



Installation and wiring
Programming
Configuring software
Applications
Technical data

RIEVTECH USER MANUAL

+ Programming guide

Ver. 2.0.0.1

Introduction-----	1
Valid range of this manual-----	2
Safety Guideline-----	2
Qualified Personnel-----	3
Prescribed Usage-----	3
Warning-----	3
Trademarks-----	3
Copyright Rievtech 2016 all rights reserved-----	4
Disclaim of Liability-----	4
Additional support-----	4
1 .what is Xlogic ?-----	5
1.1 Overview-----	5
1.2 Highlight feature-----	5
1.3 Some of the things xLogic can do for you?-----	7
1.4 xLogic devices: -----	7
xLogic Basic is available in two voltage classes-----	7
Expansion modules-----	8
Communication cable and module-----	8
2. Hardware introduction-----	11
2.1 Naming Rules of PR Series PLC-----	11
2.2 Hardware model selection-----	11
2.3 Structure & dimension-----	13
3 .Installing/removing xLogic-----	16
Dimensions-----	16
3.1 DIN rail mounting-----	17
3.2 Wall-mounting-----	18
3.3 wiring xLogic-----	21
3.4 Connecting the power supply-----	21
3.4.1 Connecting xLogic inputs-----	22
3.4.2 Connecting xLogic Outputs-----	25
3.4.3 Communication port instructions:-----	28
4.Quick reference manual-----	30
4.1 Special memory area:-----	30
4.2Interrupt Events:-----	30
4.3 High speed counter: -----	30
X Ladder direction for use-----	32
5.The detailed annotation of operation interface-----	32
5.1The main menu-----	32
5.1.1 File-----	32
5.1.2 Edit-----	33
5.1.3 View-----	33
5.1.4 PLC-----	35
5.1.5 Debug-----	36
5.1.6 Help-----	36

5.2 toolbar-----	37
5.3 Instruction tree-----	40
5.3.1 Project-----	41
5.3.2 Data block-----	41
5.3.3 System block-----	42
5.3.4 Program block-----	48
5.3.5 Function symbol-----	48
5.3.6 Variable symbol-----	48
5.3.7 Status chart-----	49
5.3.8 Cross reference-----	49
5.3.9 Communication-----	50
5.3.10 Instructions-----	51
5.3.11 The program editor-----	52
5.3.12 Status chart, information output-----	53
5.4 Programming concepts-----	54
5.4.1 How the program works-----	54
5.4.2 Addressing overview-----	54
5.4.3 How to organize the program-----	56
5.5 How to enter the ladder logic program-----	57
5.5.1 How to build a new project-----	57
5.5.2 Ladder logic element and its working principle-----	57
5.5.3 Network rules for series and parallel in LAD-----	58
5.5.4 How to input commands in LAD-----	58
5.5.5 How to enter the address in LAD-----	60
5.5.6 How to edit program elements in LAD-----	60
5.5.7 How to use find / replace-----	63
5.5.8 How to display errors in LAD in the program editor-----	64
5.5.9 How to compile in LAD-----	64
5.5.10 How to save the project-----	65
5.6 How to set up a communication and download program-----	65
5.6.1 Communication settings-----	65
5.6.2 Download program-----	67
5.6.3 How to correct compilation errors and download errors-----	68
5.7 How to monitor and debug the program-----	69
5.8 PLC operation and options-----	73
6.X Ladder instructions descriptions-----	74
6.1 Bit logic-----	74
6.1.1 Normally open and normally closed-----	74
6.1.2 Normally open immediate and normally closed immediate-----	76
6.1.3 NOT Reverse instruction-----	76
6.1.4 Rising edge and falling edge-----	77
6.1.5 Output-----	78
6.1.6 Output immediate-----	78
6.1.7 Set and reset-----	79

6.1.8 Set immediate and reset immediate-----	80
6.1.9 SR instruction-----	80
6.1.10 RS instruction-----	81
6.1.11 NOP instruction-----	82
6.2 Clock instruction-----	83
6.2.1 Read and set the real time clock-----	83
6.3 Communication-----	84
6.3.1 Get port address-----	84
6.3.2 Set port address-----	84
6.4 Compare-----	85
6.4.1 Byte compare-----	85
6.4.2 Integer comparison-----	87
6.4.3 Double integer comparison-----	88
6.4.4 Real number comparison-----	89
6.4.5 String comparison-----	90
6.5 Convert-----	91
6.5.1 Byte to integer-----	91
6.5.2 Integer to byte-----	92
6.5.3 Integer to double integer-----	92
6.5.4 Integer to string-----	92
6.5.5 Double integer to integer-----	94
6.5.6 Double integer to real number-----	94
6.5.7 Double integer to string-----	95
6.5.8 BCD to integer, integer to BCD conversion-----	96
6.5.9 ROUND-----	97
6.5.10 TRUNC-----	98
6.5.11 Real number to string-----	99
6.5.12 Integer to ASCII code-----	101
6.5.13 Double integer to ASCII code-----	103
6.5.14 Real number to ASCII code-----	104
6.5.15 ATH&HTA-----	106
6.5.16 String to integer-----	107
6.5.17 String to double integer-----	109
6.5.18 String to real number-----	111
6.5.19 DECO-----	113
6.5.20 ENCO-----	114
6.5.21 Seven segment code-----	115
6.6 Counter-----	116
6.6.1 CTU-----	116
6.6.2 CTD-----	117
6.6.3 CTUD-----	118
6.7 Floating point calculation-----	119
6.7.1 ADD-R&SUB-R-----	119
6.7.2 MUL - R&DIV - R-----	121

6.7.3	SQRT	122
6.7.4	SIN	123
6.7.5	COS	124
6.7.6	TAN	125
6.7.7	LN	126
6.7.8	EXP	127
6.7.9	PID	128
6.8	Integer operations	131
6.8.1	ADD-I&SUB-I	131
6.8.2	ADD- DI & SUB- DI	133
6.8.3	MUL & DIV	134
6.8.4	MUL -I & DIV-I	135
6.8.5	MUL -DI & DIV -DI	137
6.8.6	INC-B & DEC-B	138
6.8.7	INC-W & DEC-W	139
6.8.8	INC -DW & DEC -DW	140
6.9	Interrupt	141
6.9.1	ENI & DISI	141
6.9.2	RETI instruction	143
6.9.3	ATCH	144
6.9.4	DTCH	146
6.9.5	Clear interrupt event	148
6.10	Logic operation	149
6.10.1	INV -B	149
6.10.2	INV -W	150
6.10.3	INV -DW	151
6.10.4	WAND-B、WOR -B、WXOR -B	152
6.10.5	WAND-W、WOR -W、WXOR -W	153
6.10.6	WAND- DW、WOR -DW、WXOR -DW	154
6.11	Move	155
6.11.1	Byte move	155
6.11.2	Word move	156
6.11.3	Double word move	157
6.11.4	Real number move	158
6.11.5	BLKMOV -B	159
6.11.6	BLKMOV -W	160
6.11.7	BLKMOV -D	161
6.11.8	SWAP	162
6.11.9	MOV -BIR	163
6.11.10	MOV -BIW	163
6.12	Program control	164
6.12.1	FOR、NEXT	164
6.12.2	Jump to label and label	166
6.12.3	Sequence control relay	167

6.12.4 Return from subroutine-----	169
6.12.5 Conditional end-----	170
6.12.6 STOP-----	171
6.12.7 Watchdog Reset-----	172
6.12.8 Diagnosis LED-----	173
6.13 Shift cycle-----	174
6.13.1 SHR -B & SHL -B-----	174
6.13.2 SHR -W & SHL -W-----	176
6.13.3 SHR -DW & SHL -DW-----	177
6.13.4 ROR -B & ROL -B-----	178
6.13.5 ROR -W & ROL -W-----	179
6.13.6 ROR -DW & ROL -DW-----	180
6.13.7 SHRB-----	181
6.14 Character string-----	182
6.14.1 String length-----	182
6.14.2 Copy string-----	183
6.14.3 SSTR-CPY-----	184
6.14.4 String catenate-----	185
6.14.5 STR -FIND-----	186
6.14.6 Look for the first character in the string-----	187
6.15 Table-----	188
6.15.1 Last in first out-----	188
6.15.2 FIFO-----	190
6.15.3 Add to table-----	192
6.15.4 Memory fill-----	194
6.15.5 Table Find-----	195
6.16 Timer-----	197
6.16.1 Switch on delay timer-----	197
6.16.2 TONR-----	199
6.16.3 Disconnect delay timer-----	200
6.16.4 Start time interval-----	201
6.16.5 Calculation interval time-----	202
6.17 Pulse train output-----	203
6.17.1 Pulse output-----	203
6.17.2 Pulse width modulation-----	205
6.18 Subroutine-----	206
6.18.1 Using subroutine-----	206
6.18.2 Using parameters to call subroutine-----	207
6.18.3 How to set up a subroutine-----	208
6.18.4 How to call a subroutine-----	209
7.PLC storage area-----	212
7.1 Storage area types and properties-----	212
7.2 Direct and indirect addressing-----	213
7.3 Bit, byte, word and double word access-----	216

7.4 Memory address range-----	216
7.5 Data type-----	217
7.6 Constant-----	218
8.Assignment and function of SM special storage area-----	219
9.Easy ladder communication-----	220
9.1 PR series PLC basic introduction of network communication-----	220
9.2 PR series PLC communication-----	222
9.3 Optimize network performance-----	230
10.Additional chapter-----	231
10.1 How to switch PLC mode-----	231
10.2 Value range of analog quantity:-----	232
10.3 Extension module address-----	232
10.4 PLC host address range-----	233
10.5 Formula-----	233
10.6 Set extension module address with a dial switch-----	234
10.7 Additional instructions-----	235
10.7.1 LCD related instructions-----	235
10.7.2 CAN, serial port initialization instructions-----	244
10.8 Example of serial port free port communication-----	245
10.9 Example of CAN free port-----	247
10.10 MODBUS communication master program-----	249
10.11 The example of using PID instruction-----	250
MODBUS ADDRESS-----	254

Introduction

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

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

The xLogic 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 Rievtech-to-use xLogic soft. DIN-rail and panel-mounted options are both available, offering full flexibility to the various installation needs of your application.

The xLogic Micro PLC is available in 120V/240V AC or 12V and 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 xLogic Micro PLC to work. Installation, programming and use of the unit are detailed in this manual. The feature-rich xLogic 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 xLogic Micro PLC.

Valid range of this manual

The manual applies to devices of PR 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 xLogic 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@rievech.com

1 .what is Xlogic ?

1.1 Overview

xLogic is a universal logic module made by Rievtech.

xLogic , 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 xLogic 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 and 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 xLogic to act as slave or master in certain Modbus communication network. (easy connect to other factory touch screen by RS232 cable, RS485 module)
- | Support free port communication, CAN communication and MODBUS communication
- | Expandable up to 16 linked I/O expansion modules reaching 282 I/O points in maximum
- | Optional RS232, RS485 connectivity
- | Multiple channels analog inputs available with DC 0-10V signal , PT100

signal & 0/4...20mA.

- I Default Real Time Clock (RTC) and summer/winter timer is available
- I Backup at Real Time Clock (RTC) at 25 ° C: 20 days
- I 4 channels high-speed counting
- I Pre-configured standard functions, e.g. on/ off-delays, pulse relay and softkey
- I 2 PWM channels(10KHz in maximum)
- I Retentive memory capability (Not applied to PR-6&PR-12-E series CPU)
- I RS232 and USB communication download cable with photo-electricity isolation
- I Support Ladder diagram programming(Not applied to PR-6&PR-12-E series CPU)
- I Mounting via modular 35mm DIN rail or screw fixed mounting plate
- I On-line monitor capability(Free charge SCADA for all series xlogic)
- I Datalogging
- I Kinds of analog signals process capacity (DC 0..10V , 0/4...20mA and PT100 probe inputs and DC 0..10V and 0/4...20mA outputs)
- I Low cost

1.3 Some of the things xLogic can do for you?

The xLogic 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 xLogic devices:

xLogic Basic is available in two voltage classes:

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

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

In the versions:

* With Display: with “-HMI” model, such as PR-12DC-DA-R-HMI

* Without Display: PR-6 series and with “-CAP” model, such as PR-12DC-DA-R-CAP. Only PR-12 has -CAP version. PR-18, PR-24 all have display in default.

Expansion modules:

PR-E (applied to PR-18/PR-24 CPU)

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

* xLogic analog modules are available for operation with 12~24 V DC and are equipped with six digital and 4 analog inputs.

Communication cable and module:

I xLogic: RS232 communication cable (Model: ELC-RS232)

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 xLogic CPU through xLogicsoft 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.

I xLogic: USB communication cable (Model: ELC-USB).

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

I xLogic: PRO-RS485 cable (Model: PRO-RS485).

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

I xLogic: RS485 module (Model: PR-RS485)

isolated 485 converter, used to bring out the terminals of RS485 port built-in PR-18, PR-24 series CPU for connection with third party devices.

Communication / Network

xLogic offers different ways to communicate within the system.

RS485 port

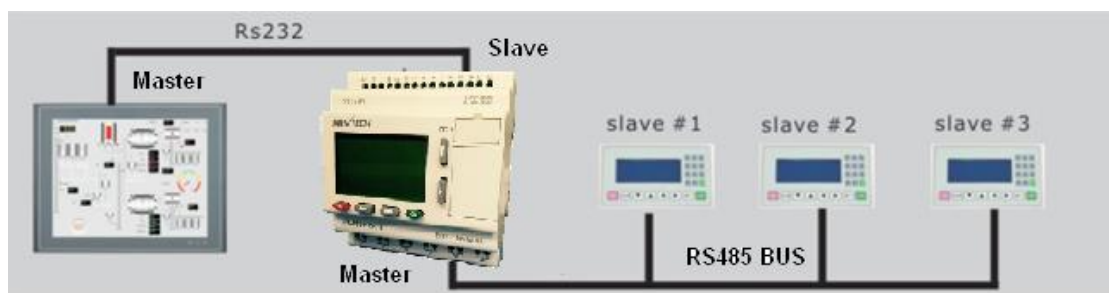
The RS485 port is used for communication between the CPU and various devices or equipments which have the standard RS485 port. Communicate using Modbus RTU/ASCII protocol.



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

RS232 or USB port (ELC-ES232/ ELC-USB 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.



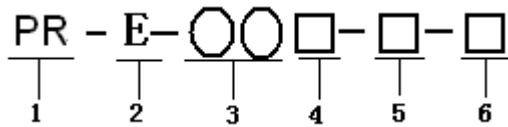
Note: PR-E-RS485 module is required to connect the CPU to RS485 BUS.

Note

xLogic CPU 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
2. E: expansion module
3. Points of input and output
4. Supply power AC or DC
5. Digital/Analog D: digital DA: digital/analog
6. Output type R: relay TP: “NPN” transistor; TN: “PNP” transistor

2.2 Hardware model selection

PR-12 Series CPU Units(Non-expandable)								
Model	Expansion	Supply voltage	Inputs	Outputs	High-speed count	PWM	HMI	RTC
PR-12AC-R-HMI	no	AC110~AC240V	8 digital	4 relays (10A)	no	no	yes	yes
PR-12DC-DA-R-HMI	no	DC12-24V	4(0...10V)+4 digital	4 relays (10A)	4(15-18)(60KHZ)	no	yes	yes
PR-12DC-DA-TN-HMI	no	DC12-24V	4(0...10V)+4 digital	4Transistor(0.3A/PNP)	4 (15-18)(60KHZ)	Yes(10KHZ)	yes	yes

PR-14 Series CPU Units(Expandable)-built-in RS485 port								
Model	Expansion	Supply voltage	Inputs	Outputs	High-speed count	PWM	HMI	RTC
PR-14AC-R-HMI	yes	AC110~AC240V /DC110-DC240V	10 digital	4 relays (10A)	no	no	yes	yes
PR-14DC-DA-R-HMI	yes	DC12-24V	6(0...10V)/6digital+4 digital	4 relays (10A)	4(17-1A)(60KHZ)	no	yes	yes

PR-18 Series CPU Units(Expandable)								
Model	Expansion	Supply voltage	Inputs	Outputs	High-speed count	PWM	HMI	RTC
PR-18AC-R-HMI	yes	AC110~AC240V /DC110-DC240V	12 digital	6 relays (10A)	no	no	yes	yes
PR-18DC-DA-R-HMI	yes	DC12-24V	6(0...10V)/6digital+6 digital	6 relays (10A)	4(19-1C)(60KHZ)	no	yes	yes
PR-18DC-DA-RT-HMI	yes	DC12-24V	6(0...10V)/6digital+6 digital	4 relays (10A)+ 2 transistor(0.3A)	4(19-1C)(60KHZ)	yes(10khz)	yes	yes

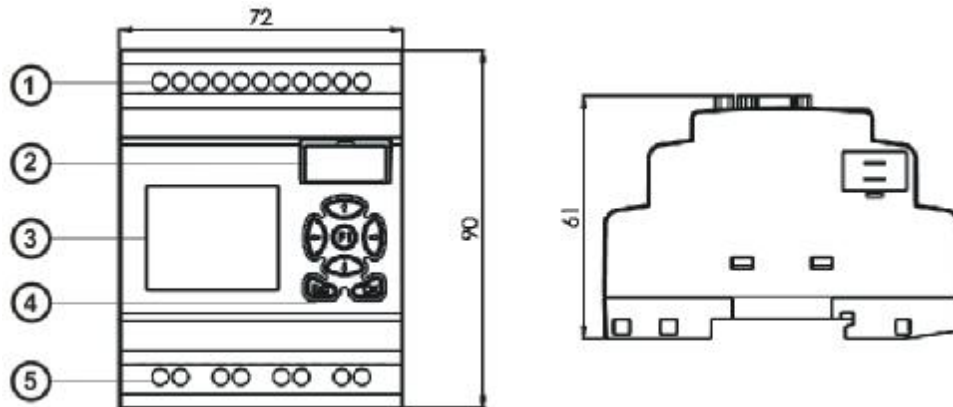
PR-24 Series CPU Units(Expandable)-built-in RS485 port								
Model	Expansion	Supply voltage	Inputs	Outputs	High-speed count	PWM	HMI	RTC
PR-24AC-R-HMI	yes	AC110~ AC240V	16 digital	10 relays (10A)	no	no	yes	yes
PR-24DC-DA-R-HMI	yes	DC12-24V	6(0...10V)/6digital+8 digital	10 relays (10A)	4(I9-IC)(60KHZ)	no	yes	yes
PR-24DC-DAI-RTA	yes	DC12-24V	2(0/4...20mA)+ 4(0...10V)/4digital+8 digital	6 relays(10A)+2Transistor(0.3A/PNP) +1(0...10V)/(0...20mA)	4(I9-IC)(60KHZ)	YES(10khz)	yes	yes

Expansion Modules(For PR-18,PR-24 series)			
Model	Supply voltage	Inputs	Outputs
PR-E-16AC-R	AC110~ AC240V	8 digital	4 relays (10A) + 4 relays(3A)
PR-E-16DC-DA-R	DC12-24V	4digital+4analog(0..10V)/digital	4 relays (10A) + 4 relays(3A)
PR-E-16DC-DA-TN	DC12-24V	4digital+4analog(0..10V)/digital	8 transistors(PNP)(0.3A)
PR-E-PT100	DC12-24V	3 Channels PT100, resolution: 0.5°, temperature range : -50°C- 200°C	none
PR-E-AQ-VI	DC12-24V	none	2 Channels (DC 0...10V/0...20mA)
PR-E-AI-I	DC12-24V	4 Channels (0/4.....20 mA), Current Signal	none
PR-RS485	DC12-24V	isolated 485 converter,used to bring out the terminals of RS485 port built-in PR-18&PR-24 series CPU for connection with third party devices.	

Accessories	
RS232 Cable	RS232 communication module /download cable between PC and xLogic CPU units
USB Cable	USB communication module /download cable between PC and xLogic CPU units
PRO-RS485	Converter cable from program port to RS485 port.
ELC-BATTERY	RTC BATTERY, the RTC can be backup for 20days in default, but with this battery, the RTC shall be backup for 1 year(only can be applied with PR-18 CPU).

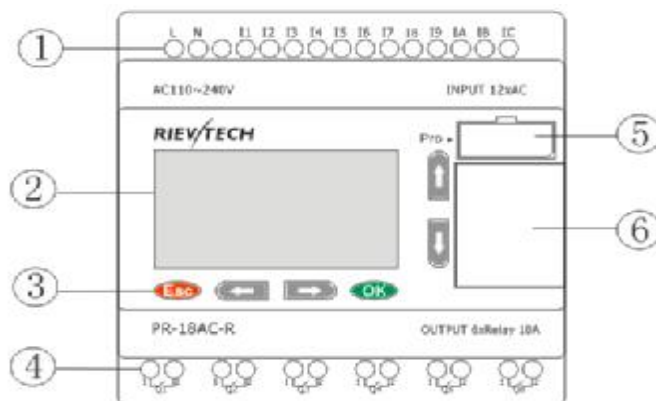
2.3 Structure & dimension

1. Standard PR-12 series with LCD model :



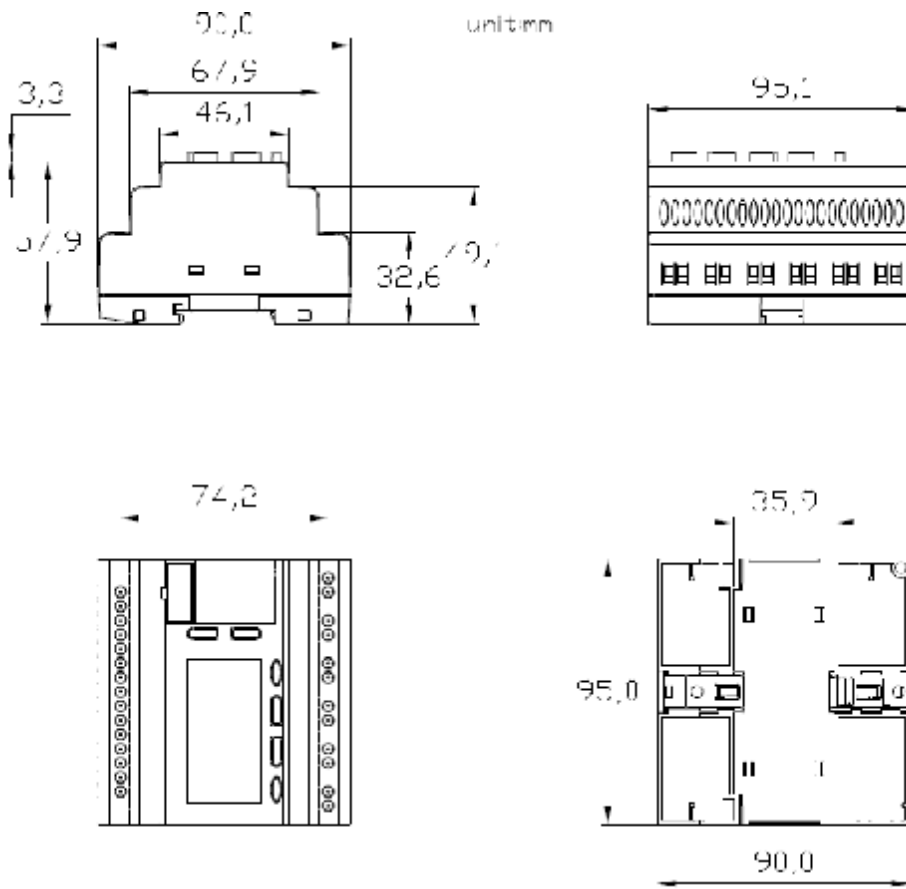
1. Power supply & Input terminals 2. Program Port (can be used as RS232 port with ELC-RS232 or RS485 port with PRO-RS485) 3. HMI /LCD panel 4. keypad 5. Output terminals

2. PR-14 and PR-18 series model :

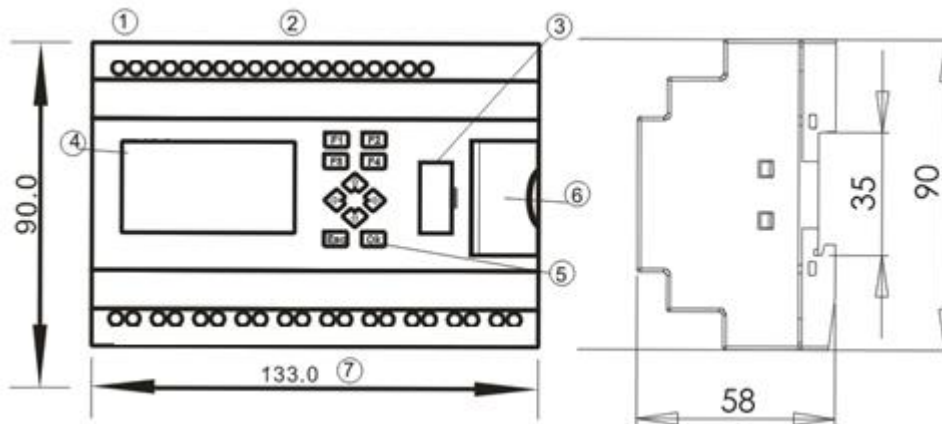


1. Power supply & Input terminals 2. HMI /LCD panel 3. keypad 4. Output terminals 5. Program Port (can be used as RS232 port with ELC-RS232 or RS485 port with PRO-RS485) 6. Extension port

Dimensions of PR-14 and PR-18:

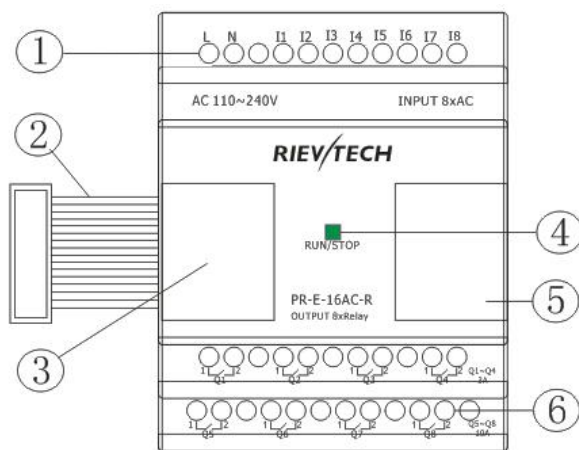


3. PR-24 series CPU



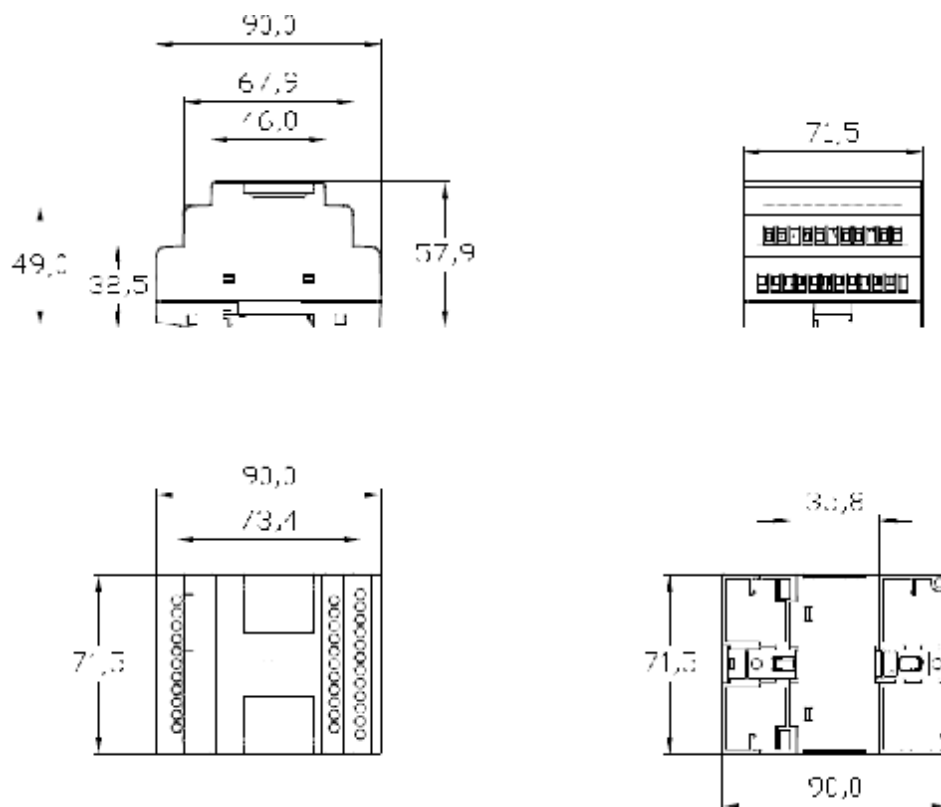
1. Power supply 2. Input 3. Program/RS232 port 4. HMI/LCD panel
 5. keypad 6. Extension/RS485 port 7. Output

4. PR-E extension module:



1. Power supply&Input terminals 2. Connection cable between CPU and extension(Detached) 3.Extension port(Left) 4. RUN/STOP indicator 5. Extension port(Right) 6. Output terminals

Dimensions of PR-E:



3 .Installing/removing xLogic

Dimensions

The xLogic installation dimensions are compliant with DIN 43880.
xLogic can be snap-mounted to 35 mm DIN rails to EN 50022 or on the wall.

xLogic width:

- I PR-14, PR-18 Series CPU has a width of 95mm.
- I PR-E expansion modules have a width of 72mm.
- I PR-24 Series CPU has a width of 133mm.
- I PR-12 Series CPU has a width of 72mm

Warning



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

3.1 DIN rail mounting

Mounting

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

1. Hook the xLogic 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 xLogic expansion module onto the rail
4. Slide the module towards the left until it touches the xLogic 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 xLogic:

if you have installed only one xLogic Basic:

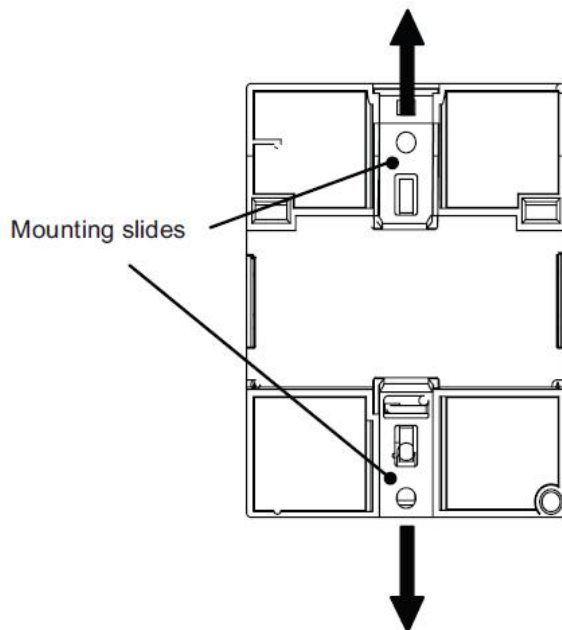
1. Insert a screwdriver into the eyelet at the bottom of the slide interlock and move the latch downward.
2. Swing the xLogic Basic off the DIN rail.

if you have connected at least one expansion module to xLogic 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 xLogic by means of two mounting slides and two $\emptyset M4$ screws (tightening torque 0.8 to 1.2 Nm).



Drilling template for wall-mounting

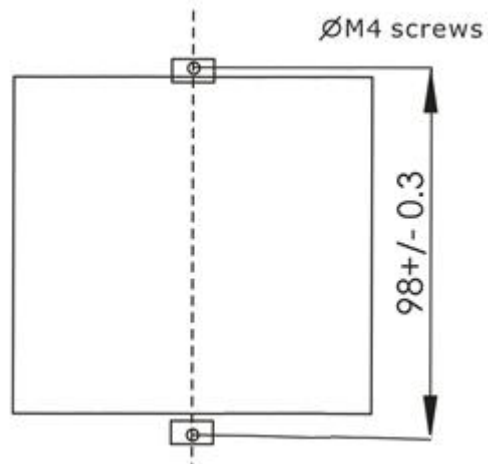
Before you can wall-mount xLogic, you need to drill holes using the template shown below.

All dimensions in mm

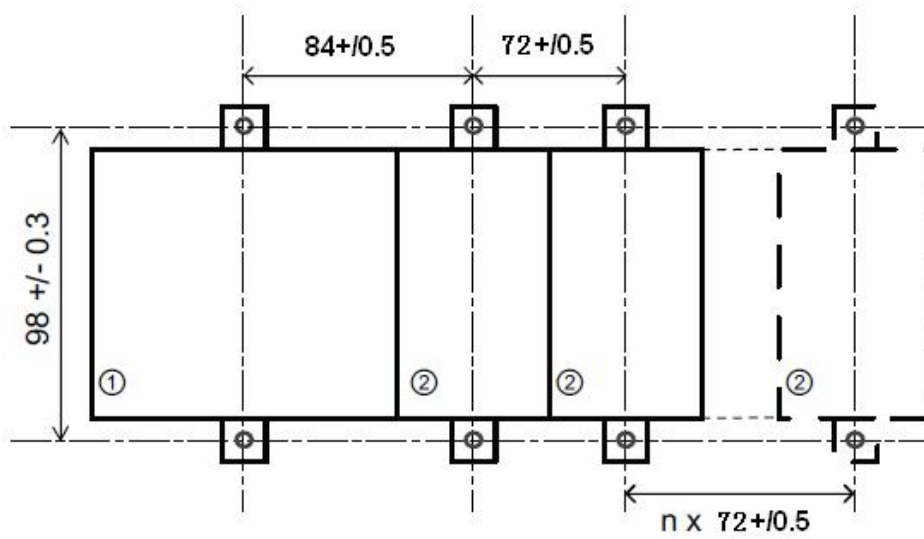
Bore hole for $\emptyset M4$ screw, tightening torque 0.8 to 1.2 Nm

1. xLogic CPU

PR-12 Series CPU

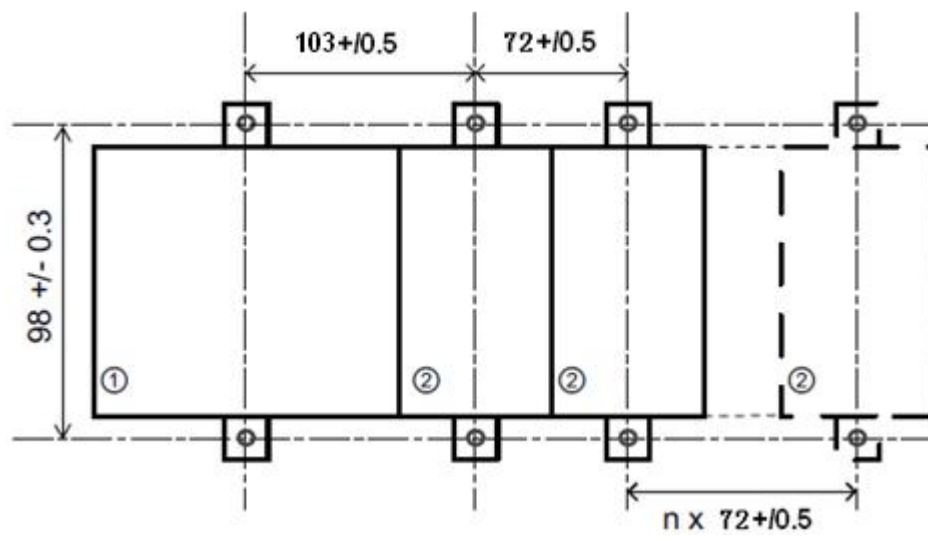


PR-14 and PR-18 series:



1. PR-18 CPU; 2. PR-E extension

PR-24 series



3.3 wiring xLogic

Wire the xLogic 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 xLogic adequately from impermissible contact to live parts, local standards must be complied with.

3.4 Connecting the power supply

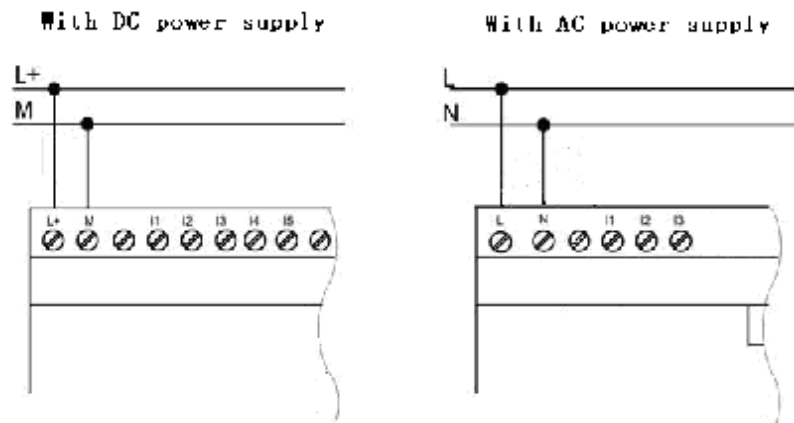
The PR-12AC, PR-18AC, PR-24AC versions of xLogic are suitable for operation with rated voltages of 110 V AC and 240 V AC. The PR-12DC, PR-18DC, PR-24DC versions can be operated with a 12 or 24 VDC power supply.

Note

A power failure may cause an additional edge triggering signal.

Data of the last uninterrupted cycle are stored in xLogic

To connect xLogic to the power supply:



3.4.1 Connecting xLogic inputs

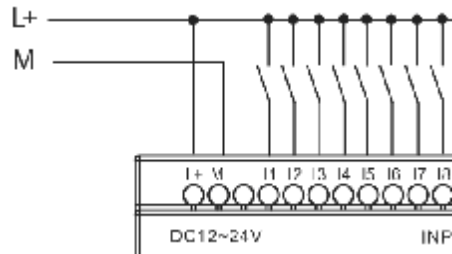
1. Requirements

the inputs you connect sensor elements such as: momentary switches, switches, light barriers, daylight control switches etc.

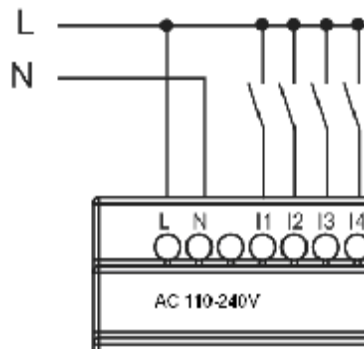
	AC Type	DC Type
Signal status 0 Input current	<40VAC <0.03mA	<5VDC <0.1mA
Signal status 1 Input current	>79VAC Typical 0.06 0.24mA	>10VDC Typical 0.3mA
Analogue input	NO	AIW0-AIW6(0-10V DC) (PR-6, PR-12)

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

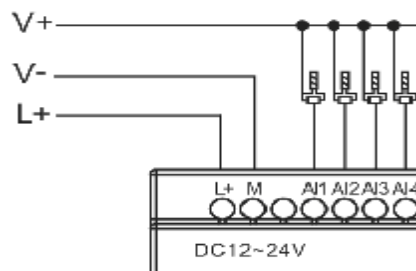
* DC type digital inputs



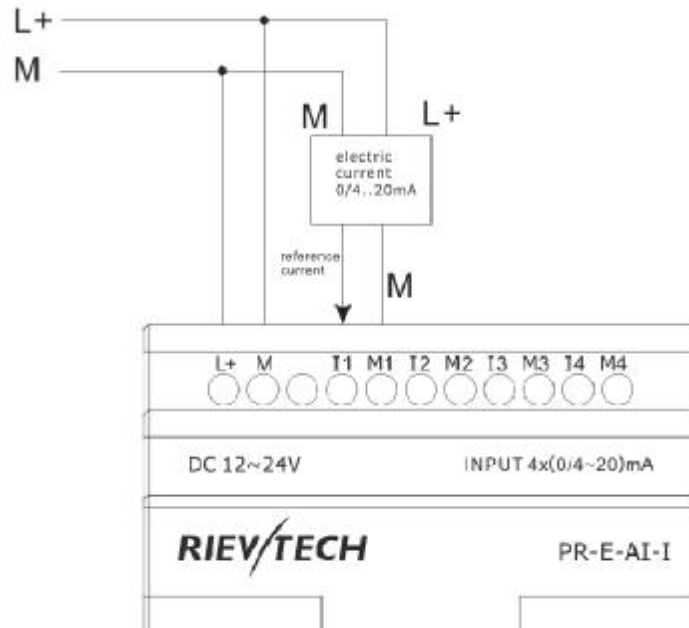
* AC type digital inputs



* Analog Inputs (DC 0...10V)



* Analog inputs current Inputs (0...20mA)



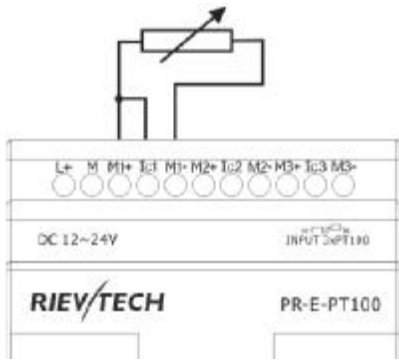
The above figure shows how to make a four-wire current measurement.

PR-E-PT100

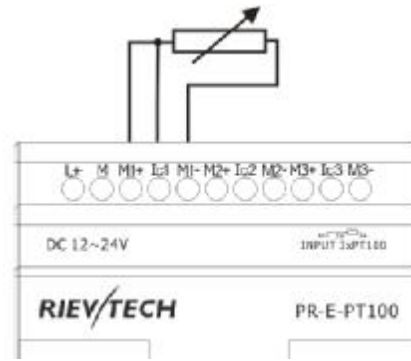
It can be connected with one two-wire or three-wire resistance-type thermocouple.

When two-wire technology applied, the terminals “M1+ and IC1” (this rule also shall be applied to “M2+ and IC2”, “M3+ and IC3”) would be short connected. Such connection can not compensate error/tolerance caused by the resistance in measurement loop. The measurement error of 1 Ω impedance of power cord is proportional to +2.5 $^{\circ}$ C

The three-wire technology can inhibit the influence of measurement results caused by cable length (ohmic resistance).



2-wire (short circuit M+ and I_c)

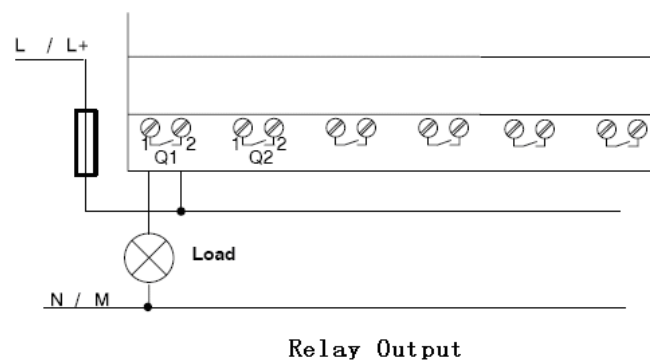


3 wire

3.4.2 Connecting xLogic Outputs

1. Requirement for the relay output

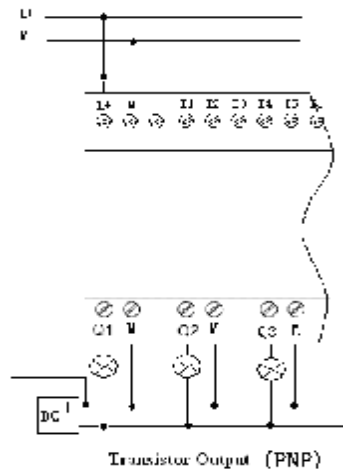
Various loads such as lamp, fluorescent tube, motor, contact, etc., can be connected to the outputs of xLogic. The maximum ON output current that can be supplied by xLogic is 10A for the resistance load and 3A for the inductive load. The connection is in accordance with the following figure:



2. Requirement for the electronic transistor output:

The load connected to xLogic must have the following characteristics:

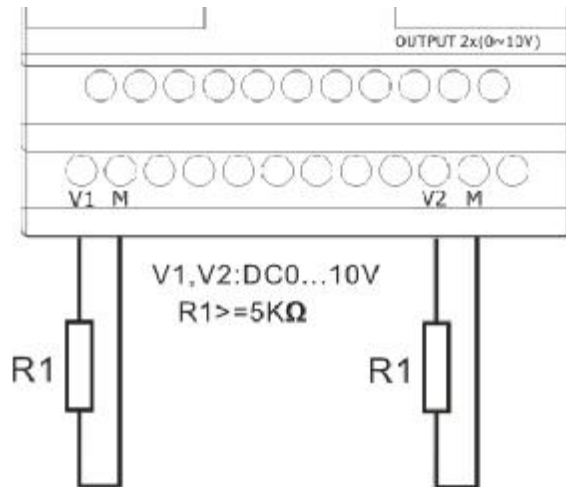
- * The maximum switch current cannot exceed 0.3A.
- * When the switch is ON (Q=1), the maximum current is 0.3A.



Notes (PNP):

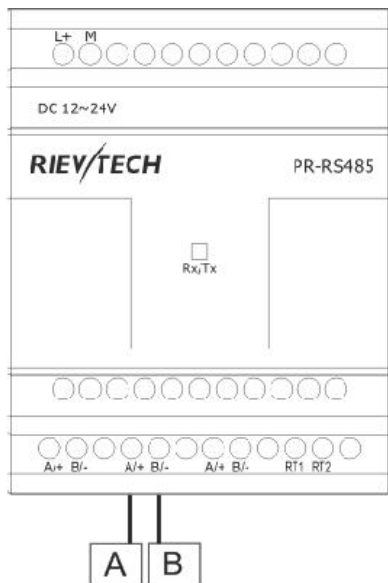
- * The load connecting voltage must be $\leq 60\text{VDC}$ and it must be DC.
- * The “+” terminal of the output wiring must be connected with the DC positive voltage, and it must be connected with the “L+” terminal of the xLogic power, a load must be connected with the “-” terminal of the DC negative voltage.

PR-E-A0-VI (DC0..10V analog output).



PR-RS485

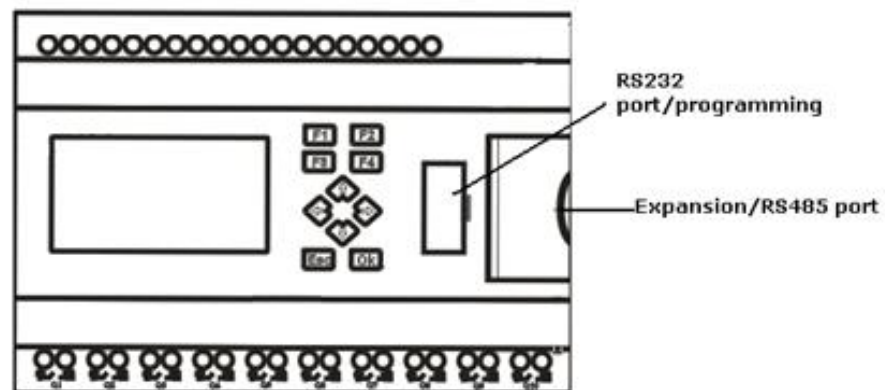
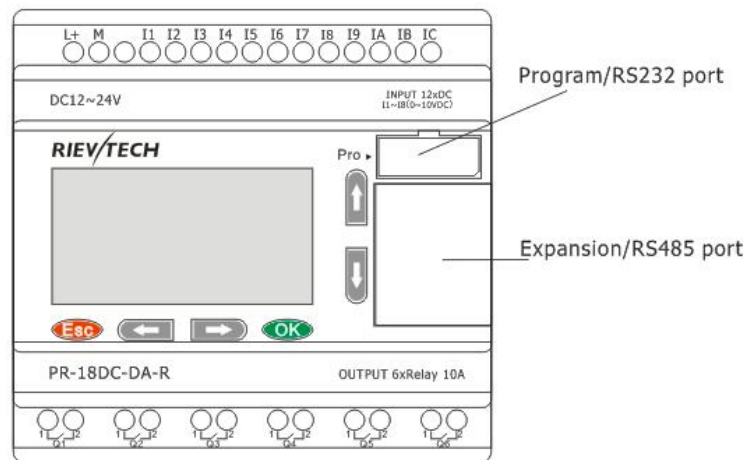
Actually, PR-RS485 is just a converter with photo isolation bringing out 3 wiring terminals (short circuited inner of such 3 terminals, so only one channel RS485 bus is available) from RS485 port (2x8pin) of CPU (PR-18/ELC-22/PR-24) for your easy connection with other devices.



If "RT1", "RT2" terminal are short connected, one 120R resistor will be connected between A/+ and B/-

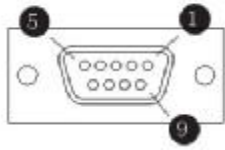
3.4.3 Communication port instructions:

PR-14, PR-18 , PR-24 CPUs



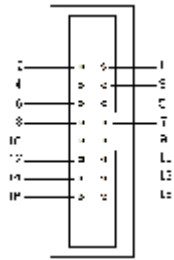
1. Programming port/RS232 port (RS232 cable ,USB cable, ELC-MEMORY, ELC-BATTERY, PR0-RS485) should be inserted in this port.

When the programming port should be used as the standard RS232 port (D-shape 9 pin header) , the RS232 cable is needed. Below is show you the pin definition of the pin:



PIN	function
2	RXD
3	TXD
5	GND
others	NULL

1. Expansion port/RS485 (pin definition)



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

A

Communication between CPU and expansion module will use 4, 7, 9, 15 pin.
 PR-RS485 module is required when PR-18/PR-24 CPU communicate with the third party devices via RS485 bus.

4.Quick reference manual

4.1 Special memory 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 overflow 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

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:

PR-24DC-DA-R/PR-24DC-DAI-RTA

I1.0	HC0
I1.1	HC1
I1.2	HC2
I1.3	HC3

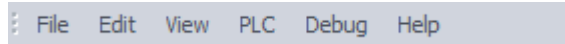
Each high speed counter occupies an input point for receiving the pulse. No reset, adjust the direction, start and other functions. The high speed counter can only be used for recording number in the PLC. The number of

high speed counters of different PLC are different. PR-24DC-DA-R PLC and PR-24DC-DAI-RTA PLC have four high speed counters. The input points of high speed counters are I1.0, I1.1, I1.2 and I1.3. HC0, HC1, HC2, HC4 are used for storing the values of high speed counters.

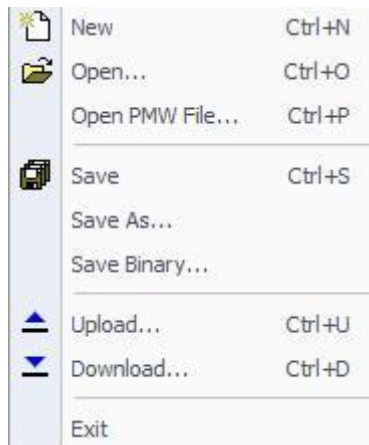
X Ladder direction for use

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. Opening the PMW is the PMW format file, you can use the shortcut CTRL + P to open.

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.






Save as binary: Save the project in a binary format.

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 X Ladder 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.

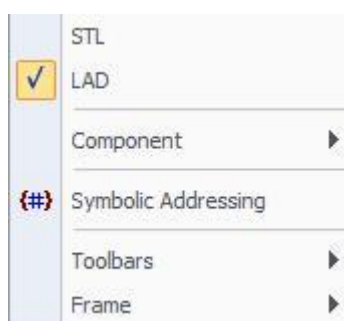
Copy: Select the object, copy the operation, and the copy of the object is placed in the WINDOWS clipboard buffer.

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.

The screenshot shows a 'Variable Symbol' table with the following data:

Symbol	Adress	Data Type	Comment
✓ Start	M0.0	BOOL	
✓ Stop	M0.2	BOOL	
✓ Motor	M0.3	BOOL	
		BOOL	

Below the table is a Ladder Logic (LAD) diagram. It features three normally open contacts labeled 'M0.0 (Start)', 'M0.2 (Stop)', and 'M0.3 (Motor)'. The 'M0.2 (Stop)' contact is connected to a coil labeled 'M0.3 (Motor)'. The 'M0.3 (Motor)' contact is connected to a coil labeled 'M0.3 (Motor)'.

Toolbar: The toolbar of the contents are as follows

The toolbar contains the following checked options:

- ✓ Standard
- ✓ Debug
- ✓ Instruction
- ✓ Instruction (FxMode)

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

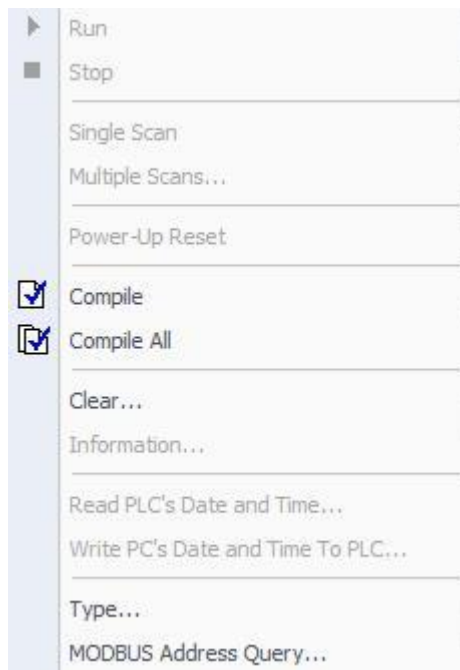
Floating window: Window that can be moved in the operating interface. Its contents are as follows:

The floating window contains the following checked options:

- ✓ Status Chart
- ✓ Project Manager
- ✓ Information Output
- ✓ Debug Information

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.

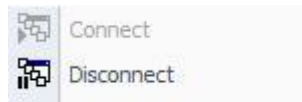
Read PLC date/time: In the connection mode, read the PLC internal date time.

Write PC time to PLC: In the connection mode, the PC time is written to the PLC.

Type: Can choose the type of PLC.

MODBUS address query: The corresponding MODBUS address of the variable.

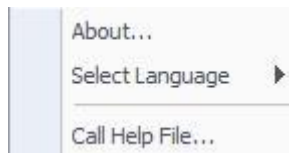
5.1.5 Debug



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 not be modified.

5.1.6 Help



About: the information of the software.

Choose Language: Can choose Chinese or English

Call help file: Can call help file.

5.2 toolbar



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

5.2.2 Open: Open an existing project, VCV format files, which can be opened by the shortcut CTRL + O. Opening the PWM is the PWM format file, can use the shortcut CTRL + P to open.

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

5.2.4 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.

5.2.5 Redo: Contrary to the function of the undo.

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

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

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

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

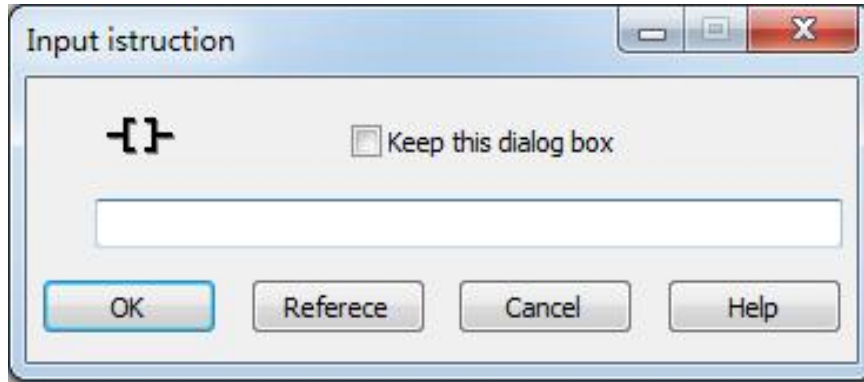
5.2.10 Compile: Compile the program of the current page.

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

5.2.12 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.

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

- 5.2.14 Connect: Display PLC data status in the program editor window.
- 5.2.15 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.
- 5.2.16 Run: Run instructions in the main menu.
- 5.2.17 Stop: Stop instructions in the main menu
- 5.2.18 Erase: Select the instruction that has been written, click erase, delete instruction.
- 5.2.19 Choose: 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.
- 5.2.20 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.
- 5.2.21 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.
- 5.2.22 Rising edge contact: Click to select the rising edge of the contact, enable input will lead to a scan cycle .
- 5.2.23 Falling edge contact: Click to select the falling edge contact , when the enable input is disconnected, it will lead to a scan cycle.
- 5.2.24 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.
- 5.2.25 Function block: Click the function block, the interface is as follows.



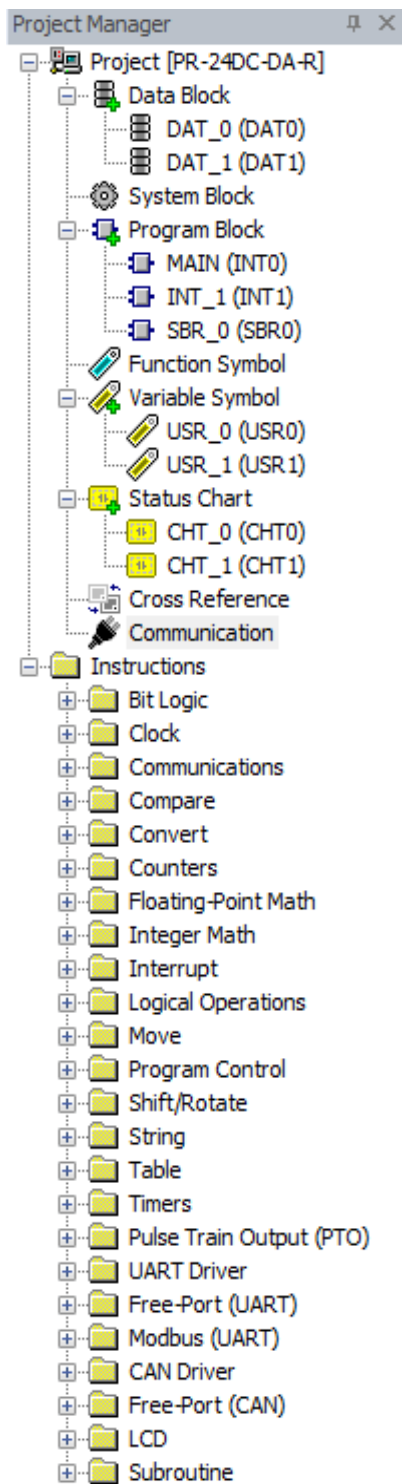
Enter the required function block instruction in the dialog box, letters are capital.

5.2.26 Level: The instruction and function blocks are connected in series.

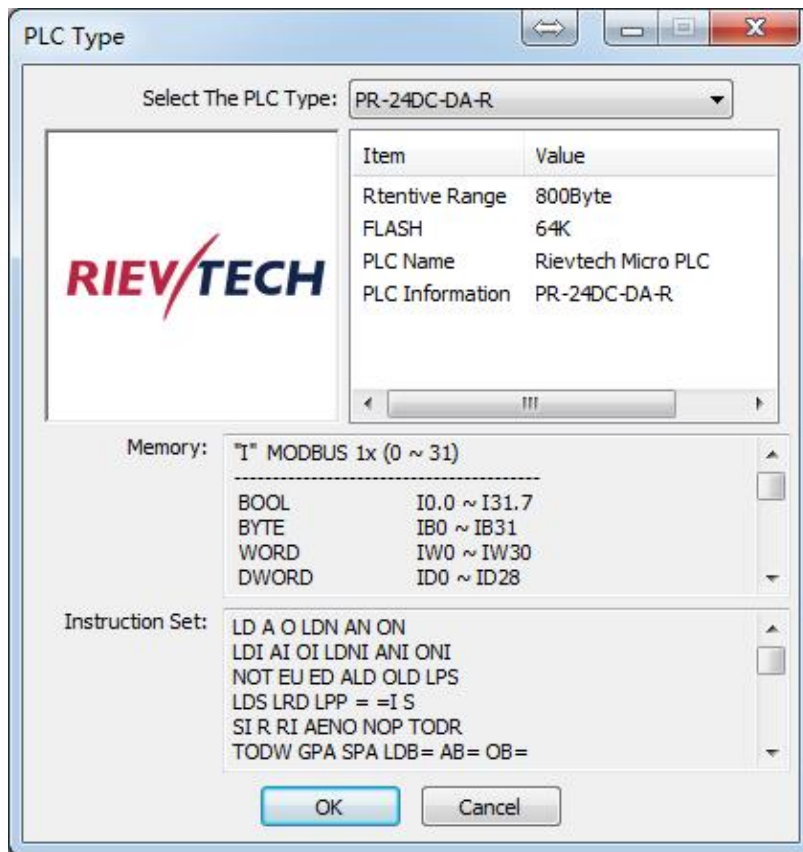
5.2.27 Vertical Line: The instruction and function blocks are connected in parallel.

5.2.28 Take back: When the enable input, the output is 0, when the enable to disconnect, the output is 1.

5.3 Instruction tree

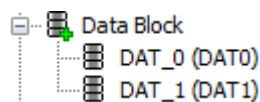


5.3.1 Project



You can choose the type of PLC, When PLC type is determined, the PLC parameter will appear below.

5.3.2 Data block

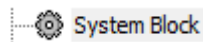


Data blocks contain DAT-0 and DAT-1, you can right-click to insert new data blocks for programmers to use. The contents of the data block are as follows:

	Address	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:

Retentive Ranges	Interrupt Time	Force Table	
RS232/RS485	RS232/RS485	CAN	Password

Defaults

	Port 0	Port 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 1... 255)
Interval frame delay (B)	10	10	(range 1... 255)

Configuration parameters must be downloaded before they take effect

OK Cancel

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

You can set the station number, baud rate, data bit, stop bit, parity, timeout and frame interval time.

CAN interface

The screenshot shows a 'System Block' configuration window with a blue title bar and a close button (X) in the top right corner. The window is divided into several sections:

- Retentive Ranges:** Includes 'RS232/RS485' and 'CAN'.
- Interrupt Time:** Includes 'RS232/RS485' and 'CAN'.
- Force Table:** Includes 'Password'.

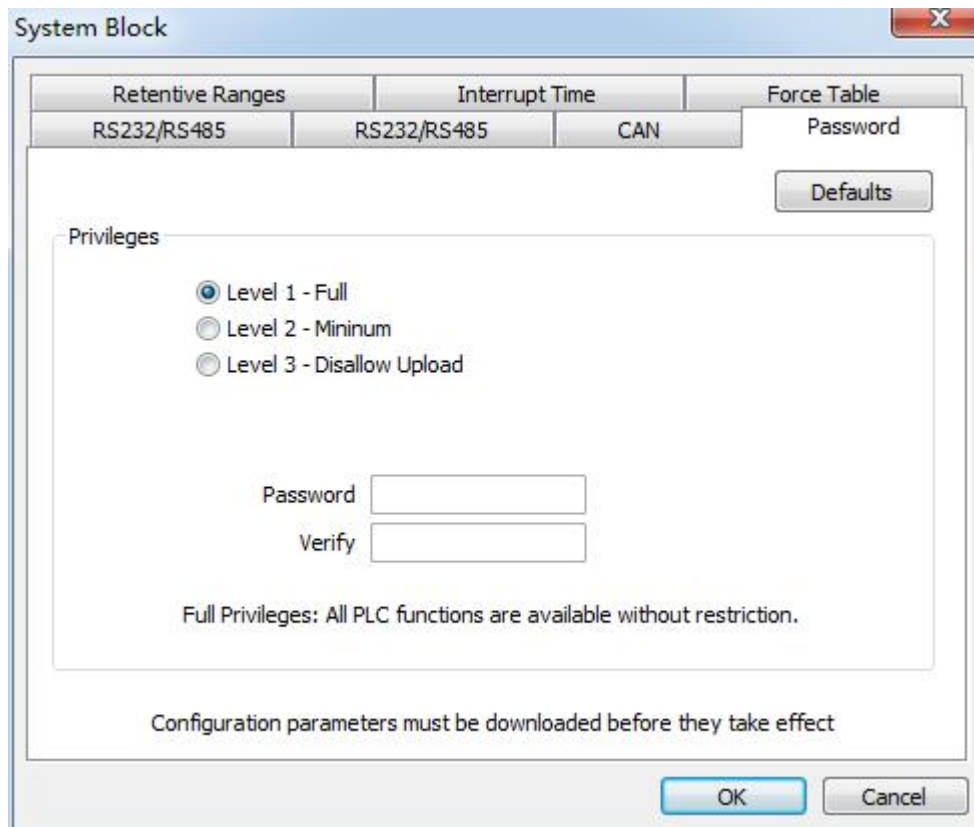
Below these sections is a 'Defaults' button. The main configuration area is split into two columns: 'Port 0' and 'Port 1'. Each column has the following settings:

- Bit rate:** A dropdown menu set to '500,000 bps'.
- Station number:** A text input field containing '1'.
- Rx Msg ID:** A text input field containing '22'.
- Tx Msg ID:** A text input field containing '21'.
- Reset timeout:** A text input field containing '500', with a note '(10 - 1000ms)' to its right.

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

PLC supports CAN communication. CAN communication will be introduced in the communication block.

Password interface



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.

And if you forget the password:

1. Power off PLC
2. Keep pressing the UP key and ESC key, then power up to PLC.
3. When "Are you sure turn to FBD" appears, you press the OK key.
4. Repeat these steps, you turn the PLC to Ladder diagram. PLC program is empty. You can download a new program.

The length of the password is 1 to 16 bits.

Retentive Ranges interface

Range	Data Area	Offset	Number of Elements	Clear
Range 0	VB	0	256	Clear
Range 1	VB	0	0	Clear
Range 2	T	0	32	Clear
Range 3	T	64	32	Clear
Range 4	C	0	16	Clear
Range 5	MB	14	18	Clear

Clear Memory Clear EEPROM

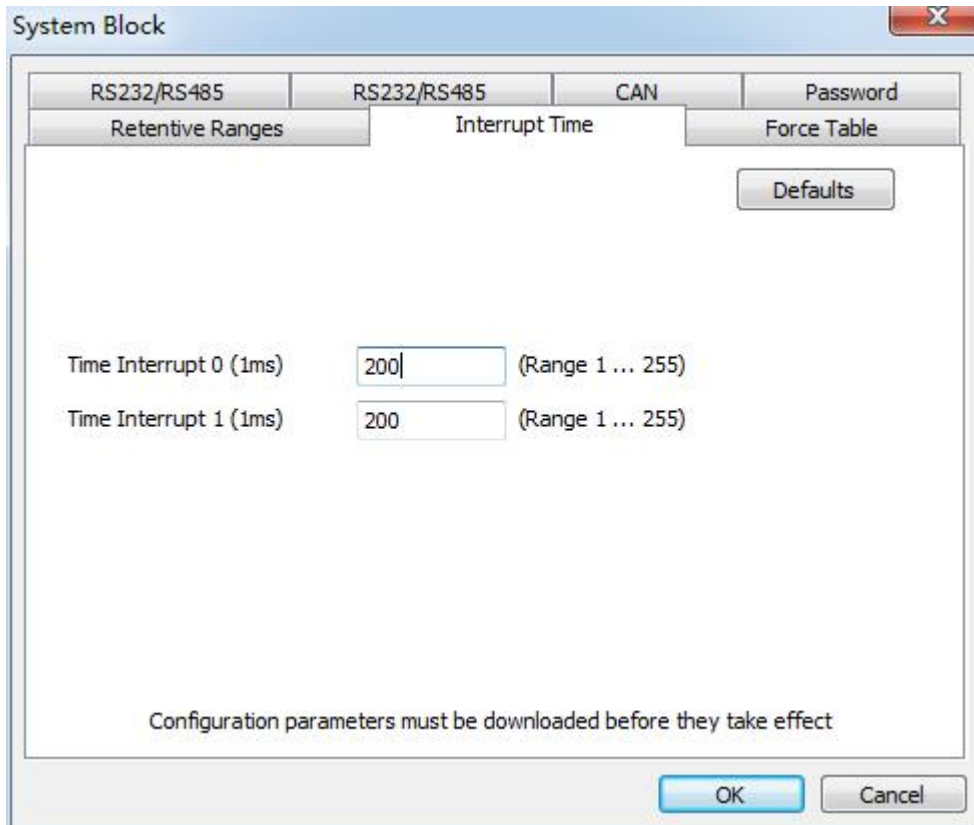
Configuration parameters must be downloaded before they take effect

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 .

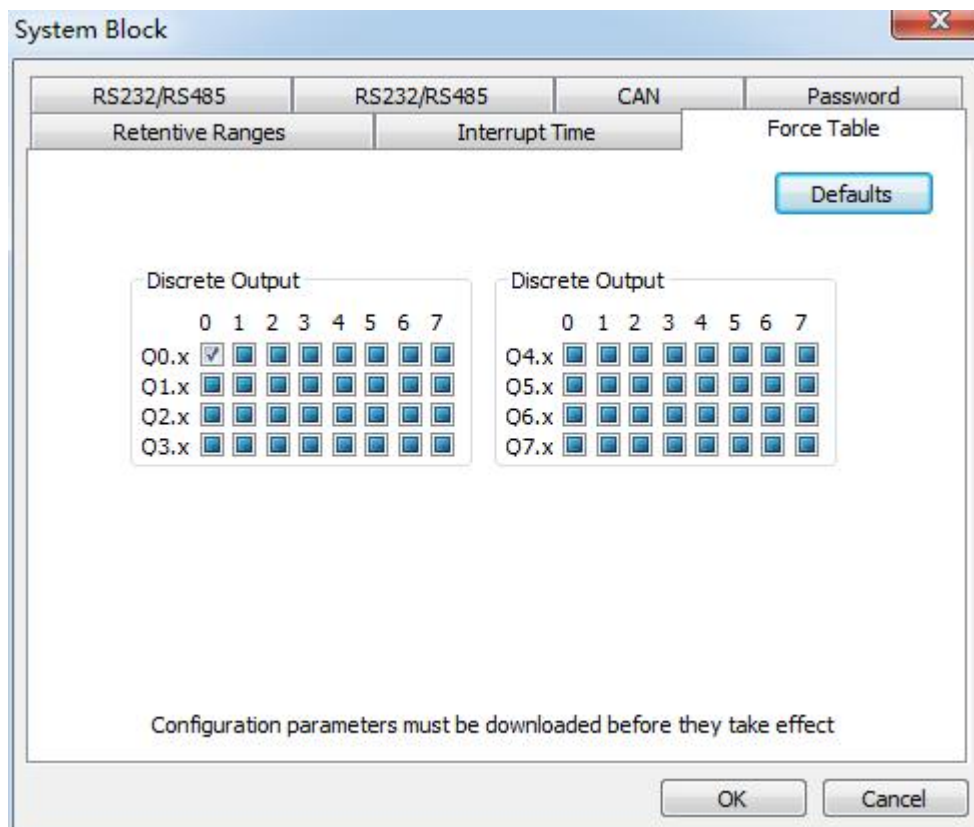
All the variables in the retentive ranges are saved permanently. PLC can hold up to 800 bytes.

Interrupt time parameter setting interface:



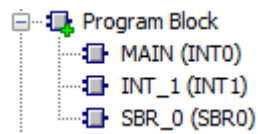
There are two time interrupt events. respectively, the time of the interrupt event 1 and the time of the interrupt event 0. The interrupt time you can set is 1 to 255 milliseconds.

Force table interface:



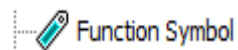
When PLC is converted from RUN mode to STOP mode, the selected output points will be 1.

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 not be added or deleted.

5.3.5 Function symbol



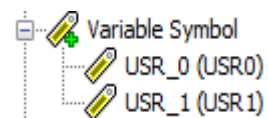
Double click the function symbol, the pop-up interface is as follows:

	Symbol	Adress	Comment
✓	MAIN	INT0	
✓	INT_1	INT1	
✓	SBR_0	SBR0	

You can modify the symbols, addresses, and comments.

	Symbol	Adress	Comment
✓	zcx	INT0	zhu cheng xu
✓	zdcx	INT1	zhong duan cheng xu
✓	zcx	SBR0	zi cheng xu

5.3.6 Variable symbol

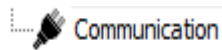


Double click the variable symbol, the pop-up interface is as follows:

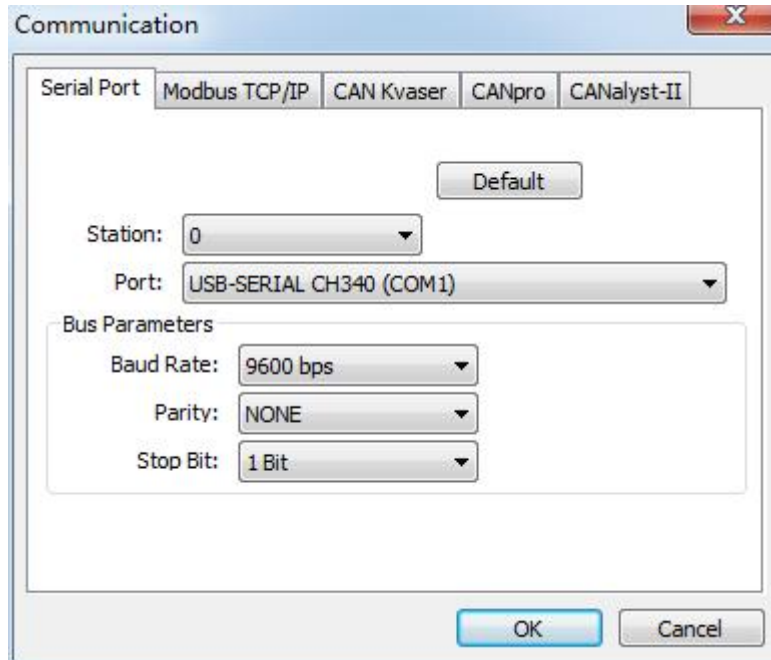
	Symbol	Adress	Data Type	Comment
			BOOL	
			BOOL	
			BOOL	
			BOOL	

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

5.3.9 Communication

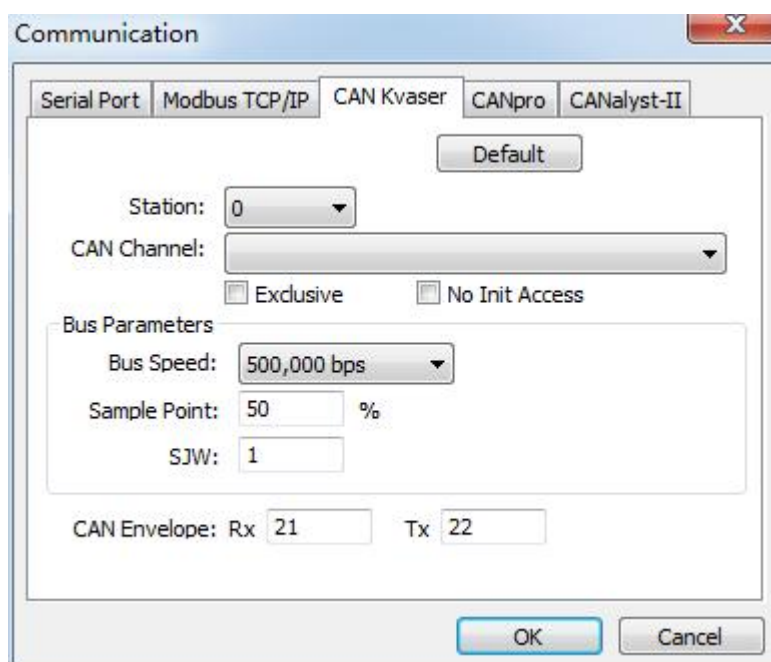


Set the PLC communication, the setting interface is as follows:

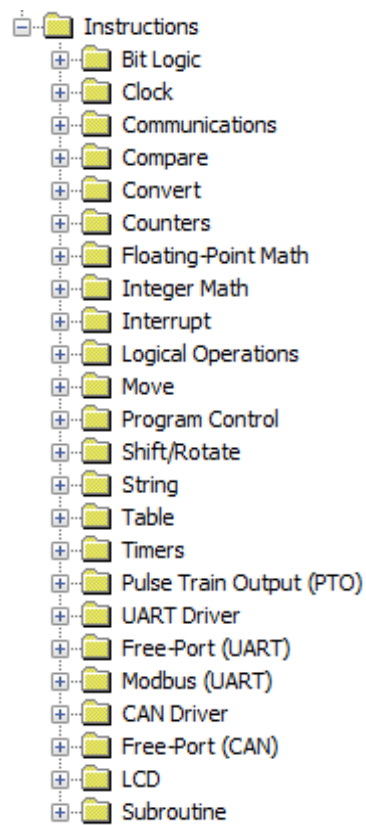


Serial Port: You can set the station number, port, baud rate, parity and stop bit.

MODBUS: PLC doesn't support the MODBUS TCP/IP function temporarily .
CAN KVASER 、CANPRO、CANALSYT -II are three kinds of CAN drivers, you can choose the corresponding CAN driver to use.



5.3.10 Instructions



Instructions will be explained in detail in the instructions section.

5.3.11 The program editor

The screenshot shows the 'Program Editor' window. At the top, there is a table for local variables:

Symbol	Var Type	Data Type	Comment
	TEMP	BOOL	
	TEMP	BOOL	
	TEMP	BOOL	
	TEMP	BOOL	

Below the table, the editor displays three networks of ladder logic:

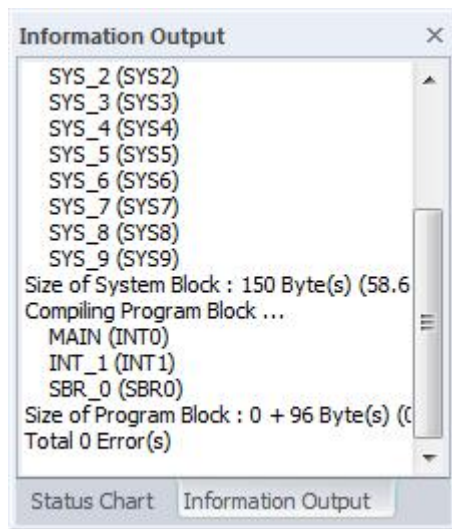
- NETWORK 0:** A normally open contact labeled M0.0 is connected to two parallel outputs. The top output is a coil labeled Q0.0, and the bottom output is a coil labeled Q0.1.
- NETWORK 1:** A normally open contact labeled M0.1 is connected to a set coil (S) labeled V0.0. The coil has a time delay of 10 units.
- NETWORK 2:** This network is currently empty.

At the bottom of the editor, there is a navigation bar with buttons for navigating between programs: MAIN (INT0), INT_1 (INT1), and SBR_0 (SBR0).

Local variable table: It will be described in detail in the PLC X Ladder storage area and variable.

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

5.3.12 Status chart, information output



Status Chart: 5.3.7 chapter.

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

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: 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.

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:

10.0 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 X Ladder program editor.

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

local variable

Local variables can be used in the X Ladder 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

CPU PR-X control 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 , Or click on the file drop-down menu  icon, Create a new project.

Open an existing project


Click the "file" icon, select "open" or "open the PWM file".

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 not 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

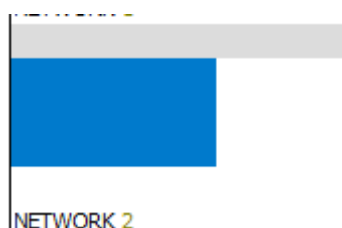
Line

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

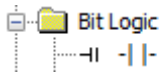


Double click the instruction tree

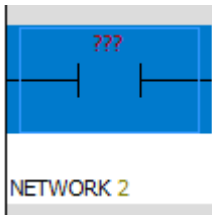
1. Place the cursor in the position you want to edit in the program editor window. Click the mouse, there will be a selection box.



2. Select the required instruction, double click it.



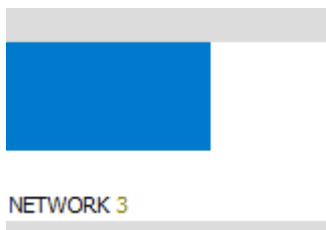
Instruction will appear in the selected editing area.



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.

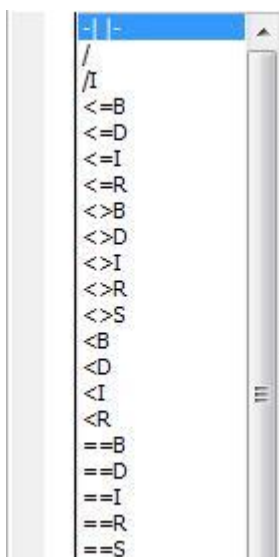


1. Select the required button in the toolbar



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.

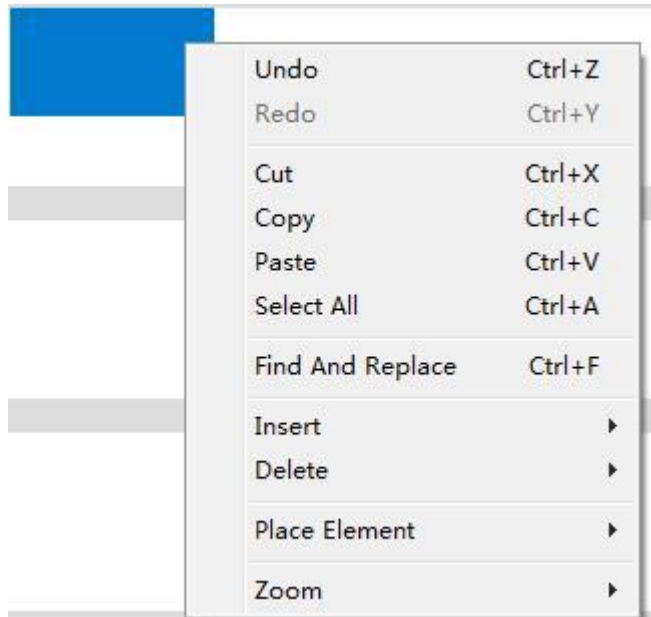
First of all, you should select a project, and then you can use the copy function. The contents of the copy are placed in the Windows clipboard buffer.

You can choose the following objectives in the project:

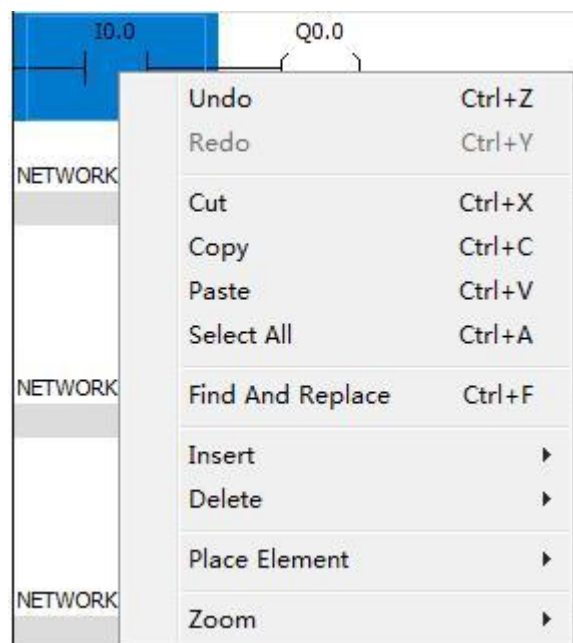
1. Program text or data domain
2. Instructions in the LAD, FBD, and STL editors
3. Single network
4. Multiple adjacent networks
5. All networks
6. Symbol table, row and column of the symbol table
7. State table, row and column of the state table

Edit cells, instructions, addresses, and networks

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

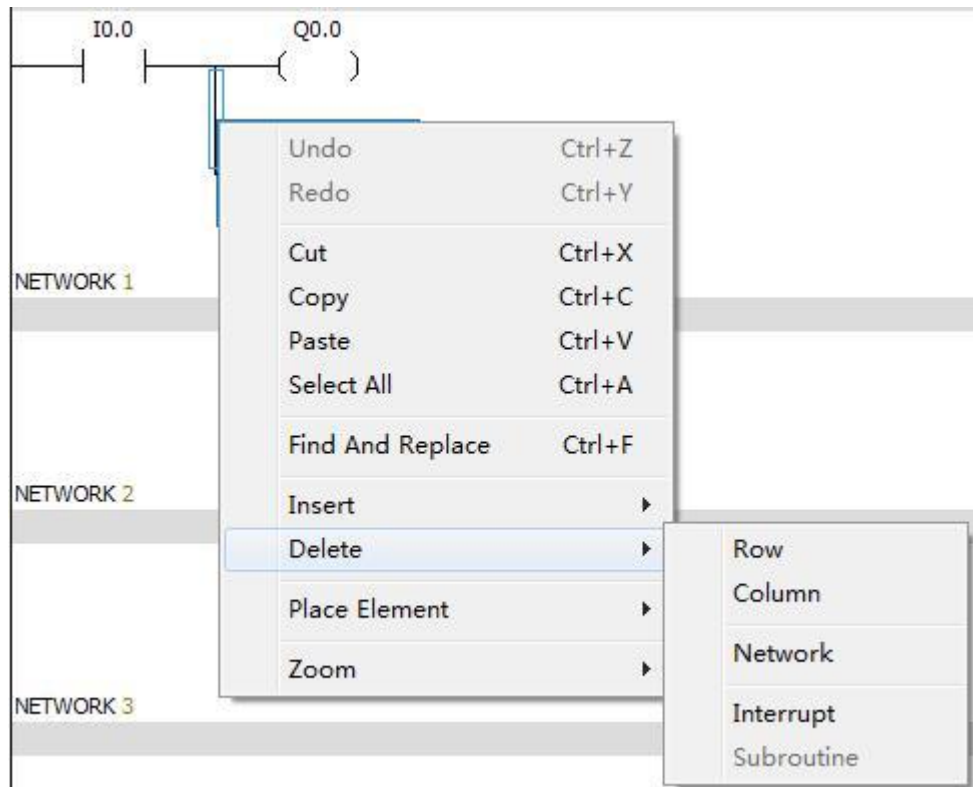


2. Select an instruction, you can use the right key to select the operations as follows:



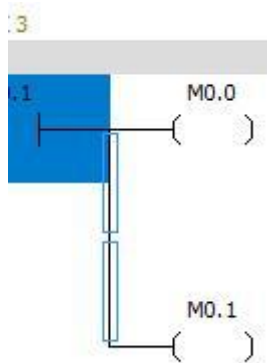
3.You can cut and paste elements and rows, delete rows or columns.

Delete element:



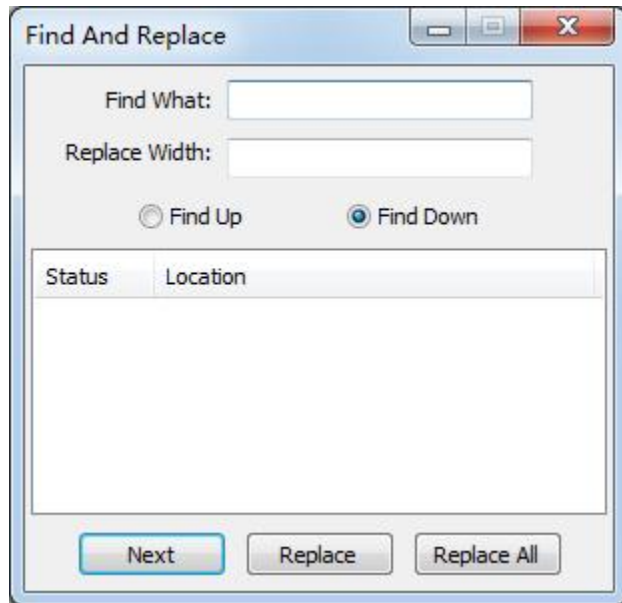
You can use the DELETE or BACKSPACE key to delete the cells; You can select the elements that need to be deleted, use the right key to select the “delete” function component.

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 "search content" field.
2. You can use the "Find up" and "Find down" functions.

Replacement function

1. Enter the string you want to search in the "search content" field.
2. Enter the string you want to replace in the "replacement content" 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 they work

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

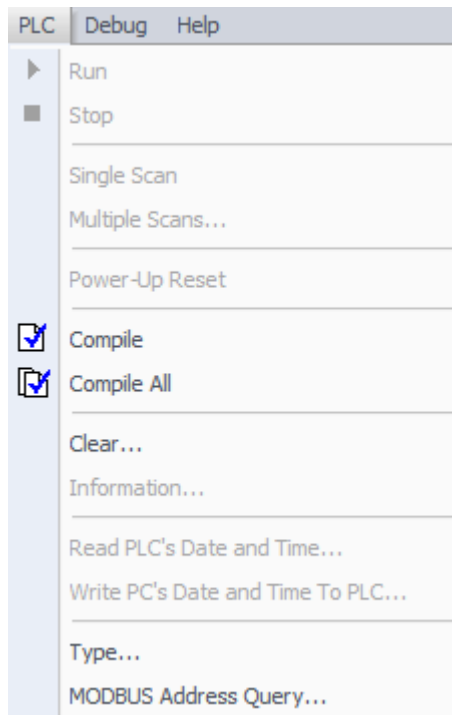
5.5.8 How to display errors in LAD in the program editor


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.

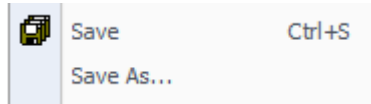
"All compile"  Compiles the program editor, system block, and data block. When you use the "All compile" command, all windows are compiled.

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 on the toolbar to save your project, or use the shortcut key CTRL+S to save your project.



"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

How to build a communication between the personal computer and the PLC in the Xladder. It depends on the hardware that you installed. Use the communication cable to connect PLC and the computer, set up the correct communication parameters in the Xladder and then PLC and computer can communicate.

You can set up the communication or edit the communication settings at any time.

Steps to establish a communication:

1. Use the communication cable to connect PLC and the computer.

Default parameters:

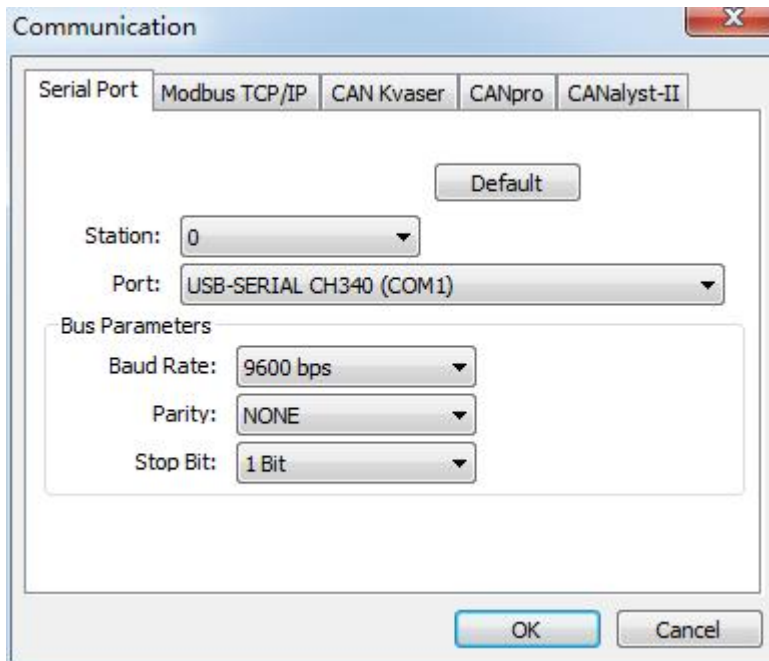
Station Number:0

Port:Select the correct port

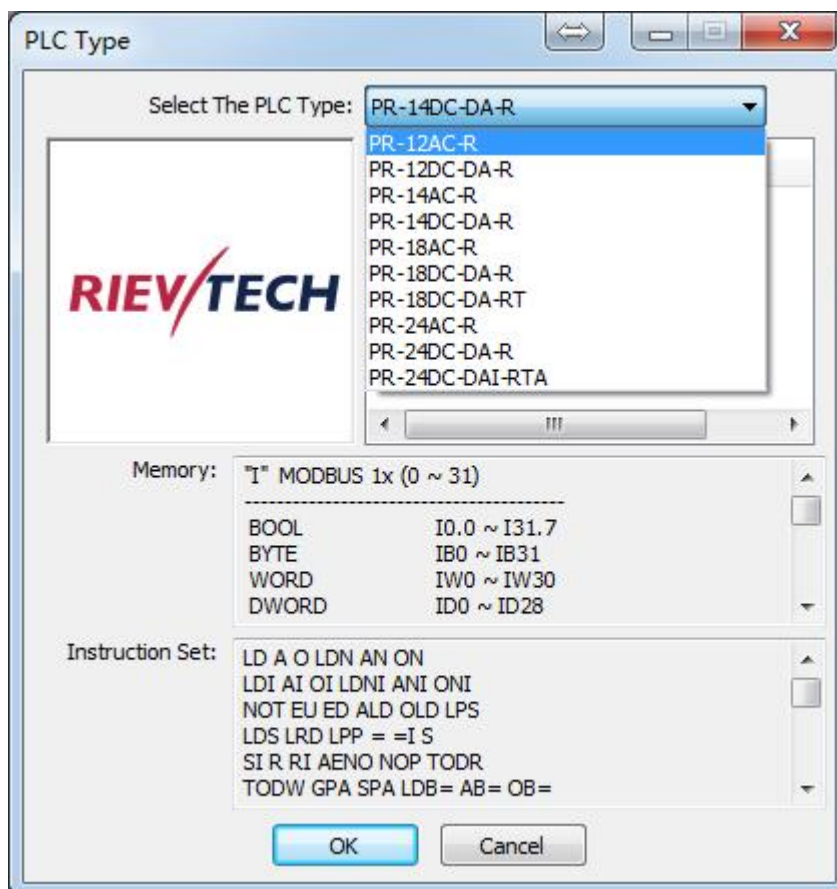
Baud rate:9600 bps

Check:EVEN

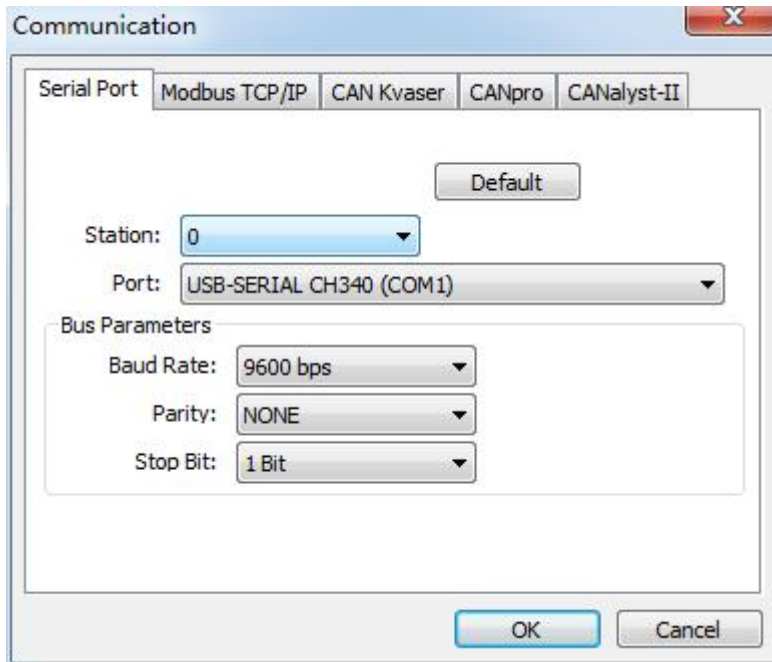
Stop bit: 1 bit



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



3. In this PLC, you can choose 5 kinds of communication. Right click to open the “communication”, the interface is as follows:




5.6.2 Download program

If XLadder and PLC communicate successfully, you can download the program to 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 .

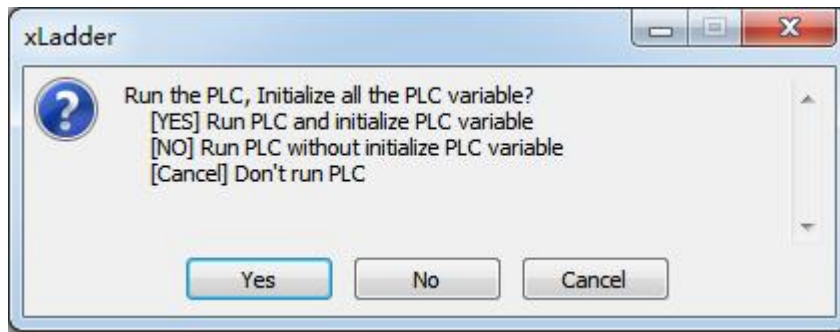
2. After the success of the compiler, click the "download"  button in the toolbar, or select File > download.

The interface is as follows:






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

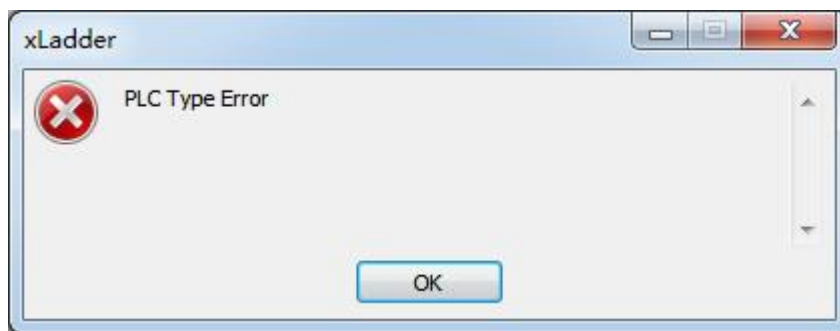
3. When the program is downloaded successfully, the interface is as follows:



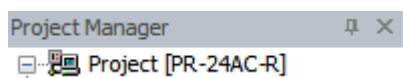
There are three options, you can choose one of them.

5. 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.

6. 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.



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



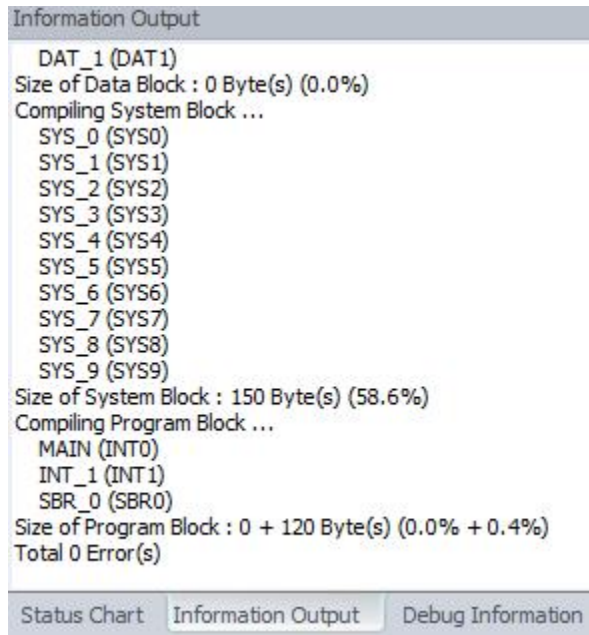
8. Click the "download" button to download the program again.

9. 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.



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

5.7 How to monitor and debug the program

After the program is downloaded, you can use the "debug" toolbar diagnostic function.

Debug Toolbar: 

You can find the toolbar instructions in the "detailed annotation of operation interface".

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.X Ladder 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.

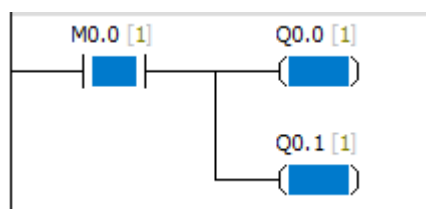
How to view data status

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 .



State values are collected in a continuous manner or snapshot manner

Continuity

1. Open the program editor window and start the "program status monitoring". When PLC is in the RUN mode, you can view the continuous state of the program data.
2. Open the status table window and start the "status table monitoring". When PLC is in the RUN mode, you can view the continuous state of the program data.

Snapshot

The PLC is converted to STOP mode, you can collect a single status update. When PLC is in the STOP mode, you can use the "multiple scan" and "single scan" functions.

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.

Attention:

When the PLC is located in STOP mode, you can perform the following operations:

1. Use the status table or the program status monitoring window to see the current value of the data.
2. Perform a limited number of scans.

When PLC is in RUN mode, you can't use the "first scan" or "multiple scan" function.

When PLC is in RUN mode, you can write and force data in the status table. You can also perform the following operations:

1. Use status table to view the continuous state of the program data.
2. Use the program status monitoring window to view the continuous state of the program data.

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 not be forced.

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

How to perform a limited number of scans

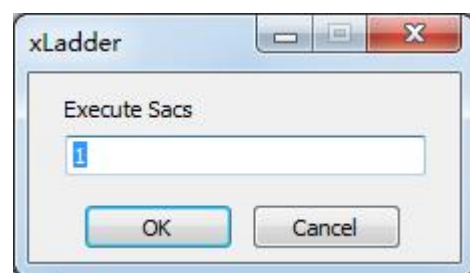
Single scan:

- 1.PLC must be set to STOP mode.
- 2.Select PLC> single scan from the menu bar.

Multiple scans:

- 1.PLC must be set to STOP mode.
- 2.Select PLC> Multiple scans from the menu bar.

Dialog box appears as follows:



- 3.Enter the value of the number of scans, click "OK".

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.

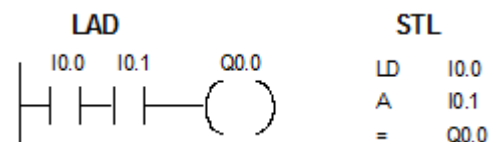
Normally closed contact: a contact that is closed in nature.

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:

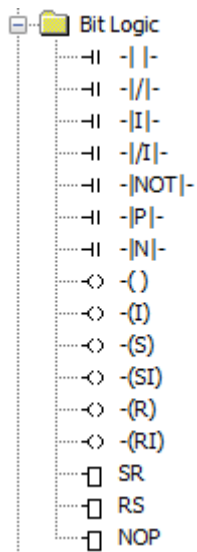


System blocks configuration

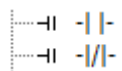
Instructions: The detailed annotation of operation interface--->System blocks

6.X Ladder instructions descriptions

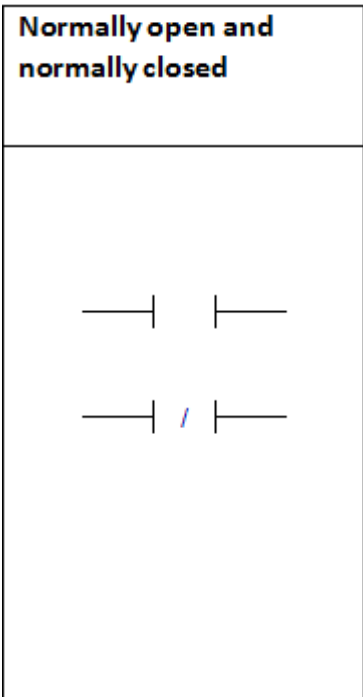
6.1 Bit logic



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
Input (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
Output (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean



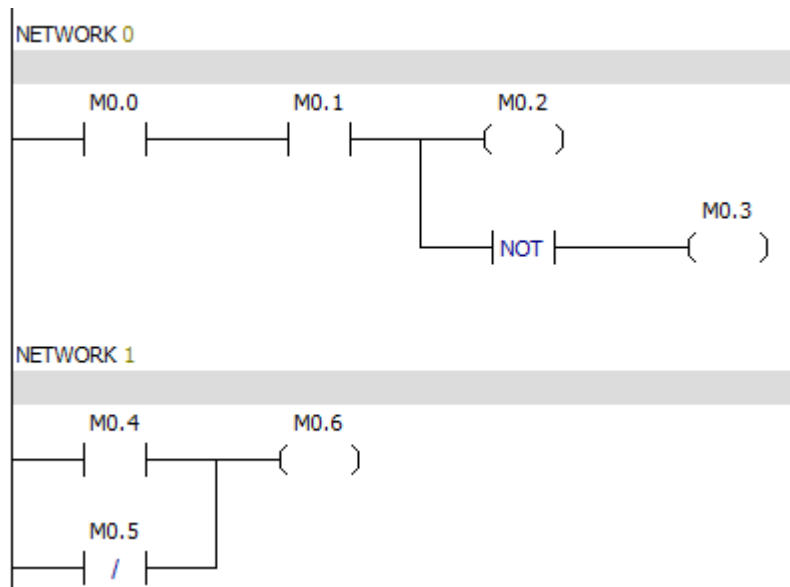
When the bit is equal to 1, the normally open contact is closed, and the normally closed contact is disconnected.

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

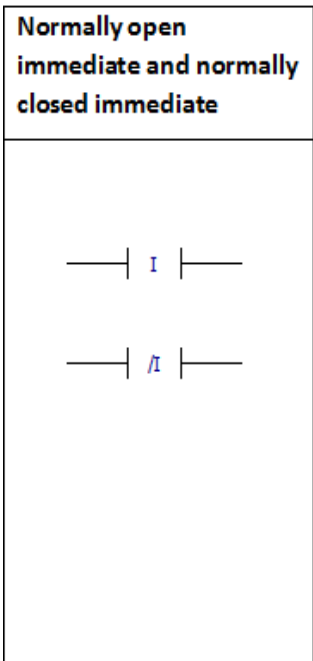
In STL, the normally open contact is represented by "LD", "And" and "Or" instructions.

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.



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.

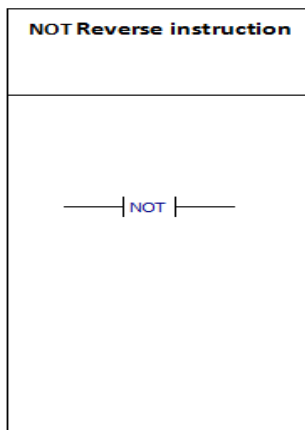
When the actual input point is 1, normally open immediate is closed.

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



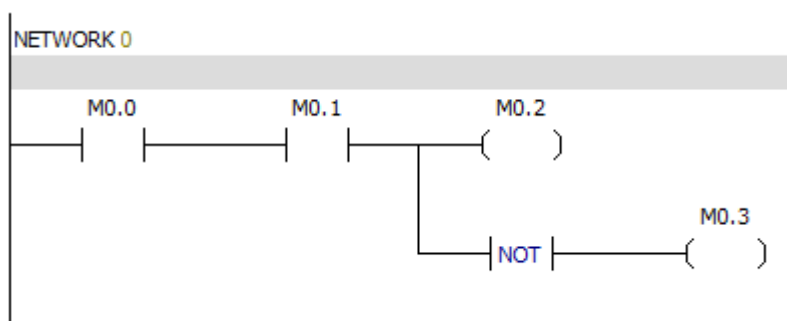
The functions of "NOT instruction" are as follows:

When the input is 0, the output is 1.

When the input is 1, the output is 0.

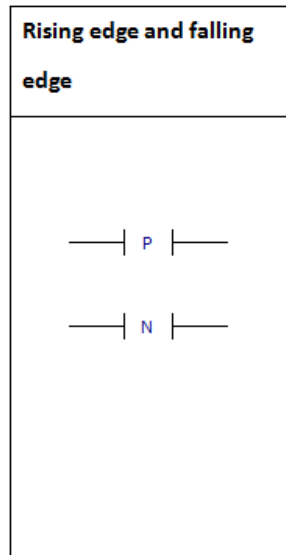
In LAD, the NOT instruction is represented by a contact.

Example:



6.1.4 Rising edge and falling edge

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

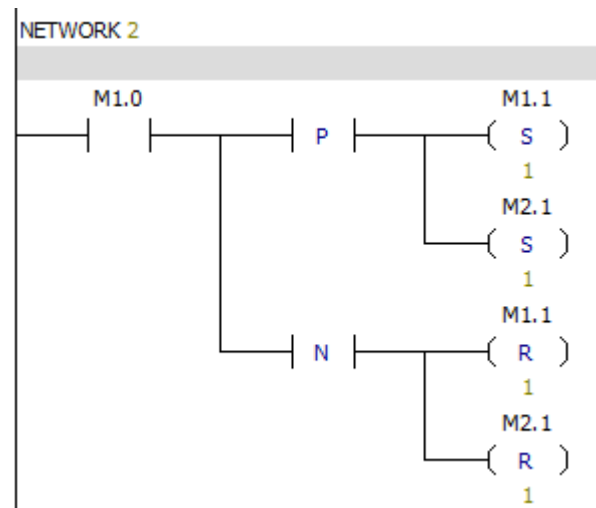


When left logic is converted from 0 to 1, Rising edge contact conduction time is a scan cycle.

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

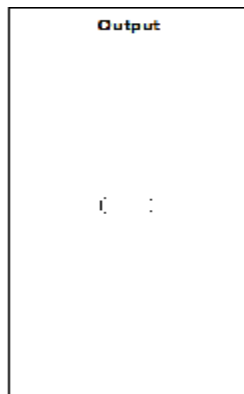
In LAD, the rising edge and the falling edge are represented by the contacts.

Example:



6.1.5 Output

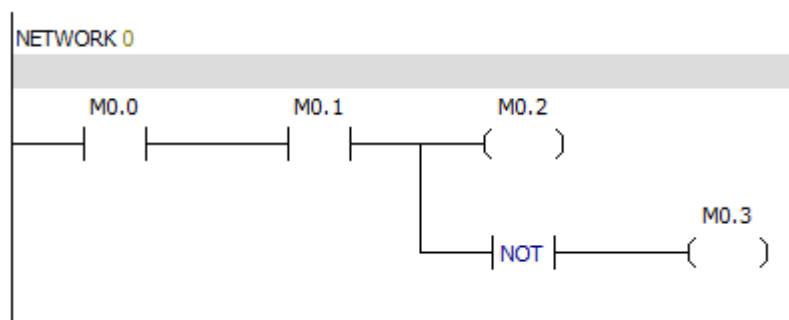
Input / output	Operand	Data type
Bit	I, Q, M, SM, T, C, V, S, L	Boolean
Input (LAD)	Enable bit	Boolean
Input (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean



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

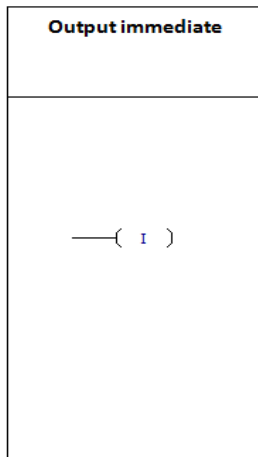
In LAD and FBD, when the output instruction is executed, the PLC will open or close the output bit in the process image register.

Example:



6.1.6 Output immediate

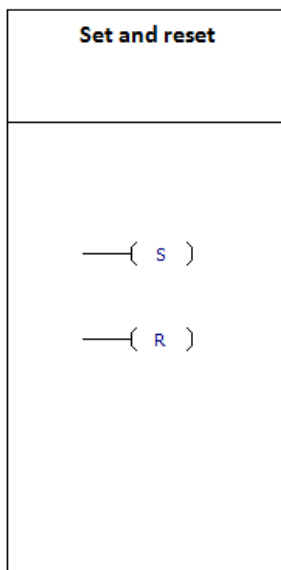
Input / output	Operand	Data type
Bit	Q	Boolean
Input (LAD)	Enable bit	Boolean
Input (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean



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



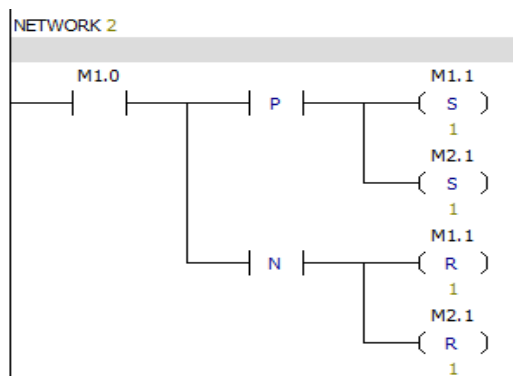
"Set" instruction can make a bit or a series of bits be 1.

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

The value of N is between 1 and 255.

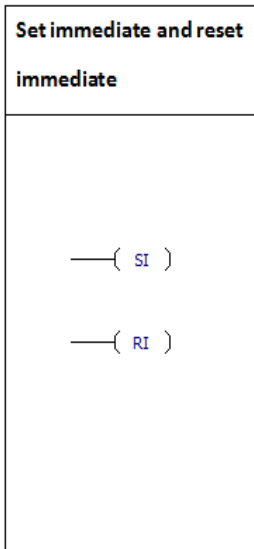
The "reset" instruction can reset the bits of the timer and counter, and can clear the current value of the timer and counter.

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.

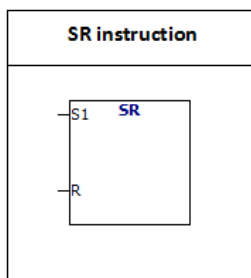
“Reset immediate” can reset many of points immediately.

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 (LAD)	Enable bit	Boolean
S1, R (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
OUT (LAD)	Enable bit	Boolean
OUT (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
xxx	I, Q, M, V, S	Boolean

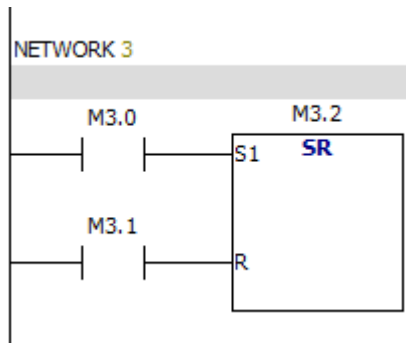


Bistable trigger is a latch. When both R and S1 are equal to 1, the output is 1.

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

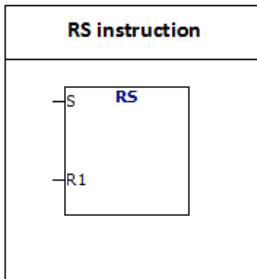
Instruction	S1	R	OUT
SR	0	0	Previous state
	0	1	0
	1	0	1
	1	1	1

Example:



6.1.10 RS instruction

Input / output	Operand	Data type
S, R1 (LAD)	Enable bit	Boolean
S, R1 (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
OUT (LAD)	Enable bit	Boolean
OUT (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
xxx	I, Q, M, V, S	Boolean

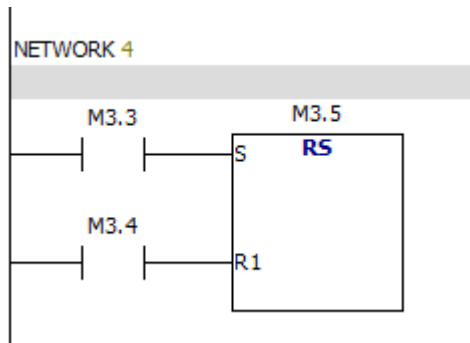


Bistable trigger is a latch. When both R1 and S are equal to 1, the output is 0.

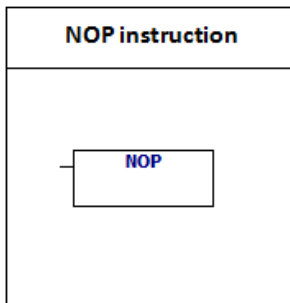
The truth table of the "RS" instruction is as follows:

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

Example:

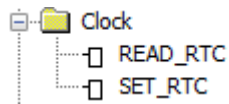


6.1.11 NOP instruction



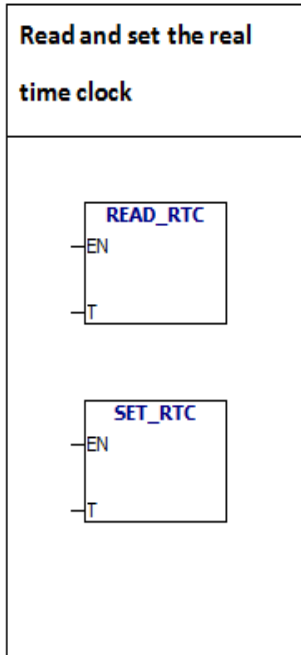
NOP instruction is invalid for user program execution. Can not use NOP instruction in FBD mode. The value of N is between 0 and 255.

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



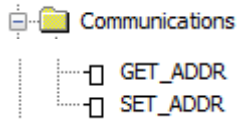
TODR instruction reads the current time and date from the hardware clock and load it into the time buffer of 7bytes starting at the address T.

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.

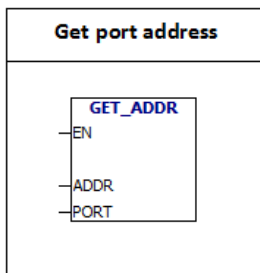
T byte	direction	byte data type
0	second	USINT
1	minute	USINT
2	hour	USINT
3	date	USINT
4	week	USINT
5	month	USINT
6	year	USINT

6.3 Communication



6.3.1 Get port address

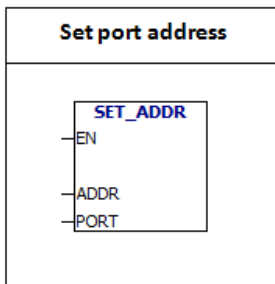
Input / output	Operand	Data type
ADDR	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC	byte
PORT	Constant (0 or 1)	byte



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

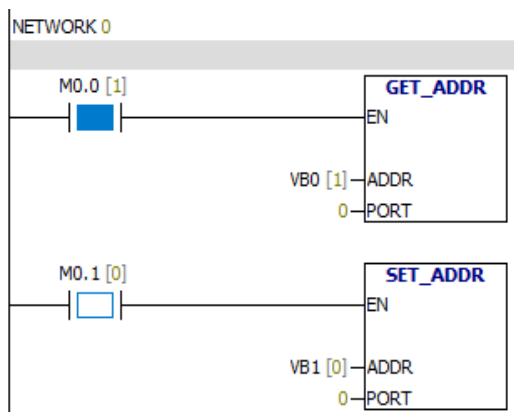
Input / output	Operand	Data type
ADDR	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	byte
PORT	Constant (0 or 1)	byte



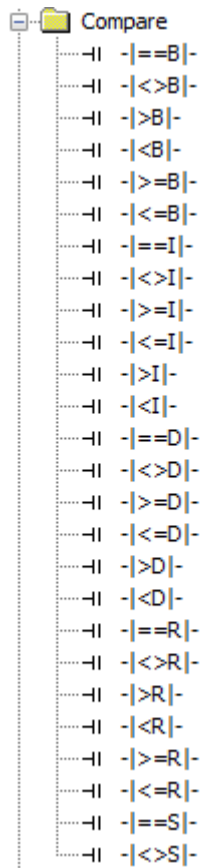
SET- ADDR instruction set the PORT site to the value specified in the ADDR.

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
output (FBD)	I, Q, M, SM, T, C, V, S, L,	Boolean

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.

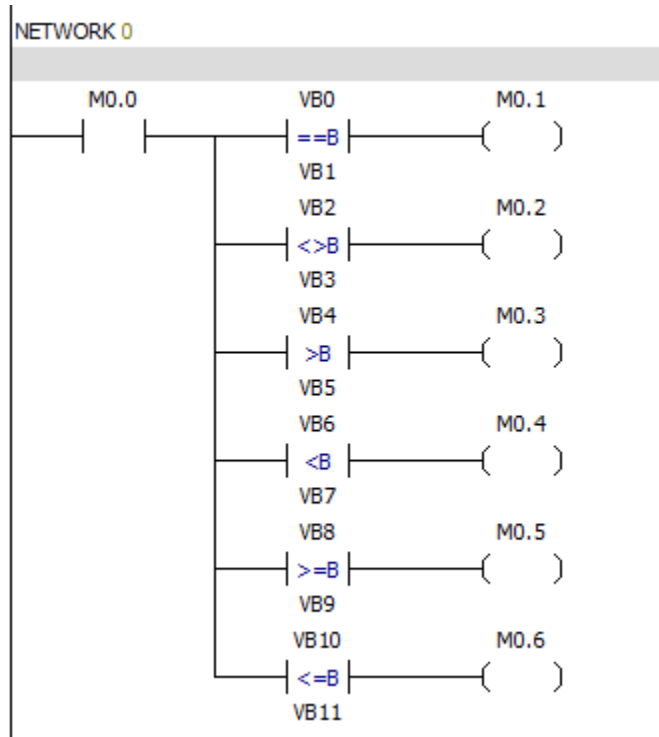
Attention:

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.

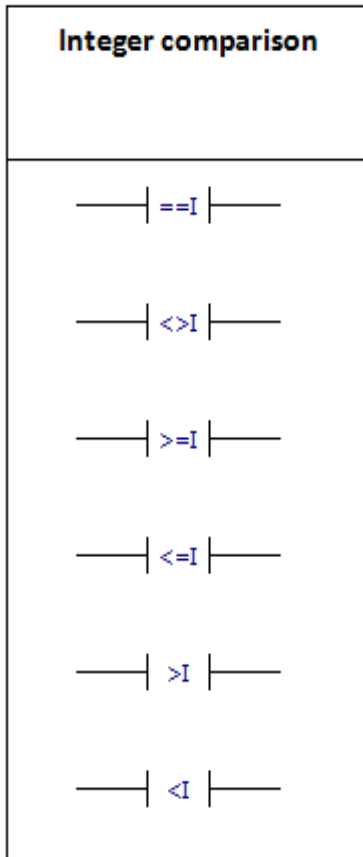
2. Enter the illegal real number

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
output (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean



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.

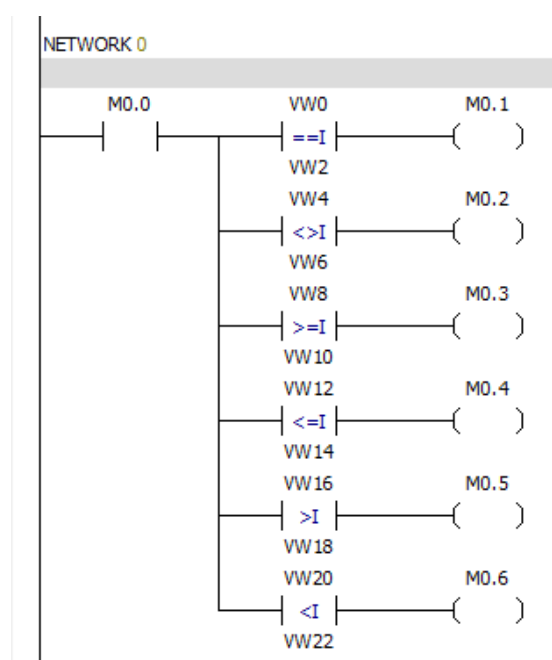
In FBD, when the comparison result is true, the output will be open.

Attention: The following conditions are serious errors .

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

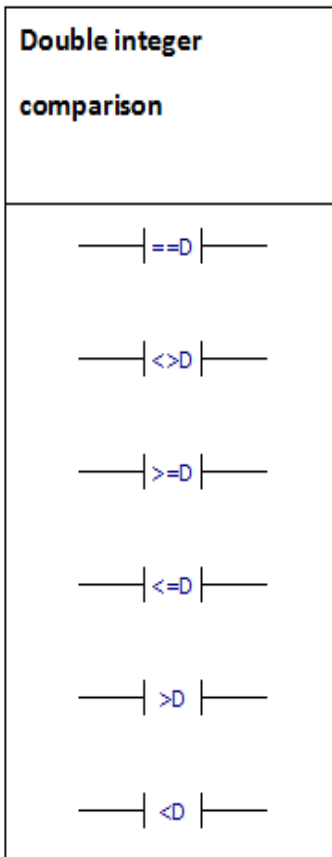
- 1.Enter illegal indirect address.
- 2.Enter the illegal real number.

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
output (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean



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.

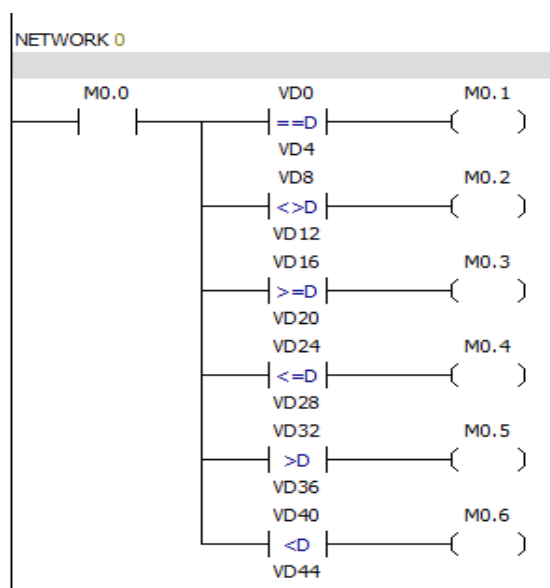
In FBD, when the comparison result is true, the output will be open.

Attention: The following conditions are serious errors .

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

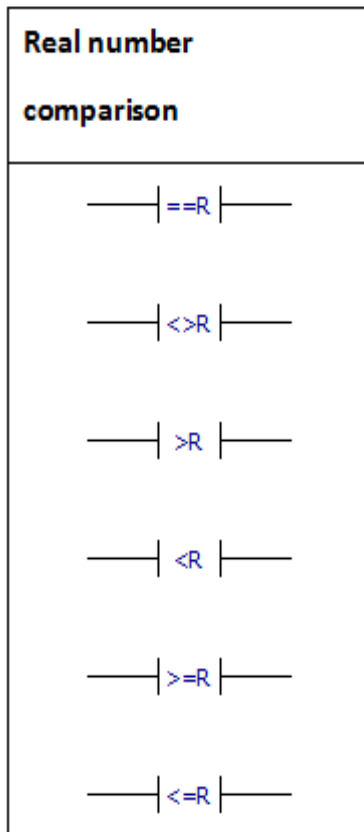
1. Enter illegal indirect address.
2. Enter the illegal real number.

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
Output (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean



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.

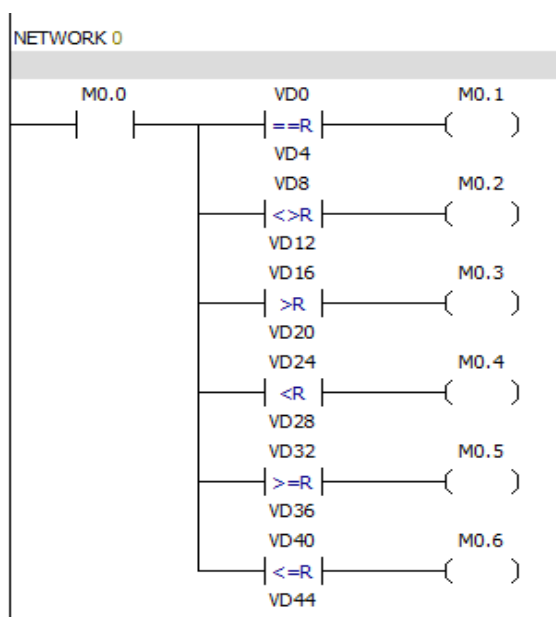
In FBD, when the comparison result is true, the output will be open.

Attention: The following conditions are serious errors .

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

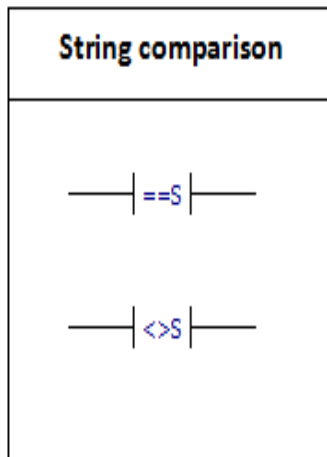
1. Enter illegal indirect address.
2. Enter the illegal real number.

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
Output (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean



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.

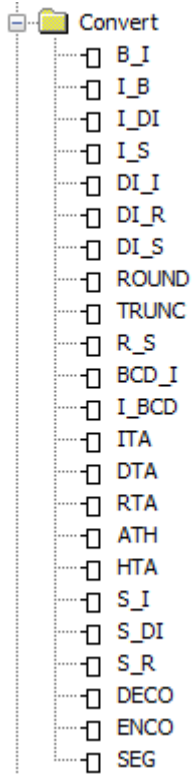
ASCII constant string data type format:

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.

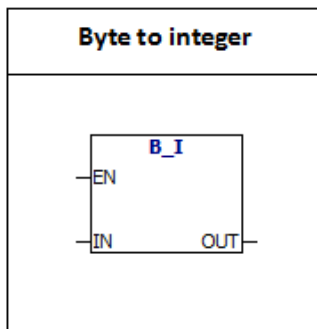
String length	Character 1	Character 2	Character 3	Character 4	Character 5	Character 254
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5		Byte 254

6.5 Convert



6.5.1 Byte to integer

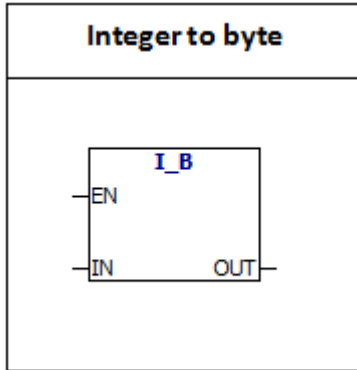
Input / output	Operand	Data type
IN	VB, IB, QB, MB, SB, SMB, LB, AC, constant, *AC, *VD, *LD	Byte
OUT	VW, IW, QW, MW, SW, SMW, LW, AQW, T, C, AC, *VD, *LD, *AC	Integer



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

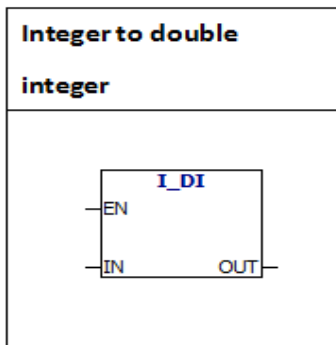
Input / output	Operand	Data type
IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, constant, *VD, *LD, *AC	Integer
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	Byte



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

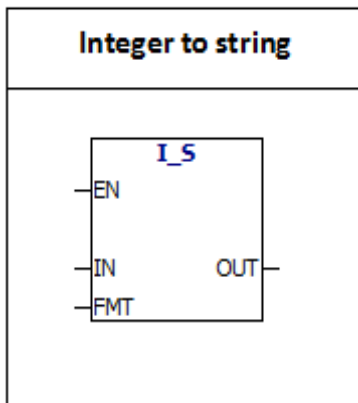
Input / output	Operand	Data type
IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, constant, *VD, *LD, *AC	Integer
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	Double-integer



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:



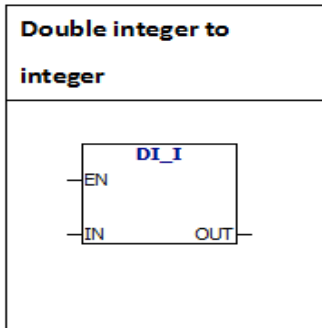
C = comma (1) or decimal point (0)

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

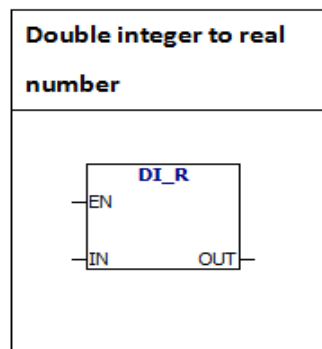


Double integer to integer : DI-I instruction converts the value of double integer to a integer value, and the result is inserted into the variable specified by the OUT.

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

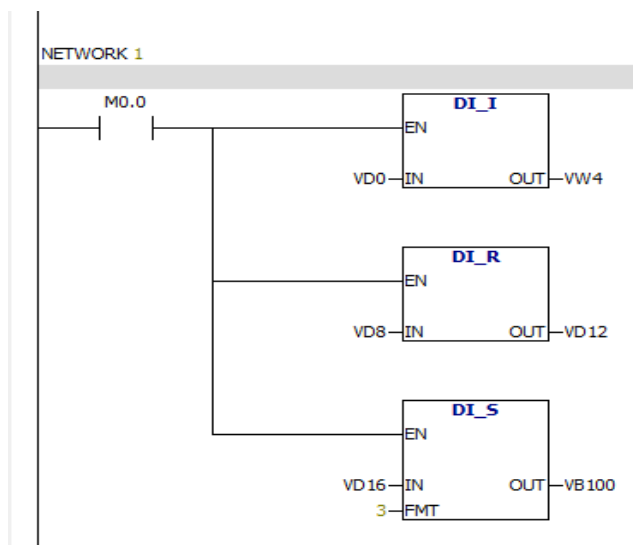
6.5.6 Double integer to real number

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



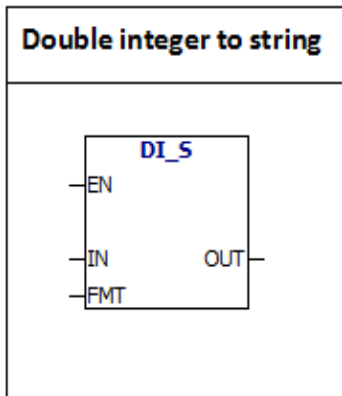
Double integer to real number: Instruction converts 32 bit signed integer to 32 bit real number, and the result is inserted into the variable specified by the OUT.

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.

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:



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

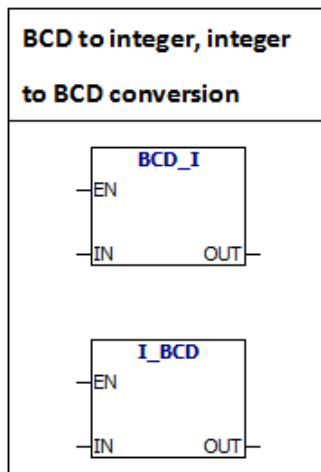
nnn = The number of digits on the right side of the decimal point

The length of the output string is always 12 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 12 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.

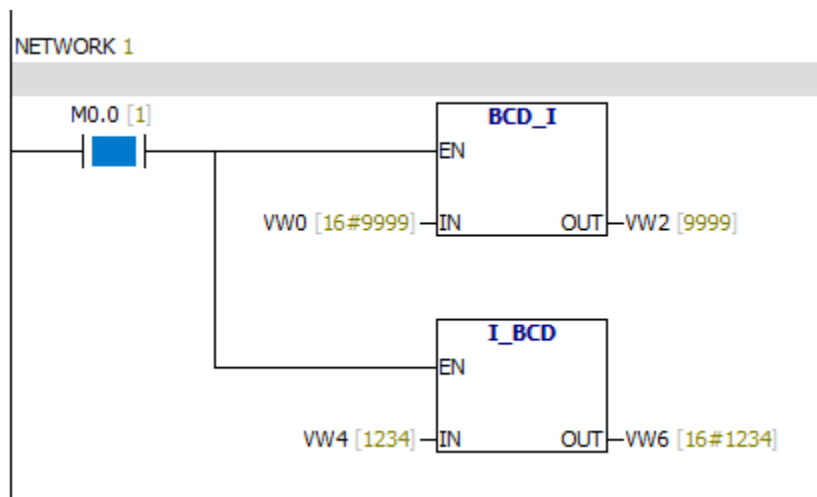
6.5.8 BCD to integer, integer to BCD conversion

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



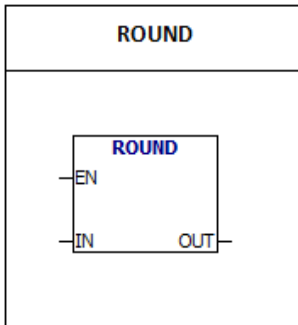
The BCD -I instruction converts the binary coded decimal value to the integer value, and loads the result into the variable specified by the OUT. "IN" BCD value range is 0 to 9999. Integer to BCD instruction converts the integer value to binary coded decimal value and loads the result into the variable specified by the OUT. The range of input values is 0 to 9999.

Example:



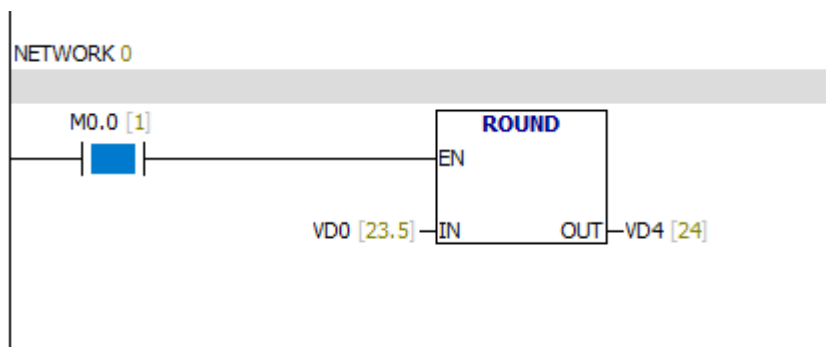
6.5.9 ROUND

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



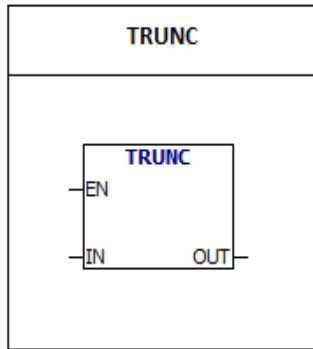
The ROUND instruction converts the real number value to a double integer value and the result is inserted into the variable specified by the OUT. If the fractional part is equal to or greater than 0.5, the integer part will be added to 1.

Example:



6.5.10 TRUNC

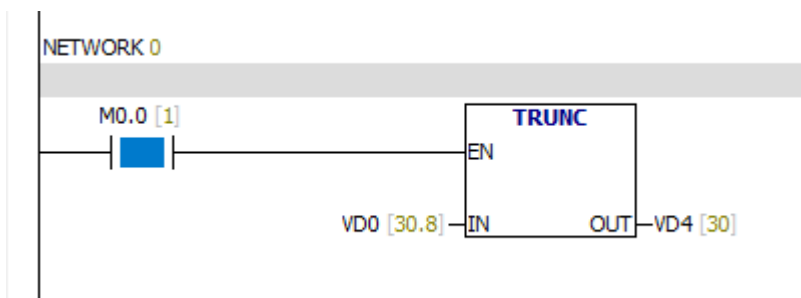
Input/output	Operand	Data type
IN	VD, ID, QD, MD, SD, SMD, LD, AC, constant, *VD, *LD, *AC	Real number
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	Double integer



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

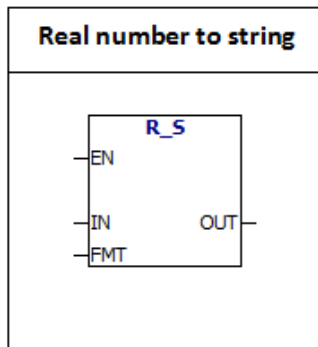
Only the integer part of the real number is converted, and the fractional part is discarded.

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)

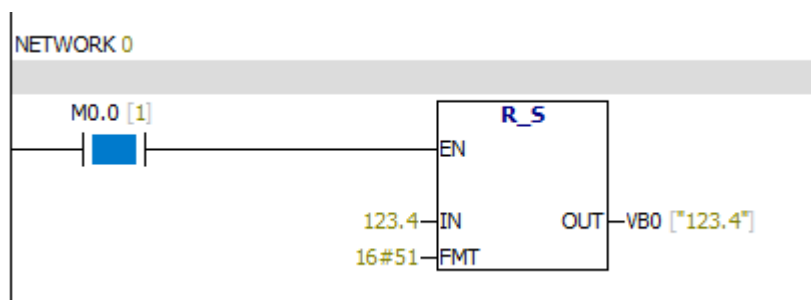
nnn = The number of characters of the right of the decimal point.

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

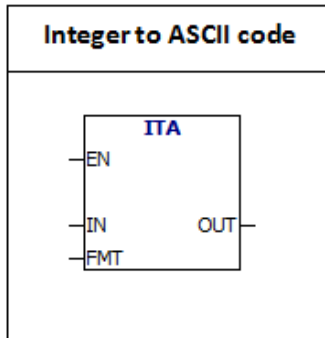
1. Positive number is written to output buffer without a sign.
2. Negative number is written to the output buffer with “-” .
3. The starting zero on the left of the decimal point is compressed.
4. The size of the output string must be 3 bytes larger than “nnn”
5. The value in the output string must be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

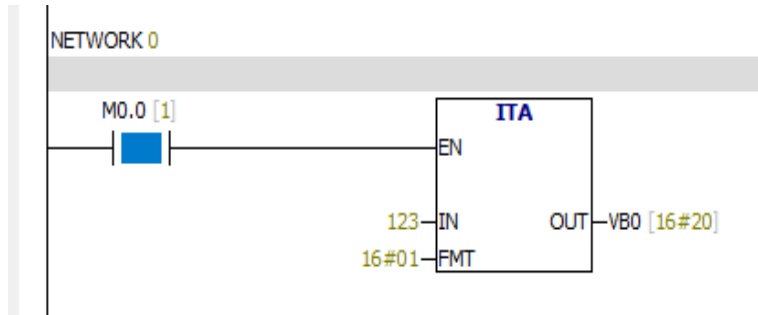
Prompt: The output according to the following rules

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

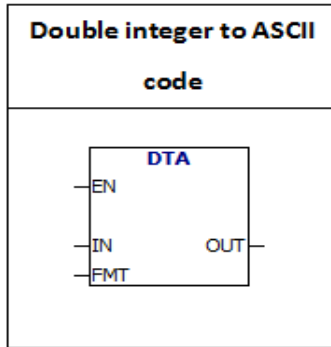
As shown in Figure: The integer input is 123; nnn=1

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

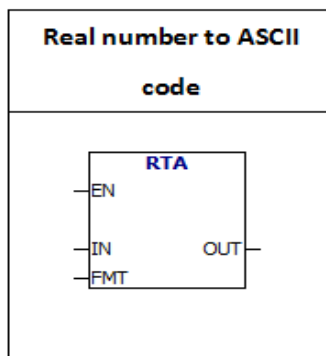
1. Positive number is written to output buffer without a sign.
2. Negative number is written to the output buffer with “-” .
3. The starting zero on the left of the decimal point is compressed.
4. The value in the output string must be aligned to the right.

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:

$nnn > 5$

$ssss < 3$

$ssss < \text{Number of characters in OUT}$

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:

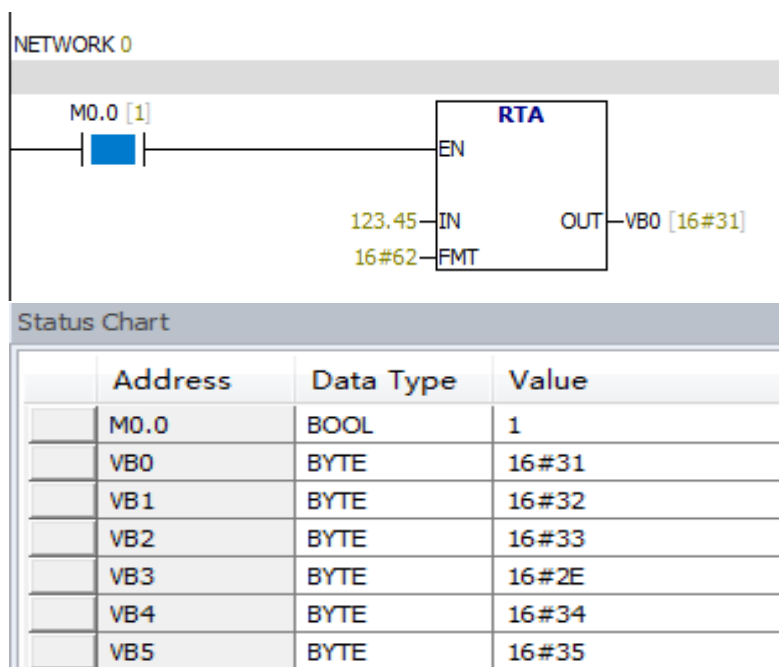
1. Positive number is written to output buffer without a sign.
2. Negative number is written to the output buffer with “-”.
3. The starting zero on the left of the decimal point is compressed.
4. The number of characters of the right of the decimal point is equal to the value of “nnn”.

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 +1	OUT +2	OUT +3	OUT +4	OUT +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:

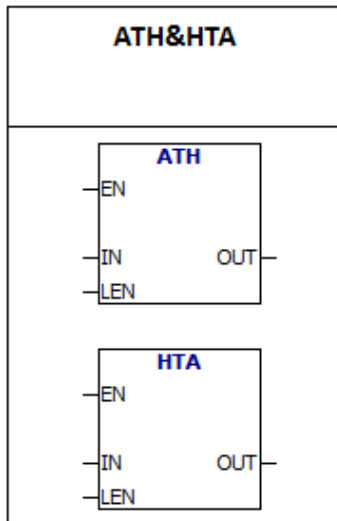
VB0	VB1	VB2	VB3	VB4	VB5
16#31	16#32	16#33	16#2E	16#34	16#35
1	2	3	.	4	5

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.

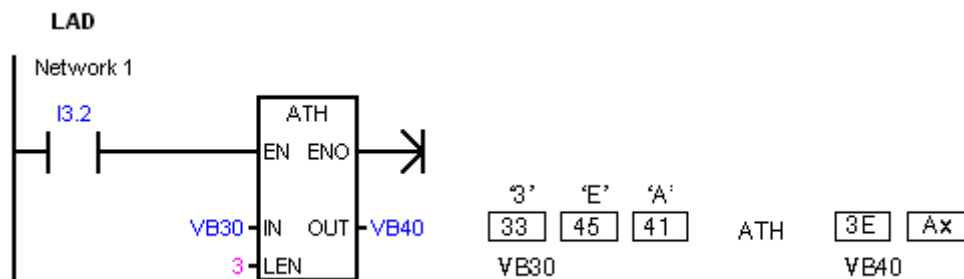
Valid ASCII input character:

Numbers 0 to 9 and capital letters A to F.

ASCII Codes: 30 to 39 and 41 to 46.

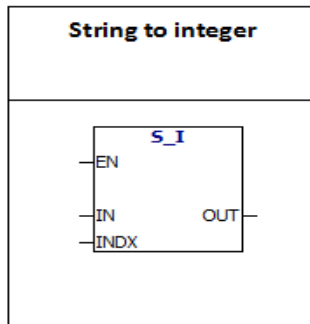
Error condition: Illegal ASCII code

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

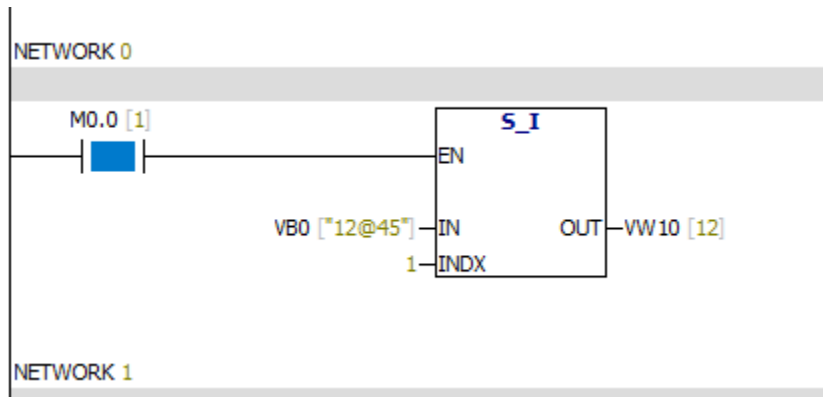
Valid Input Strings for String to Real

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:



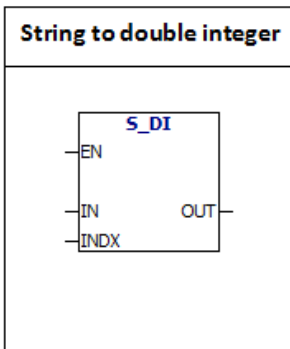
Status Chart

	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

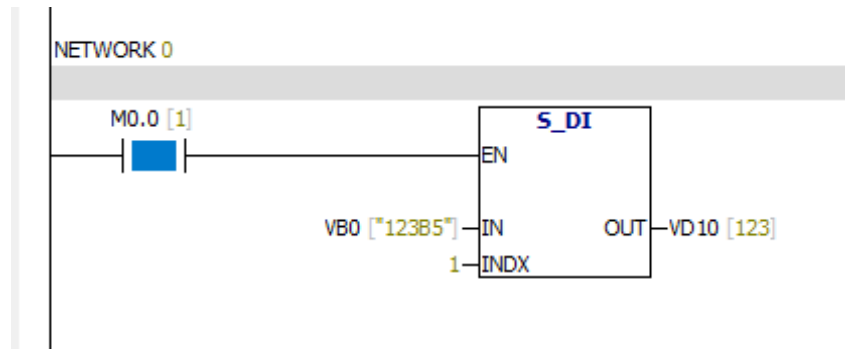
Valid Input Strings for String to Real

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

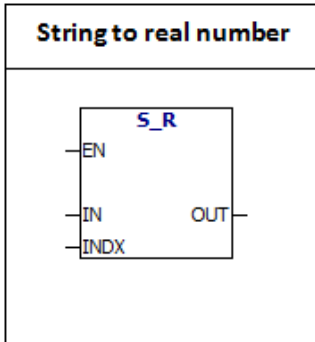
Address	Data Type	Value
M0.0	BOOL	1
VB0	BYTE	16#05
VB1	BYTE	16#31
VB2	BYTE	16#32
VB3	BYTE	16#33
VB4	BYTE	16#42
VB5	BYTE	16#35

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

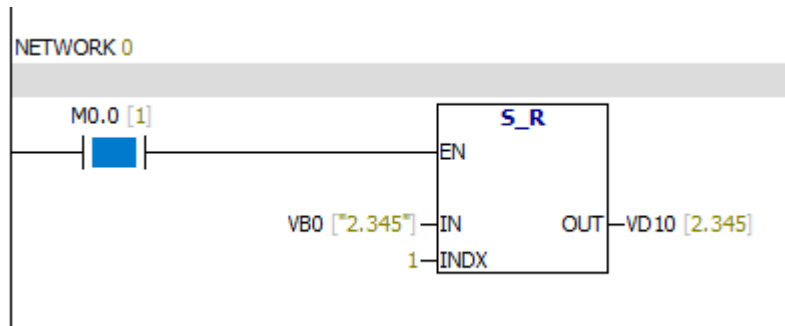
Valid Input Strings for String to Real

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:



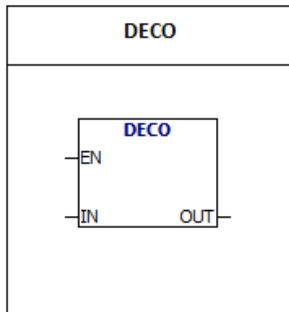
Status Chart

	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

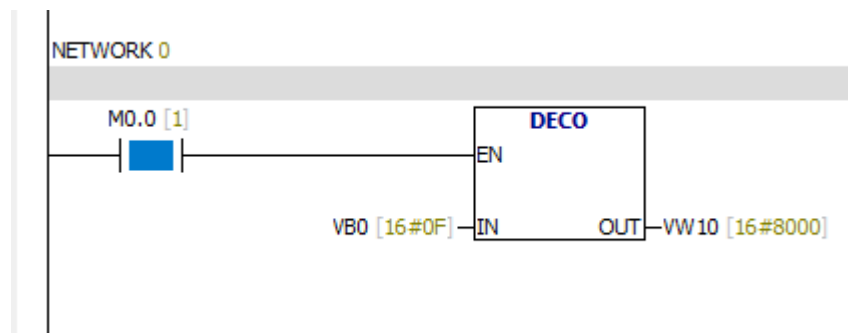
Input/output	Operand	Data type
IN	VB, IB, QB, MB, SMB, LB, SB, AC, constant, *VD, *LD, *AC	Byte
OUT	VW, IW, QW, MW, SMW, LW, SW, AQW, T, C, AC, *VD, *AC, *LD	word



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:

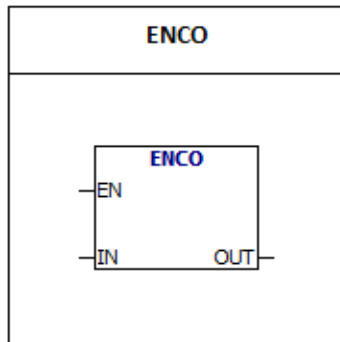


The low four bits value of VB0 is 15, the 15th bit of the VW10 is equal to 1.

The other bits of the VW10 are set to 0.

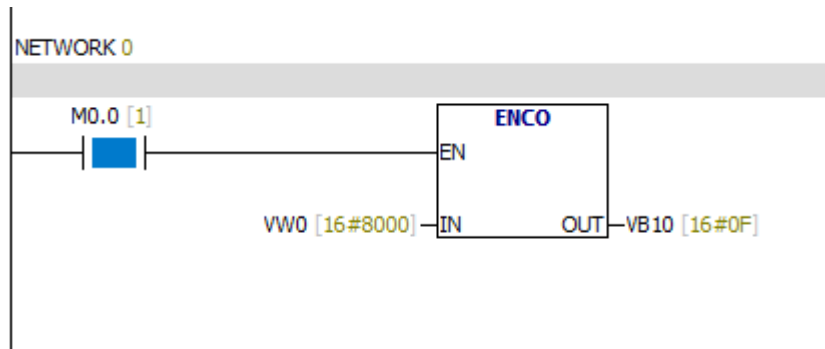
6.5.20 ENCO

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.

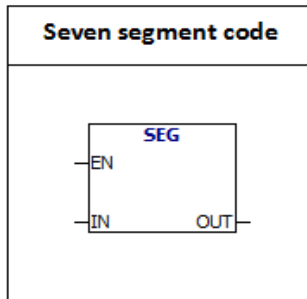
Example:



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 segment.

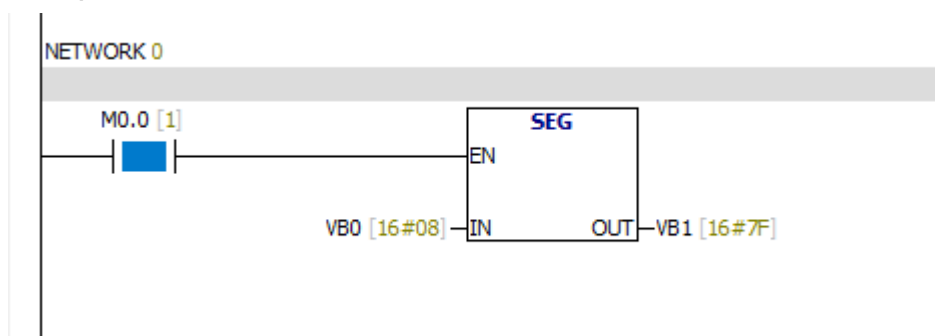
The low four bits value of input byte is converted.

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

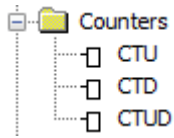
Example:



Analysis:

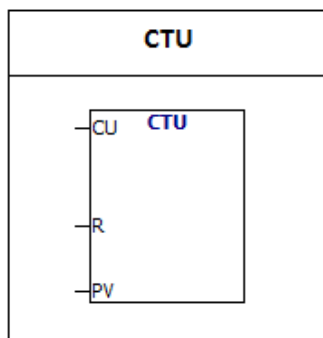
The low four bits value of VB0 is 8. The value of the output byte VB1 is 16#7F. The result of converting VB1 to binary is 2# 0111 1111 .

6.6 Counter



6.6.1 CTU

Input/output	Operand	Data type
C xxx	Constant(C0—C255)	Word
CU (LAD)	Enable bit	Boolean
CU (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
R (LAD)	Enable bit	Boolean
R (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
PV	VW, IW, QW, MW, SMW, LW, AIW, AC, T, C, constant, *VD, *AC, *LD, SW	Integer

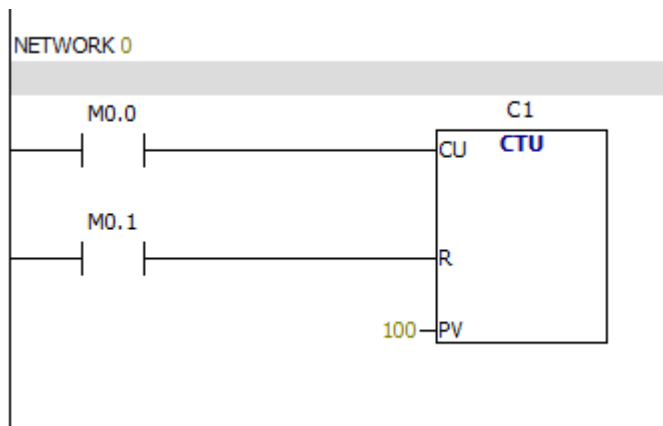


CU bit gets a high level and the current value of the counter plus 1. When the current value is greater than or equal to the preset value, the counter bit opens. When R gets a high level, the counter is restored. The maximum value of the counter is 32767.

Counter range: C xxx=C0 ~ C255

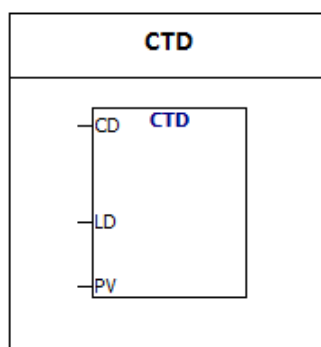
The counter number of each counter is different.

Example:



6.6.2 CTD

Input/output Operand		Data type
Cxxx	Constant(C0—C255)	Word
CD (LAD)	Enable bit	Boolean
CD (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
LD (LAD)	Enable bit	Boolean
LD (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
PV	VW, IW, QW, MW, LW, SMW, AC, T, C, AIW, constant, *VD, *AC, *LD, SW	Integer

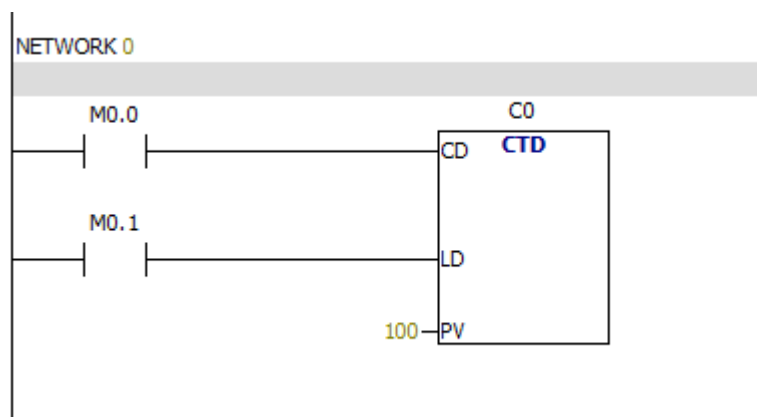


The bit of CD is converted from 0 to 1 and the current value minus 1. When the current value is equal to 0, the counter is opened and counter stops count. When the LD bit is equal to 1, counter bit is restored and the preset value is loaded into the current value.

Counter range: C xxx=C0-C255

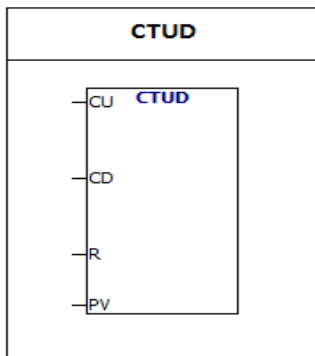
Attention: The counter number of each counter is different.

Example:



6.6.3 CTUD

Input/output	Operand	Data type
C xxx	Constant(C0—C255)	word
CU, CD (LAD)	Enable bit	Boolean
CU, CD (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
R (LAD)	Enable bit	Boolean
R (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
PV	VW, IW, QW, MW, LW, SMW, AC, T, C, AIW, constant, *VD, *AC, *LD, SW	Integer



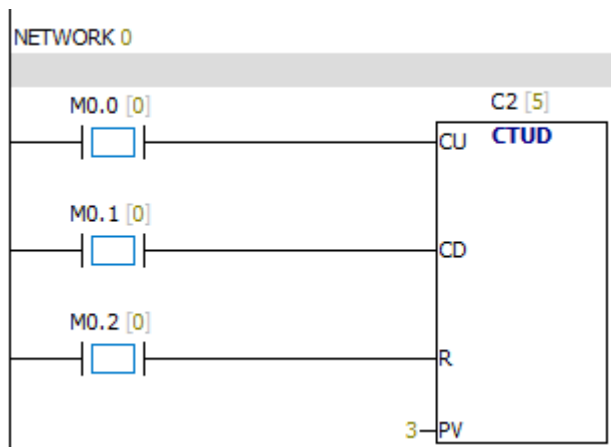
CU bit gets a high level and the current value of the counter plus 1. The bit of CD is converted from 0 to 1 and the current value minus 1. When the current value is greater than or equal to the preset value, the counter bit opens. The maximum value of the counter is 32767, and the minimum value is -32768.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

counter is restored.

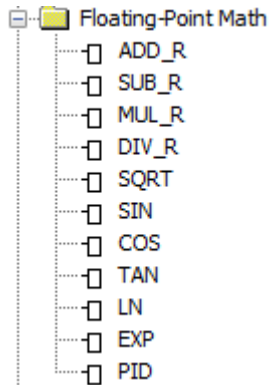
Counter range: C xxx=C0~C255

Attention: The counter number of each counter is different.

Example:

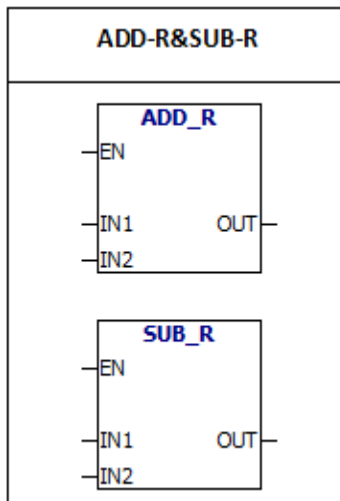


6.7 Floating point calculation



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.

In LAD and FBD: $IN1 + IN2 = OUT$

$IN1 - IN2 = OUT$

Special memory bit:

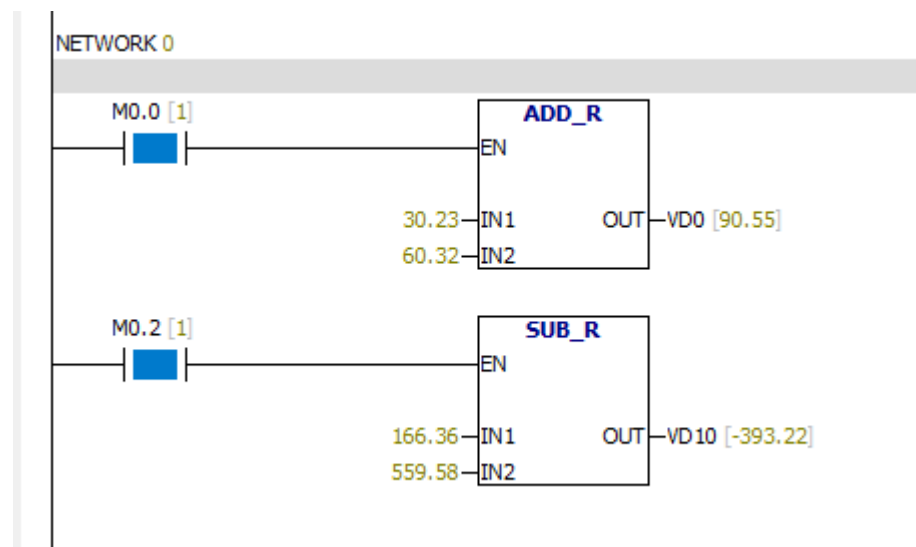
SM1.0 Zero result

SM1.1 Overflow

SM1.2 Negative result

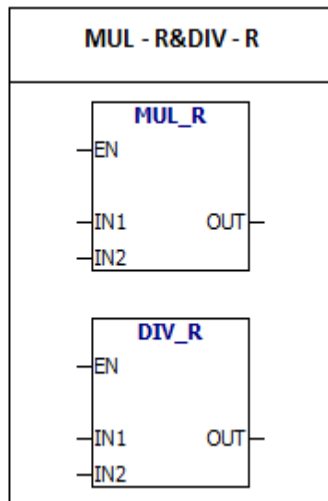
SM1.1 is used to indicate overflow errors and illegal values.

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.

In LAD and FBD: $IN1 * IN2 = OUT$

$IN1 / IN2 = OUT$

error conditions:

SM1.1 Overflow

SM1.3 The divisor is 0

Special memory bit:

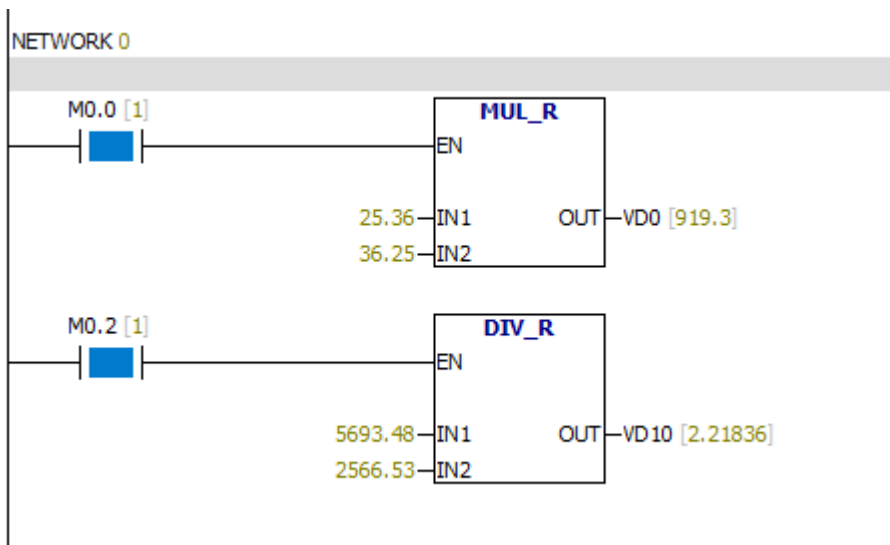
SM1.0 Zero result

SM1.1 Overflow

SM1.2 Negative result

SM1.3 The divisor is 0

Example:



6.7.3 SQRT

Input/output Operand

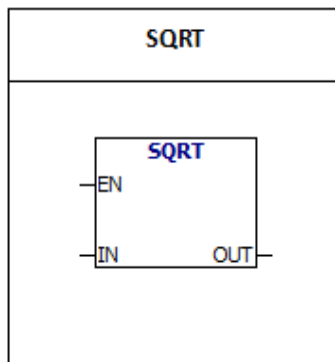
Data type

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

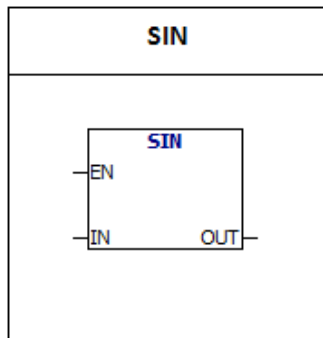
SM1.1 Overflow

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:

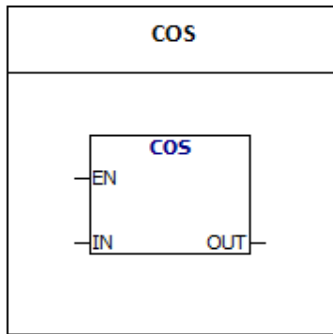
SM1.0 Zero result

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:

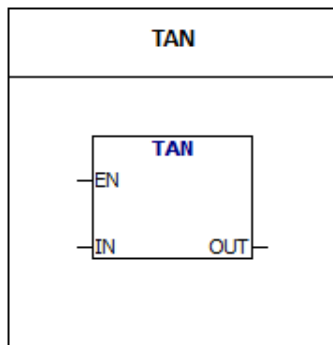
SM1.0 Zero result

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:

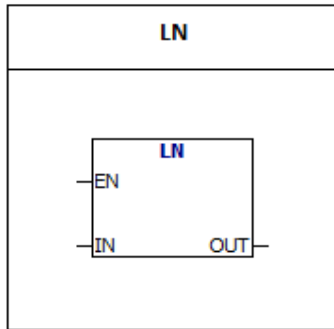
SM1.0 Zero result

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:

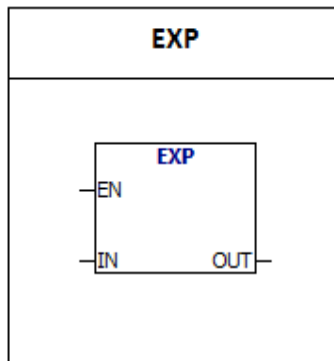
SM1.0 Zero result

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:

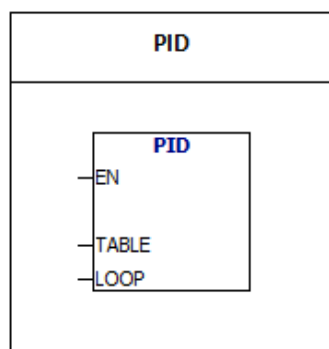
SM1.0 Zero result

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 not 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

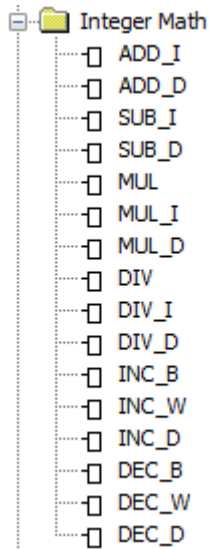
Differential term:

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

- MD_n : Differential term

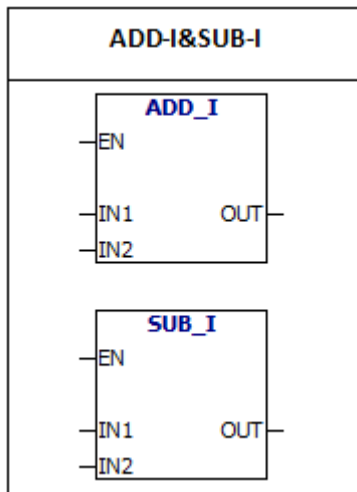
- K_c : gain
- T_D : Differential time
- T_s : Sampling time
- PV_{n-1} : Last time process variable
- PV_n : Process variable

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.

In LAD and FBD: $IN1 + IN2 = OUT$

$$IN1 - IN2 = OUT$$

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

error conditions:

0006 Indirect address

SM1.1 overflow

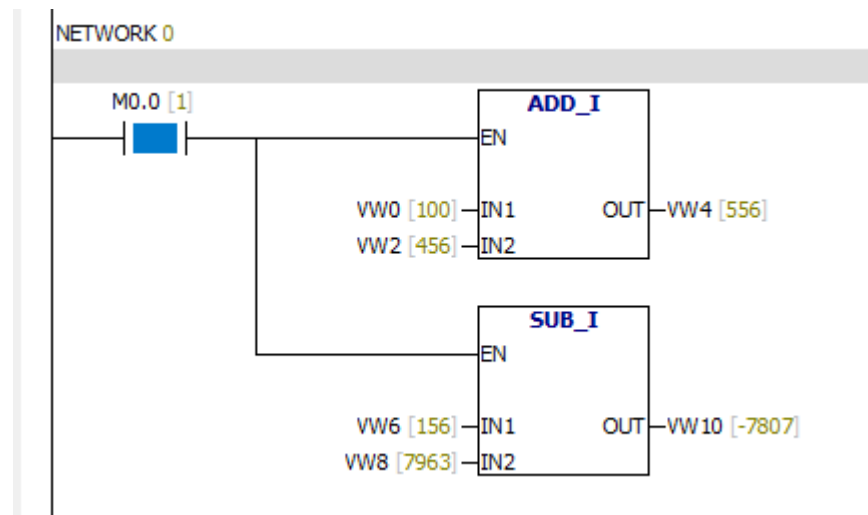
Special memory bit:

SM1.0 Zero result

SM1.1 overflow

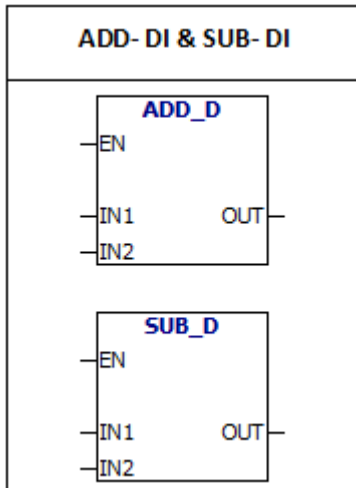
SM1.2 Negative result

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.

In LAD and FBD: $IN1 + IN2 = OUT$

$IN1 - IN2 = OUT$

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

error conditions:

0006 Indirect address

SM1.1 overflow

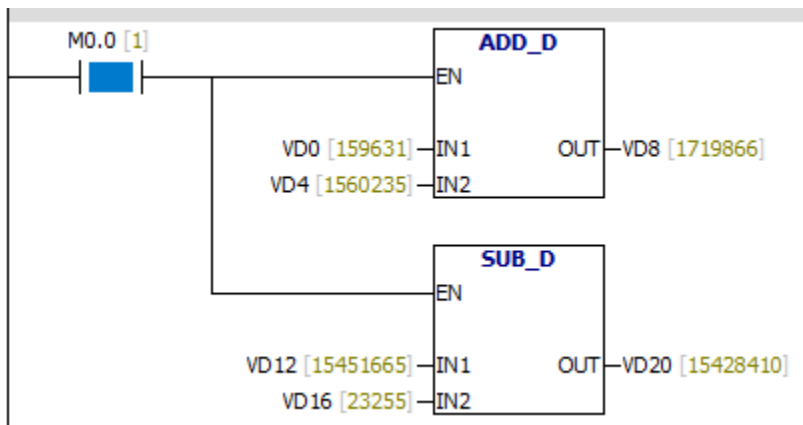
Special memory bit:

SM1.0 Zero result

SM1.1 overflow

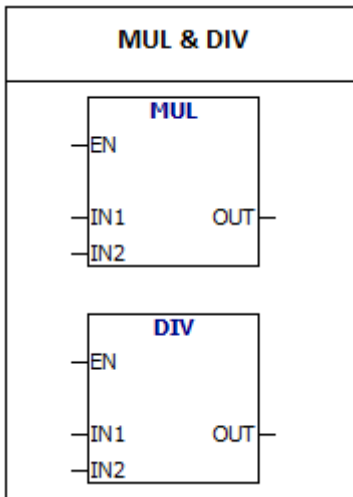
SM1.2 Negative result

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).

In LAD and FBD: $IN1 * IN2 = OUT$

$IN1 / IN2 = OUT$

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:

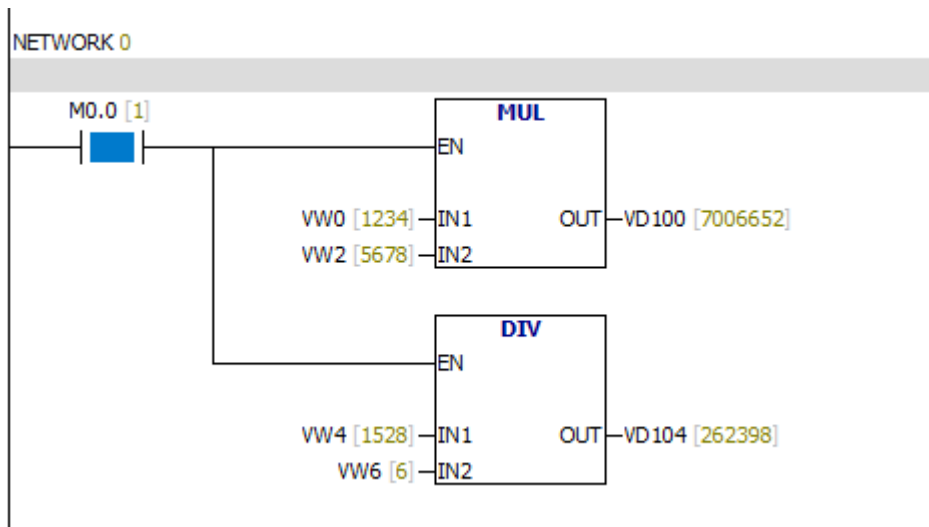
SM1.0 Zero result

SM1.1 overflow

SM1.2 Negative result

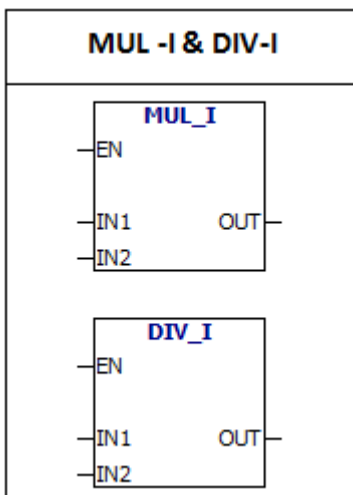
SM1.3 The divisor is 0

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.

In LAD and FBD: $IN1 * IN2 = OUT$

$IN1 / IN2 = OUT$

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:

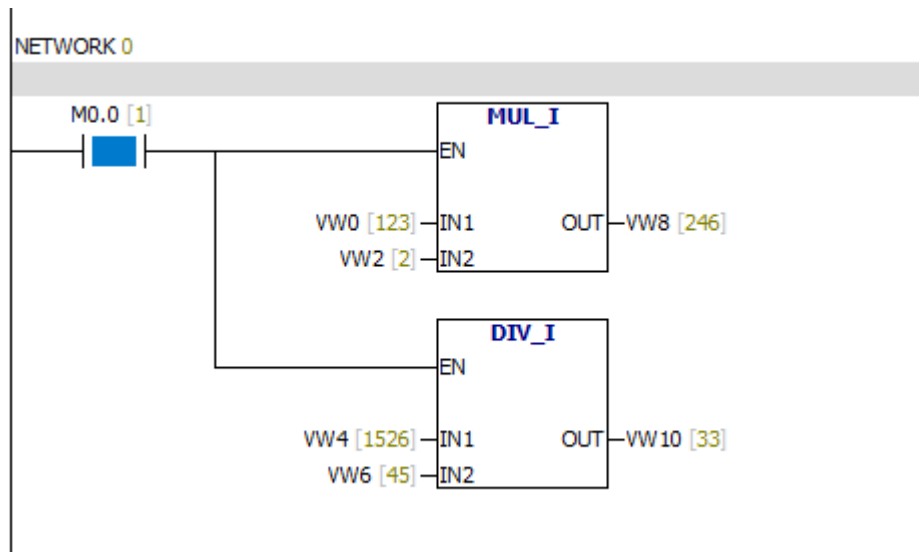
SM1.0 Zero result

SM1.1 overflow

SM1.2 Negative result

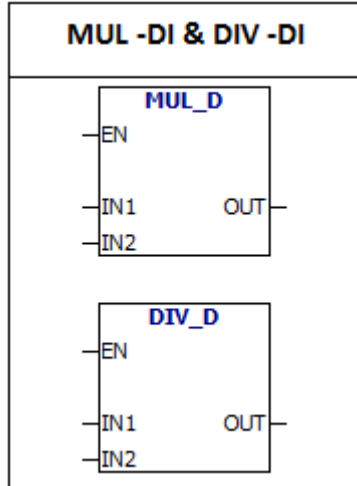
SM1.3 The divisor is 0

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.

In LAD and FBD: $IN1 * IN2 = OUT$

$IN1 / IN2 = OUT$

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:

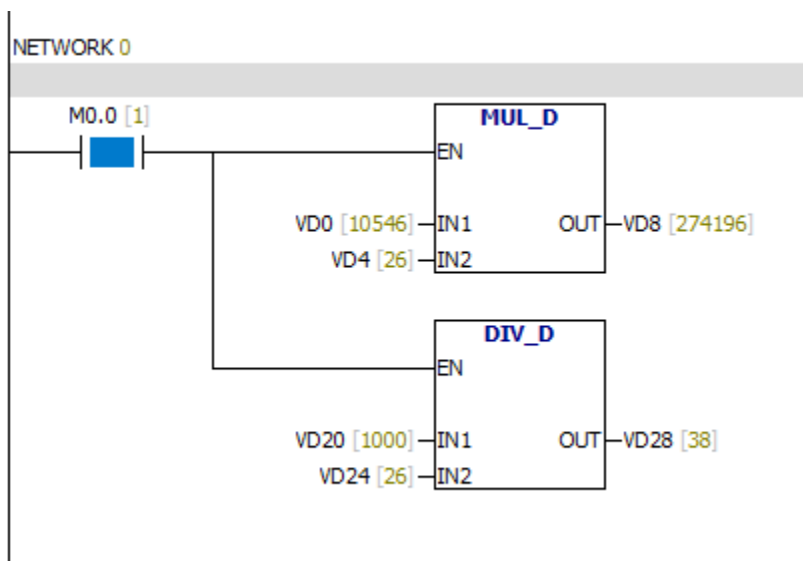
SM1.0 Zero result

SM1.1 overflow

SM1.2 Negative result

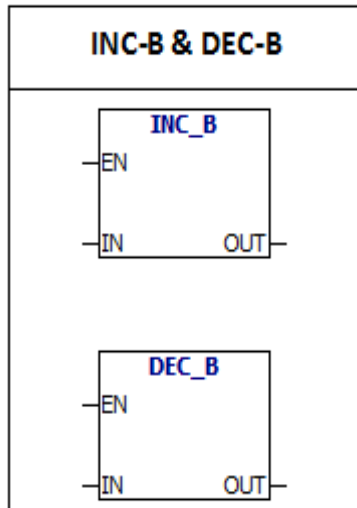
SM1.3 The divisor is 0

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 LAD and FBD: $IN + 1 = OUT$

$IN - 1 = OUT$

error conditions:

0006 Indirect address

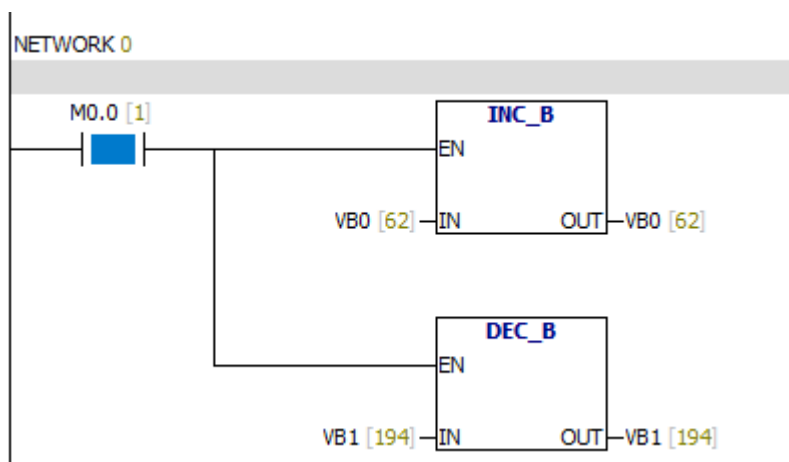
SM1.1 overflow

Special memory bit:

SM1.0 Zero result

SM1.1 overflow

Example:



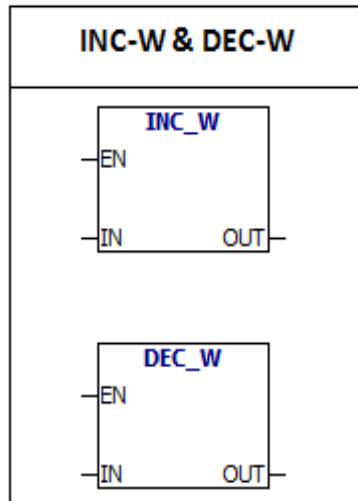
6.8.7 INC-W & DEC-W

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 LAD and FBD: $IN + 1 = OUT$

$IN - 1 = OUT$

error conditions:

0006 Indirect address

SM1.1 overflow

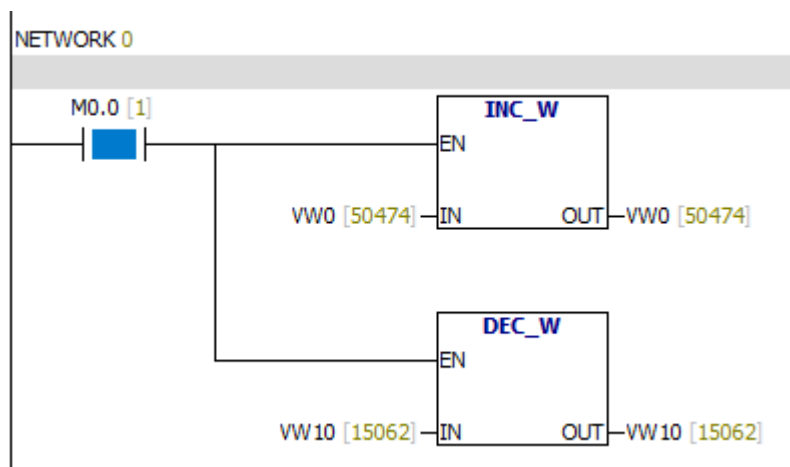
Special memory bit:

SM1.0 Zero result

SM1.1 overflow

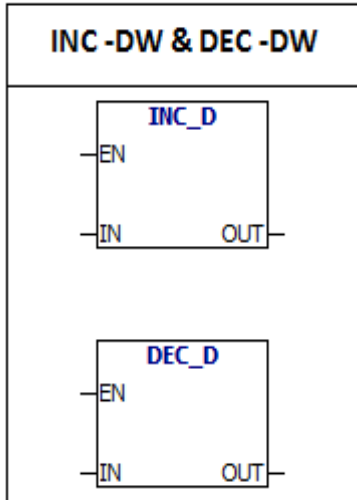
SM1.2 Negative result

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 LAD and FBD: $IN + 1 = OUT$

$IN - 1 = OUT$

The two instructions operations take with symbols.

(16#7FFFFFFF > 16#80000000).

error conditions:

0006 Indirect address

SM1.1 overflow

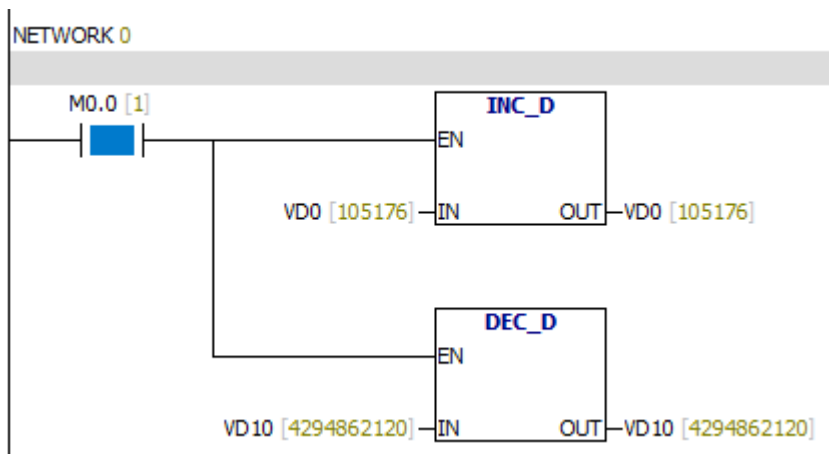
Special memory bit:

SM1.0 Zero result

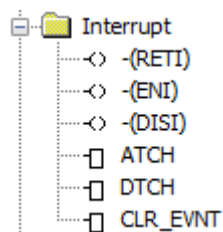
SM1.1 overflow

SM1.2 Negative result

Example:



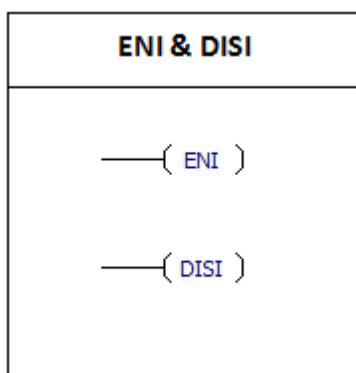
6.9 Interrupt



6.9.1 ENI & DISI

Operand Data type

Nothing Nothing



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

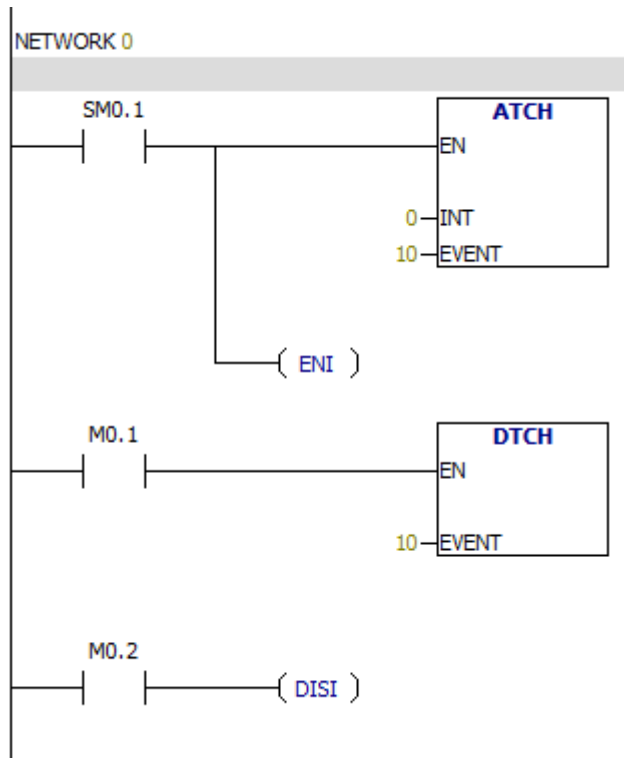
Interrupt disable (DISI): If the instruction is activated, all interrupts can not be used.

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

Interrupt Events:

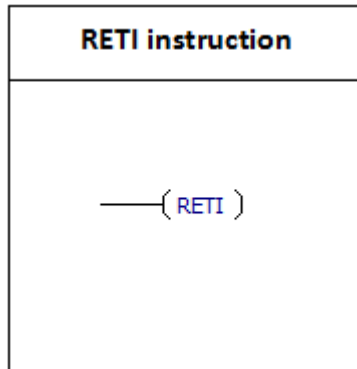
I1.2 Rising edge	PLC_EVENT_INPUTP0	0	Highest priority
I1.4 Rising edge	PLC_EVENT_INPUTP1	1	High priority
Timer interrupt 0	PLC_EVENT_TIMER0	10	Low priority
Timer interrupt 1	PLC_EVENT_TIMER1	11	Lowest priority

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

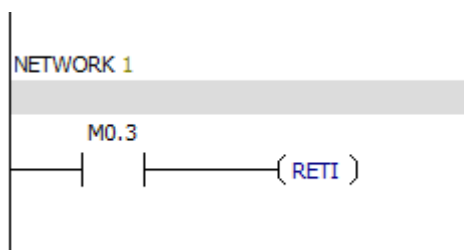
Interrupt use guide

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.

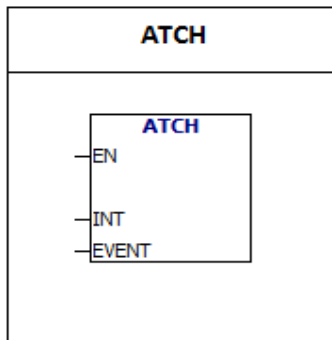
Limit: DISI, ENI, HDEF, LSCR, and END instructions can not be used in the interrupt routine.

Example:



6.9.3 ATCH

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



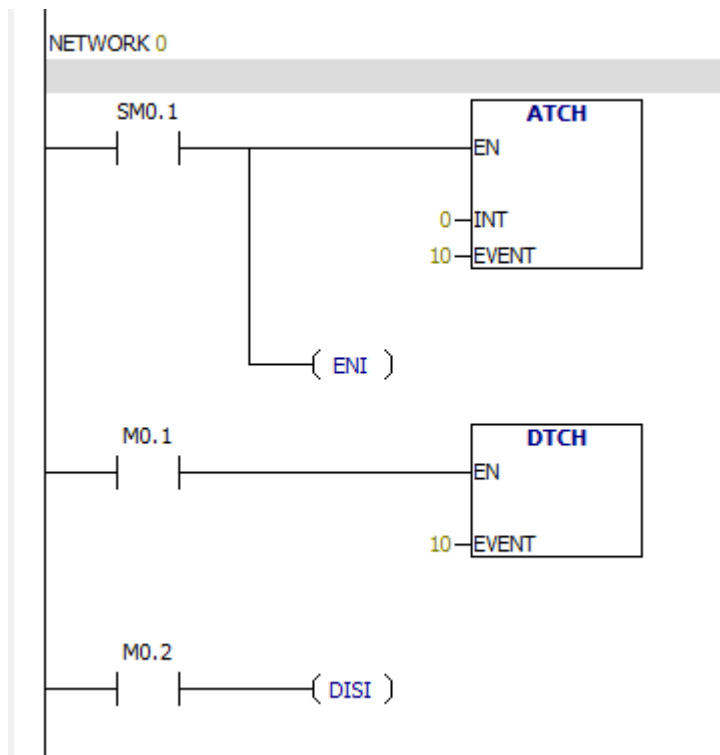
ATCH: The interrupt event (EVNT) is connected to the interrupt routine number (INT) by the "ATCH" instruction, and then activates the interrupt event.

You can attach more than one interrupt events to an interrupt routine. However, an interrupt event can not 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

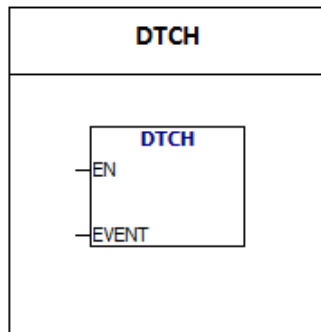
Example:



“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.

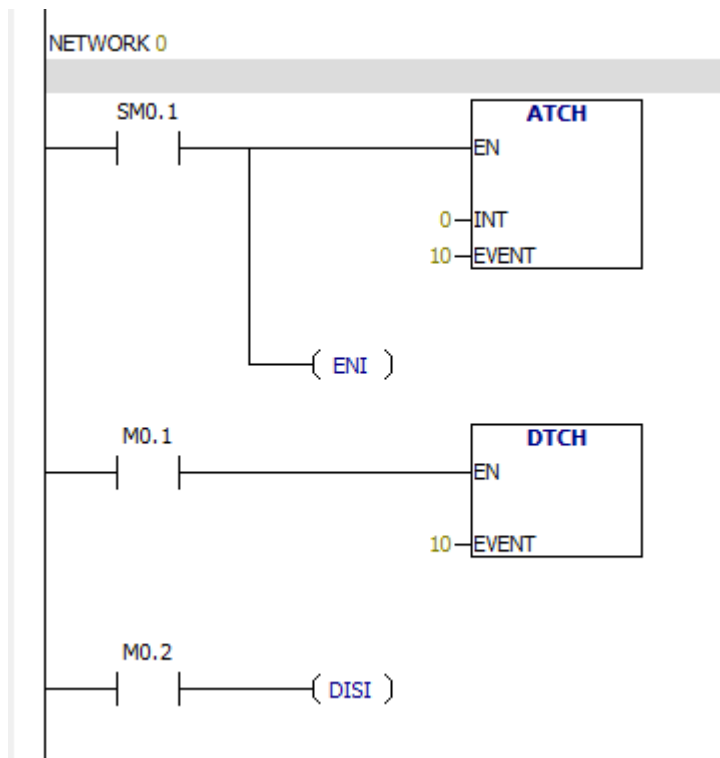
The interrupt event (EVNT) is connected to the interrupt routine number (INT) by the "ATCH" instruction, and then activates the interrupt event.

You can attach more than one interrupt events to an interrupt routine. However, an interrupt event can not 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:

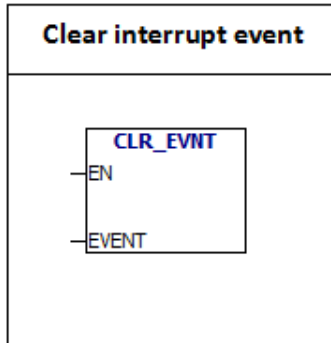
I1.2 Rising edge	PLC_EVENT_INPUTP0	0	Highest priority
I1.4 Rising edge	PLC_EVENT_INPUTP1	1	High priority
Timer interrupt 0	PLC_EVENT_TIMER0	10	Low priority
Timer interrupt 1	PLC_EVENT_TIMER1	11	Lowest priority

Example:



6.9.5 Clear interrupt event

Input/output	Operand	Data type
EVNT	Constant	Byte

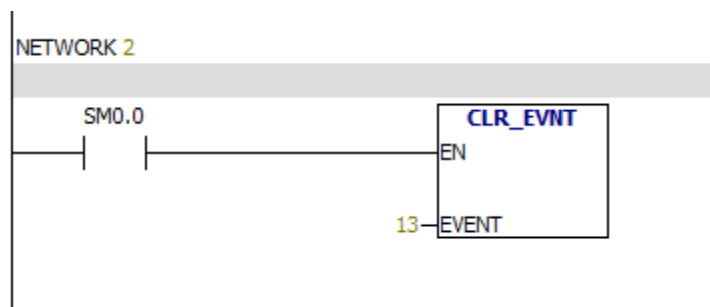


CLR - EVNT instruction will remove all types of EVNT interrupt events in interrupt queue. This instruction is used for removing unnecessary interrupts.

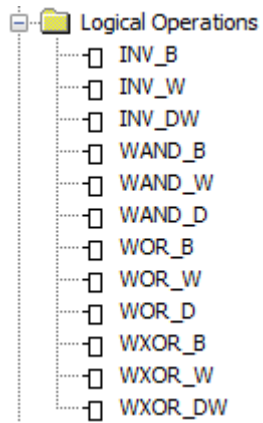
Interrupt Events:

I 1.2 Rising edge	PLC_EVENT_INPUTP0	0	Highest priority
I 1.4 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:

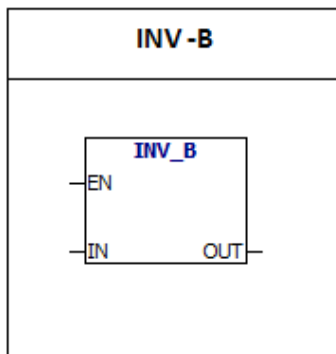


6.10 Logic operation



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.

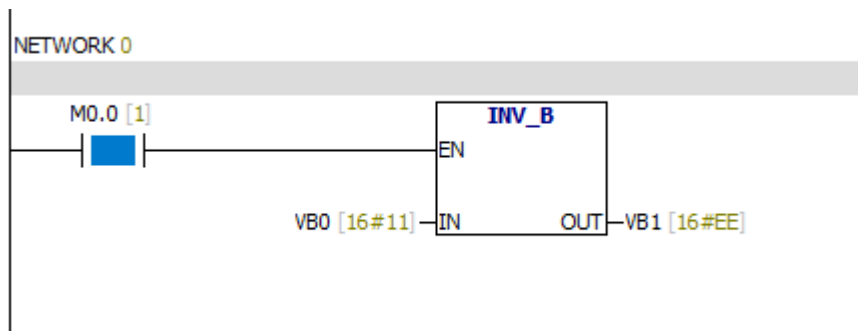
error condition:

0006 Indirect address

Special memory bit:

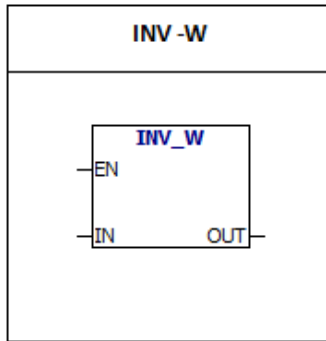
SM1.0 Zero result

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.

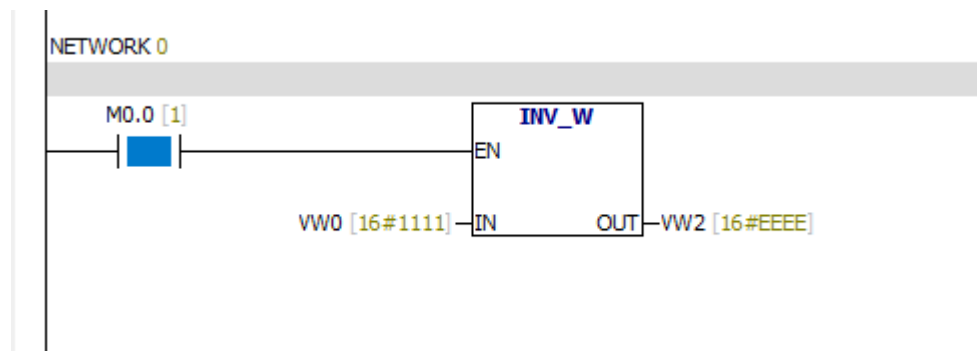
error condition:

0006 Indirect address

Special memory bit:

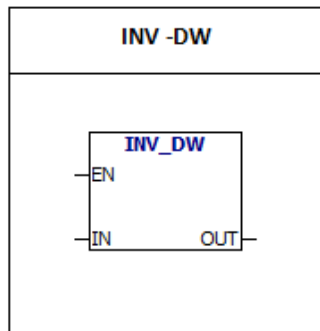
SM1.0 Zero result

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.

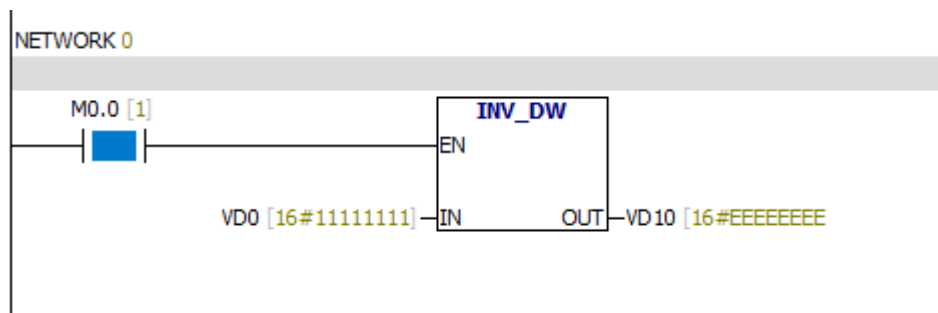
error condition:

0006 Indirect address

Special memory bit:

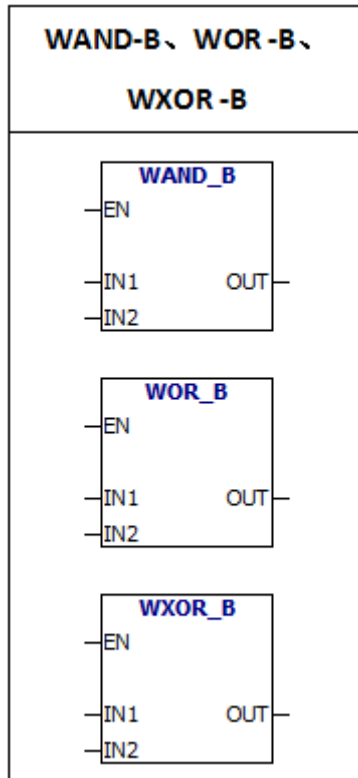
SM1.0 Zero result

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.

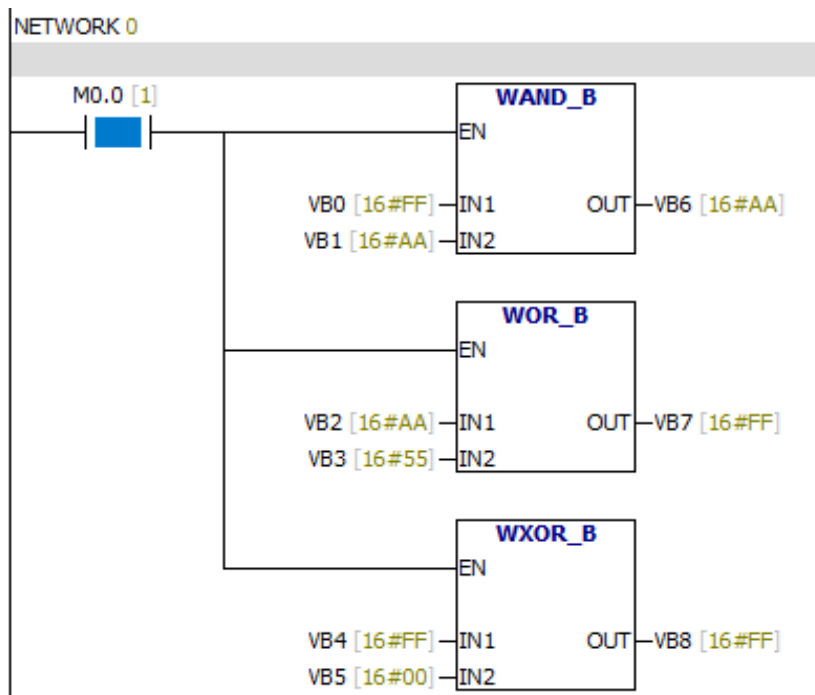
error condition:

0006 Indirect address

Special memory bit:

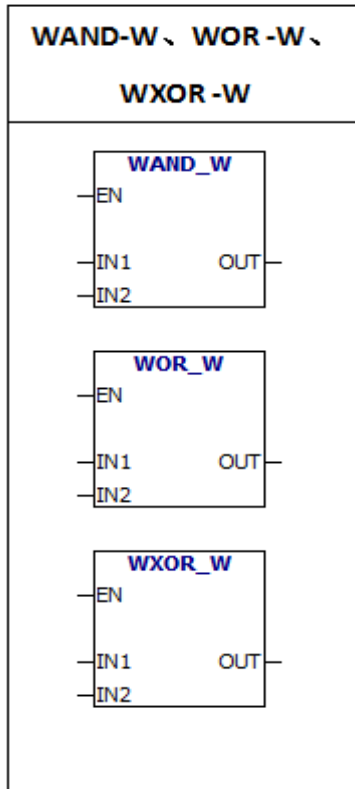
SM1.0 Zero result

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.

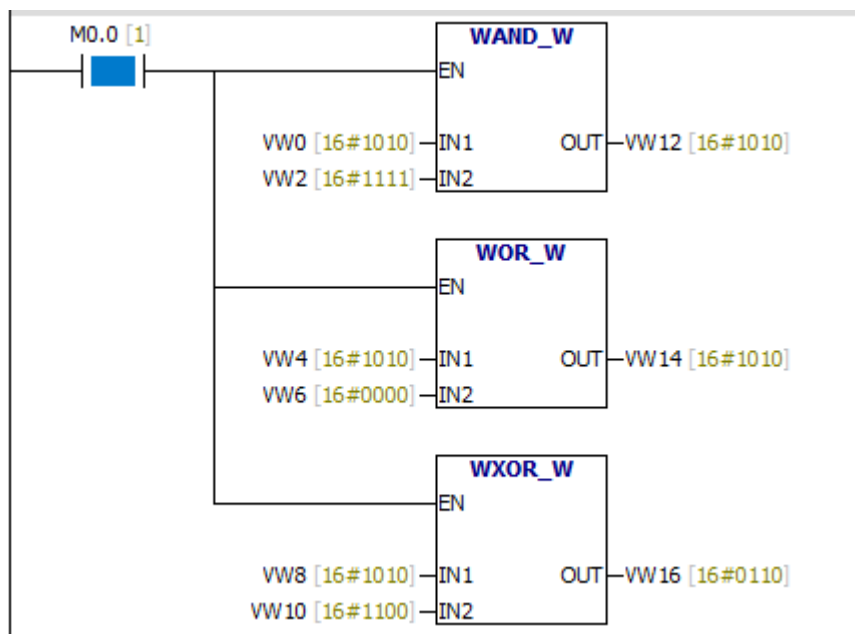
error condition:

0006 Indirect address

Special memory bit:

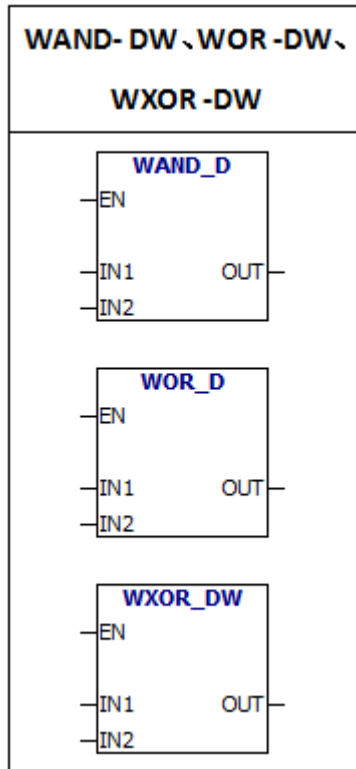
SM1.0 Zero result

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.

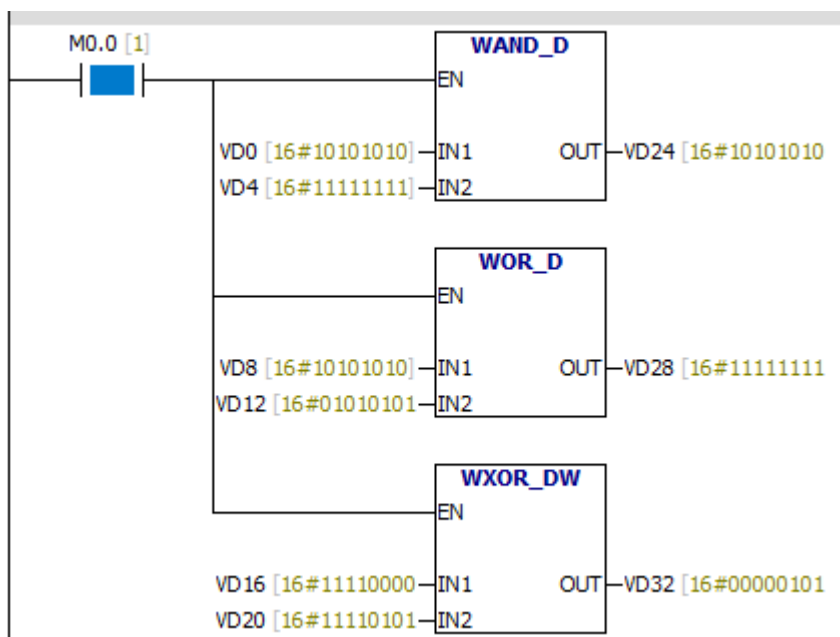
error condition:

0006 Indirect address

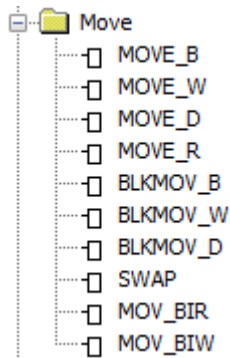
Special memory bit:

SM1.0 Zero result

Example:



6.11 Move



6.11.1 Byte move

Input/output Operand

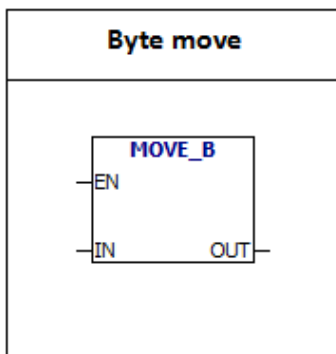
Data type

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

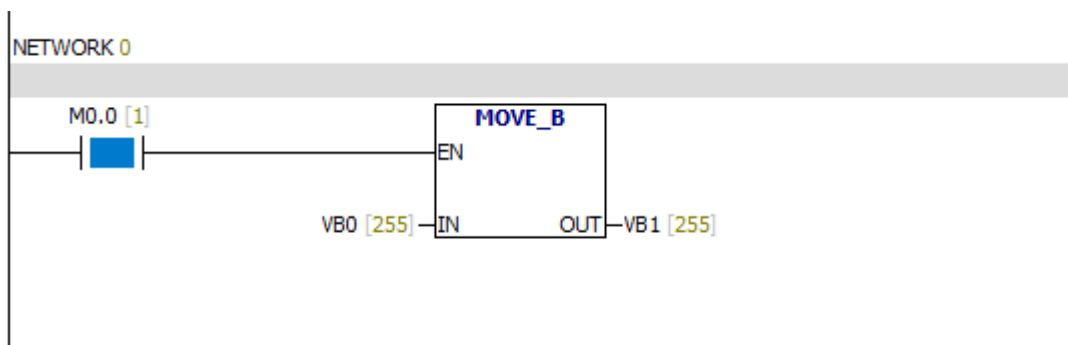


MOV -B: The instruction moves the input byte (IN) to the output byte (OUT), which does not change the original value.

error condition:

0006 Indirect address

Example:



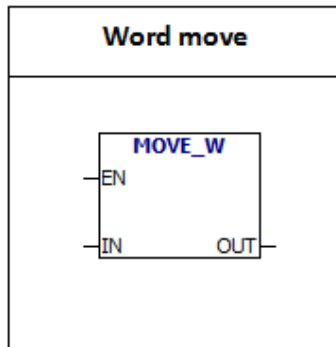
6.11.2 Word move

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

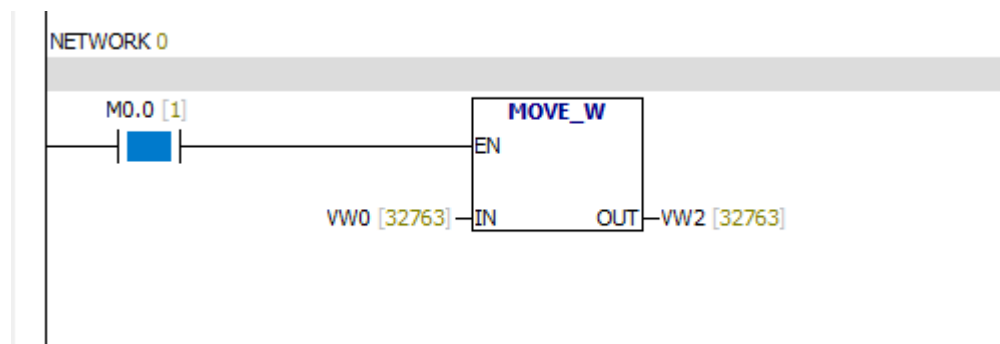


MOV -W: The instruction moves the input word (IN) to the output word (OUT), which does not change the original value.

error condition:

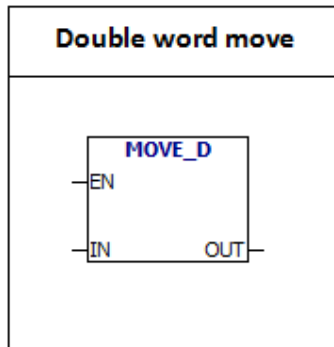
0006 Indirect address

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



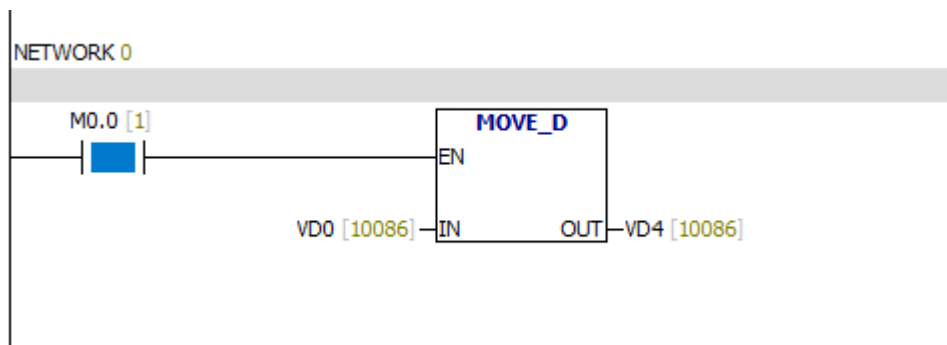
MOV -DW:The instruction moves the input double word (IN) to the output double word (OUT), which does not change the original value.

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

error condition:

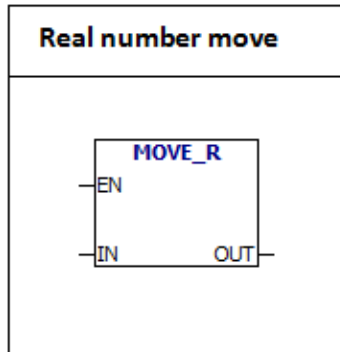
0006 Indirect address

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

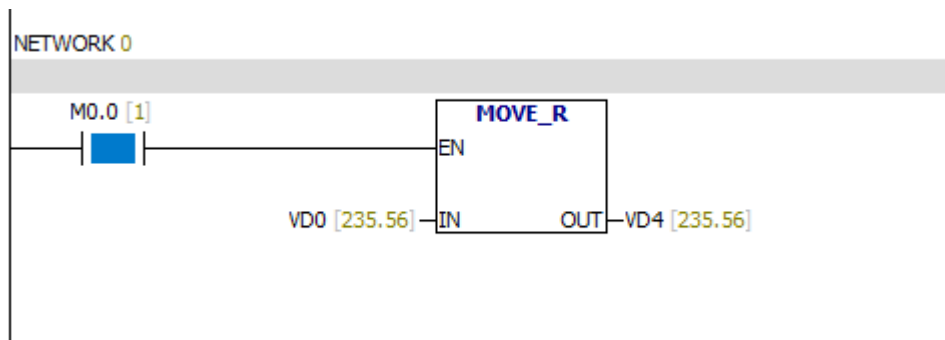


MOV -R: The instruction moves the input real number (IN) to the output real number (OUT), which does not change the original value.

error condition:

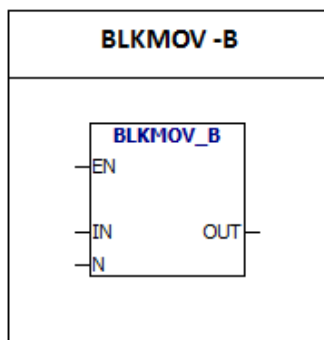
0006 Indirect address

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.

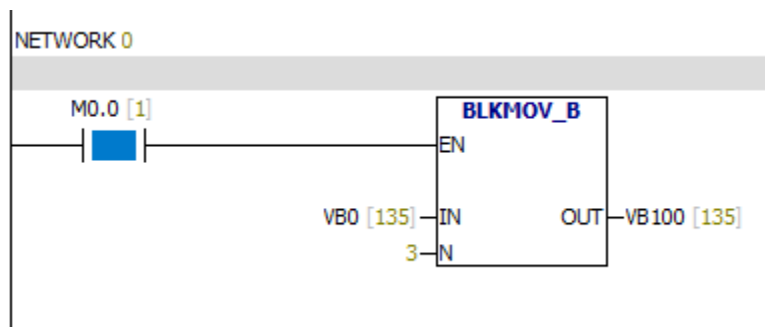
The range of N is from 1 to 255.

error conditions:

0006 Indirect address

0091 Operating number is out of range

Example:

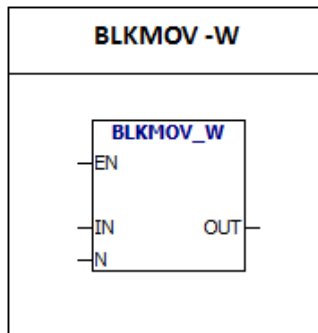


Status Chart

	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.

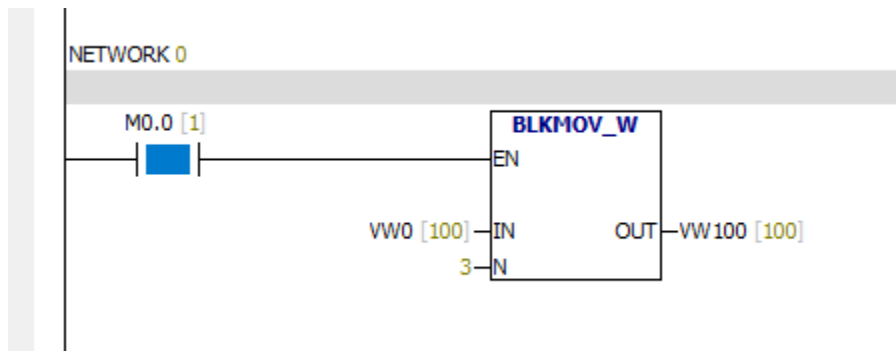
The range of N is from 1 to 255.

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

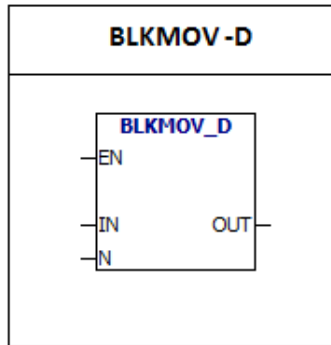
Data type

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.

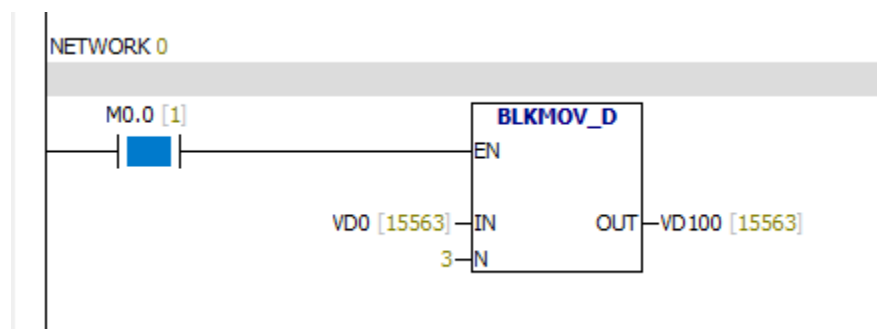
The range of N is from 1 to 255.

error conditions:

0006 Indirect address

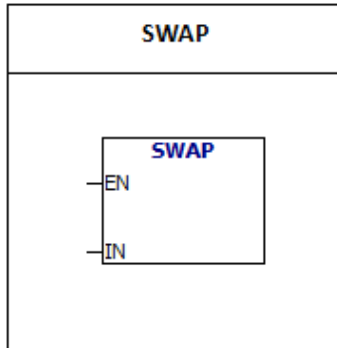
0091 Operating number is out of range

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

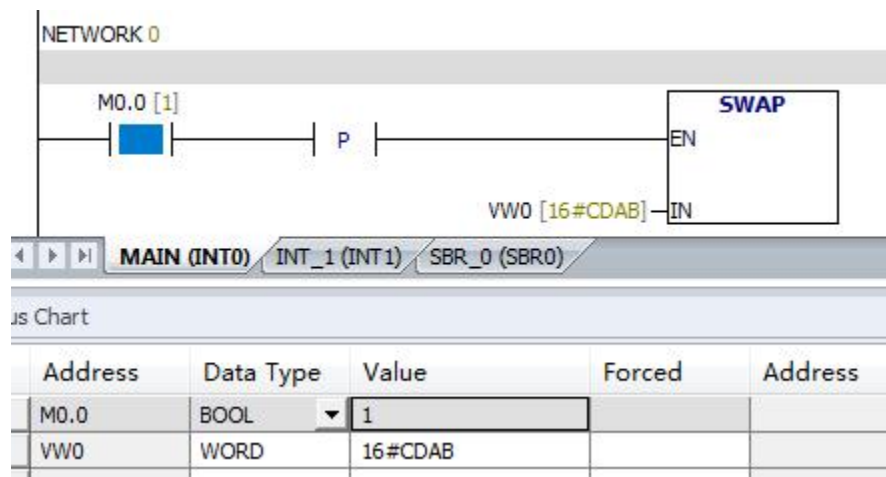
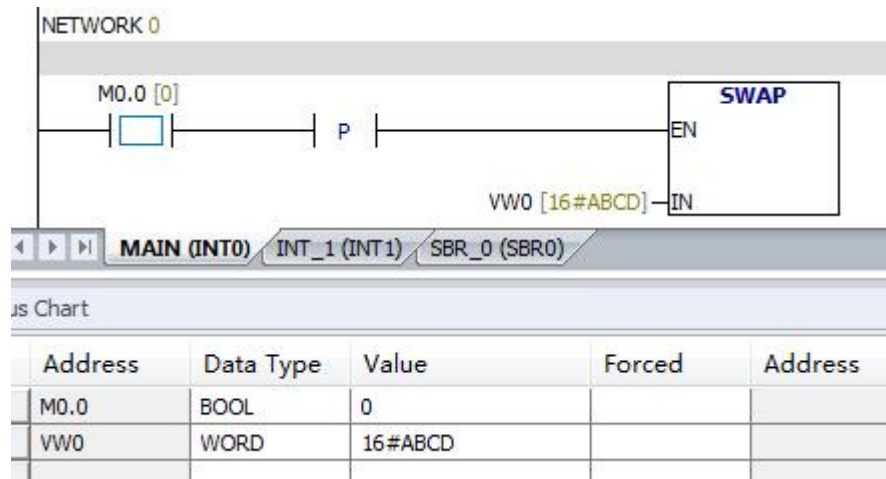


SWAP: The instruction interchanges high byte and low byte of the input word.

error conditions:

0006 Indirect address

Example:

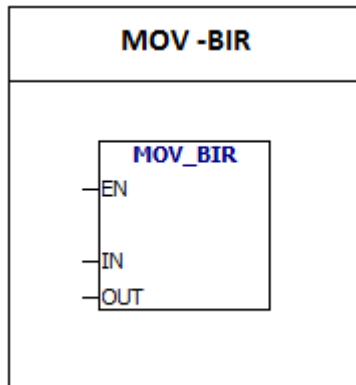


6.11.9 MOV -BIR

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:

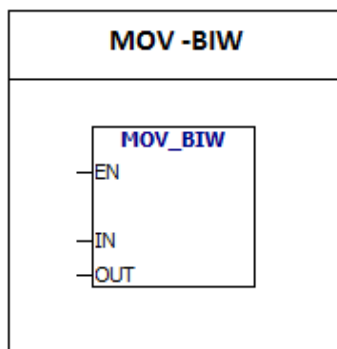
0006 Indirect address

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
-----	-------------------	------

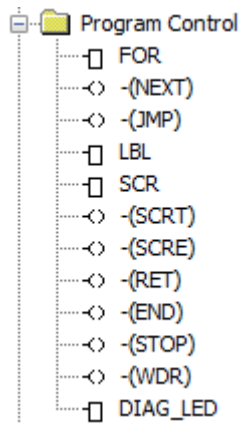


MOV -BIW: The instruction writes the input value(IN) to the actual input(OUT) and update the corresponding process image register.

error conditions:

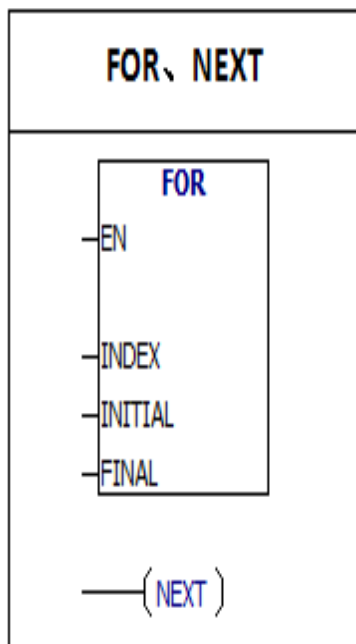
0006 Indirect address

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

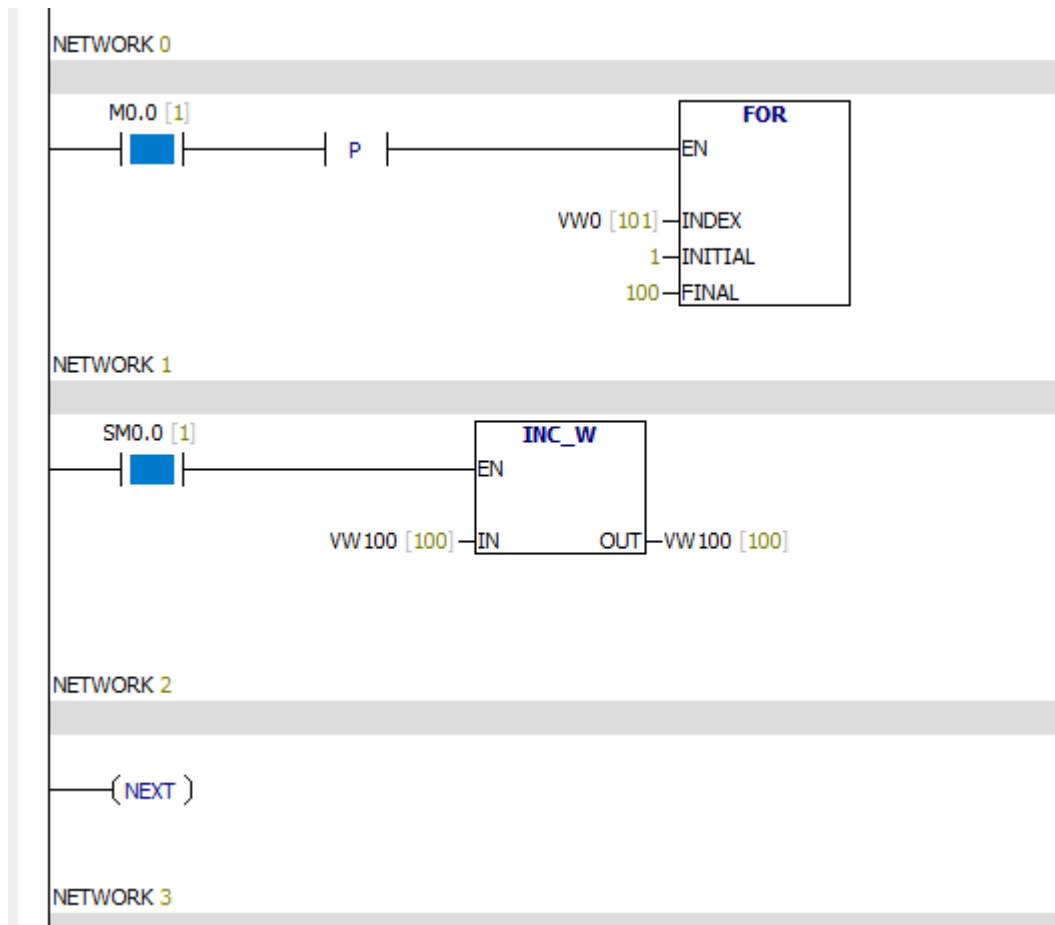


FOR instruction executes instructions between FOR and NEXT. You have to specify the current cycle count (INDX), start value (INIT), and end value (FINAL). NEXT (NEXT) instruction marks the end of the FOR loop, and the top value of the stack is set to 1. Use FOR/NEXT to set the number of loops. Each FOR 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.

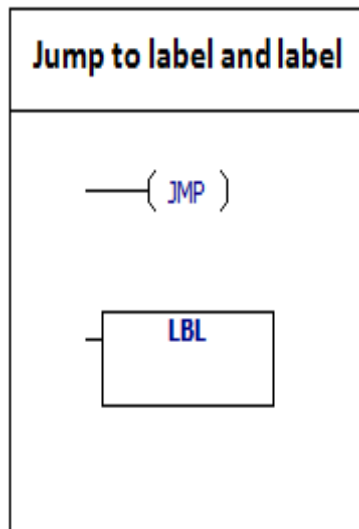
6.12.2 Jump to label and label

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.

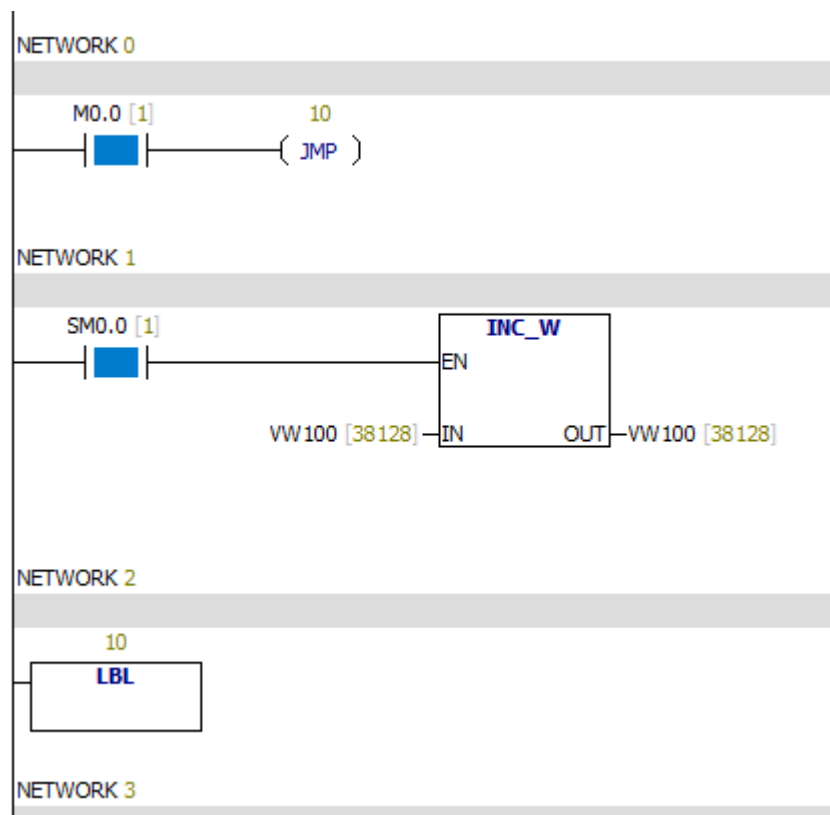
LBL instruction signs the location of n.

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.

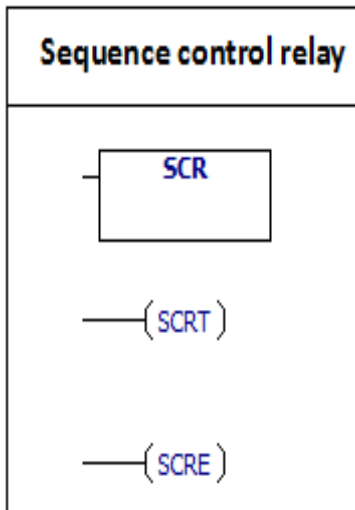
Example:



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



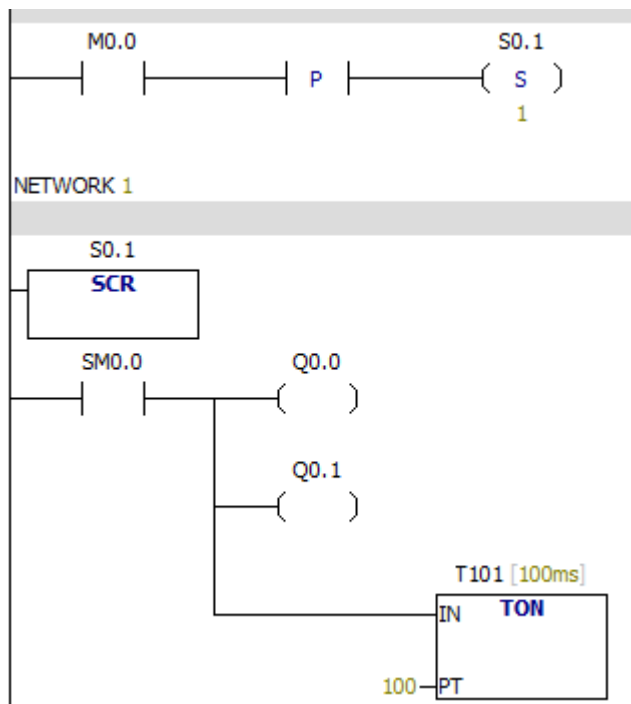
SCR instruction is good at dealing with repetitive operations.

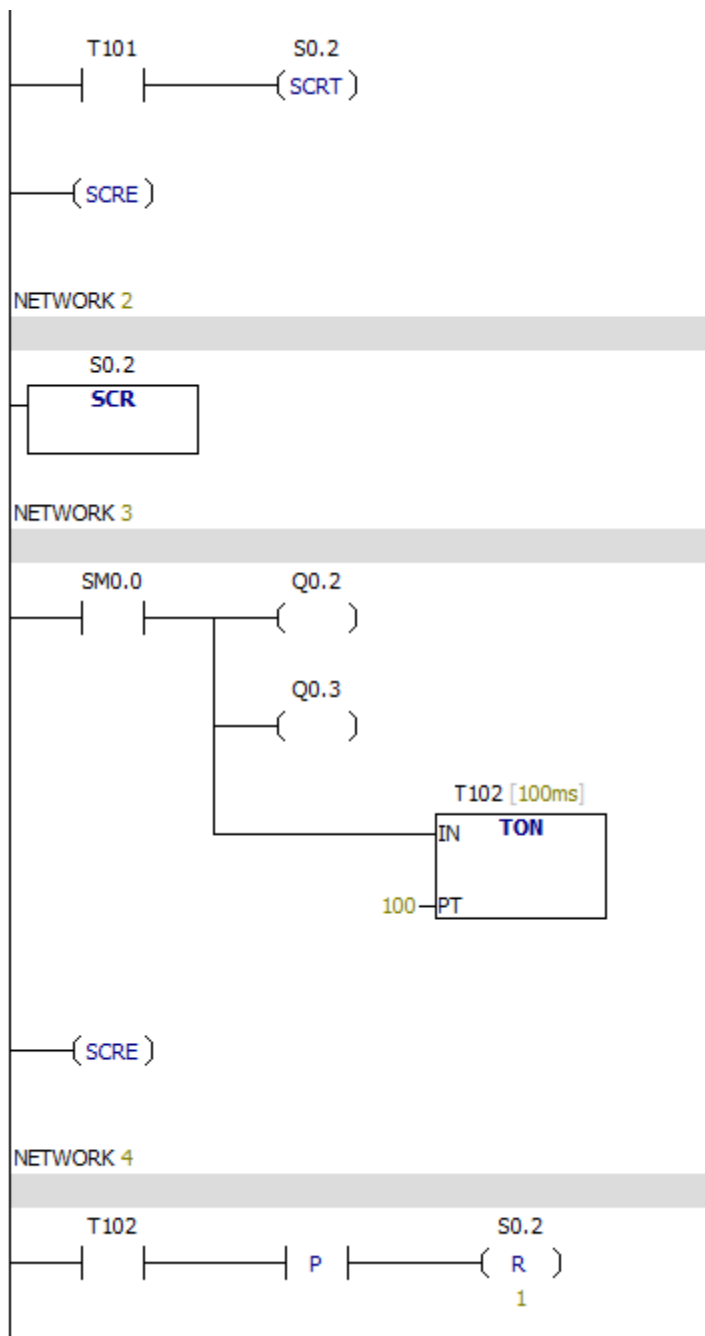
SCR: load the SCR section, you can use the SET instruction.

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

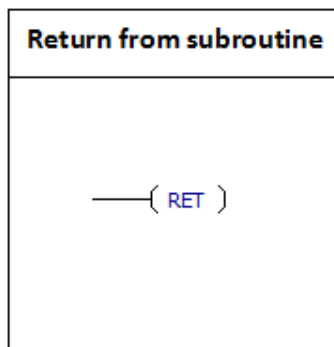
SCRE: The instruction signs the end of SCR segment.

Example:





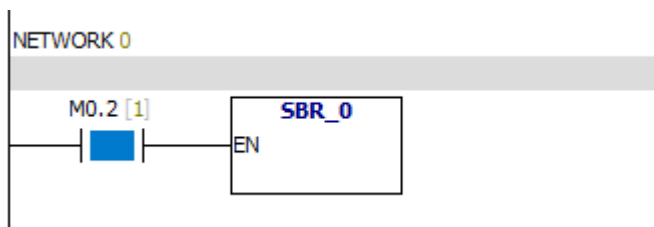
6.12.4 Return from subroutine



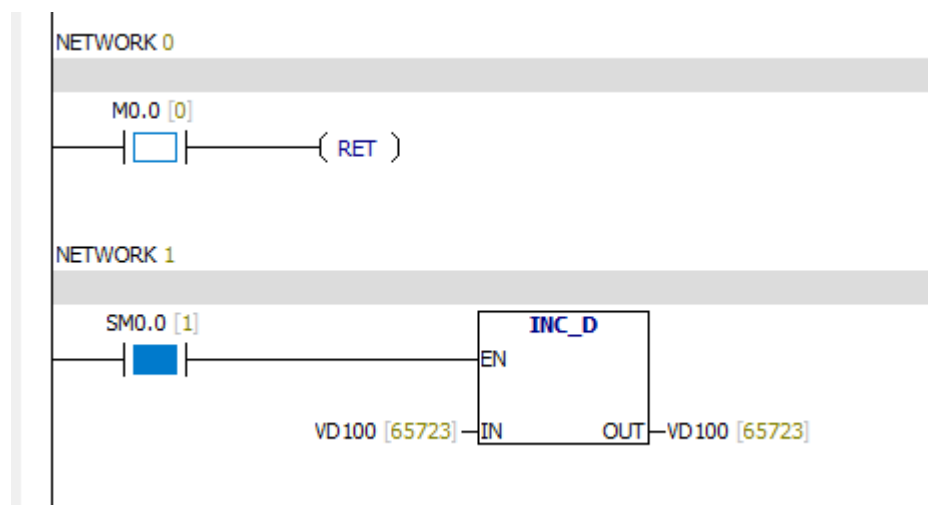
RET: Return from the subroutine to the main program.

Example:

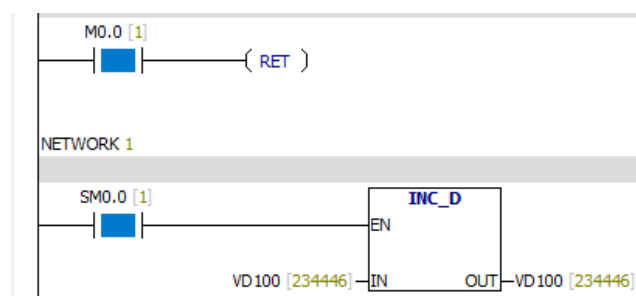
Main program:



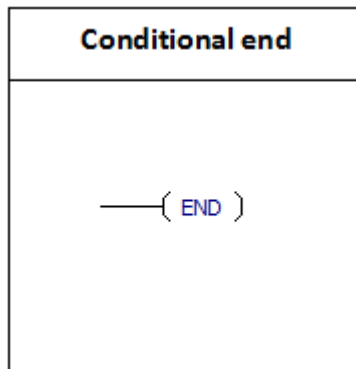
Subroutine:



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



6.12.5 Conditional end

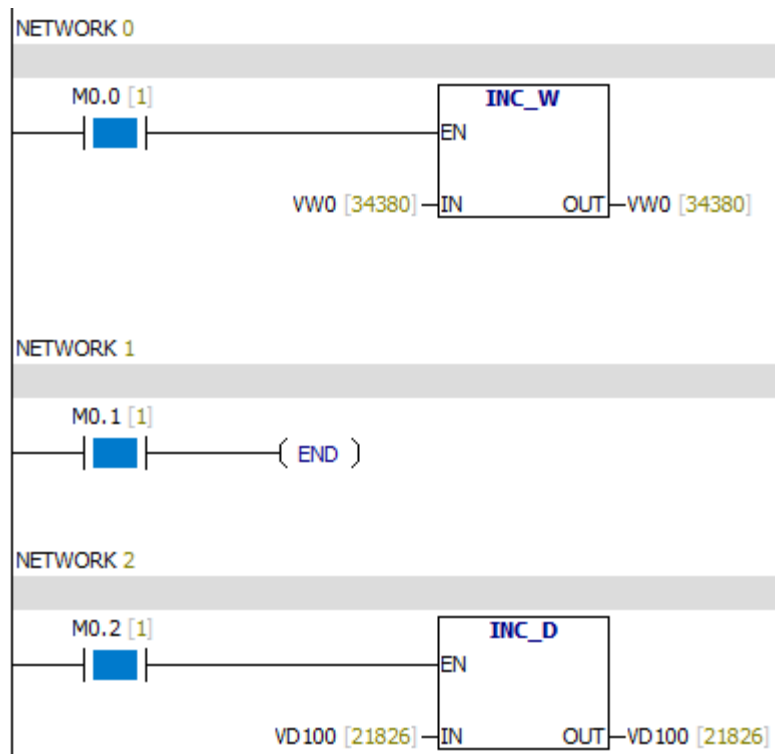


The END instruction terminates the user program .

Notes:

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

Example:

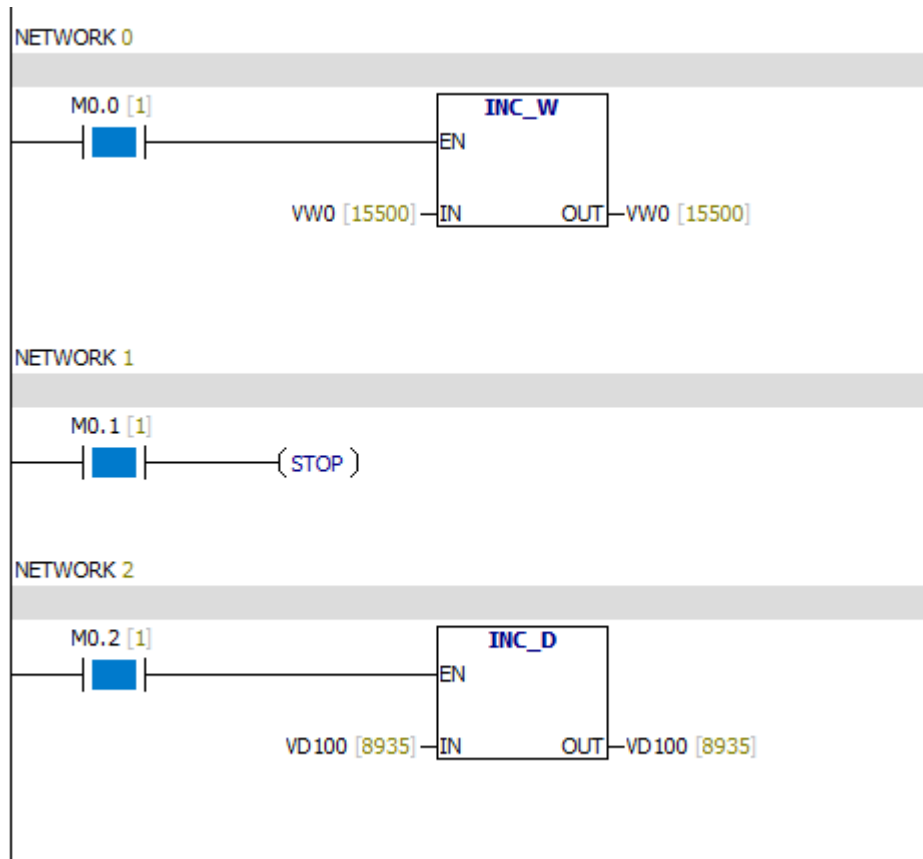


When the M0.1 bit is 1, the program will not be scanned.

6.12.6 STOP

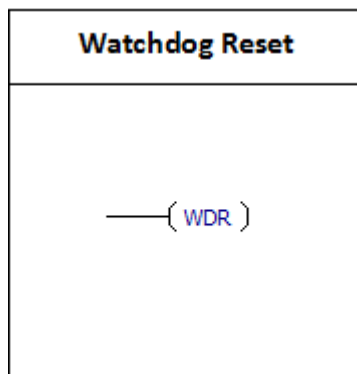
STOP instruction: STOP

Example:



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

6.12.7 Watchdog Reset



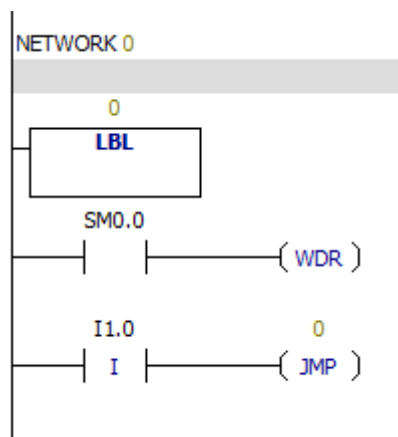
WDR clear watchdog time. When the scan cycle is greater than the watchdog time, the WDR makes the watchdog not issue a warning.

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

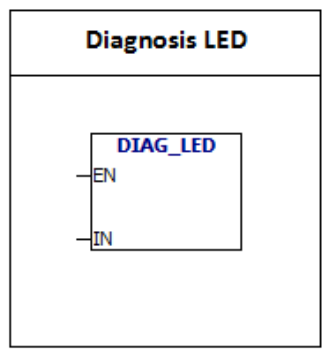
Attention :If you expect scan time will be more than 500 ms, you should use the WDR instruction to re trigger the watchdog timer.

Example:



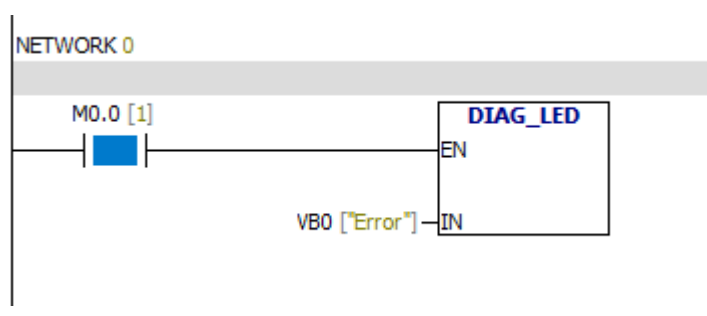
6.12.8 Diagnosis LED

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



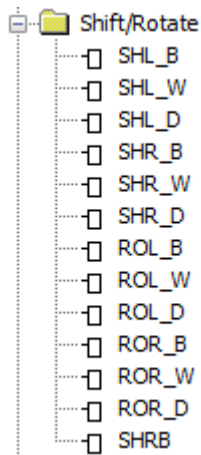
If the value of “EN” is 1, then the LCD will display the string from “IN”.

Example:



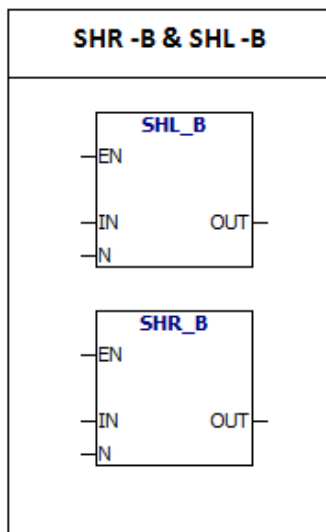
When the value of M0.0 is equal to 1, the LCD will display “Error”.

6.13 Shift cycle



6.13.1 SHR -B & SHL -B

Input/output	Operand	Data type
IN (LAD, FBD)	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:

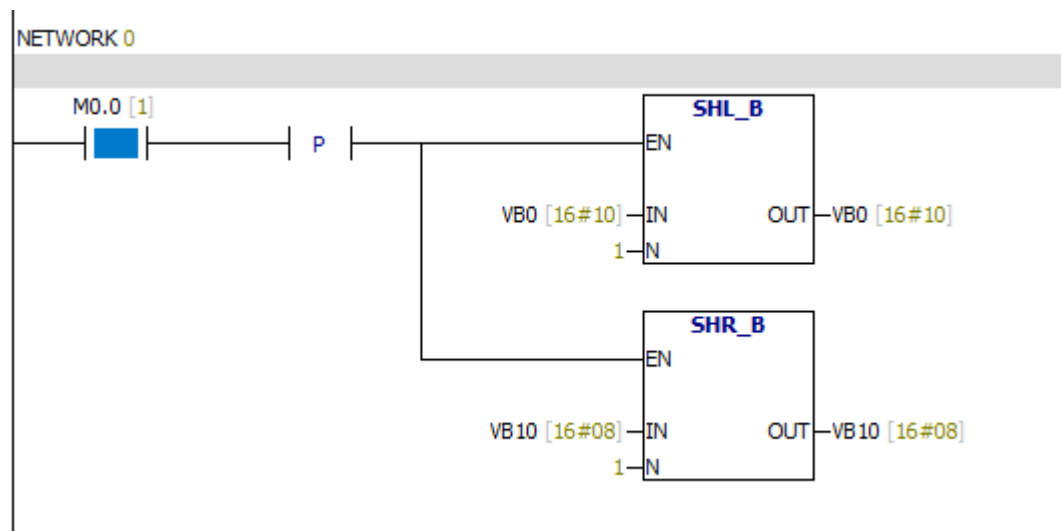
0006 Indirect address

Special memory bit:

SM1.0 Zero Result

SM1.1 Overflow

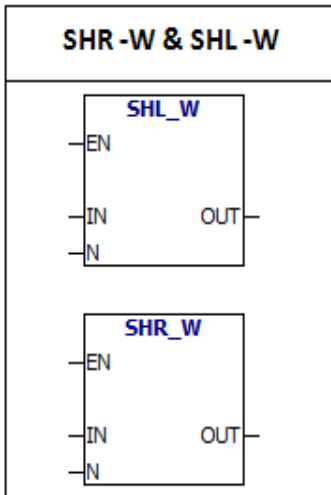
Example:



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 (LAD, FBD)	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:

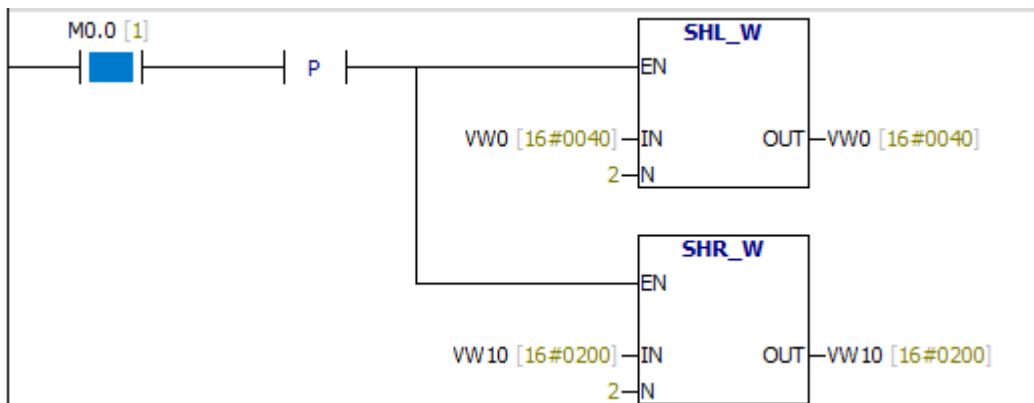
0006 Indirect address

Special memory bit:

SM1.0 Zero Result

SM1.1 Overflow

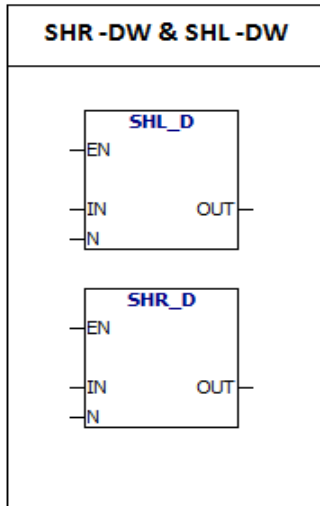
Example:



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 (LAD, FBD)	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:

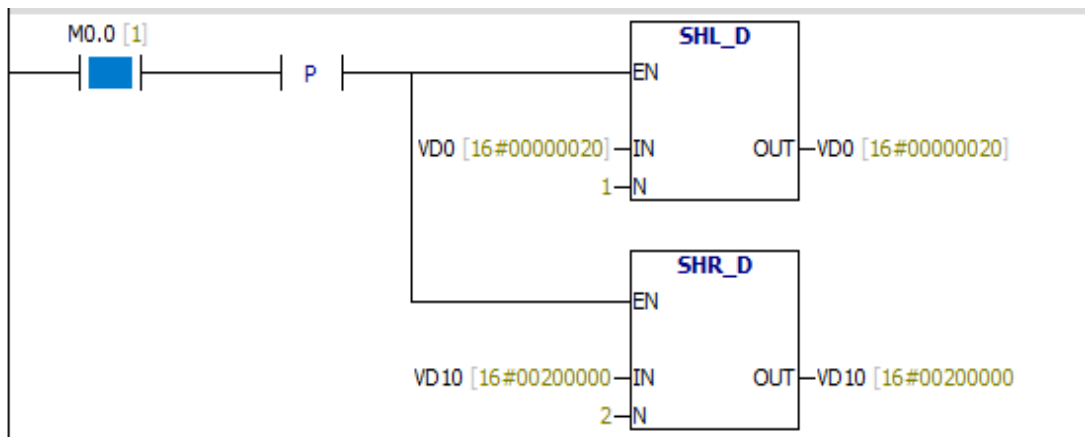
0006 Indirect address

Special memory bit:

SM1.0 Zero Result

SM1.1 Overflow

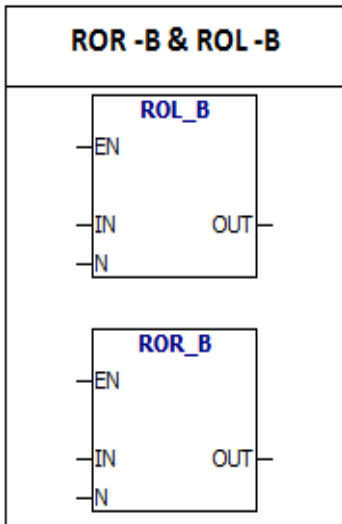
Example:



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 (LAD, FBD)	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:

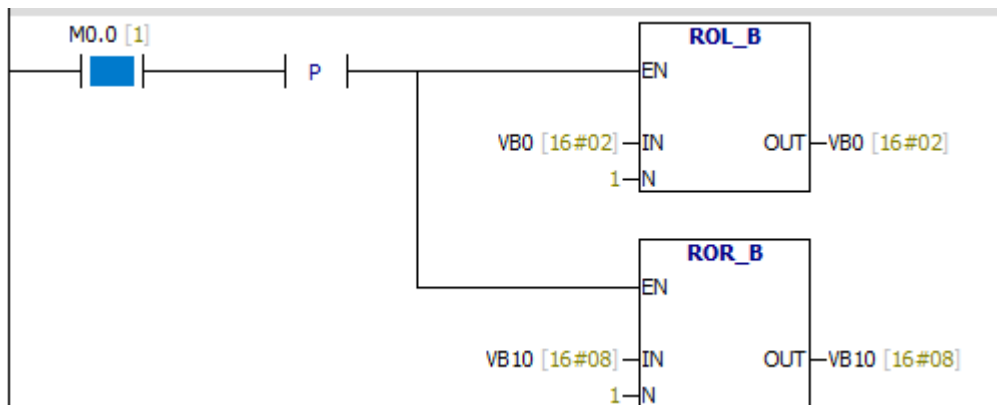
0006 Indirect address

Special memory bit:

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

SM1.1 Overflow bit

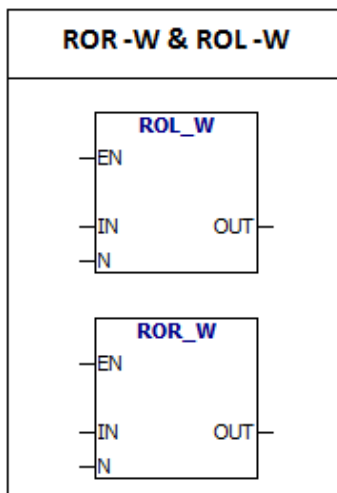
Example:



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 (LAD, FBD)	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:

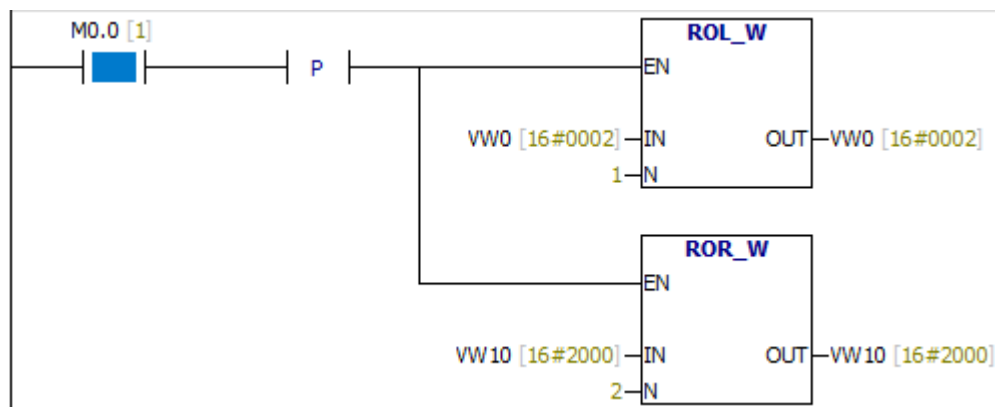
0006 Indirect address

Special memory bit:

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

SM1.1 Overflow bit

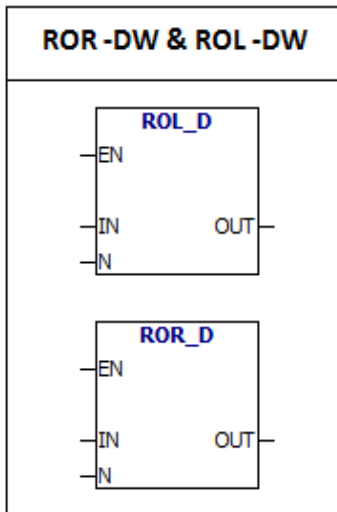
Example:



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 (LAD, FBD)	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:

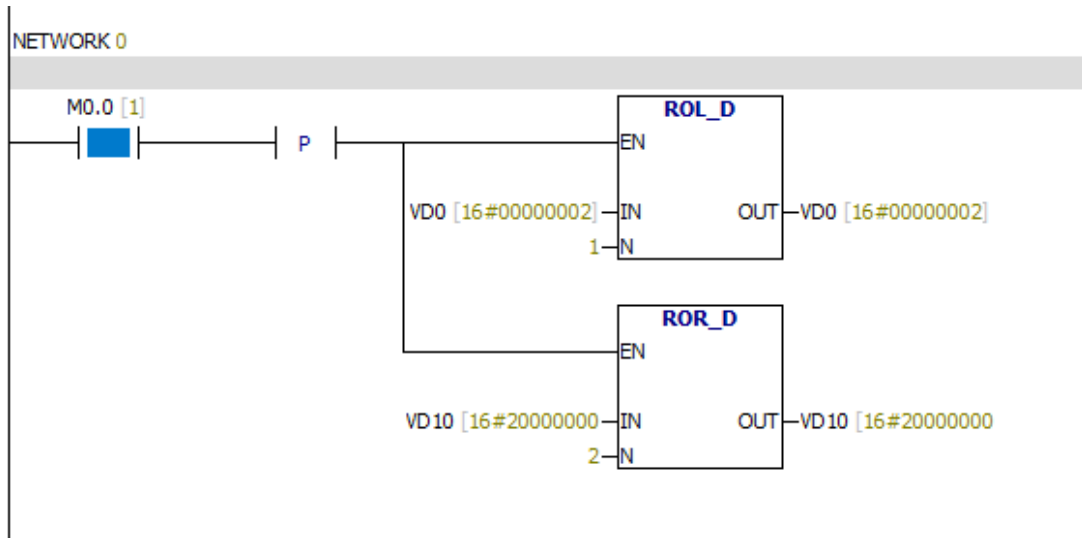
0006 Indirect address

Special memory bit:

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

SM1.1 Overflow bit

Example:

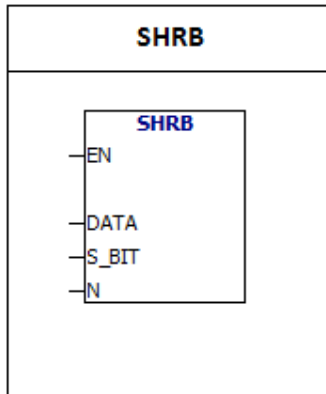


6.13.7 SHRB

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.

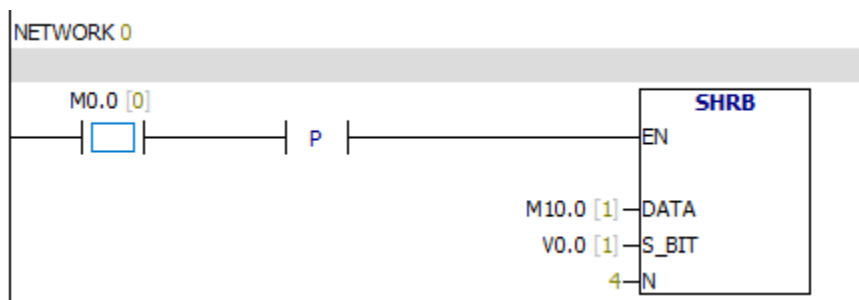
error conditions:

- 0006 Indirect address
- 0091 Operating number is out of range
- 0092 Count field error

Special memory bit:

SM1.1 Overflow bit

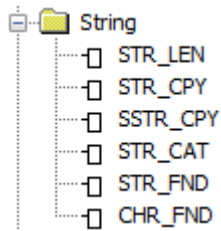
Example:



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.

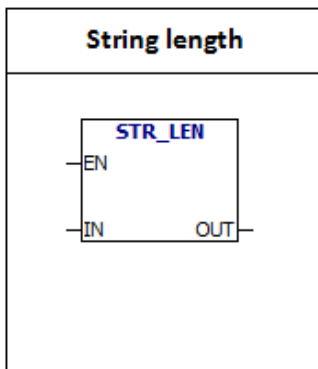
If N is negative, the shift direction is opposite.

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:

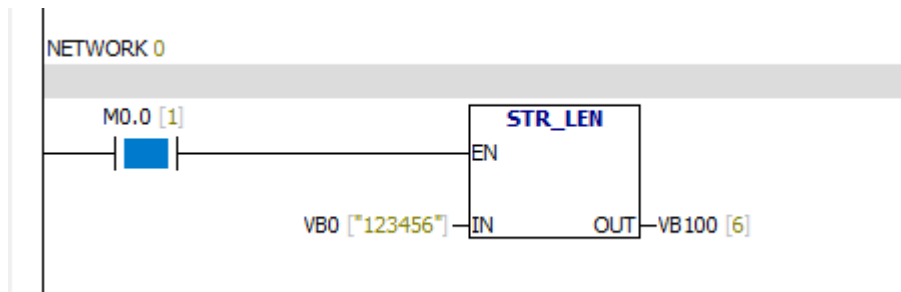
0006 Indirect address

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").

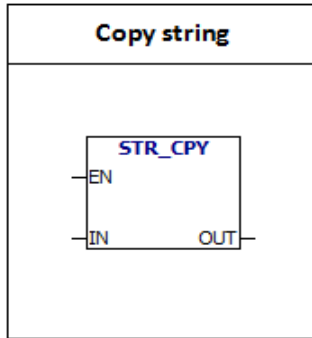
Example:



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:

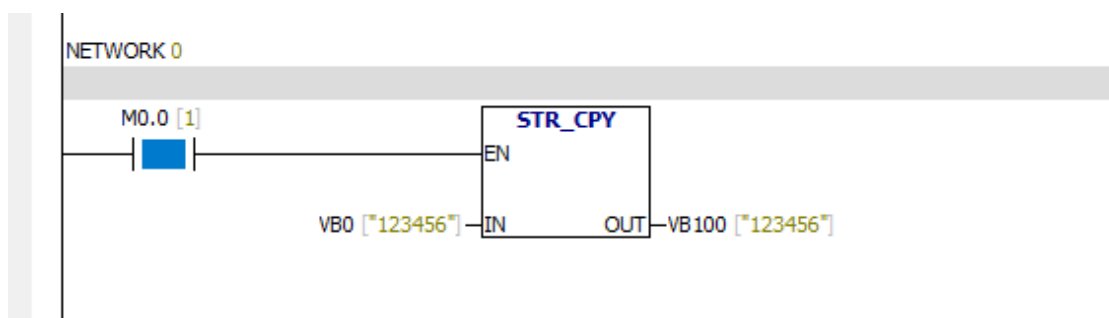
0006 Indirect address

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").

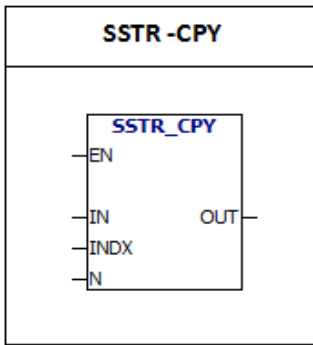
Example:



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.

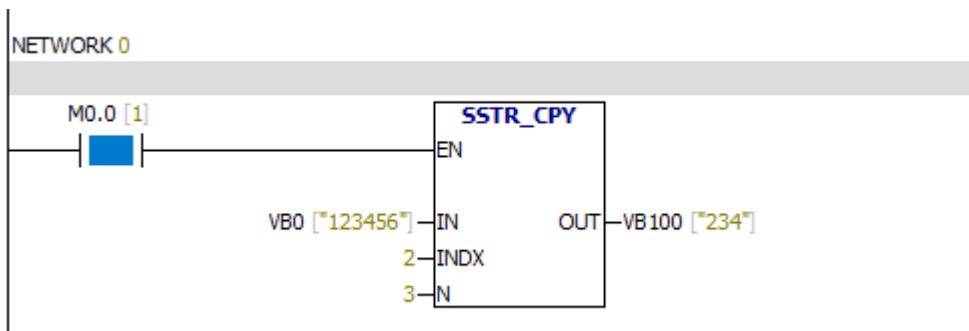
error conditions:

- 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").

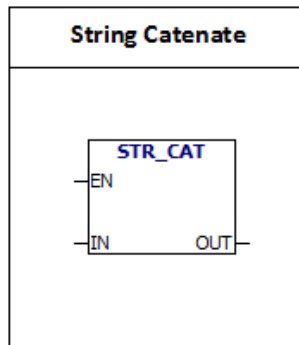
Example:



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:

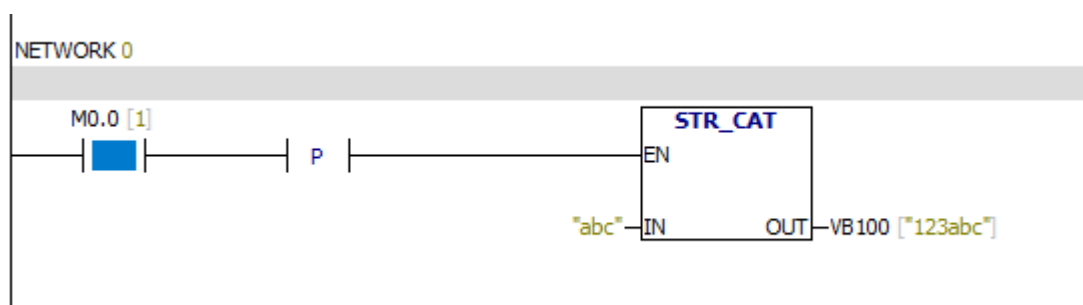
0006 Indirect address

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").

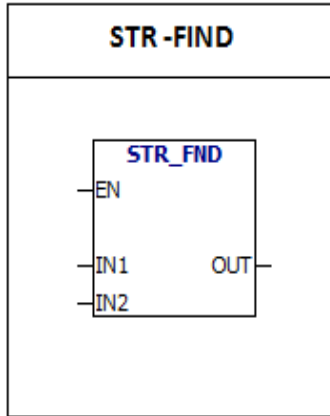
Example:



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.

error conditions:

0006 Indirect address

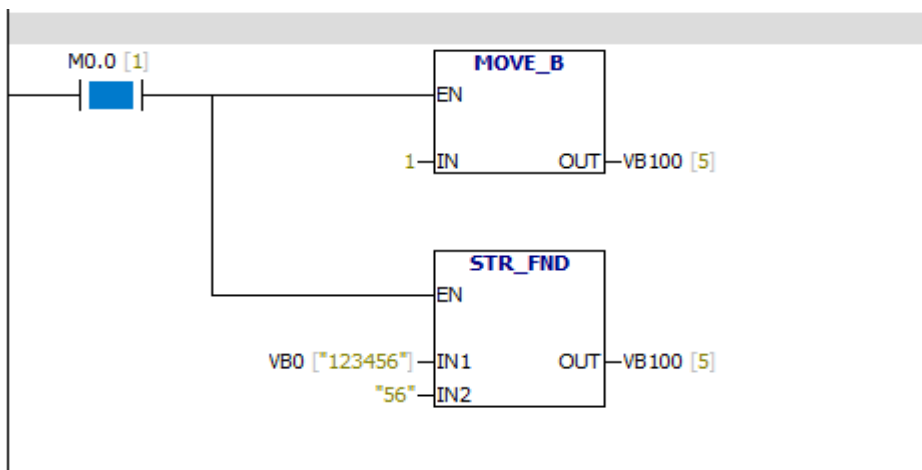
0091 Operand range

009B Illegal index

ASCII constant string data type format:

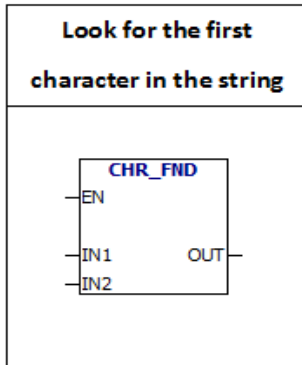
String is a series of characters, each character is stored as a byte. The first byte of a string defines the length of the string, that is the number of characters. If a constant string is entered directly into the program editor or data block, the string must start and end with double quotation marks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

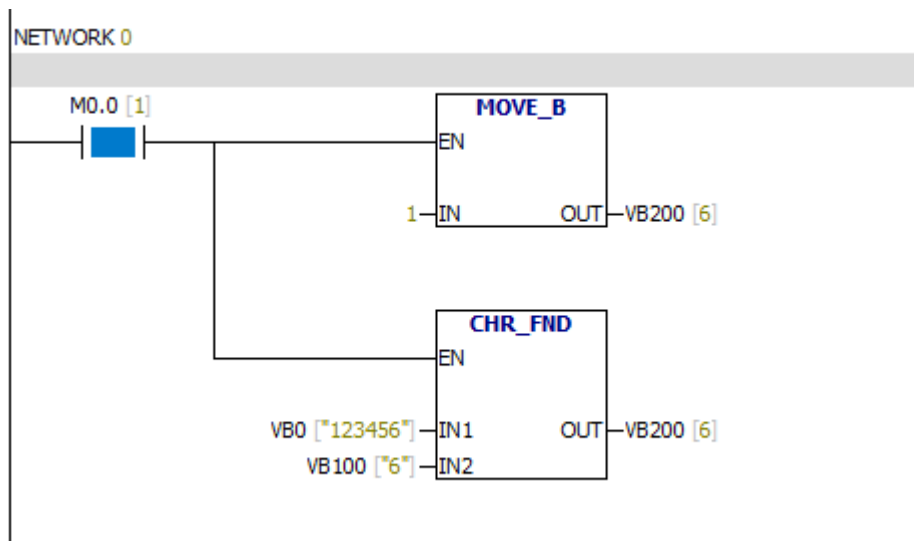
0091 Operand range

009B Illegal index

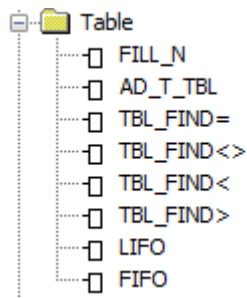
ASCII constant string data type format:

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

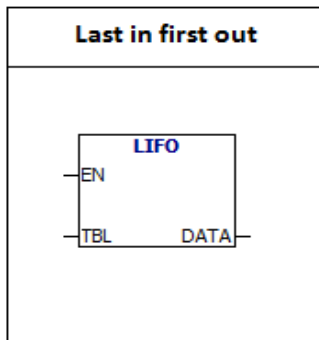
Data type

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:

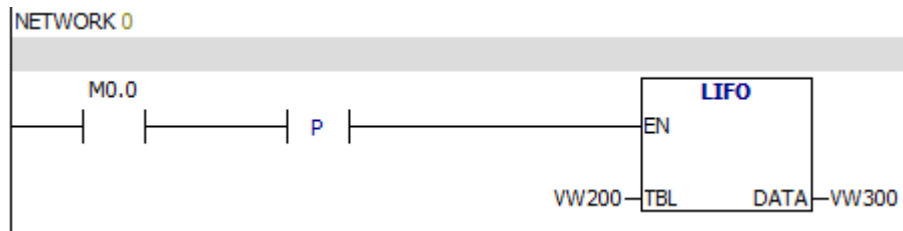
SM1.5 Empty list

Table Format:

VW200	The maximum number of entries
VW202	Entry count
W204	Data 0
VW206	Data 1
VW208	Data 2
.....

For example:

PLC program:

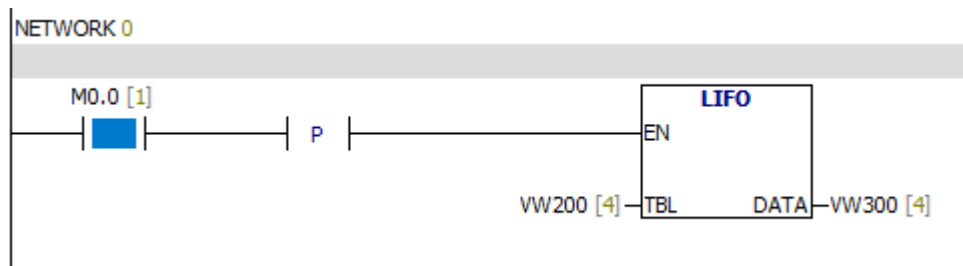


Data block:

	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

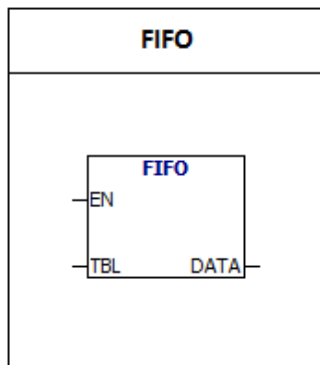
Data type

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:

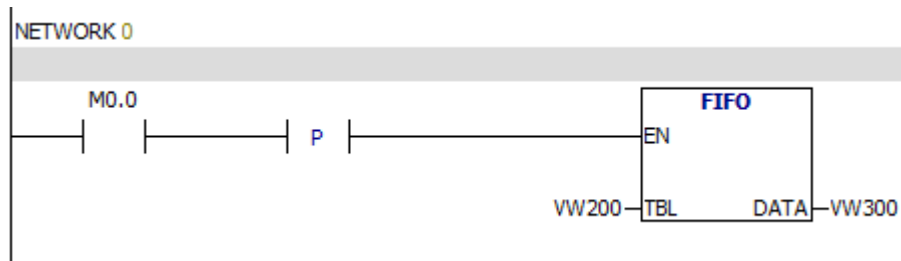
SM1.5 Empty list

Table Format:

VW200	The maximum number of entries
VW202	Entry count
W204	Data 0
VW206	Data 1
VW208	Data 2
.....

For example:

PLC program:

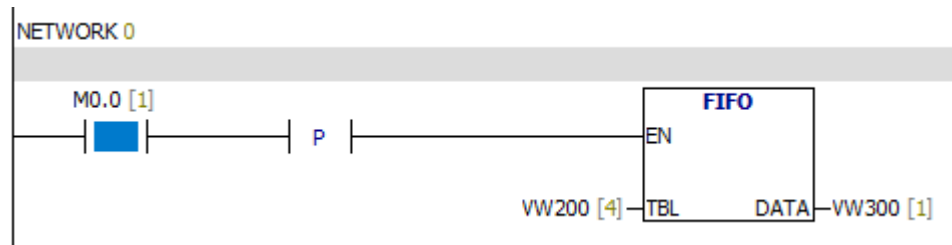


Data block:

	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 :

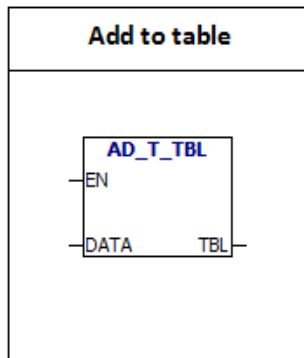
VW202=3

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.

error conditions:

0006 Indirect address

0091 Operand range

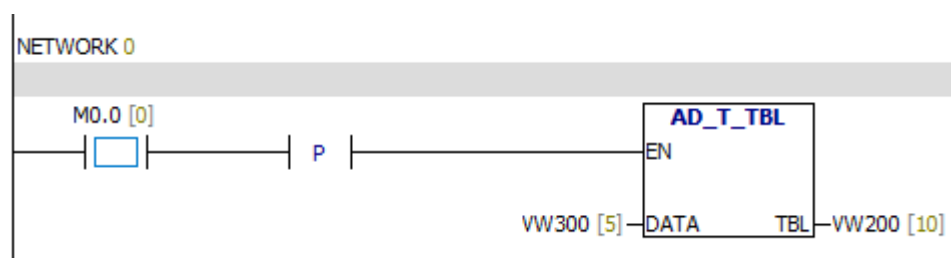
SM1.4 Table overflow

Special memory bit:

SM1.4 Table overflow

For example:

PLC program:



Data block:

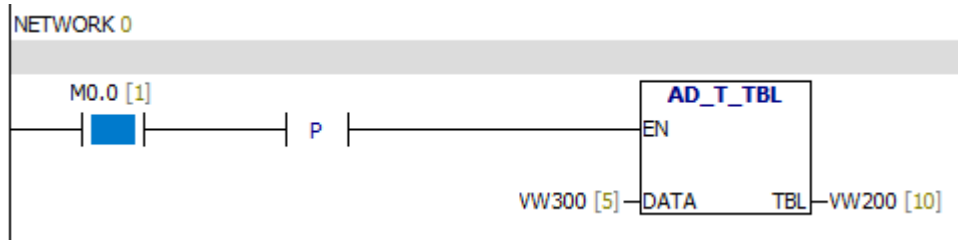
	Address	Data Type	Value
✓	VW200	INT	10
✓	VW202	INT	4
✓	VW204	INT	1
✓	VW206	INT	2
✓	VW208	INT	3
✓	VW210	INT	4

When the value of M0.0 is equal to 1:

The value of VW202 + 1

The Table will have a new entry

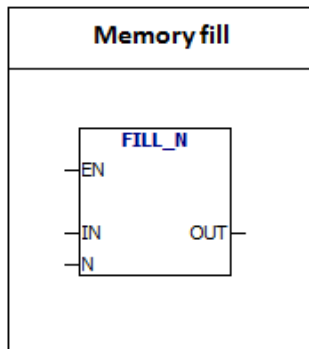
The value of the new entry is equal to the value of VW300.



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.

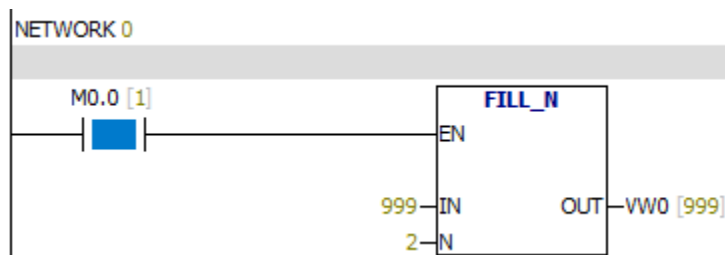
The range of N is from 1 to 255.

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

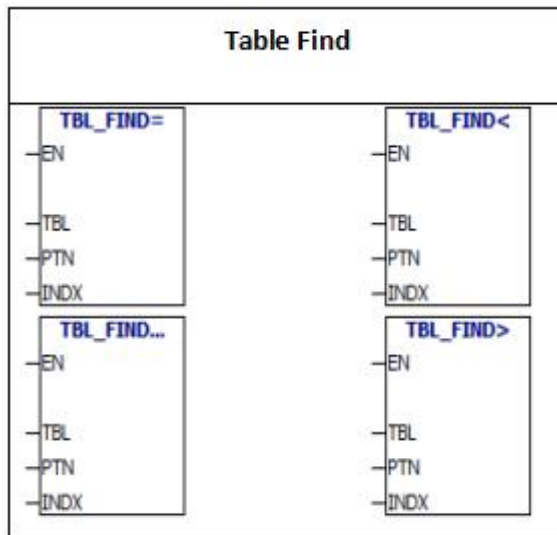
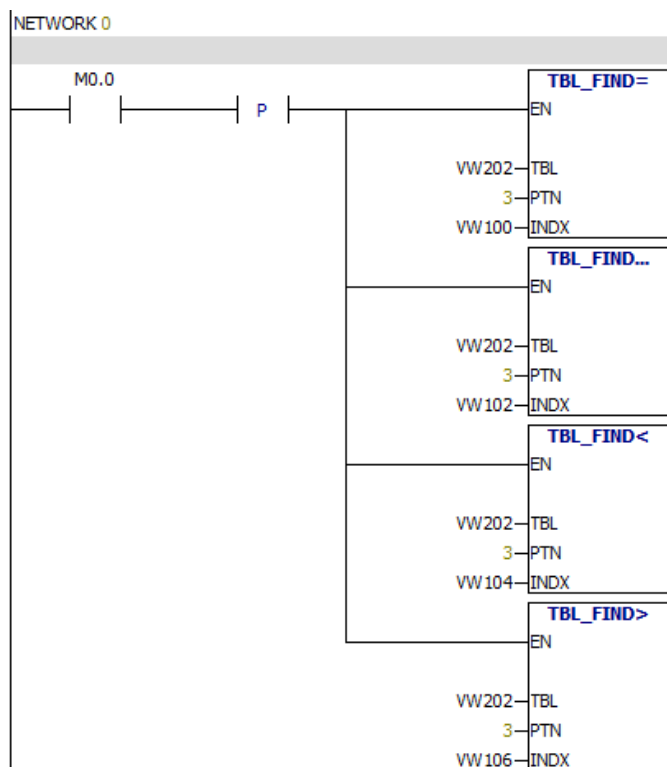


Table Find instruction: The instruction searches the same data as “PTN” in the table. “Table Find” starts from the entry specified by INDX. If a matching entry is found, the INDX points to the entry in the table. To find the next matching entry, you must add 1 to the INDX before using the “Table Find” instruction. If matching entry is not

found, the value of INDX is equal to the number of entries.

For example:

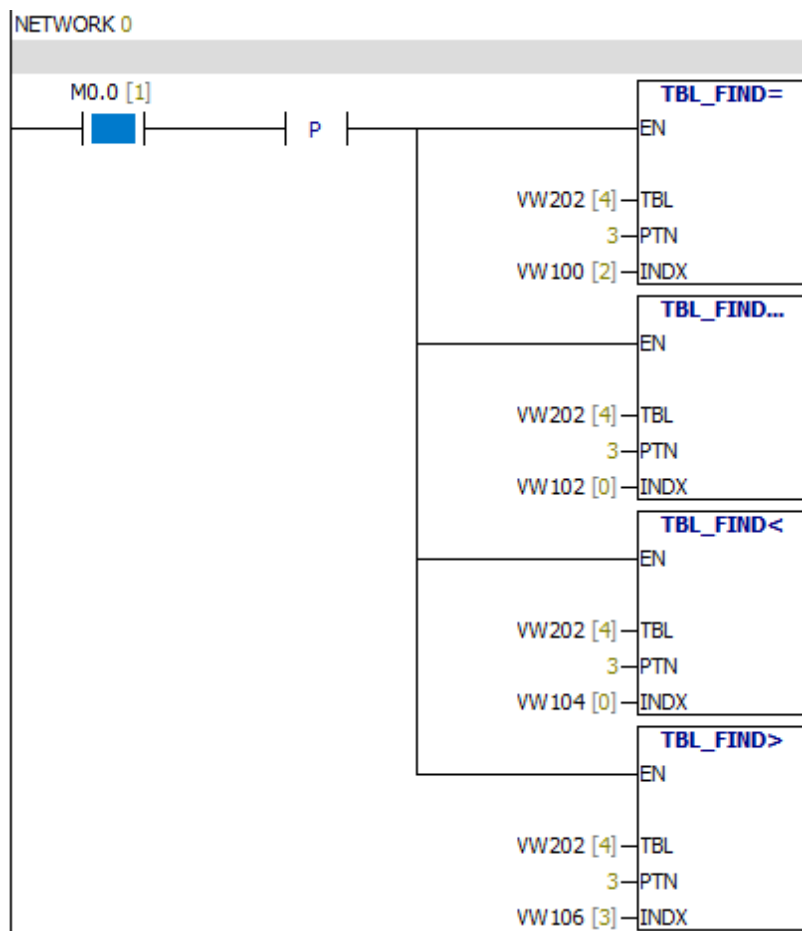
PLC program:



Data block:

	Adress	Data Type	Value
✓	VW200	INT	10
✓	VW202	INT	4
✓	VW204	INT	1
✓	VW206	INT	2
✓	VW208	INT	3
✓	VW210	INT	4

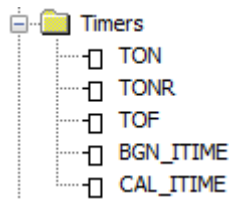
When the value of M0.0 is equal to 1:



The table format of the “Table-Find” begins with the entry count. It doesn’t have the “maximum number of entries”:

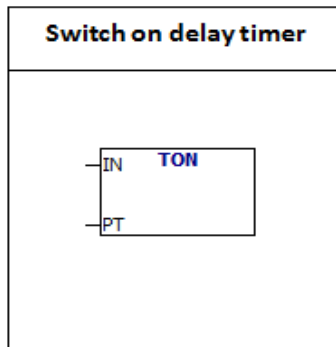
VW202	Entry Count
VW204	Data 0
VW206	Data 1
VW208	Data 2
VW210	Data 3

6.16 Timer



6.16.1 Switch on delay timer

Input/output	Operand	Data type
Txxx	constant(T0 -T255)	word
IN (LAD)	Enable bit	Boolean
IN (FBD)	I, Q, M, SM, T, C, V, S, L, 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 preset 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:

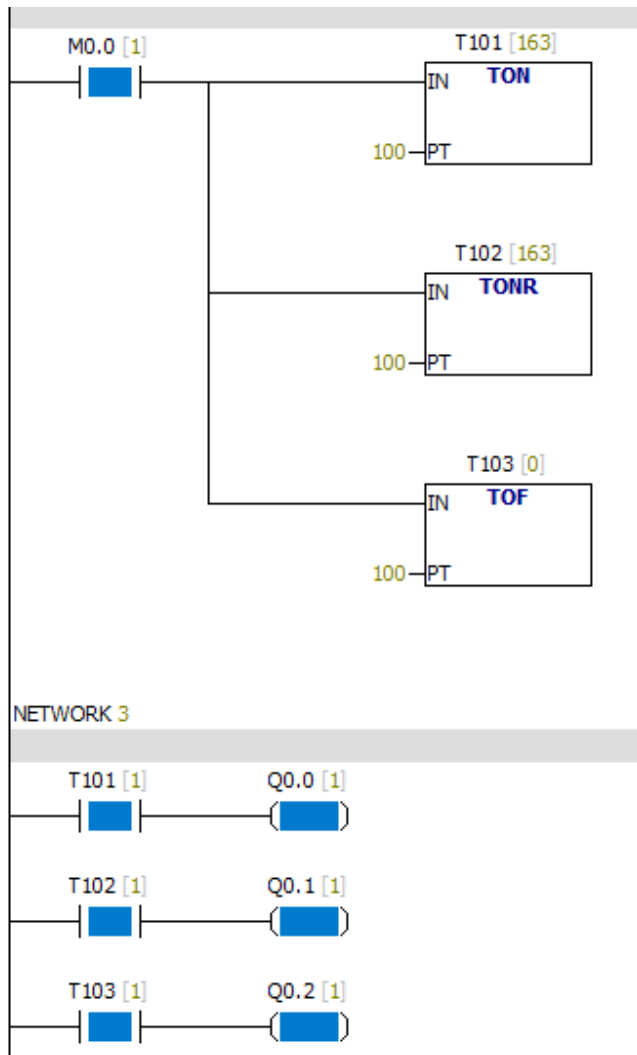
Timer number	Time base (ms)	Time range (s)
T0	1	65.535
T1~T4	10	655.35
T5~T31	100	6553.5
T32	1	65.535
T33~T36	10	655.35
T37~T63	100	6553.5
T64	1	65.535
T65~T68	10	655.35
T69~T95	100	6553.5
T96	1	65.535
T97~T100	10	655.35

T101-T127	100	6553.5
-----------	-----	--------

Attention:

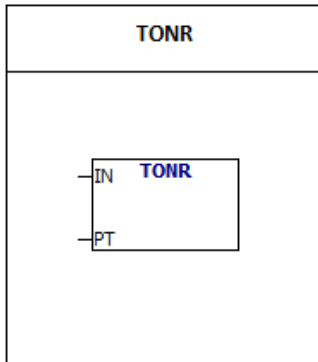
- 1.The value of each timer TXXX is different.
- 2.The resolution of the timer depends on the time base.For example, the error range of the 10 millisecond timer is 10 milliseconds.

Example:



6.16.2 TONR

Input/output Operand		Data type
Txxx	Constant(T0—T255)	word
IN (LAD)	Enable bit	Boolean
IN (FBD)	I, Q, M, SM, T, C, V, S, L, 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 preset 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 preset value, timer current value is retained. Otherwise, the current value of the timer is cleared.

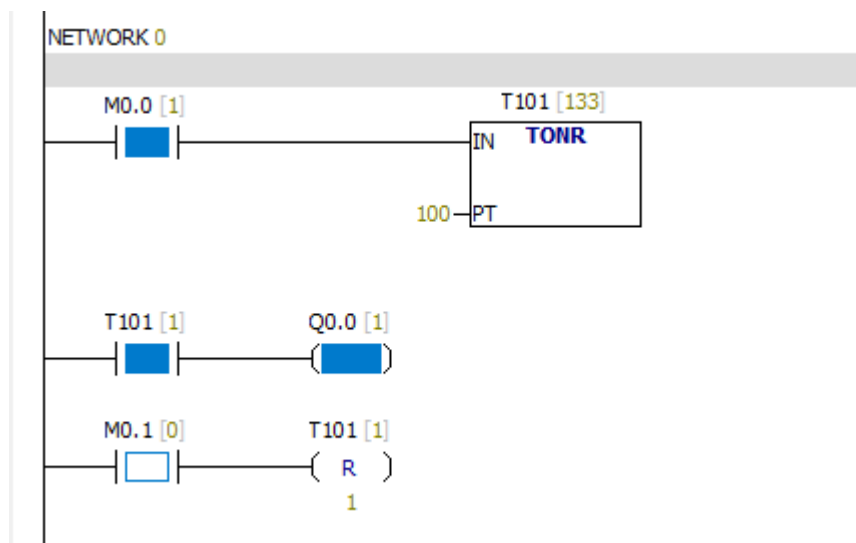
Notes:

You can use TONR to accumulate multiple time intervals.

You can use the "recovery" (R) instruction to recover any timer.

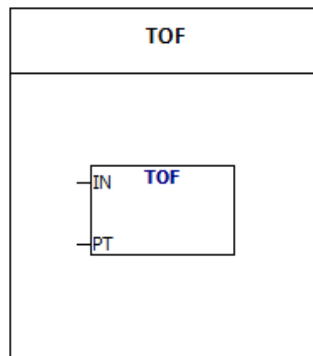
You can only use the "recovery" instruction to recovery the TONR timer.

Example:



6.16.3 Disconnect delay timer

Input/output Operand		Data type
Txxx	constant(T0—T255)	word
IN (LAD)	Enable bit	Boolean
IN (FBD)	I, Q, M, SM, T, C, V, S, L, Enable bit	Boolean
PT	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, constant, *VD, *LD, *AC	Integer



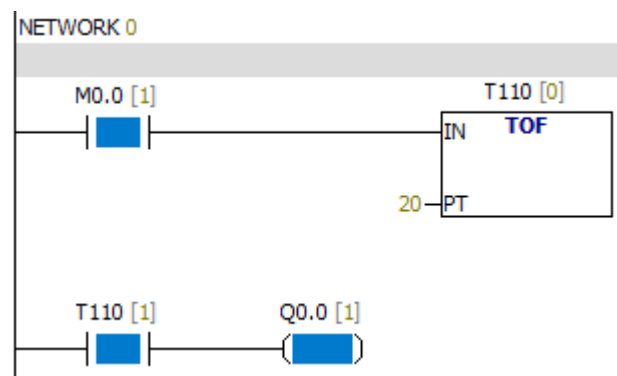
TOF: When the input is closed, the output will be closed for a period of time. When the value of IN is 1, the bit of the timer is 1 immediately and timer current value is set to 0. When the value of IN is 0, the timer starts time. When the current value is equal to the preset value, the bit of the timer is 0.

Notes:

The value of each timer TXXX is different.

You can use the "recovery" (R) instruction to recover TOF timer.

Example:

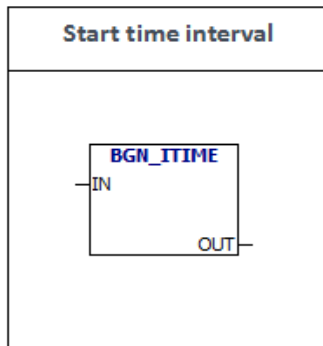


6.16.4 Start time interval

Input/output Operand

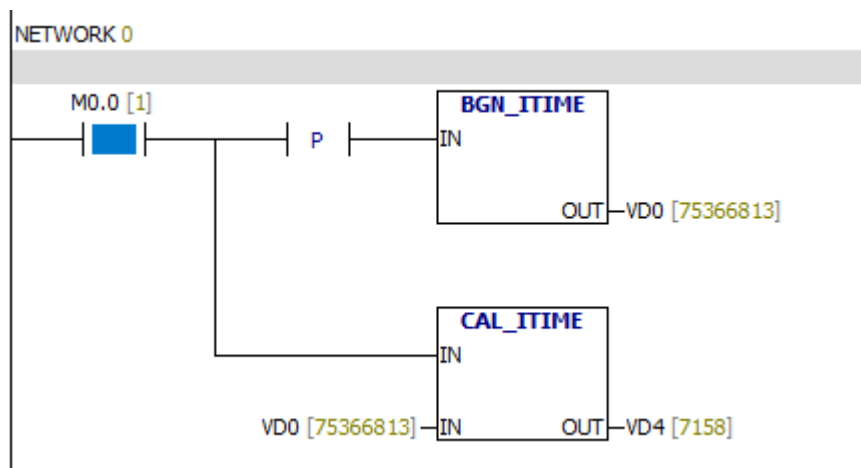
Data type

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.

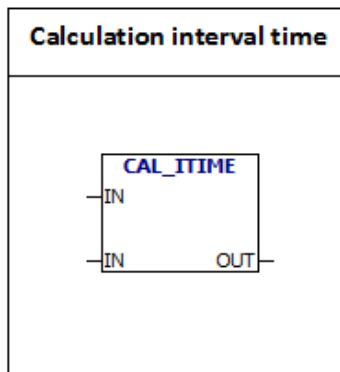
Example:



The value of VD4 is the conduction time of MO.0

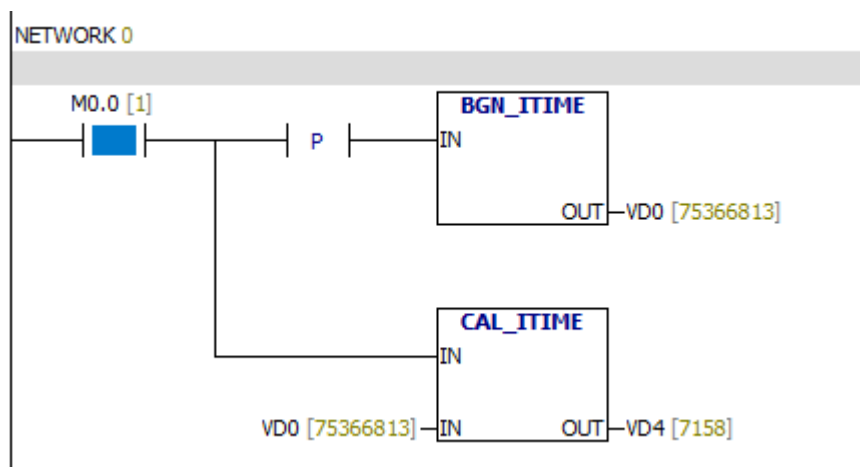
6.16.5 Calculation interval time

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



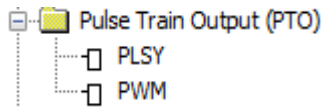
Calculates the time difference between the current time and the time provided by the IN, and stores the time difference in the OUT.

Example:



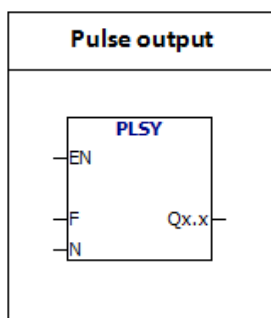
The value of VD4 is the conduction time of MO.0

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

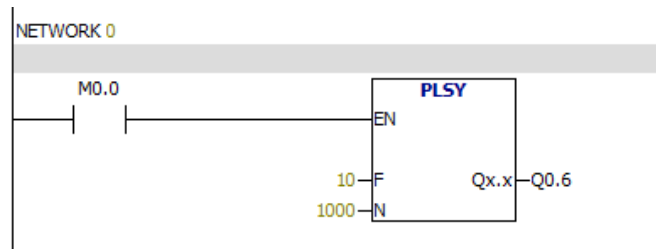


PLSY:When the value of the enable bit is 1,Instruction issues N pulses.The pulse frequency is F.

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:



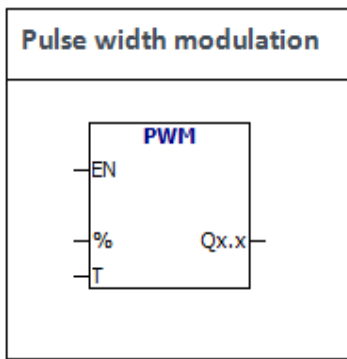
Attention:

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

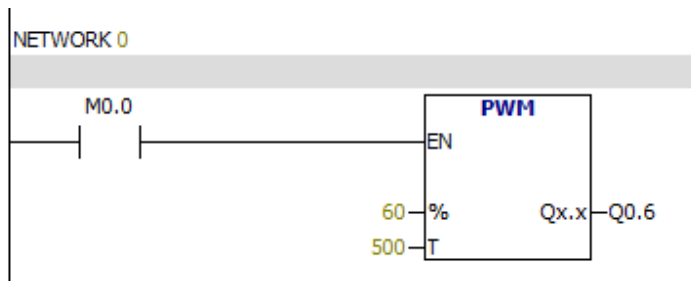


Pulse width modulation (PWM) instruction initializes the PWM hardware and sends high speed pulses.
 The input value of "%" =conduction time/period.
 The input value of "T" is the period of the pulse

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:



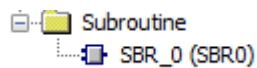
The pulse period is 500 ms, the conduction time is 300 ms.

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 not 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.

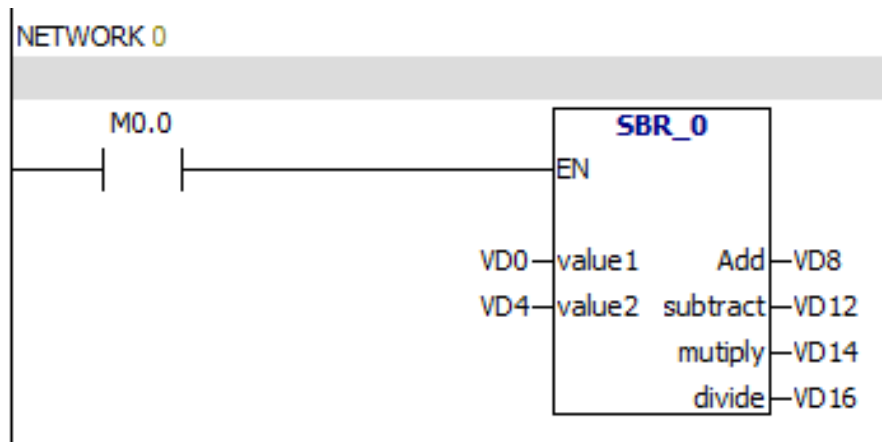
In LAD, the subroutine generates a block instruction. You can call the block instruction to call the subroutine.

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 operation

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	

The diagram shows a normally closed contact labeled SM0.0. This contact is connected to the EN (enable) input of four parallel subroutines: ADD_D, SUB_D, MUL_D, and DIV_D. Each subroutine has two data inputs: LD0 connected to IN1 and LD4 connected to IN2. The outputs of these subroutines are LD8, LD12, LD16, and LD20 respectively.

Navigation: 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:

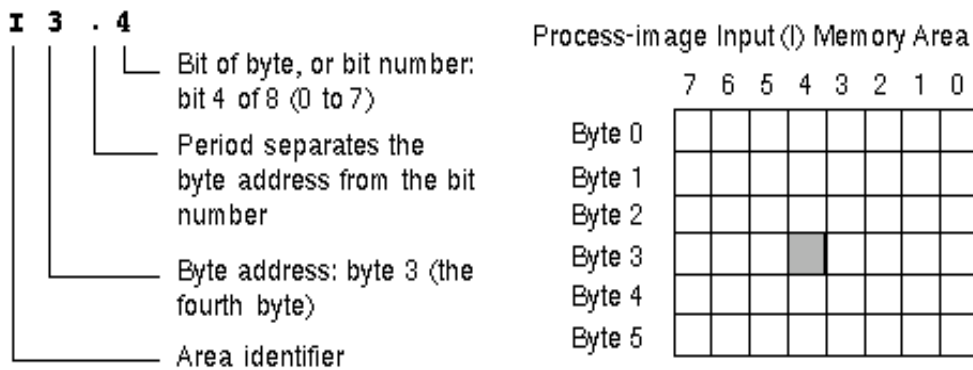
1. Direct addressing
2. Symbol addressing
3. indirect addressing

Direct addressing

PLC can directly specify the memory area, size, and location;

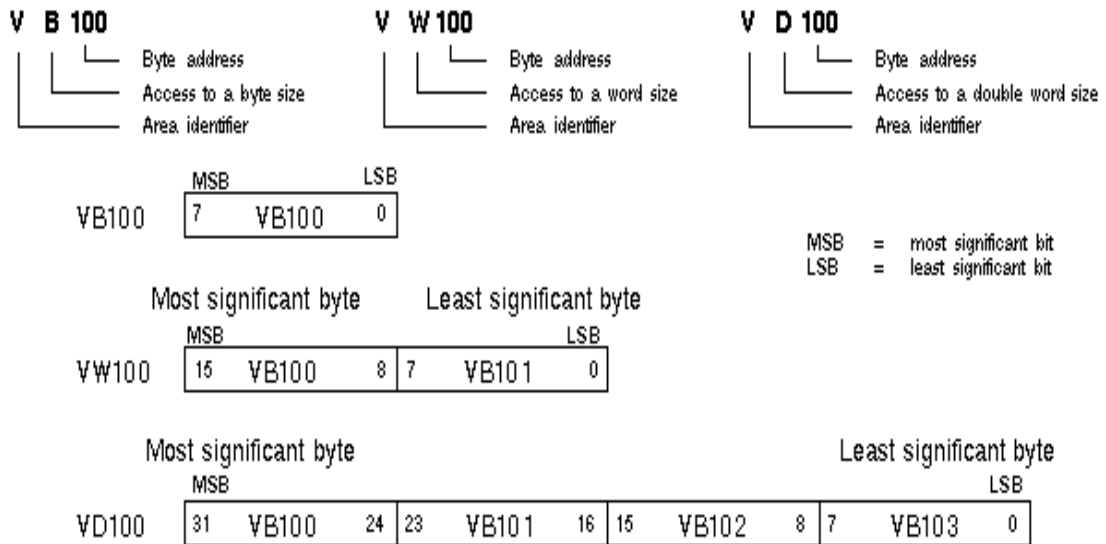
In order to read/write a bit in the memory area, you need to specify the address. The address includes memory area identifier, byte address, a period and number.

Example:



Specifying byte、 word and double word addresses are similar to specifying bit address.

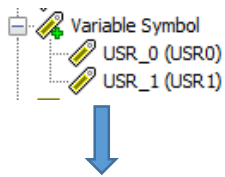
Example:



Symbolic addressing

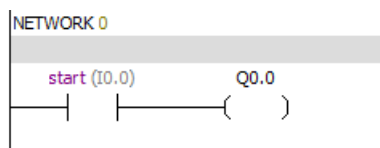
Symbol addressing consists of letters, numbers and characters.

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



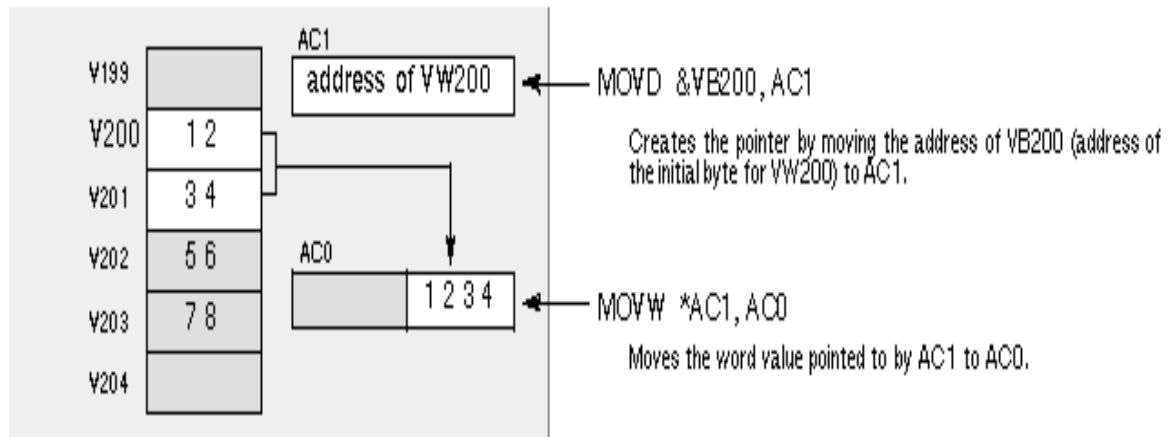
Indirect addressing

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.

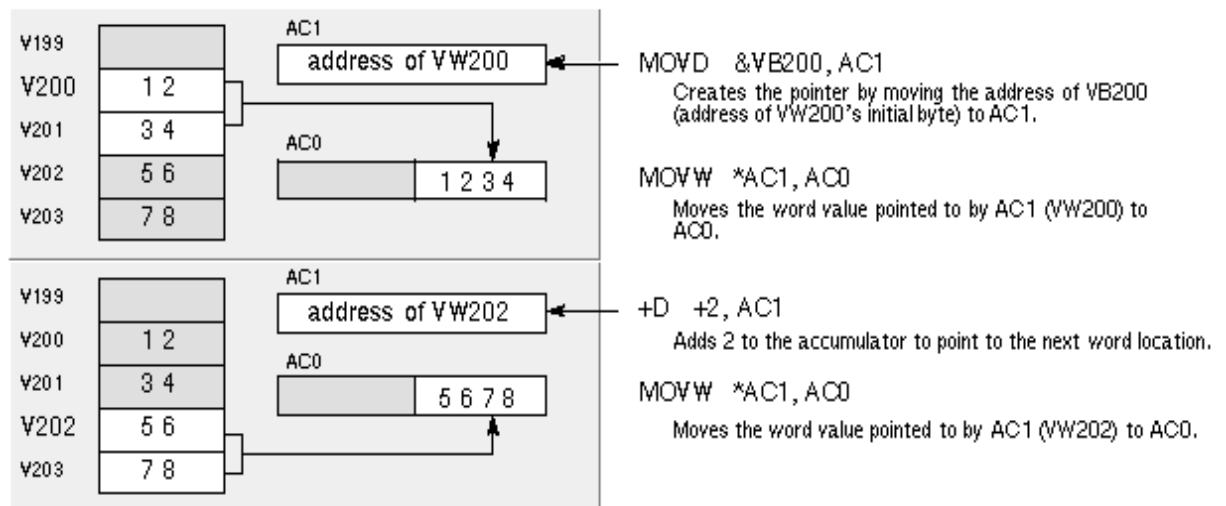
Pointer consists of memory location and symbol "&".

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~SW 30	SD	SD0~SD2 8
SM	SM0.0~S M551.7	SMB	SMB0~SM B551	SMW	SMW0~S MW550	SMD	SMD0~S MD548
T	T0~T255			T	T0~T255		
C	C0~C255			C	C0~C255		
V	V0.0~V81 91.7	VB	VB0~VB8 191	VW	VW0~VW 8190	VD	VD0~VD8 188
L	L0.0~L63. 7	LB	LB0~LB63	LW	LW0~LW6 2	LD	LD0~LD6 0
		AC	AC0~AC3	AC	AC0~AC3	AC	AC0~AC3
						HC	HC0~HC1 5

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 overflow 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 addresses 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

SMW22 Scan time of the last scan.

SMW24 Minimum scan time

SMW26 Maximum scan time

9. Easy ladder communication

9.1 PR series PLC basic introduction of network communication

PR series PLC is designed to solve your communications and networking needs. It supports both simple networks and complex networks. Easy ladder makes it simple to set up and configure your network.

Master slave network definition

PR series PLC supports master slave network. It can be used as the master station or the slave station in the network. Easy ladder 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; 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.

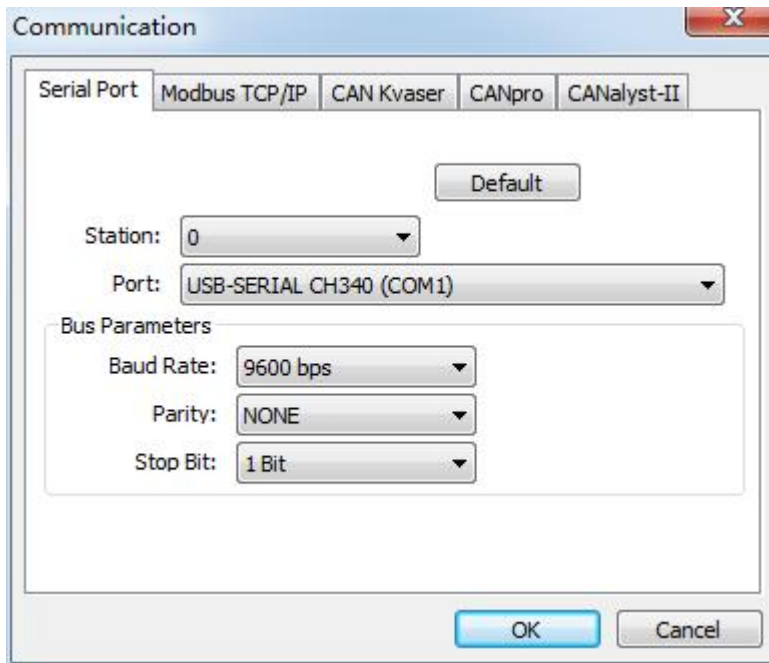
The range of PR series PLC baud rate is 1200 bps~115200 bps. The default value is 9600bps.

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 PR series PLC network addresses is 0~255. If PLC has two ports, each port can have a network address.


Set the baud rate and network address of EASY Ladder

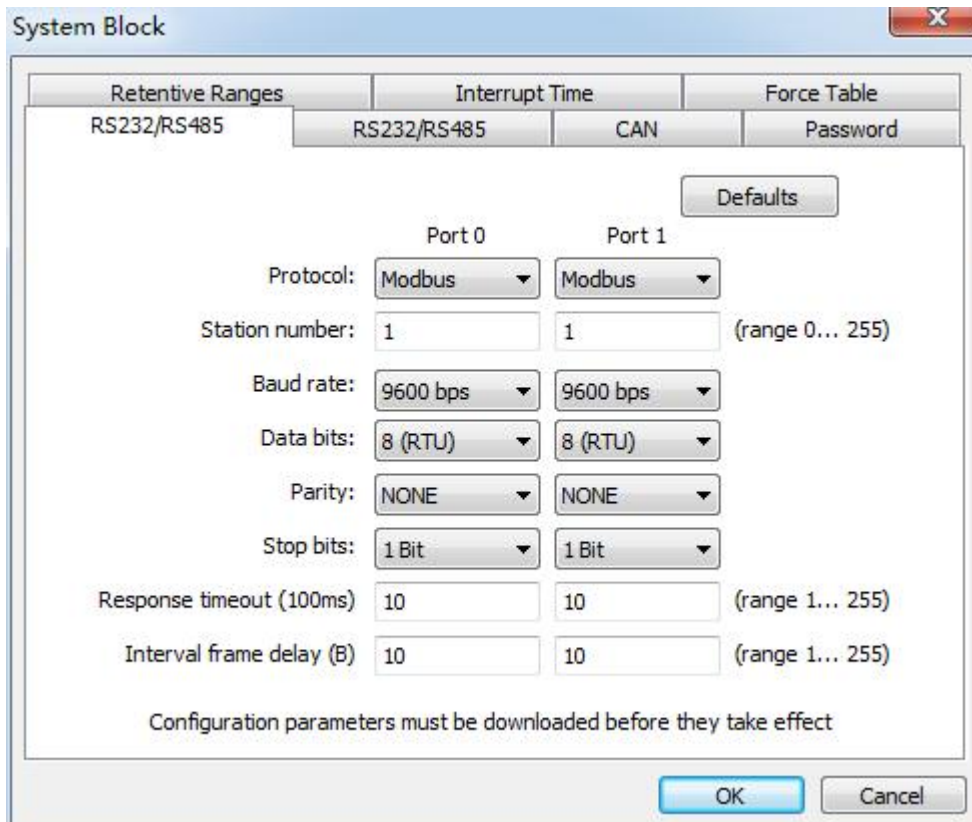
Open the communication in project management:  Communication



You can set the station number, port, baud rate, parity and stop bit of easy ladder. The default station number is 0. The default baud rate is 9600 bps.

Set the baud rate and network address of PR series PLC

Open system block in project management  System Block



You can set the station number, baud rate, data bits, parity bit and stop bit of PLC.

Attention: Only when Easy Ladder software station number is equal to 0 or PLC station number, you can download the program to PLC.

9.2 PR series PLC communication

PR series PLC support free port communication, MODBUS communication and CAN communication.

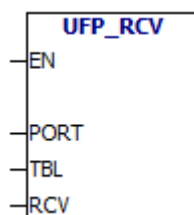
Free port communication:

Free port communication is a half duplex communication based on RS-485 communication. Users can make their own communication protocol in free port communication. Third party devices mostly support RS-485 serial communication. The core of the free port communication are receiving and sending instructions. RS-485 communication can not 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

- I PORT: Communication port.
- I 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.

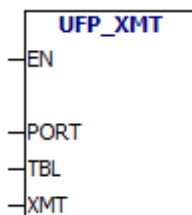
I RCV: receive data, If the input is MB400:

(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

I PORT Communication port.

I 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

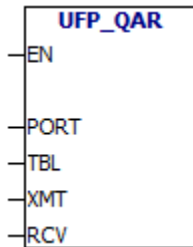
(Instruction input) M200.5 Receive CRC check

(Instruction input) M200.6 Receive CRC check

(Instruction output) MB201 Error number: 0 indicates no error.

- I 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

- I PORT Communication port.
- I 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.
- I 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

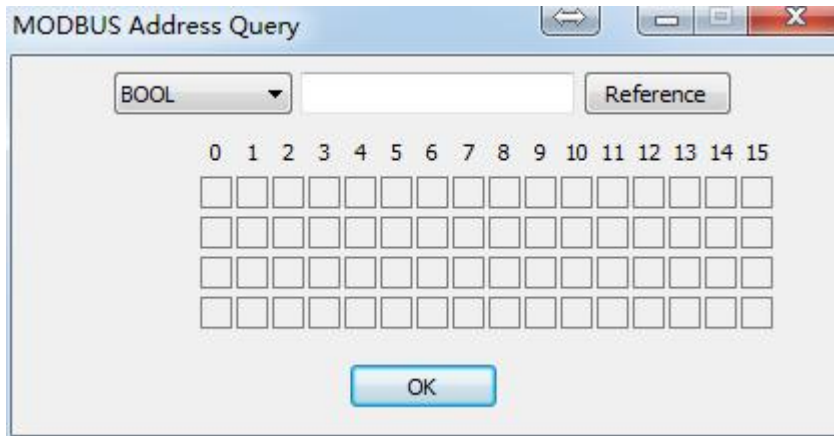
3XXXX are analog inputs

4XXXX are hold registers

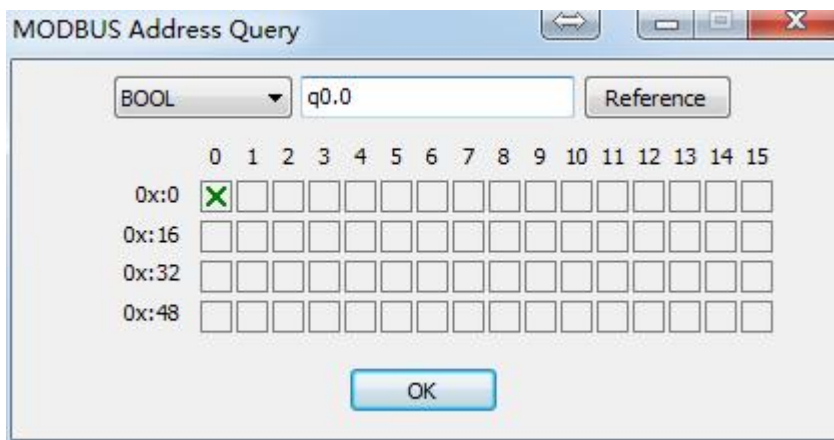
You can use “MODBUS address query” to query the MODBUS address of the variable,

Steps are as follows:

Menu bar→PLC→MODBUS address query

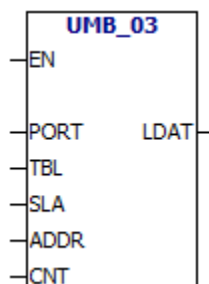


For example: Query the MODBUS address of Q0.0 :



MODBUS instructions:

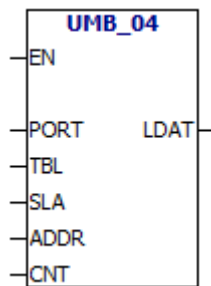
UMBO3



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

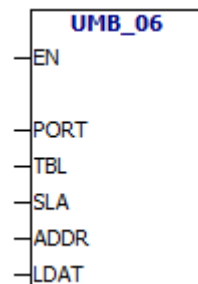
UMB04



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

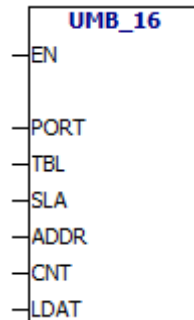
UMB06



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

- I EN: enable or not enable
- I 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.
- I SLA: MODBUS slave address
- I ADDR: The offset of hold register (The offset of 4X)
- I 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.

CAN communication

CAN communication is not stable at present. Suggest you choose other means of communication.

The use of CAN 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 online. 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 not be placed between the master station addresses. Or it will increase the additional requirements of the network.

10. Additional chapter

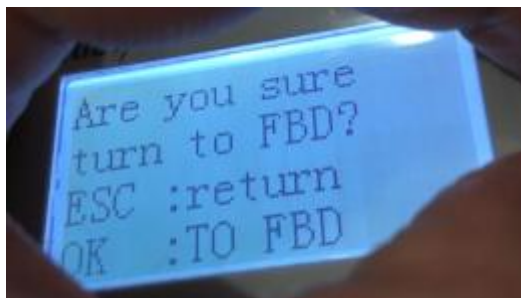
10.1 How to switch PLC mode

Ladder diagram → FBD

1. Ensure that there is no power supply to the PLC.
2. Press ESC key and UP key, Keep pressing.



3. Power supply to PLC. Keep pressing until the following picture appears.



4. Release the hand, press the OK key

FBD → Ladder diagram

Repeat the above steps.

10.2 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 module, The value range of the analog quantity is -500~2000.It corresponds to the temperature that is -50°C~200°C.

10.3 Extension module address

You can use the dial switch to set the address.The address of each extension module can not be the same.

Digital quantity input extension address table:

Extension Address	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Start Address	I2.0	I3.0	I4.0	I5.0	I6.0	I7.0	I8.0	I9.0	I10.0	I11.0	I12.0	I13.0	I14.0	I15.0	I16.0	I17.0

Digital quantity output extension address table:

Extension Address	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Start Address	Q2.0	Q3.0	Q4.0	Q5.0	Q6.0	Q7.0	Q8.0	Q9.0	Q10.0	Q11.0	Q12.0	Q13.0	Q14.0	Q15.0	Q16.0	Q17.0

Analog quantity input extension address table:

Extension Address	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Start Address	AIW 20	AIW 30	AIW 40	AIW 50	AIW 60	AIW 70	AIW 80	AIW 90	AIW 100	AIW 110	AIW 120	AIW 130	AIW 140	AIW 150	AIW 160	AIW 170

Analog quantity output extension address table:

Extension Address	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Start Address	AQW 20	AQW 30	AQW 40	AQW 50	AQW 60	AQW 70	AQW 80	AQW 90	AQW 100	AQW 110	AQW 120	AQW 130	AQW 140	AQW 150	AQW 160	AQW 170

10.4 PLC host address range

Digital input: I0.0~I1.7

Digital output: Q0.0~Q1.7

Analog input: AIW0~AIW18

Analog output: AQW0~AQW18

10.5 Formula

Digital quantity extension addressing formula:

Digital input Start Address= I (Extension Address+1) .0

Digital output Start Address= Q (Extension Address+1) .0

Analog quantity extension addressing formula:

Analog input Start Address= AIW (Extension Address $\times 10 + 10$)

Analog output Start Address= AQW (Extension Address $\times 10 + 10$)

Up to 16 extension modules can be connected.

10.6 Set extension module address with a dial switch



The address of extension module = The value of dial switch + 1

Dial switch	Value
1 →	1
2 →	2
3 →	4
4 →	8

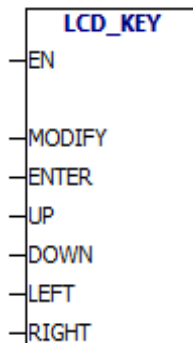
The value of dial switch				The address of extension module
1	2	3	4	
OFF	OFF	OFF	OFF	1
ON	OFF	OFF	OFF	2
OFF	ON	OFF	OFF	3
OFF	OFF	ON	OFF	5
OFF	OFF	OFF	ON	9
ON	ON	OFF	OFF	4
ON	OFF	ON	OFF	6
ON	OFF	OFF	ON	10
OFF	ON	OFF	ON	11
OFF	OFF	ON	ON	13
OFF	ON	ON	OFF	7
ON	ON	ON	OFF	8

ON	ON	OFF	ON	12
ON	OFF	ON	ON	14
OFF	ON	ON	ON	15
ON	ON	ON	ON	16

10.7 Additional instructions

10.7.1 LCD related instructions

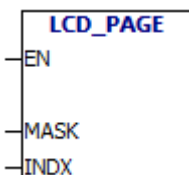
LCD_KEY



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.

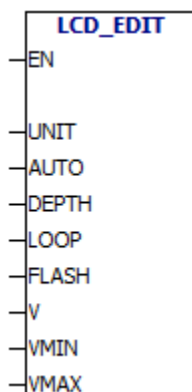
.

.

.

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.

I EN: Enable

I UNIT: Edit the number of objects in the page

I AUTO: Whether uses LCD keys to edit.

I DEPTH: The current edit depth of edit object.

I LOOP: LOOP edit.

I FLASH: The edit object is flashing or not.

I V: The current value of edit object.

I VMIN: The minimum value of edit object.

I VMAX: The maximum value of edit object.

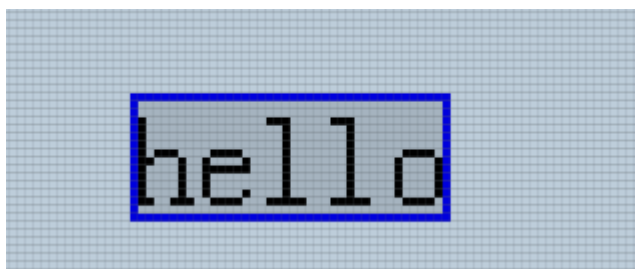
For example:

You have to edit display pages in LCD software.

Display page 1:



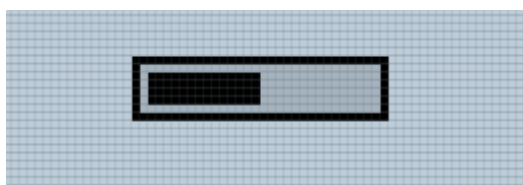
Display page 2:



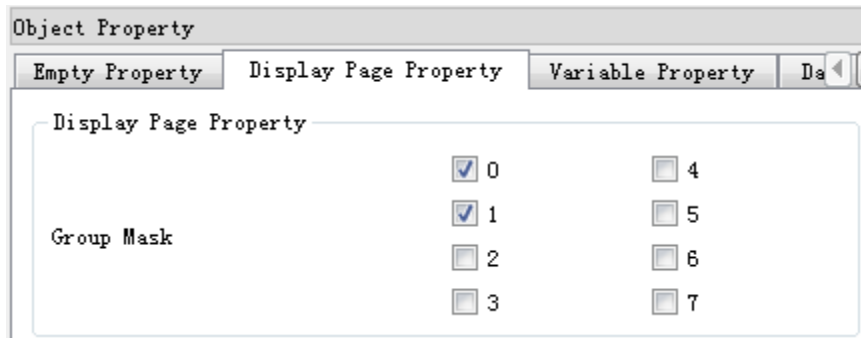
Display page 3:



Display page 4:



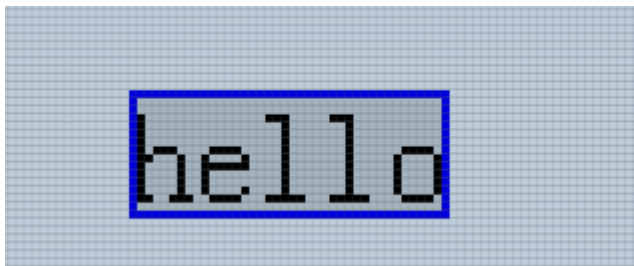
Grouping of display pages in display page property:



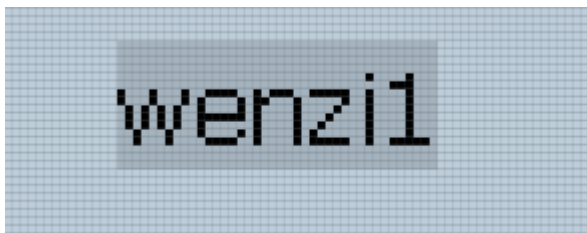
Display page 1: Display page 1 is divided into 0 group and 1 group.



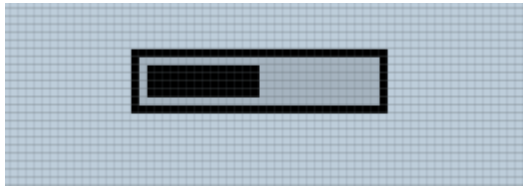
Display page 2: Display page 2 is divided into 0 group and 1 group.



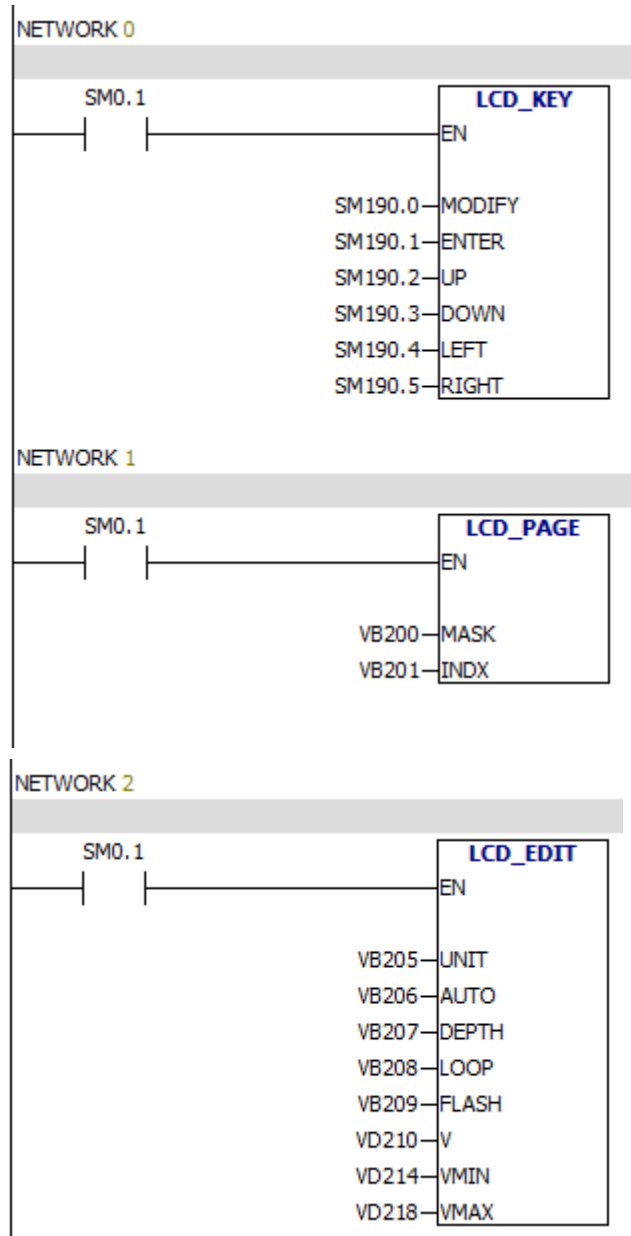
Display page 3: Display page 3 is divided into 0 group.



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.

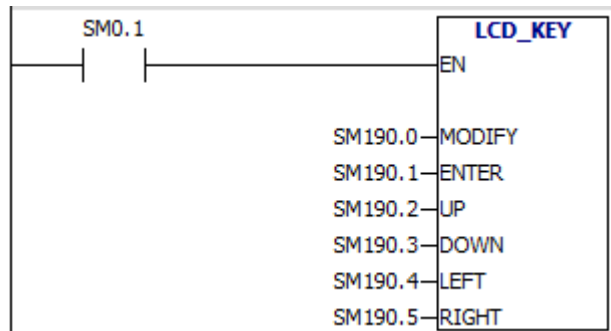
F1 corresponds to SM191.0

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.

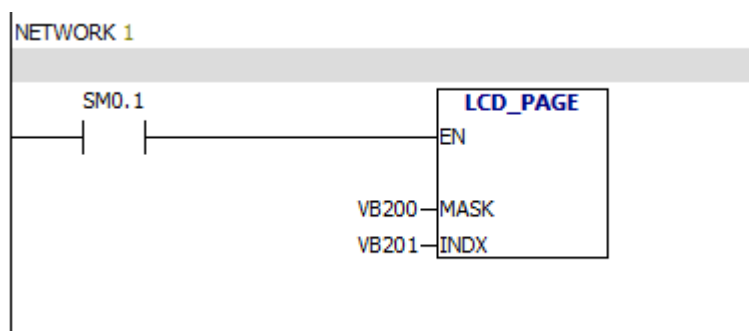
ESC is used for modifying values and exiting edit.

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:

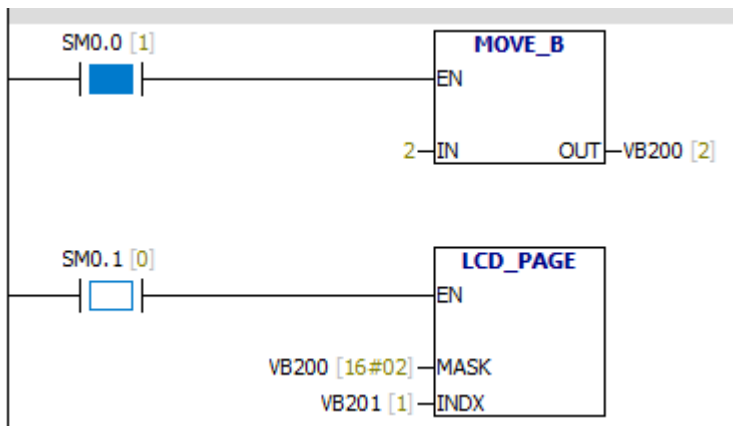
Address	Data Type	Value
VB200	BYTE	16#01
VB201	SINT	0

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 display page 1.

You can use the program to specify the display group and the display page .

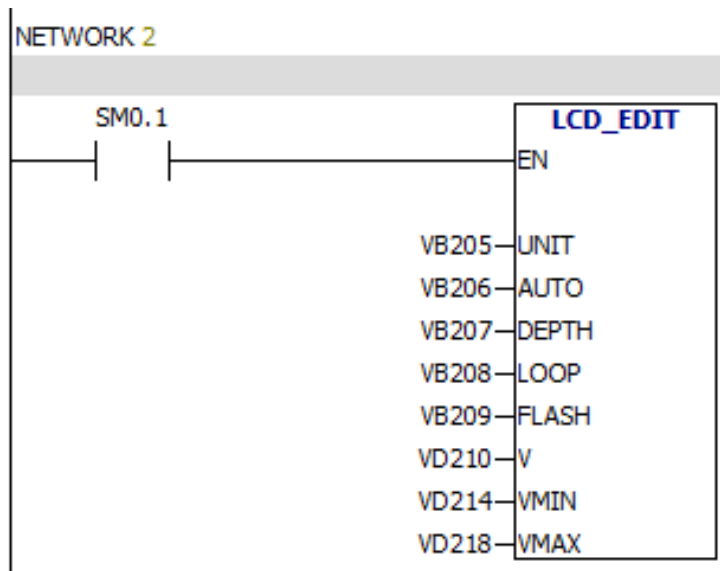
For example:



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:



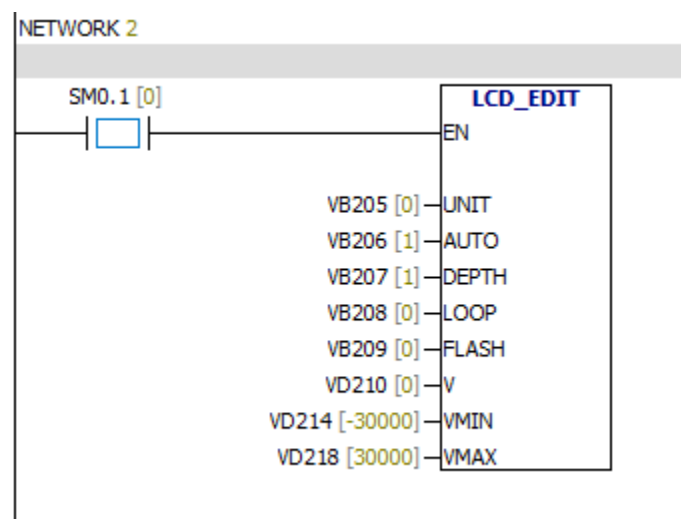
LCD -EDIT instruction binds the PLC variables and LCD edit states.

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

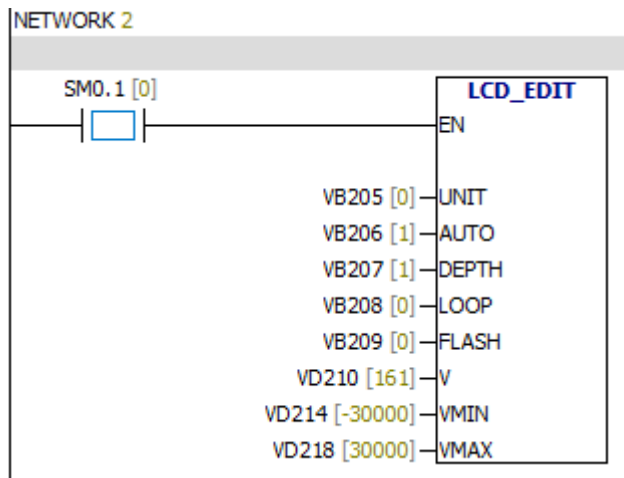
Vd210 = 0 The current value of variable is 0

Vd214 = -30000 The minimum value is -30000

Vd218 = 30000 The maximum value is 30000

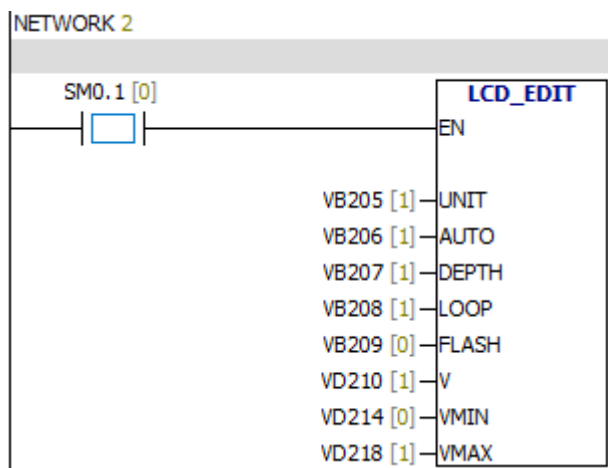
For example:

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

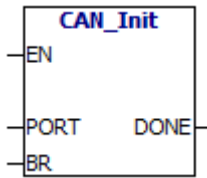
VD210 The current value of variable is 1

VD214 The minimum value is 0

VD218 The maximum value is 1

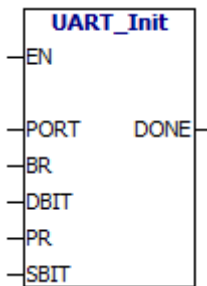
10.7.2 CAN, serial port initialization instructions

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

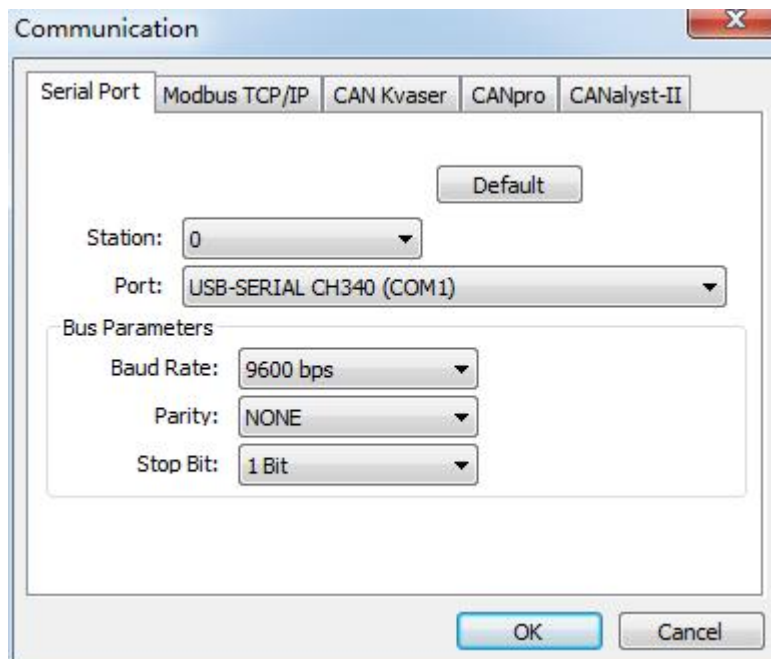


UART_Init instruction is used to initialize the serial port.

- | 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

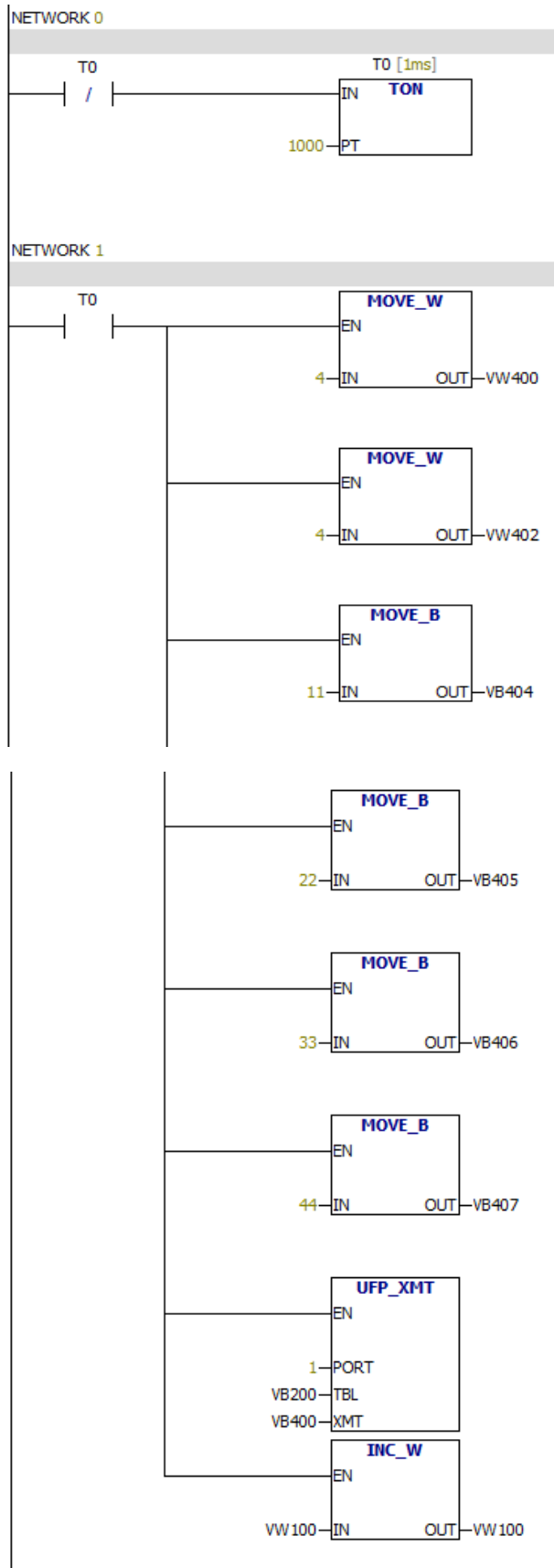
You can also set these parameters in the programming software.

As shown in the following picture:



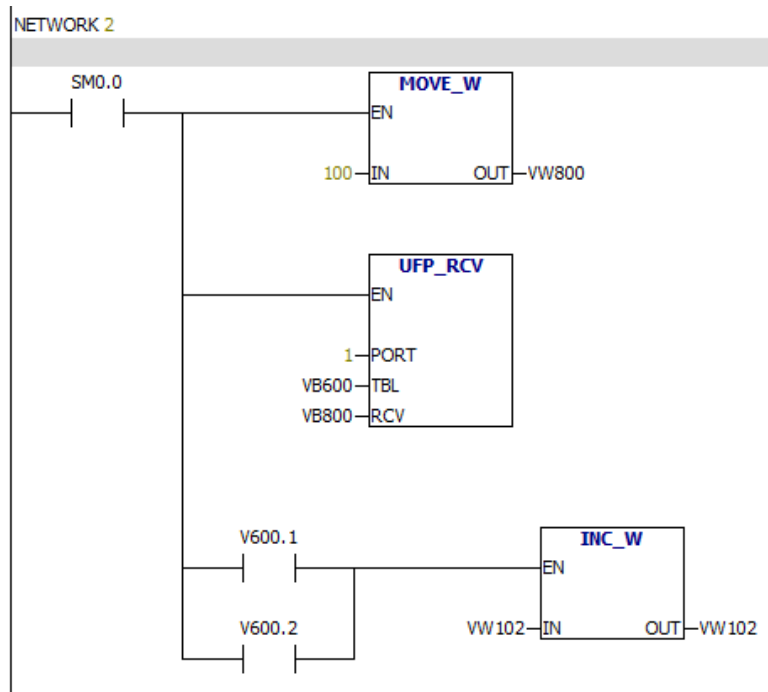
10.8 Example of serial port free port communication

Program 1:



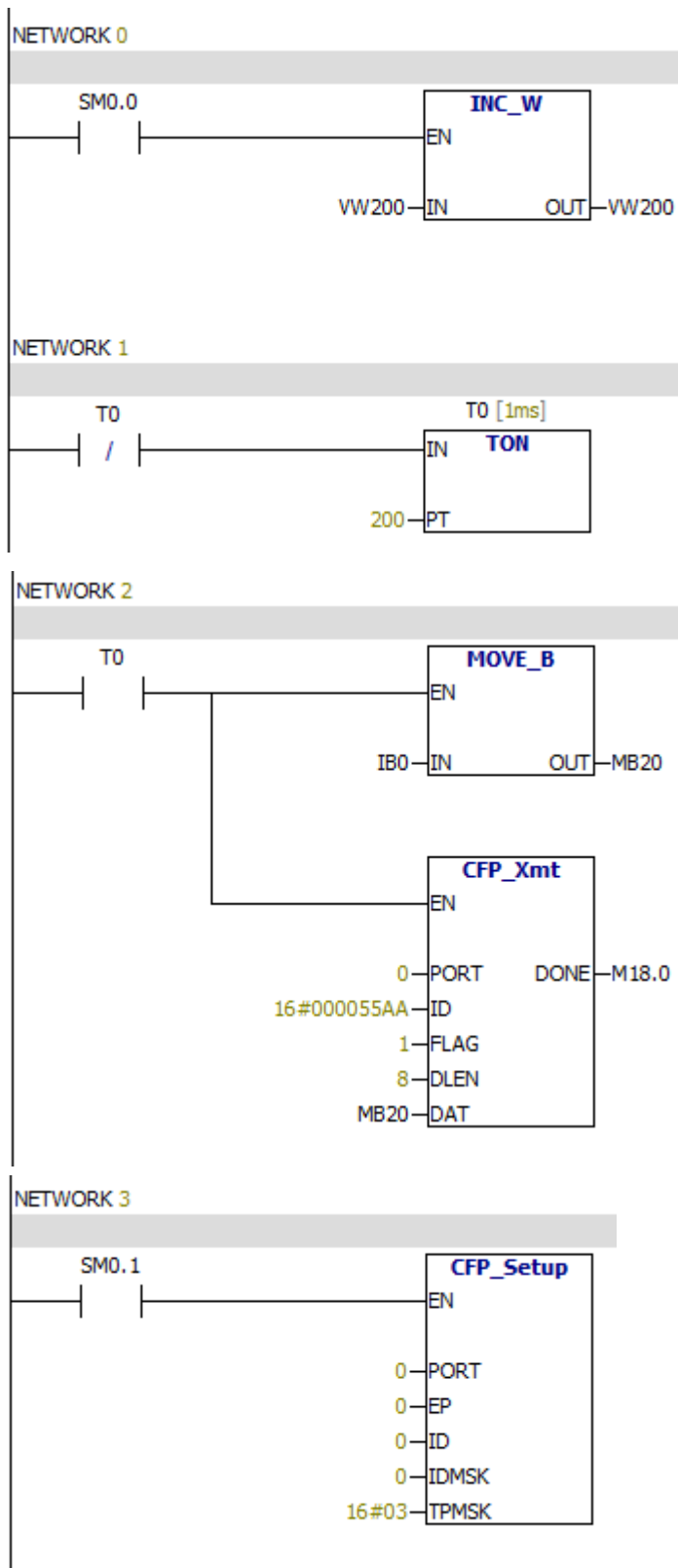
Explain: Send data 11 22 33 44 per second through port 1. And record the counts of sending data.

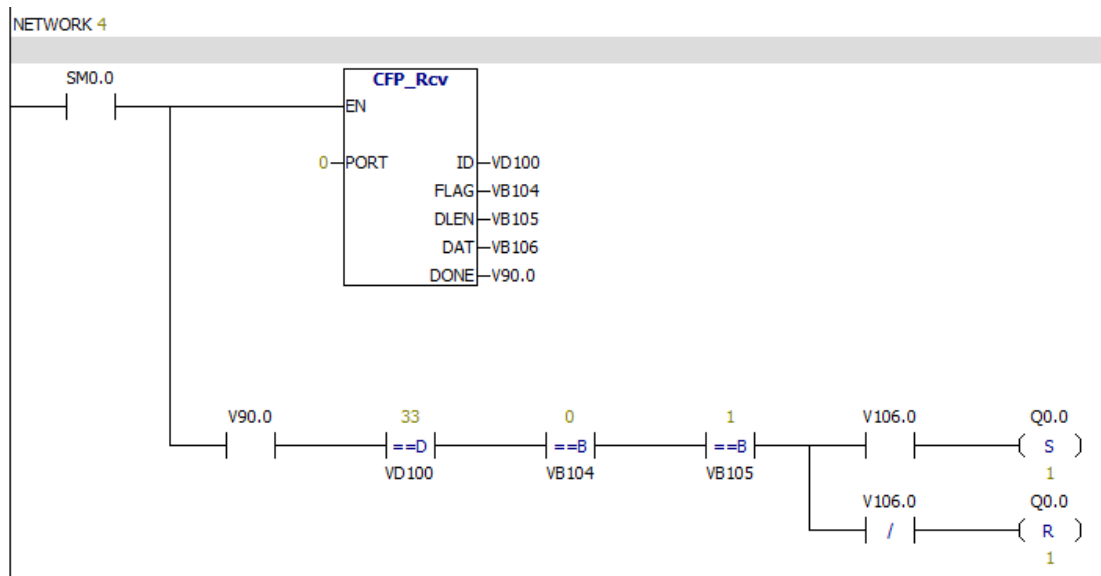
Program 2:



Explain: Receive data through port 1. The maximum length of the data is 100 bytes.

10.9 Example of CAN free port

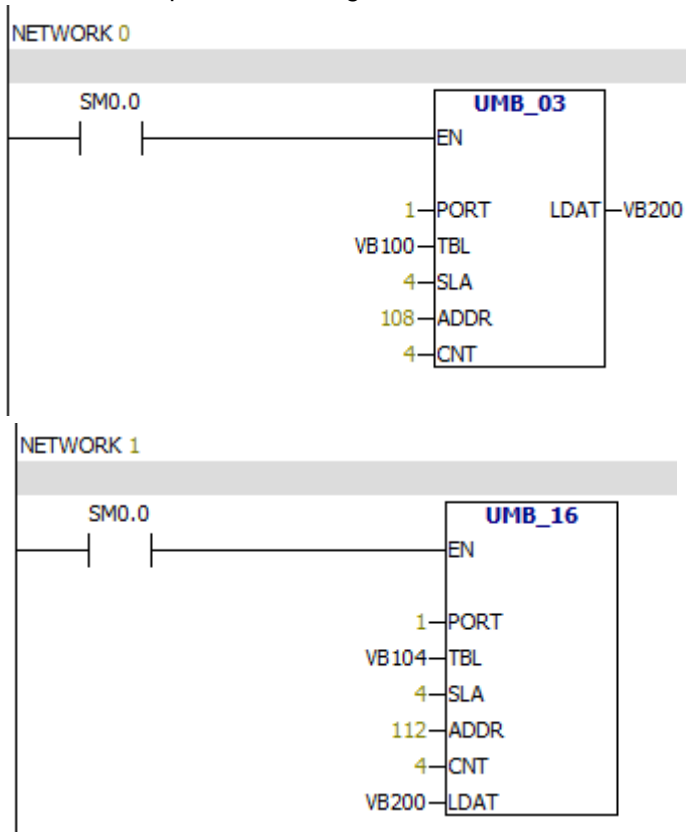




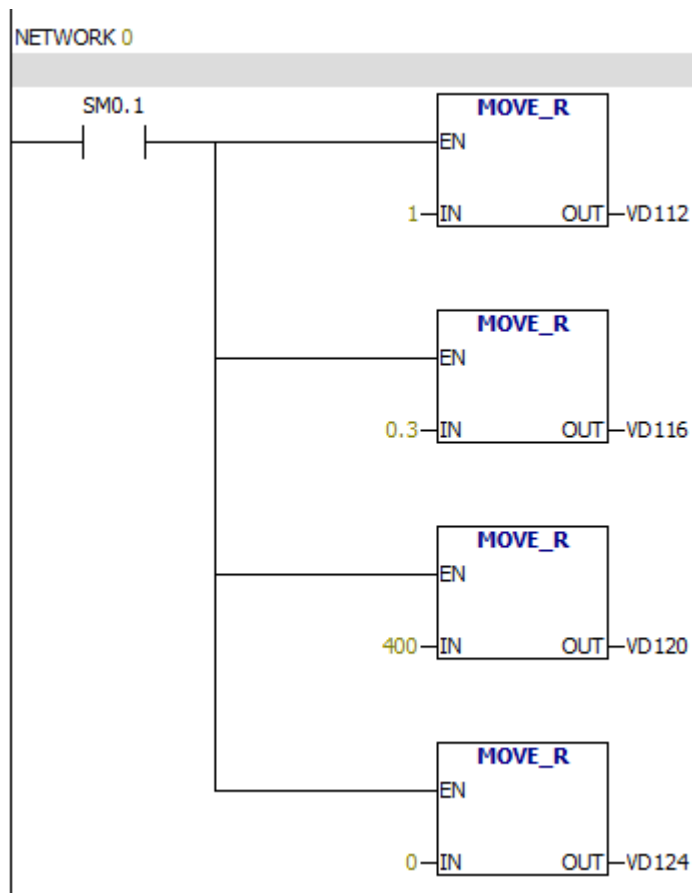
Explain: The state of the IO.0 is transmitted through port 0. Receive data VB106 through port0. The state of the first bit of VB106 is the state of Q0.0

10.10 MODBUS communication master program

Read multiple hold registers and write multiple hold registers



10.11 The example of using PID instruction



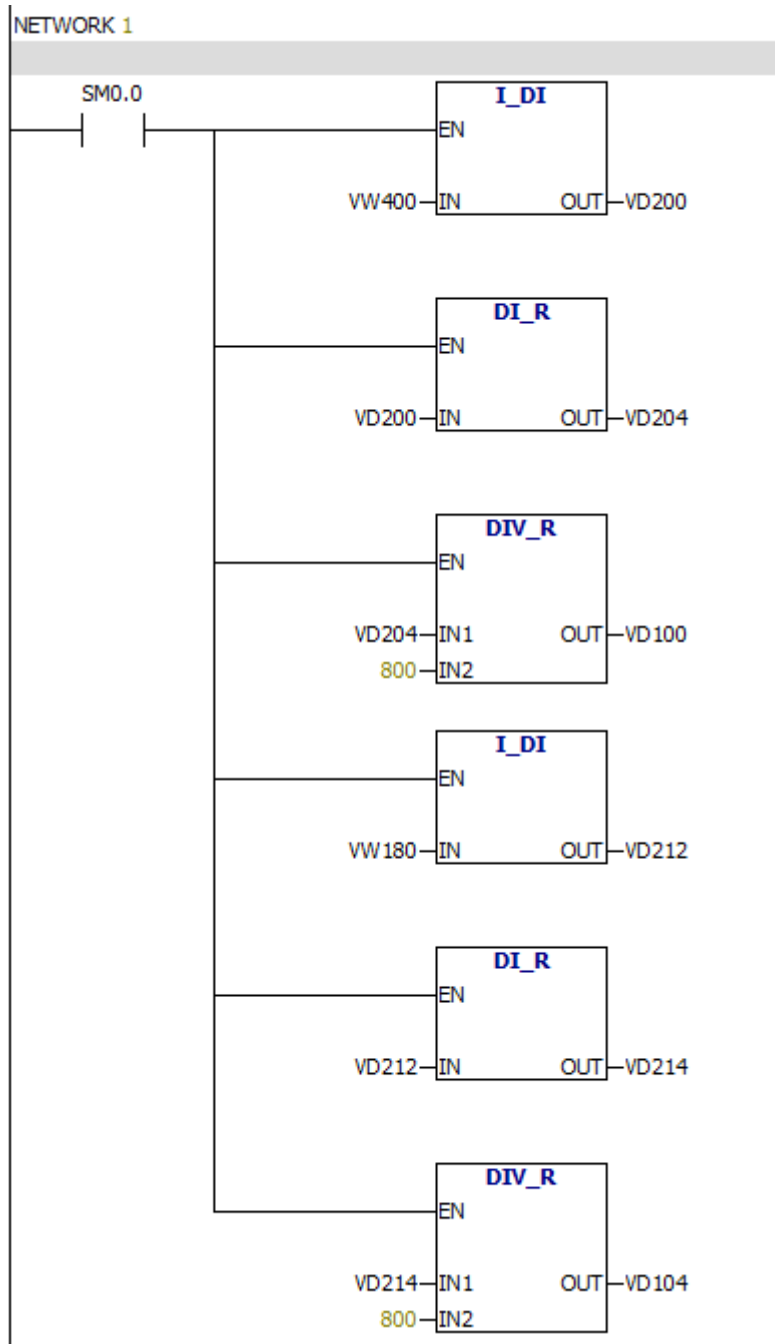
The initialization of PID parameters

VD112 gain

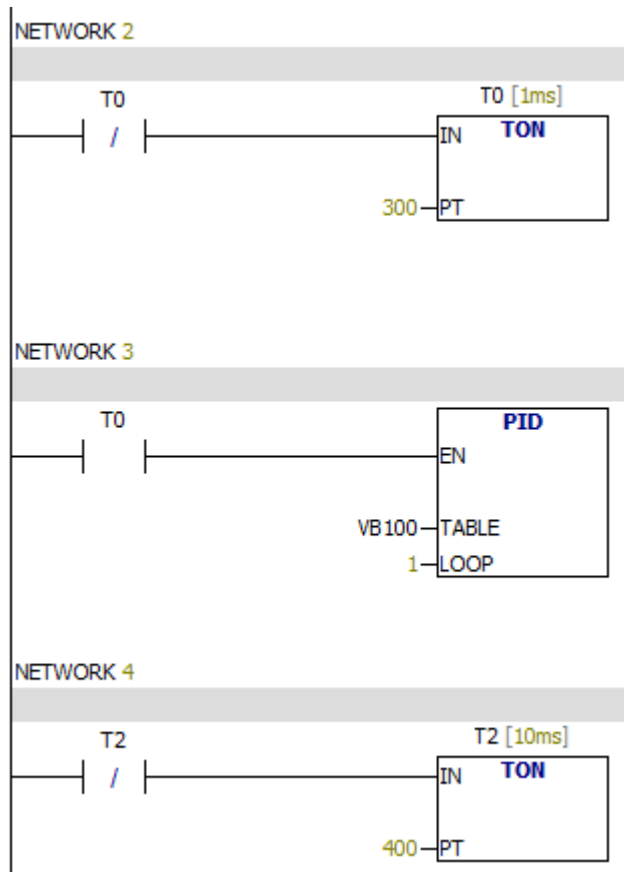
VD116 Sampling time

VD120 Integral time

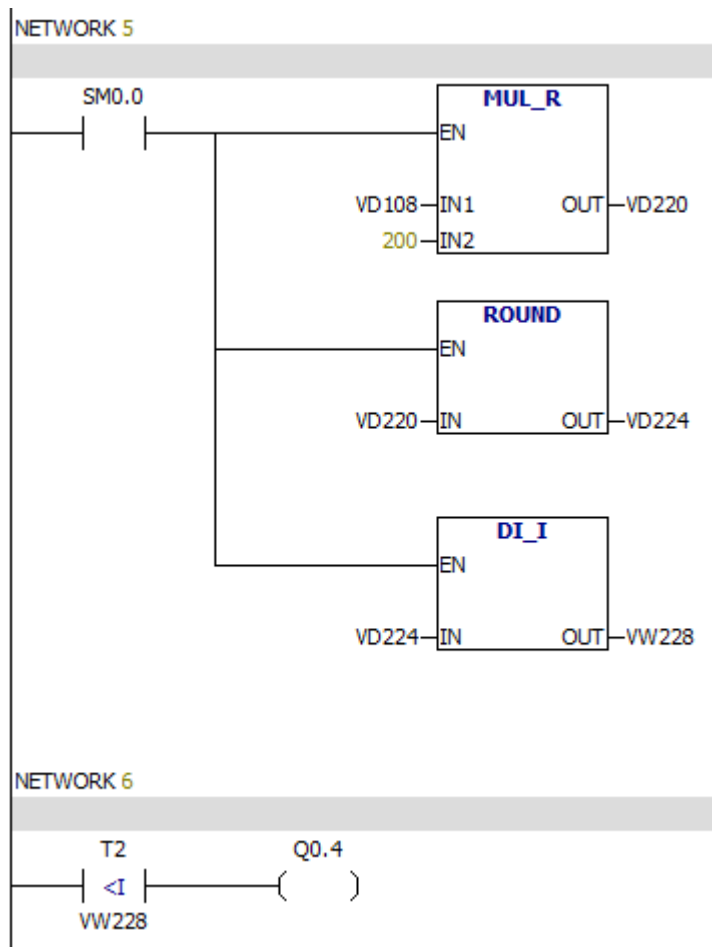
VD124 Differential time



Conversion of Process quantity and set value unit



Call a PID command every 0.3 seconds.



The conversion of output value unit .

MODBUS ADDRESS

Bit		Modbus Adr.	Modbus code	R/W
I	I0.0--I31.7	0--255	02	R
SM	SM0.0--SM551.7	512--4927	02	R
Q	Q0.0--Q31.7	0--255	01/05 (15)	R/W
M	M0.0--M31.7	512--767	01/05 (15)	R/W
V	V0.0--V8063.7	768--65279	01/05 (15)	R/W
S	S0.0--S31.7	65280--65535	01/05 (15)	R/W
T	T0-T255	256--511	02	R
C	C0-T255	5000--5255	02	R

Word		Modbus Adr.	Modbus code	R/W
VW	VW0--VW8190	0--8190	03/06 (16)	R/W
MW	MW0--MW30	8200--8230	03/06 (16)	R/W
SMW	SMW0--SMW550	8300--8850	03/06 (16)	R/W
SW	SW0--SW30	9000--9030	03/06(16)	R/W
T	T0-T255	9100--9355	03/06 (16)	R/W
C	C0-T255	9500--9755	03/06 (16)	R/W
AIW	AIW0--AIW178	0--178	04	R
AQW	AQW0--AQW178	200--378	04	R
IW	IW0--IW30	400--430	04	R
QW	QW0--QW30	500--530	04	R

DWord		Modbus Range	Adr. Address calculation formula	Modbus code	R/W
VD	VD0--VD8188	10000--26376	VD: 10000+(2*No.)	03/(16)	R/W
MD	MD0--MD28	30000--30056	MD: 30000+(2*No.)	03/(16)	R/W
SMD	SMD0--SMD548	31000--32096	SMD: 31000+(2*No.)	03/(16)	R/W
SD	SD0--SD28	33000--33056	SD: 33000+(2*No.)	03/(16)	R/W
ID	ID0--ID28	600--656	ID: 600+(2*No.)	04	R
QD	QD0--QD28	700--756	QD: 700+(2*No.)	04	R

For example :

