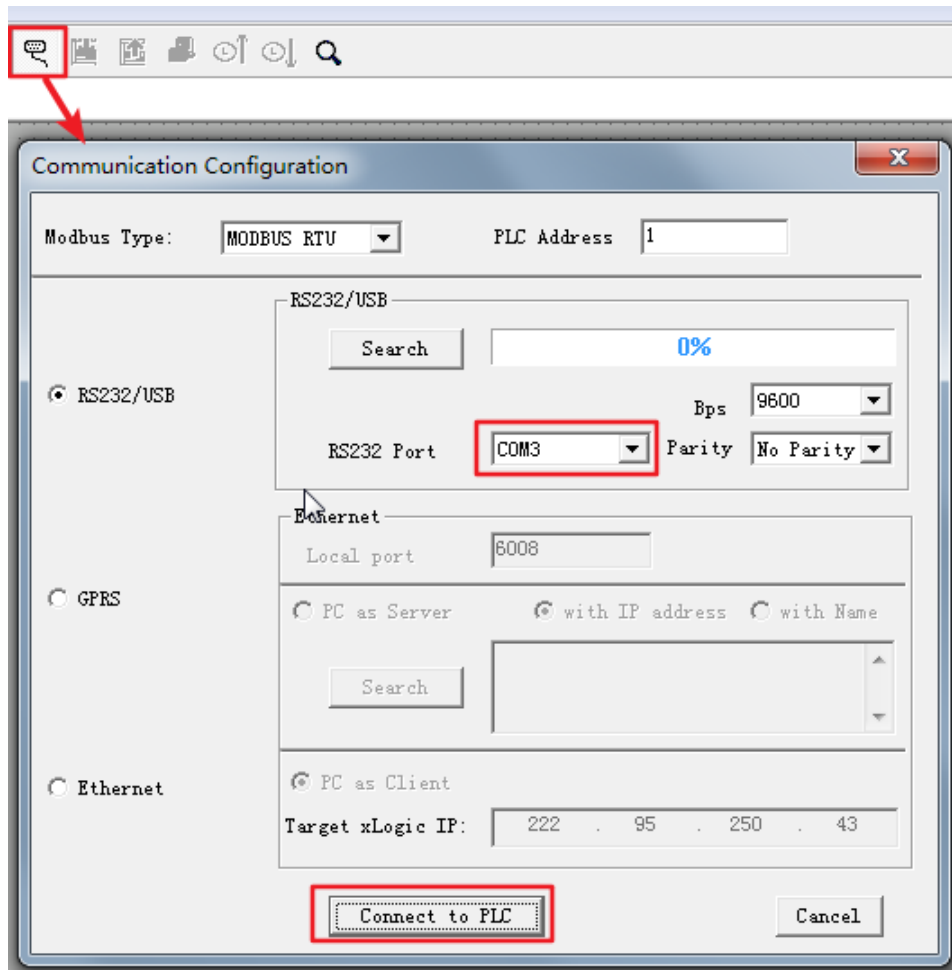
(3) PR-12N, 18N, 26N series

These models do not require mode conversion, and programs in Rievtech software and xLadder software can be downloaded directly to the PLC.

2, Ladder → FBD

The following method is applicable to PR-12,14,18,24 series, EXM-12DC-DA-R-T-WIFI.

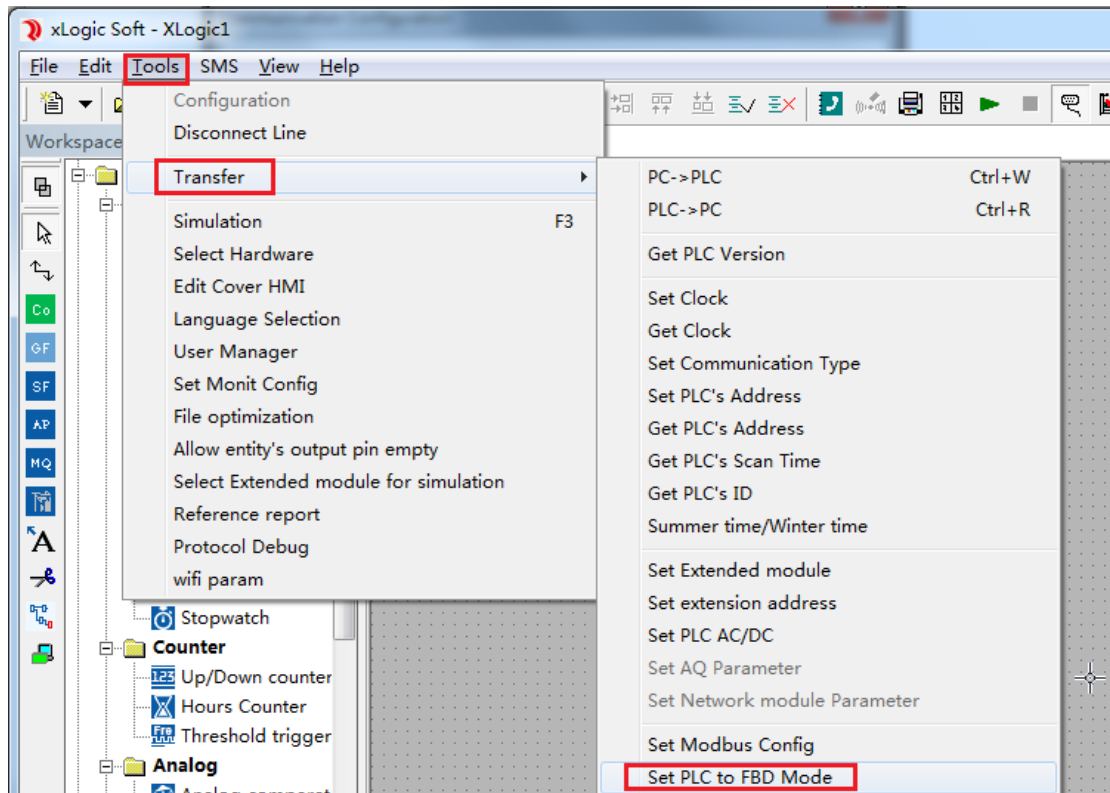
1. Open the Rievtech software and select the COM port of the programming cable, as shown below:



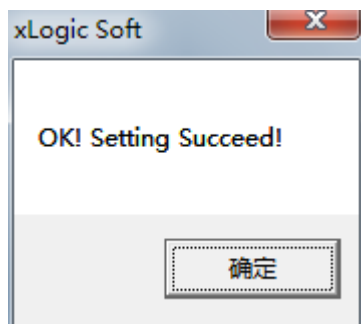
After the connection is successful, the gray invalid tool button on the RieviTech software becomes valid.



2. Follow the path in the figure below to find the 'Set PLC to FBD Mode' menu, and then click.



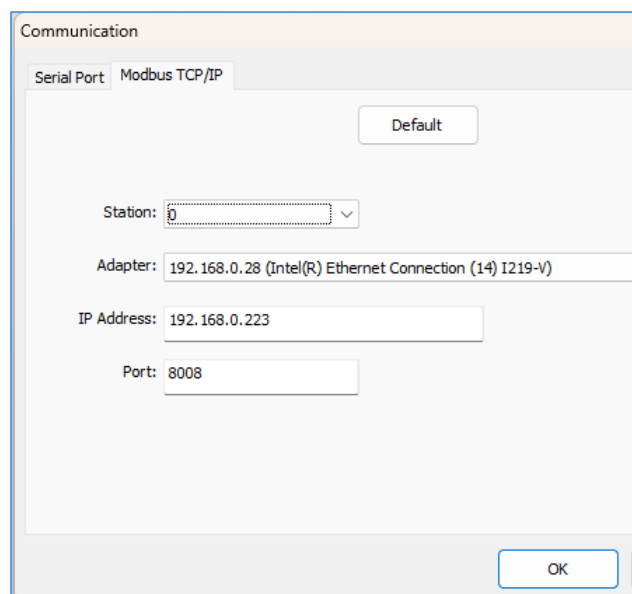
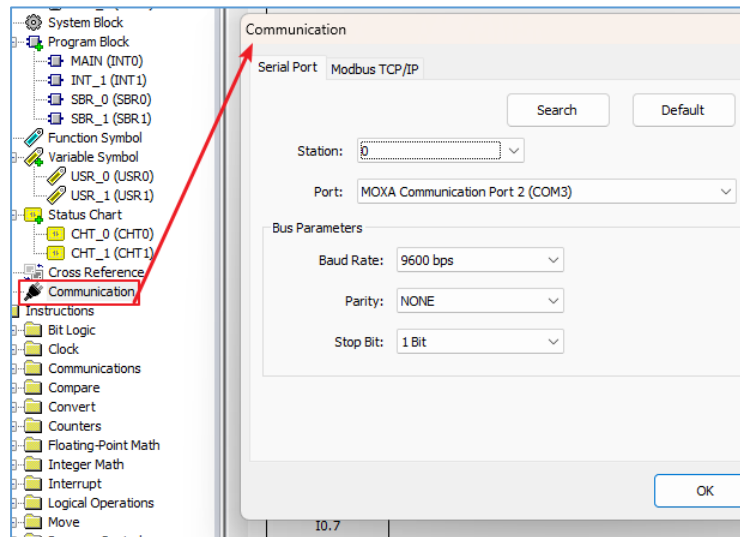
Then the following picture prompt will pop up.



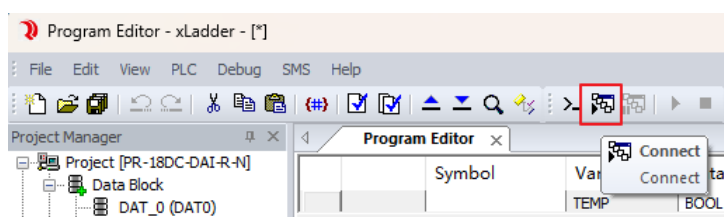
At this time, the PLC will automatically restart, and it will run in FBD mode after restarting.

10.2 How to obtain the firmware version number of the PLC

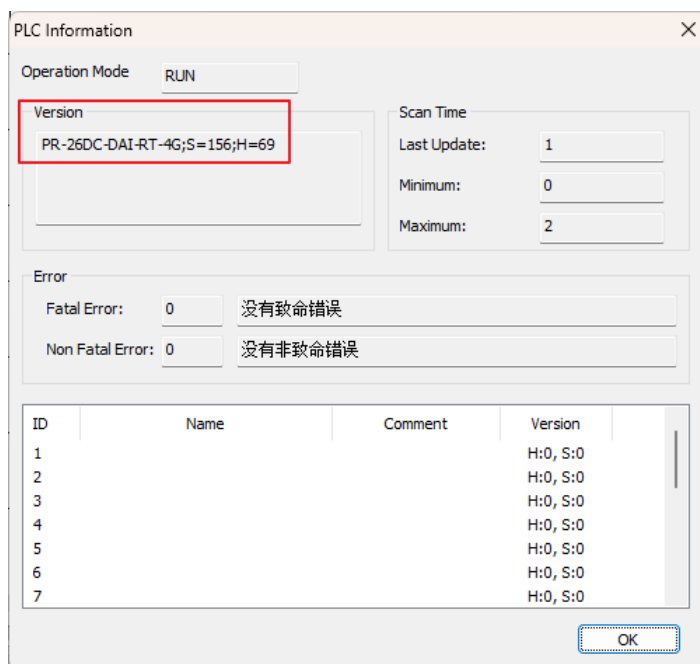
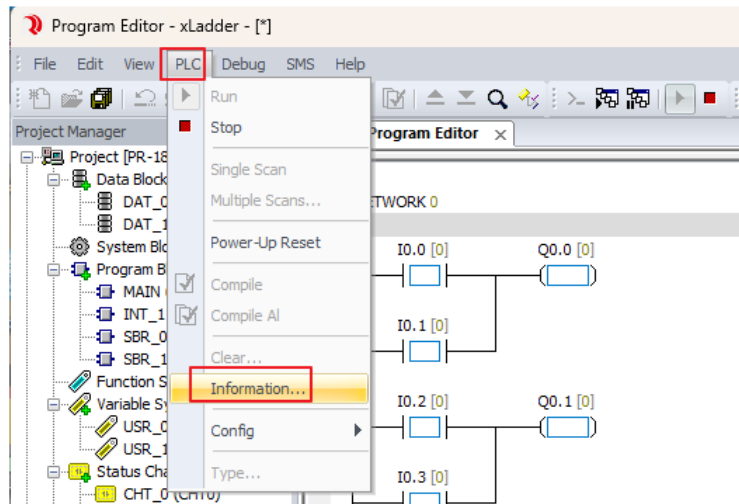
1. Configure the parameters in 'Communication' so that xLadder can communicate with PLC. xLadder can communicate with PLC via serial port or Ethernet.



2. Click the 'Connect' button in the toolbar.



3. Use the 'Information' menu under 'PLC' to read the firmware version number of the PLC in Ladder mode.



10.3 Value range of analog quantity:

0~10V → 0~1000

0~20mA → 0~1000

4~20mA → 0~1000

-50°C~200°C→-500~2000

When you use the PT100/PT1000 module, The value range of the analog quantity is -500~2000.It corresponds to the temperature that is -50°C~200°C.

10.4 Extension module address

10.4.1 Digital Input addressing

Divided into the following two situations:

1. Common I/O expansion, such as PR-E-16DC-DA-R^① .
2. Pure input expansion, such as PR-E-DC-16IN^② . The PR12N,18N,26N PLCs with firmware version number >=V150, the version of PR-18DC-DA-R >=10 supports these newly added addresses.
3. SR-12 supports 3 expansion modules

| | Input | FBD | LADDER | LAD MODBUS ADD. Modbus code: 02 |
|-------|-------------------|----------|--------------------------------|------------------------------------|
| CPU | | I1 -- I8 | I0.0 -- I0.7 | 0 -- 7 |
| | | I9 -- IG | I1.0 -- I1.7 | 8 -- 15 |
| EXT.1 | 8DI ^① | I1 -- I8 | I2.0 -- I2.7 | 16 -- 23 |
| | 16DI ^② | I1 -- I8 | I21.0 -- I21.7 or I2.0 -- I2.7 | 5800 - 5807 or 16 - 23 |
| | | I9 -- IG | I22.0 -- I22.7 | 5808 - 5815 |
| EXT.2 | 8DI | I1 -- I8 | I3.0 -- I3.7 | 24 -- 31 |
| | 16DI | I1 -- I8 | I23.0 -- I23.7 or I3.0 -- I3.7 | 5816 - 5823 or 24 - 31 |
| | | I9 -- IG | I24.0 -- I24.7 | 5824 - 5831 |
| EXT.3 | 8DI | I1 -- I8 | I4.0 -- I4.7 | 32 -- 39 |
| | 16DI | I1 -- I8 | I25.0 -- I25.7 or I4.0 -- I4.7 | 5832 - 5839 or 32 -- 39 |
| | | I9 -- IG | I26.0 -- I26.7 | 5840 - 5847 |
| EXT.4 | 8DI | I1 -- I8 | I5.0 -- I5.7 | 40 -- 47 |
| | 16DI | I1 -- I8 | I27.0 -- I27.7 or I5.0 -- I5.7 | 5848 - 5855 or 40 - 47 |
| | | I9 -- IG | I28.0 -- I28.7 | 5856 - 5863 |
| EXT.5 | 8DI | I1 -- I8 | I6.0 -- I6.7 | 48 -- 55 |
| | 16DI | I1 -- I8 | I29.0 -- I29.7 or I6.0 -- I6.7 | 5864 - 5871 or 48 - 55 |
| | | I9 -- IG | I30.0 -- I30.7 | 5872 - 5879 |
| EXT.6 | 8DI | I1 -- I8 | I7.0 -- I7.7 | 56 -- 63 |
| | 16DI | I1 -- I8 | I31.0 -- I31.7 or I7.0 -- I7.7 | 5880 - 5887 or 56 - 63 |
| | | I9 -- IG | I32.0 -- I32.7 | 5888 - 5895 |
| EXT.7 | 8DI | I1 -- I8 | I8.0 -- I8.7 | 64 -- 71 |
| | 16DI | I1 -- I8 | I33.0 -- I33.7 or I8.0 -- I8.7 | 5896 - 5903 or 64 - 71 |
| | | I9 -- IG | I34.0 -- I34.7 | 5904 - 5911 |
| EXT.8 | 8DI | I1 -- I8 | I9.0 -- I9.7 | 72 -- 79 |

| | | | | |
|--------|------|----------|----------------------------------|--------------------------|
| | 16DI | I1 – I8 | I35.0 -- I35.7 or I9.0 -- I9.7 | 5912 - 5919 or 72 - 79 |
| | | I9 -- IG | I36.0 -- I36.7 | 5920 - 5927 |
| EXT.9 | 8DI | I1 – I8 | I10.0 -- I10.7 | 80 -- 87 |
| | 16DI | I1 – I8 | I37.0 -- I37.7 or I10.0 -- I10.7 | 5928 - 5935 or 80 - 87 |
| | | I9 -- IG | I38.0 -- I38.7 | 5936 - 5943 |
| EXT.10 | 8DI | I1 – I8 | I11.0 -- I11.7 | 88 -- 95 |
| | 16DI | I1 – I8 | I39.0 -- I39.7 or I11.0 -- I11.7 | 5944 - 5951 or 88 - 95 |
| | | I9 -- IG | I40.0 -- I40.7 | 5952 - 5959 |
| EXT.11 | 8DI | I1 – I8 | I12.0 -- I12.7 | 96 -- 103 |
| | 16DI | I1 – I8 | I41.0 -- I41.7 or I12.0 -- I12.7 | 5960 - 5967 or 96 - 103 |
| | | I9 -- IG | I42.0 -- I42.7 | 5968 - 5975 |
| EXT.12 | 8DI | I1 – I8 | I13.0 -- I13.7 | 104 -- 111 |
| | 16DI | I1 – I8 | I43.0 -- I43.7 or I13.0 -- I13.7 | 5976 - 5983 or 104 - 111 |
| | | I9 -- IG | I44.0 -- I44.7 | 5984 - 5991 |
| EXT.13 | 8DI | I1 – I8 | I14.0 -- I14.7 | 112 -- 119 |
| | 16DI | I1 – I8 | I45.0 -- I45.7 or I14.0 -- I14.7 | 5992 - 5999 or 112 - 119 |
| | | I9 -- IG | I46.0 -- I46.7 | 6000 - 6007 |
| EXT.14 | 8DI | I1 – I8 | I15.0 -- I15.7 | 120 -- 127 |
| | 16DI | I1 – I8 | I47.0 -- I47.7 or I15.0 -- I15.7 | 6008 - 6015 or 120 - 127 |
| | | I9 -- IG | I48.0 -- I48.7 | 6016 - 6023 |
| EXT.15 | 8DI | I1 – I8 | I16.0 -- I16.7 | 128 -- 135 |
| | 16DI | I1 – I8 | I49.0 -- I49.7 or I16.0 -- I16.7 | 6024 - 6031 or 128 - 135 |
| | | I9 -- IG | I50.0 -- I50.7 | 6032 - 6039 |
| EXT.16 | 8DI | I1 – I8 | I17.0 -- I17.7 | 136 -- 143 |
| | 16DI | I1 – I8 | I51.0 -- I51.7 or I17.0 -- I17.7 | 6040 - 6047 or 136 - 143 |
| | | I9 -- IG | I52.0 -- I52.7 | 6048 - 6055 |

10.4.2 Digital Output addressing

Divided into the following two situations:

1. Common I/O expansion, such as PR-E-16DC-DA-R^① .
2. Pure output expansion, such as PR-E-DC-16DO^② . The PR12N,18N,26N PLCs with firmware version number $\geq V150$, the version of PR-18DC-DA-R ≥ 10 supports these newly added addresses.

3. SR-12 supports 3 expansion modules

| | Output | FBD | LADDER | LAD MODBUS ADD. Modbus code: 01/05(15) |
|--------|-------------------|----------|----------------|---|
| CPU | | Q1 -- Q8 | Q0.0 -- Q0.7 | 0 -- 7 |
| | | Q9 -- QA | Q1.0 -- Q1.1 | 8 -- 9 |
| EXT.1 | 8DO ^① | Q1 -- Q8 | Q2.0 -- Q2.7 | 16 -- 23 |
| | 16DO ^② | Q1 -- Q8 | Q2.0 -- Q2.7 | 16 -- 23 |
| | | Q9 -- QG | Q21.0 -- Q21.7 | 168 -- 175 |
| EXT.2 | 8DO | Q1 -- Q8 | Q3.0 -- Q3.7 | 24 -- 31 |
| | 16DO | Q1 -- Q8 | Q3.0 -- Q3.7 | 24 -- 31 |
| | | Q9 -- QG | Q22.0 -- Q22.7 | 176 -- 183 |
| EXT.3 | 8DO | Q1 -- Q8 | Q4.0 -- Q4.7 | 32 -- 39 |
| | 16DO | Q1 -- Q8 | Q4.0 -- Q4.7 | 32 -- 39 |
| | | Q9 -- QG | Q23.0 -- Q23.7 | 184 -- 191 |
| EXT.4 | 8DO | Q1 -- Q8 | Q5.0 -- Q5.7 | 40 -- 47 |
| | 16DO | Q1 -- Q8 | Q5.0 -- Q5.7 | 40 -- 47 |
| | | Q9 -- QG | Q24.0 -- Q24.7 | 192 -- 199 |
| EXT.5 | 8DO | Q1 -- Q8 | Q6.0 -- Q6.7 | 48 -- 55 |
| | 16DO | Q1 -- Q8 | Q6.0 -- Q6.7 | 48 -- 55 |
| | | Q9 -- QG | Q25.0 -- Q25.7 | 200 -- 207 |
| EXT.6 | 8DO | Q1 -- Q8 | Q7.0 -- Q7.7 | 56 -- 63 |
| | 16DO | Q1 -- Q8 | Q7.0 -- Q7.7 | 56 -- 63 |
| | | Q9 -- QG | Q26.0 -- Q26.7 | 208 -- 215 |
| EXT.7 | 8DO | Q1 -- Q8 | Q8.0 -- Q8.7 | 64 -- 71 |
| | 16DO | Q1 -- Q8 | Q8.0 -- Q8.7 | 64 -- 71 |
| | | Q9 -- QG | Q27.0 -- Q27.7 | 216 -- 223 |
| EXT.8 | 8DO | Q1 -- Q8 | Q9.0 -- Q9.7 | 72 -- 79 |
| | 16DO | Q1 -- Q8 | Q9.0 -- Q9.7 | 72 -- 79 |
| | | Q9 -- QG | Q28.0 -- Q28.7 | 224 -- 231 |
| EXT.9 | 8DO | Q1 -- Q8 | Q10.0 -- Q10.7 | 80 -- 87 |
| | 16DO | Q1 -- Q8 | Q10.0 -- Q10.7 | 80 -- 87 |
| | | Q9 -- QG | Q29.0 -- Q29.7 | 232 -- 239 |
| EXT.10 | 8DO | Q1 -- Q8 | Q11.0 -- Q11.7 | 88 -- 95 |
| | 16DO | Q1 -- Q8 | Q11.0 -- Q11.7 | 88 -- 95 |

| | | | | |
|--------|------|----------|----------------|------------|
| | | Q9 -- QG | Q30.0 -- Q30.7 | 240 -- 247 |
| EXT.11 | 8DO | Q1 -- Q8 | Q12.0 -- Q12.7 | 96 -- 103 |
| | 16DO | Q1 -- Q8 | Q12.0 -- Q12.7 | 96 -- 103 |
| | | Q9 -- QG | Q31.0 -- Q31.7 | 248 -- 255 |
| EXT.12 | 8DO | Q1 -- Q8 | Q13.0 -- Q13.7 | 104 -- 111 |
| | 16DO | Q1 -- Q8 | Q13.0 -- Q13.7 | 104 -- 111 |
| | | Q9 -- QG | Q32.0 -- Q32.7 | 256 -- 263 |
| EXT.13 | 8DO | Q1 -- Q8 | Q14.0 -- Q14.7 | 112 -- 119 |
| | 16DO | Q1 -- Q8 | Q14.0 -- Q14.7 | 112 -- 119 |
| | | Q9 -- QG | Q33.0 -- Q33.7 | 264 -- 271 |
| EXT.14 | 8DO | Q1 -- Q8 | Q15.0 -- Q15.7 | 120 -- 127 |
| | 16DO | Q1 -- Q8 | Q15.0 -- Q15.7 | 120 -- 127 |
| | | Q9 -- QG | Q34.0 -- Q34.7 | 272 -- 279 |
| EXT.15 | 8DO | Q1 -- Q8 | Q16.0 -- Q16.7 | 128 -- 135 |
| | 16DO | Q1 -- Q8 | Q16.0 -- Q16.7 | 128 -- 135 |
| | | Q9 -- QG | Q35.0 -- Q35.7 | 280 -- 287 |
| EXT.16 | 8DO | Q1 -- Q8 | Q17.0 -- Q17.7 | 136 -- 143 |
| | 16DO | Q1 -- Q8 | Q17.0 -- Q17.7 | 136 -- 143 |
| | | Q9 -- QG | Q36.0 -- Q36.7 | 288 -- 295 |

10.4.3 Analog Input addressing

1)PR14,18,24 Series

CPU: AI1~AI6 → AIW0~AIW10

EXT1: AI1~AI4 → AIW20~AIW26

•

EXT16: AI1~AI4 → AIW170~AIW176

Calculation formula:

AI Start Address=AIW (Extension Address × 10+10)

2)SR12,PR12N,18N Series

CPU: AI1~AI8 → AIW0~AIW14

EXT1: AI1~AI4 → AIW20~AIW26

EXT2: AI1~AI4 → AIW36~AIW42

EXT3: AI1~AI4 → AIW52~AIW58

EXT4: AI1~AI4 → AIW68~AIW74
EXT5: AI1~AI4 → AIW84~AIW90
EXT6: AI1~AI4 → AIW100~AIW106
EXT7: AI1~AI4 → AIW116~AIW122
EXT8: AI1~AI4 → AIW132~AIW138
EXT9: AI1~AI4 → AIW148~AIW154
EXT10: AI1~AI4 → AIW164~AIW170
EXT11: AI1~AI4 → AIW180~AIW186
EXT12: AI1~AI4 → AIW196~AIW202
EXT13: AI1~AI4 → AIW212~AIW218
EXT14: AI1~AI4 → AIW228~AIW234
EXT15: AI1~AI4 → AIW244~AIW250
EXT16: AI1~AI4 → AIW260~AIW266

*SR-12 supports 3 expansion modules

3)PR26N,23N Series

CPU: AI1~AI8 → AIW0~AIW14
 AID~AIG → AIW24~AIW30
EXT1: AI1~AI4 → AIW32~AIW38
EXT2: AI1~AI4 → AIW48~AIW54
EXT3: AI1~AI4 → AIW64~AIW70
EXT4: AI1~AI4 → AIW80~AIW86
EXT5: AI1~AI4 → AIW96~AIW102
EXT6: AI1~AI4 → AIW112~AIW118
EXT7: AI1~AI4 → AIW128~AIW134
EXT8: AI1~AI4 → AIW144~AIW150
EXT9: AI1~AI4 → AIW160~AIW166
EXT10: AI1~AI4 → AIW176~AIW182
EXT11: AI1~AI4 → AIW192~AIW198
EXT12: AI1~AI4 → AIW208~AIW214
EXT13: AI1~AI4 → AIW224~AIW230
EXT14: AI1~AI4 → AIW240~AIW246
EXT15: AI1~AI4 → AIW256~AIW262
EXT16: AI1~AI4 → AIW272~AIW278

10.4.4 Analog Output addressing

1)PR14,18,24 Series

PR24CPU: AQ1 → AQW0

EXT1: AQ1,AQ2 → AQW20,AQW22

•

EXT16: AQ1,AQ2 → AQW170,AQW172

Calculation formula:

AQ Start Address=AQW (Extension Address × 10+10)

2)SR12,PR12N,18N,26N,23N Series

EXT1: AQ1,AQ2 → AQW4,AQW6

EXT2: AQ1,AQ2 → AQW8,AQW10

•

EXT16: AQ1,AQ2 → AQW64,AQW66

Calculation formula:

AQ Start Address=AQW (Extension Address × 4)

*SR-12 supports 3 expansion modules

10.5 Set extension module address with a DIP switch



The address of extension module = The value of DIP switch + 1

The default address of PR/SR-E expansions is 1 and the dip switches are set as below (■ means the switch position):

Address 1:



Address 2:



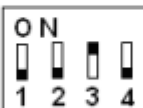
Address 3:



Address 4:



Address 5:



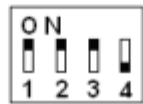
Address 6:



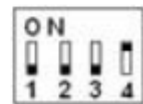
Address 7:



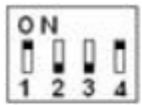
Address 8:



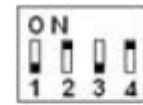
Address 9:



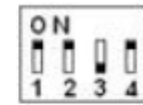
Address 10:



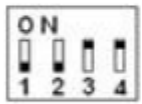
Address 11:



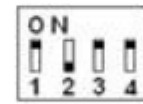
Address 12:



Address 13:



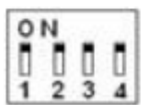
Address 14:



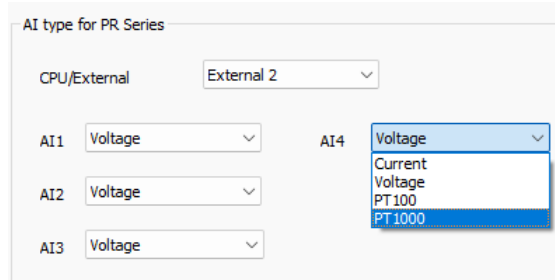
Address 15:



Address 16:



select the working mode of the corresponding analog input channel.

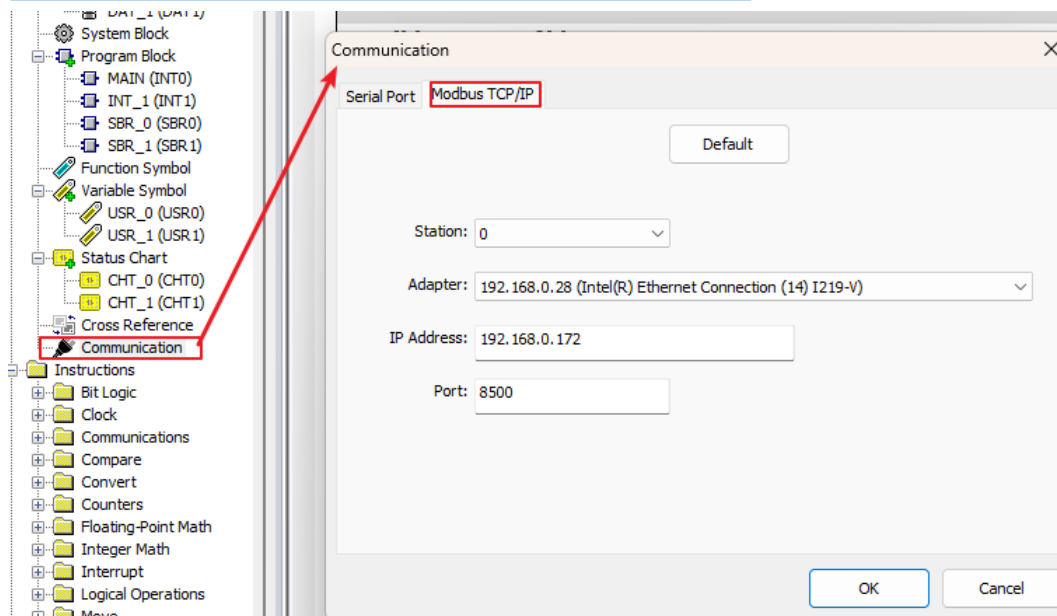
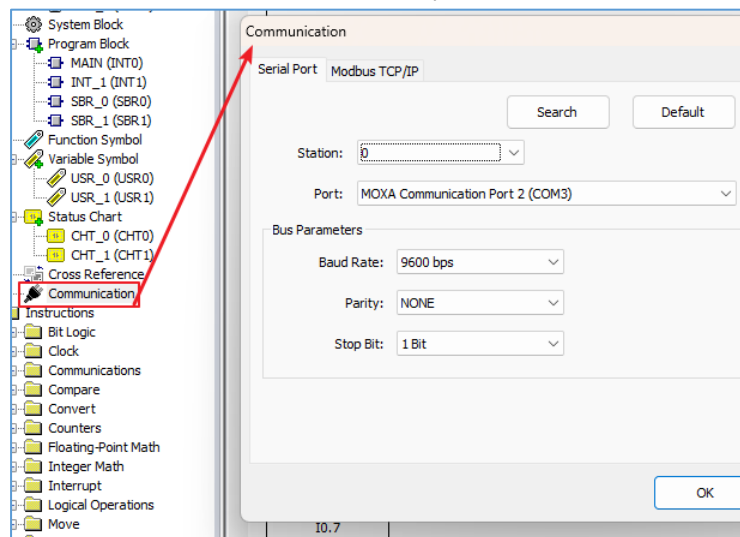


After setting, click the 'OK' button. Then download these settings to the PLC with the program, and the new settings will take effect after the PLC restarts.

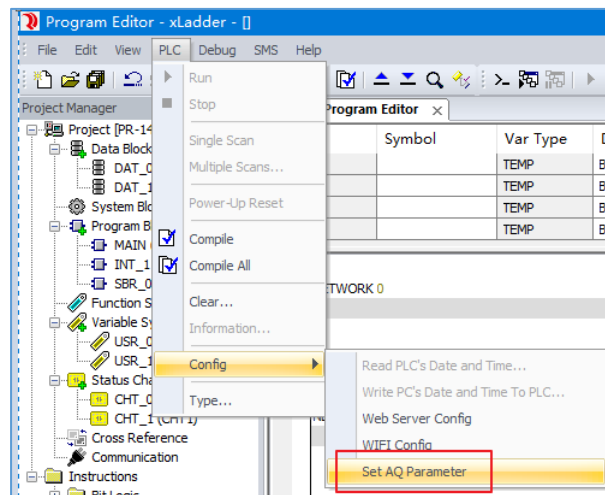
10.6.2 AQ: Set to voltage or current mode

For models such as PR-E-AQ-VI and PR-26DC-DAI-RA-N, their AQ can be set to output voltage signals (0~10V) or current signals (0/4~20mA.)

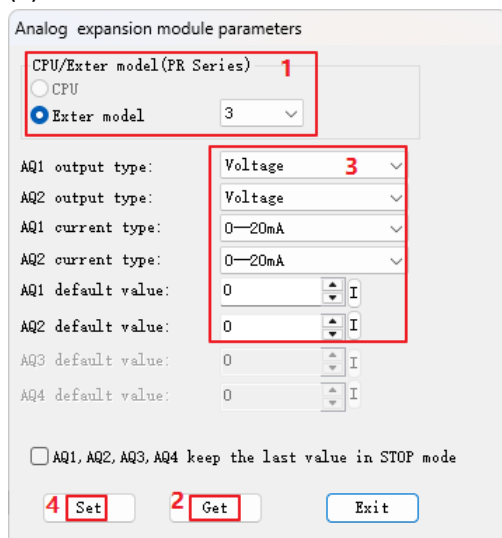
(1) Configure the parameters in 'Communication' so that xLadder can communicate with PLC.
xLadder can connect to PLC via serial port or Ethernet.



(2) Use the 'Set AQ Parameter' menu.



(3) Get or set the AQ mode

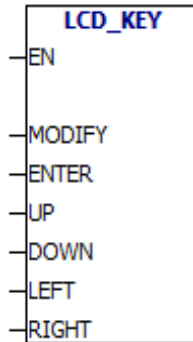


After successful setting, it will take effect immediately.

10.7 Additional instructions

10.7.1 LCD related instructions

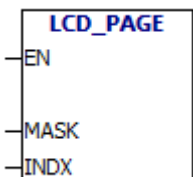
LCD_KEY



LCD_KEY binds LCD keys and PLC variables.

- EN: Enable
- MODIFY: Modifies the corresponding variables.
- ENTER: Confirms the corresponding variables.
- UP: The corresponding variable of UP key.
- DOWN: The corresponding variable of DOWN key.
- LEFT: The corresponding variable of LEFT key.
- RIGHT: The corresponding variable of RIGHT key.

LCD_PAGE



LCD_PAGE instruction binds the LCD display page.

- EN: Enable
- MASK: The current page group mask, generally 1.
- INDX: Currently displayed page number. You can modify the page number, the LCD will display the page.

Supplementary explanation: MASK input is a byte. Take VB0 as an example:

VB0

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

When 0 bit is equal to 1, LCD will display 0 group.

When 1 bit is equal to 1, LCD will display 1 group.

When 2 bit is equal to 1, LCD will display 2 group.

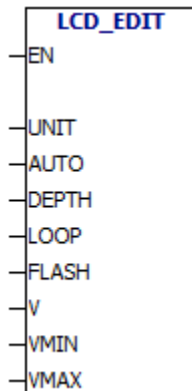
.

.

.

When 7 bit is equal to 1, LCD will display 7 group.

LCD_EDIT



LCD_EDIT: Binds the PLC variable to the edit state of the LCD.

- EN: Enable
- UNIT: Edit the number of objects in the page
- AUTO: Whether uses LCD keys to edit.
- DEPTH: The current edit depth of edit object.
- LOOP: LOOP edit.
- FLASH: The edit object is flashing or not.
- V: The current value of edit object.
- VMIN: The minimum value of edit object.
- VMAX: The maximum value of edit object.

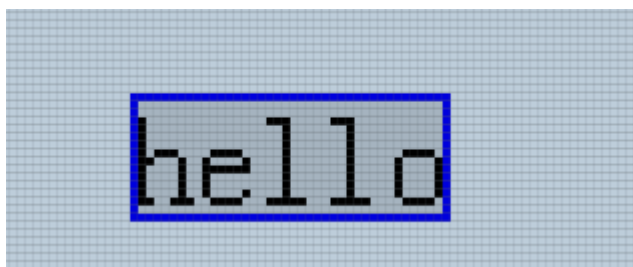
For example:

You have to edit display pages in LCD software.

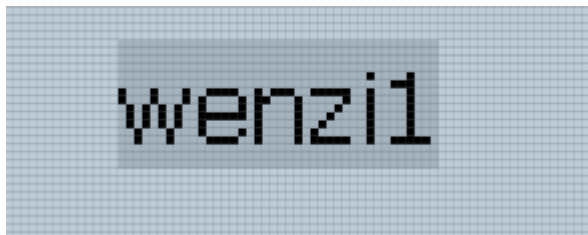
Display page 1:



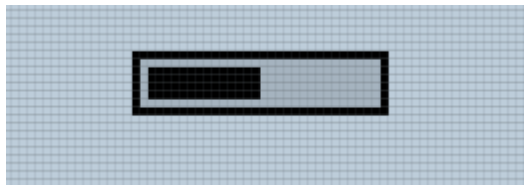
Display page 2:



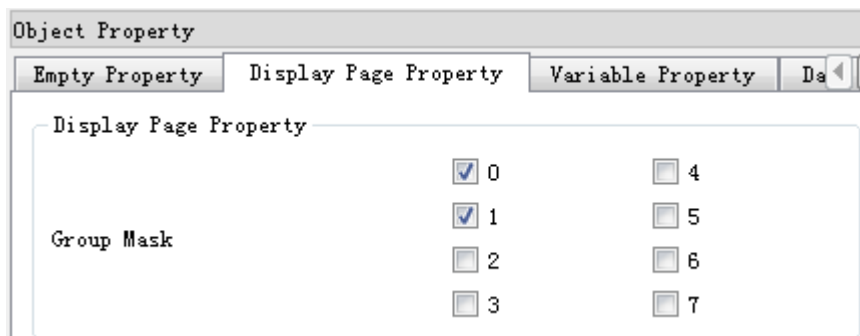
Display page 3:



Display page 4:



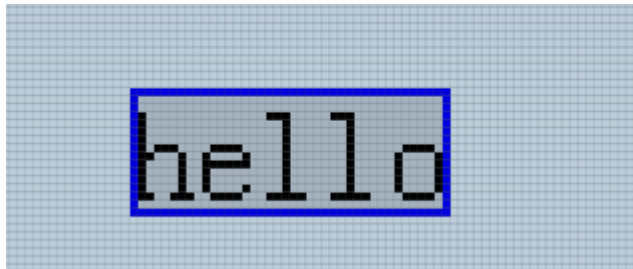
Grouping of display pages in display page property:



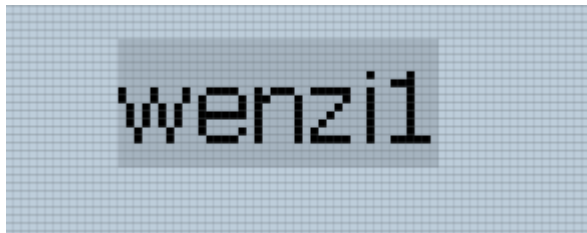
Display page 1: Display page 1 is divided into 0 group and 1 group.



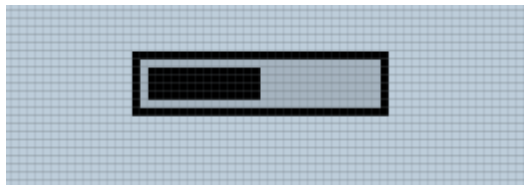
Display page 2: Display page 2 is divided into 0 group and 1 group.



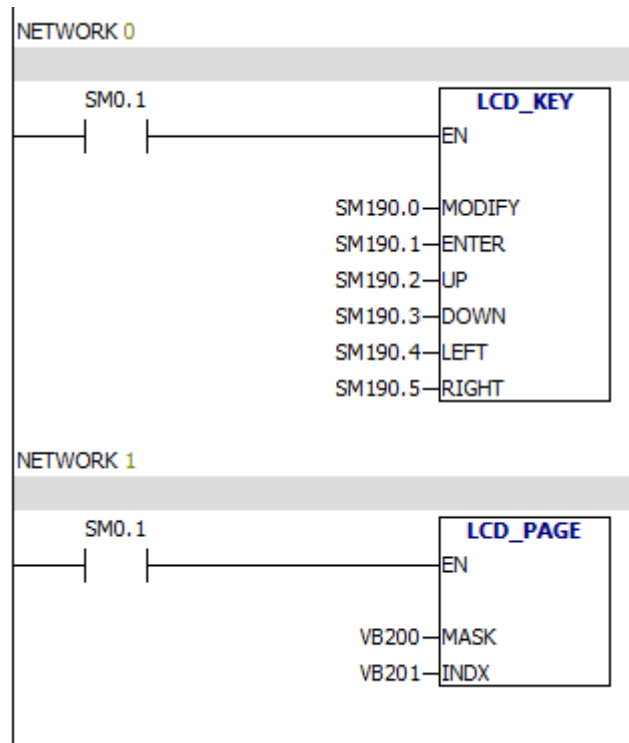
Display page 3: Display page 3 is divided into 0 group.

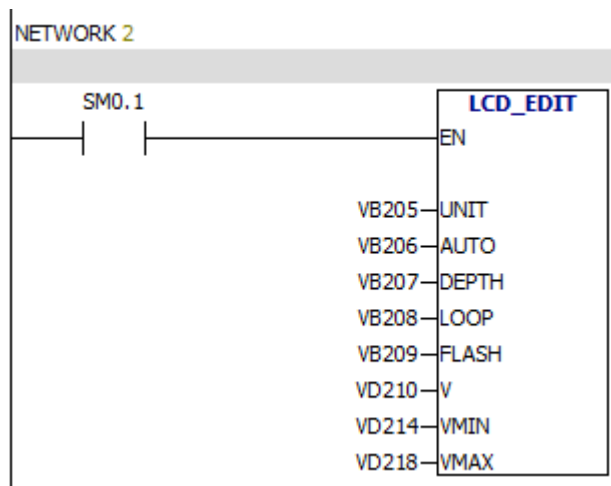


Display page 4: Display page 4 is divided into 0 group.



PLC program:





Analysis:

In network 0, the program binds LCD keys and PLC variables.

PLC has ten function keys. Each function key corresponds to a PLC variable.

F1 corresponds to SM191.0

F2 corresponds to SM191.1

F3 corresponds to SM191.2

F4 corresponds to SM191.3

ESC corresponds to SM190.0

OK corresponds to SM190.1

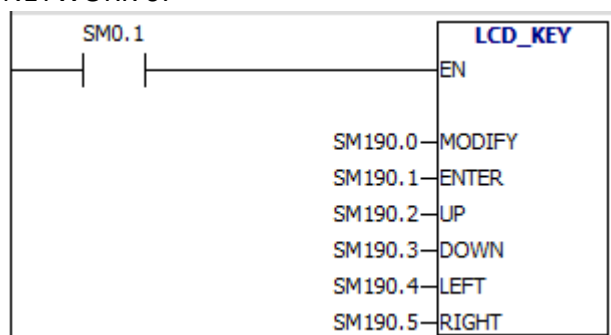
UP corresponds to SM190.2

DOWN corresponds to SM190.3

LEFT corresponds to SM190.4

RIGHT corresponds to SM190.5

NETWORK 0:



MODIFY is ESC function key, corresponds to SM190.0

The functions of function keys:

You can customize F1~F4.

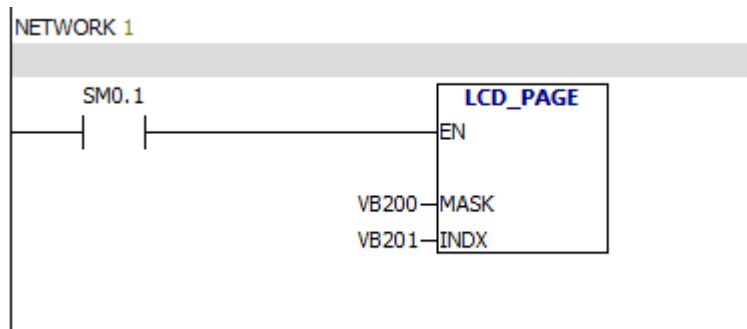
ESC is used for modifying values and exiting edit.

OK is used for confirming modified values.

UP and DOWN function keys can toggle display page. They can also increase or decrease values.

LEFT and RIGHT function keys can be used for toggling edit objects.

The function of NETWORK1 is binding PLC variables and LCD pages.



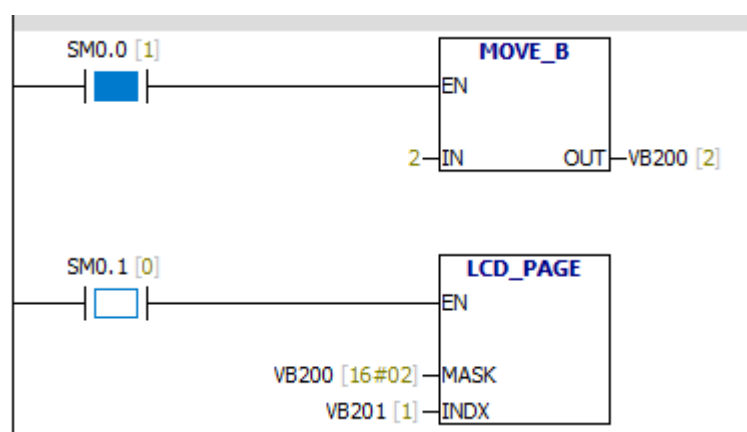
Operation result:

| | Address | Data Type | Value |
|--|---------|-----------|-------|
| | VB200 | BYTE | 16#01 |
| | VB201 | SINT | 0 |

The display page 1 of the 0 group is displayed by default. The value of VB200 is 1.0 bit is equal to 1, so LCD displays 0 group. The value of VB201 is 0, which means the first display page. The first display page is displaying page 1.

You can use the program to specify the display group and the display page.

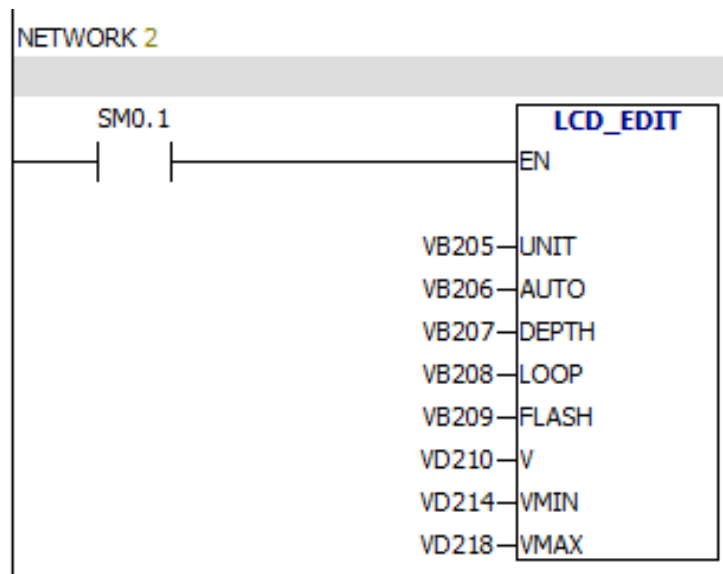
For example:



The LCD will display group 1 and display page2.

If you use LCD function keys to toggle the display pages, the value of VB201 will change.

NETWORK2:



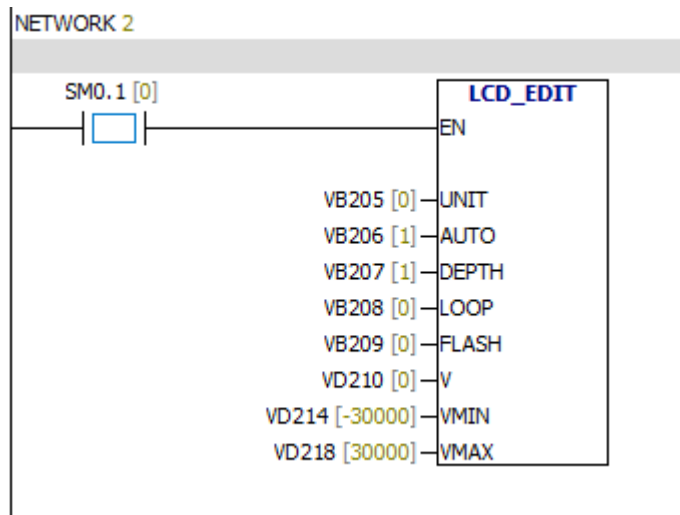
LCD -EDIT instruction binds the PLC variables and LCD edit states.

For example:

When you modify the first variable of display page 1:



The instruction will display as follows:



Vb205 = 0 Variable 0,the first variable.

Vb206 = 1 It means that you can use LCD function keys to edit variables.

Vb207 = 1 It means you can modify single digit.Vb207 = 2, you can modify single digit and tens digit.

Vb208 = 0 No loop

Vb209 = 0 No flicker

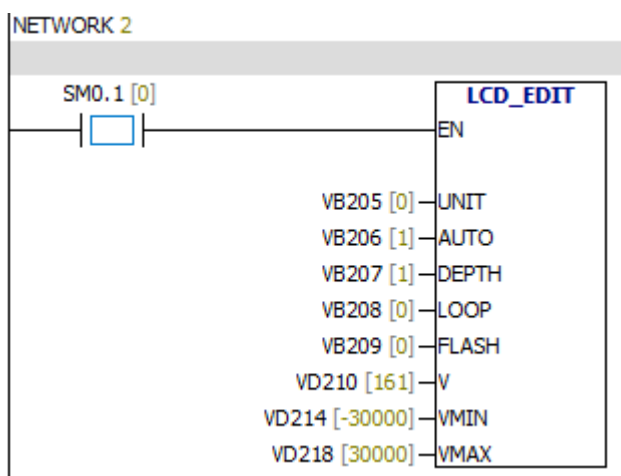
Vd210 = 0 The current value of variable is 0

Vd214 = -30000 The minimum value is -30000

Vd218 = 30000 The maximum value is 30000

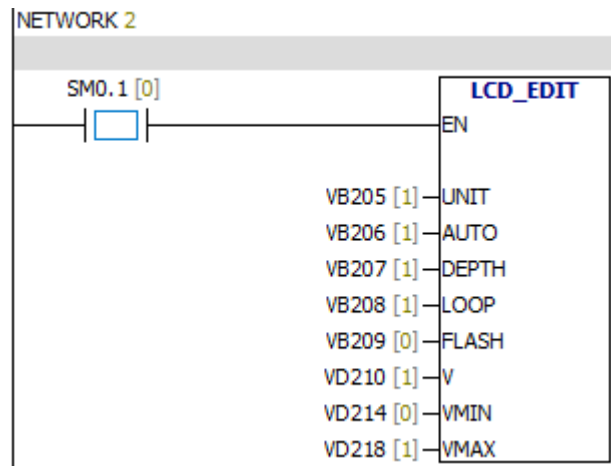
For example:

Modify the value of variable to 161.



For example:

Modify text list



VB205 Variable 1, the second variable.

VB206 It means that you can use LCD function keys to edit variables.

VB207 Edit depth is 1.

VB208 LOOP

VB209 No flicker

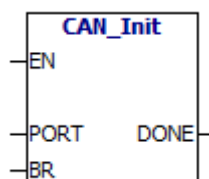
VD210 The current value of variable is 1

VD214 The minimum value is 0

VD218 The maximum value is 1

10.7.2 CAN, serial port initialization instructions

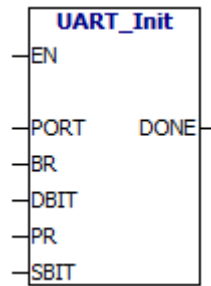
UART_Init CAN_Init



CAN_Init instruction is used to initialize the CAN port.

- EN: If the input value is 1, the instruction will initialize the CAN port.
- PORT: port number, 0~1。
- BR: CAN port baud rate

UART_Init

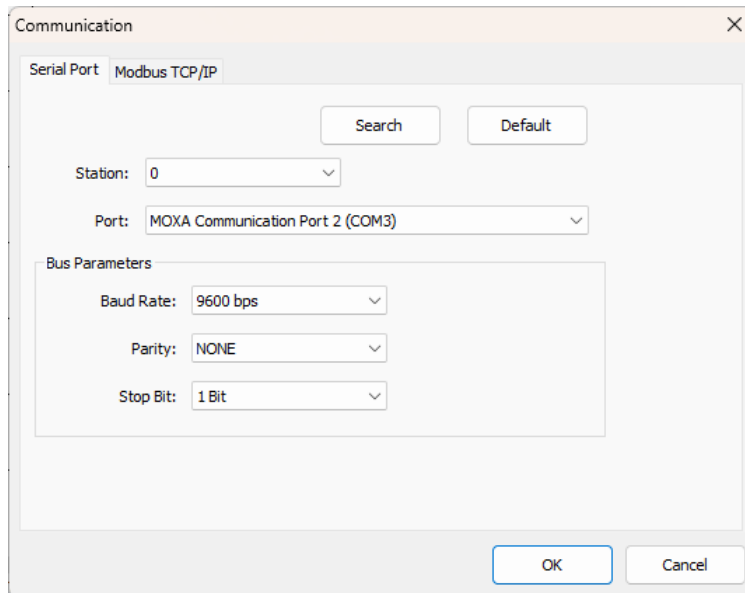


UART_Init instruction is used to initialize the serial port.

- EN: If the input value is equal to 1, the instruction will initialize the serial port.
- PORT: port number, 0 - 2。
- BR: Serial port baud rate.
- DBIT: The number of serial data bit.
- PR: Serial port check bit, 0=No parity, 1=Odd check, 2=Parity check
- SBIT: Stop bit
- DONE: success=1, fail=0

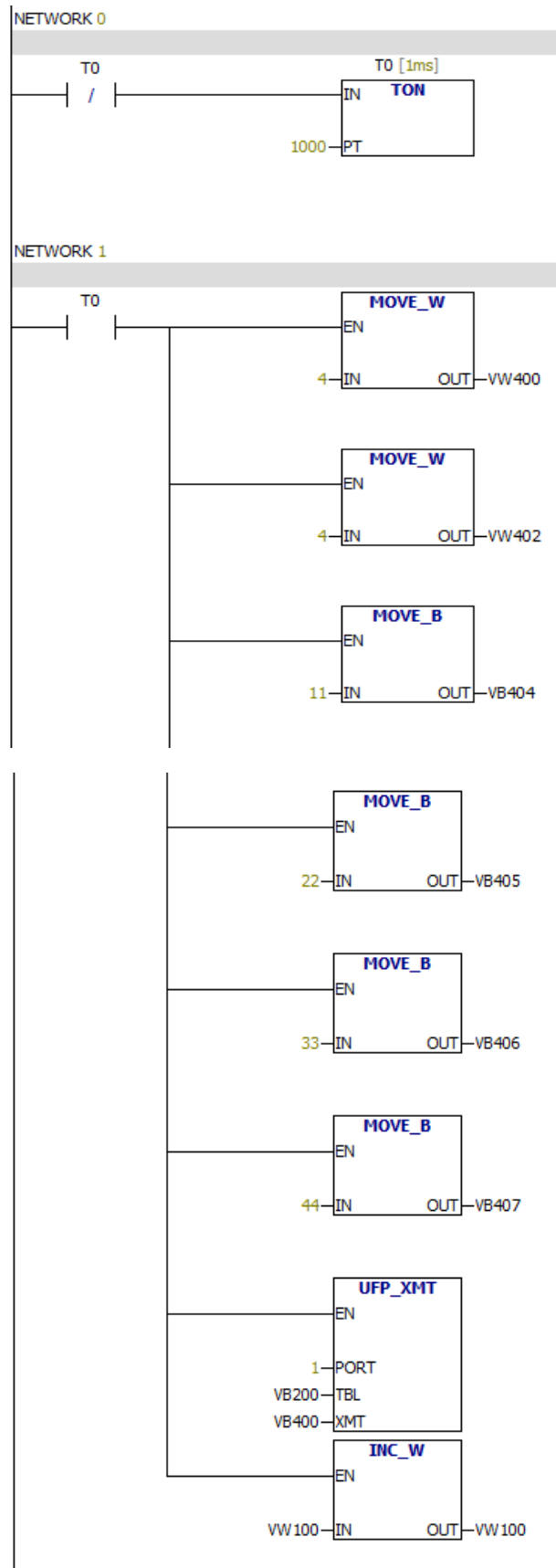
You can also set these parameters in the programming software.

As shown in the following picture:



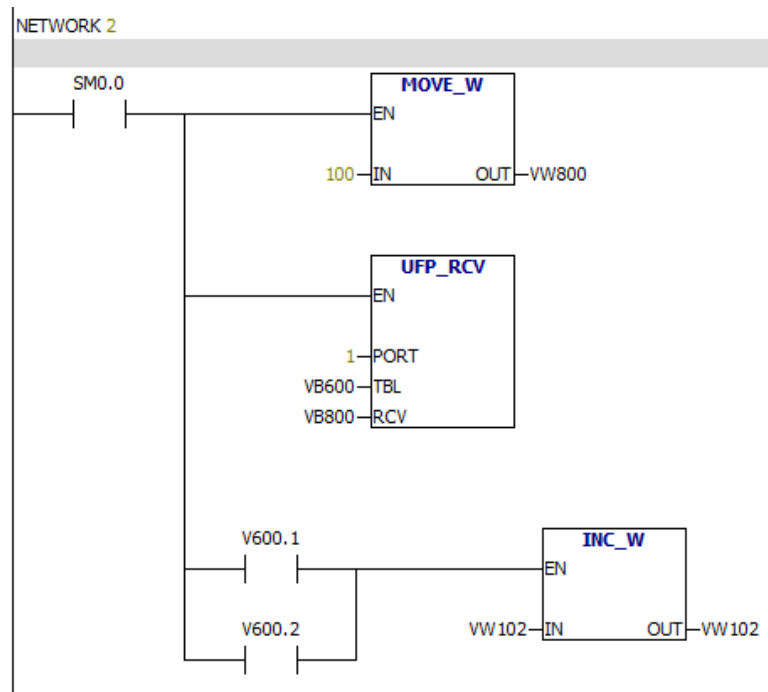
10.8 Example of serial port free port communication

Program 1:



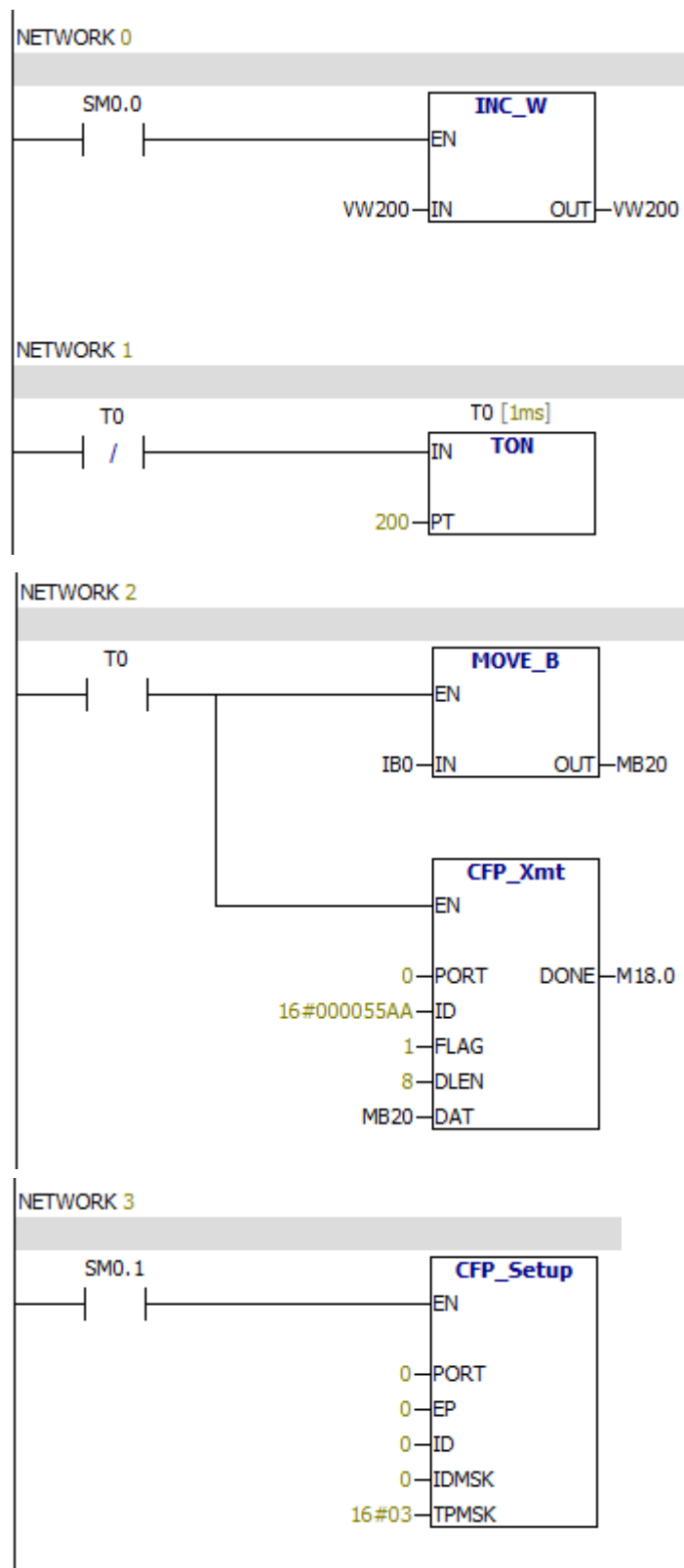
Explain: Send data 11 22 33 44 per second through port 1. And record the counts of sending data.

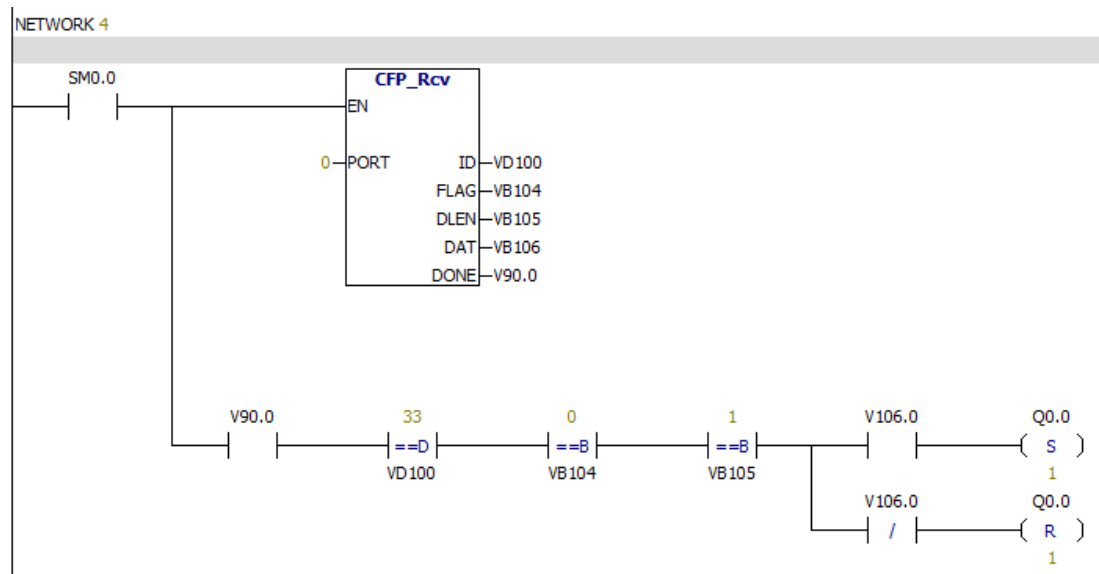
Program 2:



Explain: Receive data through port 1. The maximum length of the data is 100 bytes.

10.9 Example of CAN free port

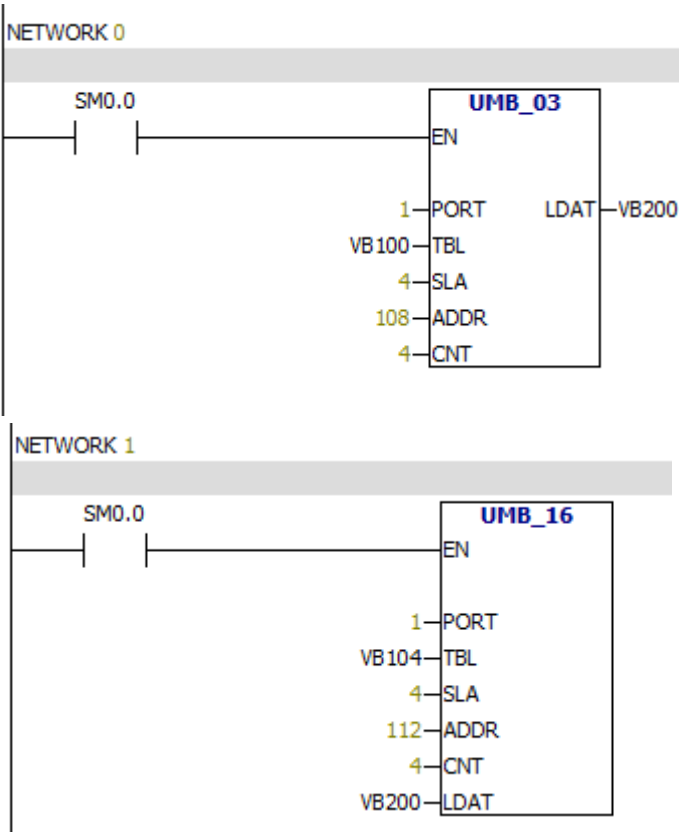




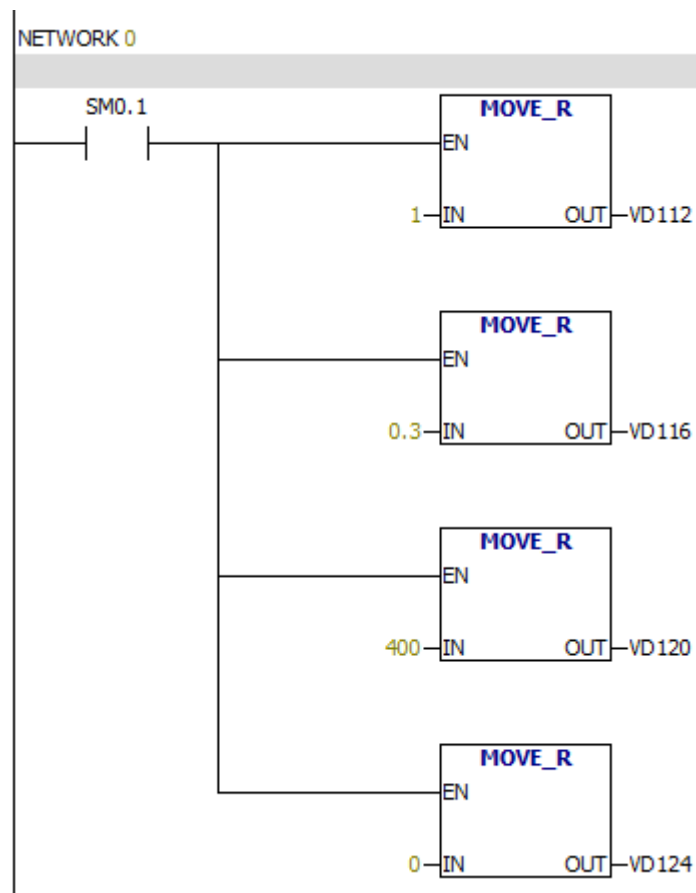
Explain: The state of the I0.0 is transmitted through port 0. Receive data VB106 through port0. The state of the first bit of VB106 is the state of Q0.0

10.10 MODBUS communication master program

Read multiple hold registers and write multiple hold registers



10.11 The example of using PID instruction



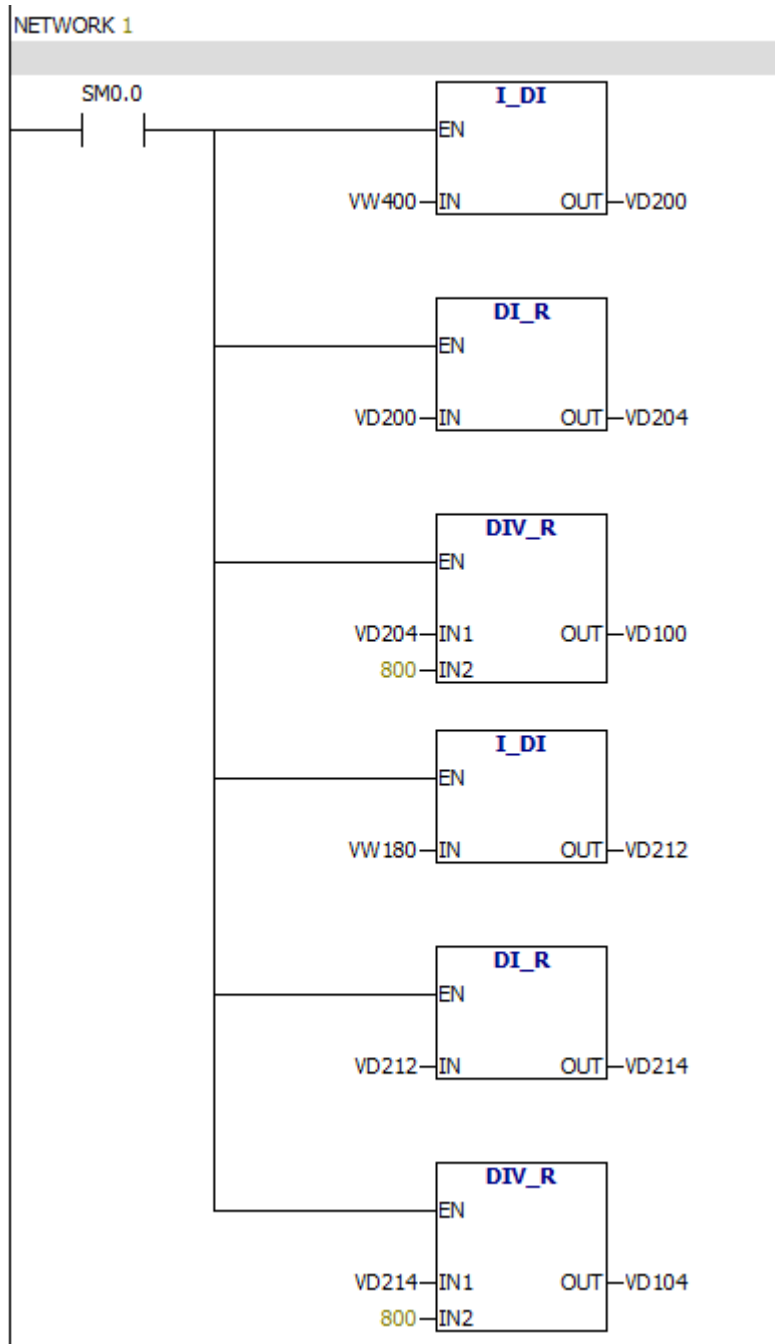
The initialization of PID parameters

VD112 gain

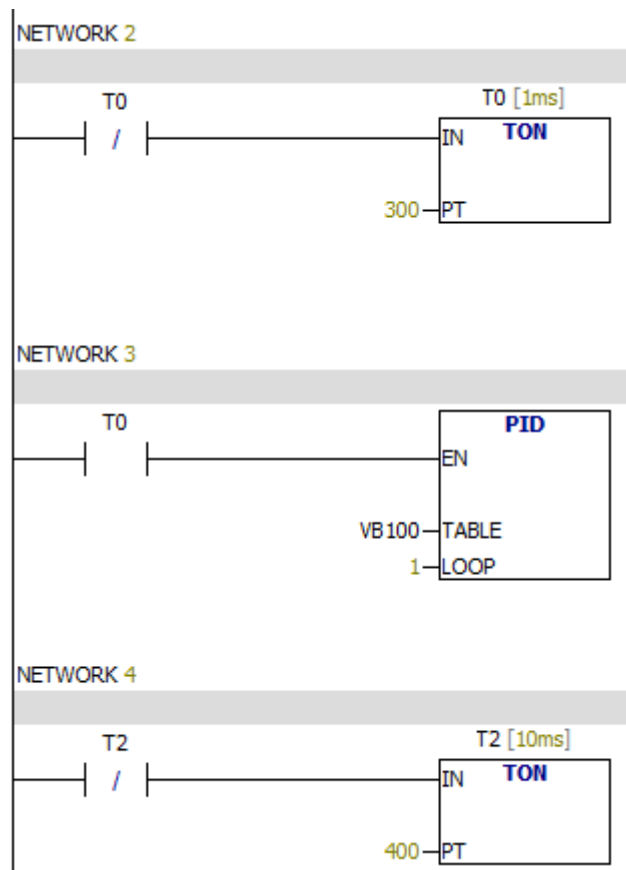
VD116 Sampling time

VD120 Integral time

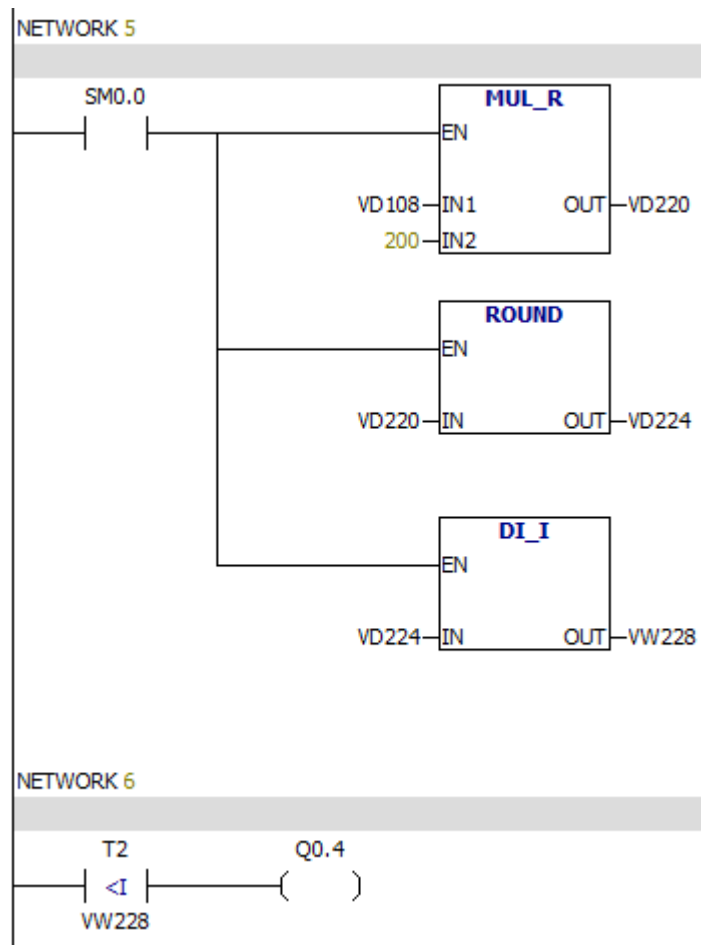
VD124 Differential time



Conversion of Process quantity and set value unit



Call a PID command every 0.3 seconds.



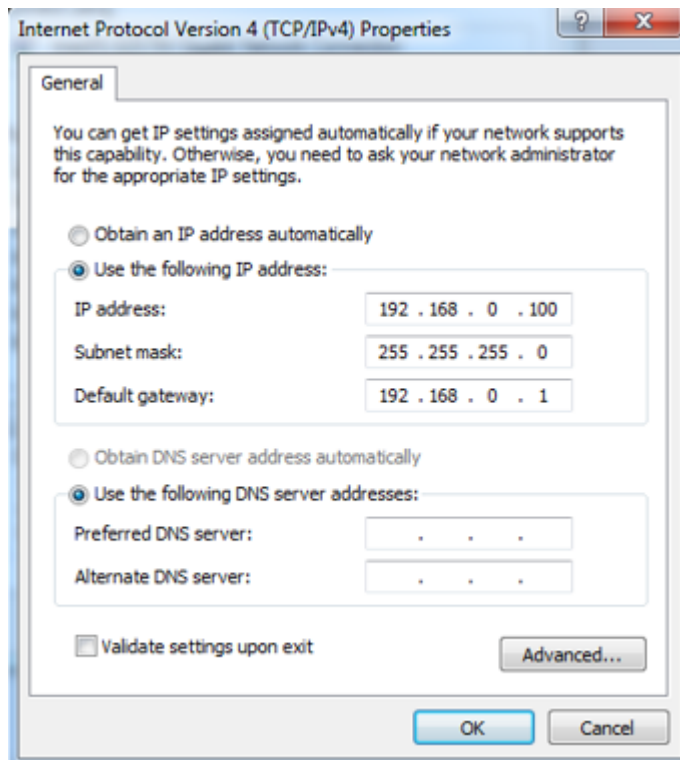
The conversion of output value unit.

10.12 Read / Write Program via Ethernet

First, connect the PLC to the LAN where the PC is located. And belong to the same network segment. The PC and PLC can be connected via a switch/router, or the two can be connected directly via a network cable.

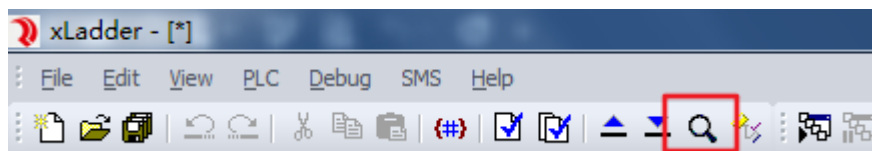
When the PLC and PC are connected via a network cable, both the PLC and PC need to be set to fixed IP addresses in the same LAN.

The following figure shows the IP address of the PC:



10.12.1 PR Ethernet Series

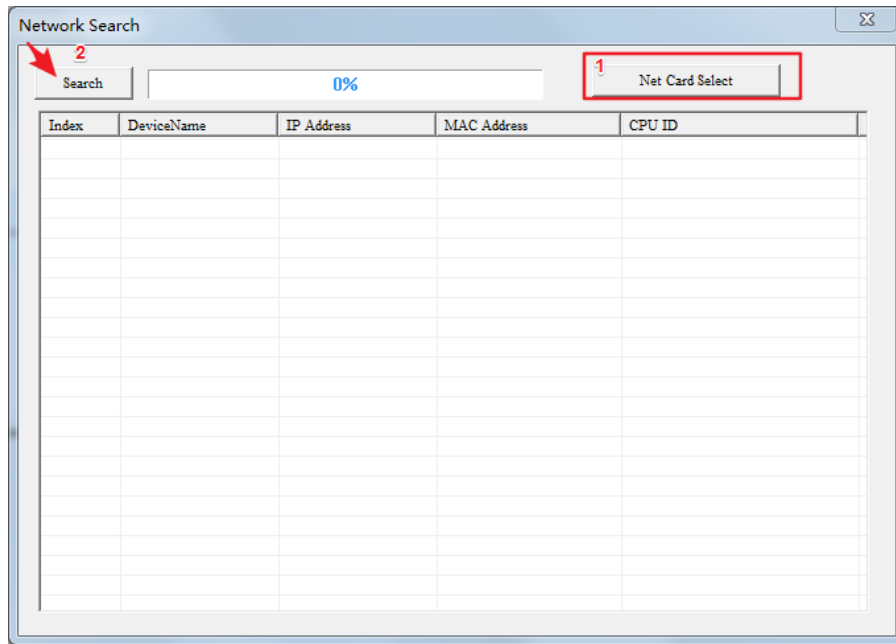
(a) Configure IP parameters without USB CABLE:



(1) Click the button

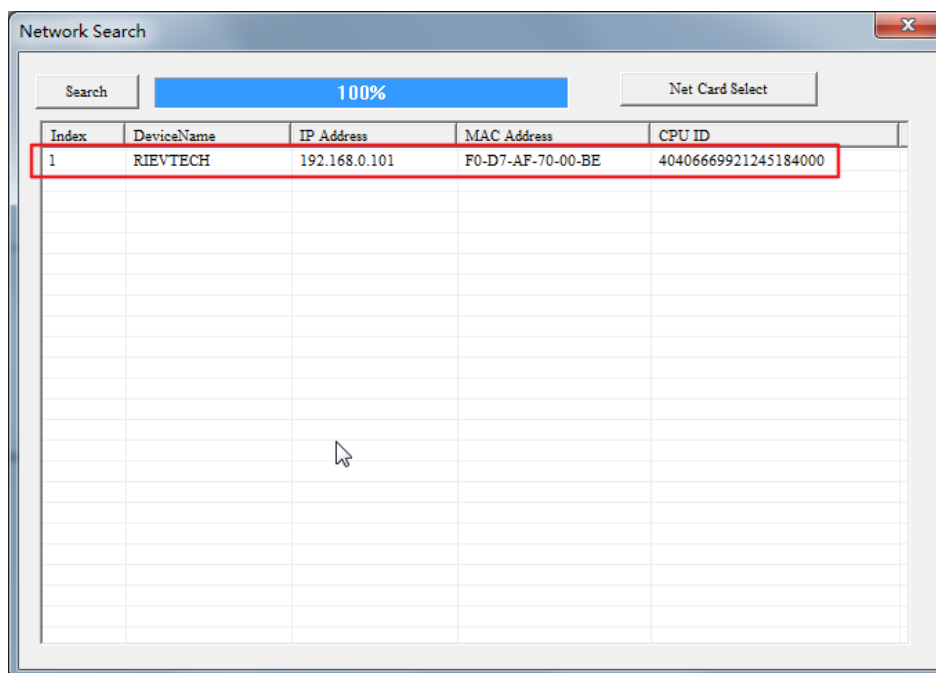


The following window will pop up:

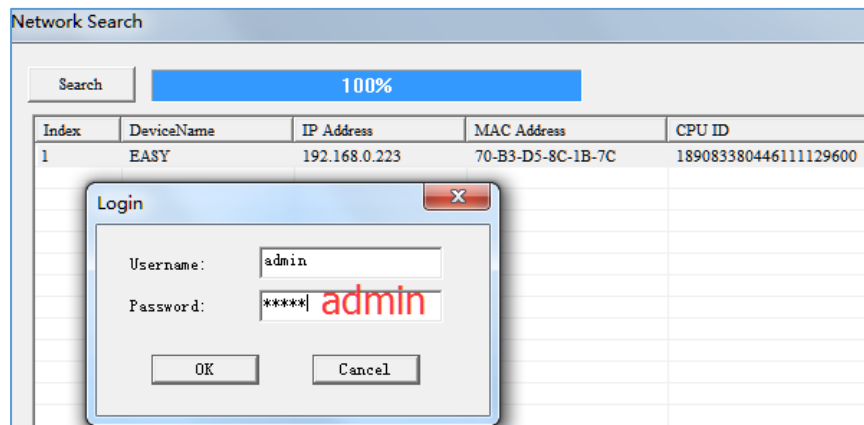


- 1) Click the "Net Card Select" button to select the network card on the PC that is connected to the PLC. Failure to select the correct network card will cause no PLC to be searched later.
- 2) Click the 'search' button and the following figure identifies that the Ethernet PLC has been searched.

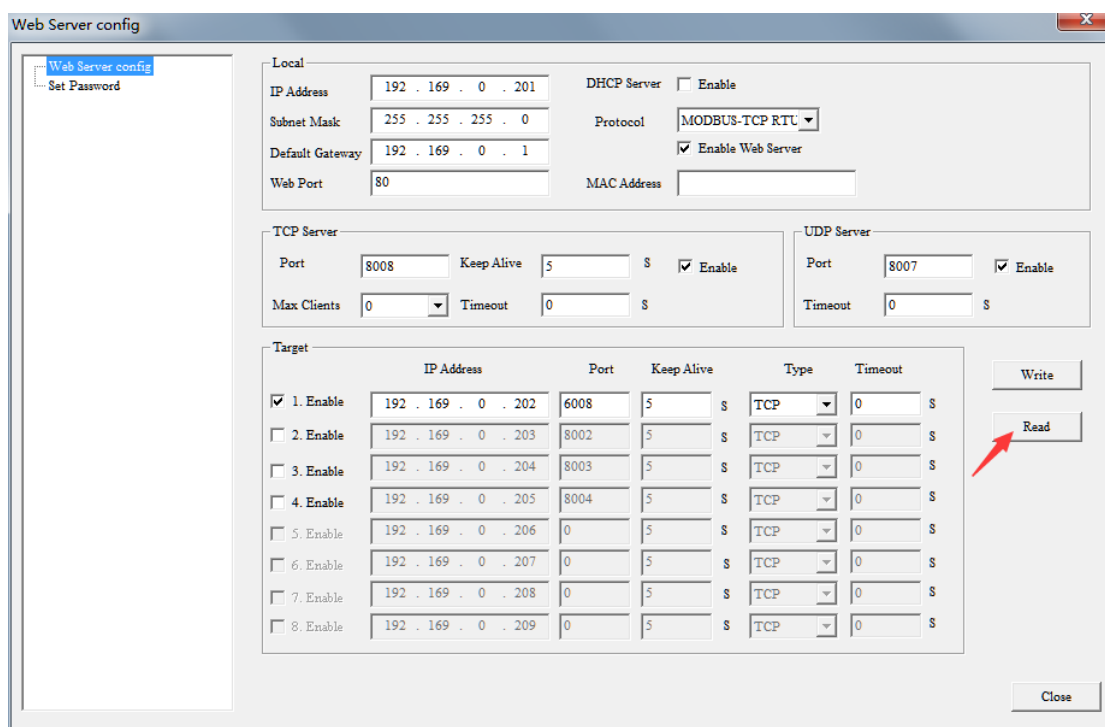
The default IP of the PR-12DC-DA-R-N and other models is 192.168.0.245.



Double-click on the IP in the above figure, the login screen will pop up. Username and Password are both admin. See below:



When you are done filling out, click the 'OK' button. Wait a little while, the IP configuration window will pop up – 'Web Server config'.



First, click the 'Read' button.

Web Server config

Web Server config
Set Password

Local

IP Address: 192 . 168 . 0 . 223
Subnet Mask: 255 . 255 . 255 . 0
Default Gateway: 192 . 168 . 0 . 1
Web Port: 80

DHCP Server: ☐ Enable
Protocol: MODBUS-TCP RTU
☒ Enable Web Server
MAC Address: 70-B3-D5-8C-1B-7C

TCP Server

Port: 8008 Keep Alive: 5 s ☒ Enable
Max Clients: 4 Timeout: 0 s

UDP Server

Port: 8007 ☒ Enable
Timeout: 0 s

Target

| | IP Address | Port | Keep Alive | Type | Timeout |
|---|---------------------|------|------------|------|---------|
| <input checked="" type="checkbox"/> 1. Enable | 192 . 168 . 0 . 118 | 6008 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 2. Enable | 192 . 168 . 0 . 24 | 8002 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 3. Enable | 192 . 168 . 0 . 24 | 8003 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 4. Enable | 192 . 168 . 0 . 24 | 8004 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 5. Enable | 0 . 0 . 0 . 0 | 0 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 6. Enable | 0 . 0 . 0 . 0 | 0 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 7. Enable | 0 . 0 . 0 . 0 | 0 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 8. Enable | 0 . 0 . 0 . 0 | 0 | 5 s | TCP | 0 s |

Write
Read
Close

Configuration parameters, the specific method is described below (page 5).
When the configuration is complete, click the 'Write' button.

Web Server config

Web Server config
Set Password

Local

IP Address: 192 . 169 . 0 . 201
Subnet Mask: 255 . 255 . 255 . 0
Default Gateway: 192 . 169 . 0 . 1
Web Port: 80

DHCP Server: ☐ Enable
Protocol: MODBUS-TCP RTU
☒ Enable Web Server
MAC Address:

TCP Server

Port: 8008 Keep Alive: 5 s ☒ Enable
Max Clients: 0 Timeout: 0 s

UDP Server

Port: 8007 ☒ Enable
Timeout: 0 s

Target

| | IP Address | Port | Keep Alive | Type | Timeout |
|---|---------------------|------|------------|------|---------|
| <input checked="" type="checkbox"/> 1. Enable | 192 . 169 . 0 . 201 | 6008 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 2. Enable | 192 . 169 . 0 . 202 | 8002 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 3. Enable | 192 . 169 . 0 . 203 | 8003 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 4. Enable | 192 . 169 . 0 . 204 | 8004 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 5. Enable | 192 . 169 . 0 . 206 | 0 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 6. Enable | 192 . 169 . 0 . 207 | 0 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 7. Enable | 192 . 169 . 0 . 208 | 0 | 5 s | TCP | 0 s |
| <input type="checkbox"/> 8. Enable | 192 . 169 . 0 . 209 | 0 | 5 s | TCP | 0 s |

Write
Read
Close

NetSearch

OK
确定

Click 'OK'.

Click the 'Close' button. Prompt you need to restart the PLC and select 'Yes(Y)'.

Info

Do you want to reboot the system ?

是(Y) 否(N)

The newly configured IP parameters take effect after the PLC restarts.。

(2) Parameter Description:

The screenshot shows the 'Web Server config' window. It contains several sections:

- Local (A):** Fields for IP Address (192.168.0.172), Subnet Mask (255.255.255.0), Default Gateway (192.168.0.1), and Web Port (80).
- DHCP Server (B):** Includes a 'Protocol' dropdown menu set to 'MODBUS-TCP RTU', a checked 'Enable Web Server' checkbox, and a 'MAC Address' field (70-B3-D5-8C-10-0E).
- TCP Server (C):** Fields for Port (8500), Keep Alive (5s), Max Clients (4), and Timeout (5s). The 'Enable' checkbox is checked.
- UDP Server (D):** Fields for Port (8007) and Timeout (0s). The 'Enable' checkbox is checked.
- Target Channel (E):** A table with 8 rows. The first two rows are checked. The table has columns for Channel, IP Address, Port, Keep Alive, Type, and Timeout.
- Buttons (F):** A group of buttons including 'Write', 'Read', 'Confirm & Reset', and 'Exit'.

| Target Channel | IP Address | Port | Keep Alive | Type | Timeout |
|---------------------------------------|---------------|------|------------|------|---------|
| <input checked="" type="checkbox"/> 1 | 192.168.0.246 | 8003 | 5s | TCP | 0s |
| <input checked="" type="checkbox"/> 2 | 192.168.0.210 | 8002 | 5s | TCP | 0s |
| <input type="checkbox"/> 3 | 0.0.0.0 | 0 | 5s | TCP | 0s |
| <input type="checkbox"/> 4 | 0.0.0.0 | 0 | 5s | TCP | 0s |
| <input type="checkbox"/> 5 | 0.0.0.0 | 0 | 5s | TCP | 0s |
| <input type="checkbox"/> 6 | 0.0.0.0 | 0 | 5s | TCP | 0s |
| <input type="checkbox"/> 7 | 0.0.0.0 | 0 | 5s | TCP | 0s |
| <input type="checkbox"/> 8 | 0.0.0.0 | 0 | 5s | TCP | 0s |

The parameters in the above figure:

Area A: Set the parameters such as the IP address of the PLC to connect to the network (the Web Port remains unchanged), meaning the same as the IP address of the configuration computer.

Area B: By default, it stays the same.

Area C: The parameter when the PLC is used as a server (TCP protocol).

Port: The port of the PLC when doing Server, try not to use 8001.

Keep Alive: Keep the connection time, the default is 5s.

Max Clients: 0 – 8, When the PLC is used as a server, you can select how many channels are used by the client at the same time, up to 8.

Timeout: overtime time.

Enable: Checked by default. Must be checked when using TCP protocol.

Area D: The parameter when the PLC is used as a server (UDP protocol).

Port: The port of the PLC when doing Server, try not to use 8001. Not the same as TCP.

Timeout: overtime time.

Enable: Checked by default. Must be checked when using UDP protocol.

Area E: Set remote target IP parameters.

Used to configure the IP and port of the servers when the PLC is used as a client. The sum of the number of servers that can be configured and the number of Max Clients set in Area C is 8. If Area C's Max Clients is set to 8, channels 1 – 8 in Area E are not available. If Area C's Max Clients is set to 4, channels 1 – 4 in Area E are available, and channels 5 – 8 are not available.

Keep Alive: Keep the connection time, the default is 5s.

Type: TCP or UDP protocol, the default is TCP.

Timeout: overtime time, the default is 0s.

Area F:

Read: Read network configuration parameters.

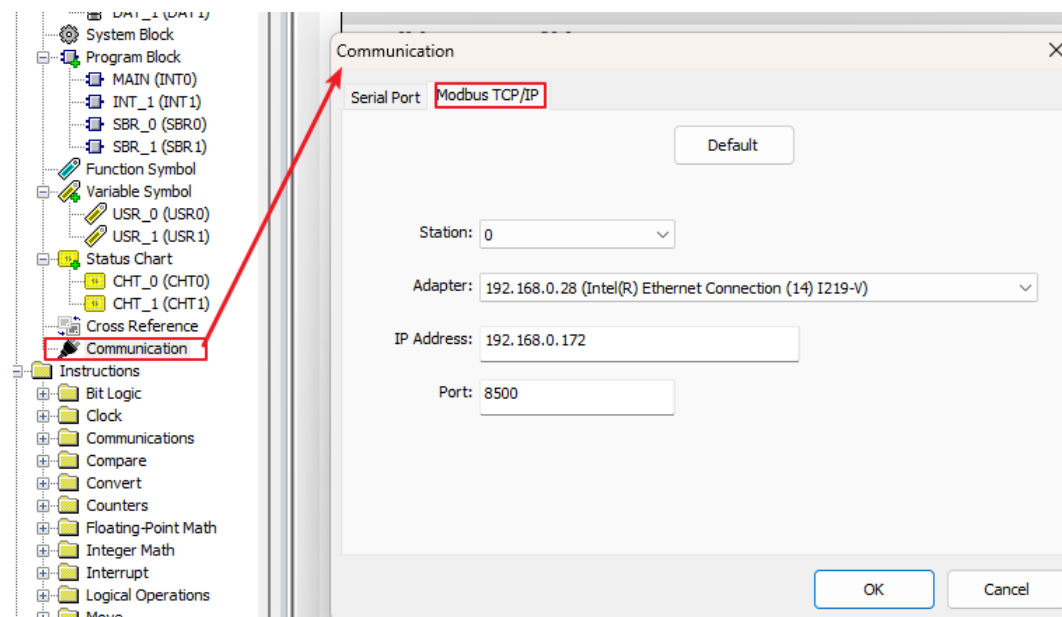
Write: Write network configuration parameters.

Confirm & Reset: After writing the configuration, click this button to complete the configuration and the PLC will restart.

After the parameters are configured, it is best to test whether there is successful access to

the network. The method is to run CMD.exe (on PC): **ping 192.168.0.172**

(3) Software reads/writes programs over the network:

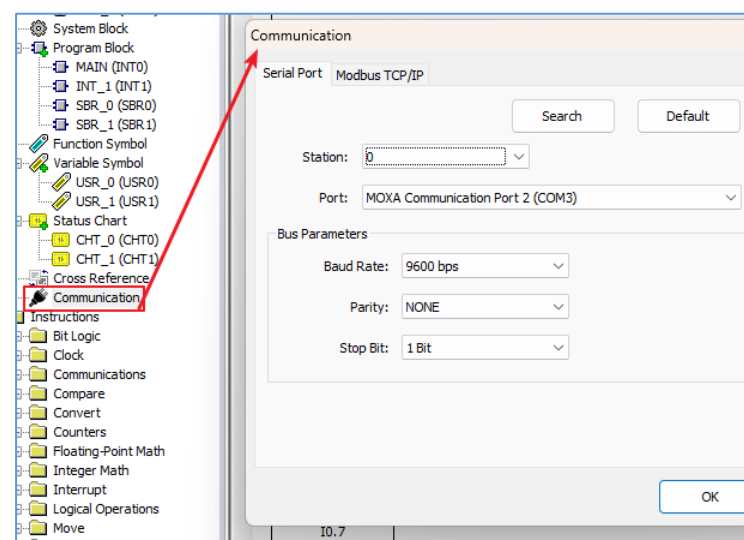


The PLC in xLadder is used as the server by default, so the IP and Port of the PLC need to be filled in the above figure.

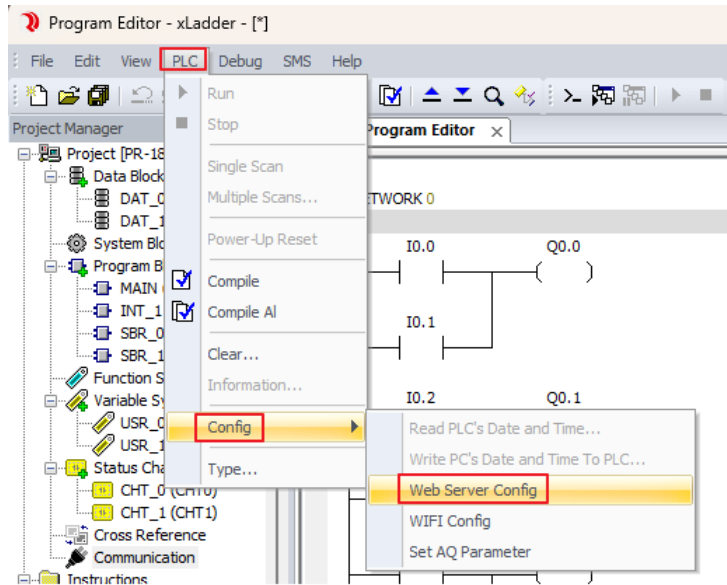
After filling in, click the 'OK' button.

(b) Configure IP parameters using USB CABLE:

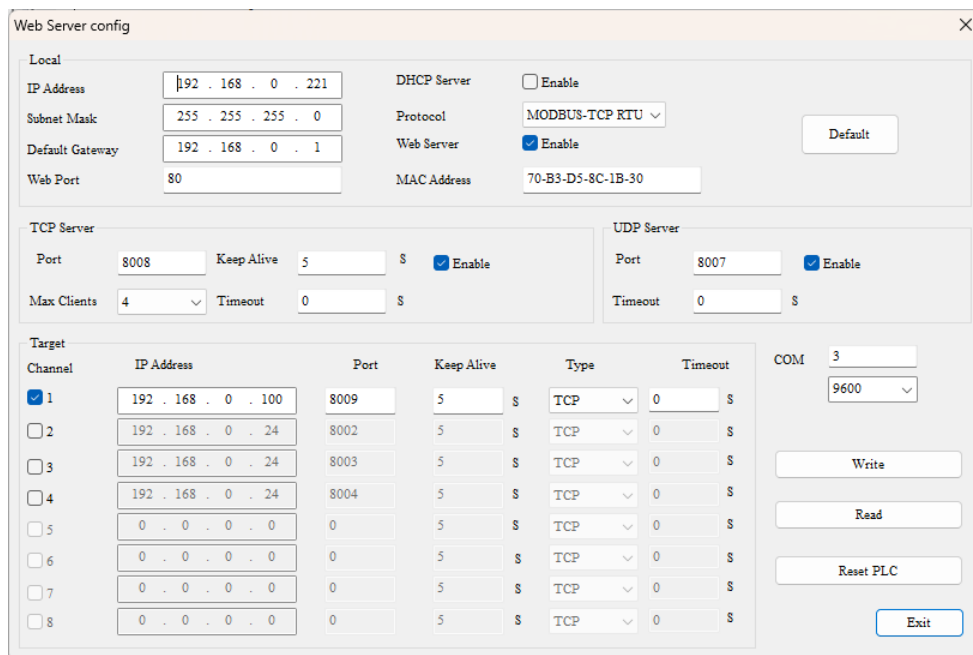
(1) Configure the parameters in 'Communication' so that xLadder can communicate with PLC.



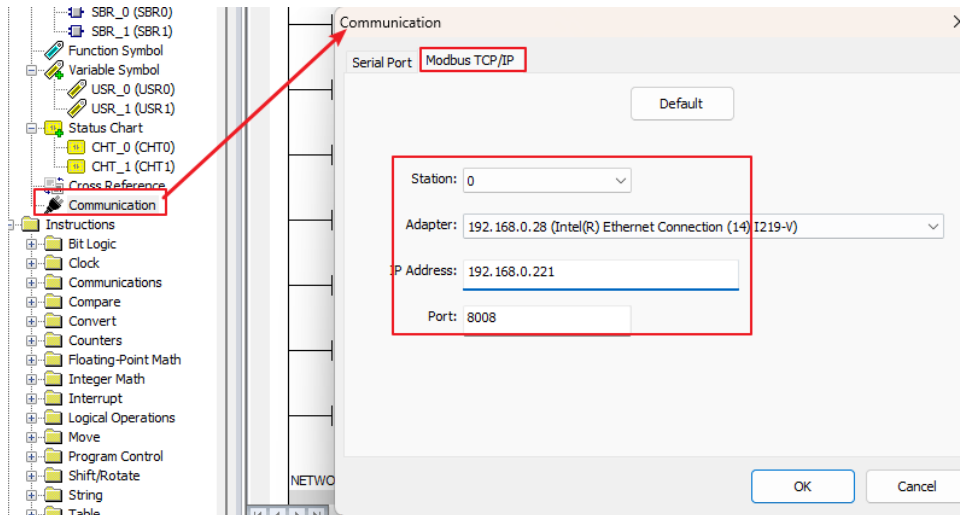
Use the "PLC - Config - Web Server Config" menu to open the configuration window.



xLadder software automatically reads the current configuration (or click the 'Read' button), as shown below:



After modifying the parameters, click the "Write" button to write the parameters to the PLC. Then click the "Reset PLC" button to restart the PLC. The modified parameters will take effect after the PLC restarts.



xLadder can communicate with PLC via Ethernet.

10.12.2 SR-12 Series

SR12 PLC supports 2 TCP connections. In the first TCP connection (Socket 1), SR12 is fixedly used as the server.

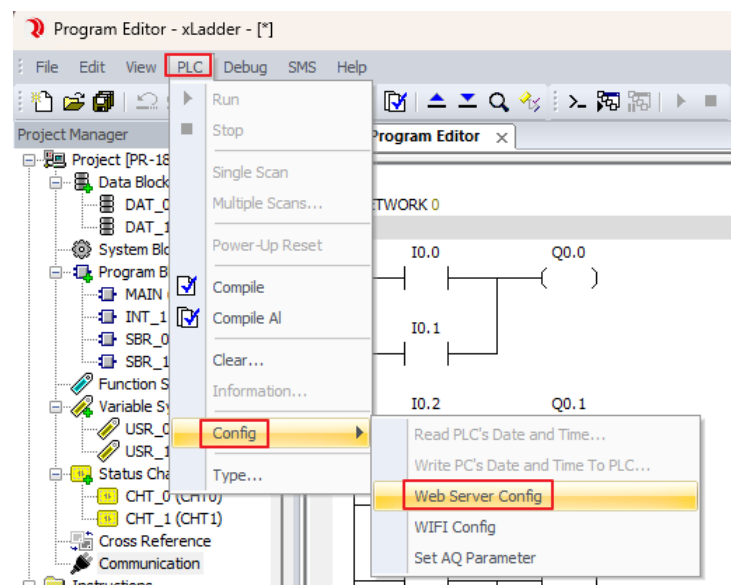
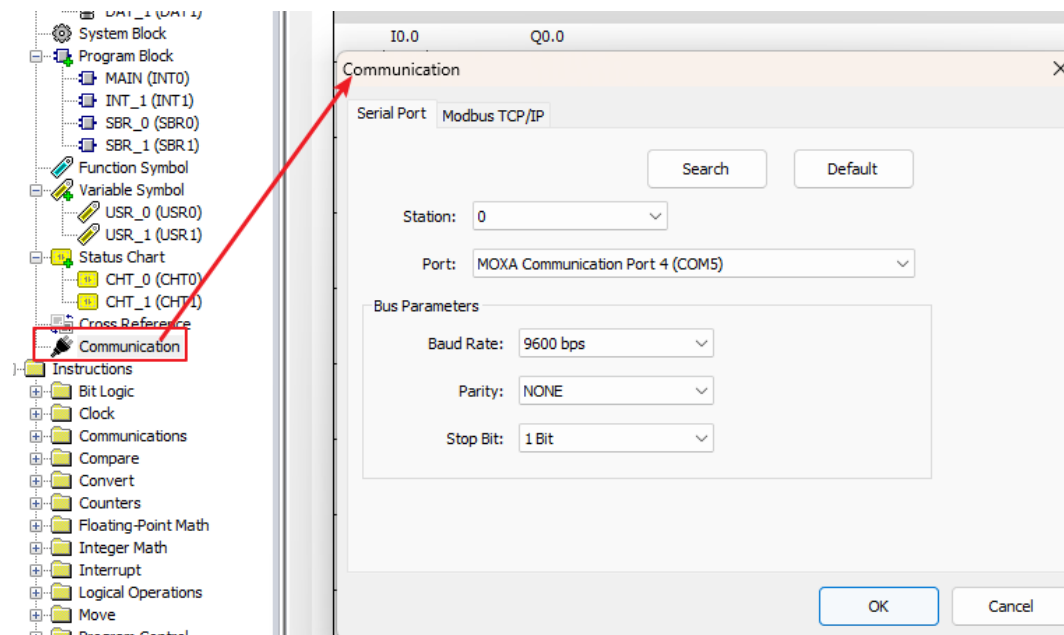
In the second TCP connection (Socket 2), SR12 can be used as a server or client. The detailed operation steps are as follows.

You can use the RS485 channel of SR-RS485. You can also do the following via Ethernet channel. The following recommendations need to be followed:

The default IP of SR-12 is 192.168.0.245, server port: 8008. The PC is set to a fixed IP address, which is in the same network segment as the PLC IP address. The PC and PLC can be connected via a switch/router or directly via a network cable.

(1) Open the SR12 Ethernet parameter configuration window ('Web Server config'), There are 2 ways to open this window, one is using RS485 (SR-RS485), the other is Ethernet.

a) RS485

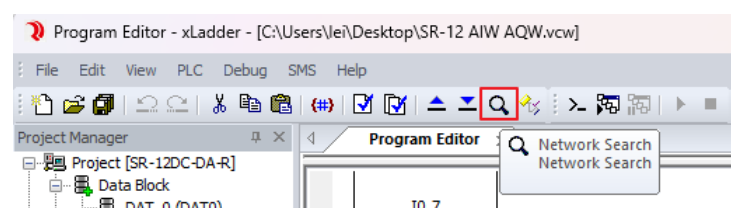


b) Ethernet

The PC and PLC can be connected via a switch/router, or the two can be connected directly via a network cable.

When the PLC and PC are connected via a network cable, both the PLC and PC need to be set to fixed IP addresses in the same LAN.

Then click the 'Network Search' button in the toolbar.



Network Search

2 Search 100% 1 Net Card Select

| Index | DeviceName | IP Address | MAC Address | CPU ID |
|-------|------------|---------------|-------------------|-----------------------|
| 1 | RIEVTECH | 192.168.0.222 | 38-3B-26-87-4A-ED | 6464007491439500 |
| 2 | RIEVTECH | 192.168.0.240 | 38-3B-26-87-DD-C3 | 6464383391439500 |
| 3 | RIEVTECH | 192.168.0.202 | F0-D7-AF-70-38-D4 | 404066699295341772800 |
| 4 | RIEVTECH | 192.168.0.245 | F0-D7-AF-70-4E-09 | 134540551185201083210 |

After selecting, double-click.

Enter the username and password, both are admin by default.

Login

Username: admin

Password: ●●●●●●

OK Cancel

(2) The 'Web Server config' window that pops up is as follows:

Ethernet config

Local

IP Address A 192 . 168 . 0 . 222 Protocol MODBUS-TCP RTU

Subnet Mask 255 . 255 . 255 . 0 Keep Alive 5 s

Default Gateway 192 . 168 . 0 . 1 Timeout 10 s

DHCP Server ☐ Enable

Socket 1 B

Mode Server Port 8008 Protocol Type TCP

Socket 2 C

Mode Source Port Dest Port Dest IP Address Protocol Type

☒ Enable Client 8009 8012 192 . 168 . 0 . 28 TCP

Read Write Reset PLC Exit

Explain the main parameters in the above figure:

Area A:

Set the PLC's IP, Subnet Mask, and Gateway parameters here. If you check 'Enable' of the 'DHCP Server' option, the above settings will be invalid and they will automatically obtain a new IP, Subnet Mask, and Gateway from the router.

It should be noted that when using the firmware update package to update the firmware, it is forbidden to check the "DHCP Server" option, otherwise the update will fail.

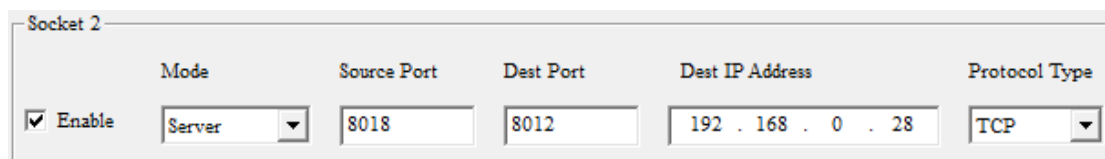
Keep the default values for 'Keep Alive' and 'Timeout'.

Area B - Socket 1:

Set the server port of SR12 in the first TCP connection. Here, SR12 can only be used as a server.

Area C - Socket 2:

SR12 can be selected to be used as server or client through the options here. It should be noted that the 'Source Port' will be used regardless of the mode in which SR12 works, so its value cannot be the same as the port value in 'Socket 1'.



The screenshot shows the 'Socket 2' configuration window. It has a title bar 'Socket 2'. Below it, there are six fields: 'Enable' (checked), 'Mode' (set to 'Server'), 'Source Port' (8018), 'Dest Port' (8012), 'Dest IP Address' (192 . 168 . 0 . 28), and 'Protocol Type' (TCP). The 'Mode' dropdown is currently set to 'Server'.

When 'Mode' is set to 'Server', only 'Source Port' is used. The value of 'Source Port' is the server port.



The screenshot shows the 'Socket 2' configuration window. It has a title bar 'Socket 2'. Below it, there are six fields: 'Enable' (checked), 'Mode' (set to 'Client'), 'Source Port' (8018), 'Dest Port' (8012), 'Dest IP Address' (192 . 168 . 0 . 28), and 'Protocol Type' (TCP). The 'Mode' dropdown is currently set to 'Client'.

When 'Mode' is set to 'client', 'Source Port', 'Dest Port', and 'Dest IP Address' will be used. 'Dest Port' and 'Dest IP Address' are the server IP and server port of the other device (Server) to be connected.

TCP is selected by default for 'Protocol Type'.

The 4 buttons below:

'Read' : Read the Ethernet parameters of the SR12 PLC. Usually when this window is opened, xladder will read automatically, and it is necessary to ensure that xladder and SR12 are connected normally.

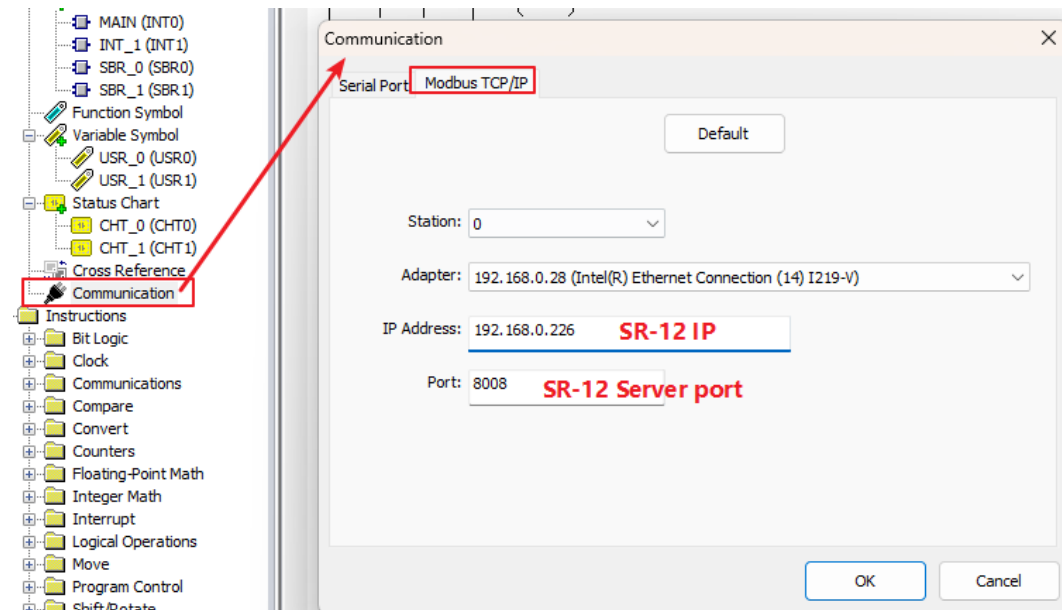
'Write' : After modifying the parameters in this window, you need to click this button to write the new parameters into the SR12 PLC.

'Reset PLC' : After writing the new parameters into SR12, click this button to restart the PLC. The new parameters will take effect after the restart.

'Exit' : Exit this window to complete the Ethernet parameter modification operation.

(3) Read and write SR12 program via Ethernet on xladder It should be noted that the PC

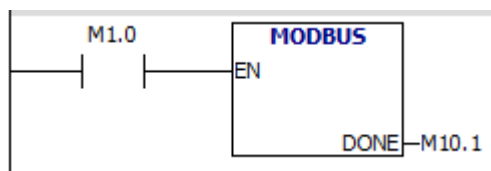
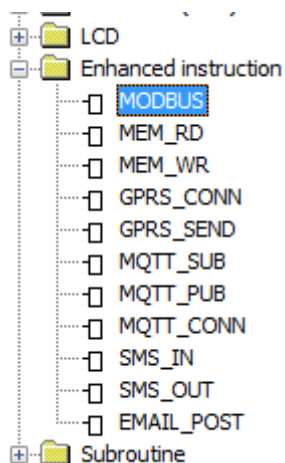
where xladder is installed needs to be in the same LAN as SR12. For example, the IP of PC is 192.168.0.28, and the IP of PLC is 192.168.0.226



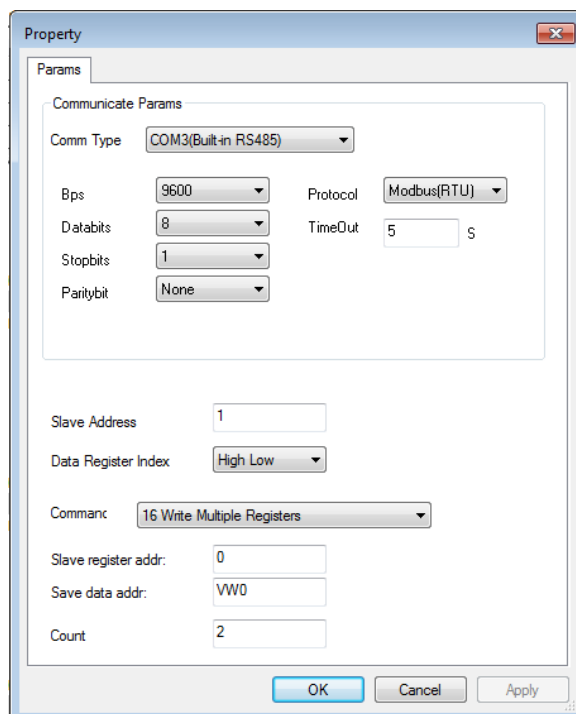
Finally, it should be noted that the Ethernet connection of the SR12 PLC is a hardware connection and is not controlled by software. When the Ethernet parameters of both parties are successfully configured, as long as they are connected through the network cable and router, the PLC hardware will automatically connect to both parties. As long as the physical connection is not disconnected, the TCP connection will not be disconnected.

For example, when two SR12s have been configured the Ethernet parameters and physically connected, and want to achieve Modbus TCP communication, even if the modbus block in the client program is not triggered, the TCP connection used between them still exists, and the third device/software can no longer use this TCP connection. You need to pay attention to this when using it, which is different from the PR series Ethernet.

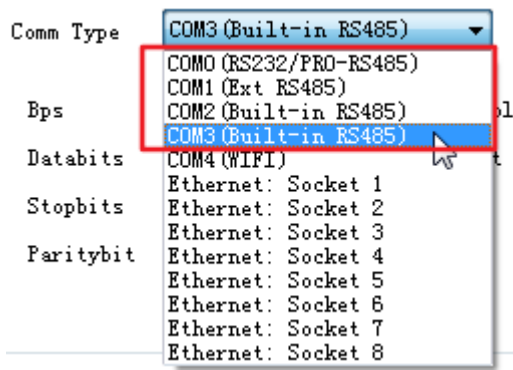
10.13 Modbus block (New block for Ethernet series PLC)



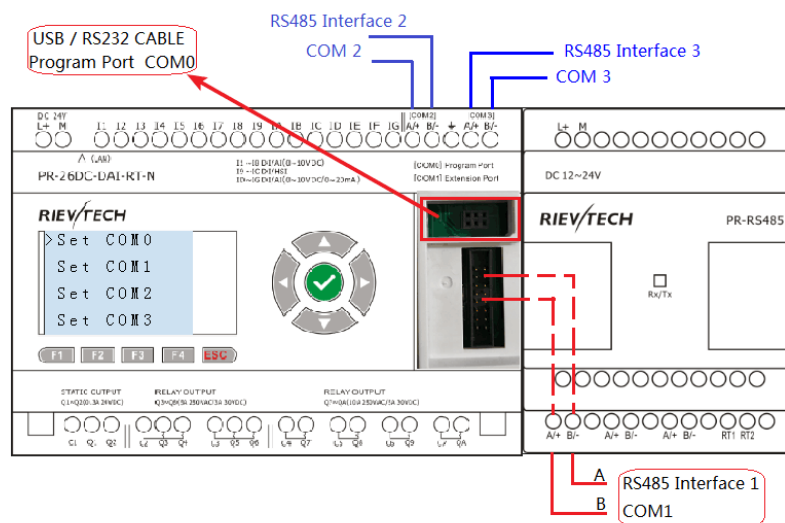
1, Modbus RTU



The 4 options selected in the red box in the above figure are serial communication ports, which support Modbus RTU protocol.



COM port diagram:



2, Modbus TCP

First, connect the PLCs to the LAN where the PC is located. And belong to the same network segment. Or 2 PLCs are on the same network segment.

1, Configure Ethernet IP parameters on xLadder software.

See chapter 10.12 for details

2, Block parameter settings.

A) This PLC is used as a client, and the block configuration is as follows:

Property

Params

Communicate Params

Comm Type: Ethernet: Socket 2

Local CPU: Client Protocol: Modbus(TCP)

TCP/UDP: TCP TimeOut: 5 s

Remote: 1

Target IP: 192.168.0.21

Target Port: 0

Slave Address: 1

Data Register Index: High Low

Command: 16 Write Multiple Registers

Slave register addr: 0

Save data addr: VW0

Count: 2

OK Cancel Apply

Property

Params

Communicate Params

Comm Type: Ethernet: Socket 2

Local CPU: Client Protocol: Modbus(TCP)

TCP/UDP: TCP TimeOut: 5 s

Remote: 1

Target IP: 192.168.0.21

Target Port: 0

Slave Address: 1

Data Register Index: High Low

Command: 16 Write Multiple Registers

Slave register addr: 0 Server register address

Save data addr: VW0 The obtained value is stored in the register starting from VW0

Count: 2 Set the number of registers (read)

确定 取消 应用(A)

Web Server config

Local

IP Address: 192.168.0.210 DHCP

Subnet Mask: 255.255.255.0

Default Gateway: 192.168.0.1

Web Port: 80

TCP Server

Port: 8008 Keep Alive: 5 s

Max Clients: 4 Timeout: 0 s

Target

Channel

IP Address

Port

1 192.168.0.211 8009

2

3

4

5

6

7

8

Channel number comes from here. Fill in the channel number of the IP address of the server you want to connect to.

B) This PLC is used as a server, and the block configuration is as follows:

The IP and port information of the MQTT server is no longer obtained from 'Web Server Config'. Instead, get from this block, which triggers this block to connect to the server.

Publish and subscribe function blocks also do not configure the parameters of the MQTT server, they are simplified to have only publish and subscribe features.

Only one Connect to MQTT Server block can be placed in the program.

| Connection | Description |
|------------|---|
| En | You enable/disable the block with the signal at input En (1/0). |
| Parameter | <p>Network: Ethernet/4G</p> <p>IP Address: The IP address of the MQTT server.</p> <p>Domain Name: The domain Name of MQTT server.</p> <p>Port: The port of the MQTT server.</p> <p>Connection timeout: Timeout setting for cloud connection in seconds.</p> <p>PING interval time: Ping timer setting in seconds. If there is no message to publish before the timer expires, the PLC will send a ping command to the cloud server to check connection status.</p> <p>Enable SSL/TLS: After checking, select to use SSL encryption mode.</p> <p>CA Certificate File 1: When using encryption mode, select the SSL certificate by path.</p> <p>CA Certificate File 2: Temporarily not supported.</p> <p>CA Certificate File 3: Temporarily not supported.</p> <p>Client ID: Click the 'Create only one ID' button to generate a unique string of numbers.</p> <p>Name / Password: Before you can publish / subscribe a message, you need to configure your cloud server account and 'Topics'. Each cloud server may have different configuration procedures. Check the cloud server instructions on how to do this.</p> <p>Hide: Hide the password. You can set a secret password when hiding, and you need this secret password when you unhide it. You can also modify the content without a secret password. You can also choose not to set a secret password when hiding.</p> |
| Out 1 | <p>Digital output.</p> <p>Shows if the server is connected successfully.</p> <p>1: Successfully connected to the MQTT server</p> |

| | |
|-------|---|
| | 0: The module is not triggered or is connecting to the server |
| Out 2 | <p>Analog output.</p> <p>The status code of the connection server.</p> <p>2: Connecting to the target server</p> <p>3: MQTT PING is sending PING packet</p> <p>5: Disconnect</p> <p>20: Resolving domain name</p> <p>21: Domain name resolution successful</p> <p>22, 23, 24: Domain name resolution failed</p> <p>50: Connection failed</p> <p>255: The connection is successful</p> |

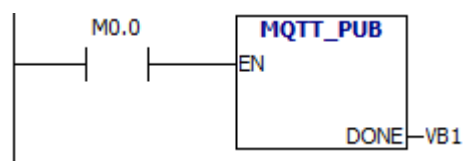
NOTE:

- 'Name' and 'Password' are obtained/set when creating a project on the MQTT server.
- The SSL certificate is issued by a trusted digital certificate authority CA (e.g. Symantec) after verifying the identity of the server. It has server authentication and data transfer encryption.

| | |
|---------|---|
| CA File | F:\00-桌面\backup2024-11\AWS CA\AmazonRootCA1.pem |
|---------|---|

- Check the 'Clean Session' option to clear the session cache. Checked by default.

10.14.2 Publish with MQTT



Description of Function

After successfully connecting to the MQTT server, activate this function block to publish the message to a topic in the MQTT server.

| Connection | Description | | | | | | | | | | |
|------------|---|---------|----------|-------------|--------|-------------|--|------|---|------|--|
| En | You enable/disable the block with the signal at input En (1/0). | | | | | | | | | | |
| Parameter | <p>Network: Ethernet / 4G</p> <p>Retained:</p> <p>Typically, if a publisher publishes a message to a topic, when no one subscribes to the topic, then the message will be discarded (not retained) by the MQTT server.</p> <p>However, the publisher can tell the MQTT server to keep the last message of the topic by setting a reserved message flag. When a device subscribed to this topic goes online, the MQTT server sends the retained information to the subscriber.</p> <p>You can check this option to implement this ‘information retention’ feature. A powered-down device can receive the last message of its subscribed topic after powering on.</p> <p>QoS Level:</p> <p>QoS 0(no answer) or QoS 1(answer). If you wish to receive a response from the MQTT server, select ‘QoS 1’.</p> <p>Publish interval time:</p> <p>Publish timer setting. If enabled, the xLadder will publish the message to the MQTT server every x seconds. If disabled, the message will only be sent once.</p> <p>Publish topic:</p> <p>Set when creating a new topic project on the MQTT server.</p> <p>Parameter:</p> <p>Set the message content.</p> <div><div>Comm</div><div>Params</div><table><tr><th>Comment</th><th>Register</th><th>count</th><th>Format</th><th>Description</th></tr><tr><td></td><td>VW10</td><td>2</td><td>UINT</td><td></td></tr></table></div> | Comment | Register | count | Format | Description | | VW10 | 2 | UINT | |
| Comment | Register | count | Format | Description | | | | | | | |
| | VW10 | 2 | UINT | | | | | | | | |

Property

Comm Params

Network: Ethernet

QoS Level: ☐ QoS1 ☒ QoS0 ☐ Retained

☐ Publish interval time: 0 s

Publish topic: PLCmqtttest01/results

确定 取消 应用(A)

| Comment | Register | count | Format | Description |
|----------|----------|-------|--------|-------------|
| { "sumRe | | 1 | | sult |
| "{ | | 1 | | |
| "n1": | VW0 | 1 | INT | , |
| "n2": | VW2 | 1 | INT | , |
| "result" | | 1 | | : |
| | VW4 | 1 | INT | }} |

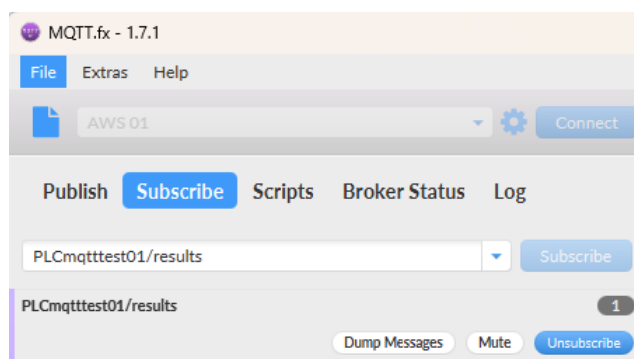
- Publish Topic: Before you can publish a message, you need to configure your MQTT cloud server account and include setting up of 'Topics' usually under the Themes area on your account. Each MQTT cloud server may have different configuration procedures for their Topics. Check the MQTT cloud server instructions on how to do this. There are generally five steps:
 1. Create an account on a cloud server.
 2. Add a device to your account.
 3. Set-up an identity for your account.
 4. Create a policy for your account.
 5. Create a topic for your account.

Once you have configured the Topic name on the MQTT cloud server you are using, enter it here.

MQTT Publish Status Codes

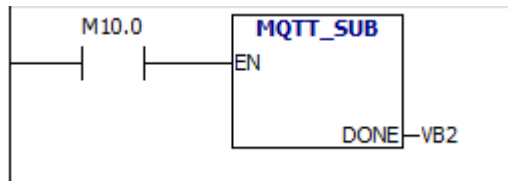
| Code | Description |
|------|---|
| 00 | Connection Accepted |
| 51 | Connection Refused: unacceptable protocol version |
| 52 | Connection Refused: identifier rejected |
| 53 | Connection Refused: server unavailable |
| 54 | Connection Refused: bad username or password |
| 55 | Connection Refused: not authorized |
| 2 | Wait to connect to the server. |
| 3 | The PING data packet is being sent. |
| 4 | Time to time start send the published data to the server and wait for server response (QoS1). |
| 5 | Timeout – Disconnect and retry server connection. |
| 13 | MQTT PING success |
| 14 | Successful response to server response (QoS1) . |
| 255 | Successful connection to the server and ready to Publish. |

Subscribe to plcmqtt_topic/40 on the MQTT client software MQTT.fx.



Set M0.0 to 1.

10.14.3 Subscribe with MQTT



Description of Function

You can use this function to allow the xLadder to subscribe to a message on the MQTT cloud server. Only 8 Subscribe blocks can be placed in one program.

Once you have added it to your program, double-click the block to configure it.

| Connection | Description |
|------------|--|
| En | You enable/disable the block with the signal at input En (1/0). |
| Parameter | <p>Network: Ethernet / 4G</p> <p>QoS Level: QoS 0(no answer) or QoS 1(answer). If you wish to receive a response from the cloud server, select 'QoS 1'.</p> <p>Subscribe interval time: Subscribe timer setting. If enabled, the xLadder will Subscribe to the message from the cloud server every x seconds. If disabled, the message will only be subscribed to once.</p> <p>Subscribe topic: Fill in the topics you want to subscribe to. However, it should be noted that the topic of the multiple subscribe blocks cannot be the same.</p> <p>Parameter: Set the start register to store the received information data.</p> |
| Out 1 | <p>Digital output.</p> <p>Indicates whether the information was successfully subscribed.</p> <p>0: Not triggered or did not receive the topic it subscribed to 1: Received a subscription topic</p> |
| Out 2 | <p>Analog output.</p> <p>Subscribe to the status code.</p> <p>0: Block is not triggered 1: There are no free subscription channels (maximum 8 subscription blocks). 2: Wait for the MQTT server to connect or wait for the subscription channel to be idle. 6: Sending a subscription request 16: The subscription request is successful, waiting to receive the topic of the subscription. 26: Received a topic subscribed to by this block 36: Subscription request failed, waiting for the block subscription timeout 8: Unsubscribe</p> |

Property

Comm Params

Network: Ethernet

QoS Level: ☐ QoS1 ☒ QoS0

☐ Subscribe interval time: 0 s

Subscribe topic: \$aws/things/PLC_MQTT_01/shadow/name/PLC11/update/PLC

☐ Azure C2D

确定 取消 应用(A)

Property

Comm Params

Message saved to:

Strings Start byte: VB 6001 (Total length: 256 chars)

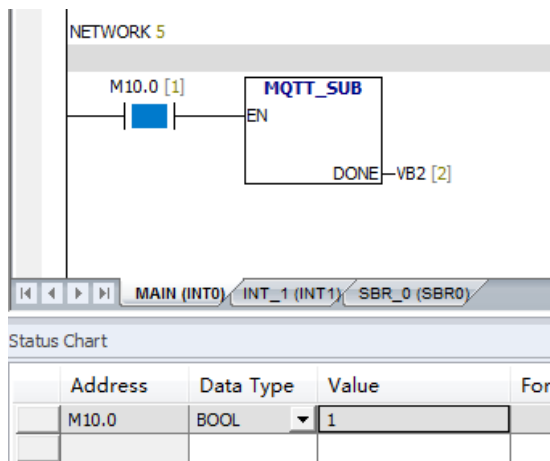
Receive number: As Received 1

Option 1: User can set number of chars could read "Chars to Receive"

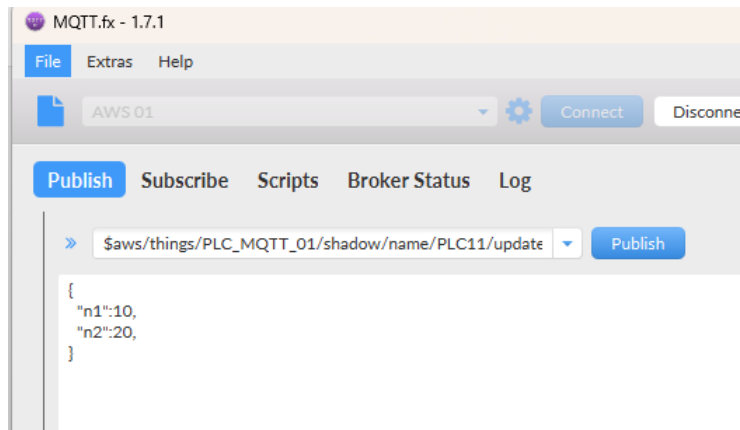
Option 2: User can NOT set number of chars, could read "As Received"

确定 取消 应用(A)

Set M10.0 to 1 on Xladder.



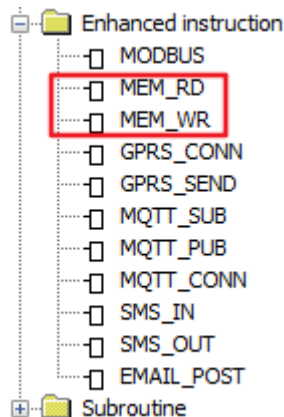
“\$aws/things/PLC_MQTT_01/shadow/name/PLC11/update/PLC” are published on the MQTT client software MQTT.fx.



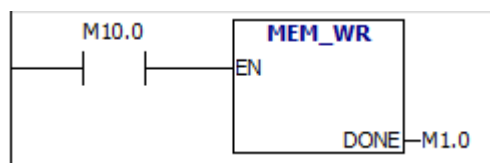
| Address | Data Type | Value | Forced |
|---------|-----------|----------------------|--------|
| VB6000 | USINT | 254 | |
| VB6000 | STRING | "{"n1":10,"n2":20,}" | |

The value received by VB is ASCII code.
VB6000 specifies the length of the string.

10.15 Memory Read & Memory Write (New block for Ethernet series PLC)



10.15.1 Memory Write



Description of function

Only when there is a low to high trigger at Trg pin, the Memory Write block will be activated and the pre-configured record action will be performed, at the same time the output will switch on if

the record action had been done successfully.

Property

Params

Record position: Internal Card

File Params

File Name: 1 VW .txt

Record Title: 2 Value:

File Write Mode: 3 Append ☒ Save Record Time

Separator: 4 ,

File Size: 5 512k

After Memory Full: 6 Overlay

8 ☐ Record Index 0

Register Params 7

Register: VW0

Count: 3

Data Sequence: HI-LO

Format: INT

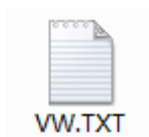
Decimals: 0

OK Cancel Apply

1. Filename

Place where you can set the name of the file used to save the registers' data

File in SD card



Above is an example of the "VW.TXT"

2. Record title

```
2020-05-07 20:21:33 Value: 3 200 300
2020-05-07 20:21:34 Value: 3 200 300
2020-05-07 20:21:35 Value: 4 200 300
```

The above range circled in red is just pre-set contents in the "Record title" of the Memory write block's property dialogue box.

3. File Write Mode

Two options available:

Option A. Append (This option would be selected if a certain file already exists in the Mini SD card inserted in PLC).

B. Create (This option should be chosen, if no file existed or the existing file has a different name from that pre-set in the “filename” in the Mini SD card inserted in PLC).

☒ Save Record Time

If this box has been ticked, the file content will show the time when the data starts to be recorded.

2020-05-07 20:21:58 Value: 9 200 300

4. Separator

Such separator shall be required while more than one analogue values would be stored and displayed for easier observation and convenient analysis.

5. File Size

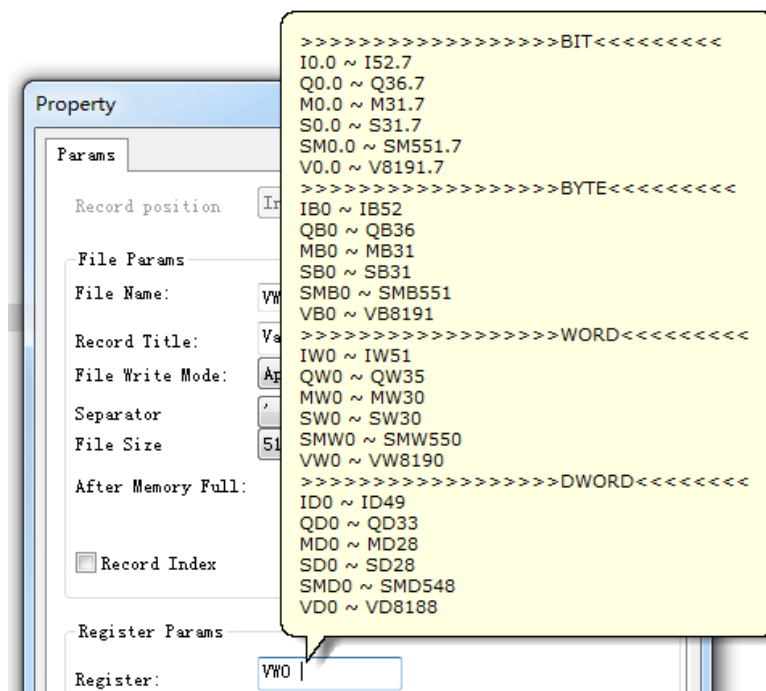
Sets the size of the file to be stored.

6. After memory Full

Two options can be selected after the memory is full (it means the relative file has reached its pre-configured size), one is to over-write and the other is to stop recording.

7. Register params

This section is for register’s parameters setting. The register includes the following choices:



8. Record Index

Property

Params

Record position: Internal Card

File Params

File Name: VW .txt

Record Title: Value:

File Write Mode: Append ☒ Save Record Time

Separator: ,

File Size: 512k

After Memory Full: Overlay

☒ Record Index: 3

Register Params

Register: VW0

Count: 3

Data Sequence: HI-LO

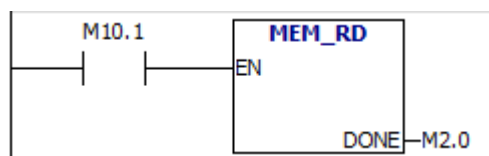
Format: INT

Decimals: 0

OK Cancel Apply

The new data always overwrites the existing third record in the VW.TXT file.
If the third record does not exist in the VW.TXT file, this block cannot work normally.

10.15.2 Memory Read

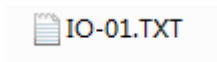


Description of function

Only when there is a low to high trigger at Trg pin, the Memory Read block will be activated once and PLC CPU will read correlative data (bit or short) to set pre-configured register from the file in the SD card, at the same time the output will switch on if the read action had been done successfully.

1. Filename

The name of the file which you want to access is stored in the mini-SD card.



Above is an example of the "IO-01.TXT"

2. Record Title

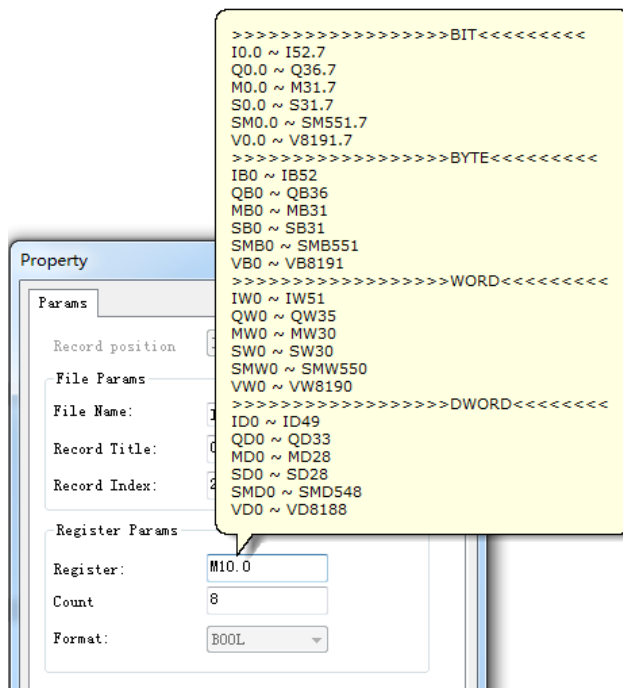
```
2020-05-08 21:57:43 Output: 00000000
2020-05-08 21:57:46 Output: 11111111
2020-05-08 21:57:48 Output: 11111111
```

3. Record Index

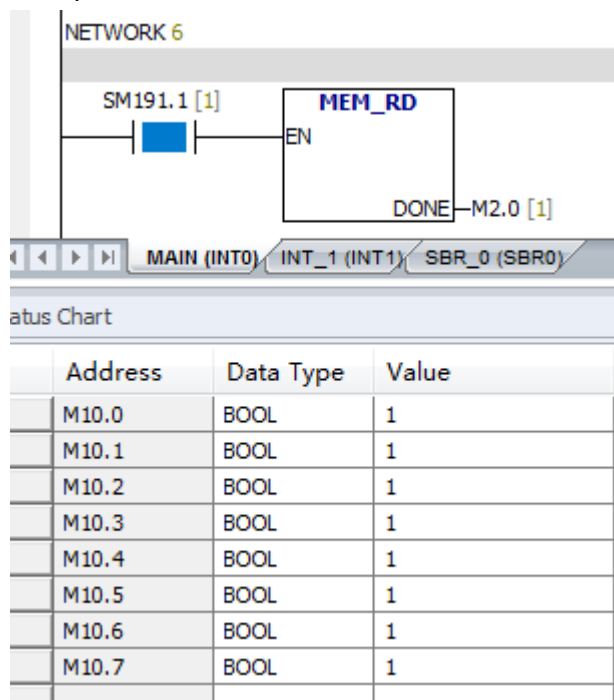
Here is used to set which line the CPU will access via this Memory Read block

4. Register Params

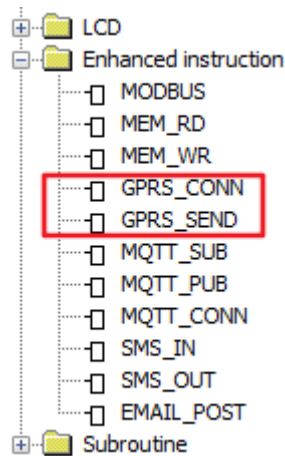
Here is how to set the parameters of the register, all these registers have a "write" property.



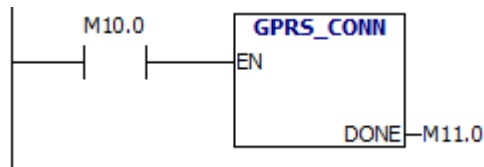
For example,



10.16 GPRS_CONN & GPRS_SEND (New block for Ethernet series PLC)



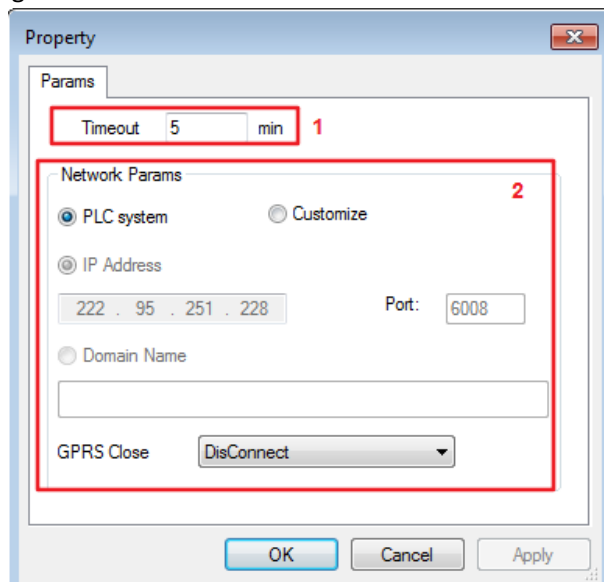
10.16.1 GPRS_CONN



Description of Function:

The block can be used to enable/disable the GPRS connection of the CPU.

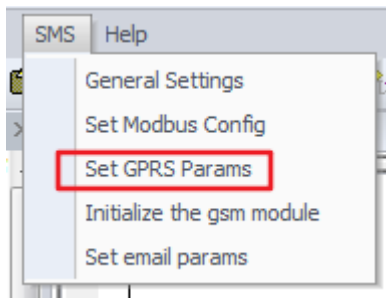
Configuration of Function



1. Timeout Setting – This is the length of time for the timeout if there is no data transmission during this period. The GPRS will disconnect automatically.
2. Network Params – There are two options to choose from.
 - a. PLC System

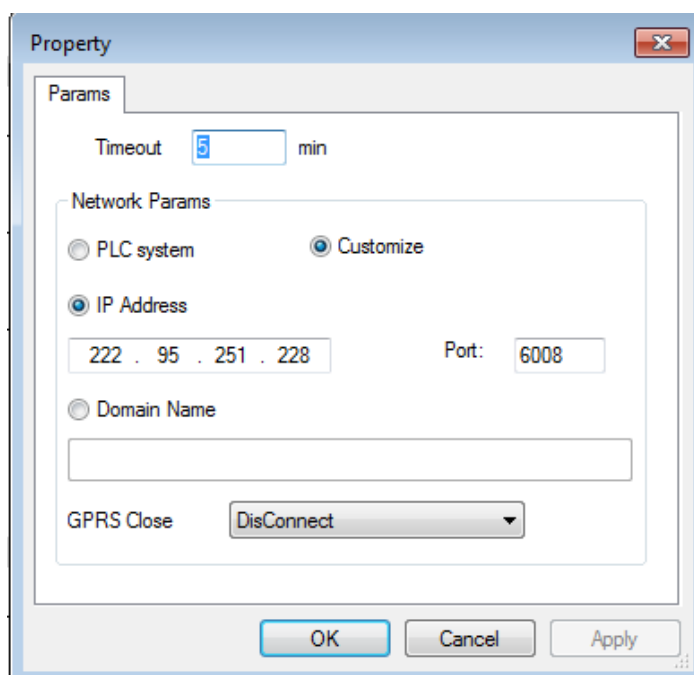
If this setting is selected, the IP and port number of the server shall be the same as configure in

the menu option... "SMS...Set GPRS Params".



b. Customize

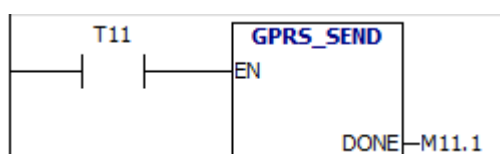
This option allows you to select another server.



NOTICE:

- 1) The configuration of this block has a HIGHER priority than the settings set using the menu option. If the block is activated, the GPRS connection configured via the menu will be disconnected automatically and these settings used instead.
- 2) The data transmission between the CPU and the server is based on standard Modbus/TCP protocol.

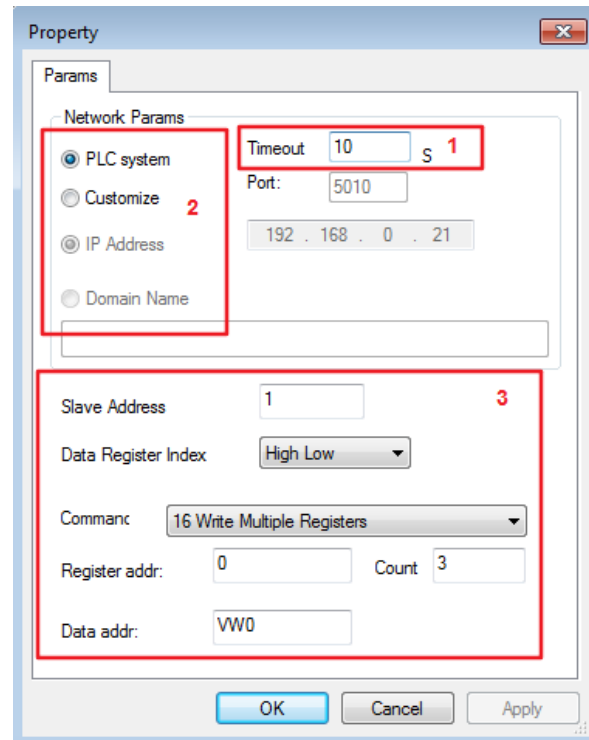
10.16.2 GPRS_SEND



Description of Function

This block can be used to enable a GPRS connection and then transfer some information to the network server.

Configuration of Function



1) Timeout – If GPRS server does not respond within the set timeout period, the CPU will attempt the data transmission up to a maximum of three times after which the GPRS connection will be automatically disconnected.

2) Network Params – There are two options, see [GPRS Connect](#) for further information.

3) If your program has three VW with the following values:

VW0 = 100

VW2 = 1000

VW4 = 10000

The CPU will upload the VW0 to VW4 values to the server (Slave ID 1) as follows:

00 01 00 00 00 0D 01 10 00 00 00 03 06 00 64 03 E8 27 10

The request and response are prefixed by six bytes as follows:

byte 0: transaction identifier - copied by server

byte 1: transaction identifier - copied by server

byte 2: protocol identifier = 0

byte 3: protocol identifier = 0

byte 4: length field (upper byte) = 0 (since all messages are smaller than 256)

byte 5: length field (lower byte) = number of bytes following

byte 6: unit identifier (previously 'slave address')

byte 7: MODBUS function code
byte 8: Register of the slave start address
byte 9: Register of the slave start address
byte 10: number of registers
byte 11: number of registers
byte 12: data length field (lower byte) = number of bytes following

So, AF1 = 00 64 (DECIMAL 100)

AF2 = 03 E8 (DECIMAL 1000)

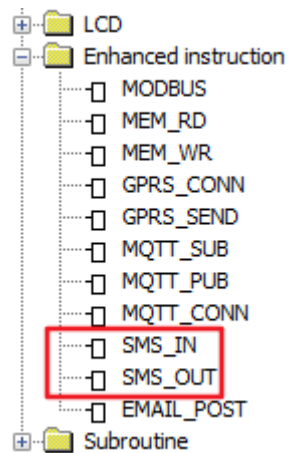
AF3 = 27 10 (DECIMAL 10000)

The server end(slave1) responds

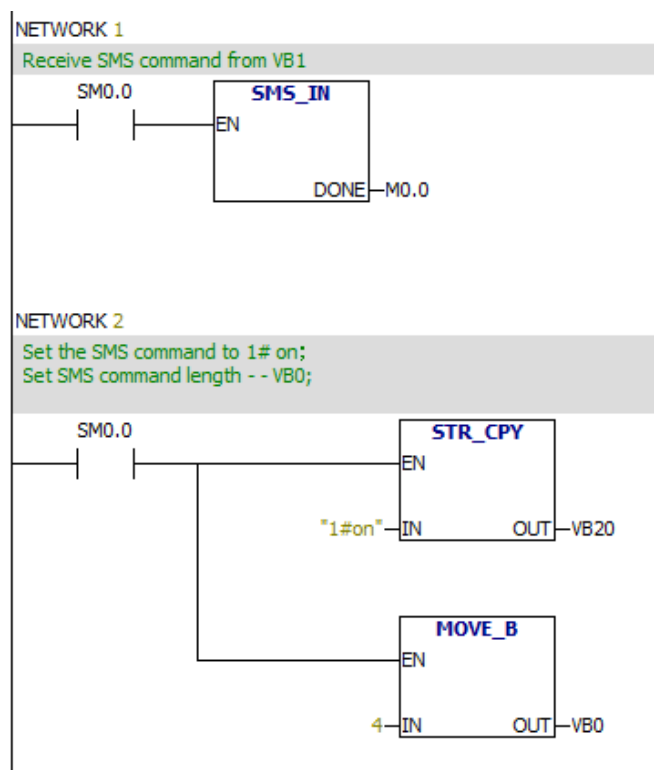
00 01 00 00 00 06 01 10 00 00 00 03

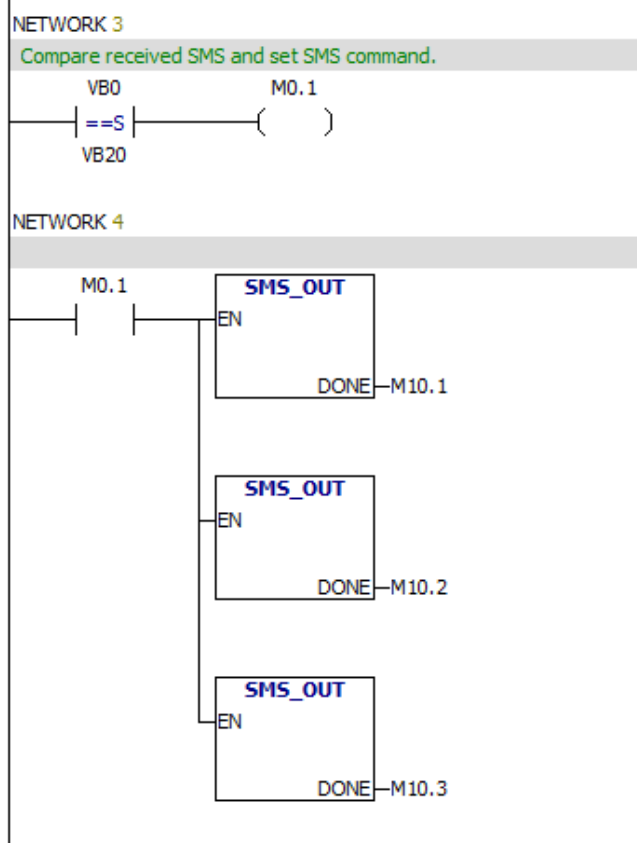
So, if the "Trg" input remains high, then the output will be high also after the CPU gets the above, correct response.

10.17 SMS_IN & SMS_OUT (New block for Ethernet series PLC)

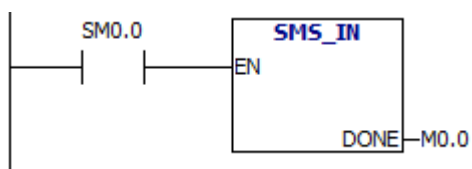


Example:





10.17.1 SMS_IN



Description of Function

They are used in a similar way to other digital input blocks. But they enable the switching of their state via a SMS message sent to the CPU from a phone.

Users can send a pre-set message to the CPU to change the state of the inputs, from “0/OFF” to “1/ON” or vice versa. The ability to define other commands is also provided such as “Switch Pump On”, “Close Drain Valve”, etc.

Configuration of Function

The 'Property' dialog box has a 'Params' tab. It contains the following elements highlighted with red boxes and numbers:

- 1**: A box around the 'Message' section, which includes a 'Start' dropdown set to 'VB 1' and a 'Max count' spinner set to '8'.
- 2**: A box around the 'Phonebook' button, which has a small icon of a phonebook.
- 3**: A box around the 'Telephone' section, which contains five 'Receiver' dropdown menus. The first dropdown is set to '+8618652905032'.
- 4**: A box around the 'Telephone identification' checkbox, which is checked.

At the bottom of the dialog are 'OK', 'Cancel', and 'Apply' buttons.

1. Receive SMS command characters and store them in 8 VBs starting from VB1.
Max count - You can set a certain data length to receive SMS command characters.
2. Phonebook – Clicking this button will display the phonebook and allow editing of it.

The 'Phonebook' dialog box displays a table of contacts with the following data:

| Number | Phone Number/eMail |
|--------|--------------------|
| 01 | +41000000000 |
| 02 | +8618652905032 |
| 03 | |
| 04 | |
| 05 | |
| 06 | |
| 07 | |
| 08 | |
| 09 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |

On the right side of the dialog, there are buttons for 'Add', 'Edit', 'Delete', 'Write', 'Read', 'OK', and 'Cancel'.

3. Telephone Receiver Configuration

In this section you can choose up to a maximum of five contacts from the phonebook for which this block will accept instructions for changing state.

4. Telephone Identification – If this box is ticked, then only SMS received from those configure in the Telephone section will be acted upon.

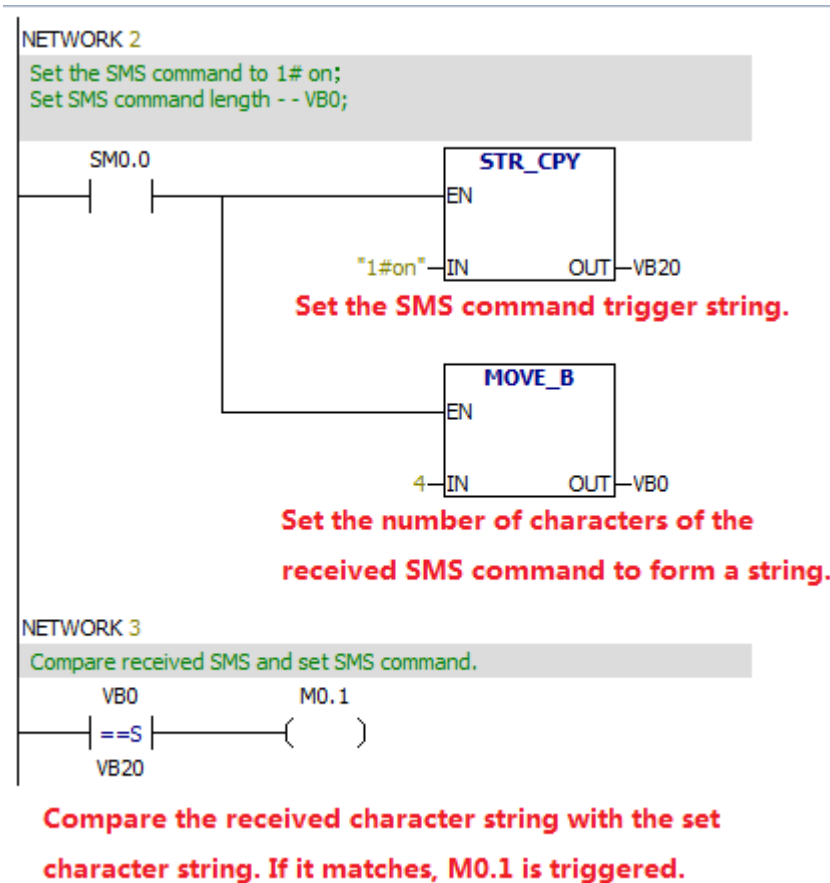
Format of ASCII Constant STRING Data Type:

A string is a sequence of characters, with each character being stored as a byte. The first byte of the string defines the length of the string, which is the number of characters. If the constant string is entered directly in the program editor or data block, then the string must begin and end with double quote characters ("string constant").

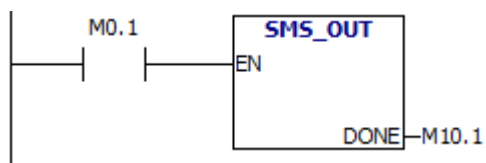
The memory map below shows the format of a STRING data type. The string can have a length of 0 to 254 characters. The maximum length for a string is 255 bytes (254 characters plus the length byte).

| Length | Character 1 | Character 2 | Character 3 | Character 4 | ... | Character 254 |
|--------|-------------|-------------|-------------|-------------|-----|---------------|
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | | Byte 254 |

You need to convert the received SMS characters into a string, as shown in the following example:



10.17.2 SMS_OUT



Description of Function

Each one can send a SMS message or Call (without sound) to all contacts configured in the block. This block is only triggered when there is a LOW to HIGH trigger.

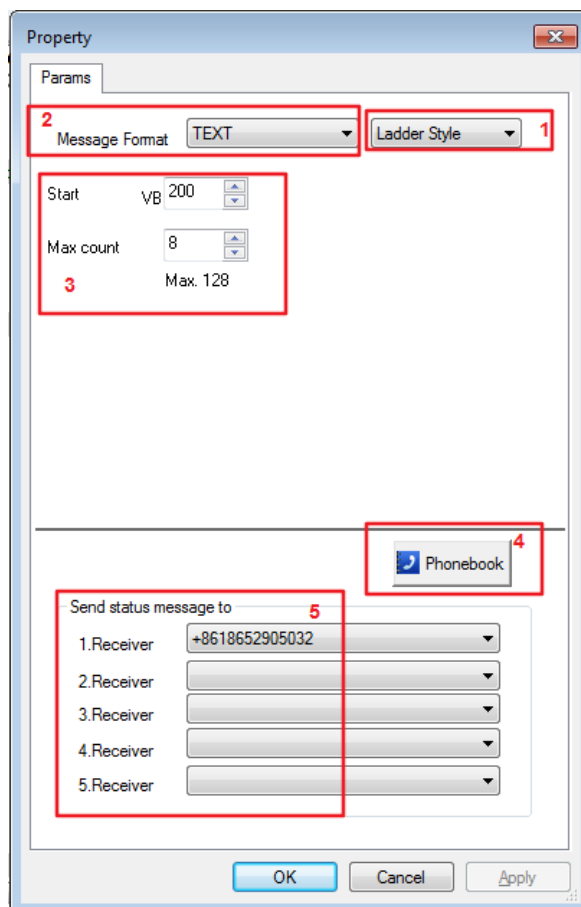
The output pin of this block will only become high once ALL messages have been successfully sent.

NOTICE:

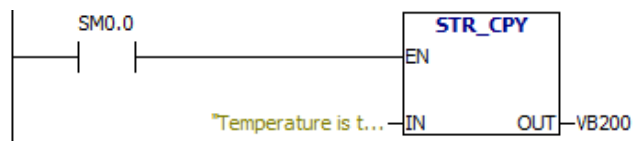
1. At least ONE number MUST be configured.
2. Only use standard ASCII characters in the SMS.
3. Any 'special' characters will cause an error.

Two configuration methods are provided, as shown below:

1). Ladder Style



1. There are two options: Ladder Style and FBD Style.
2. Message Format-TEXT and PDU (UCS2) two options.
3. The string that the PLC will send.



4. Phonebook – Clicking this button will display the phonebook and allow editing of it.
5. Telephone Receiver Configuration
In this section you can choose up to a maximum of five contacts from the phonebook for which this block will accept instructions for changing state.

2) FBD Style

A) Message Format

Property

Params

Message Format: TEXT FBD Style

☒ Text Message ☐ Params Message Set Message

Message Editor

xLadder soft test~!

Remaining chars for message (Max. 100 chars) 81

Phonebook

Send status message to

1.Receiver +8618652905032

2.Receiver

3.Receiver

4.Receiver

5.Receiver

OK Cancel Apply

- a. Text – Support for ASCII characters
- b. PDU – Support for foreign language characters

B) Parameter format

Property

Params

Message Format: TEXT FBD Style

☐ Text Message ☒ Params Message

Set Message

Message Editor

Remaining chars for message (Max. 100 chars): 100

Phonebook

Send status message to

1.Receiver: +8618652905032

2.Receiver

3.Receiver

4.Receiver

5.Receiver

OK Cancel Apply

The message for parameters window should appear:

message for parameters

VB100

VW20

VD30

Email test!

Decimals: 0

Decimals: 0

Decimals: 0

Decimals: 0

☒ Register VB100 Count: 1 Max: 1

☐ Current date/time

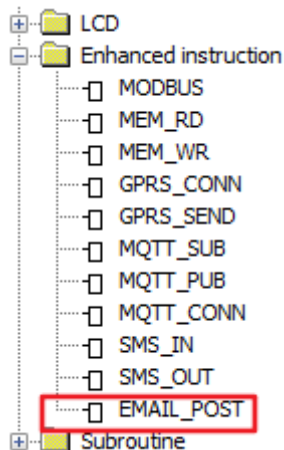
Insert

OK

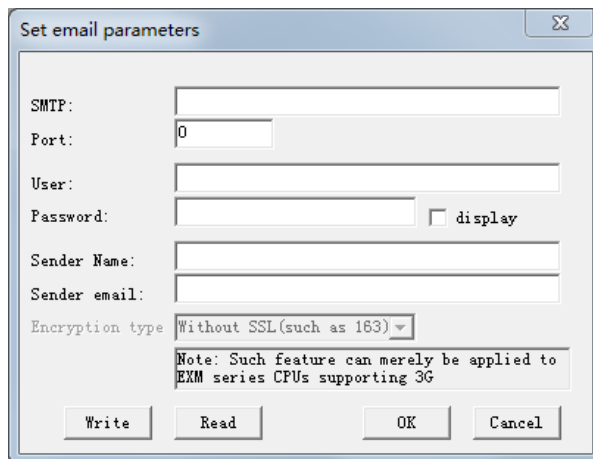
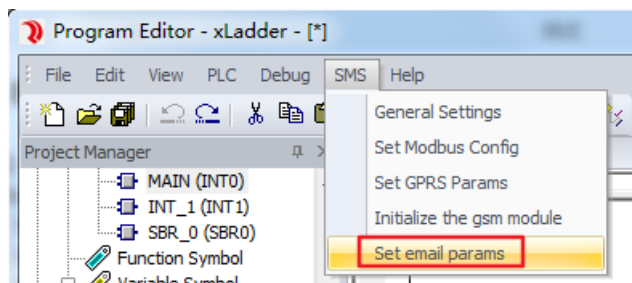
Cancel

- (1) Select the parameter insertion point.
- (2) Enter the parameters to be inserted.
- (3) Click the "Insert" button.
- (4) Click 'OK' to complete the parameter setting.

10.18 EMAIL_POST (New block for Ethernet series PLC)



10.18.1 Set email params



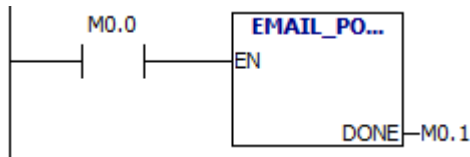
- SMTP: - Enter the IP address of the SMTP Server you are using
- Port: - Enter the port number of the SMTP Server you are using
- User: - Enter the username for SMTP Server email account authentication
- Password: - Enter the password for the SMTP Server email account authentication
- Sender Name: - Enter a name that the receiver will recognise such as Tank Farm Monitor
- Sender Email: - Enter an email address for the PLC

Note that the string cannot contain spaces and some unusual special characters.

Click the Write button to save the settings to the PLC.

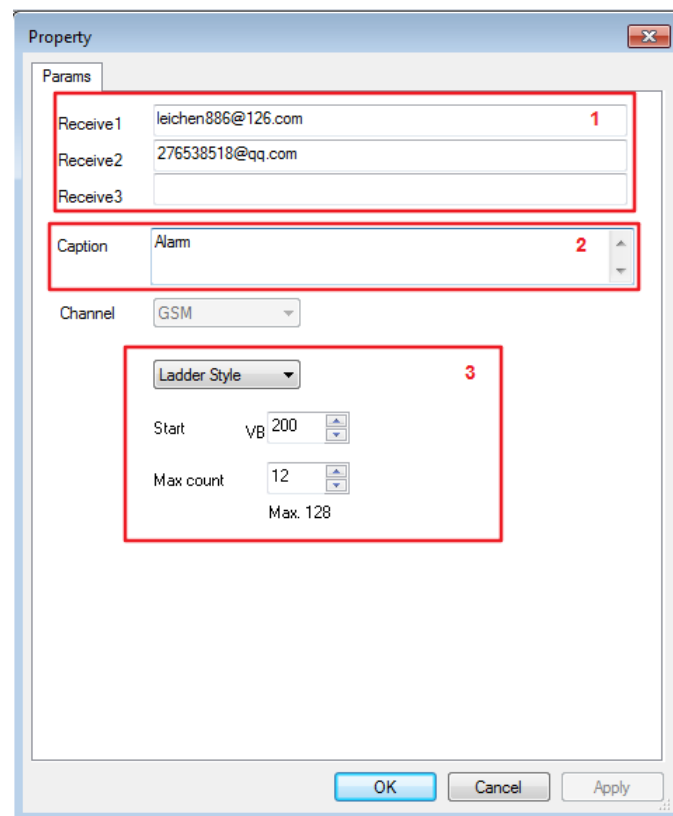
With the settings correctly applied, your PLC is now able to send emails using the EMAIL_POST block within your xLadder program.

10.18.2 EMAIL_POST



When the EN pin has been activated, the CPU will email the address listed in the pre-set receivers:

1). Ladder Style

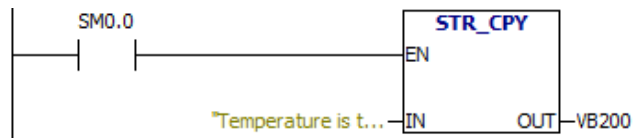


1. Send Message To – Configure up to a maximum of 3 numbers who are to receive the email.

Note that the email name string cannot contain spaces and some unusual special characters.

2. Caption – Email caption. **Note that it cannot contain spaces and some unusual special characters.**

3. The string that the PLC will send.



2) FBD Style

A) Message Format

Property

Params

Receive1: leichen886@126.com

Receive2: 276538518@qq.com

Receive3:

Caption: Email 01

Channel: GSM

FBD Style

☒ Text Message ☐ Params Message

Set Message

Message Editor

Hi,
I am testing xLadder Email function!

Remaining chars for message (Max. 100 chars): 58

OK Cancel Apply

B) Parameter format

Property

Params

Receive1: leichen886@126.com

Receive2: 276538518@qq.com

Receive3:

Caption: Email 02

Channel: GSM

FBD Style

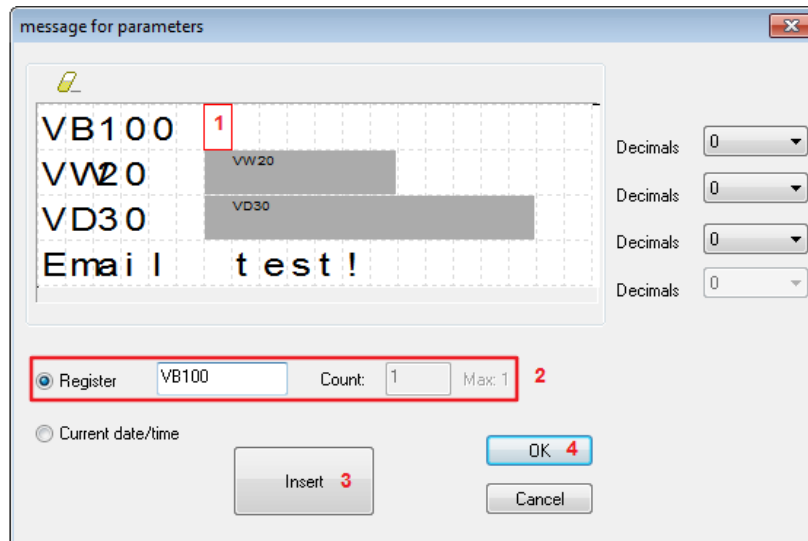
☐ Text Message ☒ Params Message

Set Message

Message Editor

Remaining chars for message (Max. 100 chars): 100

OK Cancel Apply



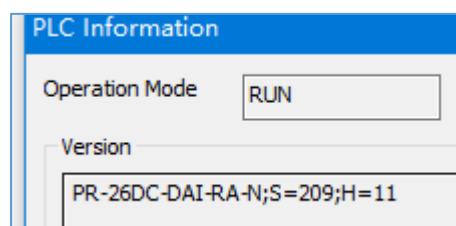
- (1) Select the parameter insertion point.
- (2) Enter the parameters to be inserted.
- (3) Click the "Insert" button.
- (4) Click 'OK' to complete the parameter setting.

10.19 Bacnet Protocol

NOTE:

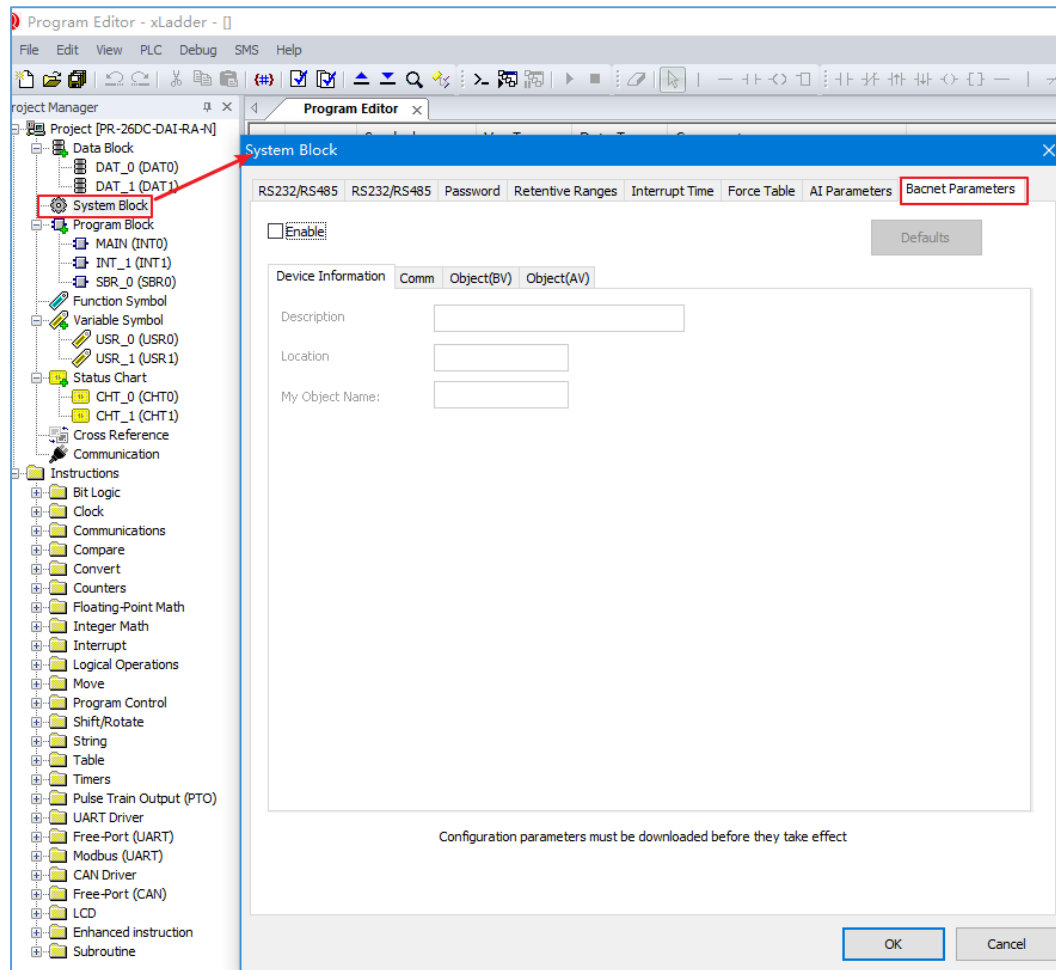
PR Ethernet PLC with firmware version number $\geq V1.57$ supports BACnet. Firmware version ≥ 2.05 , 250 BV and 250 AV can be used. 4G and 4GWIFI models currently do not support BACnet.

1. Firmware version requirements, as shown below:



2. Set PLC BACnet MSTP / BACnet IP

Open the BACnet configuration window, as shown in the figure below:



Divided into 4 pages: 'Device Information', 'Comm', 'Object(BV)' and 'Object(AV)', the following four pages are introduced respectively.

'Device Information' page

This screenshot shows a close-up of the 'Device Information' page. The 'Enable' checkbox is checked. The 'Device Information' tab is selected. The 'Description' field contains the text 'xLadder_bacnet_test'. The 'Location' field contains the text 'rievech'. The 'My Object Name' field contains the text 'RIEVTECH_PLC'.

On this page, you can fill in relevant characters in the 'Description', 'Location', 'My Object Name' directories.

'Description' is limited to 50 characters.

'Location' is limited to 12 characters.

'My Object Name' is limited to 32 characters.

'Comm' page

(1) Bacnet MSTP

| Device Information | | Comm | Object(BV) | Object(AV) |
|--------------------|----------------------|-----------------|------------|------------|
| Protocol | Bacnet MSTP | | | |
| Time out | 10 | (1---30 Second) | | |
| Device number | 8 | (1---4194303) | | |
| Channel | COM2(Built-in RS485) | | | |
| BPS | 9600 | | | |
| Parity | None | | | |
| PLC BACNET MAC | 1 | (0---255) | | |
| BACNET MAC MAX | 127 | (0---255) | | |

'Protocol'

Select BACnet protocol, there are 2 options: 'Bacnet MSTP' for RS485 communication and 'Bacnet IP' for Ethernet communication. Here select 'Bacnet MSTP'.

'Time out'

A value from 1 -- 30 (Second) can be set.

'Device number'

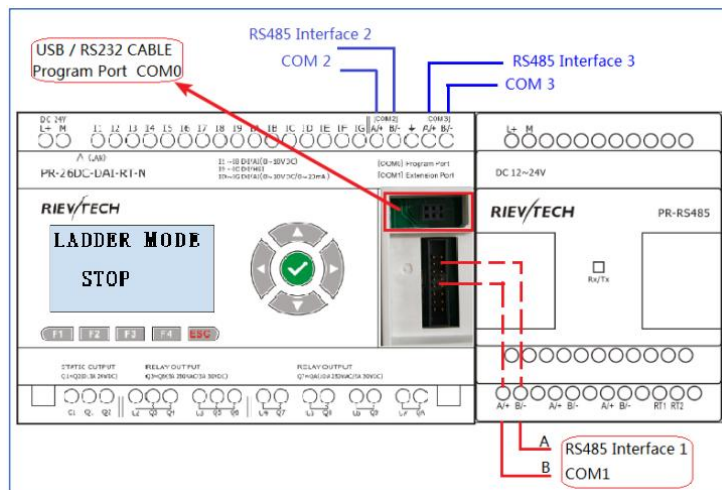
A value from 1 to 41943303 can be set. It has the same function as the device name and is used to identify the device in the network. Be careful not to repeat the device number.

| Device Information | | Comm | Object(BV) | Object(AV) |
|--------------------|-----------------------|---------------|------------|------------|
| Protocol | Bacnet MSTP | | | |
| Time out | 30 | (1---30 Seco | | |
| Device number | 11 | (1---4194303) | | |
| Channel | COM2 (Built-in RS485) | | | |
| BPS | 9600 | | | |
| Parity | None | | | |
| PLC BACNET MAC | 1 | (0---127) | | |
| BACNET MAC MAX | 127 | (0---127) | | |

| Property | Value |
|--------------------|-------|
| MacAddress | 1 |
| SNET | 0 |
| SADR | |
| vendor-name | BACne |
| max-apdu-length... | 1476 |

'Channel'

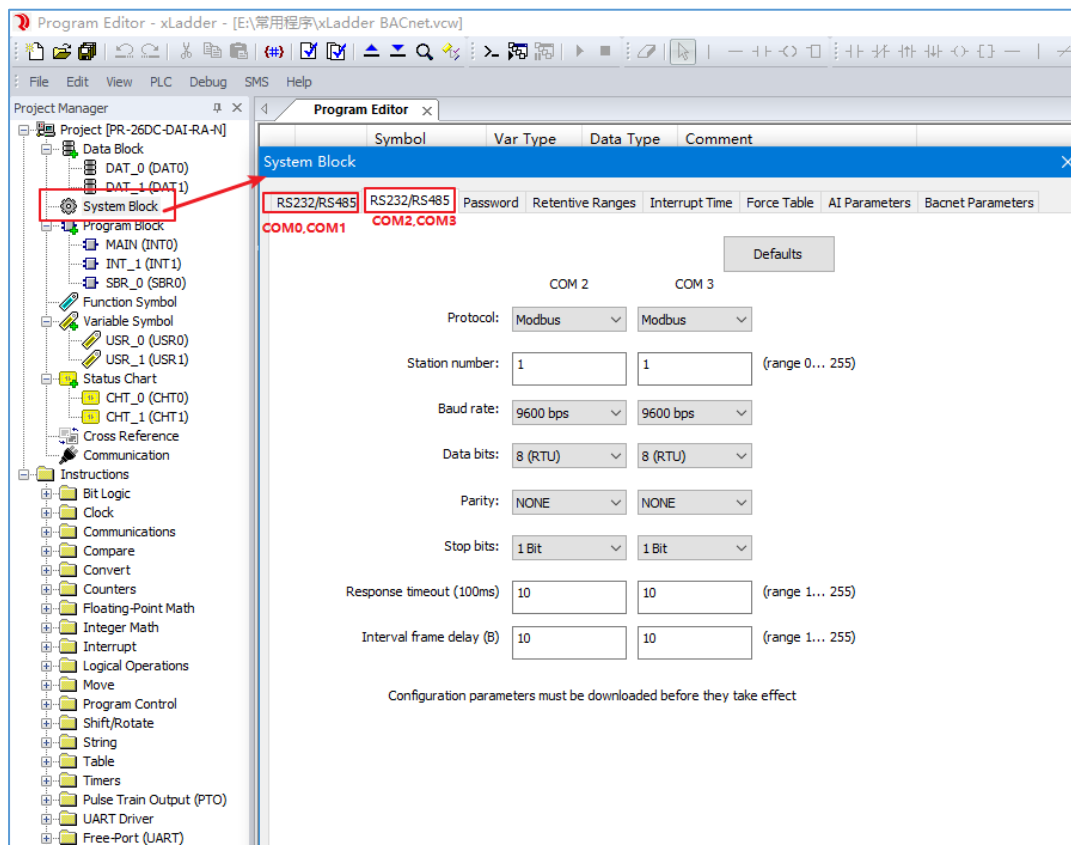
Select the RS485 channel supported by the PLC. The number of RS485 channels supported by different models is also different. The distribution of com ports on the PLC is shown in the figure below, taking PR-26DC-DAI-RT-N as an example:



‘BPS’

Set the baud rate for RS485 communication. Different coms have different baud rate options. The default baud rate of PLC is 9600.

The method to set the baud rate of COM0, COM1, COM2, and COM3 on xLadder is as follows:



‘Parity’

There are three options: None, Odd, and Even, and the PLC defaults to None.

‘PLC BACNET MAC’

It is necessary to ensure that the MAC of each device is unique, and a value from 0 to 127 can be set.

‘BACNET MAC MAX’

The default value is 127, which sets the maximum mac value of BACnet devices in the network.

The input range is 0 --127. Set the appropriate value according to the requirement.

(2) Bacnet IP

Enable

Device Information Comm Object(BV) Object(AV)

Protocol: Bacnet IP

Time out: 10 (1---30 Second)

Device number: 8 (1---4194303)

Port: 47808 (1---65535)

APDU Timeout: 1 (1---100 s)

'Protocol'

Select BACnet protocol, there are 2 options: 'Bacnet MSTP' for RS485 communication and 'Bacnet IP' for Ethernet communication. Here select 'Bacnet IP'.

'Time out'

A value from 1 -- 30 (Second) can be set.

'Device number'

A value from 1 to 41943303 can be set. It has the same function as the device name and is used to identify the device in the network. Be careful not to repeat the device number.

Property

Device Information Comm Object(BV) Object(AV)

Protocol: Bacnet IP

Time out: 30 (1---30 Second)

Device number: 16 (1---4194303)

Port: 47808 (1---65535)

APDU Timeout: 1 (1---100 s)

BACnet

- BACnet Ethernet
- BACnet IP
- Device:16
- BACnet MS/TP

| Property | Value |
|--------------------|------------------------|
| Address | 192.168.0.222:47808 |
| SNET | 0 |
| SADR | BACnet Stack at Source |
| vendor-name | BACnet Stack at Source |
| max-apdu-length... | 1476 |

'Port'

The default is 47808.

'APDU Timeout'

A value from 1 -- 100s can be set, and the default value is 1.

'Object(BV)' page

Enable

Device Information Comm Object(BV) Object(AV)

Add Delete

Binary Value, 0

Binary Value, 1

Binary Value, 2

Binary Value, 3

Binary Value, 4

Binary Value, 5

Binary Value, 6

Binary Value, 7

Binary Value, 8

Binary Value, 9

Binary Value, 10

Binary Value, 11

Binary Value, 12

Binary Value, 13

Binary Value, 14

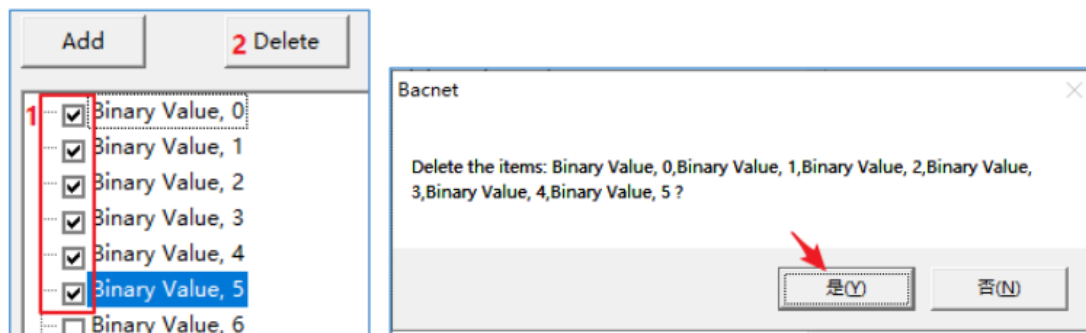
BV 0

Name: M0.0

Register: M0.0

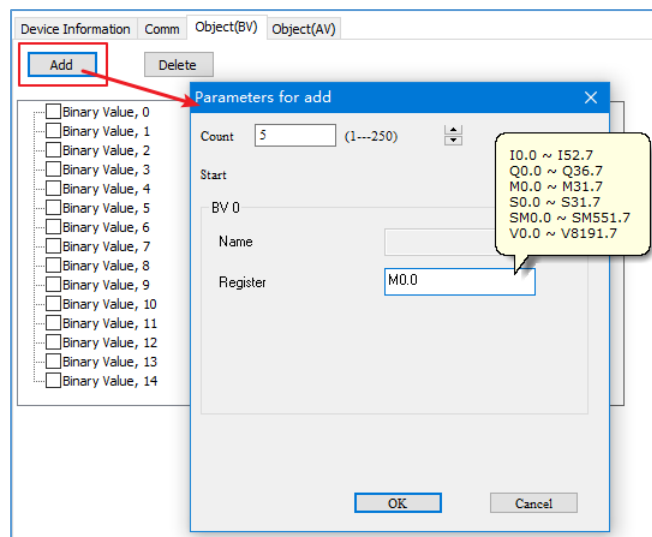
Delete BV

Check the checkbox in front of the BV that needs to be deleted (multiple selections are allowed), and then click the 'Delete' button above.



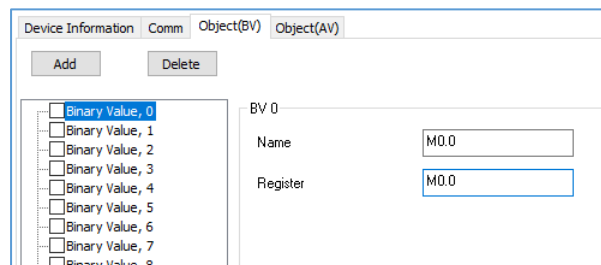
Add BV

Click the 'Add' button above.



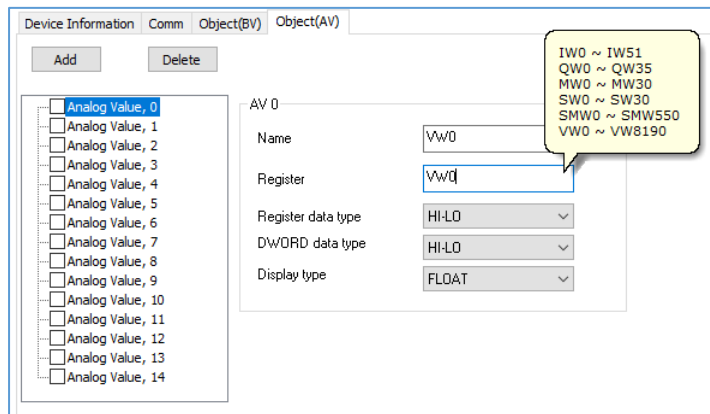
Count: Set the number of BV to add. Note that the total number of BV will not exceed 250. For example, if some BV already exists, even if 250 is filled in here, the number of newly added BV will not be 250, but 250 minus the existing number.

Register: You can choose I, Q, M, S, SM, V or empty. With the 'Count' above, multiple BV of the same register can be added at one time.



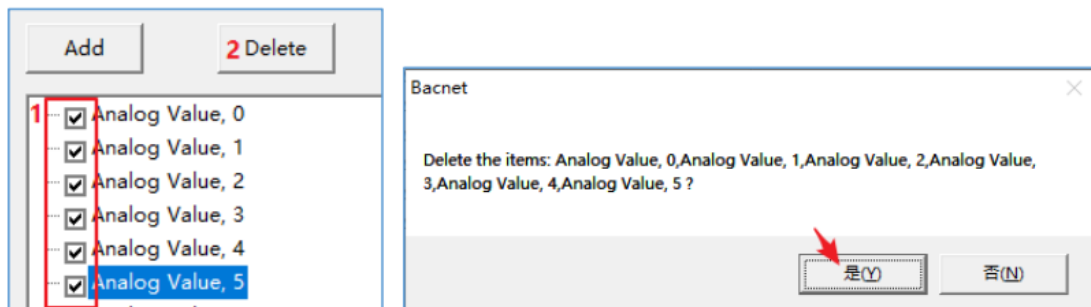
First select 'Binary Value, 0' in the list on the left, set the name of the object on the right, and assign a register to it. 'Name' only allows up to 8 commonly used characters.

'Object(AV)' page



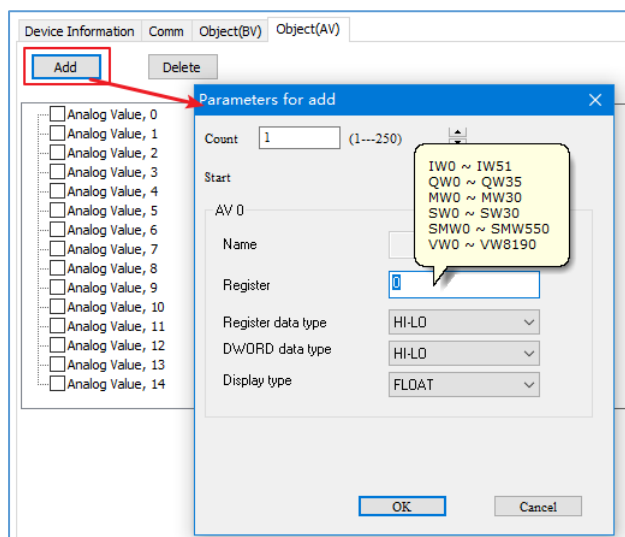
Delete AV

Check the checkbox in front of the AV that needs to be deleted (multiple selections are allowed), and then click the 'Delete' button above.



Add AV

Click the 'Add' button above.



Count: Set the number of AV to add. Note that the total number of AV will not exceed 250. For example, if some AV already exists, even if 250 is filled in here, the number of newly added AV will not be 250, but 250 minus the existing number.

Register: You can choose IW, QW, MW, SW, SMW, VW or empty. With the 'Count' above, multiple AV of the same register can be added at one time.

'Register data type'

The byte order when providing the value in register, you can choose 'HI-LO' and 'LO-HI' modes.

'DWORD data type'

The byte order of DWORD data, you can choose 'HI-LO' and 'LO-HI' modes.

'Display type'

Select the data format when displaying values, only support 'FLOAT'.

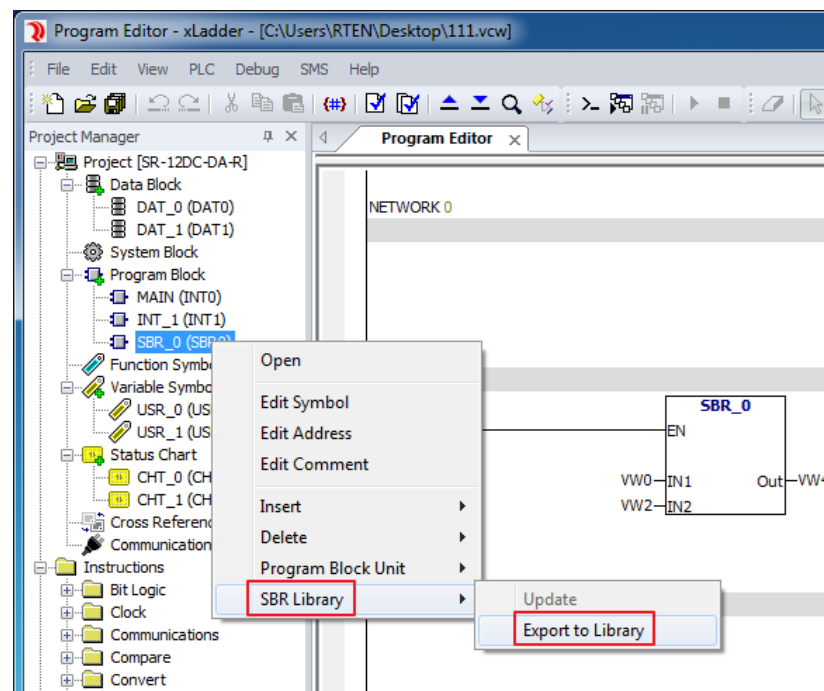
After configuring all the parameters, click the 'OK' button in the lower right corner, as shown in the figure below. Then download the BACnet settings to the PLC along with the xLadder program, and the settings will take effect after the PLC restarts.

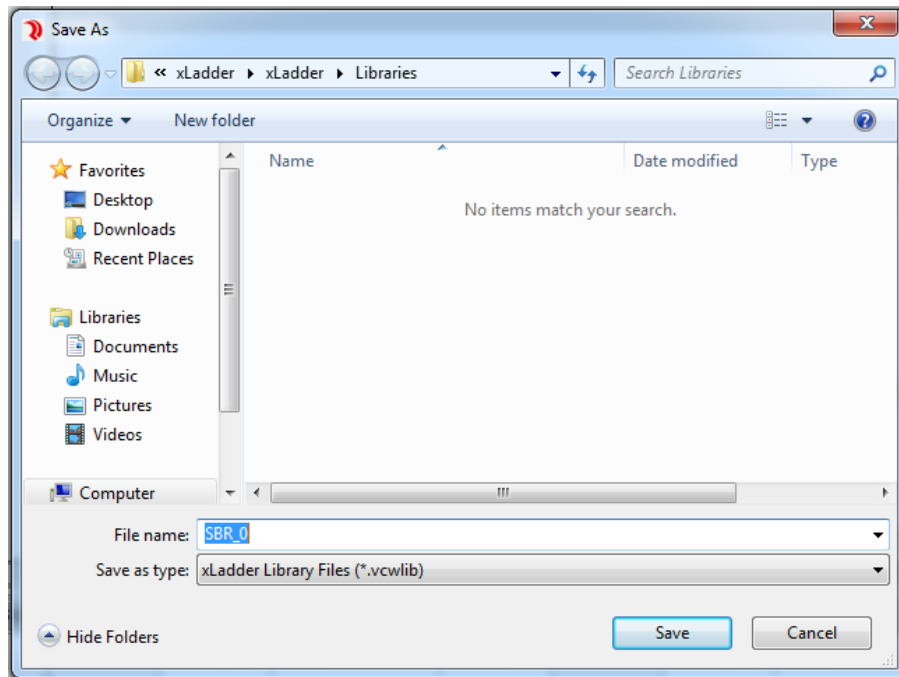
For how to use the BACnet protocol of RIEVRECH HMI, please refer to the document on the website – 'Rievtech PLC BACnet MSTP BACnet IP parameter configuration method and usage examples-V2.pdf', download address: <https://www.rievtech.com/download.html>

10.20 Libraries files function

The created subroutine can be exported into a library file.

(1) Select the subroutine and then use the 'Export to Library' menu to export the subroutine.



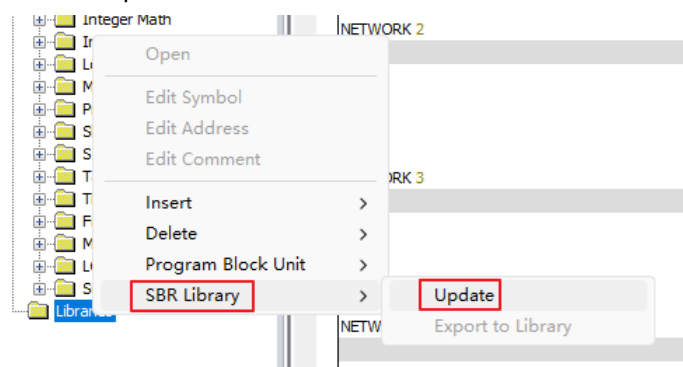


Select the storage path for the library file, name the library file, and then click the 'Save' button.

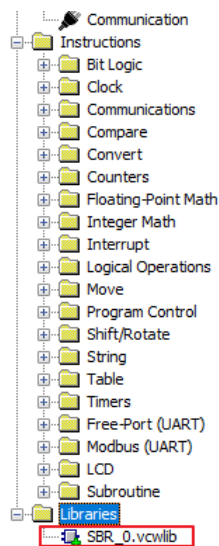
(2) Add library files to xLadder.

Copy the library file to the xLadder default library file storage path. The path is "C:\Program Files (x86)\RIEVIEWTECH\xLadder\xLadder\Libraries".

Select the 'Library' directory and right-click the mouse to pop up the right-click menu. Select and use the 'Update' menu.



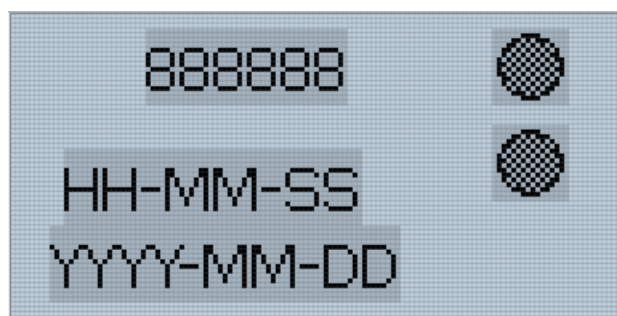
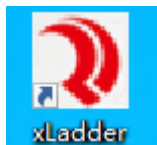
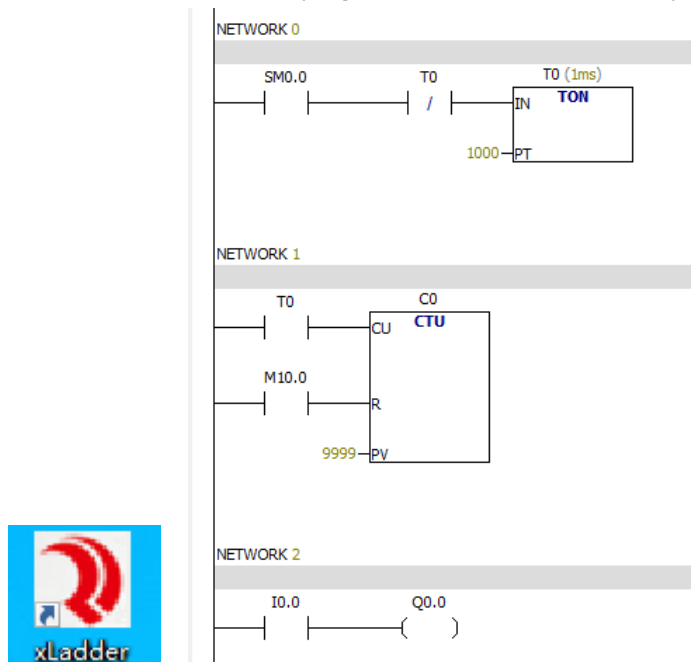
At this time, the added library files will appear in the 'Library' directory.



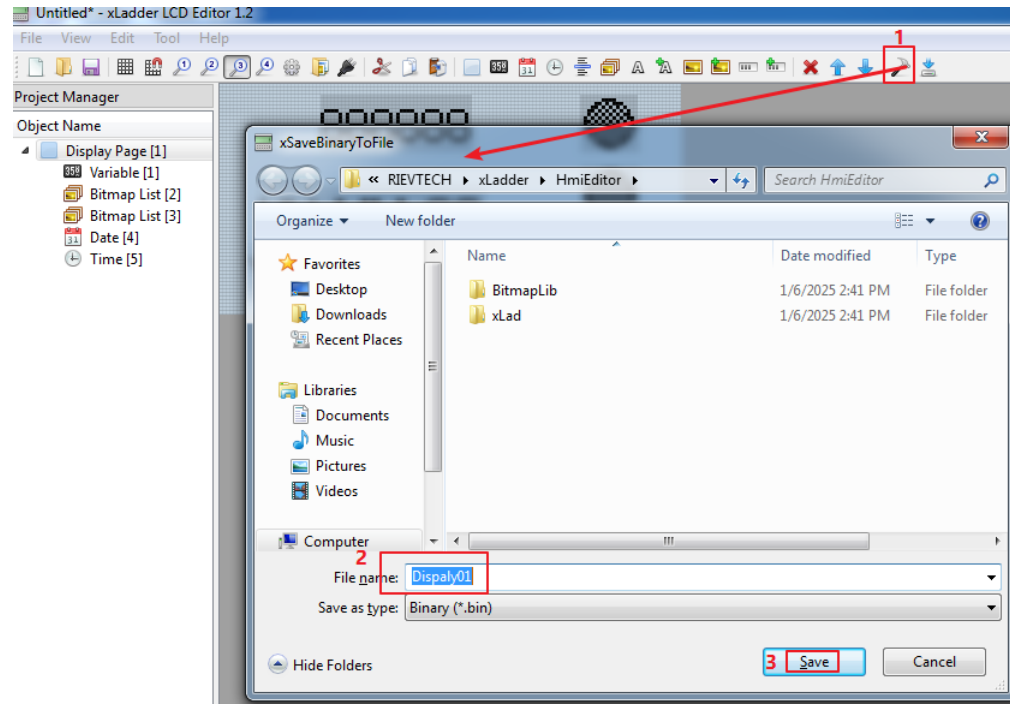
Double-click to add it to the program.

10.21 Offline simulation

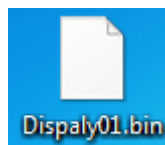
1. xLadder creates ladder program, HmiEditor creates LCD panel program.



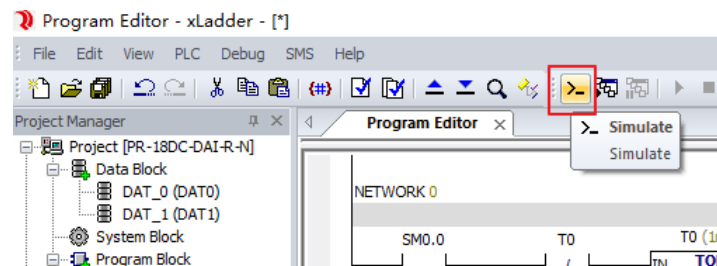
2. Generate a bin file on the HmiEditor software.



A bin file will be generated under the selected path, as shown in the figure below:

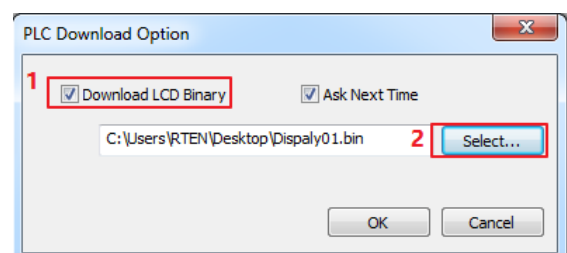


3. Perform simulation operations on xLadder software.



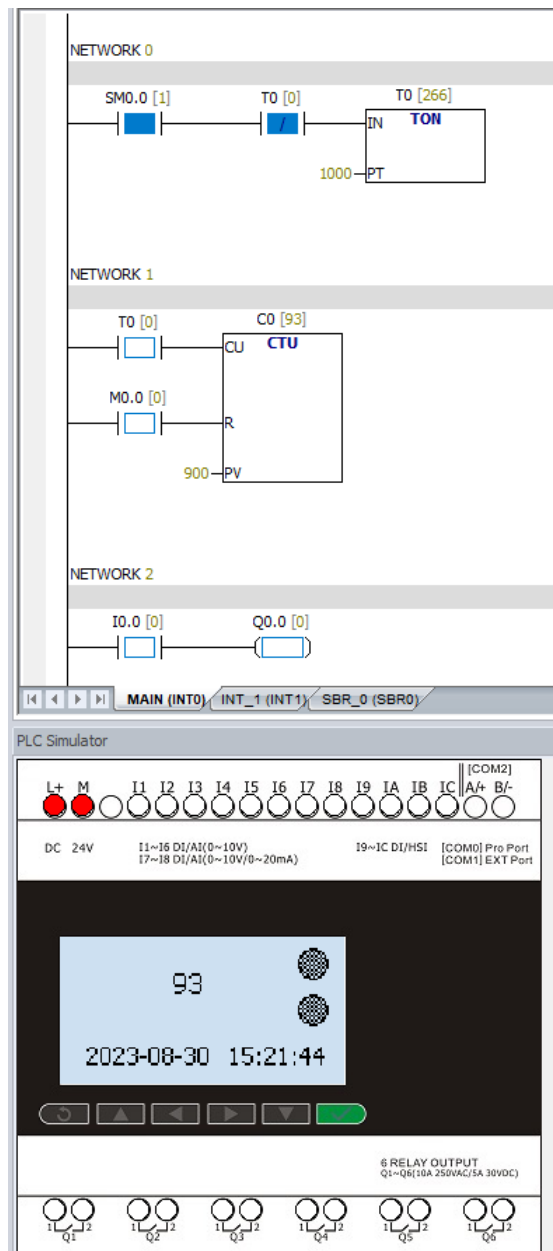
Click the 'Simulate' button in the toolbar to pop up the following dialog box, check 'Download LCD Binary', and then select the HmiEditor software to generate the bin file.

In addition, it should be noted that this step is also performed when downloading the xladder program to the PLC. The bin file of the LCD panel can be downloaded to the PLC together. When 'Download LCD Binary' is not checked, the bin file of the LCD panel will not be downloaded.

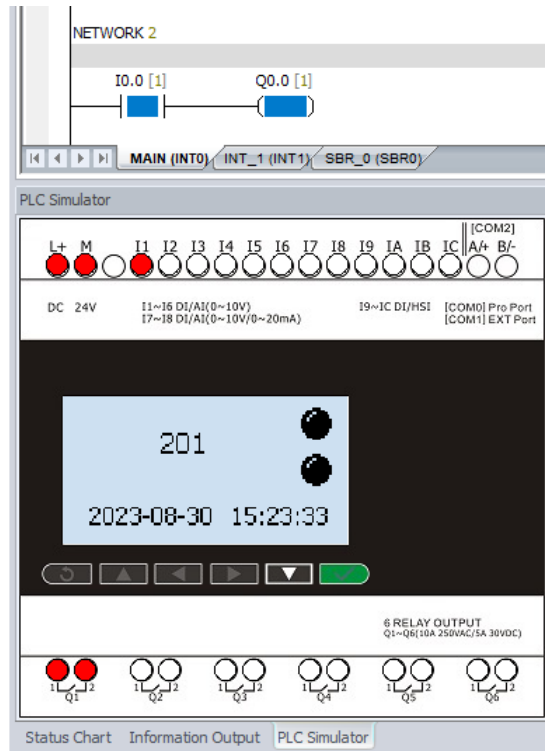


Then click the 'OK' button in the picture above.
You can also choose not to download LCD Binary.

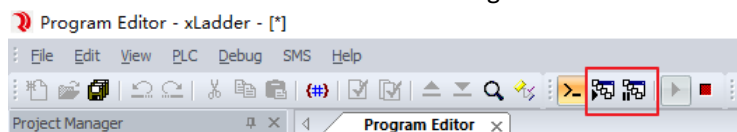
4. The xLadder software enters the simulation state, as shown below:



You can click on I1 on the picture, I0.0 in the program will change to '1', and Q0.0 will also change to '1', as shown below:



5. The two tool buttons in the figure below can pause the simulation and continue the simulation. Click the 'Simulate' button in the toolbar again to exit the simulation.



10.22 C language subroutine

NOTE:

Supports standard C language and its library functions (need to add header files).

(1) Supported models and firmware versions

Model list:

PR-12AC-R-N

PR-12DC-DA-R-N

PR-18AC-R-N

PR-18DC-DAI-R-N

PR-18DC-DAI-TN-N

PR-26AC-R-N

PR-26DC-DAI-RA-N

PR-26DC-DAI-RT-WIFI

PR-26DC-DAI-RT-N

PR-23DC-PTDAI-RT-N

Firmware version requirements:

The firmware version of the above models needs to be greater than or equal to V2.07.

(2) Instructions for use

a) **Special function description**

PLC variable reading function

| Function | Description |
|----------------------|----------------------------------|
| GetU8(R, O) | Get 8-bit unsigned data |
| GetU16(R, O) | Get 16-bit unsigned data |
| GetU32(R, O) | Get 32-bit unsigned data |
| GetS8(R, O) | Get 8-bit signed data |
| GetS16(R, O) | Get 16-bit signed data |
| GetS32(R, O) | Get 32-bit signed data |
| GetF32(R, O) | Get 32-bit floating point number |
| GetT(R, O, B) | Get 1 digit data |

PLC variable writing function

| Function | Description |
|-------------------------|------------------------------------|
| SetU8(R, O, V) | Write 8-bit unsigned data |
| SetU16(R, O, V) | Write 16-bit unsigned data |
| SetU32(R, O, V) | Write 32-bit unsigned data |
| SetS8(R, O, V) | Write 8-bit signed data |
| SetS16(R, O, V) | Write 16-bit signed data |
| SetS32(R, O, V) | Write 32-bit signed data |
| SetF32(R, O, V) | Write 32-bit floating point number |
| SetT(R, O, B, V) | Write 1 digit data |

Function parameter description:

R: PLC variable area, support I、Q、M、S、SM、V、T、C、HC、AI、AQ、L、AC。

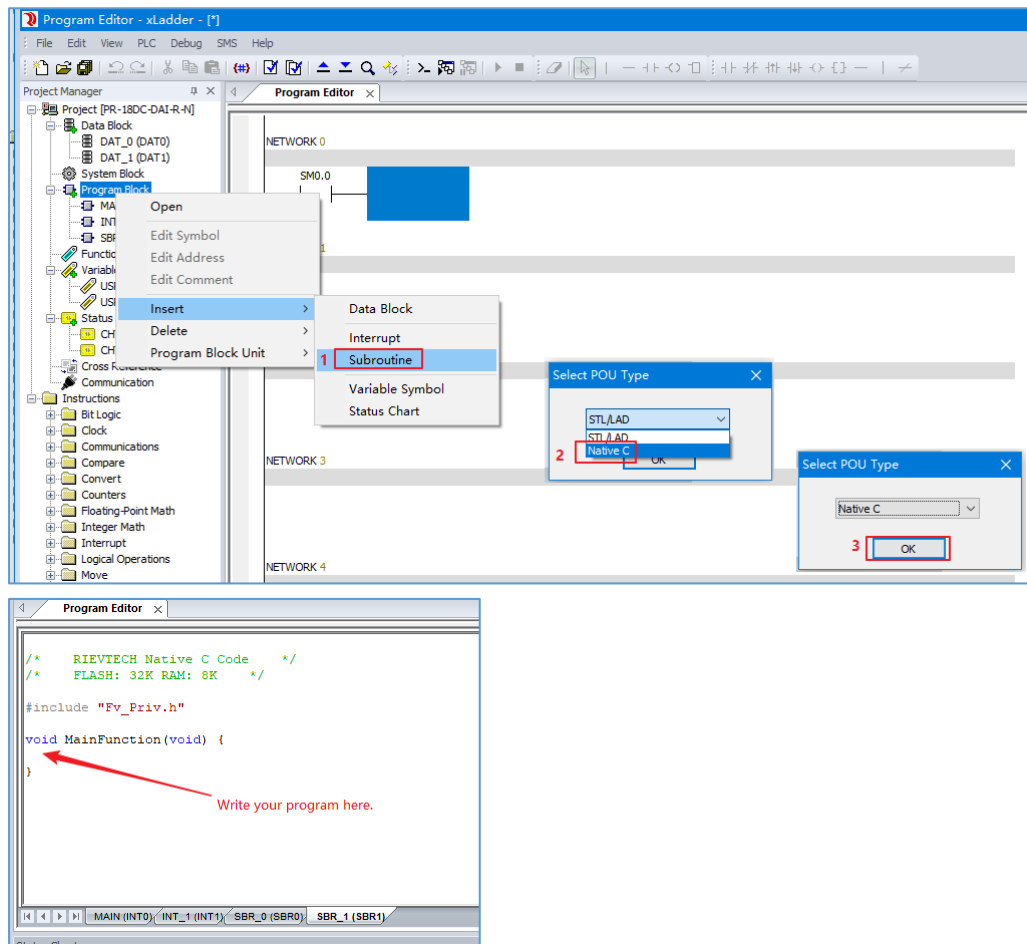
O: Offset of PLC variable, unit: byte.

B: Bit offset of PLC variable.

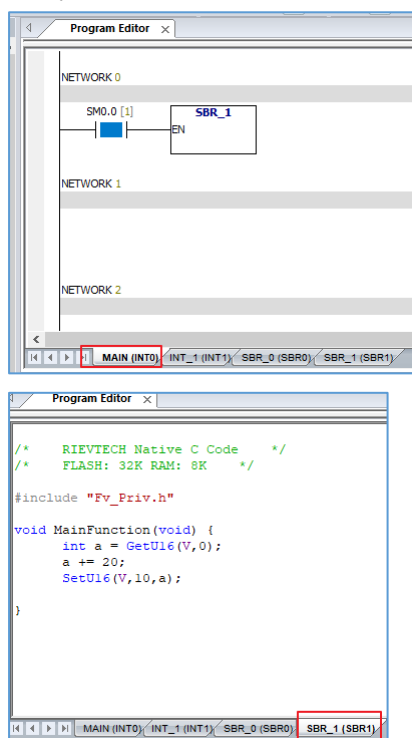
V: The value of the PLC variable.

b) **Example description**

Create a C language subroutine. The steps are as follows:



Such as the following sample program:
Compile and download to PLC.



| MAIN (INT0) / INT_1 (INT1) / SBR_0 (SBR0) / SBR_1 | | | |
|---|-----------|-------|-----|
| Status Chart | | | |
| Address | Data Type | Value | For |
| VW0 | INT | 11 | |
| VW10 | INT | 31 | |

C language subroutines do not support offline simulation yet.

(3) Use the 'Find And Replace' menu under 'Edit', or use the 'Ctrl + F' shortcut to open the Find/Replace window.

For example, after entering VW0 in the 'Find What' area, the search results will list all the locations where VW0 is used, as shown below:

The screenshot shows the 'Program Editor' window with the following C code:

```

/*  RIEVTECH Native C Code  */
/*  FLASH: 32K RAM: 8K  */

#include "Fv_Priv.h"

void MainFunction(void) {

    int a,b,c;

    a = GetUI6(V,0);
    b = GetUI6(V,2);

    c = a+b;

    SetUI6(V,4,c);

}

```

Overlaid on the code is the 'Find And Replace' dialog box. The 'Find What' field contains 'vw0'. The 'Find Up' radio button is selected. The 'Status' tab is active, showing the following search results:

| Status | Location |
|--------|-----------------------------------|
| | MAIN (INT0) NETWORK 0 Col 2 Row 0 |
| | SBR_1 (SBR1) Ln 15 |
| | SBR_2 (SBR2) Ln 11 |

At the bottom of the dialog box are buttons for 'Next', 'Replace', and 'Replace All'.

You can also replace the searched content with the content in the 'Replace With' area of this window.

MODBUS ADDRESS

| Bit | | | | |
|-----------|----------------|-------------|-------------|-----|
| | | Modbus Adr. | Modbus code | R/W |
| I | I0.0--I31.7 | 0--255 | 02 | R |
| SM | SM0.0--SM551.7 | 300--4715 | 02 | R |
| T | T0-T255 | 5000--5255 | 02 | R |
| C | C0-C255 | 5500--5755 | 02 | R |
| Q | Q0.0--Q31.7 | 0--255 | 01/05 (15) | R/W |
| M | M0.0--M31.7 | 320--575 | 01/05 (15) | R/W |
| S | S0.0--S31.7 | 680--935 | 01/05 (15) | R/W |
| V | V0.0--V8063.7 | 1024--65535 | 01/05 (15) | R/W |

| Word | Version number requirements: PR Series >= 15; PR-N Series >= 154 | | | |
|------------|--|-------------|-------------|-----|
| | | Modbus Adr. | Modbus code | R/W |
| VW | VW0--VW8190 | 0--4095 | 03/06 (16) | R/W |
| MW | MW0--MW30 | 4100--4115 | 03/06 (16) | R/W |
| SMW | SMW0--SMW550 | 4150--4425 | 03/06 (16) | R/W |
| SW | SW0--SW30 | 4500--4515 | 03/06(16) | R/W |
| T | T0-T255 | 4550--4805 | 03/06 (16) | R/W |
| C | C0-C255 | 4850--5105 | 03/06 (16) | R/W |
| AIW | AIW0--AIW178 | 0--89 | 04 | R |
| AQW | AQW0--AQW178 | 200--289 | 04 | R |
| IW | IW0--IW30 | 300--315 | 04 | R |
| QW | QW0--QW30 | 400--415 | 04 | R |

* Please refer to Chapter '[Extension module address](#)' for a detailed list of I, O, AIW, AQW, etc.

Example

| RIEVTECH-PLC(LAD MODE) | Modbus Adr. | RIEVTECH-PLC(LAD MODE) | Modbus Adr. | RIEVTECH-PLC(LAD MODE) | Modbus Adr. |
|------------------------|-------------|------------------------|-------------------|------------------------|-------------------|
| VW0 | 0 | | Base address:4100 | | Base address:4150 |
| VW2 | 1 | NW0 | 4100 | SMW0 | 4150 |
| VW4 | 2 | NW2 | 4101 | SMW2 | 4151 |
| VW6 | 3 | NW4 | 4102 | SMW4 | 4152 |
| VW8 | 4 | NW6 | 4103 | SMW6 | 4153 |
| VW10 | 5 | NW8 | 4104 | SMW8 | 4154 |
| VW12 | 6 | NW10 | 4105 | SMW10 | 4155 |
| . | . | NW12 | 4106 | SMW12 | 4156 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| VW150 | 75 | . | . | . | . |
| VW152 | 76 | NW30 | 4115 | SMW550 | 4425 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| VW8190 | 4095 | . | . | . | . |

| RIEVTECH-PLC(LAD MODE) | Modbus Adr. | RIEVTECH-PLC(LAD MODE) | Modbus Adr. |
|------------------------|-------------------|------------------------|-------------------|
| | Base address:4550 | | Base address:4850 |
| T0 | 4550 | C0 | 4850 |
| T1 | 4551 | C1 | 4851 |
| T2 | 4552 | C2 | 4852 |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| T255 | 4805 | C255 | 5105 |

For example:

