EC Series Small PLC

Programming Manual

Version V1.0
Revision date March 17, 2019

Control Technology provides customers with technical support. Users may contact the nearest COMPANY local sales office, service center or headquarters.

Control Technology Co., Ltd.

Control Technology Co., Ltd.
Address:
Postal code:
Website:
E-mail:

Preface

Target reader

This book is suitable for the automation personnel who need to master the PLC programming, system design and commissioning. This book can also serve as a reference for anyone who are interested in futhering their PLC programming knowledge.

Content of this book

This book details the principles, hardware resources, programming languages and instructions of EC series small PLC. A variety of application illustrations are used to help you understand the rich functions of PLC.

Features of this book

The chapters in this book develop from general to details, each having its independent topic. You can either read thoroughly to gain overall knowledge of EC series small PLC or consult in some of the chapters for technical reference.

Reading instructions

1. For readers unfamiliar with PLC

It is recommended to start with chapters 1~4 to learn the basic PLC knowledge, including PLC function description, programming languages, elements & data, addressing modes, program annotating function, main program and subprograms. Afterwards, you can read other chapters to cater for your needs.

2. For readers familiar with PLC

You can jump directly to ***Error! Reference source not found.Error! Reference source not found.*** and ***Error! Reference source not found.****Application Instructions*, which provide complete and detailed explanation for the instructions of COMPANY EC series PLC. For sequential function chart (SFC), high-speed I/O, interrupts and communication function, please refer to chapters 7~10. For positioning control, please refer to *Appendix 10 Positioning Function Guideline*. In addition, *Appendix 9Instruction index* and *Appendix 10Classified instruction index* provide tools for locating the instructions in the orders of alphabet and classification respectively.

Related documents and references

You can refer to the following manuals while reading this manual:

- *EC10 Series PLC User Manual*
- *EC20 Series PLC User Manual*
- *Programmer Programming Software User Manual*

Content

Chapter 1 Product overview

1.1 Product introduction

The EC series small PLC, comprising the EC10, EC10A, EC10V mini-scale series and EC20, EC20H small series, is a high performance product suitable for modern industrial control.

The EC series PLC products have integrated structure, built-in high performance microprocessor, operation control system, integrated I/O and extension bus. The series also include I/O extension modules and special modules. The main module has 2~3 communication ports, and the sytem can connect to the profibus network through a profibus extension module. The main module I/O also has high-speed counting and high-speed output that can be used for exact positioning. The powerful Programmer programming software provides 3 standard programming languages and commissioning & monitoring functions, and boasts complete user program protection mechanism.

1.1.1 Product specification

Table 1-1 Specification of PLC main module

Name			EC20H	EC20	EC10V	EC10	EC10A
I/O	Digital I/O		16 inputs/16 outputs	20 inputs/12 outputs 32 inputs/32 outputs, 40 inputs/40 outputs	8 inputs/6 outputs 10 inputs/6 outputs 12 inputs/8 outputs 14 inputs/10 outputs	10 inputs/6 outputs 14 inputs/10 outputs 16 inputs/14 outputs 24 inputs/16 outputs 36 inputs/24 outputs 16 inputs/14 outputs/2 analog inputs/1 analog output	10 inputs/6 outputs 14 inputs/10 outputs 16 inputs/14 outputs 24 inputs/16 outputs 36 inputs/24 outputs
	Max. I/O		512	512	172	128	60
	Max. special function modules		8	8	7	7	Without
	High-speed pulse output		2×200kHz, 4×100kHz (6-axis) or 2×200kHz, 2×100kHz (4-axis)	2×100kHz (only apply to transistor output)	2×100kHz 2×10kHz (only apply to transistor output)	2×100kHz (only apply to transistor output)	2×50kHz (only apply to transistor output)
	Single-phase counting channel		8×100kHz	6: 2 50kHz/4 10kHz			
	Dual-phase counting channel		4×50kHz	2: 1 30kHz/1 5kHz			
	Max. frequency sum of high-speed counter		800kHz	80kHz	200kHz	60kHz	60kHz
	Digital filtering		X0~X7 adopt digital filtering, input filtering constant range: 0~60ms	X0~X17 adopt digital filtering, input filtering constant range: 0~60ms	X0~X7 adopt digital filtering, input filtering constant is selectable among 0, 2, 4, 8, 16, 32 and 64ms, 7 in total	X0~X7 adopt digital filtering, input filtering constant is selectable among 0, 2, 4, 8, 16, 32 and 64ms, 7 in total	X0~X7 adopt digital filtering, input filtering constant is selectable among 0, 2, 4, 8, 16, 32 and 64ms, 7 in total
	Max. relay output	Resistive load	2A/1 point; 8A/4 points group common terminal; 8A/8 points group common terminal				

Name			EC20H	EC20	EC10V	EC10	EC10A
	current	Inductive load	220Vac, 80VA				
		Light load	220Vac, 100W				
	Max. transistor output current	Resistive load	Output point: 0.3A/1 point; other: 0.3A/1 point; 0.8A/4 points; 1.6A/8 points If above 8 points, allow the total current to increase 0.1A for every additional 1 point				
		Inductive load	Y0~Y7: 7.2W/24Vdc; other: 12W/24Vdc	Y0, Y1: 7.2W/24Vdc; other: 12W/24Vdc			
		Light load	Y0~Y7: 0.9W/24Vdc; other: 1.5W/24Vdc	Y0, Y1: 0.9W/24Vdc; other: 1.5W/24Vdc			
Memory	User program		32k steps (64kByte)	12k steps (24kByte)	16k steps (32kByte)	16k steps (32kByte)	6k steps (12kByte)
	Program power-off permanent storage		Yes				
	Max. hold elements at power off		All elements except R elements	User setting (Max. C elements: 200)	Range of bit elements, 1700 word elements	320 bit elements, 180 word elements	320 bit elements, 180 word elements
	Hardware support and hold time		Standby batteries, 3-year hold time	Standby batteries, 1-year hold time	EEPROM, permanent storage	EEPROM, permanent storage	EEPROM, permanent storage
Element resource	Timer		100ms accuracy: T0~T209 10ms accuracy: T210~T479 1ms accuracy: T480~T511	100ms accuracy: T0~T209 10ms accuracy: T210~T251 1ms accuracy: T252~T255			
	Counter		16bit up counter: C0~C199 32bit up/down counter: C200~C235 32bit high-speed counter: C236~C255, C301~C306	16bit up counter: C0~C199 32bit up/down counter: C200~C235 32bit high-speed counter: C236~C255			
	Data register		D0~D7999, R0~R32767	D0~D7999	D0~D7999		D0~D3999
	Local data register		V0~V63				
	Indexed addressing register		Z0~Z15				
	Special data register		SD0~SD511	SD0~SD511	SD0~SD511	SD0~SD255	SD0~SD255
	Auxiliary relay		M0~M10239	M0~M1999	M0~M2047		M0~M1023
	Local auxiliary relay		LM0~LM63				
	Special auxiliary relay		SM0~SM511	SM0~SM511	SM0~SM511	SM0~SM255	SM0~SM255
	State relay		S0~ S4095	S0~ S991	S0~ S1023		S0~ S511
Interrupt resource	Internal timer interrupt		3	3	3		3
	External timer interrupt		16	16	16		16
	High-speed counter interrupt		8	6	6		6
	Serial port interrupt		12	12	12	8	4
	PTO output interrupt		6	2	4		2

Name		EC20H	EC20	EC10V	EC10	EC10A
	Interpolation interrupt	3	/	/	/	/
	Passed position interrupt	6	/	/	/	/
	Power loss interrupt	1	1	1		1
General	Running time of basic instruction	0.065μS	0.09μS	0.2μS	0.3μS	0.3μS
	Realtime clock	Support (at least 3-year hold time at power off)	Support (at least 3-year hold time at power off)	Support (at least 3-year hold time at power off)	Support (100-hour hold time at power off)	Support (100-hour hold time at power off)
	Analog potentiometer	Without	2/8-bit accuracy	Without	2/8-bit accuracy	Without
Communication	Communication port	PORT0: RS232 PORT1: RS485 PORT2: RS485	PORT0: RS232 PORT1: RS232/RS485	PORT0: RS232 PORT1: RS485 PORT2: RS485	PORT0: RS232 PORT1: RS232/RS485	PORT0: RS232
	Communication protocol	Modbus/free port/N:N/programming port protocol				
Access control and user program protection	Set password type	Uploading password, downloading password, monitoring password, subprogram password, prohibit formatting				
	Prohibit uploading	Support				
Application instruction	Realtime clock, clock instruction	With	With	With	With	Without
	Date and clock compare instruction	With	With	With	With	Without
	Floating point instruction	With	With	With	With	Without
	Locating instruction	With	With	With	With	Only support DRVI
	High-speed IO instruction	With	With	With	With	Not support PLS
	MODBUS and inverter instruction	With	With	With	With	Without
	Read and write EEPROM instruction	Without	With	With	With	Without
	Computation control instruction	With	With	With	With	Only support PID
	String instruction	With	Without	Without	Without	Without
	Batch data processing instruction	With	Without	Without	Without	Without
	Data sheet instruction	With	Without	Without	Without	Without
	Memory card instruction	With	Without	Without	Without	Without
MTBF	Relay output	200,000 hours (for ground fixation, mechanical stress close to zero, with temperature and humidity control)				
		100,000 hours (for ground fixation, mechanical stress close to zero, no temperature and humidity control)				
	Transistor output	300,000 hours (for ground fixation, mechanical stress close to zero, with temperature and humidity control)				
		150,000 hours (mechanical stress close to zero, no temperature and humidity control)				
Contact life of output relay	220Vac/15VA/ inductance	1s ON/1s OFF, 3,200,000 times				
	220Vac/30VA/ inductance	1s ON/1s OFF, 1,200,000 times				
	220Vac/72VA/ inductance	1s ON/1s OFF, 300,000 times				
Power feature	Input voltage range	85Vac~264Vac (normal)				

Name	EC20H	EC20	EC10V	EC10	EC10A

Note:

1. For detailed product specifications, installation instructions, operation and maintenance guidelines of EC10 series PLC, please refer to *EC10 Series PLC User Manual*
2. For detailed product specifications, installation instructions, operation and maintenance guidelines of EC20 series PLC, please refer to *EC20 Series PLC User Manual*
3. For detailed product specifications, installation instructions, operation and maintenance guidelines of EC20H series PLC, please refer to *EC20H Series PLC User Manual*
4. Under 25℃ running environment temperature, the hold time of standby batteries is 3 years

1.1.2 Outline of EC10/10V series main module

The outline and structure of EC10/10Vseries main module are shown in the following figure (take EC10-1614MAR for example):

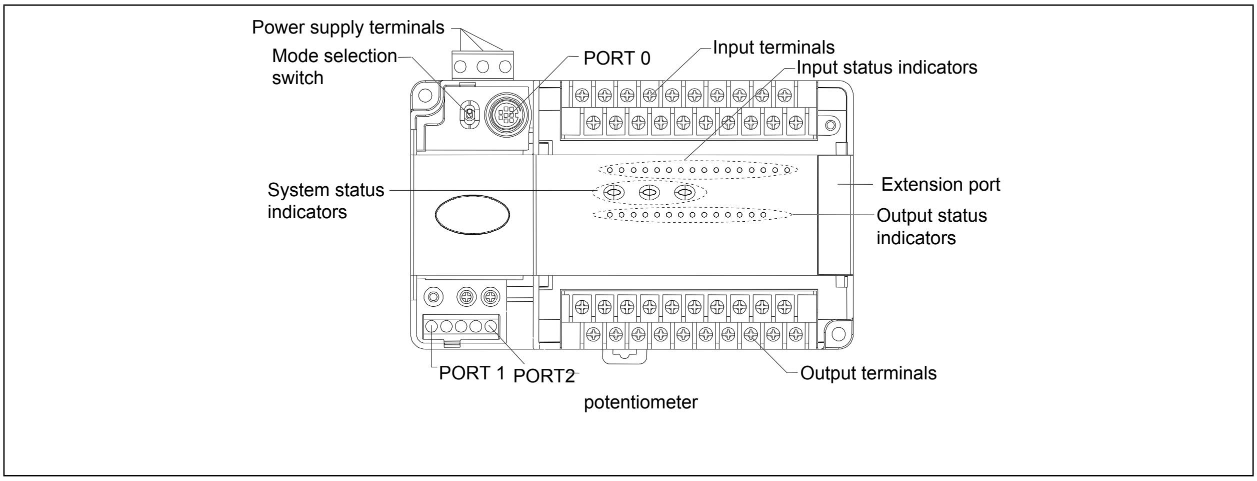


Figure 1-1 Outline and structure of EC10/10V series main module

PORT0 and PORT1 are for communication. PORT0 is RS232, and use socket Mini DIN8, while EC10 series PORT1 is RS485 or RS232, EC10V series PORT1 and PORT2 is RS485. The bus socket is for connecting extension modules. The mode selector switch can be set to ON, TM or OFF.

1.1.3 Outline of EC20 series main module

The outline and structure of EC20 series main module are shown in the following figure (take 64-point main module for example):

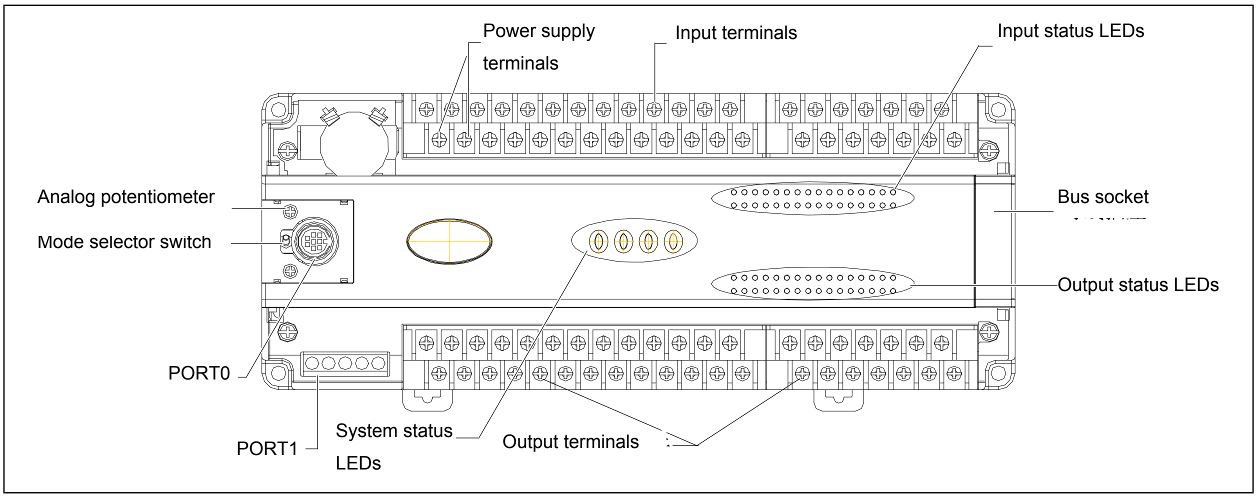


Figure 1-2 Outline and structure of EC20 series main module

The battery socket is designed for CR2354 lithium battery. The bus socket is for connecting extension modules. PORT0 is RS232 and uses socket Mini DIN8, while the communication port PORT1 is RS485 or RS232. The mode selector switch can be set to ON, TM or OFF.

1.1.4 Outline of EC20H series main module

The outline and structure of EC20H series main module are shown in the following figure (take 32-point main module for example):

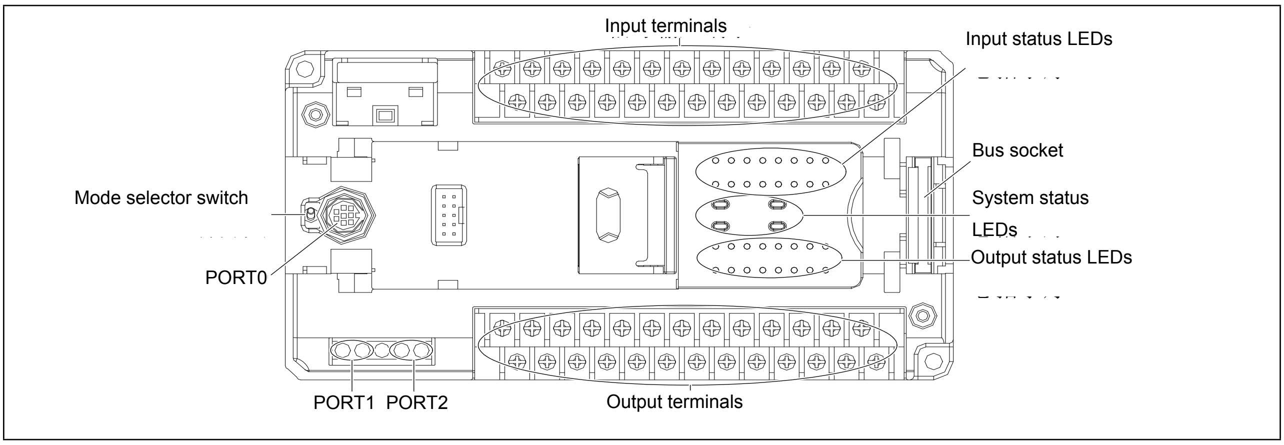


Figure 1-3 Outline and structure of EC20H series main module

The battery socket is designed for CR2354 lithium battery. The bus socket is for connecting extension modules. PORT0 is RS232 and uses socket Mini DIN8, while the communication port PORT1 and PORT2 are RS485. The mode selector switch can be set to ON, TM or OFF.

1.2 Programmer programming software

Programmer is a programming software specialized for EC10, EC10A, EC10V, EC20 and EC20H series PLC. You can download it at www.Company.com.

Programmer programming software is a standard Windows-based diagram programming-tool, operated through the mouse and keyboard. Three programming languages are available: ladder diagram (LAD), instruction list (IL) and Sequential Function Chart (SFC).

The serial port programming cable is used to connect Programmer programming platform with PLC. You can realize Modbus network programming through serial port conversion and remote programming through a modem. Refer to *Programmer Programming Software User Manual* for Modbus programming and remote monitoring.

1.2.1 Basic configuration

Programmer programming software requires an IBM PC and Microsoft Windows series OS. The compatible OSs include Windows 7, Windows 10 and Windows XP.

The minimum and recommended configuration is listed below:

Table 1-1 Basic configuration of Programmer programming environment

Item	Minimum	Recommended
CPU	Equivalent to Intel Pentium 233 or above	Equivalent to Intel Pentium 1G or above
Memory	64M	128M
Display card	Support 640×480 resolution and 256 colors	Support 800×600 resolution and 65535 colors
Communication port	A RS232 serial port with DB9 socket (or a USB port and a USB-RS232 converter)	
Others	Programming cable special for COMPANY PLC	

1.2.2 Programmer installation

The Programmer installation package issued by Control Technology Co., Ltd. (COMPANY for short) is an executable program. Double click it to start the installation, and follow the prompts step by step. You can select an installation path according to your actual need.

After the installation, **COMPANY** program group will be added to the start menu. An Programmer shortcut icon will also be added to the desktop. Double click the shortcut icon to run the program.

You can uninstall the Programmer software through the Windows control panel. To install the Programmer software in a new version, you have to uninstall the old version at first.

1.2.3 Programmer operation interface

The main interface includes 7 sections: menu, toolbar, project management window, instruction tree window, information window, status bar and operation area.

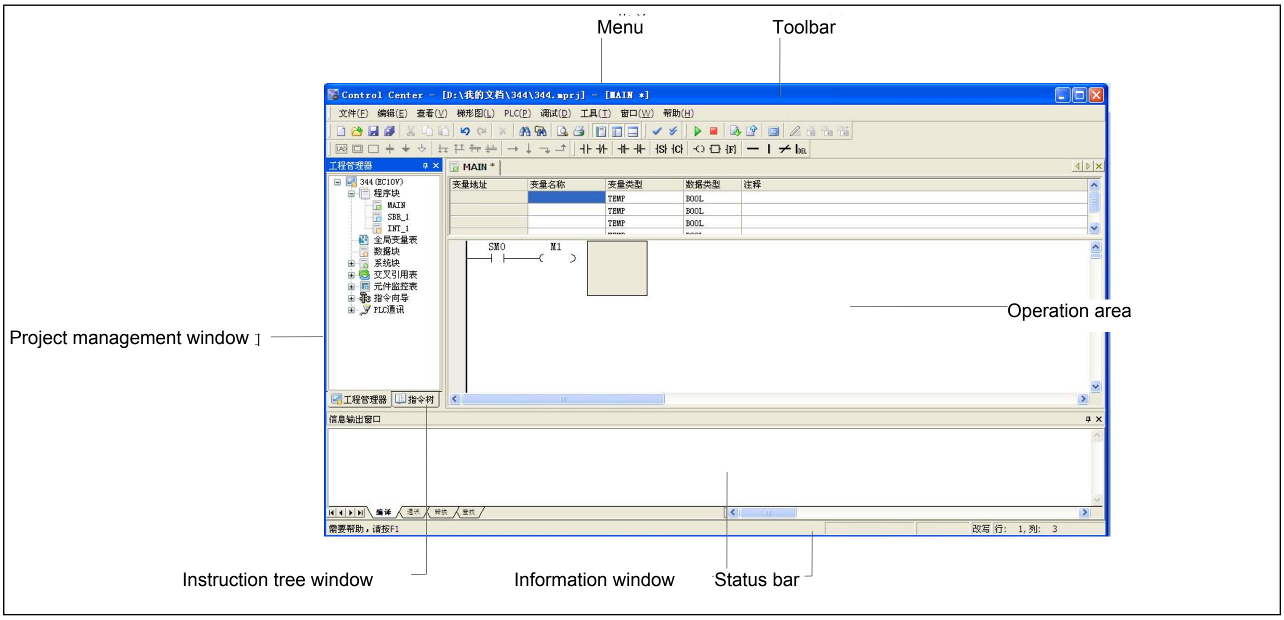

Menu
Toolbar
Control Center - [D:\我的文档\344\344.mprj] - [MAIN *]
文件(F) 编辑(E) 查看(V) 梯形图(L) PLC(P) 调试(D) 工具(T) 窗口(W) 帮助(H)
Project management window
Operation area
Instruction tree window
Information window
Status bar

Figure 1-1 Main interface of Programmer

For the usage of Programmer programming software, refer to *Programmer Programming Software User Manual*.

1.2.4 Programming cable

You can use the programming cable provided by Control Technology Co., Ltd. to program and debug the PLC. There are three kinds of cables: one is optically isolated and hot swappable; one is non-isolated and not hot swappable; another is USB converted to RS232 and hot swappable. None of them requires setting jumpers.

See the following figure for the connection of the programming cable.

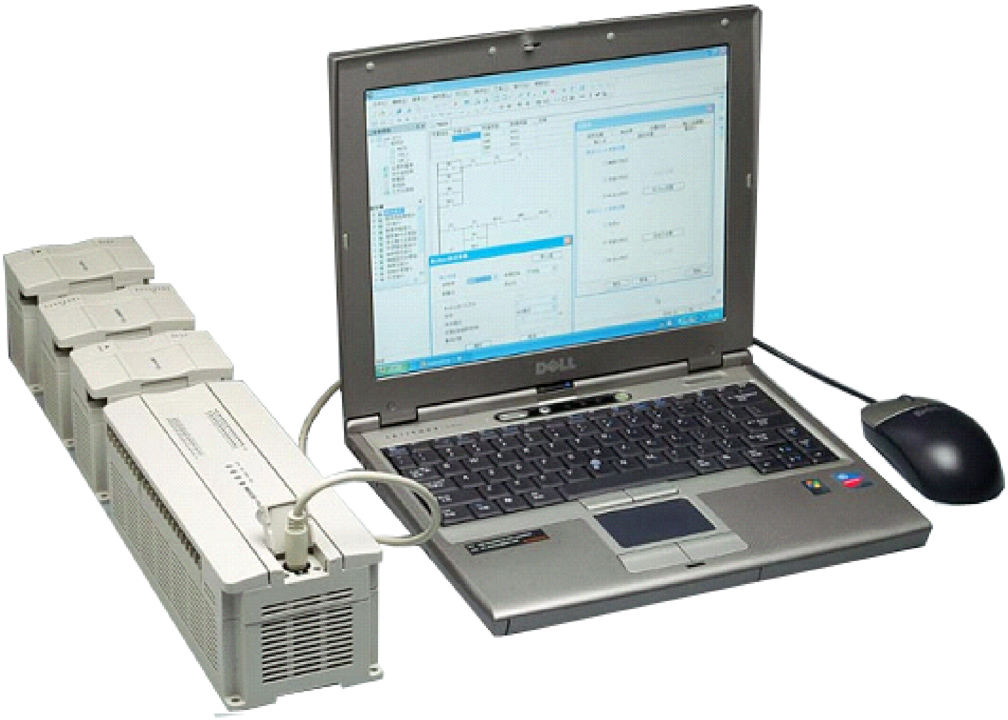


Figure 1-1 Connection of programming cable

1.3 Communication function

The main module of EC10/2L series small PLC has two integrated serial ports: PORT0 and PORT1, the main module of EC10A series small PLC has one integrated serial port: PORT0, and the main module of EC10V and EC20H series small PLC has three integrated serial ports: PORT0, PORT1 and PORT2. The extension modules including 485 communication module are also available for the communication in a fieldbus network.

Three serial ports are compatible with Modbus, N:N and user-defined free port protocols.

1.3.1 Modbus protocol network

The main module can set up a RS485 Modbus network with multiple inverters, PLCs and other intelligent devices through the RS485 port on PORT1 and PORT2, or through PORT0 and a RS232/485 converter. The maximum communication distance is 1200 meters and the maximum baud rate is 115200bit/s. RTU and ASCII transmission modes are optional.

The main module can communicate one-to-one with inverters, PLCs, touch screens and instruments through the RS232 port on PORT0 and PORT1. The maximum communication distance is 15 meters and the maximum baud rate is 115200bit/s.

For details about the Modbus network, see **Error! Reference source not found.Error! Reference source not found.** and *Appendix 7Modbus communication protocol (EC10, EC20 series)*.

1.3.2 N:N protocol network

EC10/EC10V/EC20/EC20H series PLC is embedded with COMPANY-developed N:N communication protocol, capable of setting up an N:N communication network through the RS485 port on PORT1 and PORT2, or through PORT0 and a RS232/485 converter.

The N:N communication protocol allows single/double-layer networking and data exchange among 2~32 PLCs with the maximum baud rate of 115200bps.

For details about the N:N network, see **Error! Reference source not found.Error! Reference source not found.**.

1.3.3 Free port protocol network

The free port protocol allows communication with customized data format and supports ASCII and binary system. In this communication mode, the PLC can communicate with various equipment with customized formats, such as inverter, barcode scanner, instrument and other intelligent devices. PLC can communicate with a single device in the RS232 or RS485 mode, or form a RS485 network when there are multiple devices.

For details about the free port protocol communication, see **Error! Reference source not found.Error! Reference source not found.**.

1.4 Documents of EC series small PLC

You can download the documents of EC series small PLC at www.Company.com. If you need the printed copy, please contact your agent.

1.4.1 Selection manual

EC10 Selection Manual

EC20 Technical Manual

EC20H Selection Manual

1.4.2 User manual of main module

EC10 series
Quick Start User Manual of EC10 Series PLC
User Manual of EC10 Series PLC

EC20 series
Quick Start User Manual of EC20 Series PLC
User Manual of EC20 Series PLC

EC10A series
Quick Start User Manual of EC10A Series PLC
User Manual of EC10A Series PLC

EC20H series
Quick Start User Manual of EC20H Series PLC
User Manual of EC20H Series PLC

EC10V series
Quick Start User Manual of EC10V Series PLC
User Manual of EC10V Series PLC

1.4.3 Programming manual

Programming Manual of EC Series Small PLC

1.4.4 User manual of programming software

User Manual of Programmer Programming Software

1.4.5 User manual of I/O extension module

EC10 series
User Manual of EC10 Series Passive I/O Extension Module

EC20 series
User Manual of EC20 Series Passive I/O Extension Module
User Manual of EC20 Series Active I/O Extension Module

1.4.6 User manual of special module

EC10 series
User Manual of EC10-4AD Analog Input module
User Manual of EC10-4DA Analog Output module

User Manual of EC10-4PT RTD Input Module
User Manual of EC10-4TC Thermalcouple Input Module

1.4.7 User manual of communication module

User Manual of ECS-EPM Communication Module
User Manual of EC20-RS485 Communication Module

EC20 series
User Manual of EC20-4AD Analog Input module
User Manual of EC20-4AM Analog Input/Output module
User Manual of EC20-4DA Analog Output module
User Manual of EC20-4PT RTD Input Module
User Manual of EC20-4TC RTD Input Module
User Manual of EC20-8AD Analog Input module
User Manual of EC20-8TC Thermalcouple Input Module

Chapter 2 Function description

This chapter introduces the programming resources, theories and system configuration of EC series PLC as well as how to set PLC running and operation modes. The system commissioning functions and commissioning software are also introduced.

Programming resources and theories

2.1.1 Programming resources

Table 2-1 EC10 programming resources

Name		Specification and description
I/O configuration	Max. I/O	172 (theoretical)
	Qty. of extension modules	The sum of I/O extension modules and special modules is no more than 7
User file capacity	Program capacity	16k steps
	Datablock capacity	8000 D elements
Instruction speed	Basic instruction	0.3μs/instruction
	Application instruction	Several μs/instruction~several hundred μs/instruction
Instruction number	Basic instruction	32
	Application instruction	226
Element resource[Note7]	Input/output	128 I/128 O (input: X0~X177, output: Y0~Y177)[Note1]
	Auxiliary relay	2048 (M0~M2047)
	Local auxiliary relay	64 (LM0~LM63)
	Special auxiliary relay	256 (SM0~SM255)
	State relay	1024 (S0~S1023)
	Timer	256 (T0~T255)[Note2]
	Counter	256 (C0~C255)[Note3]
	Data register	8000 (D0~D7999)
	Local data register	64 (V0~V63)
	Indexed addressing register	16 (Z0~Z15)
	Special data register	256 (SD0~SD255)
Interrupt resource	External input interrupt	16 (triggering edge is user configurable, corresponding to the rising&falling edge of terminals X0~X7)
	High-speed counter interrupt	6
	Internal timer interrupt	3
	Serial port interrupt	8
	PTO output interrupt	2
	Power loss interrupt	1
Communication function	Communication port	2 asynchronous serial communication ports. Port0: RS232. Port1: RS232 or RS485
	Communication protocol	Modbus, freeport and N:N protocols; capable of setting up 1:N and N:N communication networks

Name		Specification and description	
Special function	High-speed counter	X0, X1	Single input: 50kHz. Total frequency (X0~X5): no more than 80kHz
		X2~X5	Single input: 10kHz
	High-speed pulse output	Y0, Y1	100kHz 2 independent outputs (only for transistor outputs)
	Digital filtering	X0~X7 adopt digital filtering and other terminals adopt hardware filtering	
	Analog potentiometer[Note4]	2	
	Calling of subprograms	Maximum number: 64. Maximum nesting levels: 6. Local variables and variable alias are supported. Each subprogram can provide up to 16 parameter transfer	
	User program protection	Upload password	3 kinds of password. Not longer than 8 letters or numbers. Case sensitive
		Download password	
		Monitor password	
		Subprogram password	Not longer than 16 letters or numbers. Case sensitive.
		Other protections	Formatting and uploading ban enabled
	Programming mode[Note5]	Programmer programming software[Note6]	IBM PC or compatible computer is required
	Realtime clock	Built-in, 100h of working time after power failure (the main module must have worked for more than 2mins before the power failure)	

Table 2-2 EC10A programming resources

Name		Specification and description
I/O configuration	Max. I/O	60
	Qty. of extension modules	No
User file capacity	Program capacity	6k steps
	Datablock capacity	4000 D elements
Instruction speed	Basic instruction	0.3μs/instruction
	Application instruction	Several μs/instruction~several hundred μs/instruction
Instruction number	Basic instruction	32
	Application instruction	200
Element resource[Note7]	Input/output	128 I/128 O (input: X0~X177, output: Y0~Y177)[Note1]
	Auxiliary relay	1024 (M0~M1023)
	Local auxiliary relay	64 (LM0~LM63)
	Special auxiliary relay	256 (SM0~SM255)
	State relay	1024 (S0~S1023)
	Timer	256 (T0~T255)[Note2]
	Counter	256 (C0~C255)[Note3]
	Data register	4000 (D0~D3999)
	Local data register	64 (V0~V63)
	Indexed addressing register	16 (Z0~Z15)
	Special data register	256 (SD0~SD255)
Interrupt resource	External input interrupt	16 (triggering edge is user configurable, corresponding to the rising&falling edge of terminals X0~X7)
	High-speed counter interrupt	6
	Internal timer interrupt	3
	Serial port interrupt	4

Name		Specification and description	
	PTO output interrupt	2	
	Power loss interrupt	1	
Communication function	Communication port	1 asynchronous serial communication port. Port0: RS232	
	Communication protocol	Modbus and freeport protocols	
Special function	High-speed counter	X0, X1	Single input: 50kHz. Total frequency (X0~X5): no more than 80kHz
		X2~X5	Single input: 10kHz
	High-speed pulse output	Y0, Y1	50kHz 2 independent outputs (only for transistor outputs)
	Digital filtering	X0~X7 adopt digital filtering and other terminals adopt hardware filtering	
	Analog potentiometer[Note4]	2	
	Calling of subprograms	Maximum number: 64. Maximum nesting levels: 6. Local variables and variable alias are supported. Each subprogram can provide up to 16 parameter transfer	
	User program protection	Upload password	3 kinds of password. Not longer than 8 letters or numbers. Case sensitive
		Download password	
		Monitor password	
		Subprogram password	Not longer than 16 letters or numbers. Case sensitive.
		Other protections	Formatting and uploading ban enabled
	Programming mode[Note5]	Programmer programming software[Note6]	IBM PC or compatible computer is required
	Realtime clock	Built-in, 100h of working time after power failure (the main module must have worked for more than 2mins before the power failure)	

Table 2-3 EC10V programming resources

Name		Specification and description
I/O configuration	Max. I/O	172 (theoretical)
	Qty. of extension modules	The sum of I/O extension modules and special modules is no more than 7
User file capacity	Program capacity	16k steps
	Datablock capacity	8000 D elements
Instruction speed	Basic instruction	0.2μs/instruction
	Application instruction	Several μs/instruction~several hundred μs/instruction
Instruction number	Basic instruction	32
	Application instruction	234
Element resource[Note7]	Input/output	128 I/128 O (input: X0~X177, output: Y0~Y177)[Note1]
	Auxiliary relay	2048 (M0~M2047)
	Local auxiliary relay	64 (LM0~LM63)
	Special auxiliary relay	512 (SM0~SM511)
	State relay	1024 (S0~S1023)
	Timer	256 (T0~T255)[Note2]
	Counter	256 (C0~C255)[Note3]
	Data register	8000 (D0~D7999)
	Local data register	64 (V0~V63)
	Indexed addressing register	16 (Z0~Z15)
	Special data register	512 (SD0~SD512)

Name		Specification and description	
Interrupt resource	External input interrupt	16 (triggering edge is user configurable, corresponding to the rising&falling edge of terminals X0~X7)	
	High-speed counter interrupt	6	
	Internal timer interrupt	3	
	Serial port interrupt	12	
	PTO output interrupt	4	
	Power loss interrupt	1	
Communication function	Communication port	3 asynchronous serial communication ports. Port0: RS232. Port1: RS485 . Port2: RS485	
	Communication protocol	Modbus, freeport and N:N protocols; capable of setting up 1:N and N:N communication networks	
Special function	High-speed counter	X0, X1	Single input: 100kHz. Total frequency (X0~X5): no more than 200kHz
		X2~X5	Single input: 10kHz
	High-speed pulse output	Y0, Y1	100kHz 2 independent outputs (only for transistor outputs)
		Y2, Y3	10kHz 2 independent outputs (only for transistor outputs)
	Digital filtering	X0~X7 adopt digital filtering and other terminals adopt hardware filtering	
	Analog potentiometer[Note4]	without	
	Calling of subprograms	Maximum number: 64. Maximum nesting levels: 6. Local variables and variable alias are supported. Each subprogram can provide up to 16 parameter transfer	
	User program protection	Upload password	3 kinds of password. Not longer than 8 letters or numbers. Case sensitive
		Download password	
		Monitor password	
		Subprogram password	Not longer than 16 letters or numbers. Case sensitive.
		Other protections	Formatting and uploading ban enabled
	Programming mode[Note5]	Programmer programming software[Note6]	IBM PC or compatible computer is required
	Realtime clock	Built-in, the standby battery supplies power	

Table 2-4 EC20 programming resources

Name		Specification and description
I/O configuration	Max. I/O	512 (256 I/256 O)
	Qty. of extension modules	8, the sum of special modules is no more than 8
User file capacity	Program capacity	12k steps
	Datablock capacity	8000 D elements
Instruction speed	Basic instruction	0.09μs/instruction
	Application instruction	5μs/instruction~280μs/instruction
Instruction number	Basic instruction	32
	Application instruction	221
Element resource[Note7]	Input/output	256 I/256 O(input: X0~X377, output: Y0~Y377)[Note1]
	Auxiliary relay	2000 (M0~M1999)
	Local auxiliary relay	64 (LM0~LM63)
	Special auxiliary relay	256 (SM0~SM255)
	State relay	992 (S0~S991)
	Timer	256 (T0~T255)[Note2]
	Counter	256 (C0~C255)[Note3]
	Data register	8000 (D0~D7999)
	Local data register	64 (V0~V63)
	Indexed addressing register	16 (Z0~Z15)

Name		Specification and description	
	Special data register	256 (SD0~SD255)	
Interrupt resource	External input interrupt	16 (triggering edge is user configurable, corresponding to the rising&falling edge of terminals X0~X7)	
	High-speed counter interrupt	6	
	Internal timer interrupt	3	
	PTO output interrupt	2	
	Serial port interrupt	12	
	Power loss interrupt	1	
Communication function	Communication port	2 asynchronous serial communication ports. Port0: RS232. Port1: RS232 or RS485. Port2 (external 485 communication module): RS422 or RS485	
	Communication protocol	Modbus and freeport protocols; capable of setting up 1:N communication network	
Special function	High-speed counter	X0, X1	Single input: 50kHz. Total frequency (X0~X5): no more than 80kHz
		X2~X5	Single input: 10kHz
	High-speed pulse output	Y0, Y1	100kHz 2 independent outputs (only for transistor outputs)
	Digital filtering	X0~X17 adopt digital filtering and other terminals adopt hardware filtering	
	Analog potentiometer[Note4]	2	
	Calling of subprograms	Maximum number: 64. Maximum nesting levels: 6. Local variables and variable alias are supported. Each subprogram can provide up to 16 parameter transfer	
	User program protection	Upload password	3 kinds of password. Not longer than 8 letters or numbers. Case sensitive
		Download password	
		Monitor password	
	Programming mode[Note5]	Programmer programming software[Note6]	IBM PC or compatible computer is required
	Realtime clock	Built-in, standby batteries supply power	

Table 2-4 EC20H programming resources

Name		Specification and description
I/O configuration	Max. I/O	512 (256 I/256 O)
	Qty. of extension modules	8 modules, the sum of special modules is no more than 8
User file capacity	Program capacity	32k steps
	Datablock capacity	8000 D elements, 32K R elements
Instruction speed	Basic instruction	0.065μs/instruction
	Application instruction	Several μs/instruction~several hundred μs/instruction
Instruction number	Basic instruction	32
	Application instruction	286
Element resource[Note7]	Input/output	256 I/256 O(input: X0~X377, output: Y0~Y377)[Note1]
	Auxiliary relay	10240 (M0~M1999)
	Local auxiliary relay	64 (LM0~LM63)
	Special auxiliary relay	512 (SM0~SM511)
	State relay	4096 (S0~S4095)
	Timer	512 (T0~T511)[Note2]
	Counter	262 (C0~C306)[Note3]
	Data register	40768 (D0~D7999, R0~R32767)
	Local data register	64 (V0~V63)
	Indexed addressing register	16 (Z0~Z15)
	Special data register	512 (SD0~SD511)
Interrupt resource	External input interrupt	16 (triggering edge is user configurable, corresponding to the rising&falling edge of terminals X0~X7)
	High-speed counter interrupt	8
	Internal timer interrupt	3

Name		Specification and description	
	Serial port interrupt	12	
	PTO output interrupt	6	
	Power loss interrupt	1	
	Interpolation interrupt	3	
	Passed position interrupt	6	
Communication function	Communication port	3 asynchronous serial communication ports. Port0: RS232. Port1: RS485. Port2 (external 485 communication module): RS422 or RS485	
	Communication protocol	Modbus, freeport and N:N protocols; capable of setting up 1:N and N:N communication networks	
Special function	High-speed counter	X0~X7, 8×100kHz	
	High-speed pulse output	Y0~Y7	4×200kHz, 4×100kHz
	Digital filtering	X0~X7 adopt digital filtering and other terminals adopt hardware filtering	
	Calling of subprograms	Maximum number: 64. Maximum nesting levels: 6. Local variables and variable alias are supported. Each subprogram can provide up to 16 parameter transfer	
	User program protection	Upload password	3 kinds of password. Not longer than 8 letters or numbers. Case sensitive
		Download password	
		Monitor password	
	Programming mode[Note5]	Programmer programming software[Note6]	IBM PC or compatible computer is required
	Realtime clock	Built-in, standby batteries supply power	

Notes:

Note 1: X and Y elements are addressed in octal system. For example, X10 stands for the eighth input point.

Note 2: Based on the timing precision, T element addresses fall into three categories:

EC10/EC10A/EC10V/EC20

1) 100ms: T0~T209

2) 10ms: T210~T251

3) 1ms: T252~T255

EC20H

1) 100ms: T0~T209

2) 10ms: T210~T479

3) 1ms: T480~T511

Note 3: Based on the width and function of count value, C element addresses fall into three categories:

EC10/EC10A/EC20

1) 16bit up counter: C0~C199

2) 32bit up/down counter: C200~C235

3) 32bit high-speed counter: C236~C255

EC20H

1) 16bit up counter: C0~C199

2) 32bit up/down counter: C200~C235

3) 32bit high-speed counter: C236~C255, C301-C307, C256-C300 reserved

Note 4: The analog potentiometer is an instrument that you can use to set the PLC element value. You can use a philips screwdriver to wind the potentiometer clockwise to the maximum angle of 270°, and the element value will be set from 0 to 255. Note that the potentiometer could be damaged if you wind it clockwise more than 270°.

Note 5: The element values can be forcedly set to facilitate commissioning and analyzing user program and streamline the commissioning. You can force up to 128 bit elements and 16 word elements at the same time.

Note 6: The user program can be modified online.

Note 7: Partial PLC elements are reserved. Avoid using those elements in the user program. For details, see *Appendix 3Reserved* elements.

2.1.2 PLC running mechanism (scan cycle model)

EC series PLC main module runs according to the scan cycle model.

The system cyclically executes the following four tasks one by one: user program execution, communication, internal tasks and I/O update. Each round is called a scan cycle.

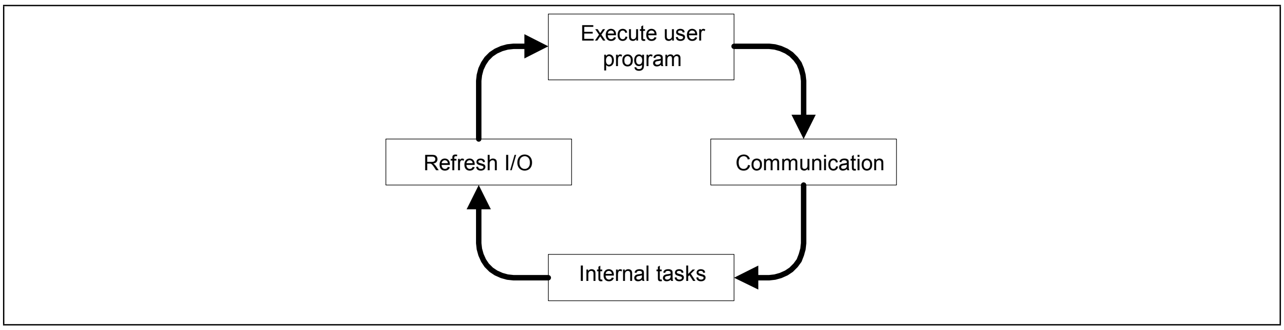


Figure 2-1 PLC running mechanism

- **User program execution**

The system will execute user program instructions one by one from the beginning till the main program ending instruction.

- **Communication**

Communicate with the programming software to receive and respond to the instructions such as download, run and stop.

- **Internal tasks**

Processing various system internal tasks, such as refreshing panel indicators, updating software timer, refreshing special auxiliary relays and special data registers.

- **I/O update**

The I/O update includes two stages: input update and output update.

Output update: open or close the output terminal based on the value of the corresponding Y element (ON or OFF).

Input update: convert the ON or OFF state of input terminals to the value of the corresponding X element (ON or OFF).

2.1.3 Watchdog function for user program execution

The watchdog function enables the system to monitor the user program execution time during every scan cycle, and stop the user program if the running time exceeds the preset limit. You can set the watchdog time in the **Set time** tab after double clicking the **System block** in Programmer main interface.

2.1.4 Constant scan mode

In the constant scan mode, every scan cycle takes the same time. You can set the constant scanning time in the **Set time** tab after double clicking the **System block** in Programmer main interface. By default, the **Constant scanning time setting** is zero, which means no constant scan. The actual scan cycle will prevail when the actual scan cycle is bigger than the constant scan cycle.

Note

The **constant scanning time setting** must not be set bigger than the **watchdog time setting**.

2.1.5 User file download and storage

You can download a user file to the main module to control the main module.

The user file includes user program, datablock, system block and auxiliary user information. The auxiliary user information includes the user program variable list and the source file of user data.

You can select to download the user program, datablock or system block. Whatever you select, the corresponding auxiliary user information will always be downloaded.

For EC20 series PLC, the downloaded user program, datablock and system block will be stored permanently in the main module EEPROM area, while the downloaded auxiliary user information will be stored in the battery backed RAM area.

For EC10 series PLC, all user files will be stored permanently in the main module FLASH area.

For EC20H series PLC, the downloaded user program, datablock and system block will be stored permanently in the main module FLASH and EEPROM areas, while the downloaded auxiliary user information will be stored in the battery backed RAM area.

🕮 **Note**

1. To embed the downloaded files into the main module, the main module power supply must be maintained for more than 30s after the download.
2. If the backup battery fails in EC20 and EC20H series PLC, the auxiliary user information will be lost, the annotation for the user program will not be uploaded, and system will report “User information file error”. But the user program will be executed after all.

2.1.6 Initialization of elements

When the PLC changes from STOP to RUN, it will initialize its elements according to battery backed data, EEPROM data, datablock and element value. The priorities of various data are listed in the following table.

Table 2-5 PLC data initialization priorities

Data type	Power OFF→ON	STOP→RUN
Battery backed data	Highest	Highest
EEPROM data	High	High
Datablock (precondition: the **Datablock enabled** is checked in the **Advanced Settings** tab of **System block**)	Mid	Mid
Element value (precondition: the **Element value retained** is checked in the **Advanced Settings** tab of **System block**)	-	Low

2.1.7 Saving data at power off

■ **Preconditions**

Upon power loss, the system will stop the user program and save the element in the specified saving range to the battery backed files.

■ **Element restore after power on**

If the battery backed files are correct, the PLC elements will restore their saved values after power on.

The elements outside of the saving range will be set to zero.

If the battery backed files are lost or incorrect, the system will set all elements to zero.

■ **Setting saving range**

You can set the element range in the **Saving Range** tab of **System block. See** 0 and the following example.

EC10/10V series PLC supports only one group of saving range.

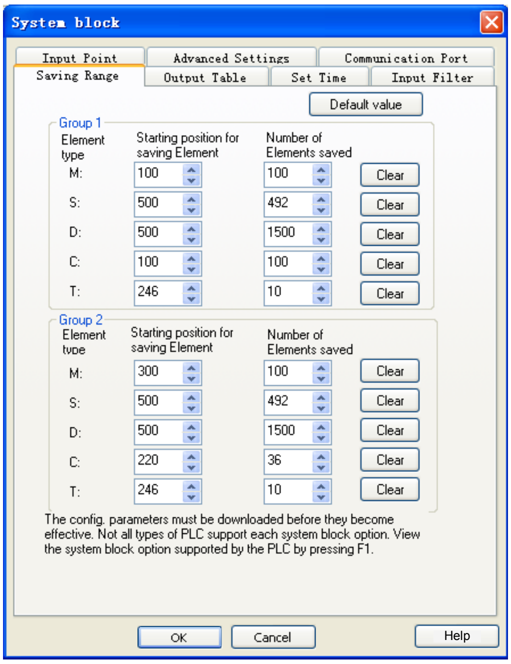
EC20 and EC20H series PLC supports two saving groups that form a union.

Example (EC20):

Set M100~M200 as the saving range in **Group 1**.

Set M300~M400 as the saving range in **Group 2**.

In effect, both M100~M200 and M300~M400 are set as the saving range.



Group 1

Element type	Starting position for saving Element	Number of Elements saved
M:	100	100
S:	500	492
D:	500	1500
C:	100	100
T:	246	10

Group 2

Element tvpe	Starting position for saving Element	Number of Elements saved
M:	300	100
S:	500	492
D:	500	1500
C:	220	36
T:	246	10

The config. parameters must be downloaded before they become effective. Not all types of PLC support each system block option. View the system block option supported by the PLC by pressing F1.

Figure 2-2 Setting saving range

Note

1. The power-off data saving function in EC20 and EC20H series PLC relies on the support of the backup battery. If batteries fail, all the saved elements will have uncertain values after power loss.
2. For EC10 series PLC, the values of its saved elements are stored in the permanent memory.

2.1.8 Permanent storage of D element data

You can use the EROMWR instruction in the user program to write the D element values (D6000~D6999) to the permanent memory EEPROM in EC10 series PLC. The EEPROM operation will make the scan cycle 2ms~5ms longer. The written data will overwrite the existing data in EEPROM.

Note

The EEPROM can be over-written for a limited number of times (usually one million). Do not overwrite EEPROM unless it is necessary, otherwise EEPROM could fail soon and lead to CPU fault.

2.1.9 Digital filtering of input terminals

The input terminals X0~X17 of EC20 series main module and X0~X7 of EC10、EC10V and EC20H series main module use digital filtering to filter the noise at the terminal. You can set the filter constant in the **Input Filter** tab of **System block**.

2.1.10 No battery mode

EC10V、EC20 and EC20H series main module can work without battery. When you select the **No battery mode** in the **Advanced Settings** tab of **System block**, the system will not report system errors caused by lack of battery (battery-backed data lost, forced-table lost and user information file error).

See the notice for the **No battery mode** in the **Advanced Settings** tab of **Datablock**.

Note

EC10 series PLC has no battery, therefore it does not support no battery mode.

2.1.11 User program protection

EC10, EC10V, EC20 and EC20H series PLCs provide mutiple levels of passwords and other protection measures.

Table 2-6 User program protection

Protection measures	Description
Formatting ban	After downloading system block to the PLC and checking the **Formatting is prohibited** option in the

Protection measures	Description
	Advanced Settings tab in **System block**, the PLC internal user program, system block and datablock are protected against formatting. To lift the formatting ban, you need to re-download the system block and uncheck the **Formatting is prohibited** option.
Download password	Download limit
Upload ban	If you select to disable the upload function during downloading process, it will be prohibited to upload the program from PLC to PC. To enable the upload function, you must re-download the program and check to enable the upload function during the downloading process.
Upload password	Upload limit
Monitor password	Download limit
Program password	The programmer can set passwords to protect the program, subprogram and interrupt subprogram against aunthrorized accessing and editing in Programmer. Password setting method: Right click the program and select **Encrypt/Decrypt** in the popped out shortcut menu, insert the password and confirm it. To cancel the password, just go through the same process and input the correct password.

📖 **Note**

If you fail to input the correct password for continuously 5 times, you will be banned from inputting password for the next 5 minutes.

2.2 System configuration

2.2.1 System block

The PLC configuration information, or system block file, is configured through the system block and is an important part of the PLC user file. Before using the PLC, you need to compile and download the system block file.

The system block configuration includes configuring the following items:

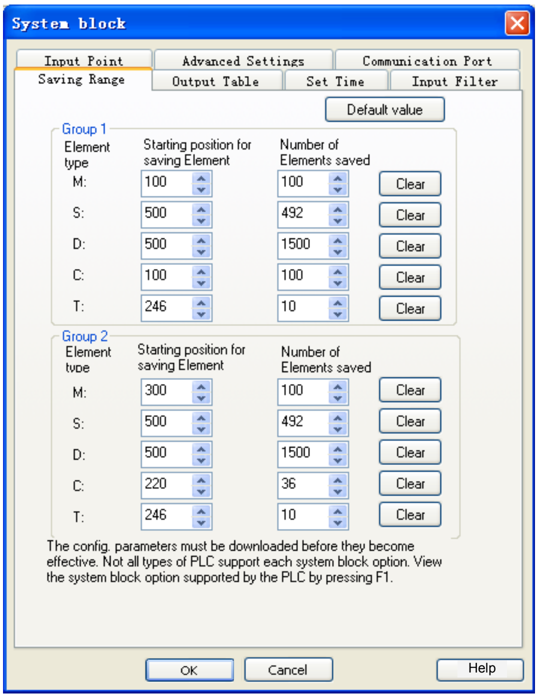
- **Saving range** (element saving range)
- **Set time** (watchdog time, constant scanning time and power loss detection time setting)
- **Input point** (startup mode of input point)
- **Communication port** (communication port and protocol setting)
- **Priority level of interruption**
- **Inverter configuration**
- **Output table**
- **Input filter**
- **Advanced settings** (datablock, element value retain, no battery mode and formatting ban)
- **Special module configuration**
- **Communication module**

After setting the system block, you can select **PLC**-> **Compile All** to compile the system block file and be ready for download.

■ **Saving range**

Upon power loss, EC10, EC10V, EC20 and EC20H series PLCs can save the data of elements in the preset saving range to SRAM, so as to use them after the power on.

You can set the saving range in the **Saving Range** tab, as shown in 0.



Group 1

Element type	Starting position for saving Element	Number of Elements saved	
M:	100	100	Clear
S:	500	492	Clear
D:	500	1500	Clear
C:	100	100	Clear
T:	246	10	Clear

Group 2

Element type	Starting position for saving Element	Number of Elements saved	
M:	300	100	Clear
S:	500	492	Clear
D:	500	1500	Clear
C:	220	36	Clear
T:	246	10	Clear

The config. parameters must be downloaded before they become effective. Not all types of PLC support each system block option. View the system block option supported by the PLC by pressing F1.

Figure 2-3 Setting element saving range

Note

The element range and group number of the saving range are different for different PLC models.

By default, the D, M, S, T and C elements in a certain range will be saved.

You can change the defaults as you need. By clicking the **Clear** button on the right will set the corresponding number to zero.

For EC20 and EC20H series PLC, you can set two groups that form a union.

For EC10 and EC10V series PLC, you can set only one group.

Note

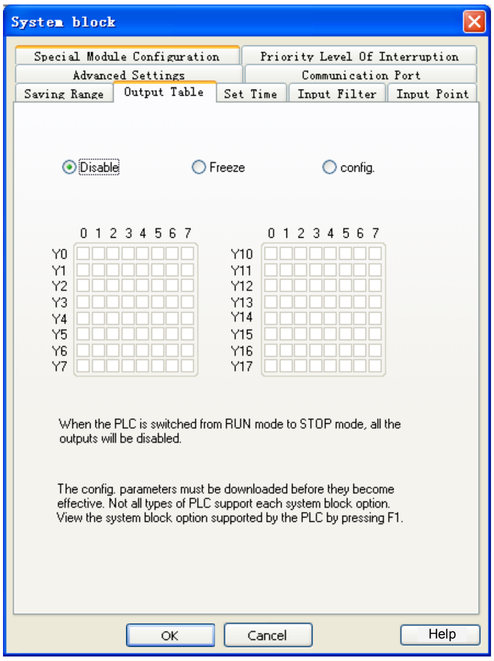
The T elements cannot be set in the saving range for EC10/10V series PLC.

System operation upon power loss: PLC will save the elements in the saving range to the battery backed files.

System operation upon power on: PLC will check the data in SRAM. If the data saved in SRAM is correct, it will remain unchanged. If the data is incorrect, PLC will clear all the elements in SRAM.

- **Output table**

In the **Output Table** tab, you can set the state of output points when the PLC is in STOP state. See Figure 2-4.



● Disable ○ Freeze ○ config.

	0	1	2	3	4	5	6	7
Y0	□	□	□	□	□	□	□	□
Y1	□	□	□	□	□	□	□	□
Y2	□	□	□	□	□	□	□	□
Y3	□	□	□	□	□	□	□	□
Y4	□	□	□	□	□	□	□	□
Y5	□	□	□	□	□	□	□	□
Y6	□	□	□	□	□	□	□	□
Y7	□	□	□	□	□	□	□	□

	0	1	2	3	4	5	6	7
Y10	□	□	□	□	□	□	□	□
Y11	□	□	□	□	□	□	□	□
Y12	□	□	□	□	□	□	□	□
Y13	□	□	□	□	□	□	□	□
Y14	□	□	□	□	□	□	□	□
Y15	□	□	□	□	□	□	□	□
Y16	□	□	□	□	□	□	□	□
Y17	□	□	□	□	□	□	□	□

When the PLC is switched from RUN mode to STOP mode, all the outputs will be disabled.

The config. parameters must be downloaded before they become effective. Not all types of PLC support each system block option. View the system block option supported by the PLC by pressing F1.

Figure 2-4 Setting output table

The output table is used to set the PLC output state when the PLC is stopped. The output states include:

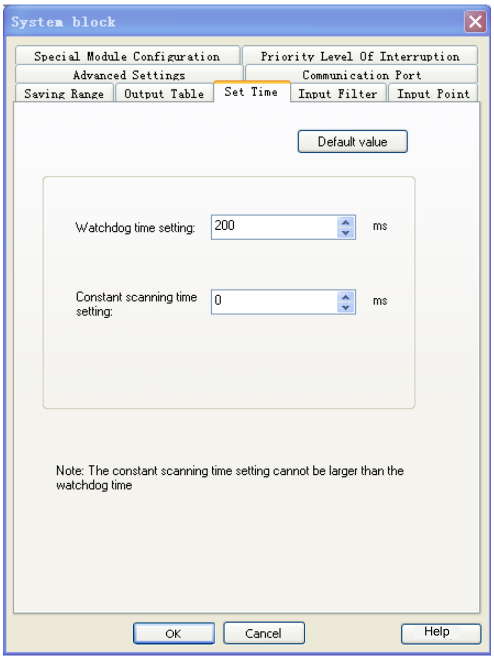
(1) Disable: When the PLC is stopped, all the outputs will be disabled.

(2) Freeze: When the PLC is stopped, all the outputs will be frozen at the last status.

(3) Configure: When the PLC is stopped, the marked outputs will be set as ON.

- **Set time**

See Figure 2-5.



Watchdog time setting: 200 ms

Constant scanning time setting: 0 ms

Note: The constant scanning time setting cannot be larger than the watchdog time

Figure 2-5 Setting time

1. Watchdog time setting

The watchdog time is the maximum user program execution time. When the actual program execution time exceeds the watchdog time, PLC will stop the execution, the ERR indicator (red) will turn on, and the system will output according to the system configuration. The watchdog time setting range is 0ms~1000ms. Default: 200ms.

2. Constant scanning time setting

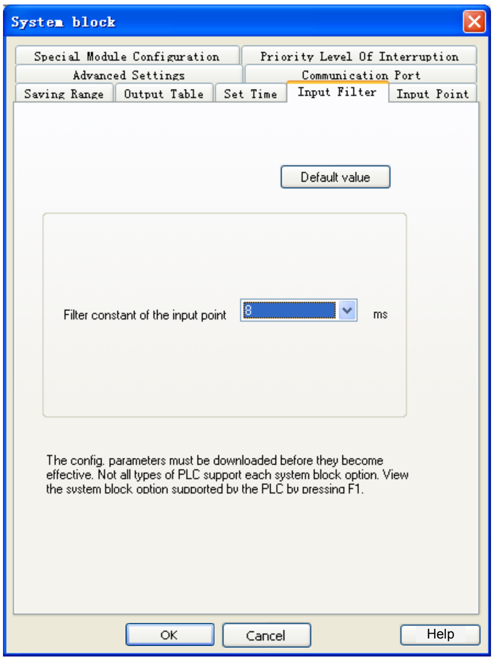
With the constant scanning time set, system will scan the registers within a constant duration. Setting range: 0ms~1000ms. Default: 0ms.

3. Power loss detection time setting (for EC20 and EC20H only)

When the duration of power loss exceeds the power loss detection time, the PLC will change to STOP. The system will save the values of elements in the Saving Range. Setting range: 0ms~100ms. Default: 0ms

■ **Input filter**

In the **Input Filter** tab, you can set the filter constant for a PLC input terminal. The digital filter can eliminate the noise at the input terminal. Only input terminals X0~X17 (for EC10 and EC20H series: X0~X7) use digital filter, while other digital input terminals use hardware filter. EC10 input filter can be in grouped (divided into X0~X3, X4~X7) and the filter constant is 0, 2, 4, 8, 16, 32 and 64; EC20H input filter can be grouped (divided into X0~X3, X4~X7) and the filter constant can be continuously set in 0~64ms; EC20 input filter cannot be grouped and the filter constant can be continuously set in 0~64ms. See Figure 2-6 EC10 input filter setting.



Filter constant of the input point 8 ms

The config. parameters must be downloaded before they become effective. Not all types of PLC support each system block option. View the system block option supported by the PLC by pressing F1.

Figure 2-6 Setting input filter

■ **Input point**

The **Input Point** setting tab is shown in Figure 2-7.

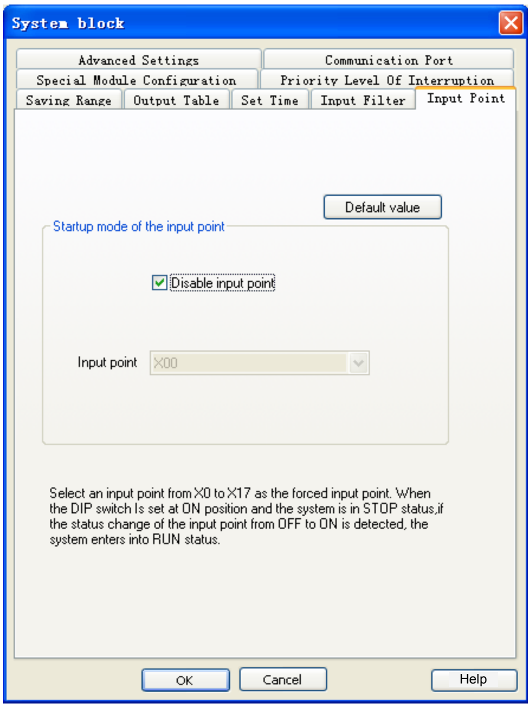
In this tab, you can set the following parameters:

1. Input point

When the **Disable input point** is not checked, you can designate an input terminal (among X0~X17) as a means of external RUN control. When the designated input terminal is ON, the PLC will be turned from STOP state to RUN state.

2. Disable input point

Check the **Disable input point** to disable the input point startup function.



Startup mode of the input point

☑ Disable input point

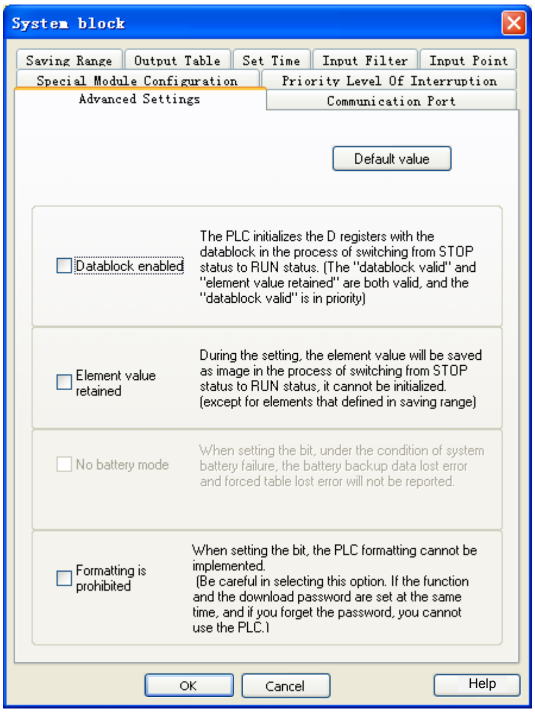
Input point X00

Select an input point from X0 to X17 as the forced input point. When the DIP switch Is set at ON position and the system is in STOP status,if the status change of the input point from OFF to ON is detected, the system enters into RUN status.

Figure 2-7 Setting input point

■ Advanced settings

The advanced settings include datablock enabled, element value retained and no battery mode.



□ Datablock enabled — The PLC initializes the D registers with the datablock in the process of switching from STOP status to RUN status. (The "datablock valid" and "element value retained" are both valid, and the "datablock valid" is in priority)

□ Element value retained — During the setting, the element value will be saved as image in the process of switching from STOP status to RUN status, it cannot be initialized. (except for elements that defined in saving range)

□ No battery mode — When setting the bit, under the condition of system battery failure, the battery backup data lost error and forced table lost error will not be reported.

□ Formatting is prohibited — When setting the bit, the PLC formatting cannot be implemented. (Be careful in selecting this option. If the function and the download password are set at the same time, and if you forget the password, you cannot use the PLC.)

Figure 2-8 Advanced settings

1. Datablock enabled

Check the Datablock enabled, and the datablock will be used to initialize the D elements when the PLC changes from STOP to RUN.

2. Element value retained

Check the Element value retained, and the elements will not be initialized, but saved when the PLC changes from STOP to RUN.

📖 **Note**

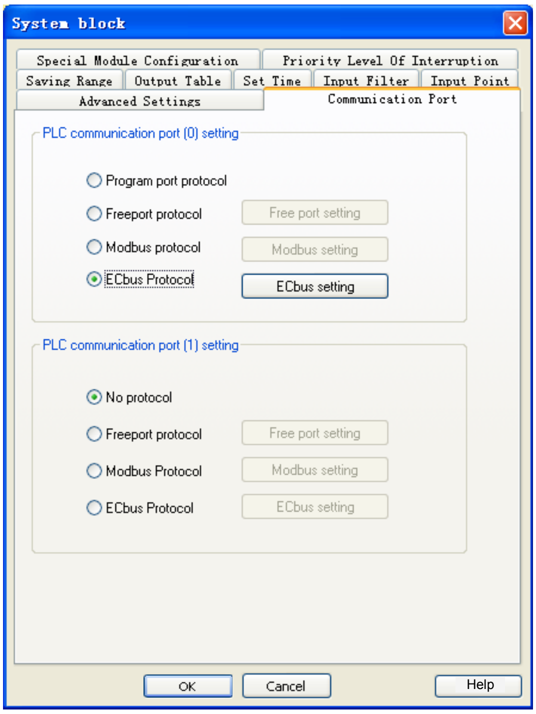
When the **Datablock enabled** and **Element value retained** are both checked, the **Datablock enabled** prevails. See *2.1.6Initialization of elements*.

3. No battery mode

Check this option, and the system will not report the battery backup data lost error and forced table lost error upon battery failure.

■ **Communication port**

You can set the two or three PLC communication ports in the **Communication port** tab of the **System block,** as shown in Figure 2-9. The setting items include protocol selection and the specific protocol parameters.



System block

PLC communication port (0) setting

○ Program port protocol

○ Freeport protocol

○ Modbus protocol

● ECbus Protocol

PLC communication port (1) setting

● No protocol

○ Freeport protocol

○ Modbus Protocol

○ ECbus Protocol

Figure 2-9 Setting communication ports

By default, the communication port 0 uses program port protocol, while the communication port 1 and 2 use no protocol. You can set as you need.

1. Program port protocol

By default, the communication port 0 uses the program port protocol, the dedicated protocol for the communication of EC series PLC programming software. Under this protocol, you can set the communication baud rate between PC and port 0 through the serial port configuration tool of AutoStation. In the TM state, port 0 can only be used for programming communication.

2. Free port protocol

The free port protocol supports customized data file format, either ASCII or binary code. Only in the RUN state can a PLC use the free port communication, which cannot be used to communicate with the programming device. In the STOP state, port 0 can only be used for programming communication.

The configurable parameters include baud rate, data bit, parity check, stop bit, allow start character detection, allow end character detection, intercharacter timeout and interframe timeout.

3. Modbus protocol

The Modbus communication equipment include a master and a slave. The master can communicate with the slave (including inverters) and send control frames to the slave, and the slave will respond to the master’s requests.

Communication port 0 can be set as a slave, while communication port 1 can be set as a slave or a master.

The configurable parameters include baud rate, data bit, parity check, stop bit, master/slave mode, station No., transmission mode, timeout time of the main mode and retry times.

4. N:N bus protocol

N:N bus is an COMPANY-developed communication protocol that supports N to N communication in a small PLC network. The PLCs in a N:N bus network can automatically exchange part of their D and M elements.

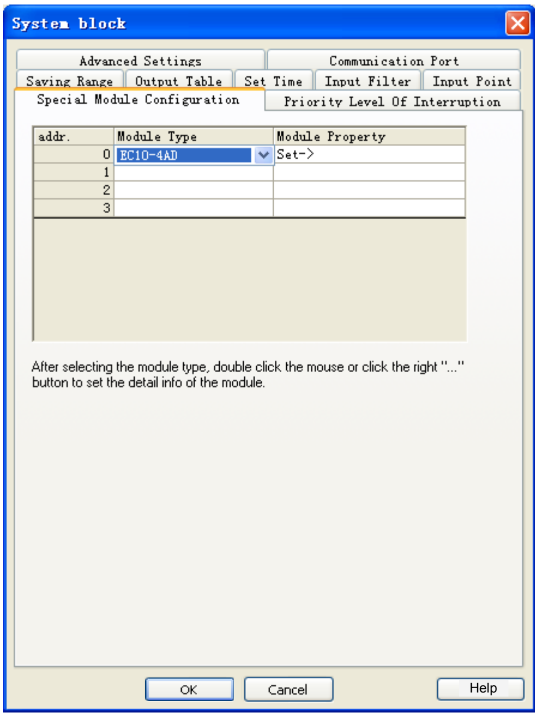
Port 0, port1 and port 2 can use N:N bus protocol.

Note

For the detailed information of communication protocols, see *Chapter 10 Using Communication Function.*

■ **Special module configuration**

You can set the **Module type** and **Module property** in the **Special module configuration** tab, as shown in **Error! Reference source not found.**.



addr.	Module Type	Module Property
0	EC10-4AD	Set->
1		
2		
3		

After selecting the module type, double click the mouse or click the right "..." button to set the detail info of the module.

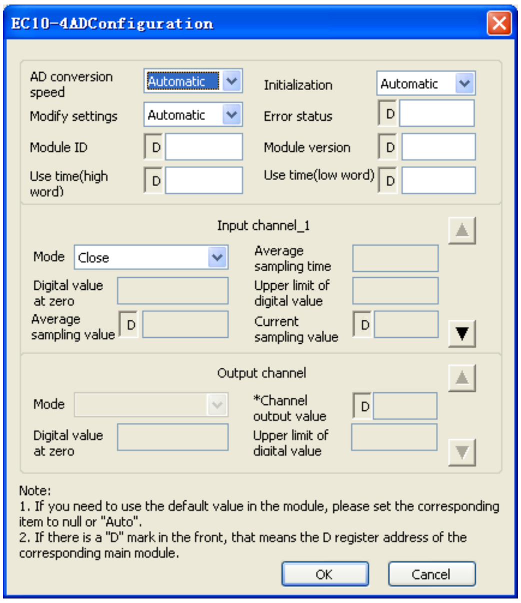
Figure 2-10 Special module configuration

1. Module type

As shown in **Error! Reference source not found.**, you can set the module type for No.0~No.3 special modules.

2. Module property

After selecting the **Module type**, the corresponding **Module property** will be activated. Open the dialogue box as shown below.



AD conversion speed: Automatic
Initialization: Automatic
Modify settings: Automatic
Error status: D
Module ID: D
Module version: D
Use time(high word): D
Use time(low word): D

Input channel_1

Mode: Close
Average sampling time
Digital value at zero
Upper limit of digital value
Average sampling value: D
Current sampling value: D

Output channel

Mode
*Channel output value: D
Digital value at zero
Upper limit of digital value

Note:
1. If you need to use the default value in the module, please set the corresponding item to null or "Auto".
2. If there is a "D" mark in the front, that means the D register address of the corresponding main module.

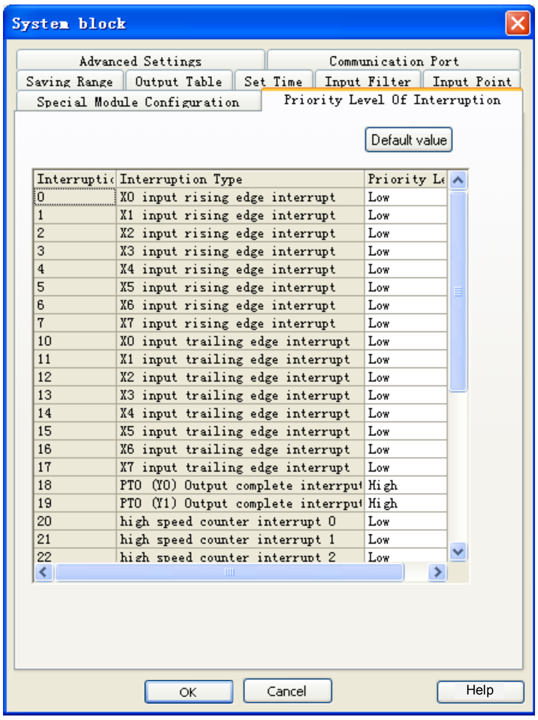
Figure 2-11 Setting special module property

In the dialogue box as shown in **Error! Reference source not found.**, you can configure the channel for the special module, including mode (signal features), digital value at zero, upper limit of digital value and average sampling value. Refer to the user manual of the specific special module for the meanings and configuration methods of the various parameters.

- **Priority level of interruption**

The priority level of interruption is shown in Figure 2-12.

The PLC built-in interrupts can be set as high priority or low priority.

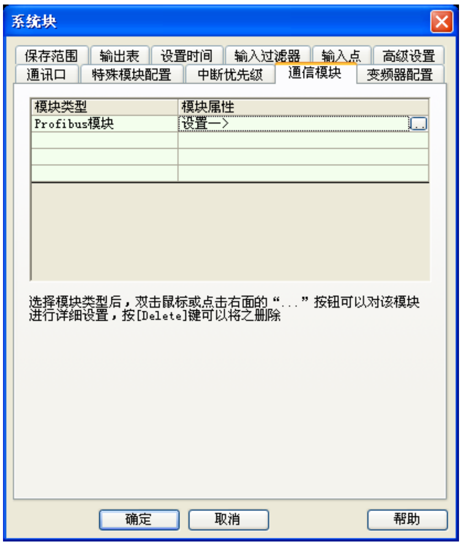


Interruptic	Interruption Type	Priority Le
0	X0 input rising edge interrupt	Low
1	X1 input rising edge interrupt	Low
2	X2 input rising edge interrupt	Low
3	X3 input rising edge interrupt	Low
4	X4 input rising edge interrupt	Low
5	X5 input rising edge interrupt	Low
6	X6 input rising edge interrupt	Low
7	X7 input rising edge interrupt	Low
10	X0 input trailing edge interrupt	Low
11	X1 input trailing edge interrupt	Low
12	X2 input trailing edge interrupt	Low
13	X3 input trailing edge interrupt	Low
14	X4 input trailing edge interrupt	Low
15	X5 input trailing edge interrupt	Low
16	X6 input trailing edge interrupt	Low
17	X7 input trailing edge interrupt	Low
18	PTO (Y0) Output complete interrupt	High
19	PTO (Y1) Output complete interrupt	High
20	high speed counter interrupt 0	Low
21	high speed counter interrupt 1	Low
22	high speed counter interrupt 2	Low

Figure 2-12 Setting interrupt priority

- **Communication module**

You can set the **Communication module**, as shown in Figure 2-13.

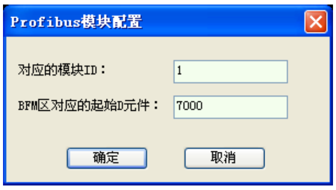


模块类型	模块属性
Profibus模块	设置—>

选择模块类型后，双击鼠标或点击右面的“...”按钮可以对该模块进行详细设置，按[Delete]键可以将之删除

Figure 2-13 Setting communication module

The following dialog box will pop up by clicking Setting:



Profibus模块配置

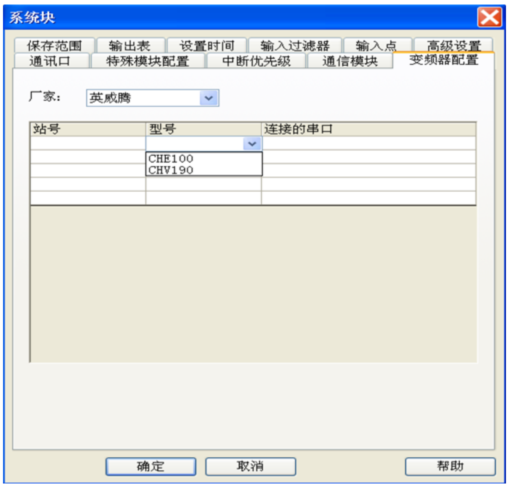
对应的模块ID： 1

BFM区对应的起始D元件： 7000

Figure 2-14 Profibus module configuration

- **Inverter configuration**

You can select the inverter model and set the station number, as shown below:



系统块

厂家： 英威腾

站号	型号	连接的串口
	CHE100 CHV190	

Figure 2-15 Inverter configuration

2.2.2 Datablock

The datablock is used to set the defaults for D elements. If you download the compiled datablock settings to the PLC, the PLC will use the datablock to initialize the related D elements upon PLC startup.

The datablock editor enables you to assign initial data to the D register (data memory). You can assign data to words or double words, but not to bytes. You can also add comments by inputting “//” to the front of a character string.

Besides the datablock of D elements, EC20H series support the datablock of R elements.

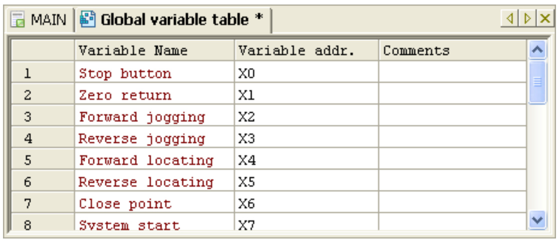
See *Programmer Programming Software User Manual* for detailed datablock instruction.

2.2.3 Global variable table

The global variables table enables you to give meaningful names for certain PLC addresses. The names are accessible anywhere in the project, and using them is in effect using the corresponding device. The global variable table includes three columns: variable name, variable addr. and comments.

The variable name can be made up of letters (case insensitive), numbers, underline or their mixture, but no spaces. The name cannot start with a number, nor be completely made up of numbers. Length: not longer than 8 bytes. The format of “device type + number” is illegal. No keywords shall be used. The keywords include: basic data type, instructions and the operators in the IL programming language.

For EC20H/EC20/EC10 series small PLC, the uploading number the global variables allow shall not exceed 1000/500/140. If beyond the number, the variables can be only saved at local. See Figure 2-16.



	Variable Name	Variable addr.	Comments
1	Stop button	X0	
2	Zero return	X1	
3	Forward jogging	X2	
4	Reverse jogging	X3	
5	Forward locating	X4	
6	Reverse locating	X5	
7	Close point	X6	
8	System start	X7	

Figure 2-16 Global variable table

2.2.4 Setting BFM for EC20 and EC20H series special modules

There is no need to set the addresses for EC20 and EC20H series special modules, for the main module can detect and address them automatically upon power on.

Among the special modules, the analog extension module includes the analog input module and analog output module.

The parameters of these two special modules, such as the channel characteristics, zero point and maximum digital signal are by default applicable directly. However, when necessary, you can change the parameters in order to cater for your actual needs.

- **EC20 and EC20H analog input module**

EC20 and EC20H analog input module exchanges information with its main module through the BFM area.

When a user program runs on the main module, the TO instruction will write data to the related registers in the BFM area of EC20 special module, and change the default settings. The configuration data that can be changed includes zero digital signal, maximum digital signal, input channel signal characteristic, input channel ready flag, and so on.

The main module uses the FROM instruction to read the data from the BFM area of EC20 analog input module. The data may include the analog-digital conversion result and other information.

- **EC20 and EC20H analog output module**

EC20 and EC20H analog output module exchanges information with its main module through the BFM area.

When a user program runs on the main module, the TO instruction will write data to the related registers in the BFM area of EC20 special module, and change the default settings. The configuration data that can be changed includes zero digital signal, maximum digital signal, output channel signal characteristic, output channel ready flag, and so on.

The main module uses the FROM instruction to read the data from, and uses the TO instruction to write the digital signal to be convertered to, the BFM area of EC20 analog output module.

For details about the TO/FROM instruction, refer to **Error! Reference source not found.***Application instructions*. As for the information about various special modules, as well as their BFM areas, see the quick start manuals of the special module.

2.3 Running mode and state control

You can start or stop the PLC in any of the following three ways.

1. Using the mode selection switch
2. Using the designated terminals by setting the startup mode of input point and external terminal in system block
3. Using the programming software by setting the mode selection switch at TM or ON

2.3.1 System RUN and system STOP states

The main module states include RUN and STOP states.

- **RUN**

When the main module is in the RUN state, the PLC will execute the user program. That is to say, all the four tasks in a scan cycle, namely the user program execution, communication, internal tasks and I/O update, will be executed.

- **STOP**

When the main module is in the STOP state, the PLC will not execute the user program, but will still execute the other three tasks in every scan cycle, namely the communication, internal tasks and I/O update.

2.3.2 RUN&STOP state change

- **How to change from STOP to RUN**

1. Resetting the PLC

If the mode selection switch is set to ON, reset the PLC (including power-on reset), and the system will enter the RUN state automatically.

📖 **Note**

If the **Control mode of input point** is valid in the main module, the corresponding input terminal must be ON, or the system will not enter the RUN state after reset.

2. Setting mode selection switch

When the PLC is in STOP state, setting the mode selection switch to ON will change the PLC to RUN state.

3. Setting startup mode of input point

If the **Startup mode of input point** is valid in the system block, in STOP state, the designated input points (X0~X17) detected by the system change from OFF to ON, and then the main module enter the RUN state.

Note

The mode selection switch must be set to ON for the input terminal startup mode to be valid.

■ **How to change from RUN to STOP**

1. Resetting the PLC

If the mode selection switch is set to OFF or TM, resetting the system (including power-on reset) will change the PLC to STOP state.

Note

Even when the mode selection switch is ON, the system will also enter the STOP state after reset if the **Control mode of input point** is valid in the main module and the designated input point is OFF.

2. Setting mode selection switch

The system will change from RUN to STOP when you set the mode selection switch from ON or TM to OFF.

3. Using the STOP command

The system will enter the STOP state after executing the STOP command in the user program.

4. Auto-stop upon faults

The system will stop executing the user program when a serious fault (like user program error, or user program execution overtime) is detected.

2.3.3 Setting output in STOP state

You can set the state of output terminals (Y) when the PLC is stopped. The three optional settings include:

1. Disable: When the PLC is stopped, all output terminals will be OFF.

2. Freeze: When the PLC is stopped, all the output terminals will be frozen at the last status.

3. Configure: You can decide which output will be ON and which will be OFF when the PLC is stopped according to the actual need.

You can find the above settings in the **Output Table** tab of the **System block**. See the *Output Table* in **Error! Reference source not found.Error! Reference source not found.**.

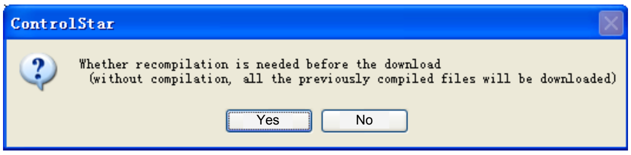
2.4 System debugging

2.4.1 Uploading&downloading program

■ **Downloading**

The system block, data block and user program edited in Programmer can be downloaded to the PLC through a serial port. Note that the PLC should be in the STOP state when downloading.

If you change a compiled program and want to download it, the system will ask you to compile it again, as shown in Figure 2-17.



ControlStar

Whether recompilation is needed before the download
(without compilation, all the previously compiled files will be downloaded)

Figure 2-17 Re-compile prompt

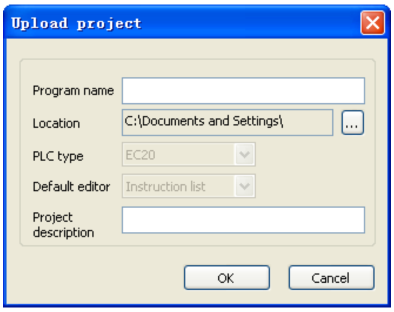
Note

If you select No, the program compiled last time will be downloaded to the PLC, which means the changes are invalid.

If you have set a download password and have not entered it after starting the Programmer this time, a window asking you to enter the password will pop up before the download can start.

■ **Uploading**

You can upload the system block, data block and user program from a PLC to your PC, and save them in a new project. If the battery backed data are valid, the user auxiliary information files will be uploaded together. See Figure 2-18.



Program name:

Location: C:\Documents and Settings\

PLC type: EC20

Default editor: Instruction list

Project description:

Figure 2-18 Upload dialog box

If you have set a upload password and have not entered it after starting the Programmer this time, a window asking you to enter the password will pop up before the upload can start.

During the download, you can select to disable the upload function, which means no PC can upload the program from the PLC. To enable the upload function, you must re-download the program and check to enable the upload function during the downloading process.

2.4.2 Error reporting mechanism

The system can detect and report two types of errors: system error and user program execution error.

A system error is caused by abnormal system operation while a user program execution error is caused by the abnormal execution of the user program.

Every error is assigned with a code. See *Appendix 6System error code.*

- **System error**

When system error occurs, the system will set the special relay SM3, and write the error code into the special data register SD3. You can obtain the system error information by accessing the error code stored in SD3.

If multiple system errors occur at the same time, the system will only write the code of the worst error into SD3.

When serious system errors occur, the user program will halt, and the ERR indicator on the main module will turn on.

- **User program execution error**

When user program execution error occurs, the system will set the special relay SM20, and write the error code into the special data register SD20.

If the next application instruction is correctly executed, the SM20 will be reset, while SD20 will still keep the error code.

The system keeps the codes of the lastest five errors in special data registers SD20~SD24 and form a stack.

If the code of the current error is different from the code in SD20, the error stack will be pushed down, as shown in Figure 2-19.

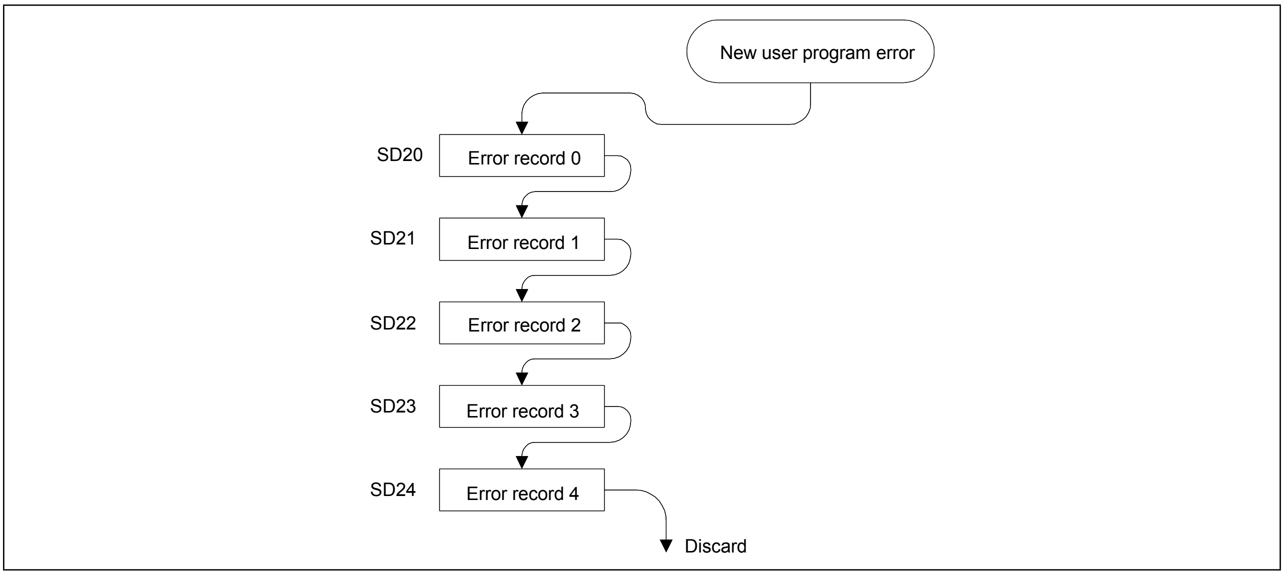


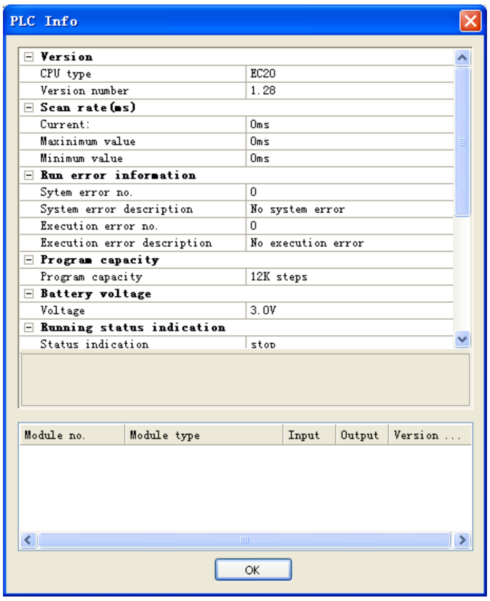
Figure 2-19 Push operation of the error stack

Only when serious user program execution error occurs will the user program halt and the ERR indicator on the main module turn on. In less serious cases, the ERR indicator on the main module will not turn on.

- **Checking the error information online**

Connect the PLC with your PC through the serial port, and you can read various PLC state information through the Programmer, including the system error and user program execution error.

In the main interface of Programmer, click **PLC**->**PLC Info** to check the PLC information, as shown below:



PLC Info

⊟ **Version**	
CPU type	EC20
Version number	1.28
⊟ **Scan rate(ms)**	
Current:	0ms
Maximinum value	0ms
Minimum value	0ms
⊟ **Run error information**	
Sytem error no.	0
System error description	No system error
Execution error no.	0
Execution error description	No execution error
⊟ **Program capacity**	
Program capacity	12K steps
⊟ **Battery voltage**	
Voltage	3.0V
⊟ **Running status indication**	
Status indication	stop

Module no.	Module type	Input	Output	Version ...

Figure 2-20 PLC information

The **System error No.** is the No. of the system errors stored in SD3, and **Execution error No.** is the No. of the execution error stored in SD20. The error description is for your reference.

2.4.3 Editing user program online

You can use the online editing function when you want to change the user program without stopping the PLC.

Warning

On occasions when casualties or property loss may occur, the online program editing function should be used by professionals with sufficient protection measures.

- **Method**

After making sure that the PC-PLC communication has been set up and the PLC is in RUN state, click **Debug**->**Online edit** in the Programmer main interface to enter the online edit state.

In the online edit state, you can edit the main program, subprograms and interrupts as usual. After the edit, click **PLC**->**Download** and the edited program will be compiled and downloaded to the PLC automatically. When the download completes, the PLC will execute the new program.

- **Limits**

1. In the online edit state, you cannot change the global variable table or any local variable table, nor add or delete any subprogram and interrupt.

2. Programmer will quit the online edit state if the PLC is stopped.

2.4.4 Clearing and formatting

You can use the clearing operation to clear PLC element value, PLC program and PLC datablock. While through formatting, you can clear all PLC internal data and program.

- **PLC element value clear**

The PLC element value clear function can clear all element values when the PLC is in STOP state.

Think it twice before using the clearing function, because clearing PLC element values may cause PLC operation error or loss of working data.

- **PLC program clear**

The PLC program clear function can clear the PLC user program when the PLC is in STOP state.

Think it twice before using the clearing function, because after the PLC user program is cleared, the PLC will have no program to execute.

- **PLC datablock clear**

The PLC datablock clear function can clear all the PLC datablocks when the PLC is in STOP state.

Think it twice before using the clearing function, because after the PLC datablock is cleared, the PLC will not initialize element D according to the presetting of the datablock.

- **PLC format**

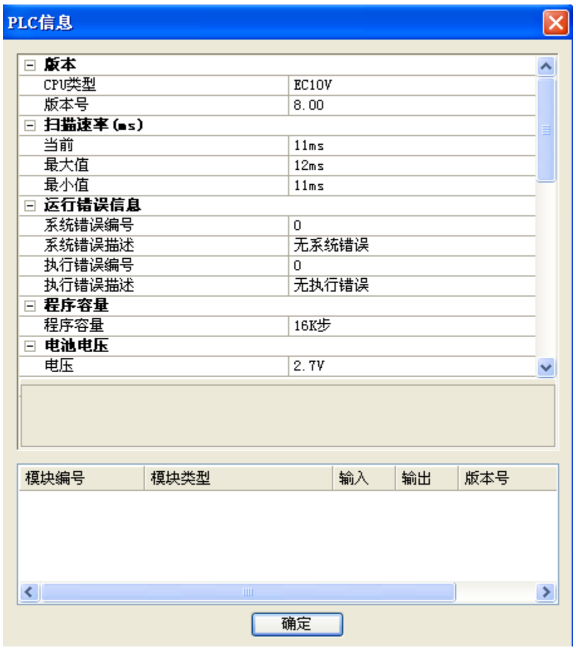
The PLC format function can format all PLC data, including clearing the user program, restoring the defaults, and clearing the datablock (when PLC is in STOP state).

Think it twice before using the formatting function, because this operation will clear all the downloads and settings in the PLC.

2.4.5 Checking PLC information online

- **PLC info**

The **PLC info** function can obtain and display various PLC running information, as shown in Figure 2-21.



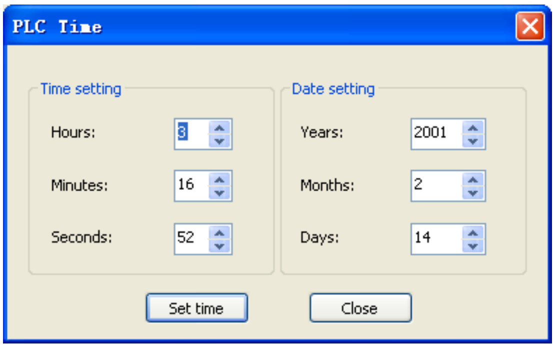
⊟ **版本**	
CPU类型	EC10V
版本号	8.00
⊟ **扫描速率(ms)**	
当前	11ms
最大值	12ms
最小值	11ms
⊟ **运行错误信息**	
系统错误编号	0
系统错误描述	无系统错误
执行错误编号	0
执行错误描述	无执行错误
⊟ **程序容量**	
程序容量	16K步
⊟ **电池电压**	
电压	2.7V

模块编号	模块类型	输入	输出	版本号

Figure 2-21 PLC current operation information

- **PLC time**

The **PLC time** function can be used to display and set PLC present time, as shown in Figure 2-22.



Time setting

Hours: 3

Minutes: 16

Seconds: 52

Date setting

Years: 2001

Months: 2

Days: 14

Figure 2-22 Setting PLC time

Displayed in the **PLC time** window is the present date and time of PLC. You can adjust the time setting and click the **Set time** button to validate it.

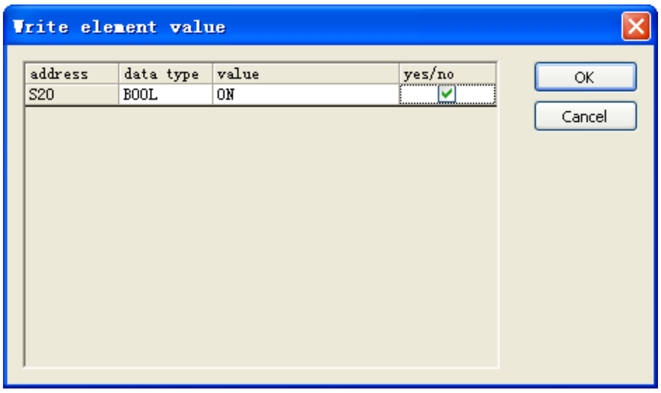
2.4.6 Write, force and element monitoring table

- **Write and force**

During the debugging, some element values may need to be changed manually. You can use the write or force function. Difference between write and force is that written element values are one-off and may change with the program operation, but forced element values will be permanently recorded in the PLC hardware until being unforced.

To use the write or force function, just select the element that needs changing, right click and select **Write selected element** or **Force selected element**. All the element addresses used by the selected element will be listed in the dialog box. Modify the address value to be written or forced, click the **OK** button, and the value will be downloaded to the PLC. If these values are effective in the hardware, you will see the change in later debugging process.

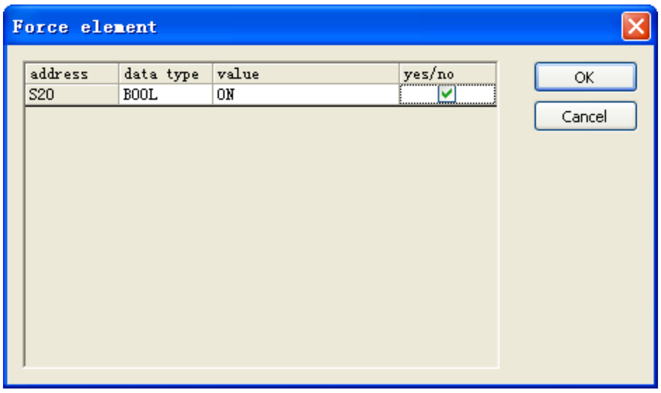
The **Write element value** dialogue box is shown in Figure 2-23:



address	data type	value	yes/no
S20	BOOL	ON	☑

Figure 2-23 Write element value

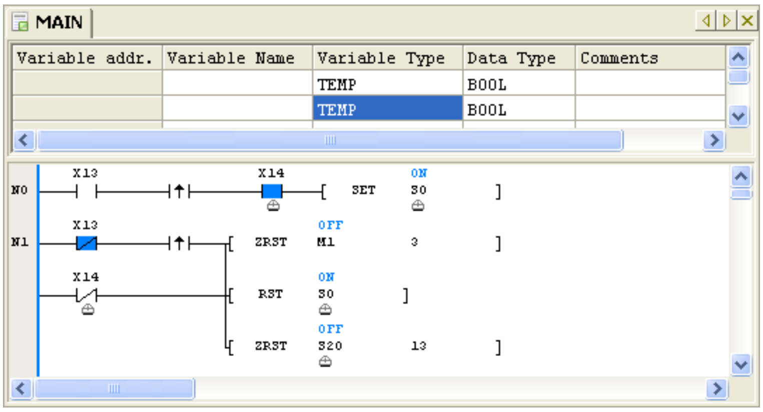
The **Force element** dialogue box is shown in Figure 2-24:



address	data type	value	yes/no
S20	BOOL	ON	☑

Figure 2-24 Force element

You can see a lock under the forced elements in the LAD, as shown in Figure 2-25:

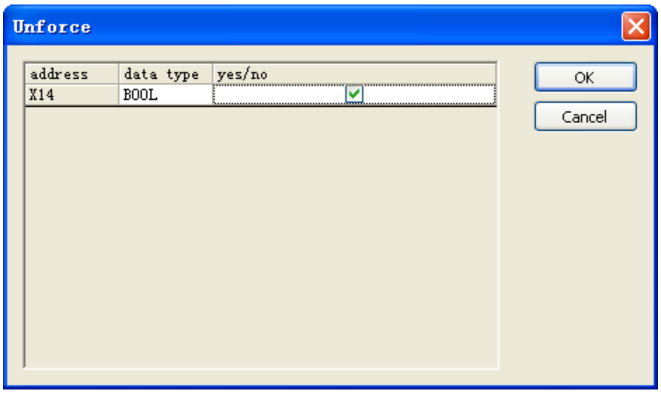


Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	

Figure 2-25 Lock signs under forced elements

- **Unforce**

You can unforce any forced elements when forcing them becomes unnecessary. To unforce an element, select the target element, right click and select **Unforce** to pop up a dialog box as shown in Figure 2-26. All the forced elements among the selected elements are listed in the dialog box. You can select to unforce any elements, and click the **OK** button to confirm. The forced value will be deleted from the PLC, so is the lock mark.



Unforce

address	data type	yes/no
X14	BOOL	☑

Figure 2-26 Unforce

- **Element monitoring table**

The element monitoring table (EMT) is responsible for monitoring the element value during the debugging. The program input and output elements can be added to the EMT so that they can be tracked after the program is downloaded to the PLC.

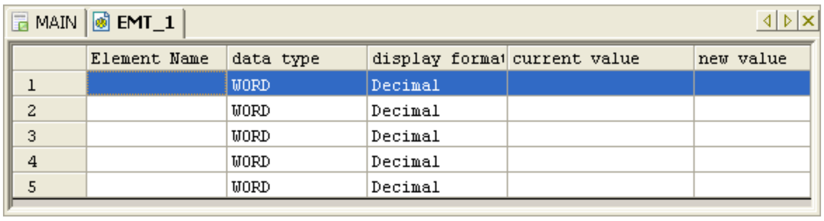
The EMT monitors the element value during the debugging. You can input the input & output elements, registers and word elements into the EMT during the debugging so that those elements can be monitored after the program is downloaded to PLC.

The EMT works in two modes: editing mode and monitoring mode. In the editing mode, no monitoring function can be carried out. In the monitoring mode, both the monitoring and editing functions are available.

In the monitoring mode, the displayed elements' values are updated automatically.

The EMT provides functions including editing, sequencing, searching, auto-updating of the current value, written value, forced value of the specified element or variable, and unforce.

See Figure 2-27 for the illustration of an EMT:



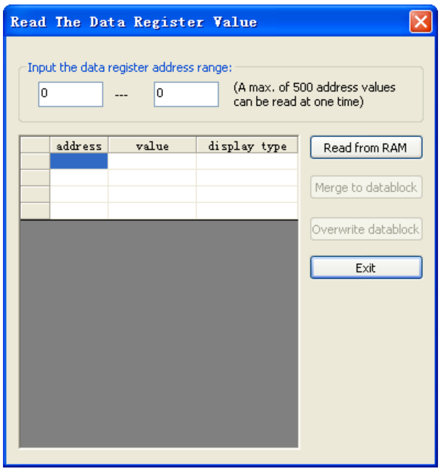
	Element Name	data type	display format	current value	new value
1		WORD	Decimal		
2		WORD	Decimal		
3		WORD	Decimal		
4		WORD	Decimal		
5		WORD	Decimal		

Figure 2-27 Element monitoring table

2.4.7 Generating datablock from RAM

This function can continuously read and display the value of up to 500 D registers in the PLC. The results can merge into the datablock or overwrite the original datablock.

Select **PLC->Generate datablock from RAM** to pop up a window as shown in Figure 2-28.



Read The Data Register Value

Input the data register address range:

0 --- 0 (A max. of 500 address values can be read at one time)

address	value	display type

Figure 2-28 Reading data register value

Enter the range of the datablock to be read, click the **Read from RAM** button, and the data will be read into the list after the instruction is correctly executed.

You can select hex, decimal or octal or binary system in the field of **Display type** to display the data.

After reading the data successfully, the buttons of **Merge to datablock** and **Overwrite datablock** are enabled. Clicking **Merge to datablock** will add the results after the current datablock. Clicking **Overwrite datablock** will replace the contents in the datablock with the generated results. After exiting the register value reading window, the software will prompt that the datablock has changed and the datablock window will be opened automatically.

Chapter 3 Element and data

This chapter details the description, classification and functions of the elements of EC series small PLC.

Element type and function

3.1.1 Element overview

The PLC elements are virtual elements configured in PLC system design in order to replace the actual relays in the relay control circuits. PLC uses the elements to calculate and configure system function. Due to their virtual nature, the elements can be used repeatedly in the program, their number is in theory unlimited (only related to program capacity), and have no mechanical or electric problems like their actual counterparts. Such features make the PLC much more reliable than relay control circuits. In addition, it is easier to program and modify the logic.

The types and functions of EC series PLC elements are shown in the following figure.

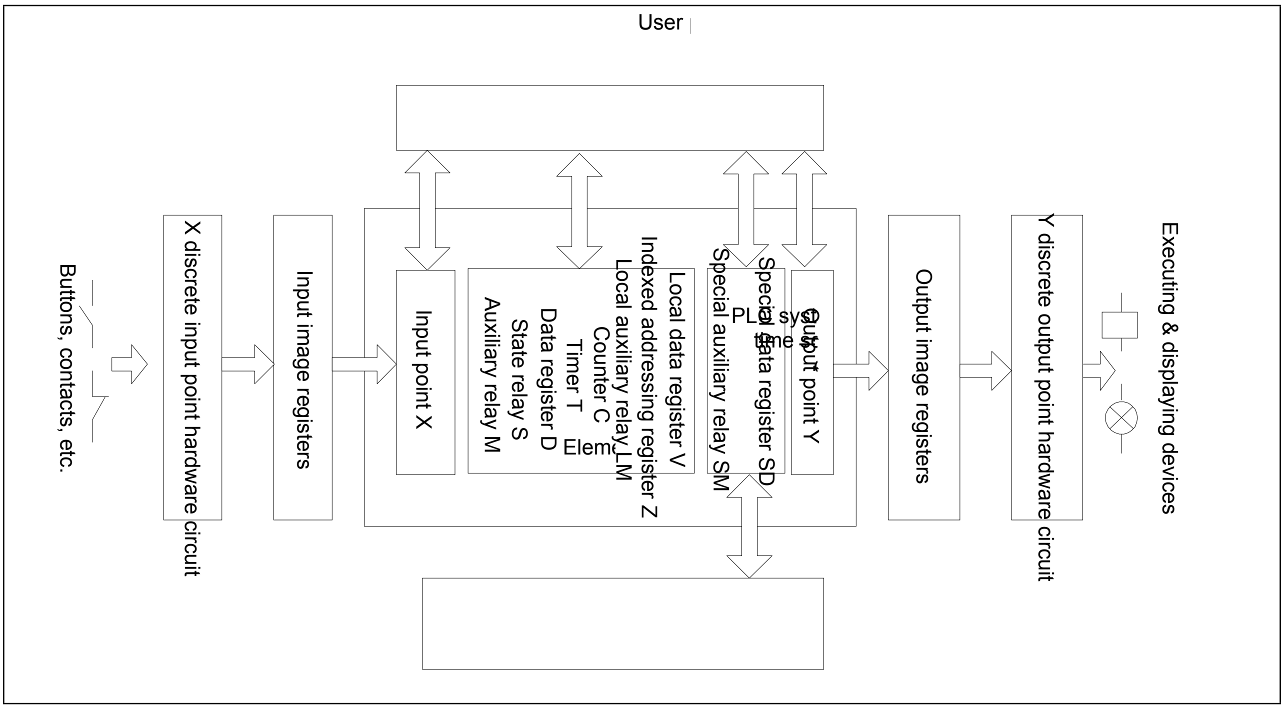


Figure3-1 Types and functions of PLC elements

In this manual, the elements are named according to their types. For example:

- Input point X, or “X element” for short
- Output point Y, or “Y element” for short
- Auxiliary relay M, or “M element” for short
- Data register D, or “D element” for short
- State relay S, or “S element” for short

3.1.2 Element list

The elements of EC series PLC are classified according to their functions, and are easily accessible.

The elements are listed in the following table.

Table 3-1 EC series PLC elements

		EC10 series	EC10V series	EC20 series	EC20H series	Numbered in
Element resources[Note4]	I/O	128 I/128 O(input X0~X177, output Y0~Y177)[Note1]	128 I/128 O(input X0~X177, output Y0~Y177)[Note1]	256 I/256 O (input X0~X377, output Y0~Y377)[Note1]	256 I/256 O (input X0~X377, output Y0~Y377)[Note1]	Octal
	Auxiliary relay	2048 (M0~M2047)	2048 (M0~M2047)	2000 (M0~M1999)	10240 (M0~M10239)	Decimal
	Local auxiliary relay[Note 5]	64 (LM0~LM63)	64 (LM0~LM63)	64 (LM0~LM63)	64 (LM0~LM63)	Decimal
	Special auxiliary relay	256 (SM0~SM255)	512 (SM0~SM511)	256 (SM0~SM255)	512 (SM0~SM511)	Decimal
	State relay	1024 (S0~S1023)	1024 (S0~S1023)	992 (S0~S991)	4096 (S0~S4095)	Decimal
	Timer	256 (T0~T255)[Note2]	256 (T0~T255)[Note2]	256 (T0~T255)[Note2]	512 (T0~T511)[Note2]	Decimal
	Counter	256 (C0~C255)[Note3]	256 (C0~C255)[Note3]	256 (C0~C255)[Note3]	307 (C0~C256)[Note3]	Decimal
	Data register	8000 (D0~D7999)	8000 (D0~D7999)	8000 (D0~D7999)	8000 (D0~D7999)	Decimal
	Data register R				32768 (R0~R32767)	Decimal
	Local data register[Note5]	64 (V0~V63)	64 (V0~V63)	64 (V0~V63)	64 (V0~V63)	Decimal
	Indexed addressing register	16 (Z0~Z15)	16 (Z0~Z15)	16 (Z0~Z15)	16 (Z0~Z15)	Decimal
	Special data register	256 (SD0~SD255)	512 (SD0~SD511)	256 (SD0~SD255)	512 (SD0~SD511)	Decimal

Notes:

1: The X and Y elements are addressed in octal system, and X10 represents the 8th input point. The I/O point number here is the system capacity while the actual system I/O point number is determined by the actual system configuration (including extension modules and power supply).

2: The T elements are addressed according to the timing precision:

- 100ms: T0~T209
- 10ms: T210~T251
- 1ms: T252~T255

(EC20H)

- 100ms: T0~T209
- 10ms: T210~T479
- 1ms: T480~T511

3: The C elements are addressed according to the counter types and functions:

- 16bit up counter: C0~C199
- 32bit up/down counter: C200~C235
- 32bit high-speed counter: C236~C255

(EC20H)

- 16bit up counter: C0~C199
- 32bit up/down counter: C200~C235
- 32bit high-speed counter: C236~C255, C301-C307, C256-C300 reserved

4: Part of PLC elements are reserved for internal tasks. Avoid using those elements in the user program. See *Appendix 3Reserved elements.*

5: These two elements are local variables that cannot be defined in the global variable table. When the user program calls subprograms or returns to the main program, they will be cleared, or be set through interface parameter transfer.

3.1.3 Input and output points

- **Element mnemonic**

- X (discrete input point)
- Y (discrete output point)

- **Function**

The X and Y elements represent respectively the input state of hardware X terminal and output state of hardware Y terminal.

The state of X elements is obtained through the input image register, while the state of Y elements is output through the output circuit driven by the output image register. The two operations are carried out in the I/O update stage of PLC scan cycle, as shown in 0. For details, see **Error! Reference source not found.Error! Reference source not found.**. It is obvious that there is a brief delay in PLC's response to the I/O. The delay is related to the input filter, communication, internal tasks and scan cycle.

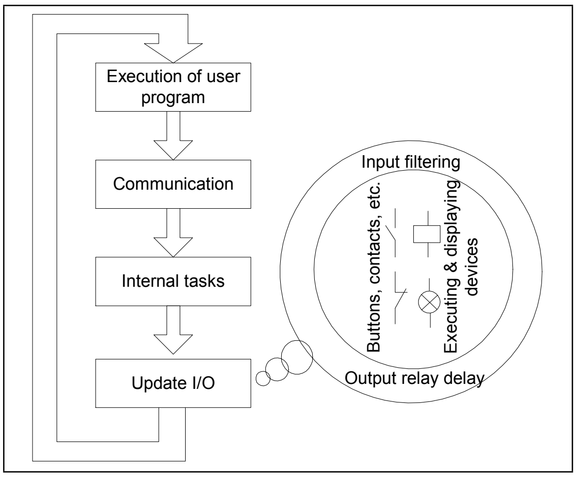


Figure 3-2 Schematic diagram of I/O update

- **Classification**

X0~X17 have digital filters whose filtering time can be set at the system block. Others use hardware filter. X0~X5 can be used as the counting input point for high-speed counters. Besides, X0~X7 can also be used for inputting external interrupts, pulse tracking and SPD frequency detecting instruction.

Y0 and Y1 can be used for high-speed output. Others are ordinary output points.

- **Elements numbered in**

Octal, starting with 0. The X and Y elements of both the main module and the I/O modules are numbered continuously. X elements are numbered in X0~X7, X10~X17 and X20~X27, etc. while Y elements are numbered in Y0~Y7, Y10~Y17 and Y20~Y27, etc.

- **Data type**

Boolean (both X and Y)

- **Available forms**

NO and NC contacts (dependent on which instruction uses it) The NO and NC contacts have opposite state values. They are sometimes referred to as “a” contact and “b” contact.

You can use NO and NC contacts of the Y element during programming.

- **Value assignment**

1. The X elements accepts only hardware input state value and forced operation state value. In the user program, they cannot be changed through output or instructions, nor be set during system debugging.

2. You can assign values to Y elements with the OUT instruction, or set the state value of Y elements, or even force or write Y element values during system debugging.

3. Through the system block, you can set the output states of Y elements in the STOP state.

3.1.4 Auxiliary relays

- **Element mnemonic**

M

- **Function**

The M state elements of discrete type are similar to the transfer relays in the actual electrical control circuits. You can use them to save various transit states in the user program.

- **Elements numbered in**

Decimal, starting with 0.

- **Data type**

Boolean

- **Available forms**

NO and NC contacts

- **Value assignment**

1. Through instructions.
2. Write or force during system debugging.

- **Power loss saving**

State	M elements in the saving range	M elements outside the saving range
Power loss	Remain unchanged	Cleared
RUN → STOP	Remain unchanged	Remain unchanged
STOP → RUN	Remain unchanged	Cleared
Note: The saving range is set through the system block. See ***Error! Reference source not found.Error! Reference source not found.***		

Note

When using the N:N bus protocol, some M elements will be used by the system.

3.1.5 State relays

- **Element mnemonic**

S

- **Alias**

Step flag

- **Function**

As the step flag, the S elements are used in the Sequential Function Chart (SFC). See ***Error! Reference source not found.Error! Reference source not found.***.

- **Classification**

S0~S19: initial step flag

Others: normal step flag

- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Boolean

- **Available forms**

1. Representation of steps (when used in STL instruction)
2. NO and NC contacts (when not used in STL instruction). Similar to M elements, the NO and NC contacts of S elements are available during programming.

- **Value assignment**

1. Through instructions.
2. Write or force during system debugging.

- **Power loss saving**

State	S elements in the saving range	S elements outside the saving range
Power loss	Remain unchanged	Cleared
RUN → STOP	Remain unchanged	Remain unchanged
STOP → RUN	Remain unchanged	Cleared
Note: The saving range is set through the system block. See ***Error! Reference source not found.Error! Reference source not found.***		

3.1.6 Timer

- **Element mnemonic**

T

- **Function**

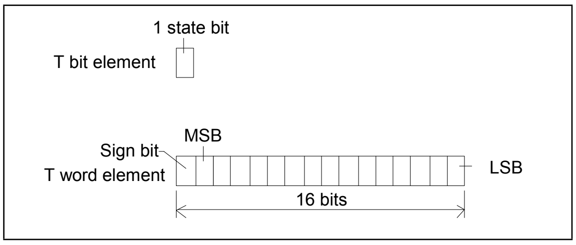


Figure 3-3 T element

- **Classification**

The T element contains a word element (2 bytes) and a bit element. The T word element can record a 16-bit value. The T bit element represents the timer coil state and is applicable to logic control.

According to the timing precision, the T elements are classified into three types:

T element	Timing precision
T0~T209	100ms
T210~T251	10ms
T252~T255	1ms

The T elements with the timing precision of 1ms are activated by interrupts, unrelated to the PLC scan cycle. Their action time is the most precise. The update and action time of other T elements are related to PLC scan cycles.

- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Boolean, word

- **Available forms**

The timing and action mode of T elements are determined by the timing instruction that uses them. There are four timing instructions: TON, TOF, TONR and TMON. See **Error! Reference source not found.Error! Reference source not found.** for details.

- **Value assignment**

1. Through instructions.
2. Write or force during system debugging.

- **Power loss saving**

State	T elements in the saving range (EC20 series only)	T elements outside the saving range
Power loss	Remain unchanged	Cleared
RUN → STOP	Remain unchanged	Remain unchanged
STOP → RUN	Remain unchanged	Cleared
Note: The saving range is set through the system block. See ***Error! Reference source not found.Error! Reference source not found.***.		

📖 **Note**

The maximum timing value of T element is 32767. The preset value is -32768~32767. Because T elements act only when the counted value reaches or exceeds the preset value, it is pointless setting the preset value as a negative number.

3.1.7 Counter

- **Element mnemonic**

C

- **Function**

The C element contains a bit element and a word (or a double word) element. The word elements can record 16-bit or 32-bit counted numbers, and is used as a value in the program. The bit element represents the state of the counter coil and is applied to logic control.

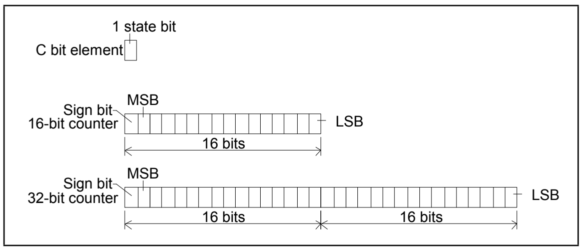


Figure 3-4 C element

- **Classification**

Two types: 16-bit counter and 32-bit counter

- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Boolean, word or double-word

- **Available forms**

The instructions that may use the C elements are classified into 4 types: CTU, CTR, DCNT and high-speed I/O. See **Error! Reference source not found.Error! Reference source not found.** and **Error! Reference source not found.***Application instructions* for details. The classification of C elements is shown below:

C element	Type	Applicable to
C0~C199	16bit up counter	CTU, CTR
C200~C235	32bit up/down counter	DCNT
C236~C255	32bit high-speed counter	High-speed I/O

- **Value assignment**

1. Through instructions.
2. Write or force during system debugging.

- **Power loss saving**

State	C elements in the saving range	C elements outside the saving range
Power loss	Remain unchanged	Cleared
RUN → STOP	Remain unchanged	Remain unchanged
STOP → RUN	Remain unchanged	Cleared
Note: The saving range is set through the system block. See ***Error! Reference source not found.Error! Reference source not found.***.		

3.1.8 Data register

- **Element mnemonic**

D, R

- **Function**

As a data element, the D or R elements are used in many calculation and control instructions as the operands.

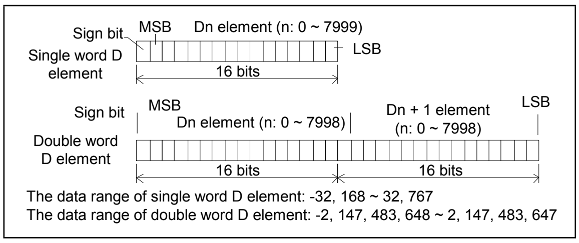
- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Every D or R element is a 16-bit register that can store data, like an 16-bit integer.

Two D or R elements can form a double-word and store a 32-bit data, such as the long integer data or floating-point data.



The data range of single word D element: -32, 168 ~ 32, 767
The data range of double word D element: -2, 147, 483, 648 ~ 2, 147, 483, 647

Figure 3-5 D or R element

Note

In a double-word D or R element, the higher 16-bit is in the first D or R element; and the lower 16-bit is in the second D or R element.

- **Available forms**

The D or R elements are used in many calculation and control instructions as the operands.

- **Value assignment**

1. Through initialization. 2. Through instructions.

3. Write or force during system debugging.

- **Power loss saving**

State	D elements in the saving range	D elements outside the saving range
Power loss	Remain unchanged	Cleared
RUN → STOP	Remain unchanged	Remain unchanged
STOP → RUN	Remain unchanged	Cleared
Note: The saving range is set through the system block. See ***Error! Reference source not found.Error! Reference source not found.***. R elements cannot be saved at power loss.		

Note

Some D elements may be reserved for internal tasks when the inverter instruction or N:N bus protocol is used.

3.1.9 Special auxiliary relay

- **Element mnemonic**

SM

- **Function**

The SM elements are closely related to the PLC system function. They reflect PLC system function and system state. For details, see *Appendix 1Special auxiliary relay*.

- **Classification**

The frequently used SM elements include:

- SM0: PLC operation monitor bit. It is ON when the PLC is in RUN state.
- SM1: initial operation pulse bit. It is ON in the first scan cycle of PLC operation.
- SM3: system error. It is ON if any system error is detected after PLC is powered on or when PLC changes from STOP to RUN.
- SM10~SM12: respetively the clock square-wave cycled at 10ms, 100ms and 1s (flipping-over twice in a cycle).

In addition, you can use, control or change the PLC system function by adjusting certain SM elements. Such elements include:

- SM40~SM68: interrupt control flag bit. Setting these SM elements will enable the corresponding interrupts.
- SM80/81: Y0/Y1 high-speed pulse output stop instruction.
- SM110~SM114: monitor bit of free port 0
- SM135/136: Modbus communication flag bit.
- SM172~SM178: integrated analog channel enabling flag (valid only for EC10-1614BRA1)

- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Boolean

- **Available forms**

NO and NC contacts

- **Value assignment**

1. Through instructions.

2. Write or force during system debugging.

Note

You cannot assign values to the read only SM elements.

3.1.10 Special data register

- **Element mnemonic**

SD

- **Function**

The SD elements are closely related to the PLC system function. They reflect PLC system function parameters, state code and instruction execution data. See *Appendix 2Special* data register for details.

- **Classification**

The frequently used SD elements include:

- SD3: system error code.
- SD50~SD57: high-speed pulse output monitor.
- SD100~SD106: real time clock data.

In addition, you can change PLC system function parameters by changing certain SD elements. Such elements include:

- SD66~SD68: cycle of timed interrupt.
- SD80~SD89: locating instruction parameters.

- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Word, double-word (integer)

- **Available forms**

Storage and calculation of integers

- **Value assignment**

1. Through instructions.
2. Write or force during system debugging.

📖 **Note**

You cannot assign values to the read only SD elements.

3.1.11 Indexed addressing register

- **Element mnemonic**

Z

- **Function**

The Z elements are 16-bit registers that can store signed integers. For detailed indexed addressing information, see **Error! Reference source not found.Error! Reference source not found.**.

- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Word

- **Available forms**

The Z elements are used for indexed addressing. You need to write the addressing offset to the Z elements before you can use them.

- **Value assignment**

1. Through instructions.
2. Write or force during system debugging.

3.1.12 Local auxiliary relay

- **Element mnemonic**

LM

- **Function**

The LM elements are local variables and can be used in the main program and subprograms. But being local variables, they are valid only in a certain program. Different programs cannot share the same LM element directly. When the system jumps from one program to another, the system will redefine the LM element. When the system returns to the main program or calls a subprogram, the redefined LM element will be cleared, or be set by the interface parameter transfer.

The LM elements can be used to define the interface parameters of subprograms to realize interface parameter transfer. For details, see **Error! Reference source not found.Error! Reference source not found.**.

- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Boolean

- **Available forms**

NO and NC contacts

- **Value assignment**

1. Through instructions.

3.1.13 Local data register

- **Element mnemonic**

V

- **Function**

The V elements are local variables and can be used in the main program and subprograms. But being local variables, they are valid only in a certain program. Different programs cannot share the same V element directly. When the system jumps from one program to another, the system will redefine the V element. When the system returns to the main program or calls a subprogram, the redefined V element will be cleared, or be set by the interface parameter transfer.

The V elements can be used to define the interface parameters of subprograms to realize interface parameter transfer. For details, see **Error! Reference source not found.Error! Reference source not found.**.

- **Elements numbered in**

Decimal, starting with 0

- **Data type**

Boolean

- **Available forms**

Word, for numeric information

- **Value assignment**

1. Through instructions.

3.2 Elements addressing mode

3.2.1 Bit-string addressing mode (Kn addressing mode)

- **Concept**

The Kn addressing mode, or combined bit-string addressing mode, realizes addressing by combining bit elements into words or double words.

- **Kn addressing method**

The format is: “K(n)(U)”, where the “n” is an integer from one to eight, standing for the length of the bit string: n×4. The “U” stands for the address of the starting element.

For example:

1. K1X0 stands for a word made up of (X0, X1, X2, X3).
2. K3Y0 stands for a word made up of (Y0, Y01, Y02, Y03), (Y04, Y05, Y06, Y07), (Y10, Y11, Y12, Y13).
3. K4M0 stands for a word made up of M0, M1, M2, M3…, M15.
4. K8M0 stands for a word made up of M0, M1, M2, M3…, M31.

- **Data storage format of Kn addressing mode**

The following is an example of how a specific data can be stored using the Kn addressing mode:

MOV 2#10001001 K2M0 (which is equal to MOV 16#89 K2M0, or MOV 137 K2M0). After executing the instruction, the result is:

Data	Highest bit	Middle bit						Lowest bit
K2M0	M7	M6	M5	M4	M3	M2	M1	M0
16#89	1	0	0	0	1	0	0	1

- **Notes**

If the destination operand uses the Kn addressing mode, while the data to be stored is longer than the length of the destination operand, the system will keep the lower bits and discard the higher bits.

For example:

Execute instruction DBITS 16# FFFFFFF0 K1M0. After executing the instruction, the operand 2 (K1M0) should store the calculation result 16# 1c (28). However, the K1M0 is only 4 bits wide, which is not enough for 16# 1c. By discarding the higher bits, the actual operand 2 is K1M0=16# c (12).

3.2.2 Indexed addressing mode (Z addressing mode)

- **Concept**

The EC20/EC20H series PLC provides the Z addressing mode, or indexed addressing mode. You can use the Z elements (indexed addressing register) to get indirect access to the targe elements.

- **Z addressing method**

Targe address=Basic element address+Address offset stored in Z element

For example:

In the indexed addressing mode, for D0Z0 (Z0=3), the target address is D3, because D0 is the basic address, and the address offset is stored in element Z0, which in this case, is 3.

Therefore when Z0=3, the instruction “MOV 45 D0Z0” is equal to “MOV 45 D3” in effect, because in both cases the D3 is set as 45 by the instruction.

- **Indexed addressing example**

1. Bit element indexed addressing example

```
LD   M01
MOV  6  Z1
SFTR  X0Z1  M0  8  2
```

The preceeding instructions are in effect equal to:

```
LD   M0 1
SFTR  X6  M0  8  2
```

The addressing process is as follows:

Z1=6

X0Z1=X(0+Z1)=X6

2. Word element indexed addressing example

```
LD   M0 1
MOV  30  Z20
MOV  D100Z20   D0
```

The preceeding instructions are in effect equal to:

```
LD   M0 1
MOV   D130   D0
```

The addressing process is as follows:

Z20=30

D100 Z20=D(100+Z20)=D130

- **Notes**

1. The Z elements store the address offset for the indexed addressing mode. They support signed integers, which means minus offset is supported.

For example:

```
MOV   -30   Z20
MOV   D100Z20   D0
```

The preceeding instructions are equal to the following one in effect:

```
MOV   D70   D0
```

2. The SM elements and SD elements do not support the Z addressing mode.

3. Pay attention to the address range when using the Z addressing mode. For example, D7999Z0 (Z0=9) is outside the address range of the D elements, which is not bigger than D7999.

3.2.3 Indexing addressing mode in bit-string combination

The indexed addressing mode can be used in combination with the bit-string addressing mode. For example: K1X0Z10.

In this mode, the starting element address is found through the Z addressing mode, and then the Kn addressing mode is used to determine the length of the bit string.

For example:

```
LD   M1
MOV   3   Z10
MOV   K1X0Z10   D0
```

The preceeding instructions are in effect equal to:

```
LD   M1
MOV   K1X3   D0
```

The addressing process is as follows:

Z10=3

K1X0Z10=K1X (0+Z10)=K1X3

3.2.4 Storing&addressing 32-bit data in D, R and V Elements

- **Storing 32-bit data in D, R and V elements**

The DINT, DWORD and REAL data are all 32-bit, while the D, R or V elements are only 16-bit. Two consecutive D, R or V elements are needed to store the 32-bit data.

The EC20 series PLC stores the 32-bit data in the Big Endian mode, which means the elements with smaller addresses are used to store the higher bits, while the elements with bigger addresses are used to store the lower bits.

For example, the signless integer “16# FEA8_67DA” is stored in the element (D0, D1). The actual storing format is:

D0	0xFEA8
D1	0x67DA

- **Addressing 32-bit data in D, R and V elements**

You can use a D or V element to locate a 16-bit data, such as an INT or WORD data, or a 32-bit data, such as a DINT or DWORD data.

If a D, R or V element address is used in an instruction, the operand data type determines whethther the data is 16-bit or 32-bit.

For example:

In the instruction “MOV   16#34   D0”, the address D0 stands for a single D0 element, because operand 2 of the MOV instruction is of the WORD data type.

In the instruction “DMOV   16# FEA867DA   D0”, the address D0 stands for two consecutive words: D0 and D1, because operand 2 of the DMOV instruction is of the DWORD data type.

3.3 Data

3.3.1 Data type

All instruction operands are of a certain data type. There are altogether six data types, as listed in the following table:

Table 3-2 Operand data types

Data type	Type description	Data width	Range
BOOL	Bit	1	ON, OFF (1, 0)
INT	Signed integer	16	-32768~32767
DINT	Signed double integer	32	-2147483648~2147483647
WORD	Word	16	0~65535 (16#0~16#FFFF)
DWORD	Double word	32	0~4294967295 (16#0~16#FFFFFFFF)
REAL	Floating point	32	±1.175494E-38~±3.402823E+38

3.3.2 Correlation between elements and data types

The elements used as instruction operands must have suitable data types. The correlations are listed in the following table.

Table 3-3 Elements and data type correlations

Data type	Elements													
BOOL										C	T			
	X	Y	M	S	LM	SM								
INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R
DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R
WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R
DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V		R
REAL	Constant							D				V		R

If an instruction uses an operand with unsuitable data type, the instruction will be deemed illegal. For example, instruction “MOV 10 X0” is illegal because operand 2 of the MOV instruction is of signed integer data type, while the X0 element can store only Boolean data.

Note

1. When the operand is of INT or WORD type, the applicable elements include KnX, KnY, KnM, KnS, KnLM and KnSM, where 1≤n≤4
2. When the operand is of DINT or DWORD type, the applicable elements include KnX, KnY, KnM, KnS, KnLM and KnSM, where 5≤n≤8
3. When the operand is of INT or WORD type, the applicable C elements are C0~C199.
4. When the operand is of DINT or DWORD type, the applicable C elements are C200~C255, C301~C306.

3.3.3 Constant

You can use constants as the instruction operands. EC20 series PLC supports input of multiple types of constants. The usual constant types are listed in the following table:

Table 3-4 Constant types

Constant type	Example	Valid range	Remarks
Decimal constant (16-bit signed integer)	-8949	-32768~32767	
Decimal constant (16-bit unsigned integer)	65326	0~65535	
Decimal constant (32-bit signed	-2147483646	-2147483648~2147483647	

Constant type	Example	Valid range	Remarks
integer)			
Decimal constant(32-bit unsigned integer)	4294967295	0~4294967295	
Hex constant (16-bit)	16#1FE9	16#0~16#FFFF	The hex, octal or binary constants are neither positive nor negative by themselves. When used as operands, the positive and negative nature of these constants are determined by the data type of the operand.
Hex constant (32-bit)	16#FD1EAFE9	16#0~16#FFFFFFFF	
Octal constant (16-bit)	8#7173	8#0~8#177777	
Octal constant (32-bit)	8#71732	8#0~8#37777777777	
Binary constant (16-bit)	2#10111001	2#0~2#1111111111111111	
Binary constant (32-bit)	2#101110011111	2#0~2#1111111111111111 1111111111111111	
Single-precision floating point	-3.1415E-16 3.1415E+3 0.016	±1.175494E-38~±3.402823E+38	Compliant with IEEE-754. The programming software can display and input floating point constants with 7-bit of operational accuracy

Chapter 4 Programming concepts

This chapter details the programming of EC series small PLC, including the programming language, program components, data type, addressing mode and annotating function. The programming and usage of subprograms are also introduced, and finally, the general explanation of instructions.

4.1 Programming language

Three programming languages are provided: ladder diagram (LAD), instruction list (IL) and sequential function chart (SFC).

4.1.1 Ladder diagram (LAD)

- **Concepts**

The LAD is a widely-used diagram programming-language, similar to the electric (relay) control diagram. It features:

1. Left bus, with right bus omitted.
2. All control output elements (coils) and functional blocks (application instructions) share the same power flow inlet.

The electric control diagram and LAD are equivalent to a certain degree, as shown in the following figure.

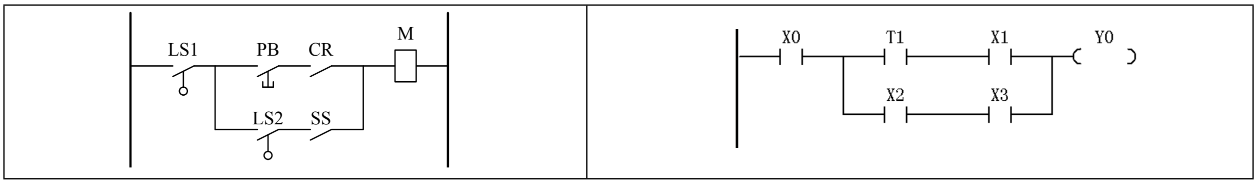


Figure 4-1 The equivalence between electric control diagram and LAD

- **LAD basic programming components**

According to the principles in electric control diagram, several basic programming components are abstracted for the LAD:

1. Left bus: Corresponding to the control bus in electric control diagram, it provides power for the control circuit.
2. Connecting line (— |): Corresponding to the electric connection in electric control diagram, it connects different components.
3. Contact (⊣⊢): Corresponding to the input contact in electric diagram, it controls the ON/OFF and direction of control currents. The parallel and serial connection of contacts stands for the logic calculation of inputs, determining the transfer of power flow.
4. Coil (-( )): It corresponds to the relay output in electric control diagram.
5. Function block (□): Or application instruction. Corresponding to the execution unit or functional device that provides special functions in electric control diagram, it can accomplish specific control function or control calculation function (like data transmission, data calculation, timer and counter).

- **Power flow**

Being an important concept in LAD, the power flow is used to drive coils and application instructions, which is similar to the control current output by the driving coil, and executed by the execution unit in electric control diagram.

In LAD, the coils or application instructions must be preceded with power flow, because the coils can output and instructions can be executed only when the power flow is ON.

The following figure demonstrates the power flow in LAD and how the power flow drives coils or function blocks.

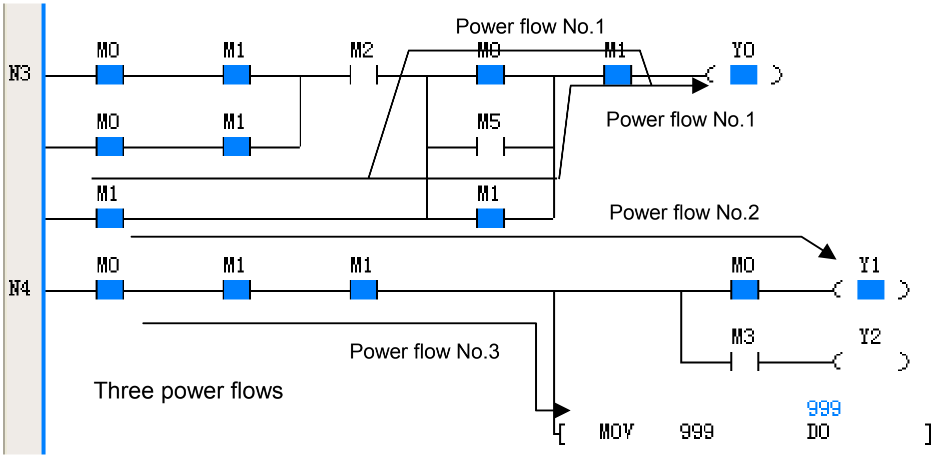


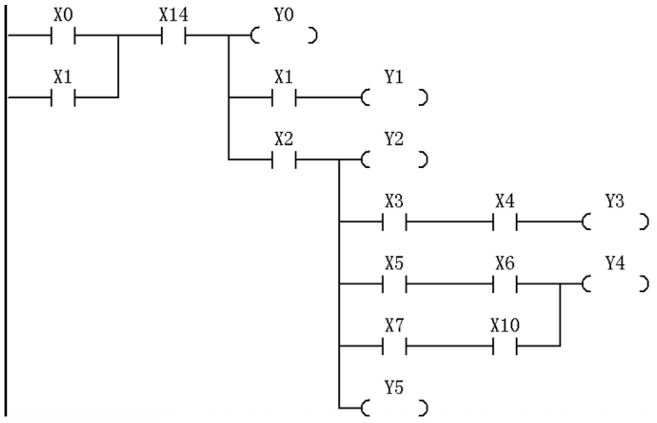
Figure 4-2 Power flow and its driving function

4.1.2 Instruction list (IL)

The IL, or the instruction list composed by users, is a text programming language.

The user program stored in the PLC main module is actually the instruction list recognizable to the main module. The system realizes the control function by executing the instructions in the list one by one.

The following is an example of equivalent LAD and IL.

LAD	IL
	LD X0 OR X1 AND X14 MPS OUT Y0 AND X1 OUT Y1 MPP AND X2 MPS OUT Y2 AND X3 AND X4 OUT Y3 MRD LD X5 AND X6 LD X7 AND X10 ORB ANB OUT Y4 MPP OUT Y5

4.1.3 Sequential function chart (SFC).

The SFC is a diagram programming-language usually used to realize sequence control, which is a control process that can be divided into multiple procedures and proceed according to certain working sequence.

The user program designed with SFC is direct and clear because it has a structure similar to the actual sequence control process.

See the following figure for a simple example of SFC.

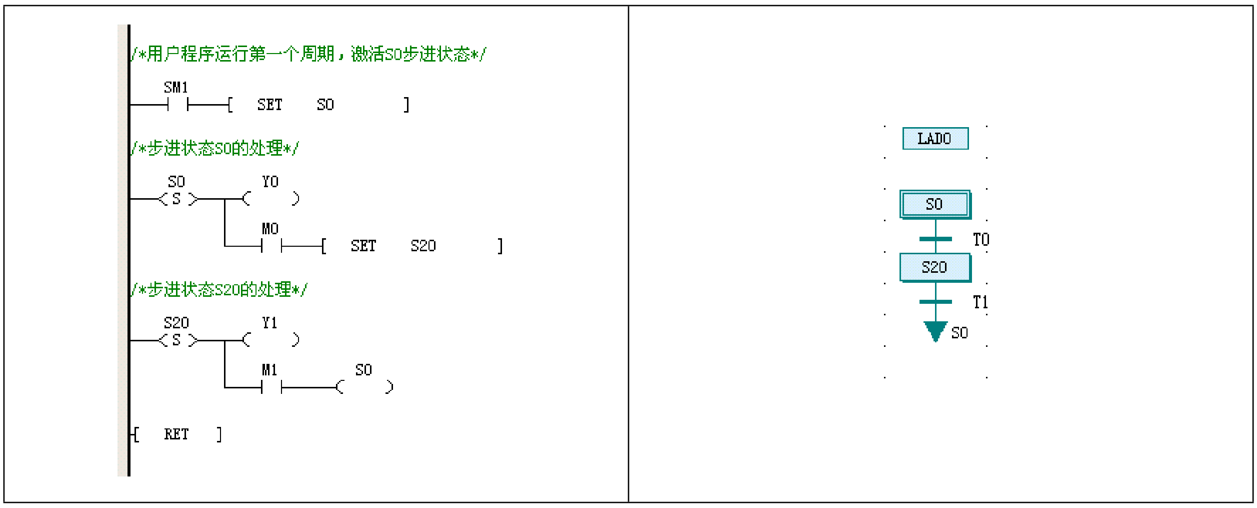


Figure 4-3 Example of SFC

4.2 Program components

The program components include user program, system block and data block. You can change these components by programming.

4.2.1 User program

A user program is the program code composed by users. It must be compiled into executable instruction list, downloaded to the PLC and executed to realize the control function.

The user program comprises three program organization units (POU): main program (MAIN), subprogram (SBR) and interrupt (INT).

- **Main program (MAIN)**

The main program is the main body and framework of the user program. When the system is in RUN state, the main program will be executed cyclically.

One user program has only one main program.

- **Subprogram (SBR)**

A subprogram is a program independent in structure and function. It can be called by other POUs. Subprograms generally have call operand interface and are executed only when being called.

A user program can have random number of subprograms, or no subprogram at all.

- **Interrupt (INT)**

An interrupt is a program section handling a specific interrupt event. A specific interrupt event always corresponds to a specific interrupt.

Upon the occurance of an interrupt event, a ordinary scan cycle will be interrupted. The system will run the corresponding interrupt until the interrupt is finished, when the system will return to the ordinary scan cycle.

A user program can have random number of interrupts, or no interrupts at all.

4.2.2 System block

The system block contains multiple system configuration parameters. You can modify, compile and download the system block to configure the operation mode of the main module.

For details, see **Error! Reference source not found.Error! Reference source not found.** or the related description in *Programmer Programming Software User Manual*.

4.2.3 Data block

The data block contains the values of D or R elements. By downloading the data block to the PLC, you can set a batch of designated D or R elements.

If the **Datablock enabled** is checked in the **Advanced Settings** tab of **System block,** the D or R elements will be initialized by the data block before the PLC executes the user program.

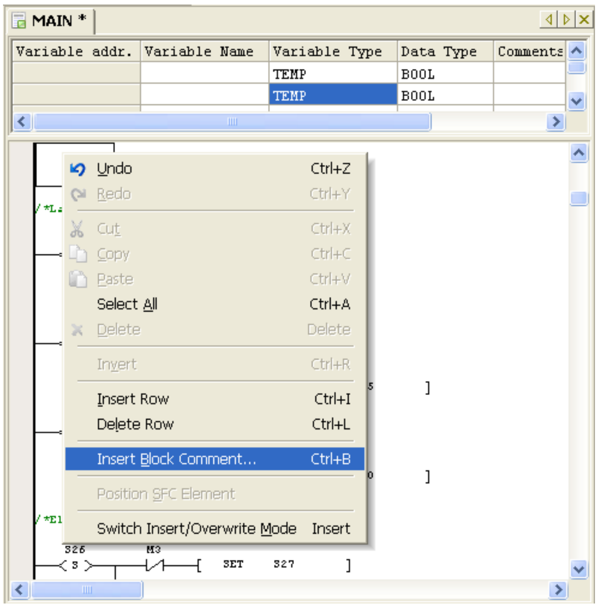
4.3 Block comment and variable comment

4.3.1 Block comment

You can add comments to the program. Occupying a whole row, each piece of comment can be used to explain the function of the following program block.

In the program, right click and select **Insert Row** to insert a row above the current row. You can use an empty row to separate two program sections.

To make a block comment, just select an empty row, right click and select **Insert Block Comment.**

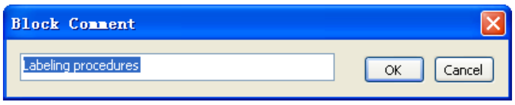


MAIN *

Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	

Figure 4-4 Adding block comment

Input your comment into the **Block Comment** dialog box that pops out and click the **OK** button

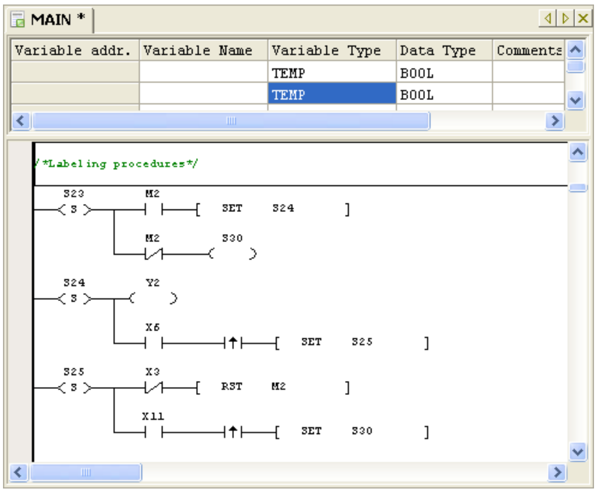


Block Comment

Labeling procedures

Figure 4-5 Block comment dialog box

The comment will appear in the empty row, as shown below:



MAIN *

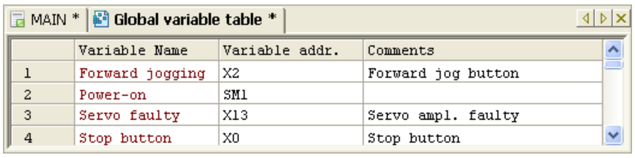
Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	

Figure 4-6 Block comment dialog box

A block comment occupies a whole row. You cannot add a block comment to an occupied row, nor can a row occupied by a comment be used for other purposes.

4.3.2 Variable comment

You can define variables in the **Local variable table** and **Global variable table**. (See *2.2.3Global variable table* and *4.4.3SBR local variable table*) , and use them in the LAD programming language. A variable can stand for a certain address to make the program more sensible. Figure 4-7 shows some variables defined in a global variable table.



	Variable Name	Variable addr.	Comments
1	Forward jogging	X2	Forward jog button
2	Power-on	SM1	
3	Servo faulty	X13	Servo ampl. faulty
4	Stop button	X0	Stop button

Figure 4-7 Variables defined in the global variable table

■ **Symbol addressing**

When the defined variables are used, you can select **View->Symbol Addressing** to display their names instead of their addresses in the LAD or IL program.

The following figure shows the LAD program when the **Symbol Addressing** is not checked.

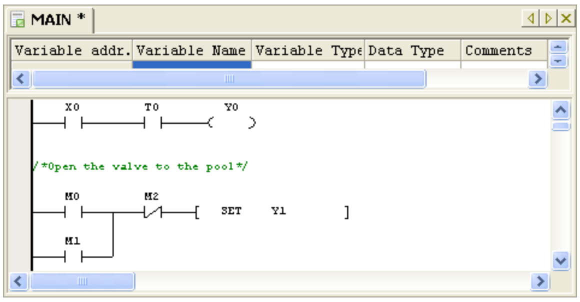


Figure 4-8 When symbol addressing is unchecked

The following figure shows the LAD program when the **Symbol Addressing** is checked.

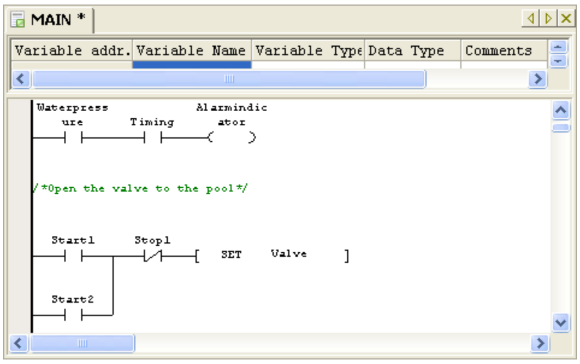
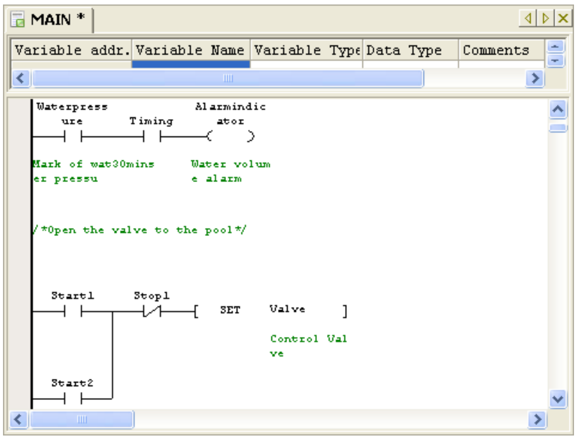


Figure 4-9 When symbol addressing is checked

■ **Element comment**

You can select **View->Element Comment** to display the element comments in the LAD program, as shown in **Error! Reference source not found.**.



Variable addr.	Variable Name	Variable Type	Data Type	Comments

Figure 4-10 LAD program displaying element comments

Note

The block comment, global variable table and local variable table can be compiled and downloaded to EC20 and EC20H series PLC. To store such information, battery backup is needed. However, although battery failure may cause information loss, comment upload failure and user information file error report, the user program can still run normally.

4.4 Subprogram

4.4.1 Concept

Being an optional part of the user program, a subprogram (SBR) is an independent program organization unit (POU) that can be called by the main program or other SBRs.

You can use SBRs in your user program to:

1. Reduce the size of the user program. You can write a repeated program section as a SBR and call it whenever necessary.
2. Clarify the program structure, particularly the structure of the main program.
3. Make the user program more transplantable.

4.4.2 Note for using SBRs

Note the following when writing or calling a SBR:

1. The PLC supports up to 6 levels of SBR nesting.

The following is an fine example of 6-level of SBR nesting:

MAIN→SBR1→SBR2→SBR3→SBR4→SBR5→SBR6 (where the “→” represents calling with the CALL instruction)

2. The PLC does not support recursive calling and cyclic calling of SBRs.

The following two examples show two illegal SBR callings.

- MAIN→SBR0→SBR0 (recursive calling, illegal)
- MAIN→SBR0→SBR1→SBR0 (cyclic calling, illegal)

3. You can define up to 64 SBRs in a user program.
4. You can define up to 16 bit variables and 16 word variables in the local variable table of a SBR.
5. When calling a SBR, the operand type of the CALL instruction must match the variable type defined in the SBR local variable table. The compiler will check the match.
6. The interrupts are not allowed to call SBRs.

4.4.3 SBR local variable table

■ **Concept**

The SBR local variable table displays all SBR interface parameters and local variables (both are called variables) and stipulates their properties.

■ **SBR variable properties**

The SBR variables (including interface parameters and local variables) have the following properties:

1. Variable address

Based on the variable data type, the software will automatically assign a fixed LM or V element address to each SBR variable in sequence.

2. Variable name

You can give each SBR variable a name (alias). You can use a variable in the program by quoting its name.

3. Variable type

The SBR variables are classified into the following four types:

- IN: The IN type variables can transfer the inputs of SBR when the SBR is being called.
- OUT: The OUT type variables can transfer the SBR execution result to the main program when a SBR calling ends.
- IN_OUT: The IN type variables can transfer the inputs of SBR when the SBR is being called, or transfer the the SBR execution result to the main program when a SBR calling ends.
- TEMP: The TEMP variables are local variables that are valid only within the SBR.

4. Data type

The variable data type specifies the range of the data. The variable data types are listed in the following table.

Table 4-1 Variable data types

Data type	Description	Occupid LM/V element address
BOOL	Bit type	One LM element address
INT	Signed integer type	One V element address
DINT	Signed double integer type	Two consecutive V element addresses
WORD	Word type	One V element address
DWORD	Double word type	Two consecutive V element addresses
REAL	Floating point type	Two consecutive V element addresses

4.4.4 SBR parameter transfer

If local input or output variables are defined in a SBR, when the main program calls the SBR, you should input the corresponding variable values, global variables or temporary variables into the SBR interface parameters. Note that the global variable should be of the same data type with the local variable.

4.4.5 Example

What follows is an example of how to write and call a SBR.

- **Function of this example SBR**

Call SBR_1 in the main program to complete a adding calculation of two integer constants 3 and 2, and assign the result 5 to D0.

- **Operation procedures**

Step 1: Insert a SBR into the project and name it as SBR_1.

Step 2: Write SBR_1.

1. Set the SBR calling interface through the SBR_1 variable table.

1) Variable 1: Name it as IN1 (variable type: IN). Set the data type as INT. The software will assign it with a V element address of V0.

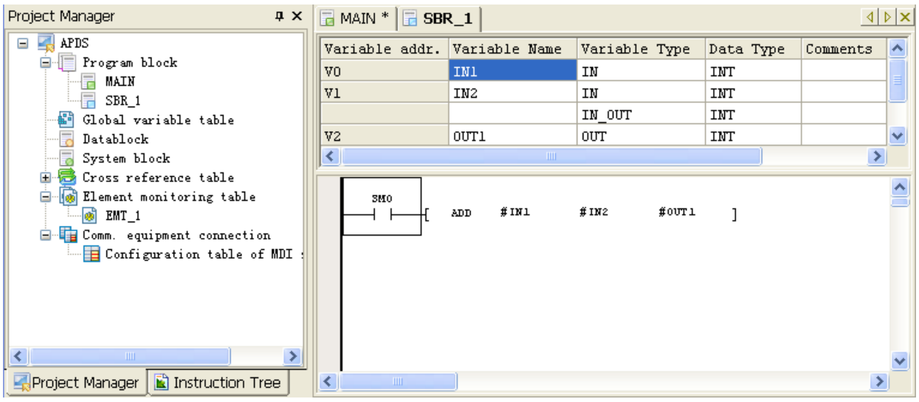
2) Variable 2: Name it as IN2 (variable type: IN). Set the data type as INT. The software will assign it with a V element address of V1.

3) Variable 3: Name it as OUT1 (variable type: OUT). Set the data type as INT. The software will assign it with a V element address of V2.

2. Write the SBR_1 as:

```
LD   SM0
ADD  #IN1  #IN2  #OUT1
```

The above program is shown in the following figure.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
V0	IN1	IN	INT	
V1	IN2	IN	INT	
		IN_OUT	INT	
V2	OUT1	OUT	INT	

```
SM0
─┤ ├─[ ADD   #IN1   #IN2   #OUT1   ]
```

Figure 4-11 Writing SBR_1

Step 3: Write the main program and call the SBR

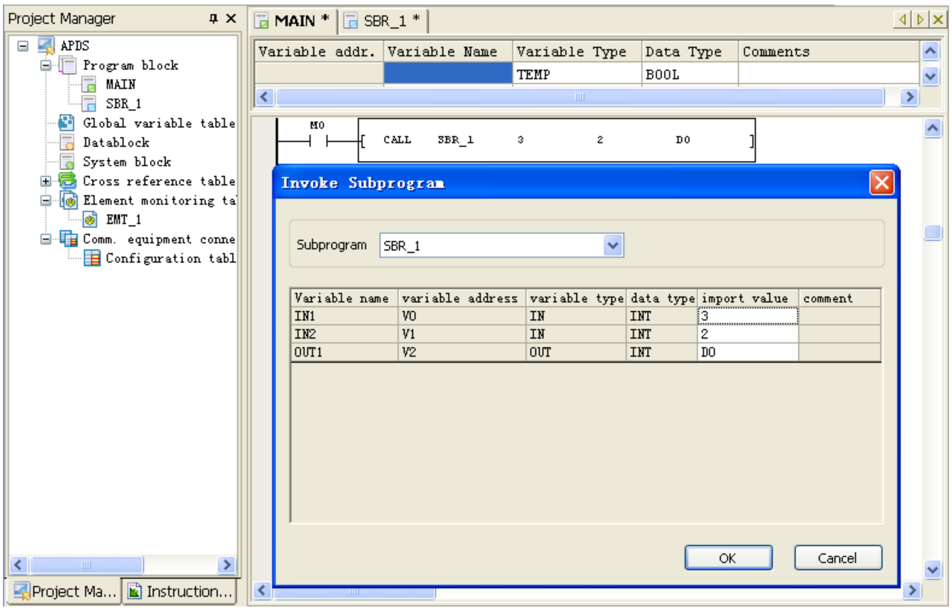
Use the CALL instruction in the main program to call SBR_1.

The corresponding main program is as shown below:

```
LD    M0
CALL  SBR_1  3  2  D0
```

You can use the parameter transfer relationship table as shown in the following figure to set the parameters transferred to the subprogram and specify the element for storing the result of the subprogram.

- Parameter IN1 is used to transfer constant integer 3
- Parameter IN2 is used to transfer constant integer 2
- The result OUT1 is stored in D0



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	

```
M0
─┤ ├─[ CALL   SBR_1   3   2   D0   ]
```

Invoke Subprogram

Subprogram SBR_1

Variable name	variable address	variable type	data type	import value	comment
IN1	V0	IN	INT	3	
IN2	V1	IN	INT	2	
OUT1	V2	OUT	INT	D0	

Figure 4-12 Calling subprogram

Step 4: Compile, download and run the user program and check the correctness of the SBR logic.

■ **Execution result**

When M0 is ON, SBR_1 will be called. Values 2 and 3 are transferred to the operands IN1 and IN2 to carry out the calculation operation. The result 5 is then returned to the main program, and in the end, D0 is 5.

4.5 General information of instructions

4.5.1 Instruction operands

The instruction operands can be classified into the following two types:

- Source operands: or S (or S_1, S_2, S_3 … when there are more than one of them in the same instruction). The instruction reads values from source operands for calculation.
- Destination operands: or D (or D_1, D_2, D_3 … when there are more than one of them in the same instruction). The instruction controls or outputs values to the destination operands.

The operands could be bit elements, word elements, double-word elements, or constants. See the specific instruction description in *Chapter 5* or *Chapter 6* for details.

4.5.2 Flag bit

The instruction result may affect three kinds of flag.

- **Zero flag SM180**

The zero flag is set when the instruction operation result is zero.

- **Carry flag SM181**

The carry flag is set when the instruction operation result is a carry.

- **Borrow flag SM182**

The borrow flag is set when the instruction operation result is a borrow.

4.5.3 Limits to instruction usage

There are some limits to the usage of certain instructions. For details, see the description of the specific instruction.

- **Exclusive hardware resources**

Some instructions requires hardware resources. When a specific hardware is being used by a certain instruction, the access to the hardware will be denied to other instructions, because the occupation of the resource is exclusive.

Take the high-speed I/O instructions and SPD instruction for example. Any of these instructions occupies a input point among X0~X7. The limited resources will make it impossible to execute these instructions at the same time.

- **Exclusive time**

The execution of certain instructions may take some time. During such period, the system will be too busy to execute other instructions.

Take the XMT instruction for example. Because of the time limit in communication, only one XMT instruction can be executed once. In the same way, the free port can execute only one RCV instruction once. Every time when a Modbus instruction is being executed, the communication channel will be unavailable to other instructions for a while. The same is true to other instructions such as high-speed output instructions, locating instructions and inverter instructions.

- **Application limit**

Some instructions cannot be used in certain situations due to their limited application scope.

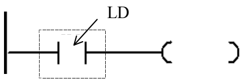
For example, instruction pair MC/MCR cannot be used in the steps of SFC.

Chapter 5 Basic instructions

This chapter details the basic instruction of EC series small PLC, including the instruction format (form), operand, influenced flag bit, function, example and sequence chart.

Contact logic instructions

LD: NO contact power-flow loading

LAD:  	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: LD *(S)*	**Program steps**	**1**

Operand	Type	Applicable elements													Indexed addressing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

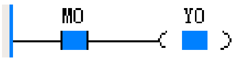
■ **Operand description**

S: Source operand

■ **Function description**

Connected to the left bus to connect (status: ON) or disconnect (status: OFF) the power flow.

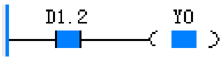
■ **Example**



LD M0

OUT Y0

When M0 is ON, Y0 is ON.



LD D1.2

OUT Y0

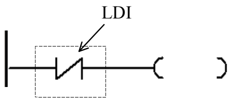
When the 2nd bit of D1 is 1, Y0 is ON.

■ **Note**

For the contact logic instructions of EC10 series, when the operands are M1536~M2047, the actual program steps will be the instruction program steps plus 1.

For the contact logic instructions of EC20H series, when the operands are M1536~M10240, C256~C511, T256~T511 and S0~S4096, the actual program steps will be the instruction program steps plus 1. When the operands are Dx.y, the program steps will be 4.

5.1.2 LDI: NC contact power-flow loading

LAD:  	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: LDI *(S)*	**Program steps**	**1**

Operand	Type	Applicable elements													Indexed addressing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

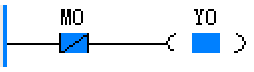
■ **Operand description**

S: Source operand

■ **Function description**

■ **Example**

Connected to the left bus to connect (status: OFF) or disconnect (status: ON) the power flow.



```
LDI    M0
OUT    Y0
```

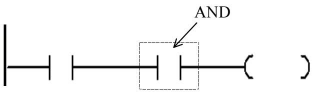
When M0 is OFF, Y0 is ON.

- **Note**

For the contact logic instructions of EC10 series, when the operands are M1536~M2047, the actual program steps will be the instruction program steps plus 1.

For the contact logic instructions of EC20H series, when the operands are M1536~M10240, C256~C511, T256~T511 and S0~S4096, the actual program steps will be the instruction program steps plus 1. When the operands are Dx.y, the program steps will be 4.

5.1.3 AND: NO contact power-flow and

LAD:	Applicable to	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: AND *(S)*	Program steps	1

Operand	Type	Applicable elements												Indexed addressing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T		

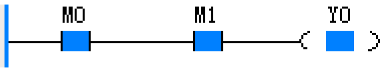
- **Operand description**

S: Source operand

- **Function description**

After conducting the “and” operation on the ON/OFF status of the designated contact (***S***) and the current power flow, assign the value to the current power flow.

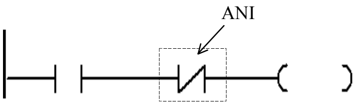
- **Example**



```
LD     M0
AND    M1
OUT    Y0
```

When M0 is ON and M1 is ON, Y0 is ON.

5.1.4 ANI: NC contact power-flow and

LAD:	Applicable to	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: ANI *(S)*	Program steps	1

Operand	Type	Applicable elements												Indexed addressing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T		

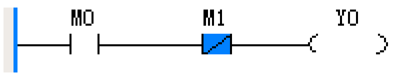
- **Operand description**

S: Source operand

- **Function description**

After reversing the ON/OFF status of the designated contact (***S***), conduct “and” operation on the reversed result and the current power flow, and then assign the value to the current power flow.

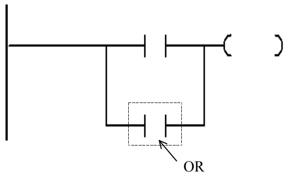
- **Example**



LD M0
ANI M1
OUT Y0

When M0 is ON and M1 is OFF, Y0 is ON.

5.1.5 OR: NO contact power-flow or

<table>
<tr><td colspan="8" rowspan="2">
LAD:</td><td colspan="3"><b>Applicable to</b></td><td colspan="5"><b>EC20 EC10 EC10A EC20H</b></td></tr>
<tr><td colspan="3"><b>Influenced flag bit</b></td><td colspan="5"></td></tr>
<tr><td colspan="8"><b>IL: OR <i>(S)</i></b></td><td colspan="3"><b>Program steps</b></td><td colspan="5"><b>1</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S</td><td>BOOL</td><td>X</td><td>Y</td><td>M</td><td>S</td><td>LM</td><td>SM</td><td></td><td>Dx.y</td><td></td><td>C</td><td>T</td><td></td><td></td><td></td></tr>
</table>

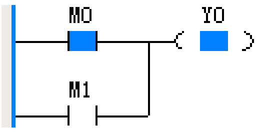
- **Operand description**

S: Source operand

- **Function description**

After conducting “OR” operation on the ON/OFF status of the designated contact (***S***) and the current power flow, assign the value to the current power flow.

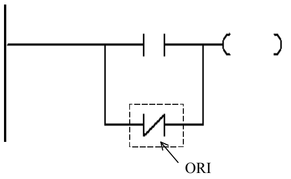
- **Example**



LD M0
OR M1
OUT Y0

When M0 or M1 is ON, Y0 is ON.

5.1.6 ORI: NC contact power-flow or

<table>
<tr><td colspan="8" rowspan="2">LAD:
 </td><td colspan="3"><b>Applicable to</b></td><td colspan="5"><b>EC10 EC10A EC10V EC20 EC20H</b></td></tr>
<tr><td colspan="3"><b>Influenced flag bit</b></td><td colspan="5"></td></tr>
<tr><td colspan="8"><b>IL: ORI <i>(S)</i></b></td><td colspan="3"><b>Program steps</b></td><td colspan="5"><b>1</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S</td><td>BOOL</td><td>X</td><td>Y</td><td>M</td><td>S</td><td>LM</td><td>SM</td><td></td><td>Dx.y</td><td></td><td>C</td><td>T</td><td></td><td></td><td></td></tr>
</table>

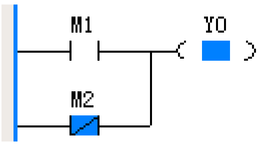
- **Operand description**

S: Source operand

- **Function description**

After reversing the ON/OFF status of the designated contact (S), conduct “OR” operation on the reversed result and the current power flow, and then assign the value to the current power flow.

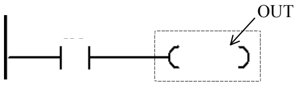
- **Example**



LD M1
ORI M2
OUT Y0

When M1 is ON or M2 is OFF, Y0 is ON.

5.1.7 OUT: Power-flow output

<table>
<tr><td>LAD:
 </td><td><b>Applicable to</b></td><td><b>EC10 EC10A EC10V EC20 EC20H</b></td></tr>
</table>

	Influenced flag bit	
IL: OUT *(S)*	**Program steps**	**1**

Operand	Type	Applicable elements													Indexed addressing
S	BOOL	X	Y	M	S	LM	SM		Dx.y		C	T			

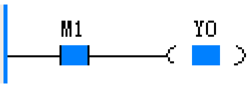
- ■ **Operand description**

S: Source operand

- ■ **Function description**

Assign the value of the current power flow to the designated coil (***D***)

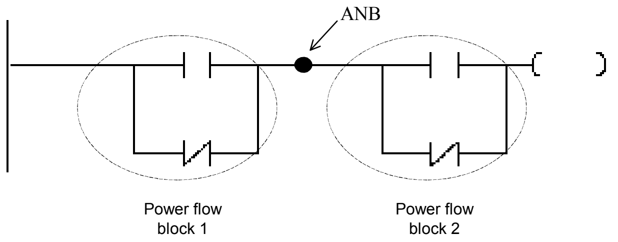
- ■ **Example**



```
LD    M1
OUT   Y0
```

When M1 is ON, Y0 is ON.

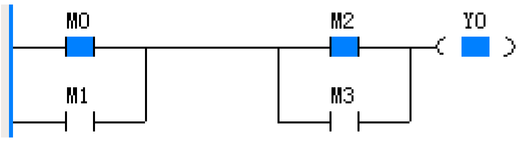
5.1.8 ANB: Power-flow block and

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: ANB	**Program steps**	**1**

- ■ **Operand description**
- ■ **Function description**

Conduct "and" operation on the power flow values of two power flow blocks, and then assign the value to the current power flow.

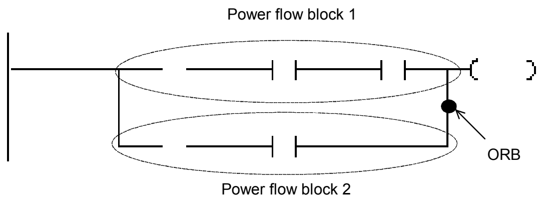
- ■ **Example**



```
LD    M0
OR    M1
LD    M2
OR    M3
ANB
OUT   Y0
```

When M0 or M1 is on, and M2 or M3 is ON, Y0 is ON.

5.1.9 ORB: Power-flow block or

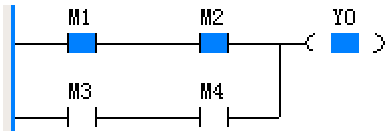
LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: ORB	**Program steps**	**1**

- ■ **Operand description**
- ■ **Function description**

Conduct "or" operation on the power flow values of two power flow blocks, and then assign the

- ■ **Example**

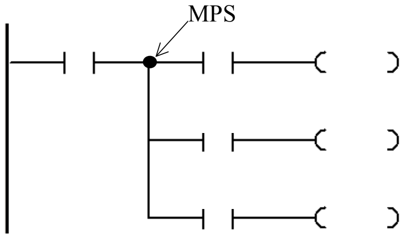
value to the current power flow.



```
LD   M1
AND  M2
LD   M3
AND  M4
ORB
OUT  Y0
```

When both M1 and M2 are ON, or both M3 and M4 are ON, Y0 outputs ON.

5.1.10 MPS: Output power-flow input stack

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: MPS	**Program steps**	**1**

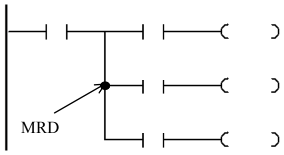
- **Function description**

Push the current power flow value into the stack for storage, so that it can be used in the power flow calculation for the subsequent output branches.

- **Note**

It is prohibited to use the MPS instruction consecutively for over 8 times in a LAD program (with no MPP instruction in between), otherwise the power flow output stack may overflow.

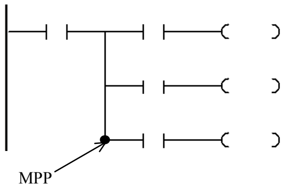
5.1.11 MRD: Read output power-flow stack top value

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: MRD	**Program steps**	**1**

- **Function description**

Assign the top value of the power flow output stack to the current power flow.

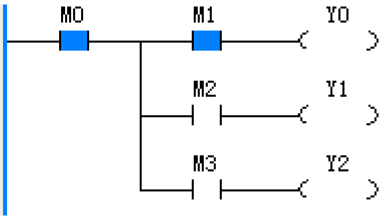
5.1.12 MPP: Output power-flow stack pop off

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: MPP	**Program steps**	**1**

- **Function description**

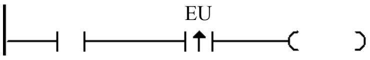
- **Example**

Pop the power flow output stack, and assign the popped value to the current power flow.



```
LD      M0
MPS
AND     M1
OUT     Y0
MRD
AND     M2
OUT     Y1
MPP
AND     M3
OUT     Y2
```

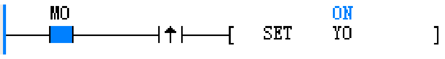
5.1.13 EU: Power flow rising edge detection

LAD:  	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: EU	**Program steps**	**2**

- **Function description**

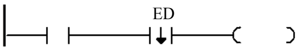
Compare the current power flow status with its previous status. If the power flow rises (OFF→ON), the output is valid in the current scan cycle.

- **Example**



```
LD      M0
EU
SET     Y0
```

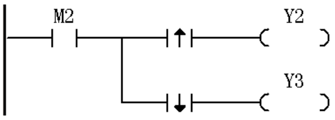
5.1.14 ED: Power flow falling edge detection

LAD:  	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: ED	**Program steps**	**2**

- **Function description**

Compare the current power flow status with its previous status. If the power flow falls (ON→OFF), the output is valid in the current scan cycle.

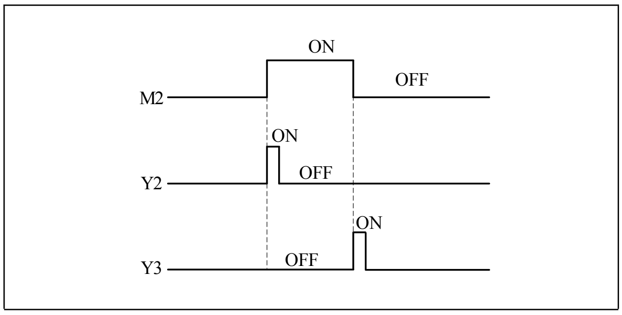
- **Example**



```
LD   M2
MPS
EU
OUT Y2
MPP
ED
OUT Y3
```

1. In two consecutive scan cycles, the status of M2 contact is OFF and ON respectively. When the EU instruction detects a rising edge, Y2 will output ON status with the width of a scan cycle.

- **Sequence chart of example**



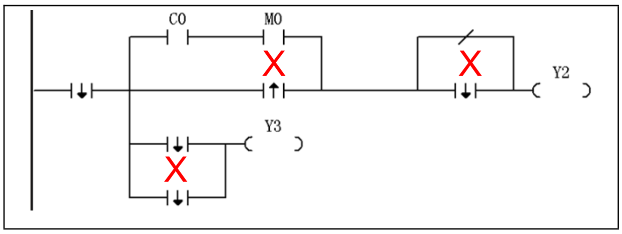
- **Note**

In LAD program, the rising edge contact or falling edge contact instruction shall be used in series rather than in parallel connection with other contact elements.

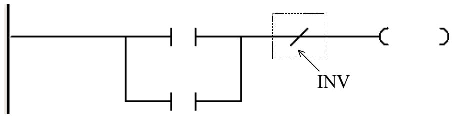
In LAD program, the rising edge contact and falling edge contact instruction cannot directly connect to the left power flow bus.

The examples of improper use of EU/ED instructions in LAD program are shown as follows:

2. In two consecutive scan cycles, the status of M2 contact is ON and OFF respectively, when the ED instruction detects a trailing edge, Y3 will output ON status with the width of a scan cycle.



5.1.15 INV: Power-flow block inverse

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: INV	**Program steps**	**1**

- **Function description**

Reverse the current power flow value and then assign to the current power flow.

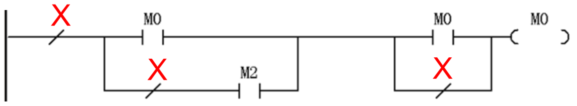
- **Note**

In LAD program, the INV instruction shall be used in series rather than in parallel connection with contacts.

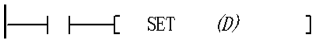
INV cannot be used as the first instruction in the input parallel branch.

In LAD program, the INV instruction cannot directly connect to the left power flow bus.

The examples of improper use of INV instructions in LAD program are shown as follows:



5.1.16 SET: Set

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: SET*(S)*	**Program steps**	**1**

Operand	Type	Applicable elements												Indexed addressing
S	BOOL		Y	M	S	LM	SM		Dx.y		C	T		

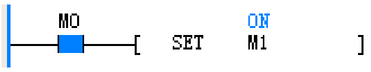
- **Operand description**

S: Source operand

Function description

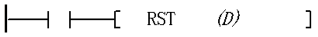
When the power flow is valid, the bit element designated by ***D*** will be set.

- **Example**



LD M0
SET M1

5.1.17 RST: Reset

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: RST*(S)*	**Program steps**	**1**

Operand	Type	Applicable elements												Indexed addressing
S	BOOL		Y	M	S	LM	SM		Dx.y		C	T		

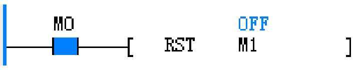
- **Operand description**

S: Source operand

- **Function description**

When the power flow is valid, the designated bit element (D) will be reset.

- **Example**

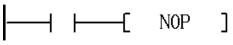


LD M0
RST M1

- **Note**

If D is C element, the corresponding count value will be reset; if D is T element, the corresponding timing value will be reset.

5.1.18 NOP: No operation

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: NOP	**Program steps**	**1**

- **Function description**

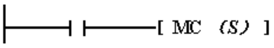
This instruction does not enable any action.

- **Note**

In LAD program, this instruction cannot directly connect to the left power flow bus.

5.2 Main control instruction

MC: Main control

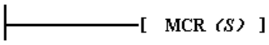
LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: MC*(S)*	**Program steps**	**3**

Operand	Type	Applicable elements													Indexed addressing
S	INT	Constant													

- **Operand description**

S: Source operand

5.2.2 MCR: Main control remove

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: MCR*(S)*	**Program steps**	**1**

Operand	Type	Applicable elements													Indexed addressing
S	INT	Constant													

- **Operand description**

S: Source operand

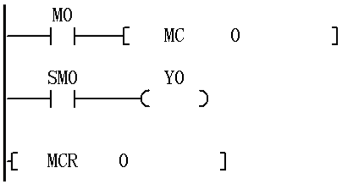
- **Function description**

1. MC and MCR form a MC-MCR structure. The MC instruction indicates the beginning a MC-MCR structure, and its operand S is the SN of the MC-MCR structure. The value of S is a constant ranging from 0 to 7. MCR indicates the end of the MC-MCR structure.

2. When the power flow before the MC instruction is valid, the instructions in the MC-MCR structure will be executed.

3. When the power flow before the MC instruction is invalid, the program will skip over the instructions in the MC-MCR structure and execute the instructions after the structure. Besides, the destination operands of instructions OUT, TON, TOF, PWM, HCNT, PLSY, PLSR, DHSCS, SPD, DHSCI, DHSCR, DHSZ, DHST, DHSP and BOUT in the structure will be cleared.

- **Example**



```
LD   M0
MC   0
LD   SM0
OUT  Y0
MCR  0
```

When M0=ON, the instructions in the MC 0-MCR 0 structure will be executed, and Y0=ON. When M0=OFF, the instructions in the MC 0-MCR 0 structure will not be executed, and the bit element Y0 designated by the designation operand of the OUT instruction in the structure will be cleared, Y0=OFF.

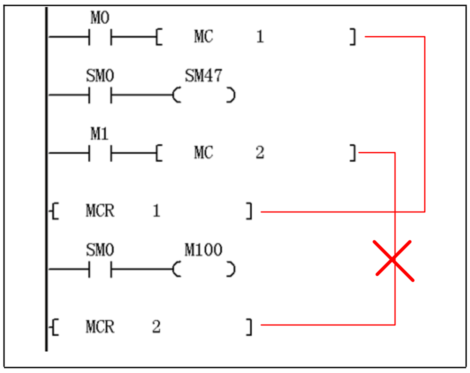
- **Note**

1. In LAD program, the MCR instruction must directly connect to the left power flow bus.

2. In LAD program, the MCR instruction cannot connect to other instructions.

3. Several MC-MCR structures of different SNs can be used through the nest structure, but the number of nest levels cannot exceed 8. The MC-MCR structures with the same SN cannot be used in the nest structure.

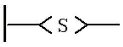
4. Crossing of two MC-MCR structures is not allowed. The following is an illegal example.



Note: It cannot be used in SFC programming.

5.3 SFC instructions

STL: SFC state load instruction

<table>
<tr><td colspan="15" rowspan="2"><b>LAD:</b>
</td><td><b>Applicable to</b></td><td><b>EC10 EC10A EC10V EC20 EC20H</b></td></tr>
<tr><td><b>Influenced flag bit</b></td><td></td></tr>
<tr><td colspan="15"><b>IL: STL<i>(S)</i></b></td><td><b>Program steps</b></td><td><b>3</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable elements</td><td colspan="2">Indexed addressing</td></tr>
<tr><td>S</td><td>BOOL</td><td></td><td></td><td></td><td></td><td></td><td>S</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="2"></td></tr>
</table>

Operand description

S: Source operand

Function description

1. It indicates the beginning of a step (***S***).

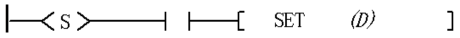
2. If a step is valid (ON), its embedded instructions will be executed.

3. If a step changes from ON to OFF (falling edge), the embedded instructions will not be executed, and the destination operands of the embedded instructions such as OUT, TON, TOF, PWM, HCNT, PLSY, PLSR, DHSCS, SPD, DHSCI, DHSCR, DHSZ, DHST, DHSP and BOUT will be cleared.

4. If a step is invalid (OFF), the embedded instructions will not be executed.

5. Consecutive STL instructions (serial connection of S elements) define a parallel merge structure. The STL instructions can be used up to 16 times in a row (the maximum number of branches of a parallel branch structure is 16).

5.3.2 SET Sxx: SFC state transfer

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: SET*(D)*	**Program steps**	**3**

Operand	Type	Applicable elements													Indexed addressing
D	BOOL						S								

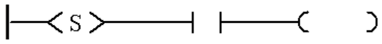
- **Operand description**

D: Destination operand

- **Function description**

When the power flow is valid, the designated step (***D***) will be set valid, and the current valid step will be set invalid, to complete the step transition.

5.3.3 OUT Sxx: SFC state jump

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: OUT*(D)*	**Program steps**	**3**

Operand	Type	Applicable elements													Indexed addressing
D	BOOL						S								

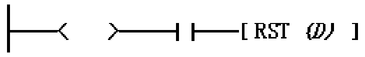
- **Operand description**

D: Destination operand

- **Function description**

When the power flow is valid, the designated step (***D***) will be set valid, and the current valid step will be set invalid, to complete the step transition.

5.3.4 RST Sxx: SFC state reset

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: RST*(D)*	**Program steps**	**3**

Operand	Type	Applicable elements													Indexed addressing
D	BOOL						S								

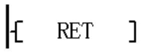
- **Operand description**

D: Destination operand

- **Function description**

When the current power flow is valid, the designated step (***D***) will be set invalid.

5.3.5 RET: SFC program end

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: RET	**Program steps**	**1**

- **Function description**

It indicates the end of a SFC program section.

- **Note**

It can only be used in the main program.

5.4 Timer instruction

TON: On-delay timing instruction

LAD: |—| |—[TON *(D)* *(S)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: TON *(D)* *(S)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
D	INT											T				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

D: Destination operand

S: Source operand

■ **Function description**

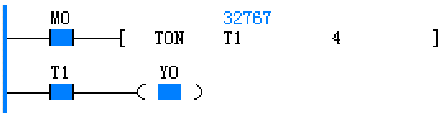
1. When the power flow is valid, and the timing value<32,767, the designated T element (***D***) will start timing (the value will increase with the lapse of time). When the timing value reaches 32,767, it will maintain at 32,767.

2. When the timing value≥the preset value (***S***), the timing coil output of the designated T element will be ON.

3. When the power flow is OFF, the timing will stop, the timing value will be cleared, and the timing coil output will be OFF.

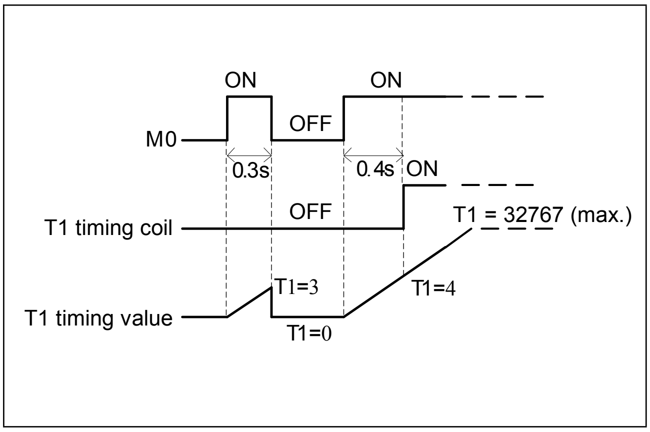
4. When the system executes the instruction for the first time, it will reset the timing coil of the designated T element, and clear the timing value.

■ **Example**



LD M0
TON T1 4
LD T1
OUT Y0

■ **Time sequence chart**



5.4.2 TONR: On-delay remember timing instruction

LAD: |—| |—[TONR *(D)* *(S)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: TONR *(D)* *(S)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
D	INT											T				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

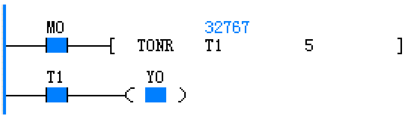
D: Destination operand

S: Source operand

■ **Function description**

1. When the power flow is valid, and the timing value<32,767, the designated T element (***D***) start timing (the value will increase with the lapse of

■ **Example**

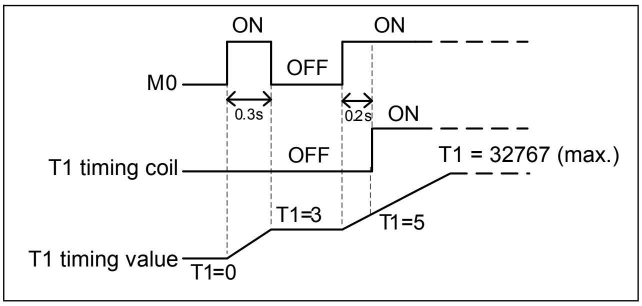


LD M0
TONR T1
5
LD T1
OUT Y0

■ **Time sequence chart**

2. When the timing value≥the preset value (***S***), the timing coil output of the designated T element will be ON.

3. When the power flow is OFF, the timing will stop, the timing coil and the timing value will maintain the current value.



5.4.3 TOF: Off-delay timing instruction

LAD: |—| |—[TOF *(D)* *(S)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: TOF *(D)* *(S)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
D	INT											T				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

D: Destination operand

S: Source operand

■ **Function description**

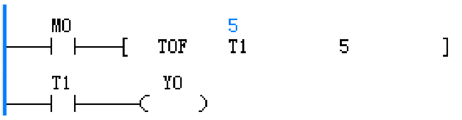
1. When the power flow changes from ON to OFF (falling edge), the designated timer T (***D***) will start timing.

2. When the power flow is OFF, if the designated timer T has started timing, it will keep timing until the timing value reaches the preset value (***S***). The timing coil output of the T element will be OFF, and the timing value will maintain at the preset value.

3. When the power flow input is OFF, if the timing has not started, the timing will not start.

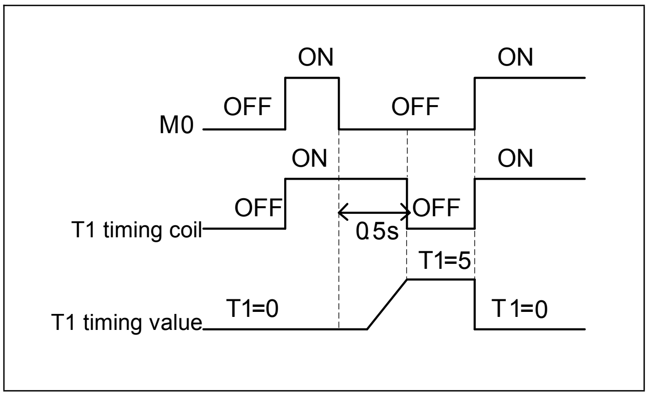
4. When the power flow is ON, the timing will stop, the timing value will be cleared, and the timing coil output is ON.

■ **Example**



```
LD   M0
TOF       T1
5
LD   T1
OUT  Y0
```

■ **Time sequence chart**



5.4.4 TMON: Monostable timing instruction

LAD: |—| |—[TMON *(D)* *(S)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: TMON *(D)* *(S)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
D	INT											T				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

D: Destination operand

S: Source operand

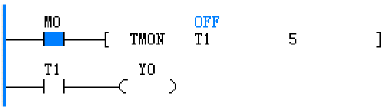
■ **Function description**

■ **Example**

1. When the input power flow changes from OFF to ON (rising edge), and the timing has not started, the designated timer T (***D***) will start timing based on the current value. In the timing status (whose length is determined by ***S***), the timing coil output will maintain ON.

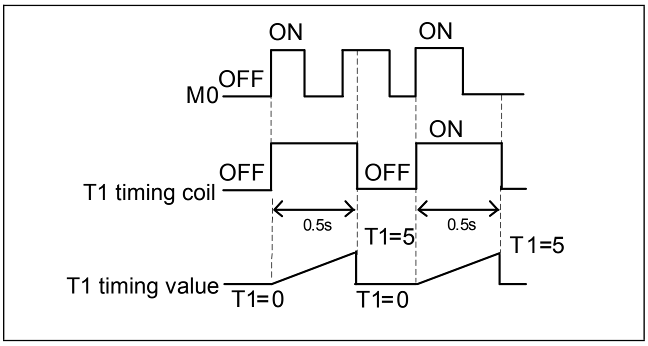
2. In the timing status (whose length is determined by ***S***), no matter how the power flow changes, the timing will keep going, and the timing coil output will keep ON.

3. When the timing value reaches the preset point, the timing will stop, the timing value will be cleared, and the timing coil output will be set OFF.



```
LD   M0
TMON   T1   5
LD   T1
OUT   Y0
```

- **Time sequence chart**



5.5 Counter instruction

CTU: 16-bit counter counting up instruction

LAD: ├─┤ ├─[CTU (D) (S)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: CTU *(D) (S)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
D	INT											C				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- **Operand description**

D: Destination operand

S: Source operand

- **Function description**

1. When the power flow changes from OFF to ON (rising edge), the 16-bit counter C (***D***) will count 1.

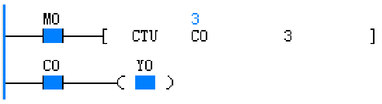
2. When the counting value reaches 32,767, it will maintain that value.

3. When the counting value is larger than or equal to the preset point (***S***), the counting coil will be set ON.

- **Note**

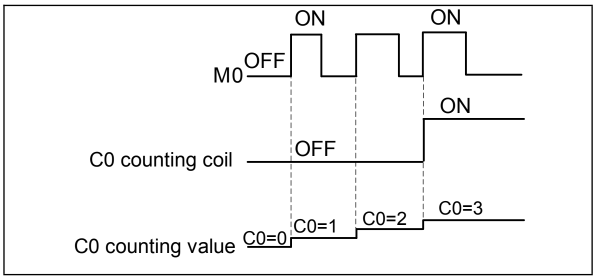
The address range of the 16-bit counter C (***D***): C0~C199.

- **Example**



```
LD   M0
CTU       C0
3
LD   C0
OUT   Y0
```

Time sequence chart



5.5.2 CTR: 16-bit counter loop cycle counting instruction

LAD: ├─┤ ├─[CTR (D) (S)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: CTR *(D) (S)*	**Program steps**	**5**

Operand	Type	Applicable elements	Indexed addressing

D	INT											C				
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

D: Destination operand

S: Source operand

■ **Function description**

1. When the power flow changes from OFF to ON (rising edge), the 16-bit counter C (***D***) will count 1.

2. When the counting value is equal to the preset point (***S***), the counting coil will be set ON.

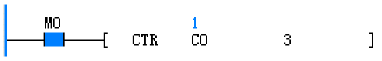
3. After the counting value reaches the preset point (***S***), if the power flow changes from OFF to ON again (rising edge), the counting value will be set to 1, and the counting coil will be set OFF.

■ **Note**

1. When the preset counting value (***S***) is less than or equal to 0, there will be no counting.

2. The address of the 16-bit counter C (***D***) shall be within C0~C199.

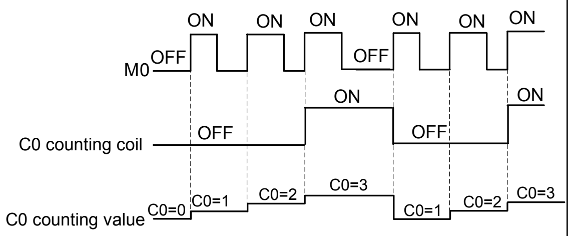
■ **Example**



LD M0

CTR C0 3

■ **Time sequence chart**



5.5.3 DCNT: 32-bit counting instruction

LAD: [DCNT *(D)* *(S)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DCNT *(D)* *(S)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
D	DINT											C				
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

D: Destination operand

S: Source operand

■ **Function description**

1. When the input power flow changes from OFF to ON (rising edge), the 32-bit counter C (***D***) will count up or down 1 (depending on the corresponding SM flag bit).

2. For a up counter, when the counting value is larger than or equal to the preset point (***S***), the counting coil will be set ON.

3. For a down counter, when the counting value is less than or equal to the preset point (***S***), the counting coil will be set OFF.

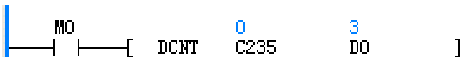
4. When the counting value is 2147483647, it will change to -2147483648 if the counter counts up once more.

5. When the counting value is -2147483648, it will change to 2147483647 if the counter counts down once more.

■ **Note**

■ **Example**

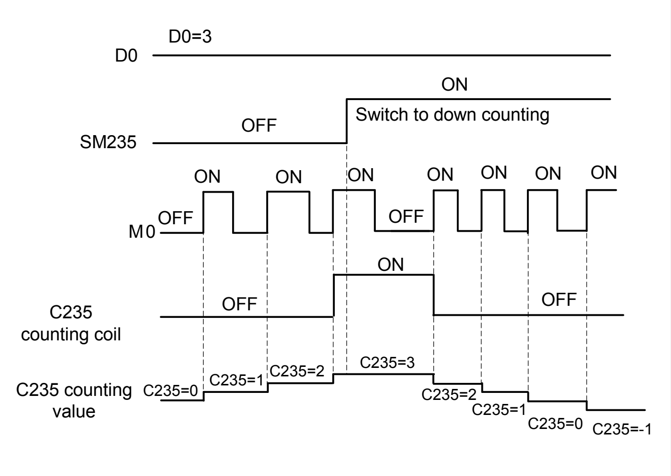
The address of the C element (***D***) shall be within C200~C235.



LD M0

DCNT C235 D0

- **Time sequence chart**



Chapter 6 Application instructions

This chapter introduces the application instructions of EC series small PLC, including the formats, operands, influenced flag bit, functions, examples and time sequence charts of the instructions.

6.1 Program flow control instruction

6.1.1 FOR: Cycle instruction

LAD: ├─┤ ├─[FOR (S)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: FOR (S)	**Program steps**	**3**

Operand	Type	Applicable elements														Indexed addressing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S: source operand

6.1.2 NEXT: Return from cycle

LAD: ├[NEXT]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: NEXT	**Program steps**	**1**

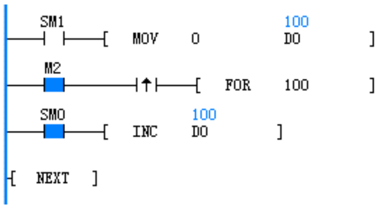
■ **Function description**

1. Instructions FOR and NEXT form a FOR-NEXT structure.

2. When the power flow before FOR is valid, and the cycle times (S) is larger than 0, the instructions in the FOR-NEXT structure will be cyclically executed S times. After that, the instructions after the FOR-NEXT structure will be executed.

3. If the power flow before FOR is invalid, or the cycle times (S) is less than or equal to 0, the program will skip over the instructions in the FOR-NEXT structure and execute the following instructions.

■ **Example**



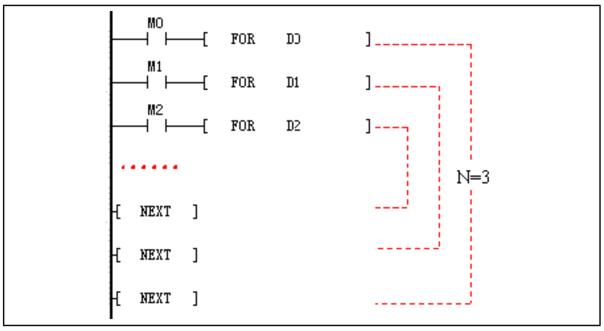
```
LD   SM1
MOV 0   D0
LD   M2
EU
FOR   100
LD   SM0
INC   D10
NEXT
```

The initial conditions for the operation are: D0=0, M2=OFF. When M2 changes from OFF to ON, the instructions in the FOR-NEXT structure will be consecutively executed for 100 times. D0 will increase one for each cycle. When the cycle is over, D0 reaches 100.

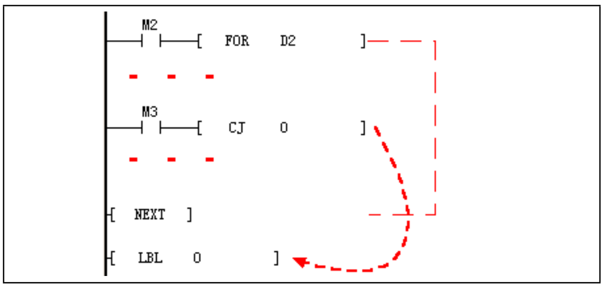
■ **Note**

1. The FOR-NEXT instruction must be used in pairs in a POU, or the program cannot pass the compiling.

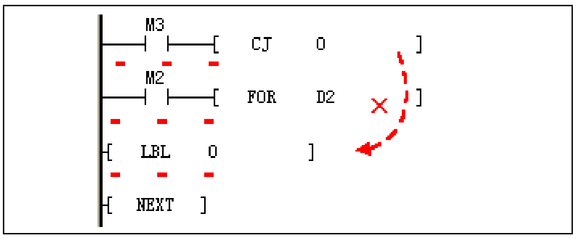
2. Nesting of several FOR-NEXT structures is supported. EC20 series PLC supports up to 8 levels of nesting. (The figure below shows a 3-level nesting of FOR-NEXT structure)



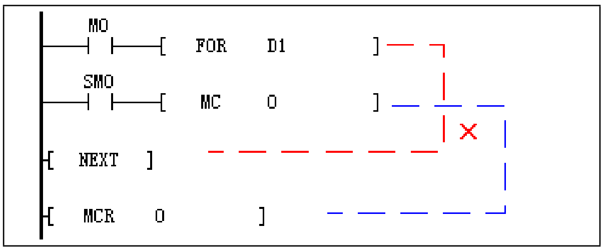
3. You can use the Conditional Jump (CJ) instruction to jump out of the structure and end the loop in advance, as shown in the following ladder diagram:



4. It is prohibited to use the CJ instruction to jump into a loop. The LAD program shown in the following figure cannot pass the compiling.



5. The crossing of the structures MC-MCR and FOR-NEXT is prohibited. LAD program shown in the following figure cannot pass the compiling.



Note

The execution of the FOR-NEXT structure is time consuming. The bigger the cycle times is, or the more instructions are contained in the loop, the longer it will take. To prevent the operation overtime error, use the WDT instruction in a time-consuming loop.

6.1.3 LBL: Jump label definition

LAD: -[LBL (S)]									**Applicable to**		**EC10 EC10A EC10V EC20 EC20H**			
									Influenced flag bit					
IL: LBL (S)									**Program steps**		**3**			
Operand	Type	Applicable elements												Indexed addressing
S	INT	Constant												

■ **Operand description**

S: label number

■ **Function description**

1. A label numbered S is defined.

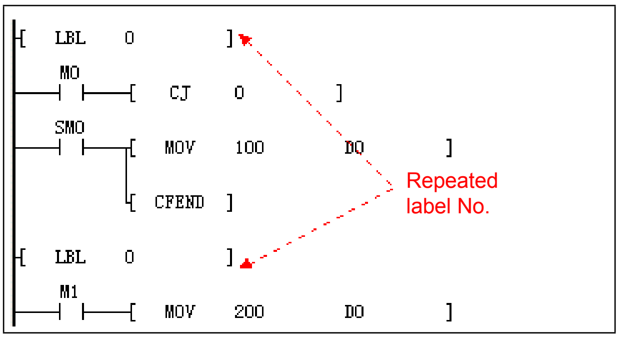
2. It is used to mark a specific jumping position for the CJ instruction.

■ **Note**

1. Range: 0≤S≤127

2. Take care not to mark two labels with the same No. in one POU, or the program cannot pass the compiling. However, you can do so in different POUs (for example, different sub-programs).

■ **Example of error program**



6.1.4 CJ: Conditional jump

<table>
<tr><td colspan="3" rowspan="2"><b>LAD:</b>
├──┤ ├──[CJ (S)]</td><td colspan="10"><b>Applicable to</b></td><td colspan="3"><b>EC10 EC10A EC10V EC20 EC20H</b></td></tr>
<tr><td colspan="10"><b>Influenced flag bit</b></td><td colspan="3"></td></tr>
<tr><td colspan="3"><b>IL: CJ (S)</b></td><td colspan="10"><b>Program steps</b></td><td colspan="3"><b>3</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

■ **Operand description**

S: label number

■ **Function description**

1. When the power flow is valid, the program will jump to execute the instruction numbered S.

2. If the power flow is invalid, the program will not jump, but execute the instruction following CJ.

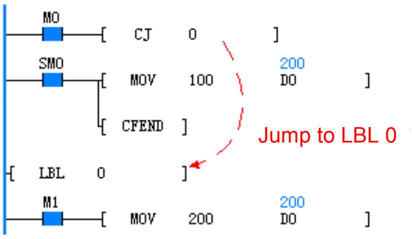
■ **Note**

1. The jumping label S (0≤S≤127) for the CJ instruction shall be a legal and defined label. Otherwise, the user program cannot pass the compiling.

2. It is not allowed to use the CJ instruction to jump into a FOR-NEXT structure.

3. It is allowable to use the CJ instruction to jump out of or into the MC-MCR structure or SFC status. However, such operation will damage the logic of the MC-MCR structure or SFC status and make the program complex. It is not recommended to do this.

■ **Example**



```
LD   M0
CJ   0
LD   SM0
MOV 100   D0
CFEND
LBL   0
LD   M1
MOV 200   D0
```

1. Initial conditions: M0=OFF, M1=ON. The CJ 0 instruction is not be executed, and D0 is 100. After executing CFEND, the current cycle of the main program ends in advance, and the following LD M1 and MOV 200 D0 instructions are not executed.

2. When M0 is ON, M1=ON, the program will execute the CJ 0 instruction, skip over the “MOV 100 200” and CFEND instructions, and jump to LBL 0 and execute “MOV 200 D0” instruction. D0 is 200 then.

6.1.5 CFEND: Conditional end from user main program

LAD: ├──┤ ├──[CFEND]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: CFEND	**Program steps**	**1**

■ **Function description**

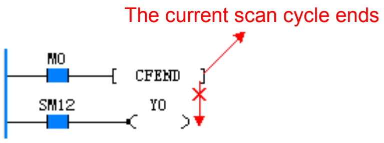
1. When the power flow of the instruction is valid, the current scan cycle of the main program ends immediately and the following instructions in the main program will not be executed.

2. When the power flow of the instruction is invalid, the instruction enables no action, and the instruction after it will be executed in order.

■ **Note**

The CFEND must be used in the main program, or the program cannot pass the compiling.

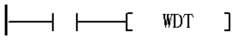
■ **Example**



```
LD   M0
CFEND
LD   SM12
OUT   Y0
```

When the program is running, if M0=OFF, the CFEND instruction will not enable any action. The following instructions LD SM12 and OUT Y0 will be executed. When M0 is ON, the CFEND instruction will be executed, the main program will end the current scan cycle immediately, and the following instructions will not be executed.

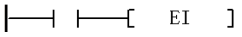
6.1.6 WDT: User program watchdog reset

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **WDT**	**Program steps**	**1**

- **Function description**

When the power flow is valid, the instruction will clear the user program watchdog, and the watchdog will restart timing.

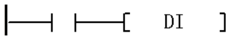
6.1.7 EI: Enable interrupt instruction

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **EI**	**Program steps**	**1**

- **Function description**

1. When the power flow of the EI instruction is valid, the interrupts in the current scan cycle will be enabled.
2. When the EI instruction is valid, the interrupt requests will be allowed to join the interrupt request queue to wait for system response.

6.1.8 DI: Disable interrupt instruction

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **DI**	**Program steps**	**1**

- **Function description**

1. When the power flow is valid, the global interrupt enable flag is inactive, that is, the global interrupt will be off.
2. When the global interrupt enable flag is inactive, the interrupt events will not generate any interrupt request.

- **Note**

When the DI instruction is valid, the system will still respond to the unprocessed interrupt requests in the request queue, but new interrupt events cannot generate interrupt requests.

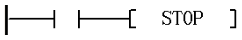
6.1.9 CIRET: Conditional return from user interrupt subprogram

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **CIRET**	**Program steps**	**1**

- **Function description**

When the power flow is valid, the system will quit the current interrupt program immediately.

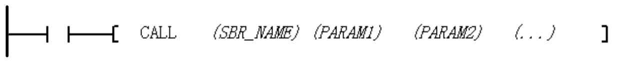
6.1.10 STOP: User program stop

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **STOP**	**Program steps**	**1**

- **Function description**

When the power flow is valid, the system will immediately stop the execution of the user program.

6.1.11 CALL: Calling a subprogram

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **CALL** ***(SBR name) (PARAM1) (PARAM2) …***	**Program steps**	**Determined by the subprogram parameters**

- **Function description**

When the power flow is valid, the system will call the designated subprogram, execute it, and then return to the main program to execute the instructions following the CALL instruction.

- **Note**

1. The subprogram called by the CALL instruction must be defined in advance in the user program, or the program cannot pass the compiling.

2. The operand element type in the CALL instruction must match the **Data Type** defined in the local variable table of the subprogram, or the program cannot pass the compiling.

The following examples demonstrates some illegal matches.

Example 1: In the local variable table of subprogram SBR1, the data type of Operand 1 is DINT/DWORD.

The following usages are illegal:

- CALL SBR1 Z0 (The data type of Z element cannot be DINT/DWORD)
- CALL SBR1 C199 (The data type of elements C0 to C199 cannot be DINT/DWORD)
- CALL SBR1 K2X0 (Kn addressing 1≤n≤3, the data type cannot be DINT/DWORD)

Example 2: In the local variable table of the SBR1 subprogram, the data type of Operand 1 is INT/WORD.

The following usages are illegal:

- CALL SBR1 C200 (The data type of element C200 to C255 cannot be INT/WORD)
- CALL SBR1 K2X0 (Kn addressing 4≤n≤8, the data type cannot be INT/WORD)

3. The operand element type in the CALL instruction must match the **Variable Type** defined in the local variable table in the subprogram, or the program will not pass the compiling.

The following examples demonstrates some illegal matches.

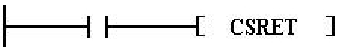
Example: In the local variable table of subprogram SBR1, the operand type of Operand 1 is OUT or IN_OUT.

The following usages are illegal:

- CALL SBR1 321 (constants cannot be changed, therefore it does not match OUT or IN_OUT)
- CALL SBR1 K4X0 (K4X0 is read-only, therefore it does not match OUT or IN_OUT)
- CALL SBR1 SD0 (SD0 is read-only, therefore it does not match OUT or IN_OUT)

4. The number of the operands in the CALL instruction must match the local variable table of the subprogram, or the program will not pass the compiling.

6.1.12 CSRET: Conditional return from user subprogram

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **CSRET**	**Program steps**	**1**

- **Function description**

When the power flow is valid, the program will quit the current subprogram and return to the upper level subprogram.

6.2 Data transmission instruction

6.2.1 MOV: Move word data transmission instruction

LAD: [MOV (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: MOV (S) (D)	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	R	√

- **Operand description**

S: source operand

D: destination operand

- **Function description**

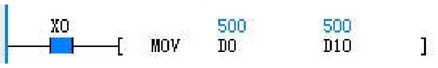
When the power flow is valid, the content of S is assigned to D, and the value of S remains unchanged.

- **Note**

1. The MOV instruction supports signed and unsigned integers. If the two operands are both elements, the data type is signed integer. If the source operand is a signed integer (for example, -10, +100), the destination operand is also a signed integer. If the source operand is an unsigned double integer (for example, 100, or 45535), the destination operand will also be an unsigned integer.

2. The corresponding element C only supports C0 to C199.

- **Example**



LD X0
MOV D0 D10

When X0 is ON, the content of D0 is assigned to D10, D10 = 500.

6.2.2 DMOV: Move double word data transmission instruction

LAD: [DMOV (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DMOV (S) (D)	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D	SD	C		V		R	√

- **Operand description**

S: source operand

D: destination operand

- **Function description**

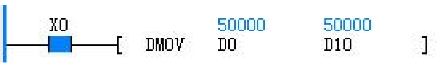
When the power flow is valid, the content of S is assigned to D, and the value of S remains unchanged.

- **Note**

1. The DMOV instruction supports signed and unsigned double integers. If the two operands of the instruction are elements, the data types are signed integers. If the source operand of the instruction is a signed double integer (for example, -10, +100), the destination operand will also be signed integer. If the source operand is the unsigned double integer (for example, 100, 45535), the destination operand will also be unsigned integer.

2. The corresponding element C only supports C200 to C255.

- **Example**



LD X0
DMOV D0 D10

When X0 is ON, the content of (D0, D1) is assigned to (D10, D11). (D10, D11) is 50000.

6.2.3 RMOV: Move floating point number data transmission

LAD: ├──┤ ├──[RMOV *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: RMOV *(S)* *(D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

■ **Operand description**

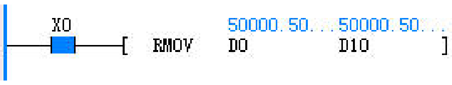
S: source operand

D: destination operand

■ **Function description**

When the power flow is valid, the content of S is assigned to D, and the value of S remains unchanged.

■ **Example**



LD X0

RMOV D0 D10

When X0 is ON, the content of (D0, D1) is assigned to (D10, D11). (D10, D11) is 50000.5.

6.2.4 BMOV: Move data block transmission instruction

LAD: ├──┤ ├──[BMOV *(S1)* *(D)* *(S2)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: BMOV *(S1)* *(D)* *(S2)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S: source operand, starting element of data block

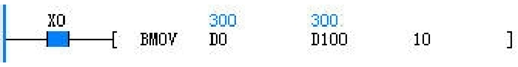
D: destination operand, starting element of data block

S2: size of data block

■ **Function description**

When the power flow is valid, the contents of S2 elements starting with S1 are assigned to the S2 elements starting with D, and the contents of S2 elements starting with S1 remain unchanged.

■ **Example**



LD X0

BMOV D0 D100 10

When X0 is ON, the contents of 10 elements starting with D0 are assigned to 10 elements starting with D100. D100=D0, D101=D1, ..., D109=D9.

6.2.5 FMOV: Fill data block instruction

LAD: ├──┤ ├──[FMOV *(S1)* *(D)* *(S2)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: FMOV *(S1)* *(D)* *(S2)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand, starting element of data block

D: destination operand, starting element of data block

S2: size of data block

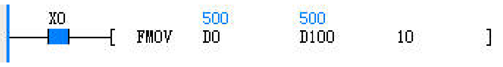
■ **Function description**

When the power flow is valid, the contents of S1 will be filled into S2 elements starting with D element, and the content of S1 remains unchanged.

■ **Note**

1. When S1, D and S2 use C element, the legal range is C0 to C199.
2. S2 is larger than or equal to 0.
3. When S1 and D both use Kn addressing, Kn shall be the same.

■ **Example**



LD X0
FMOV D0 D100 10

When X0 is ON, the content of D0 will be filled into 10 elements starting with D100. D100=D101= ... =D109=D0=500.

6.2.6 DFMOV: Fill data block double word instruction

LAD: ├──┤ ├──[DFMOV *(S1)* *(D)* *(S2)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DFMOV *(S1)* *(D)* *(S2)*	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand

D: destination operand, starting element of data block

S2: size of data block

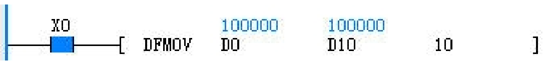
■ **Function description**

When the power flow is valid, the contents of S1 will be filled into S2 elements starting with D, and the content of S1 remains unchanged.

■ **Note**

1. When S1, D and S2 use C element, the legal range is C200 to C255.
2. S2 is larger than or equal to 0.
3. When S1 and D are both Kn addressing, Kn shall be the same.

■ **Example**



LD X0
DFMOV D0 D10 10

When X0 is ON, the content of (D0, D1) will be filled into 10×2 units starting with D10. (D10, D11)=(D12, D13)=...=(D28, D29)=(D0, D1)=100000.

6.2.7 SWAP: Swap bytes

LAD: [SWAP (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: SWAP *(D)*	**Program steps**	**3**

Operand	Type	Applicable elements														Indexed addressing
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

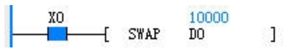
- **Operand description**

D: destination operand, the word element whose high/low bytes are swapped.

- **Function description**

When the power flow is valid, the D element whose high/low bytes has been swapped will be saved.

- **Example**



```
LD   X0
SWAP   D0
```

When X0 is ON, the high/low bytes in D0=0x1027 (4135) will be swapped and saved. D0 is then 0x2710 (10000).

6.2.8 XCH: Exchange word

LAD: [XCH (D1) (D2)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: XCH *(D1)* *(D2)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
D1	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
D2	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- **Operand description**

D1: destination operand1

D2: destination operand2

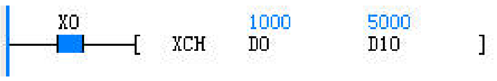
- **Function description**

When the power flow is valid, D1 and D2 will exchange their values.

- **Note**

When using the Kn addressing mode, the Kn in D1 and D2 shall be the same.

- **Example**



```
LD   X0
XCH D0   D10
```

When X0 is ON, D0 and D10 will exchange their values. Before the execution, D0 is 5000 and D10 is 1000. After the execution, D0 is 1000 and D10 is 5000

6.2.9 DXCH: Exchange double word instruction

LAD: DXCH (D1) (D2)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **DXCH** ***(D1) (D2)***	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
D1	DINT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
D2	DINT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

D1: destination operand1

D2: destination operand2

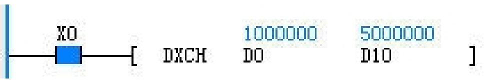
■ **Function description**

When the power flow is valid, D1 and D2 will exchange their values.

■ **Note**

When using the Kn addressing mode, the Kn in D1 and D2 shall be the same.

■ **Example**



LD X0
DXCH D0 D10

When X0 is ON, (D0,D1) and (D10,D11) will exchange their values. Before the execution and (D0, D1) is 5000000, (D10, D11) is 1000000. After the execution, (D0, D1) is 1000000 and (D10, D11) is 5000000.

6.2.10 PUSH: Push instruction

LAD: PUSH (S1) (D) (S2)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **PUSH** ***(S1) (D) (S2)***	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: push value

D: the number of elements in the stack. It is also the element at the stack bottom.

S2: stack size

■ **Function description**

1. When the power flow is valid, the value of S1 will be pushed onto the top of the stack with D element as the bottom, and D will increase by 1. At this time, the address of the stack top unit is the address of D plus the value of D.

2. When the value of D reaches S2, one more push instruction will set the operation carry flag (SM181) to 1, and the push operation will not be executed.

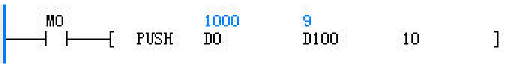
■ **Note**

1. When the stack is illegal (for example, when the stack size≤0, the number of elements in the stack<0, or when the stack size is beyond the limit), the system will report "Definition error of stack operated".

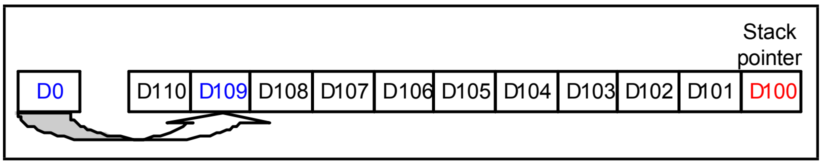
2. The stack size does not include the stack bottom element (the element designated by D).

3. S2 indicates the stack size, range＞0.

■ **Example**



LD M0
PUSH D0 D100 10



1. When M0 is ON, push D0 into the stack with D100 as the stack bottom.

2. Before the execution, D0 is 1000, D100 is 8 and D109 is 0.

3. After the execution, D0 is 1000, D100 is 9 and D109 is 1000.

6.2.11 FIFO: First-in-first-out instruction

LAD: ├──┤ ├──[FIFO (D1) (D2) (S)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **FIFO *(D1) (D2) (S)***	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
D1	INT								D				V		R	√
D2	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

D1: the number of elements in the stack. Its element address plus 1 is the address of the stack head.

D2*:* storage register for popped value

S: queue size

■ **Function description**

1. When the power flow is valid, the value of the stack head (the element immediately following D1) with D1 as the queue head is assigned to D2. At the same time, the value of D1 subtracts 1, the contents of the S units after D1 will move forward, and the last unit is filled with 0.

2. When D1 is 0, it indicates that the stack is empty, the zero flag (SM180) will be set 1.

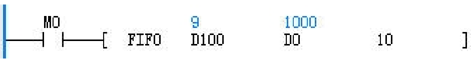
■ **Note**

1. When the stack is illegal (for example, when the stack size≤0, the number of elements in the stack<0, or when the stack size is beyond the limit), the system will report “Definition error of stack operated”.

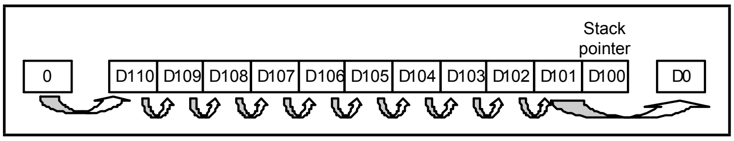
2. The stack size does not include the stack bottom element (the element designated by D1)

3. S indicates the stack size, range>0.

■ **Example**



LD M0
FIFO D100 D0 10



1. When M0 is ON, the content of D101 is filled into D0, and at the same time the contents of D101~D110 move forward, and the D110 is filled with 0.

2. Before the execution: D0=0, D100=10, D101=1000, D102=2000, ..., D109=9000, D110=10000.

3. After the execution: D0=1000, D100=9, D101=2000, D102=3000,..., D109=10000, D110=0.

6.2.12 LIFO: Last-in-first-out instruction

LAD: ├──┤ ├──[LIFO (D1) (D2) (S)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: **LIFO** ***(D1) (D2) (S)***	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
D1	INT								D				V		R	√
D2	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

D1: the number of elements in the queue. Its element address plus 1 is the address of the queue's head.

D2*:* storage register for popped value

S: queue size

■ **Function description**

1. When the power flow is valid, the value of the stack head with D1 as the stack bottom is assigned to D2, and at the same time the value of D1 subtracts 1.

2. When D1 is 0, it indicates that the stack is empty, the zero flag (SM180) will be set 1.

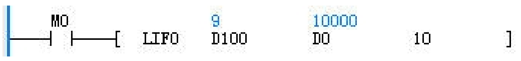
■ **Note**

1. When the stack is illegal (for example, when the stack size≤0, the number of elements in the stack<0, or when the stack size is beyond the limit), the system will report "Definition error of stack operated".

2. The stack size does not include the stack bottom element (the element designated by D1)

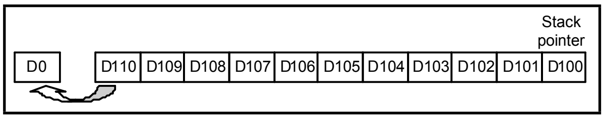
3. S indicates the stack size, range＞0.

■ **Example**



LD M0

LIFO D100 D0 10



1. When M0 is ON, the content of D110 is assigned to D0, the content of units D101~D110 remain unchanged.

2. Before the execution: D0=0, D100=10, D101=1000, D102=2000, ..., D109=9000, D110=10000.

3. After the execution: D0=10000, D100=9, D101=1000, D102=2000, ..., D109=9000, D110=10000.

6.2.13 WSFR: Shift right word instruction

LAD: ├─┤ ├──[WSFR (S1) (D) (S2) (S3)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: **WSFR** ***(S1) (D) (S2) (S3)***	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand

D: destination operand, starting unit of word string

S2: size of destination word queue

S3: number of words filled rightward

■ **Function description**

When the power flow is valid, the contents of S2 units starting with D unit will move rightward S3 words. The rightmost S3 units will be discarded. At the same time, the contents of S3 units starting with S1 will be filled into the left end of the word string.

■ **Note**

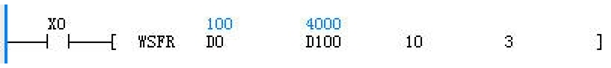
1. The elements with smaller SN are at the right, and the elements with larger SN are at the left.

2. S2≥0, S3≥0.

3. S2≥S3.

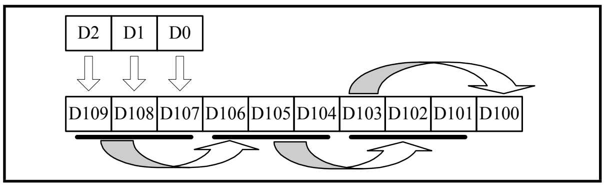
4. When S1 and D both use Kn addressing, Kn shall be the same.

■ **Example**



LD X0

WSFR D0 D100 10 3



1. When M0 is ON, the contents of 10 units starting with D100 unit will move rightward 3 words. The rightmost units D102~D100 will be discarded. At the same time, the contents of the 3 units starting with D0 will be filled into the left end of the word string.

2. Before the execution: D2=300, D1=200, D0=100. D109=10000, D108=9000, D107=8000, D106=7000, D105=6000, D104=5000, D103=4000, D102=3000, D101=2000, D100=1000.

3. After the execution: D0~D2 remain unchanged, D2=300, D1=200, D0=100. D109=300, D108=200, D107=100, D106=10000, D105=9000, D104=8000, D103=7000, D102=6000, D101=5000, D100=4000.

6.2.14 WSFL: Shift left word instruction

LAD: ├─┤ ├──[WSFL *(S1)* *(D)* *(S2)* *(S3)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: **WSFL *(S1) (D) (S2) (S3)***	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- **Operand description**

S1: source operand

D: destination operand, starting unit of word string

S2: size of destination word queue

S3: number of words filled for right forward

- **Function description**

When the power flow is valid, the contents of S2 units starting with D unit will move leftward S3 words. The leftmost S3 units will be discarded. At the same time, the contents of S3 units starting with S1 will be filled into the right end of the word string.

- **Note**

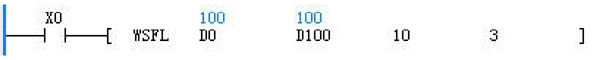
1. The elements with smaller SN are at the right, and the elements with larger SN are at the left.

2. S2≥0, S3≥0.

3. S2≥S3.

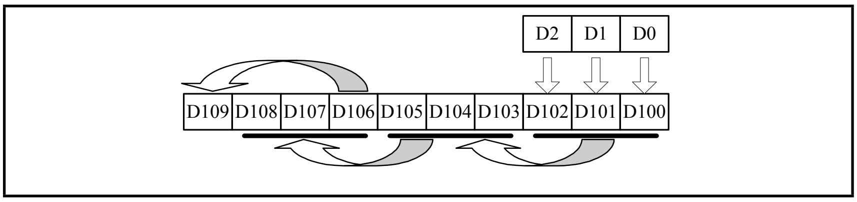
4. When S1 and D both use Kn addressing, Kn shall be the same.

- **Example**



LD X0

WSFL D0 D100 10 3



1. When X0 is ON, the contents of 10 units starting with D100 will move leftward 3 words. The leftmost units D109~D107 will be discarded. At the same time, the contents of the 3 units starting with D0 will be filled into the right end of the word string.

2. Before the execution: D0=100, D1=200, D2=300. D109=10000, D108=9000, D107=8000, D106=7000, D105=6000, D104=5000, D103=4000, D102=3000, D101=2000, D100=1000.

3. After the execution: D0~D2 remain unchanged: D2=300, D1=200, D0=100. D109=7000, D108=6000, D107=5000, D106=4000, D105=3000, D104=2000, D103=1000, D102=300, D101=200, D100=100.

6.3 Integer math instruction

6.3.1 ADD: Add integer instruction

LAD: ├──┤ ├──[ADD *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: ADD *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

S2: source operand2

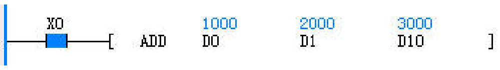
D: destination operand

■ **Function description**

1. When the power flow is valid, add S1 and S2, and assign the operation result to D.

2. When the operation result (D) is larger than 32767, the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set. When the operation result is less than -32768, the borrow flag bit (SM182) will be set.

■ **Example**



LD X0

ADD D0 D1 D10

When X0 is ON, add D0 (1000) and D1 (2000), and assign the result to D10, D10=3000.

6.3.2 SUB: Subtract integer instruction

LAD: ├──┤ ├──[SUB *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: SUB *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

S2: source operand2

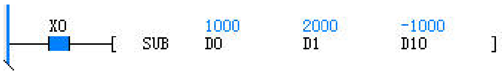
D: destination operand

■ **Function description**

1. When the power flow is valid, S1 subtracts S2, and the operation result is assigned to D.

2. When the operation result (D) is larger than 32767, the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set. When the operation result is less than -32768, the borrow flag will be set bit (SM182).

■ **Example**



LD X0

SUB D0 D1 D10

When X0 is ON, D0 (1000) subtracts D1 (2000), and the result -1000 is assigned to D10.

6.3.3 MUL: Multiply integer instruction

LAD: ├─┤ ├─[MUL *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: MUL *(S1) (S2) (D)*	**Program steps**	**8**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- **Operand description**

S1: source operand1

S2: source operand2

D: destination operand

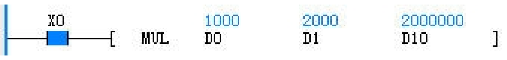
- **Function description**

When the power flow is valid, S1 multiplies S2, and the operation result is assigned to D.

- **Note**

The operation result of MUL instruction is a 32-bit data.

- **Example**



LD X0

MUL D0 D1 D10

When X0 is ON, D0 (1000) multiplies D1 (2000), and the result 2000000 is assigned to (D10, D11).

6.3.4 DIV: Divide integer instruction

LAD: ├─┤ ├─[DIV *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DIV *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- **Operand description**

S1: source operand1

S2: source operand2

D: destination operand

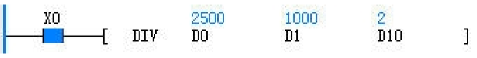
- **Function description**

When the power flow is valid, S1 is divided by S2, and the operation result is assigned to D (D includes 2 units, one storing the quotient, the other storing the remainder).

- **Note**

S2≠0, otherwise, the system will report “Divided by 0 error”, and the instruction will not be executed.

- **Example**



LD X0

DIV D0 D1 D10

When X0 is ON, D0 (2500) is divided by D1 (1000), the result is assigned to (D10, D11). D10=2, D11=500.

6.3.5 SQT: Square root integer instructions

LAD: |—| |—[SQT *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: SQT *(S)* *(D)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

S: source operand

D: destination operand

■ **Function description**

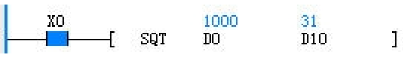
1. When the power flow is valid, S is extracted, and the operation result is assigned to D.

2. When the operation result (D) is 0, the zero flag bit (SM180) will be set. When the operation result rounds off the decimal fraction, the borrow flag bit (SM182) will be set.

■ **Note**

S≥0, otherwise, the system will report operand error, and the instruction will not be executed.

■ **Example**



LD X0
SQT D0 D10

When X0 is ON, extract D0 (1000), and assign the result to D10, D10=31.

6.3.6 INC: Increment integer instruction

LAD: |—| |—[INC *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: INC *(D)*	**Program steps**	**3**

Operand	Type	Applicable elements														Indexed addressing
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

D: destination operand

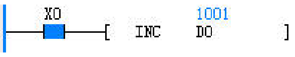
■ **Function description**

When the power flow is valid, D increases by 1.

■ **Note**

This instruction is a cyclic increase instruction. Range: -32768~32767. The supported range of C element: C0~C199.

■ **Example**



LD X0
INC D0

When X0 is ON, D0 (1000) is increased by 1. After the execution, D0 is 1001.

6.3.7 DEC: Decrement integer instruction

LAD: 	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DEC (*D*)	**Program steps**	**3**

Operand	Type	Applicable elements														Indexed addressing
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- **Operand description**

D: destination operand

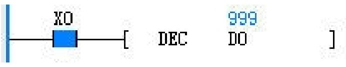
- **Function description**

When the power flow is valid, D decreases 1.

- **Note**

This instruction is a cyclic decrease instruction, with the range of -32768~32767.

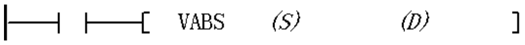
- **Example**



```
LD   X0
DEC  D0
```

When X0 is ON, D0 (1000) decreases 1. After the execution, D0=999.

6.3.8 VABS: Integer absolute value instruction

LAD:  	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: VABS (*S*) (*D*)	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- **Operand description**

S: source operand

D: destination operand

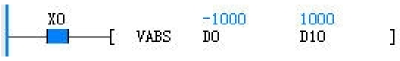
- **Function description**

When the power flow is valid, get the absolute value of S and assign it to D.

- **Note**

The range of S shall be -32767~32767. When S is -32768, the system will report operand error, and the instruction will not be executed.

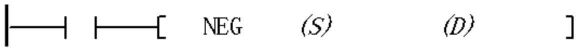
- **Example**



```
LD    X0
VABS  D0  D10
```

When X0 is ON, get the absolute value of D0 (-1000), and assign the result to D10. D10=1000.

6.3.9 NEG: Negative integer instruction

LAD:  	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: NEG (*S*) (*D*)	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- **Operand description**

S: source operand

D: destination operand

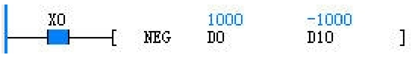
- **Function description**

When the power flow is valid, get the negative value of S and assign the result to D.

- **Note**

The range of S shall be -32767~32767. When S is -32768, the system will report operand error, and the instruction will not be executed.

- **Example**



LD X0

NEG D0 D10

When X0 is ON, get the negative value of D0 (1000) and assign the result to D10. D10=-1000.

6.3.10 DADD: Add double integer instruction

LAD: ├─┤ ├─[DADD *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DADD ***(S1) (S2) (D)***	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

Operand description

S1: source operand1

S2: source operand2

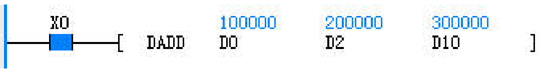
D: destination operand

Function description

1. When the power flow is valid, add S1 and S2, and assign the operation result to D.

2. When the operation result (D)>2147483647, the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set. When the operation result<-2147483648, the borrow flag bit (SM182) will be set.

Example



LD X0

DADD D0 D2 D10

When X0 is ON, add the value (100000) of (D0, D1) and the value (200000) of (D2, D3), and assign the result to (D10, D11). (D10, D11)=300000.

6.3.11 DSUB: Subtract double integer instruction

LAD: ├─┤ ├─[DSUB *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DSUB ***(S1) (S2) (D)***	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

S1: source operand1

S2: source operand2

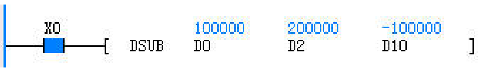
D: destination operand

■ **Function description**

1. When the power flow is valid, S1 subtracts S2, and the operation result is assigned to D.

2. When the operation result (D)>2147483647, the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set. When the operation result<-2147483648, the borrow flag bit (SM182) will be set.

■ **Example**



LD X0

DSUB D0 D2 D10

When X0 is ON, the value (100000) of (D0, D1) subtracts the value (200000) of (D2,D3), and the result -100000 is assigned to (D10, D11).

6.3.12 DMUL: Multiply double integer instruction

LAD: ├──┤ ├──[DMUL (S1) (S2) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DMUL *(S1) (S2) (D)*	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

S1: source operand1

S2: source operand2

D: destination operand

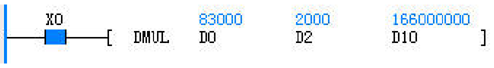
■ **Function description**

When the power flow is valid, S1 multiplies S2, and the result is assigned to D.

■ **Note**

The result of the DMUL instruction is a 32-bit data, and overflow may occur.

■ **Example**



LD X0

DMUL D0 D2 D10

When X0 is ON, the value (83000) of (D0, D1) multiplies the value (2000) of (D2,D3), and the result 1660000000 is assigned to (D10, D11).

6.3.13 DDIV: Divide double integer instruction

LAD: ├──┤ ├──[DDIV (S1) (S2) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DDIV *(S1) (S2) (D)*	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

S1: source operand1

S2: source operand2

D: destination operand

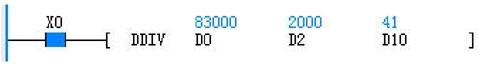
■ **Function description**

When the power flow is valid, S1 is divided by S2, and the operation result is assigned to D (D includes 4 units, with the first two storing the quotient, the other two storing the remainder).

■ **Note**

S2≠0, otherwise, the system will report “Divided by 0 error”, and the instruction will not be executed.

■ **Example**



LD X0

DDIV D0 D2 D10

When X0 is ON, the value (83000) of (D0, D1) is divided by the value (2000) of (D2, D3), and the result is assigned to (D10, D11) and (D12,D13). (D10, D11)=41, (D12, D13)=1000.

6.3.14 DSQT: Square root double integer instruction

LAD: |—| |—[DSQT (S) (D)]	Applicable to	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: DSQT (S) (D)	Program steps	7

Operand	Type	Applicable elements														Indexed addressing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- **Operand description**

S: source operand

D: destination operand

- **Function description**

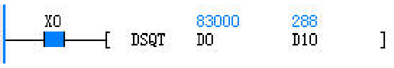
1. When the power flow is valid, S is extracted, and the operation result is assigned to D.

2. When the operation result (D) is 0, the zero flag bit (SM180) will be set. When the operation result rounds off the decimal fraction, the borrow flag bit (SM182) will be set.

- **Note**

S≥0, otherwise, the system will report operand error, and the instruction will not be executed.

- **Example**



LD X0

DSQT D0 D10

When X0 is ON, extract the value (83000) of (D0, D1), and assign the result to (D10, D11). (D10, D11)=288.

6.3.15 DINC: Increment double integer instruction

LAD: |—| |—[DINC (D)]	Applicable to	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: DINC (D)	Program steps	4

Operand	Type	Applicable elements														Indexed addressing
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- **Operand description**

D: destination operand

- **Function description**

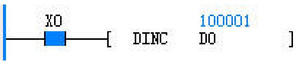
When the power flow is valid, D increases 1.

- **Note**

1. This instruction is a cyclic increase instruction. Range: -2147483648~2147483647.

2. The supported range of C element: C200~C255.

- **Example**



LD X0

DINC D0

When X0 is ON, the value (100000) of (D0, D1) increases 1. After the execution, (D0, D1)=100001.

6.3.16 DDEC: Decrement double integer instruction

LAD: ├──┤ ├──[DDEC *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**	
	Influenced flag bit		
IL: DDEC *(D)*	**Program steps**	**4**	

Operand	Type	Applicable elements													Indexed addressing
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- ■ **Operand description**

D: destination operand

- ■ **Function description**

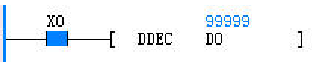
When the power flow is valid, D decreases 1.

- ■ **Note**

This instruction is a cyclic decrease instruction.

Range: -2147483648~2147483647.

- ■ **Example**



LD X0

DDEC D0

When X0 is ON, the value (100000) of (D0, D1) decreases 1. After the execution, (D0, D1)=99999.

6.3.17 DVABS: Double integer absolute value instruction

LAD: ├──┤ ├──[DVABS *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**	
	Influenced flag bit	**Zero, carry, borrow**	
IL: DVABS *(S)* *(D)*	**Program steps**	**7**	

Operand	Type	Applicable elements													Indexed addressing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- ■ **Operand description**

S: source operand

D: destination operand

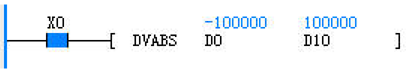
- ■ **Function description**

When the power flow is valid, get the absolute value of S and assign the result to D.

- ■ **Note**

The range of S shall be -2147483647~2147483647. When S is -2147483648, the system will report operand error, and the instruction will not be executed.

- ■ **Example**



LD X0

DVABS D0 D10

When X0 is ON, get the absolute value (-100000) of (D0, D1) and assign the result to (D10, D11). (D10, D11)=100000.

6.3.18 DNEG: Negative double integer instruction

LAD: ├──┤ ├──[DNEG *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DNEG *(S)* *(D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

S: source operand

D: destination operand

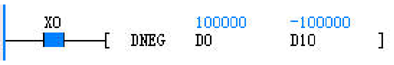
■ **Function description**

When the power flow is valid, get the negative value of S and assign the result to D.

■ **Note**

The range of S shall be -2147483647~2147483647. When the value of S is -2147483648, the system will report operand error, and the instruction will not be executed.

■ **Example**



```
LD    X0
DNEG  D0  D10
```

When X0 is ON, get the negative value (100000) of (D0, D1), and assign the result to (D10, D11). (D10, D11)=-100000.

6.3.19 SUM: Sum integer instruction

LAD: ├──┤ ├──[SUM *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: SUM *(S1)* *(S2)* *(D)*	**Program steps**	**8**

Operand	Type	Applicable elements														Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

S1: source operand, starting unit of summing

S2: source operand, number of units to be summed up

D: destination operand, summing result

■ **Function description**

When the power flow is valid, the contents of S2 units starting with the starting unit (S1) will be summed up, and the summing result is assigned to the D unit.

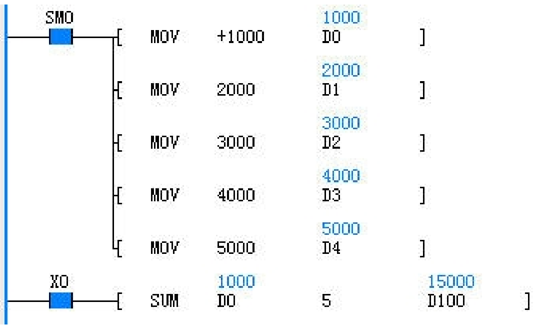
■ **Note**

1. The operation result of the SUM instruction is a 32-bit data.

2. 0≤S2≤255, or system will report operand error.

3. Since D is a 32-bit data, the carry and borrow flags are constantly 0, and the zero flag is determined by the final summing result.

■ **Example**



```
LD    SM0
MOV   1000   D0
MOV   2000   D1
MOV   3000   D2
MOV   4000   D3
MOV   5000   D4
LD    X0
SUM      D0      5
D100
```

When X0 is ON, the integers of 5 elements starting from D0 will be summed up, and the result is assigned to (D100, D101), (D100, D101)=D0+...+D4=15000.

6.3.20 DSUM: Sum double integer instruction

LAD: |—| |—[DSUM *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DSUM *(S1)* *(S2)* *(D)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	DINT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

S1: source operand, starting unit of summing

S2: source operand, number of data to be summed up

D: destination operand, summing result

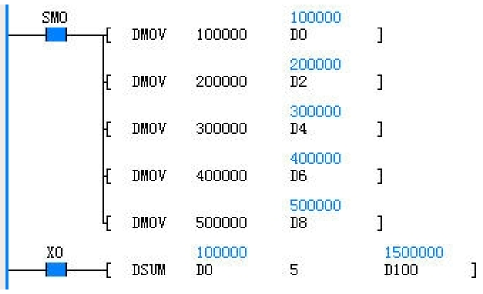
■ **Function description**

When the power flow is valid, the contents of S2×2 units starting with the starting unit (S1) will be summed up, and the summing result is assigned to the D unit.

■ **Note**

0≤S2≤255, or the system will report operand error.

■ **Example**



LD SM0
DMOV 100000 D0
DMOV 200000 D2
DMOV 300000 D4
DMOV 400000 D6
DMOV 500000 D8
LD X0
DSUM D0 5 D100

When X0 is ON, the double integers of 5×2 units starting with D0 will be summed up, and the result is assigned to (D100, D101).

(D100,D101)=(D0,D1)+ ...+(D8,D9)=1500000.

6.4 Floating-point arithmetic operation instruction

6.4.1 RADD: Add floating point number instruction

LAD: |—| |—[RADD *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: RADD *(S1)* *(S2)* *(D)*	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

■ **Operand description**

S1: source operand1

S2: source operand2

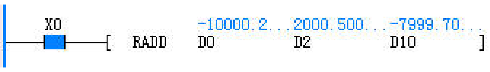
D: destination operand

■ **Function description**

1. When the power flow is valid, add S1 and S2, and assign the operation result to D.

2. When the operation result (D) is not within (-1.701412e+038)~(1.701412e+038), the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set.

■ **Example**



LD X0
RADD D0 D2 D10

When X0 is ON, add the value (-10000.2) of (D0, D1) and the value (2000.5) of (D2, D3), and the result -7999.7 is assigned to (D10, D11).

6.4.2 RSUB: Substract floating point number instruction

LAD: RSUB (S1) (S2) (D)	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: RSUB (S1) (S2) (D)	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- **Operand description**

S1: source operand1

S2: source operand2

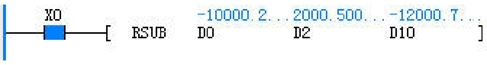
D: destination operand

- **Function description**

1. When the power flow is valid, S2 is subtracted from S1, and the operation result is assigned to D.

2. When the operation result (D) is not within (-1.701412e+038)~(1.701412e+038), the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set.

- **Example**



LD X0

RSUB D0 D2 D10

When X0 is ON, the value (2000.5) of (D2, D3) is subtracted from the value (-10000.2) of (D0, D1), and the result -12000.7 is assigned to (D10, D11).

6.4.3 RMUL: Multiply floating point number instruction

LAD: RMUL (S1) (S2) (D)	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: RMUL (S1) (S2) (D)	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- **Operand description**

S1: source operand1

S2: source operand2

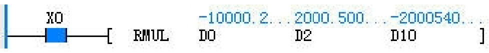
D: destination operand

- **Function description**

1. When the power flow is valid, S1 multiplies S2, and the operation result is assigned to D.

2. When the operation result (D) is not within (-1.701412e+038)~(1.701412e+038), the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set.

- **Example**



LD X0

RMUL D0 D2 D10

When X0 is ON, the value (-10000.2) of (D0, D1), multiplies the value (2000.5) of (D2, D3), and the result -20005400.0 is assigned to (D10, D11) (actually the product is -20005400.1, but is rounded off to the calculation precision).

6.4.4 RDIV: Divide floating point number instruction

LAD: ├──┤ ├──[RDIV *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: **RDIV *(S1)* *(S2)* *(D)***	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

■ **Operand description**

S1: source operand1

S2: source operand2

D: destination operand

■ **Function description**

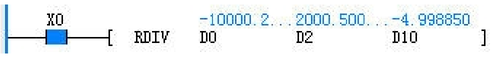
1. When the power flow is valid, S1 is divided by S2, and the operation result is assigned to D.

2. When the operation result (D) is not within (-1.701412e+038)~(1.701412e+038), the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set.

■ **Note**

S2≠0, or the system will report "Divided by 0 error", and the RDIV instruction will not be executed.

■ **Example**



LD X0

D2 D10

When X0 is ON, the value -10000.2 of (D0, D1) is divided by the value 2000.5 of (D2, D3), and the result -4.998850 is assigned to (D10, D11).

6.4.5 RSQT: Square root floating point number instruction

LAD: ├──┤ ├──[RSQT *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: **RSQT *(S)* *(D)***	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

■ **Operand description**

S: source operand

D: destination operand

■ **Function description**

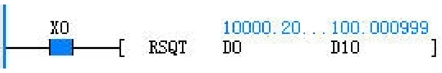
1. When the power flow is valid, S is extracted, and the operation result is assigned to D.

2. When the operation result (D) is 0, the zero flag bit (SM180) will be set.

■ **Note**

S≥0, or the system will report operand error, and the instruction will not be executed.

■ **Example**



LD X0

D10

When X0 is ON, extract the value (10000.2) of (D0, D1), and assign the result 100.000999 to (D10, D11).

6.4.6 RVABS: Floating point number absolute value instruction

LAD: ├──┤ ├──[RVABS (S) (D)]	Applicable to	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: RVABS *(S)* *(D)*	Program steps	7

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant						D				V		R	√
D	REAL							D				V		R	√

- **Operand description**

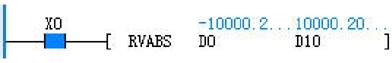
S: source operand

D: destination operand

- **Function description**

When the power flow is valid, get the absolute value of S and assign the value to D.

- **Example**



LD X0

RVABS D0 D10

When X0 is ON, get the absolute value (10000.2) of (D0, D1), and assign the result to (D10, D11).

6.4.7 RNEG: Negative floating point number instruction

LAD: ├──┤ ├──[RNEG (S) (D)]	Applicable to	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	Zero, carry, borrow
IL: RNEG *(S)* *(D)*	Program steps	7

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant						D				V		R	√
D	REAL							D				V		R	√

- **Operand description**

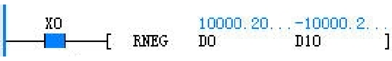
S: source operand

D: destination operand

- **Function description**

When the power flow is valid, get the negative value of S and assign the result to D.

- **Example**



LD X0

RNEG D0 D10

When X0 is ON, get the negative value -10000.2 of (D0, D1) and assign the result to (D10, D11).

6.4.8 SIN: Floating point number SIN instruction

LAD: ├──┤ ├──[SIN (S) (D)]	Applicable to	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	Zero, carry, borrow
IL: SIN *(S)* *(D)*	Program steps	7

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant						D				V		R	√
D	REAL							D				V		R	√

- **Operand description**

S: source operand

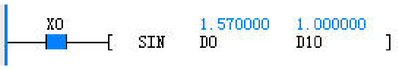
D: destination operand

- **Function description**

1. When the power flow is valid, get the SIN value of S (unit: radian), and assign the result to D.

2. When the operation result (D) is 0, the zero flag bit (SM180) will be set.

- **Example**



LD X0

SIN D0 D10

When X0 is ON, get the SIN value of (D0, D1)=1.57, and assign the value 1 to (D10, D11).

6.4.9 COS: Floating point number COS instruction

LAD:			
├──┤ ├──[COS *(S)* *(D)*]	**Applicable to**	**EC10 EC10V EC20 EC20H**	
	Influenced flag bit	**Zero, carry, borrow**	
IL: COS *(S)* *(D)*	**Program steps**	**7**	

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- **Operand description**

S: source operand

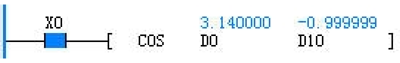
D: destination operand

- **Function description**

1. When the power flow is valid, get the COS value of S (unit: radian), and assign the result to D.

2. When the operation result (D) is 0, the zero flag bit (SM180) will be set.

- **Example**



LD X0

COS D0 D10

When X0 is ON, get the COS value of (D0, D1) 3.14, and assign the result -0.999999 to (D10, D11).

6.4.10 TAN: Floating point number TAN instruction

LAD:		
├──┤ ├──[TAN *(S)* *(D)*]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: TAN *(S)* *(D)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

- **Operand description**

S: source operand

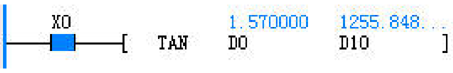
D: destination operand

- **Function description**

1. When the power flow is valid, get the TAN value of S (unit: radian), and assign the result to D.

2. When the operation result (D) is not within (-1.701412e+038)~(1.701412e +038), the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set.

- **Example**



LD X0

TAN D0 D10

When X0 is ON, get the TAN value of (D0, D1) 1.57, and assign the result 1255.848398 to (D10, D11).

6.4.11 POWER: Floating point number exponentiation instruction

LAD: ├──┤ ├──[POWER (S1) (S2) (D)]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: POWER *(S1) (S2) (D)*	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

■ **Operand description**

S1: source operand1

S2: source operand2

D: destination operand

■ **Function description**

1. When the power flow is valid, get the S2th power of S1, and assign the result to D.

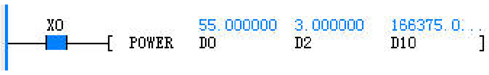
2. When the operation result (D) is not within (-1.701412e+038)~(1.701412e +038), the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set.

■ **Note**

1. When S1=0 and S2≤0, the system will report operand error, and the instruction will not be executed.

2. When S1<0 and the mantissa of S2 is not 0, the system will report operand error, and the instruction will not be executed.

■ **Example**



```
X0          55.000000  3.000000  166375.0...
──[ POWER   D0         D2        D10        ]
```

LD X0

POWER D0 D2 D10

When X0 is ON, get the (D2, D3)th power of (D0, D1) (i.e. $55.0^{3.0}$), and assign the result 166375.0 to (D10, D11).

6.4.12 LN: Floating point number LN instruction

LAD: ├──┤ ├──[LN (S) (D)]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: LN *(S) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

■ **Operand description**

S: source operand

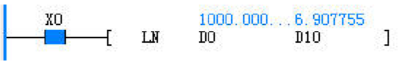
D: destination operand

■ **Function description**

1. When the power flow is valid, get the LN value of S1, and assign the result to D.

2. When the operation result (D) is not within (-1.701412e+038)~(1.701412e +038), the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set.

■ **Example**



```
X0       1000.000...6.907755
──[ LN   D0         D10      ]
```

LD X0

LN D0 D10

When X0 is ON, get the LN value of (D0, D1) 1000.0, and assign the result 6.907755 to (D10, D11).

6.4.13 EXP: Floating point number EXP instruction

LAD: ┠──┨ ┠──[EXP (S) (D)]	Applicable to	EC10 EC10V EC20 EC20H
	Influenced flag bit	Zero, carry, borrow
IL: EXP (S) (D)	Program steps	7

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V		R	√
D	REAL								D				V		R	√

■ **Operand description**

S: source operand

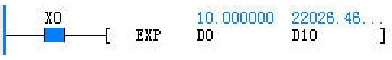
D: destination operand

■ **Function description**

1. When the power flow is valid, get the EXP value of S, and assign the result to D.

2. When the operation result (D) is not within (-1.701412e+038)~(1.701412e +038), the carry flag bit (SM181) will be set. When the operation result is 0, the zero flag bit (SM180) will be set.

■ **Example**



LD X0
EXP D0 D10

When X0 is ON, get the EXP value of (D0, D1) “10.0”, and assign the result 22026.464844 to (D10, D11).

6.4.14 RSUM: Sum floating point number instruction

LAD: ┠──┨ ┠──[RSUM (S1) (S2) (D)]	Applicable to	EC10 EC10V EC20 EC20H
	Influenced flag bit	Zero, carry, borrow
IL: RSUM (S1) (S2) (D)	Program steps	9

Operand	Type	Applicable elements													Indexed addressing
S1	REAL								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D				V		R	√
D	REAL								D				V		R	√

■ **Operand description**

S1: source operand, starting unit of summing

S2: source operand, number of units to be summed up

D: destination operand, summing result

■ **Function description**

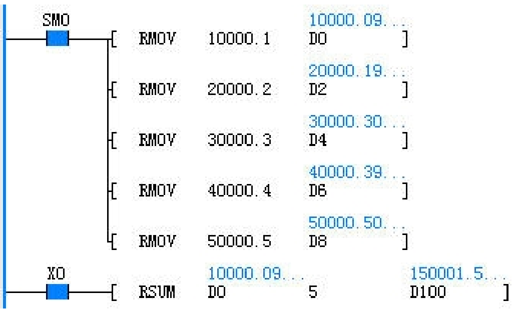
When the power flow is valid, the contents of S2×2 units starting with S1 will be summed up, and the floating point number summing result is assigned to the D unit.

■ **Note**

1. 0≤S2≤255, or the system will report operand error.

2. When overflow occurs, the summing operation will stop.

■ **Example**



LD SM0
RMOV 10000.1 D0
RMOV 20000.2 D2
RMOV 30000.3 D4
RMOV 40000.4 D6
RMOV 50000.5 D8
LD X0
RSUM D0 5 D100

When X0 is ON, the floating point numbers of the 5×2 units starting with D0 will be summed up, and the result is assigned to (D100, D101). (D100, D101)=(D0, D1) + ... + (D8, D9)=150001.5.

6.4.15 ASIN: Floating point number ASIN instruction

LAD: ┠──┨ ┠──[ASIN (S) (D)]	Applicable to	EC20H
	Influenced flag bit	Zero, carry, borrow
IL: ASIN (S) (D)	Program steps	7

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

■ **Operand description**

S: source operand

D: destination operand

■ **Function description**

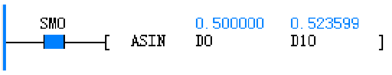
1. When the power flow is valid, get the SIN-1 value of S, and assign the result to D.

2. When the operation result (D) is 0, the zero flag bit (SM180) will be set.

■ **Note**

When S>1 or S<-1, the system will report operand error and will not execute the conversion. D will not change.

■ **Example**



LD SM0

ASIN D0 D10

When SM0 is ON, get the SIN-1 value of (D0, D1)(0.500000), and assign the result 0.523599 to (D10, D11).

6.4.16 ACOS: Floating point number ACOS instruction

LAD:	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: ACOS (*S*) (*D*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

■ **Operand description**

S: source operand

D: destination operand

■ **Function description**

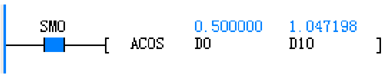
1. When the power flow is valid, get the COS^{-1} value of S, and assign the result to D.

2. When the operation result (D) is 0, the zero flag bit (SM180) will be set.

■ **Note**

When S>1 or S<-1, the system will report operand error and will not execute the conversion. ***D*** will not change.

■ **Example**



LD SM0

ACOS D0 D10

When SM0 is ON, get the COS-1 value of (D0, D1)(0.500000), and assign the result 1.047198 to (D10, D11).

6.4.17 ATAN: Floating point number ATAN instruction

LAD: ├──┤ ├──[ACOS *(S)* *(D)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: ATAN (*S*) (*D*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

■ **Operand description**

S: source operand

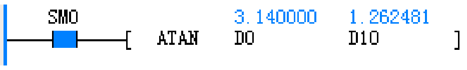
D: destination operand

■ **Function description**

1. When the power flow is valid, get the TAN-1 value of S, and assign the result to D.

2. When the operation result (D) is 0, the zero flag bit (SM180) will be set.

■ **Example**



LD SM0

ATAN D0 D10

When SM0 is ON, get the TAN-1 value of (D0, D1)(3.14), and assign the result 1.262481 to (D10, D11).

6.4.18 LOG: Floating point number LOG instruction

LAD: ├──┤ ├──[LOG *(S)* *(D)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: LOG (*S*) (*D*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

■ **Operand description**

S: source operand

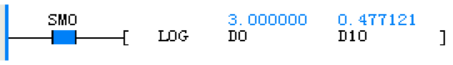
D: destination operand

■ **Function description**

1. When the power flow is valid, get the LOG value of S, and assign the result to D. LOG is a common logarithm to the base 10.

2. When the operation result (D) overflows, the carry (overflow) flag bit (SM181) will be set; when the operation result is 0, the zero flag bit (SM180) will be set.

■ **Example**



LD SM0

LOG D0 D10

When SM0 is ON, get (3.0) of D0(D1), and assign the result 0.477121 to D10(D11).

6.4.19 RAD: Floating point number angle->rad

LAD: ├──┤ ├──[RAD *(S)* *(D)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: RAD (*S*) (*D*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

■ **Operand description**

S: source operand

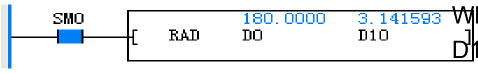
D: destination operand

■ **Function description**

1. When the power flow is valid, convert the floating point number angle of S to the radian, and assign the result to D.

2. When the operation result is 0, the zero flag bit (SM180) will be set.

■ **Example**



When SM0 is ON, get (180.0) of D0(D1), and assign the result 3.141593 to D10(D11).

LD SM0
RAD D0 D10

6.4.20 DEG: Floating point number rad->angle

LAD: ├──┤ ├──[DEG *(S)* *(D)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DEG (*S*) (*D*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S	REAL	Constant							D				V	R	√
D	REAL								D				V	R	√

■ **Operand description**

S: source operand

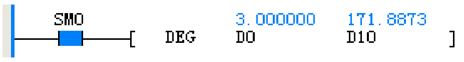
D: destination operand

■ **Function description**

1. When the power flow is valid, convert the floating point number radian of S to the angle, and assign the result to D.

2. When the operation result is 0, the zero flag bit (SM180) will be set; when the operation result overflows, the carry (overflow) flag bit (SM181) will be set.

■ **Example**



LD SM0
DEG D0 D10

When SM0 is ON, get (3.0) of D0(D1), and assign the result 171.8873 to D10(D11).

6.5 Data converting instruction

6.5.1 DTI: Double integer to integer instruction

LAD: ├──┤ ├──[DTI *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: **DTI (*S*) (*D*)**	**Program steps**	**6**

Operand	Type	Applicable elements														Indexed addressing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

S: source operand

D: destination operand

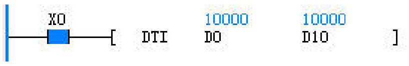
■ **Function description**

When the power flow is valid, S will be converted from double integer to integer, and the result is assigned to D.

■ **Note**

When S is not within -32768~32767, the system will report operand error and will not execute the conversion. D will not change.

■ **Example**



LD X0
DTI D0 D10

When X0 is ON, (D0, D1) 10000 will be converted from double integer to integer and the result 10000 is assigned to D10.

6.5.2 ITD: Integer to double integer instruction

LAD: ├─┤ ├─[ITD *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: ITD *(S)* *(D)*	**Program steps**	**6**

Operand	Type	Applicable elements														Indexed addressing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

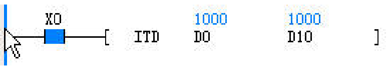
S: source operand

D: destination operand

■ **Function description**

When the power flow is valid, S will be converted from integer to double integer, and the result is assigned to D.

■ **Example**



```
LD   X0
ITD  D0  D10
```

When X0 is ON, D0 (1000) will be converted from integer to double integer, and the result 1000 is assigned to (D10, D11).

6.5.3 FLT: Integer to floating point number instruction

LAD: ├─┤ ├─[FLT *(S)* *(D)*]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: FLT *(S)* *(D)*	**Program steps**	**6**

Operand	Type	Applicable elements														Indexed addressing
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	REAL								D				V		R	√

■ **Operand description**

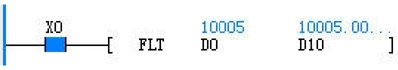
S: source operand

D: destination operand

■ **Function description**

When the power flow is valid, S will be converted from integer to floating point number, and the result is assigned to D.

■ **Example**



```
LD   X0
FLT  D0  D10
```

When X0 is ON, D0 (10005) will be converted from integer to floating point number, and the result 10005.0 is assigned to (D10, D11).

6.5.4 DFLT: Double integer to floating point number instruction

LAD: ├─┤ ├─[DFLT *(S)* *(D)*]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DFLT *(S)* *(D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	REAL								D				V		R	√

■ **Operand description**

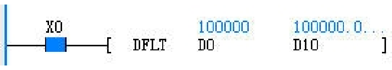
S: source operand

D: destination operand

■ **Function description**

When the power flow is valid, S will be converted from double integer to floating point number, and the result is assigned to D.

■ **Example**



```
LD    X0
DFLT  D0  D10
```

When X0 is ON, (D0, D1) 100000 will be converted from double integer to floating point number, and the result 100000.0 is assigned to (D10, D11).

6.5.5 INT: Floating point number to integer instruction

LAD: |—| |—[INT *(S)* *(D)*]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: INT *(S) (D)*	**Program steps**	**6**

Operand	Type	Applicable elements														Indexed addressing
S	REAL	Constant							D				V		R	√
D	INT			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- **Operand description**

S: source operand

D: destination operand

- **Function description**

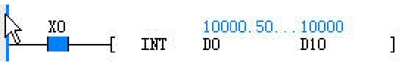
1. When the power flow is valid, S will be converted from floating point number to integer, and the result is assigned to D.

2. This instruction affects the zero flag and borrow flag. When the conversion result is 0, the zero flag will be set. When the result rounds off the decimal fraction, the borrow flag will be set. the carry (overflow) flag will be set.

- **Note**

When S>32767, D=32767. When S<-32768, D=-32768, and at the same time the carry (overflow) flag bit will be set.

- **Example**



X0

10000.50...10000

INT D0 D10]

LD X0

INT D0 D10

When X0 is ON, (D0, D1) 10000.5 will be converted from floating point number to integer and the result 10000 is assigned to D10.

6.5.6 DINT: Floating point number to double integer instruction

LAD: |—| |—[DINT *(S)* *(D)*]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DINT *(S) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	REAL	Constant							D				V		R	√
D	DINT			KnY	KnM	KnS	KnLM		D		C		V		R	√

- **Operand description**

S: source operand

D: destination operand

- **Function description**

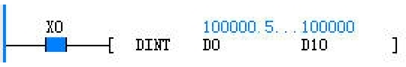
1. When the power flow is valid, S will be converted from floating point number to double integer, and the result is assigned to D.

2. When the conversion result is 0, the zero flag will be set. When the result rounds off the decimal fraction, the borrow flag will be set. When the result exceeds the range of the double integer, the carry (overflow) flag will be set.

- **Note**

When S>2147483647, D=2147483647. When S<-2147483648, D=-2147483648, and at the same time the carry (overflow) flag will be set.

- **Example**



X0

100000.5...100000

DINT D0 D10]

LD X0

DINT D0 D10

When X0 is ON, (D0, D1) 100000.5 will be converted from floating point number to double integer, and the result 100000 is assigned to (D10, D11).

6.5.7 BCD: Word to 16-bit BCD instruction

LAD: ├──┤ ├──[BCD (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: BCD *(S) (D)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

- **Operand description**

S: source operand, ≤9999

D: destination operand

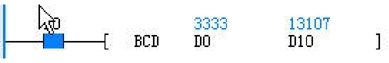
- **Function description**

When the power flow is valid, S will be converted from integer to 16-bit BCD code, and the result is assigned to D.

- **Note**

When S>9999, the system will report operand error and will not execute the instruction, and D will not change.

- **Example**



```
LD   X0
BCD  D0  D10
```

When X0 is ON, D0 0x0D05 (3333) will be converted from integer to 16-bit BCD code, and the result 0x3333 (13107) is assigned to D10.

6.5.8 DBCD: Double word to 32-bit BCD instruction

LAD: ├[DBCD (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DBCD *(S) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√

- **Operand description**

S: source operand, ≤99999999

D: destination operand

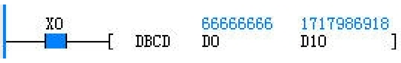
- **Function description**

When the power flow is valid, S will be converted from double integer to 32-bit BCD code, and the result is assigned to D.

- **Note**

When S>99999999, the system will report operand error and will not execute the instruction, and D will not change.

- **Example**



```
LD    X0
DBCD  D0  D10
```

When X0 is ON, (D0, D1) 0x3F940AA (66666666) will be converted from double integer to 32-bit BCD code, and the result 0x66666666 (1717986918) is assigned to (D10, D11).

6.5.9 BIN: 16-bit BCD to word instruction

LAD: ⊦[BIN (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: BIN (S) (D)	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

S: source operand, the data format of ***S*** must match the BCD code format

D: destination operand

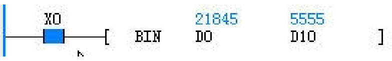
■ **Function description**

When the power flow is valid, S will be converted from 16-bit BCD code to integer, and the result is assigned to D.

■ **Note**

When the data format of S does not match the BCD code format, the system will reports illegal operand and will not execute the instruction, and D will not change.

■ **Example**



LD X0

BIN D0 D10

When X0 is ON, D0 0x5555 (21845) will be converted from 16-bit BCD code to integer, and the result 0x15B3 (5555) is assigned to D10.

6.5.10 DBIN: 32-bit BCD to double word instruction

LAD: ⊢⊣ ⊢⊣[DBIN (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DBIN (S) (D)	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

S: source operand

D: destination operand

■ **Function description**

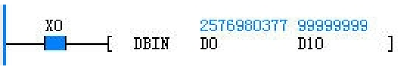
1. When the power flow is valid, S will be converted from 32-bit BCD code to double integer, and the result is assigned to D.

2. The data format of S must match the BCD code format.

■ **Note**

When the data format of S does not match the BCD code format, the system will report operand error and will not execute the instruction, and D will not change.

■ **Example**



LD X0

DBIN D0 D10

When X0 is ON, (D0, D1) 0x99999999 (2576980377) will be converted from 32-bit BCD code to double integer, and the result 0x5F5E0FF (99999999) is assigned to (D10, D11).

6.5.11 GRY: Word to 16-bit Gray code instruction

LAD: ├[GRY (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: GRY *(S) (D)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

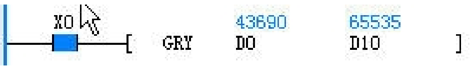
S: source operand

D: destination operand

■ **Function description**

When the power flow is valid, S will be converted from integer to 16-bit Gray code, and the result is assigned to D.

■ **Example**


X0 43690 65535
GRY D0 D10]

```
LD   X0
GRY  D0  D10
```

When X0 is ON, D0 0xAAAA (43690) will be converted form integer to 16-bit Gray code, and the result 0xFFFF (65535) is assigned to D10.

6.5.12 DGRY: Double word to 32-bit Gray code instruction

LAD: ├─┤ ├─[DGRY (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DGRY *(S) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

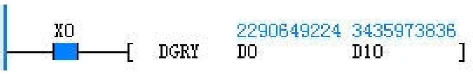
S: source operand

D: destination operand

■ **Function description**

When the power flow is valid, S will be converted from integer to 32-bit Gray code, and the result is assigned to D.

■ **Example**


X0 2290649224 3435973836
DGRY D0 D10]

```
LD    X0
DGRY  D0  D10
```

When X0 is ON, (D0, D1) 0x88888888 (2290649224) will be converted from double integer to 32-bit Gray code, and the result 0xCCCCCCCC (3435973836) is assigned to (D10, D11).

6.5.13 GBIN: 16-bit Gray code to word instruction

LAD: ├─┤ ├─[GBIN (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: GBIN *(S) (D)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

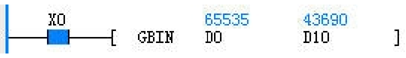
S: source operand

D: destination operand

■ **Function description**

When the power flow is valid, S will be converted from 16-bit Gray code to integer, and the result is assigned to D.

■ **Example**



LD X0

GBIN D0 D10

When X0 is ON, D0 0xFFFF (65535) will be converted from 16-bit Gray code to integer, and the result 0xAAAA (43690) is assigned to D10.

6.5.14 DGBIN: 32-bit Gray code to double word instruction

LAD: ├─┤ ├─[DGBIN *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DGBIN *(S)* *(D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√

■ **Operand description**

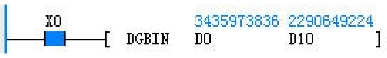
S: source operand

D: destination operand

■ **Function description**

When the power flow is valid, S will be converted from 32-bit Gray code to double integer, and the result is assigned to D.

■ **Example**



LD X0

DGBIN D0 D10

When X0 is ON, (D0, D1) 0xCCCCCCCC (3435973836) will be converted from 32-bit Gray code to double integer, and the result 0x88888888 (2290649224) is assigned to (D10, D11).

6.5.15 SEG: Word to 7-segment code instruction

LAD: ├─┤ ├─[SEG *(S)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: SEG *(S)* *(D)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

■ **Operand description**

S: source operand, ≤15

D: destination operand

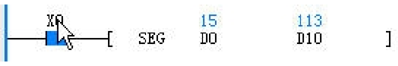
■ **Function description**

When the power flow is valid, S will be converted from integer to 7-segment code, and the result is assigned to D.

■ **Note**

When S>15, the system reports illegal operand and will not execute the instruction, and D will not change.

■ **Example**



LD X0

SEG D0 D10

When X0 is ON, D0 0x0F (15) will be converted from integer to 7-segment code, and the result 0x71 (113) is assigned to D10.

6.5.16 ASC: ASCII code conversion instruction

LAD:	Applicable to	EC10 EC10A EC10V EC20 EC20H
├─┤ ├──[ASC (S1˜S8) (D)]	Influenced flag bit	Zero, carry, borrow
IL: ASC (S1~S8) (D)	Program steps	19

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant														
S2	WORD	Constant														
S3	WORD	Constant														
S4	WORD	Constant														
S5	WORD	Constant														
S6	WORD	Constant														
S7	WORD	Constant														
S8	WORD	Constant														
D	WORD								D		C	T	V	Z	R	√

■ **Operand description**

S1~S8: source operand (If the number is less than 8, the remaining elements shall be filled with 0)

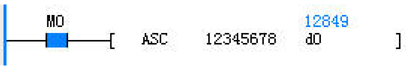
Only characters with ASCII code of 0x21~0x7E are supported (input through keyboard, if the number is less than 8, fill in with 0X00)

D: destination operand

■ **Function description**

When the power flow is valid, the string S1~S8 will be converted to ASCII code, and the result is assigned to the elements starting with D. When SM186 is OFF, the high/low byte of each D element will store two ASCII code data. When SM186 is ON, the low byte of each D element will store 1 ASCII code data.

■ **Example**



LD M0

ASC 12345678 D0

When M0 is ON, execute the ASCII conversion, and the data will be stored in two modes:

- When SM186 is OFF, the execution result is: D0=0x3231, D1=0x3433, D2=0x3635, D3=0x3837.
- When SM186 is ON, the execution result is: D0=0x31, D1=0x32, D2=0x33, D3=0x34, D4=0x35, D5=0x36, D6=0x37, D7=0x38.

6.5.17 ITA: Hexadecimal number-ASCII code conversion instruction

LAD:	Applicable to	EC10 EC10A EC10V EC20 EC20H
├─┤ ├──[ITA (S1) (D) (S2)]	Influenced flag bit	Zero, carry, borrow
IL: ITA (S1) (D) (S2)	Program steps	7

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: conversion source, hexadecimal number

D: destination operand.

S2: number of ASCII codes, 1≤***S2***≤256

■ **Function description**

When the power flow is valid, the hexadecimal number starting with S1 element will be converted to S2 ASCII codes, and the result is assigned to the elements starting with D. When SM186 is OFF, the high/low byte of each D element will store two ASCII code data. When SM186 is ON, the low byte of each D element will store 1 ASCII code data.

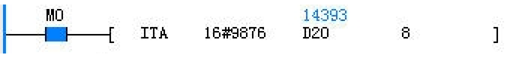
■ **Note**

1. When S1 and D use Kn addressing, Kn=4.

2. When S2 is not within 1~256, the system will report operand error and will not execute the instruction, and D will not change.

3. If S1 is a constant, S2 will be regarded as 4 by default when S2≥4, and the system will not report operand error.

■ **Example**



Source data: 0x9876

LD M0

ITA 16#9876 D20 8

When M0 is ON, execute ITA conversion, the data will be stored in two modes:

- If SM186=OFF, the execution result is: D20=0x3839, D21=0x3637.
- If SM186=ON, the execution result is D20=0x39, D21=0x38, D22=0x37, D23=0x36.

6.5.18 ATI: ASCII code-hexadecimal number conversion instruction

LAD: ┠─┤ ├─[ATI (S1) (D) (S2)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: ATI *(S1) (D) (S2)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: conversion source, ASCII code data 0x30≤***S1***≤0x39 or 0x41≤***S1***≤0x46 (when SM186 is OFF, the high byte and low byte of ***S1*** shall both be within this range)

D: destination operand

S2: number of ASCII codes; 1≤***S2***≤256

■ **Function description**

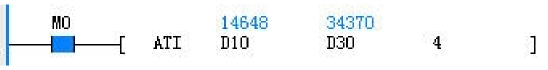
When the power flow is valid, the S2 ASCII code data starting with S1 element will be converted to hexadecimal number, and the result will be stored in the elements starting with D in every 4 bits. When SM186 is OFF, the high/low byte of each D element will store two ASCII code data. When SM186 is ON, the low byte of each D element will store 1 ASCII code data.

■ **Note**

1. When S1 is not within 0x30~0x39 or 0x41~0x46, or S2 is not within 1~256, the system will report operand error and will not execute the instruction, and D will not change.

3. If S1 is a constant, S2 will be regarded as 2 by default when SM186 is OFF and S2≥2, or as when SM186 is ON and S2≥1, and the system will not report operand error.

■ **Example**



LD M0

ATI D10 D30 4

Source data: D10=0x3938, D11=0x3736, D12=0x3534, D13=0x3332

When M0 is ON, the ATI conversion will be executed. According to the data storing mode, the results are as follows:

- If SM186 is OFF, the result is: D30=0x8967.
- If SM186 is ON, the result is: D30=0x8642.

6.5.19 LCNV: Engineering conversion instruction

LAD: ┠─┤ ├─[LCNV (S1) (S2) (D) (S3)]	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: LCNV *(S1)(S2)(D)(S3)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D				V	R	
S2	INT								D				V	R	
D	INT								D				V	R	
S3	Word	Constant							D				V	R	

■ **Operand description**

S1: starting address of source operand under conversion

S2: starting address of conversion table

D: starting address of storing conversion result

S3: number of data under conversion; 1≤***S3***≤64

■ **Function description**

When the analog input module is used to read external analog signals, the instruction can convert the original analog reading to corresponding engineering reading.

When the temperature or analog module is used for measuring, if there is deviation between the temperature or engineering reading measured by PLC and the result measured by standard thermometer or instrument, the instruction can be taken as linear correction to correct the actual measurement.

Fill four values of low point measured value V_{ML}, high point measured value V_{MH}, low point standard value V_{SL} and high point standard value V_{SH} into conversion table; when the linear conversion is executed, the source data will perform operation according to the following formulas and generate corresponding target standard values. S_n is the original input data and D_n is the result after conversion.

$$A = (V_{SL} - V_{SH})/(V_{ML} - V_{MH})*10000$$

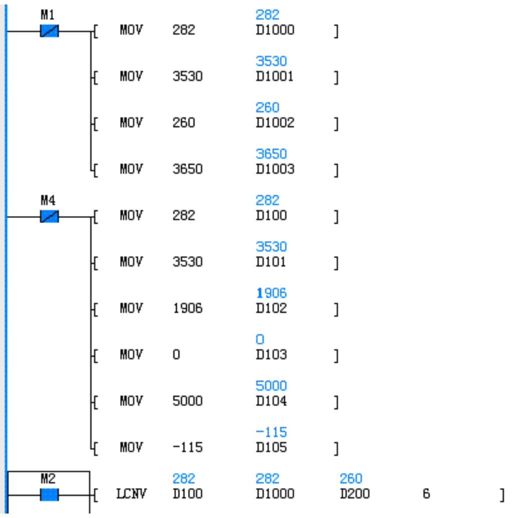
$$B = V_{SL} - (V_{ML} * A/10000)$$

$$D_n = (S_n * A/10000) + B$$

■ **Note**

The four values in the conversion table is meaningful. For example, the low point measured value should be smaller than the high point measured value. The conversion result will be not accurate if exceeding the range of integers. D_n>32767, the result is 32767; D_n<-32768, the result is -32768.

■ **Example**



```
M1                  282
─┤/├─┬─[ MOV  282   D1000   ]
     │              3530
     ├─[ MOV  3530  D1001   ]
     │              260
     ├─[ MOV  260   D1002   ]
     │              3650
     └─[ MOV  3650  D1003   ]
M4                  282
─┤/├─┬─[ MOV  282   D100    ]
     │              3530
     ├─[ MOV  3530  D101    ]
     │              1906
     ├─[ MOV  1906  D102    ]
     │              0
     ├─[ MOV  0     D103    ]
     │              5000
     ├─[ MOV  5000  D104    ]
     │              -115
     └─[ MOV  -115  D105    ]
M2          282     282     260
─┤ ├───[ LCNV D100  D1000   D200   6   ]
```

LDI M1

MOV 282 D1000

MOV 3530 D1001

MOV 260 D1002

MOV 3650 D1003

LDI M4

MOV 282 D100

MOV 3530 D101

MOV 1906 D102

MOV 0 D103

MOV 5000 D104

MOV -115 D105

LD M2

LCNV D100 D100 D1000 D200 6

When M2 is ON, the LCNV conversion will be executed. According to the data storing mode, the results are as follows:

D200=260

D201=3650

D202=1955

D203=-34

D204=5184

D205=-154

6.5.20 RLCNV: Floating point engineering conversion instruction

LAD: ┤├ ┤├─[RLCNV *(S1)* *(S2)* *(D)* *(S3)*]															**Applicable to**	**EC20 EC20H**
															Influenced flag bit	**Zero, carry, borrow**
IL: **RLCNV *(S1)(S2)(D)(S3)***															**Program steps**	**12**

Operand	Type	Applicable elements													Indexed addressing
S1	REAL								D				V	R	
S2	REAL								D				V	R	
D	REAL								D				V	R	
S3	Word	Constant							D				V	R	

■ **Operand description**

S1: starting address of source operand under conversion

S2: starting address of conversion table

D: starting address of storing conversion result

S3: number of data under conversion; 1≤***S3***≤64

- **Function description**

When the analog input module is used to read external analog signals, the instruction can convert the original analog reading to corresponding engineering reading.

When the temperature or analog module is used for measuring, if there is deviation between the temperature or engineering reading measured by PLC and the result measured by standard thermometer or instrument, the instruction can be taken as linear correction to correct the actual measurement.

Fill four values of low point measured value V_{ML}, high point measured value V_{MH}, low point standard value V_{SL} and high point standard value V_{SH} into conversion table; when the linear conversion is executed, the source data will perform operation according to the following formulas and generate corresponding target standard values. S_n is the original input data and D_n is the result after conversion.

$$A = (V_{SL} - V_{SH})/(V_{ML} - V_{MH})*10000$$

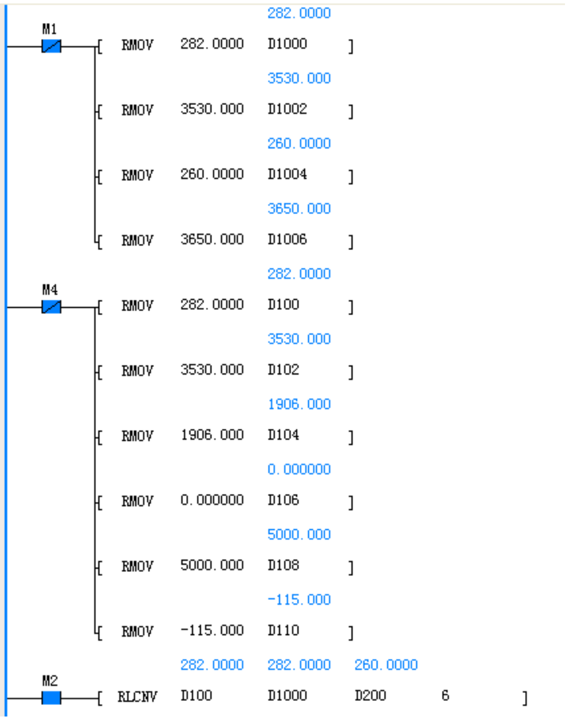
$$B = V_{SL} - (V_{ML} * A/10000)$$

$$D_n = (S_n * A/10000) + B$$

- **Note**

The four values in the conversion table is meaningful. For example, the low point measured value should be smaller than the high point measured value. The conversion result will be not accurate if exceeding the range of integers. D_n>32767, the result is 32767; D_n<-32768, the result is -32768.

- **Example**



```
                                  282.0000
 M1
-|/|--[ RMOV   282.0000  D1000    ]
                                  3530.000
      [ RMOV   3530.000  D1002    ]
                                  260.0000
      [ RMOV   260.0000  D1004    ]
                                  3650.000
      [ RMOV   3650.000  D1006    ]
                                  282.0000
 M4
-|/|--[ RMOV   282.0000  D100     ]
                                  3530.000
      [ RMOV   3530.000  D102     ]
                                  1906.000
      [ RMOV   1906.000  D104     ]
                                  0.000000
      [ RMOV   0.000000  D106     ]
                                  5000.000
      [ RMOV   5000.000  D108     ]
                                  -115.000
      [ RMOV   -115.000  D110     ]
                         282.0000 282.0000 260.0000
 M2
-| |--[ RLCNV  D100      D1000    D200     6        ]
```

LDI M1

RMOV 282 D1000

RMOV 3530 D1002

RMOV 260 D1004

RMOV 3650 D1006

LDI M4

RMOV 282 D100

RMOV 3530 D102

RMOV 1906 D104

RMOV 0 D106

RMOV 5000 D108

RMOV -115 D110

LD M2

RLCNV D100 D1000 D200 6

When M2 is ON, the RLCNV conversion will be executed. According to the data storing mode, the results are as follows:

D200(D201)=260

D202(D203)=3650

D204(D205)=1955

D206(D207)=-34.3288

D208(D209)=5184.267

D210(D211)=-154.357

6.6 Word logic operation

6.6.1 WAND: AND word instruction

LAD: ├─┤ ├─[WAND (S1) (S2) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: WAND *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

Operand description

S1: source operand1

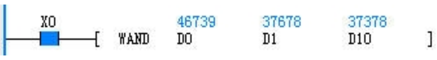
S2: source operand2

D: destination operand

Function description

When the power flow is valid, S1 and S2 will conduct logic AND operation, and the result is assigned to D.

Example



LD X0

WAND

D0 D1 D10

When X0 is ON, D0 2#1011011010010011 (46739) and D1 2#1001001100101110 (37678) will conduct logic AND operation, and the result 2#1001001000000010 (37378) is assigned to D10.

6.6.2 WOR: OR word instruction

LAD: ├─┤ ├─[WOR (S1) (S2) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: WOR *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

Operand description

S1: source operand1

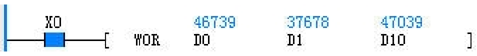
S2: source operand2

D: destination operand

Function description

When the power flow is valid, S1 and S2 will conduct logic OR operation, and the result is assigned to D.

Example



LD X0

WOR D0 D1

D10

When X0 is ON, D0 2#1011011010010011 (46739) and D1 2#1001001100101110 (37678) will conduct logic OR operation, and the result 2#1011011110111111 (47039) is assigned to D10.

6.6.3 WXOR: Exclusive-OR word instruction

LAD: ├─┤ ├─[WXOR (S1) (S2) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: WXOR *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

Operand description

S1: source operand1

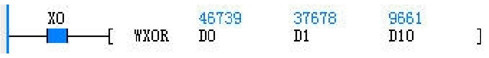
S2: source operand2

D: destination operand

Function description

When the power flow is valid, S1 and S2 will conduct logic exclusive OR operation, and the result is assigned to D.

Example



LD X0

WXOR D0 D1 D10

When X0 is ON, D0 2#1011011010010011 (46739) and D1 2#1001001100101110 (37678) will conduct logic exclusive OR operation, and the result 2#0010010110111101 (9661) is assigned to D10.

6.6.4 WINV: NOT word instruction

LAD: ├─┤ ├──[WINV (S) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: WINV *(S)* *(D)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√

Operand description

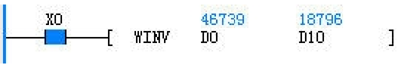
S: source operand

D: destination operand

Function description

When the power flow is valid, conduct logic NOT operation on S, and assign the result to D.

Example



LD X0

WINV D0 D10

When X0 is ON, conduct logic NOT operation on D0 (46739), and assign the result 18796 to D10.

6.6.5 DWAND: AND double word instruction

LAD: ├─┤ ├──[DWAND (S1) (S2) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DWAND *(S1) (S1) (D)*	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√

Operand description

S1: source operand1

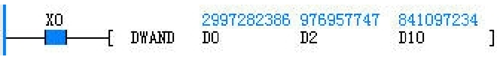
S2: source operand2

D: destination operand

Function description

When the power flow is valid, S1 and S2 will conduct logic AND operation, and the result is assigned to D.

Example



LD X0

DWAND D0 D2 D10

When X0 is ON, (D0, D1) 2#10110010101001101110011001010010 (2997282386) and (D2, D3) 2#00111010001110110011000100110011 (976957747) will conduct the logic AND operation, and the result 2#00110010001000100010000000010010 (841097234) is assigned to (D10, D11).

6.6.6 DWOR: OR double word instruction

LAD: ├─┤ ├──[DWOR (S1) (S2) (D)]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	

<table>
<tr><td colspan="11">IL: DWOR (S1) (S2) (D)</td><td colspan="3">Program steps</td><td colspan="3">10</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S1</td><td>DWORD</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>DWORD</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>DWORD</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td></td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
</table>

- **Operand description**

S1: source operand1

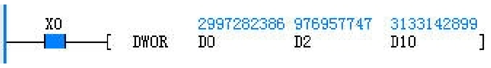
S2: source operand2

D: destination operand

- **Function description**

When the power flow is valid, S1 and S2 will conduct logic OR operation, and the result is assigned to D.

- **Example**



LD X0
DWOR D0 D2 D10

When X0 is ON, (D0, D1) 2#10110010101001101110011001010010 (2997282386) and (D2, D3) 2#00111010001110110011000100110011 (976957747) will conduct logic OR operation, and the result 2#10111010101111111111011101110011 (3133142899) is assigned to (D10, D11).

6.6.7 DWXOR: Exclusive-OR double word instruction

<table>
<tr><td colspan="11" rowspan="2">LAD:
⊣ ⊢[DWXOR (S1) (S2) (D)]</td><td colspan="3">Applicable to</td><td colspan="3">EC10 EC10A EC10V EC20 EC20H</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="3"></td></tr>
<tr><td colspan="11">IL: DWXOR (S1) (S2) (D)</td><td colspan="3">Program steps</td><td colspan="3">10</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S1</td><td>DWORD</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>DWORD</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>DWORD</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td></td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
</table>

- **Operand description**

S1: source operand1

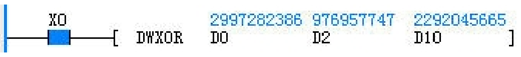
S2: source operand2

D: destination operand

- **Function description**

When the power flow is valid, S1 and S2 will conduct logic exclusive OR operation, and the result is assigned to D.

- **Example**



LD X0
DWXOR D0 D2 D10

When X0 is ON, (D0, D1) 2#10110010101001101110011001010010 (2997282386) and (D2, D3) 2#00111010001110110011000100110011 (976957747) will conduct logic exclusive OR operation, and the result 2#10001000100111011101011101100001 (2292045665) is assigned to (D10, D11).

6.6.8 DWINV: NOT double word instruction

<table>
<tr><td colspan="11" rowspan="2">LAD:
⊣ ⊢[DWINV (S) (D)]</td><td colspan="3">Applicable to</td><td colspan="3">EC10 EC10A EC10V EC20 EC20H</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="3"></td></tr>
<tr><td colspan="11">IL: DWINV (S) (D)</td><td colspan="3">Program steps</td><td colspan="3">7</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S</td><td>DWORD</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>DWORD</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td></td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
</table>

- **Operand description**

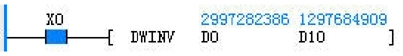
S: source operand

D: destination operand

- **Function description**

When the power flow is valid, logic NOT operation will be conducted on S, and the result is assigned to D.

- **Example**



LD X0
DWINV D0 D10

When X0 is ON, logic NOT operation will be conducted on (D0, D1) 2#10110010101001101110011001010010 (2997282386), and the result 2#01001101010110010001100110101101 (1297684909) is assigned to (D10, D11).

6.7 Shift/Rotate instruction

6.7.1 ROR: 16-bit circular shift right instruction

LAD: |—| |—[ROR (S1) (D) (S2)]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	**Carry flag SM181**
IL: ROR *(S1)* *(D)* *(S2)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

■ **Function description**

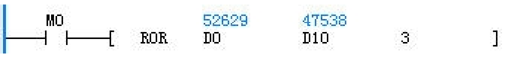
When the power flow is valid, the data of S1 will rotate rightward for S2 bits, and the result is assigned to D. At the same time the highest bit of the S2 bits will be stored into the carry flag (SM181).

■ **Note**

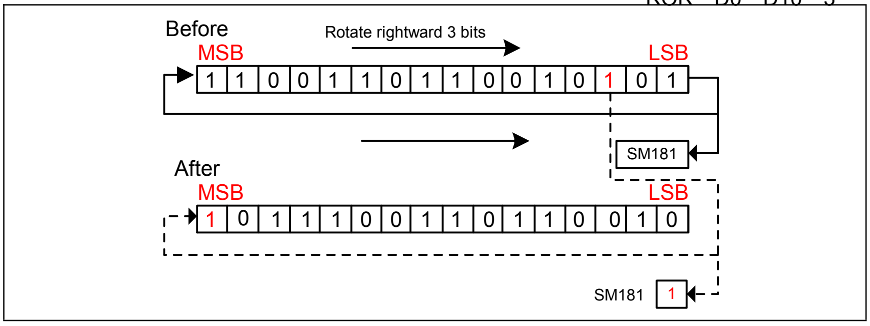
1. S2≥0.

2. When S1 uses Kn addressing, Kn must be equal to 4.

■ **Example**



LD M0
ROR D0 D10 3



When M0 is ON, D0 2#1100110110010101 (52629) rotates rightward for 3 bits, and the result 2#1011100110110010 (47538) is assigned to D10. The highest bit of the 3 bits is stored into the carry flag. SM181 is ON.

6.7.2 ROL: 16-bit circular shift left instruction

LAD: |—| |—[ROL (S1) (D) (S2)]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	**Carry flag SM181**
IL: ROL *(S1)* *(D)* *(S2)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

***S1*: source operand1**

***D*: destination operand**

***S2*: source operand2**

■ **Function description**

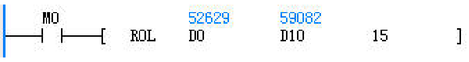
When the power flow is valid, the data of S1 will rotate leftward for S2 bits, and the result is assigned to D. At the same time the lowest bit of the S2 bits will be stored into the carry flag SM181.

■ **Note**

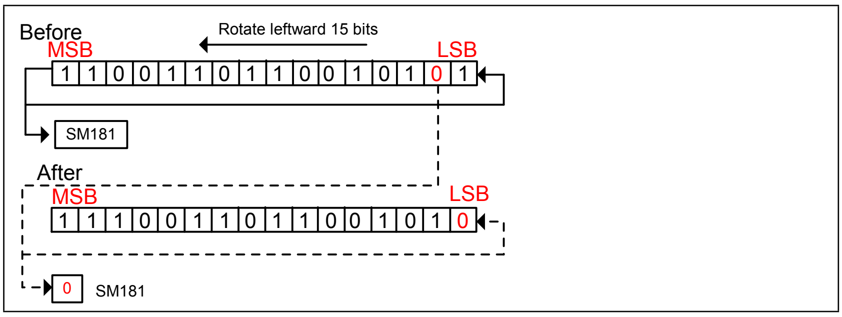
1. S2≥0.

2. When S1 uses Kn addressing, Kn must be equal to 4.

■ **Example**



LD M0
ROL D0 D10 15



When M0 is ON, D0 2#1100110110010101 (52629) rotates leftward for 15 bits, and the result 2#1110011011001010 (59082) is assigned to D10. The final bit will be stored in the carry flag bit. SM181 is OFF.

6.7.3 RCR: 16-bit carry circular shift right instruction

LAD: ┤ ├──[RCR (S1) (D) (S2)]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	**Carry flag SM181**
IL: RCR *(S1) (D) (S2)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

■ **Function description**

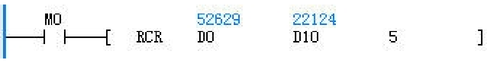
When the power flow is valid, S1 data and the carry flag (SM181) will together rotate rightward for S2 bits, and the result is assigned to D.

■ **Note**

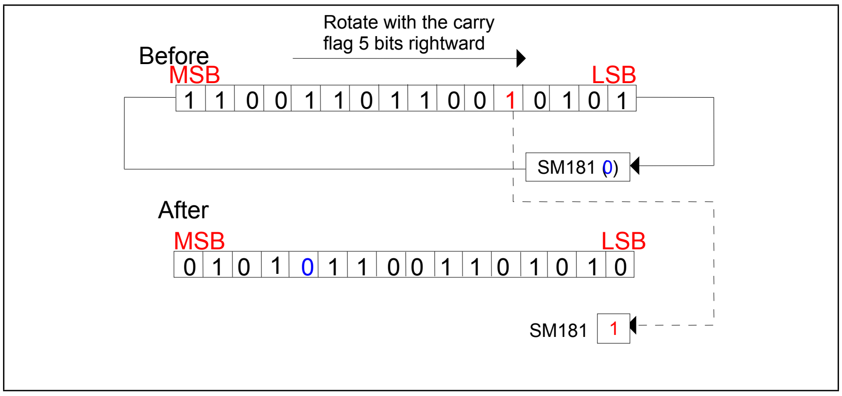
1. S2≥0.

2. When S1 uses Kn addressing, Kn must be equal to 4.

■ **Example**



LD M0
RCR D0 D10 5



When M0 is ON, D0 2#1100110110010101 (52629) and the carry SM181 (OFF) will rotate rightward for 5 bits, and the result 2#0101011001101100 (22124) is assigned to D10. SM181=ON.

6.7.4 RCL: 16-bit carry circular shift left instruction

LAD: ┤ ├──[RCL (S1) (D) (S2)]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	**Carry flag SM181**
IL: RCL *(S1) (D) (S2)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

■ **Function description**

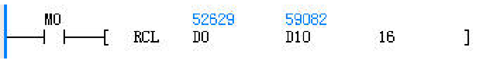
When the power flow is valid, S1 data and the carry (SM181) will together rotate leftward for S2 bits, and the result is assigned to D.

■ **Note**

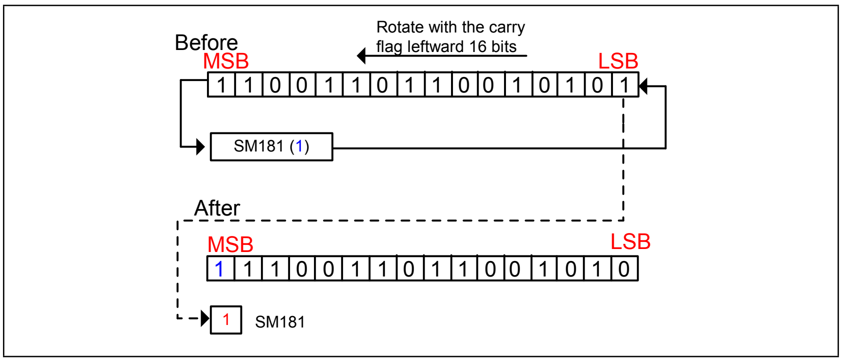
1. S2≥0.

2. When S1 uses Kn addressing, Kn must be equal to 4.

■ **Example**



LD M0
RCL D0 D10 16



When M0 is ON, D0 2#1100110110010101 (52629) and the carry SM181 (ON) will rotate leftward for 16-bits, and the result 2#1110011011001010 (59082) is assigned to D10. SM181=ON.

6.7.5 DROR: 32-bit circular shift right instruction

LAD: DROR *(S1)* *(D)* *(S2)*									**Applicable to**		EC10 EC10A EC10V EC20 EC20H					
									Influenced flag bit		**Carry flag SM181**					
IL: DROR *(S1)* *(D)* *(S2)*									**Program steps**		**9**					
Operand	Type	Applicable elements														Indexed addressing
S1	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

■ **Function description**

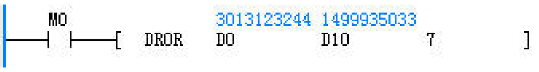
When the power flow is valid, the data of S1 will rotate rightward for S2 bits, and the result is assigned to D. At the same time the highest bit of the S2 bits will be stored into the carry flag bit SM181.

■ **Note**

1. S2≥0.

2. When S1 uses Kn addressing, Kn must be equal to 8.

■ **Example**



LD M0
DROR D0 D10 7

1. When M0 is ON, D0 (D1) 2#10110011100110001001110010101100 (3013123244) will rotate rightward for 7 bits, and the result 2#01011001011001110011000100111001 (1499935033) is assigned to (D10, D11). The final bit is stored into the carry flag bit. SM181=OFF.

2. Please refer to the ROR instruction illustration.

6.7.6 DROL: 32-bit circular shift left instruction

LAD: DROL *(S1)* *(D)* *(S2)*									**Applicable to**		EC10 EC10A EC10V EC20 EC20H					
									Influenced flag bit		**Carry flag SM181**					
IL: DROL *(S1)* *(D)* *(S2)*									**Program steps**		**9**					
Operand	Type	Applicable elements														Indexed addressing
S1	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√

S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

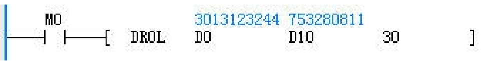
■ **Function description**

When the power flow is valid, the data of S1 will rotate leftward for S2 bits, and the result is assigned to D. At the same time the lowest bit of the S2 bits will be stored into the carry flag bit SM181.

■ **Note**

1. S2≥0.
2. When S1 uses Kn addressing, Kn must be equal to 8.

■ **Example**



```
LD     M0
DROL   D0   D10   30
```

1. When M0 is ON, (D0, D1) 2#10110011100110001001110010101100 (3013123244) will rotate leftward for 30 bits, and the result 2#00101100111001100010011100101011 (753280811) is assigned to (D10, D11). The final bit is stored into the carry flag bit. SM181=ON.
2. Please refer to the ROL instruction illustration.

6.7.7 DRCR: 32-bit carry circular shift right instruction

LAD:	Applicable to	EC10 EC10A EC10V EC20 EC20H
┤├ ─[DRCR *(S1)* *(D)* *(S2)*]	**Influenced flag bit**	**Carry flag SM181**
IL: DRCR *(S1)* *(D)* *(S2)*	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

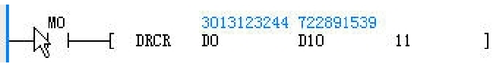
■ **Function description**

When the power flow is valid, S1 data and the carry SM181 will together rotate rightward for S2 bits, and the result is assigned to D.

■ **Note**

1. S2≥0.
2. When S1 uses Kn addressing, Kn must be equal to 8.

■ **Example**



```
LD     M0
DRCR   D0
D10    11
```

1. When M0 is ON, (D0, D1) 2#10110011100110001001110010101100 (3013123244) and the carry SM181 (OFF) will rotate rightward for 11 bits, and the result 2#00101011000101100111001100010011 (722891539) is assigned to (D10, D11). SM181=ON.
2. Please refer to the RCR instruction illustration.

6.7.8 DRCL: 32-bit carry circular shift left instruction

LAD: ├─┤ ├──[DRCL *(S1)* *(D)* *(S2)*]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	**Carry flag SM181**
IL: DRCL *(S1)* *(D)* *(S2)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

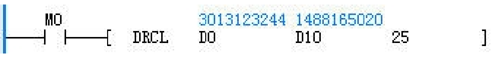
- **Function description**

When the power flow is valid, the S1 data and the carry SM181 will together rotate leftward for S2 bits, and the result is assigned to D.

- **Note**

1. ***S2***≥0.
2. When S1 uses Kn addressing, Kn must be equal to 8.

- **Example**



```
LD      M0
DRCL    D0    D10    25
```

1. When M0 is ON, (D0, D1) 2#10110011100110001001110010101100 (3013123244) and the carry SM181 (OFF) will rotate leftward for 25 bits, and the result 2#001011000101100111001100010011100 (1488165020) is assigned to (D10, D11). SM181 = ON.
2. Please refer to the RCL instruction illustration.

6.7.9 SHR: 16-bit shift right instruction

LAD: ├─┤ ├──[SHR *(S1)* *(D)* *(S2)*]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: SHR *(S1)* *(D)* *(S2)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

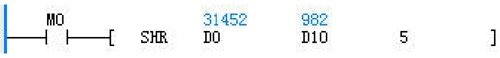
- **Function description**

When the power flow is valid, the data of S1 will shift rightward for S2 bits, and the result is assigned to D.

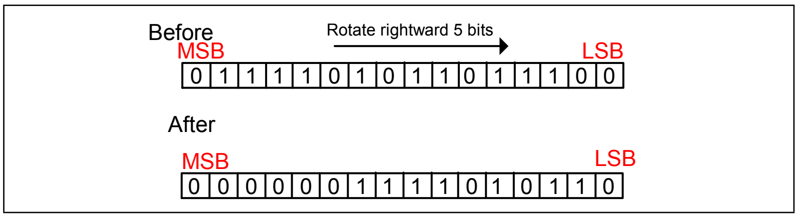
- **Note**

1. S2≥0.
2. When S1 uses Kn addressing, Kn must be equal to 4.

- **Example**



```
LD      M0
SHR     D0    D10    5
```



When M0 is ON, D0 2#0111101011011100 (31452) shifts rightward for 5 bits, and the result 2#0000001111010110 (982) is assigned to D10.

6.7.10 SHL: 16-bit shift left instruction

LAD: SHL (S1) (D) (S2)]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: SHL (S1) (D) (S2)	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

■ **Function description**

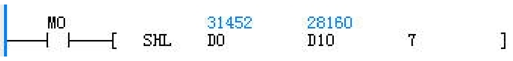
When the power flow is valid, the data of S1 will shift leftward for S2 bits, and the result is assigned to D.

■ **Note**

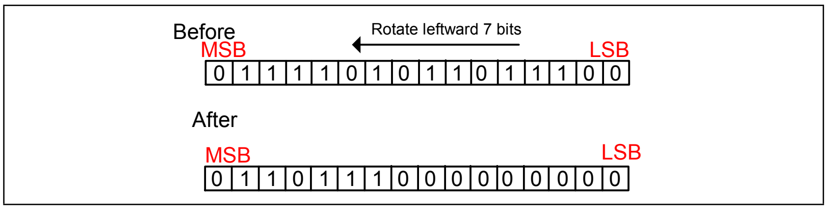
1. S2≥0.

2. When S1 uses Kn addressing, Kn must be equal to 4.

■ **Example**



```
LD   M0
SHL  D0  D10  7
```



Before: 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0

After: 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0

When M0 is ON, D0 2#0111101011011100 (31452) shifts leftward for 7 bits, and the result 2#0110111000000000 (28160) is assigned to D10.

6.7.11 DSHR: 32-bit shift right instruction

LAD: DSHR (S1) (D) (S2)]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: DSHR (S1) (D) (S2)	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

■ **Function description**

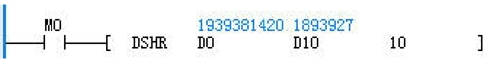
When the power flow is valid, the data of S1 will shift rightward for S2 bits, and the result is assigned to D.

■ **Note**

1. S2≥0.

2. When S1 uses Kn addressing, Kn must be equal to 8.

■ **Example**



```
LD   M0
DSHR D0 D10 10
```

1. When M0 is ON, (D0, D1) 2#01110011100110001001110010101100 (1939381420) shifts rightward for 10 bits, and the result 2#00000000000111001110011000100111 (1893927) is assigned to (D10, D11).

2. Please refer to the SHR instruction illustration.

6.7.12 DSHL: 32-bit shift left instruction

LAD: DSHL (S1) (D) (S2)]	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: DSHL *(S1)* *(D)* *(S2)*	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	DWORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	DWORD			KnY	KnM	KnS	KnLM		D		C		V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

■ **Operand description**

S1: source operand1

D: destination operand

S2: source operand2

■ **Function description**

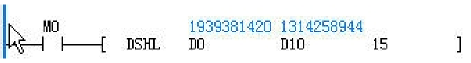
When the power flow is valid, the data of S1 will shift leftward for S2 bits, and the result is assigned to D.

■ **Note**

1. ***S2***≥0.

2. When S1 uses Kn addressing, Kn must be equal to 8.

■ **Example**



LD M0
DSHL D0 D10 15

1. When M0 is ON, (D0, D1) 2#01110011100110001001110010101100 (1939381420) shifts leftward for 15 bits, and the result 2#01001110010101100000000000000000 (1314258944) is assigned to (D10, D11).

2. Please refer to SHL instruction illustration.

6.7.13 SFTR: Shift right byte instruction

LAD: SFTR (S1) (D) (S2) (S3)	**Applicable to**	EC10 EC10A EC10V EC20 EC20H
	Influenced flag bit	
IL: SFTR *(S1)* *(D)* *(S2)* *(S3)*	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	BOOL		X	Y	M	S	LM	SM			C	T				√
D	BOOL			Y	M	S	LM				C	T				√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

S1: source operand1

D: destination operand

S2: source operand2

S3: source operand3

Function description

When the power flow is valid, S2 elements starting with D will move rightward for S3 units, and the S3 elements at the rightmost side will be discarded. At the same time, the contents of S3 elements starting with S1 will be filled into the left end of the string.

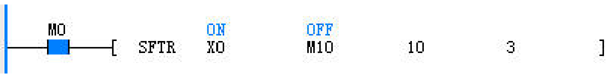
Note

1. The elements with smaller SN are at the right, and the elements with larger SN are at the left.

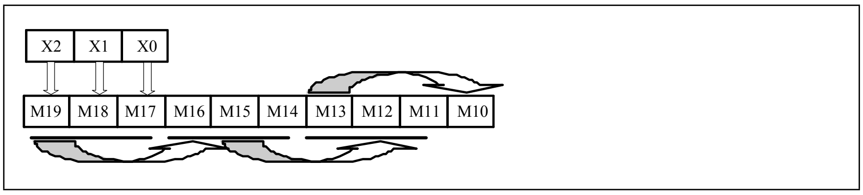
2. S2≥0.

3. S3≥0.

Example



LD M0
SFTR X0 M10 10 3



1. When M0 is ON, the contents of 10 elements starting with M10 will move rightward for 3 bits, and rightmost three elements M10~M12 will be discarded. At

the same time, the contents of the 3 elements starting with X0 will be filled into the left end of the string.

2. Before the execution: X0=1, X1=0, X2=1, M10=0, M11=1, M12=1, M13=0, M14=0, M15=1, M16=0, M17=0, M18=0, M19=1.

3. After the execution: the contents of X0 to X2 remain unchanged, M10=0, M11=0, M12=1, M13=0, M14=0, m15=0, m16=0, m17=1, m18=0, m19=1.

6.7.14 SFTL: Shift left byte instruction

LAD: ┤├ ┤[SFTL *(S1)* *(D)* *(S2)* *(S3)*	**Applicable to**	**EC20 EC10 EC10A EC20H**
	Influenced flag bit	
IL: SFTL *(S1)* *(D)* *(S2)* *(S3)*	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	BOOL		X	Y	M	S	LM	SM			C	T				√
D	BOOL			Y	M	S	LM				C	T				√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

S1: source operand1

D: destination operand

S2: source operand2

S3: source operand3

Note

1. The elements with smaller SN are at the right, and the elements with larger SN are at the left.

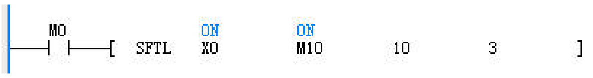
2. S2≥0.

3. S3≥0.

Function description

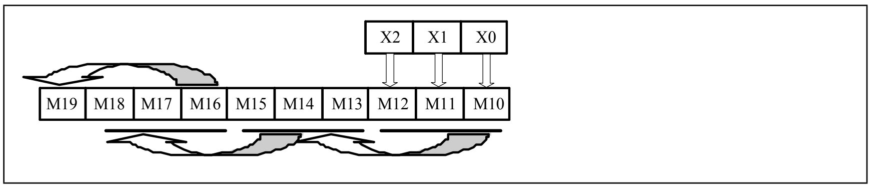
When the power flow is valid, S2 elements starting with D will more leftward for S3 units, and the S3 elements at the leftmost side will be discarded. At the same time, the contents of S3 elements starting with S1 will be filled into the right end of the string.

Example



LD M0

SFTL X0 M10 10 3



1. When M0 is ON, the contents of 10 elements starting with M10 will move leftward for 3 bits, and the leftmost elements M17~M19 will be discarded. At the same time, the contents of the 3 elements starting with X0 will be filled into the right end of the string.

2. Before the execution: X0=1, X1=0, X2=1, M10=0, M11=1, M12=1, M13=0, M14=0, M15=1, M16=0, M17=0, M18=0, M19=1.

3. After the execution: the contents of X0~X2 remain unchanged, M10=1, M11=0, M12=1, M13=0, M14=1, M15=1, M16=0, M17=0, M18=1, M19=0.

6.8 External equipment instruction

6.8.1 FROM: Read word from special module buffer register instruction

LAD: [FROM (S1) (S2) (D) (S3)]	**Applicable to** **Influenced flag bit**	**EC20 EC20H**
IL: FROM (S1) (S2) (D) (S3)	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant														
S2	INT	Constant														
D	INT								D				V		R	√
S3	INT	Constant														

Operand description

S1: SN of the special module to be read, or the target module.

Range: 0~7. If the target module does not exist, the system will report target module address invalid.

S2: the starting address in the BFM of the target module.

Range: 0~32767. If the BFM address is invalid, the system will report “BFM unit of accessed special module exceeds range”.

D: the ***D*** element where the data read from the target module will be stored.

S3: the number of consecutive buffer registers (single word) to be read.

Range: 1~32767. If the target register does not exist, the system will report “BFM unit of accessed special module exceeds range”.

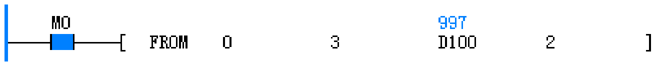
Function description

Read consecutively S3 registers, starting with S2 register, in the BFM of the target module (SN: S1) and put them into the S3 word elements starting with D.

Note

The execution time of the FROM instruction is relatively long, and closely related to ***S3***.

Example



LD M0
FROM 0 3 D100
2

When M0 is ON, read consecutively 2 registers, starting with register 3, in the BFM of the target module number 0, and put them into the word elements D100 and D101.

6.8.2 DFROM: Read double word from special module buffer register instruction

LAD: DFROM (S1) (S2) (D) (S3)	Applicable to	EC20 EC20H
	Influenced flag bit	
IL: DFROM (S1) (S2) (D) (S3)	Program steps	10

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant														
S2	INT	Constant														
D	DINT								D				V		R	√
S3	INT	Constant														

Operand description

S1: SN of the special module to be read, or the target module.

Range: 0~7. If the target module does not exist, the system will report target module address invalid.

S2: the starting address in the BFM of the target module.

Range: 0~32767. If the BFM address is invalid, the system will report "BFM unit of accessed special module exceeds range".

D: the D element where the data read from the target module will be stored.

S3: the number of consecutive buffer registers (double word) to be read.

Range: 1~32767. If the target register does not exist, the system will report "BFM unit of accessed special module exceeds range"

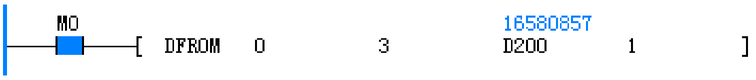
Function description

Read consecutively S3 registers, starting with S2 register, in the BFM of the target module (SN: S1) and put them into the S3 double-word elements starting with D.

Note

The execution time of the DFROM instruction is relatively long, and closely related to S3.

Example

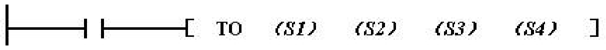


LD M0

DFROM 0 3 D200 1

When M0 is ON, read 1 double word from register 3, in the BFM of the target module number 0, and put it into the double word element (D200, D201).

6.8.3 TO: Write word to special module buffer register instruction

LAD: 	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	
IL: TO *(S1) (S2) (S3) (S4)*	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant														
S2	INT	Constant														
S3	INT								D				V		R	√
S4	INT	Constant														

Operand description

S1: the SN of the special module to be written, or the target module.

Range: 0~7. If the target module does not exist, the system will report "Using FROM/TO instruciton to access module not existing".

S2: the starting register address in the BFM of the target module.

Range: 0~32767. If the BFM address is invalid, the system will report "BFM unit of accessed special module exceeds range".

S3: the data to be written into the target module.

S4: the number of consecutive buffer registers (single word) to be written.

Range: 1~32767. If the target register does not exist, the system will report "BFM unit of accessed special module exceeds range".

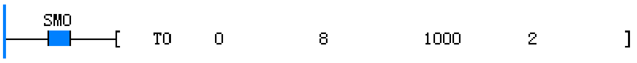
Function description

Write data from consecutive S4 registers starting with S3 to the consecutive S4 buffer registers starting with S2 in the BFM of the target module (SN: S1).

Note

The execution time of the TO instruction is relatively long, and closely related to S4.

Example



```
LD   SM0
TO   0   8   1000
2
```

When PLC runs, write 1000 respectively to buffer registers 8 and 9 in the BFM of target module number 0.

6.8.4 DTO: Write double word to special module buffer register instruction

LAD: 	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	
IL: DTO *(S1) (S2) (S3) (S4)*	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	INT	Constant													
S3	DINT								D				V		R	√
S4	INT	Constant													

Operand description

S1: the SN of the special module to be written, or the target module.

Range: 0~7. If the target module does not exist, the system will report “Using FROM/TO instruciton to access module not existing”.

S2: the starting register address in the BFM of the target module.

Range: 0~32767. If the BFM address is invalid, the system will “BFM unit of accessed special module exceeds range”.

S3: the data to be written into the target module.

S4: the number of consecutive buffer registers (double word) to be written.

Range: 1~32767. If the target register does not exist, the system will report “BFM unit of accessed special module exceeds range”.

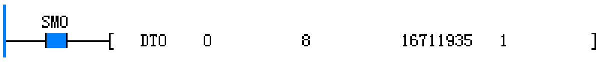
Function description

Write data from consecutive S4 registers starting with S3 to the consecutive S4 buffer registers starting with S2 in the BFM of the target module (SN: S1).

Note

The execution time of the DTO instruction is relatively long, and closely related to S4.

Example



LD SM0

DTO 0 8 16711935

1

When PLC runs, write a double word data 16711935 to buffer registers 8 and 9 (which forms a double-word element) in the BFM of target module number 0.

6.8.5 VRRD: Read analog potentiometer value instruction

LAD: 	**Applicable to**	**EC20 EC10**
	Influenced flag bit	
IL: VRRD *(S) (D)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S	WORD	Constant													
D	WORD								D				V		√

Operand description

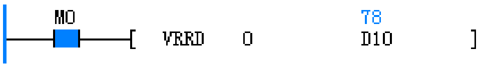
S: the specified potentiometer SN. Range: 0~255. If ***S*** is set outside this range, the system will report operand error.

D: the element where the read analog potentiometer value will be stored. Range: 0~255.

Function description

Read the value of the specified analog potentiometer and store it into the specified element.

Example



LD M0

VRRD 0 D10

When M0 is ON, read the value of analog potentiometer 0 and put the reading into D10.

6.8.6 REFF: Set input filtering constant instruction

<table>
<tr><td colspan="9"><b>LAD:</b>
├─┤ ├─[REFF <i>(S)</i>]</td><td colspan="4"><b>Applicable to</b></td><td colspan="4"><b>EC10 EC10A EC10V EC20 EC20H</b></td></tr>
<tr><td colspan="9"></td><td colspan="4"><b>Influenced flag bit</b></td><td colspan="4"></td></tr>
<tr><td colspan="9"><b>IL: REFF <i>(S)</i></b></td><td colspan="4"><b>Program steps</b></td><td colspan="4"><b>3</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S</td><td>WORD</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

Operand description

S: input filtering constant.

EC20: Range: 0~64ms. Any setting bigger than 64 will be regarded as 64.

EC10: Range: 0, 8, 16, 32, 64. Any setting between any two values will be regarded as the smaller value. For example, a setting smaller than 8 will be regarded as 0, a setting smaller than 16 will be regarded as 8, a setting smaller than 32 will be regarded as 16, a setting smaller than 64 will be regarded as 32, and other settings will be regarded as 64.

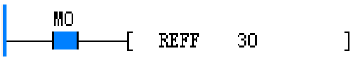
Function description

Set the input filtering constant of X0~X17.

Note

The input filtering constant is valid only for non-high-speed input points.

Example



```
LD    M0
REFF  30
```

When X10 is ON, set the input filtering constant to 30ms.

6.8.7 REF: Instant refresh I/O instruction

<table>
<tr><td colspan="8"><b>LAD:</b>
├─┤ ├─[REF <i>(D)</i> <i>(S)</i></td><td colspan="4"><b>Applicable to</b></td><td colspan="4"><b>EC10 EC10A EC10V EC20 EC20H</b></td></tr>
<tr><td colspan="8"></td><td colspan="4"><b>Influenced flag bit</b></td><td colspan="4"></td></tr>
<tr><td colspan="8"><b>IL: REF <i>(D)</i> <i>(S)</i></b></td><td colspan="4"><b>Program steps</b></td><td colspan="4"><b>5</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>D</td><td>BOOL</td><td></td><td>X</td><td>Y</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>S</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Operand description

D: the starting X or Y element to be refreshed. The specified starting element address should always be a multiple of 8 (in octal system). For example, X0, X10, X20… or Y0, Y10, Y20….

S: the number of inputs and outputs to be refreshed. It should always be a multiple of 8, for example, 8, 16, ..., 256, and so on.

Function description

Generally, the PLC will not refresh its inputs or outputs before the user program ends. However, if you want to refresh the inputs or outputs when the user program is still running, you can use this instruction.

Note

1. The subscript values of inputs (Xn, Yn) are integer multiples of 8.

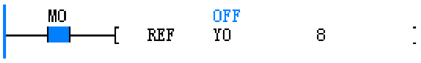
2. The refreshed (terminal) number is the integer multiple of 8.

3. Generally, the REF instruction is used to refresh I/O immediately between the FOR-NEXT instruction and the CJ instruction.

4. You can also use the REF instruction to obtain the latest input and output the operation result without delay during the execution of the interrupts with I/Os.

5. To refresh a relay output, you need to consider the response time.

Example



```
LD    M0
REF   Y0
8
```

When M0 is ON, the status at Y0~Y7 will be output immediately regardless of the scan cycle.

6.8.8 EROMWR: EEPROM write instruction

<table>
<tr><td colspan="10" rowspan="2"><b>LAD:</b>
┤ ├─[EROMWR (S1) (S2)]</td><td colspan="3"><b>Applicable to</b></td><td colspan="4"><b>EC20 EC10 EC20H</b></td></tr>
<tr><td colspan="3"><b>Influenced flag bit</b></td><td colspan="4"></td></tr>
<tr><td colspan="10"><b>IL: EROMWR <i>(S1) (S2)</i></b></td><td colspan="3"><b>Program steps</b></td><td colspan="4"><b>6</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S1</td><td>WORD</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td></td><td></td><td>R</td><td></td></tr>
<tr><td>S2</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

Operand description

S1: starting address of write elements (D6000~D6999)

S2: number of write elements (***S2***<16, ***S1***+***S2***<D7000)

Note

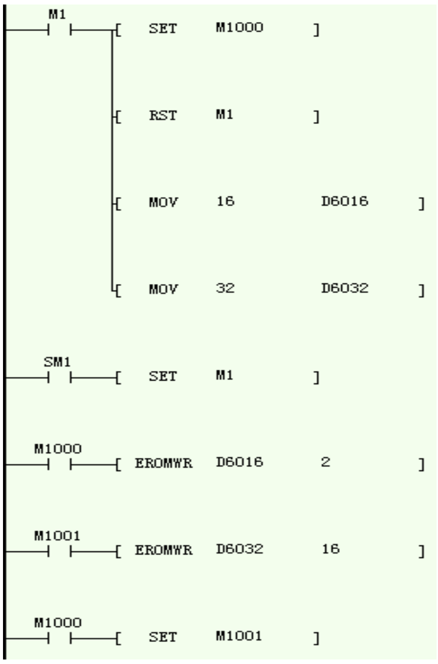
An EROMWR instruction will make the scan cycle 2~5ms longer. It is recommended to set the S1 to 6000 plus an integer multiple of 16, like D6000, D6016 and D6032.

Function description

1. Partial PLC data are battery backed. However, during the calculation, you can save the intermediate data into EEPROM with the EROMWR instruction.

2. This instruction is executed upon the rising edge.

Example



```
 M1
─┤ ├──┬─[ SET     M1000    ]
      ├─[ RST     M1       ]
      ├─[ MOV     16      D6016  ]
      └─[ MOV     32      D6032  ]
 SM1
─┤ ├────[ SET     M1       ]
 M1000
─┤ ├────[ EROMWR  D6016   2      ]
 M1001
─┤ ├────[ EROMWR  D6032   16     ]
 M1000
─┤ ├────[ SET     M1001    ]
```

```
LD      M1
SET     M1000
RST     M1
MOV     16 D6016
MOV     32 D6032
LD      SM1
SET     M1
LD      M1000
EROMWR D6016 2
LD      M1001
EROMWR D6032 16
LD      M1000
SET     M1001
```

In the preceding example, two sets of D elements are stored in the EEPROM:

1. SM1 and M1 makes M1000 generate a rising edge during the second scan cycle and triggers the execution of the first EROMWR instruction.

2. M1001 and SM196 makes the second rising edge, triggering the execution of the second EROMWR instruction.

6.8.9 PR: Print instruction

LAD: PR (S) (D)]	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	
IL: PR (S) (D)	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S	WORD								D		C	T		R	√
D	BOOL			Y											

■ Operand description

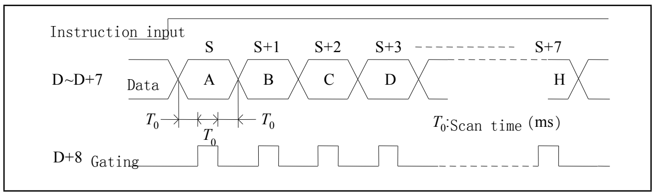
S: starting SN of the elements to be stored

D: starting Y SN of output data

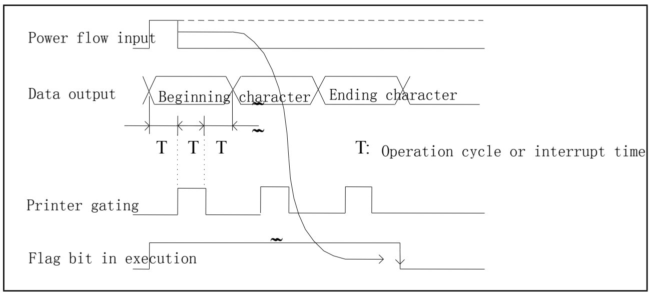
■ Function description

1. The stored data in low 8-bit (1 byte) of S~S+7 is inputted to D~D+7 by time division and the enabling signal is Y0.

2. SM71 is the mark of print instruction in execution. SM71 is set in printing process and printing reset is completed.



3. When the special register SM70 is OFF, serial output 8 bytes; when SM70 is ON, serial output 1~16 bytes. For H00 (NUL code), the previous character is the last one.



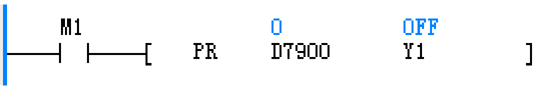
When the power flow is invalid, the flag bit of print instruction in execution will reset.

Note

1. When the power flow is valid, only print once.

2. The flag bit of print instruction in execution can be used to control the breaking of power flow in print instruction.

■ Example



■ Note

1. Only applicable to transistor output modules

2. The instruction is executed with scan cycle at the same time

3. Only one instruction can be executed at the same time. After printing is completed, execute SM71 reset.

.

6.8.10 TKY: Numeric key input instruction

LAD: —| |—[*TKY (S) (D1) (D2)*]	**Applicable to**	**EC20H**
	Influenced flag bit	
IL: TKY *(S) (D1) (D2)*	**Program steps**	**7**

Operand	Type	Applicable elements																		Indexed addressing
S	BOOL	X	Y	M	S	SM	LM													
D1	INT							KnY	KnM	KnLM	D	SD	C	T	V	Z		R		√
D2	BOOL		Y	M	S	SM	LM													

■ **Operand description**

S: starting bit of input numeric keys (occupy 10 bits)

D1: data storing units

D2: element number corresponding to input keys ON/OFF (occupy 11 points)

■ **Function description**

1. S~S+9 are key inputs, the input data will be stored in D1, D2~D2+9 are input information of output keys and D2+10 will detect inputs. When any input is ON, D2+10 will be set.

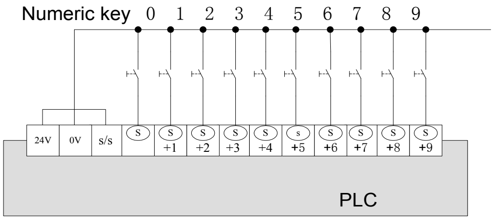
1) Values of D1

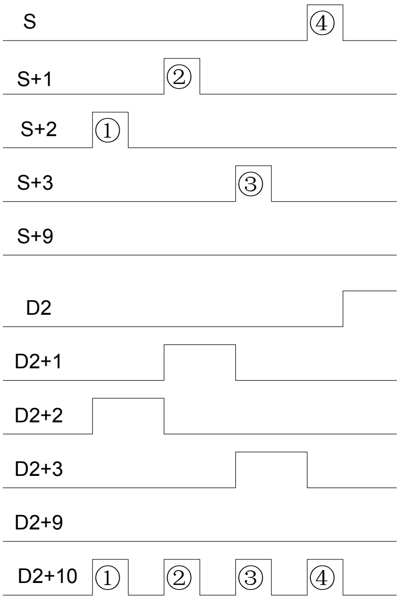
Press the numeric keys ①, ②, ③ and ④ in order, and save 2130 in D1.

2) Key information of D2-D2+10

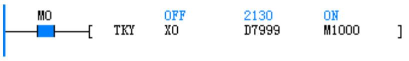
Key information of D2-D2+9, ON/OFF according to the keys pressed

When any key in 0-9 is pressed, D2+10 will output ON.





■ **Example**



M0 —[TKY OFF X0 2130 D7999 ON M1000]

LD M0

TKY X0 D7999 M1000

After pressing X2, X1, X3 and X0 in order, D7999 will be 2130. M1002 will be set after pressing X2 and before pressing other keys. It is the same with other keys. After pressing any key, M1010 will be ON at the time.

■ **Note**

1. When multiple keys are pressed at the same time, only the first pressed key is valid.

2. When the power flow is OFF, D1 remains unchanged while D2~D2+10 becomes OFF.

3. If the input exceeds 9999, overflow from high bit.

4. After an input key is pressed, D2 output bit will be set until the next input key is pressed.

5. Only one of TKY instructions can be used in the program and the indexed addressing can realize use for many times.

6.9 Real-time clock instruction

6.9.1 TRD: Read real-time clock instruction

LAD: ├──┤ ├──[TRD (D)]									**Applicable to**		**EC10 EC10V EC20 EC20H**					
									Influenced flag bit							
IL: TRD *(D)*									**Program steps**		**3**					
Operand	Type	Applicable elements														Indexed addressing
D	WORD								D				V		R	√

Operand description

D: the starting storage element for the system time, which occupies the 7 consecutive elements starting with ***D***

Function description

Read the system time and store the value in the storage elements designated by D.

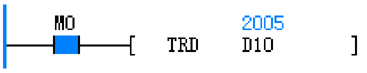
Note

The execution result of the instruction is as follows:

	Element	Item	Clock data		Element	Item
Special data register for real time clock	SD100	Year	2000~2099	————————→	D10	Year
	SD101	Month	1~12	————————→	D11	Month
	SD102	Day	1~31	————————→	D12	Day
	SD103	Hour	0~23	————————→	D13	Hour
	SD104	Minute	0~59	————————→	D14	Minute
	SD105	Second	0~59	————————→	D15	Second
	SD106	Week	0~6	————————→	D16	Week

The TRD instruction will fail upon system clock setting error.

Example



```
LD     M0
TRD    D10
```

When M0 is ON, send the system time to the 7 elements starting with D10.

6.9.2 TWR: Write real-time clock instruction

LAD: ├──┤ ├──[TWR (S)]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	
IL: TWR (S)	**Program steps**	**3**

Operand	Type	Applicable elements														Indexed addressing
S	WORD								D				V		R	√

- **Operand description**

S: the element where the system time is to be written

	Element	Item	Clock data		Element	Item
Data for clock setting	D10	Year	2000~2099	————→	SD100	Year
	D11	Month	1~12	————→	SD101	Month
	D12	Day	1~31	————→	SD102	Day
	D13	Hour	0~23	————→	SD103	Hour
	D14	Minute	0~59	————→	SD104	Minute
	D15	Second	0~59	————→	SD105	Second
	D16	Week	0~6	————→	SD106	Week

Function description

When the system time is different from the real time, you can use the TWR instruction to correct the system time.

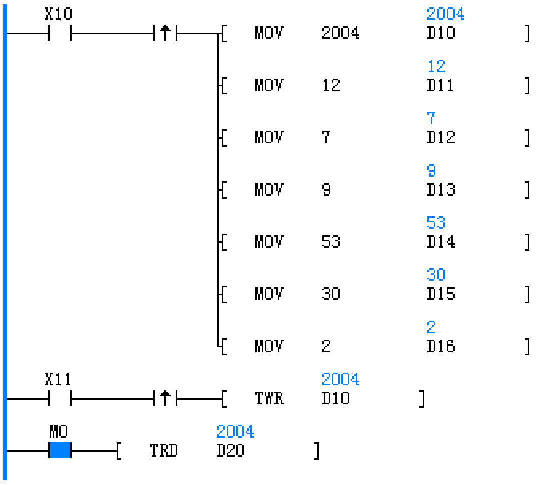
Note

1. The time must use the solar calendar, or the instruction will not be executed.

2. It is recommended to use the edge to trigger the execution of the instruction.

Example

Changing the system time with the TWR instruction is shown in the following figure:



```
LD      X10
EU
MOV     2004 D10
MOV     12 D11
MOV     7 D12
MOV     9 D13
MOV     53 D14
MOV     30 D15
MOV     2 D16
LD      X11
EU
TWR     D10
LD      M0
TRD     D20
```

1. Upon the rising edge of X10, write the time setting into the 7 consecutive units starting with D10.

2. Upon the rising edge of X11, write the values of elements D10 into the system time.

3. When M0 is On, read the system time and store it into D20.

6.9.3 TADD: Add clock instruction

LAD: ├─┤ ├─[TADD (S1) (S2) (D)]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	**Zero flag SM180 Carry flag SM181**
IL: TADD *(S1)* *(S2)* *(D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD								D	SD			V		R	√
S2	WORD								D	SD			V		R	√
D	WORD								D				V		R	√

■ **Operand description**

S1: clock data 1. The 3 storage elements designated by ***S1*** are used to store the time data. If the data is not compliant with the time format, the system will report "Illegal instruction operand value".

S2: clock data 2. The 3 storage elements designated by ***S2*** are used to store another time data. If the data is not compliant with the time format, the system will report "Illegal instruction operand value".

D: time result storage unit. The result of the time adding operation is stored in the 3 storage elements designated by ***D***. The result will affect the carry flag SM181 and the zero flag SM180.

■ **Function description**

Add two time-format data. The operation rules follow the time format.

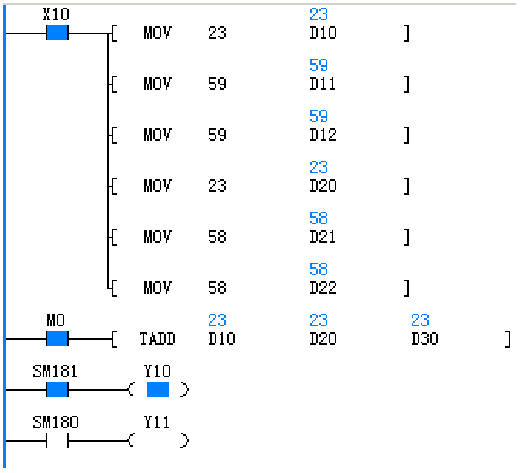
■ **Note**

The time data for the operation must meet the time setting range requirements.

- Hour: 0~23
- Minute: 0~59
- Second: 0~59

■ **Example**

S1			S2			D	
D10	23 (hour)	+	D20	23 (hour)	=	D30	23 (hour)
D11	59 (minute)		D21	58 (minute)		D31	58 (minute)
D12	59 (second)		D22	58 (second)		D32	57 (second)



```
LD    X10
MOV   23   D10
MOV   59   D11
MOV   59   D12
MOV   23   D20
MOV   58   D21
MOV   58   D22
LD    M0
TADD   D0   D20
D30
LD   SM181
OUT   Y10
LD   SM180
OUT   Y11
```

1. When X10 is ON, send the time data to the 3 storage elements starting with D10 and the 3 storage elements starting with D20.

2. When M0 is ON, add the data starting with D10 and the data starting with D20, and store the result in the 3 storage elements starting with D30.

3. The carry flag (SM181) will be set to ON, and the zero flag (SM180) will be set to OFF.

6.9.4 TSUB: Subtract clock instruction

<table>
<tr><td colspan="9" rowspan="2">LAD:
┤├──[TSUB (S1) (S2) (D)]</td><td colspan="4">**Applicable to**</td><td colspan="4">**EC10 EC10V EC20 EC20H**</td></tr>
<tr><td colspan="4">**Influenced flag bit**</td><td colspan="4">**Zero flag SM180,**
borrow flag SM182</td></tr>
<tr><td colspan="9">**IL: TADD *(S1) (S2) (D)***</td><td colspan="4">**Program steps**</td><td colspan="4">**7**</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S1</td><td>WORD</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td>SD</td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>WORD</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td>SD</td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>WORD</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D</td><td></td><td></td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
</table>

Operand description

S1: clock data 1. The 3 storage elements designated by ***S1*** are used to store the time data. If the data is not compliant with the time format, the system will report “Illegal instruction operand value”.

S2: clock data 2. The 3 storage elements designated by ***S2*** are used to store another time data. If the data is not compliant with the time format, the system will report “Illegal instruction operand value”.

D: time result storage unit. The result of the time subtracting operation is stored in the 3 storage elements designated by ***D***. The result will affect the carry flag SM181 and the zero flag SM180.

Function description

Conduct subtract operation on the time format data, with the operation rules following the time format.

Note

The time data for the operation must meet the time setting range requirements.

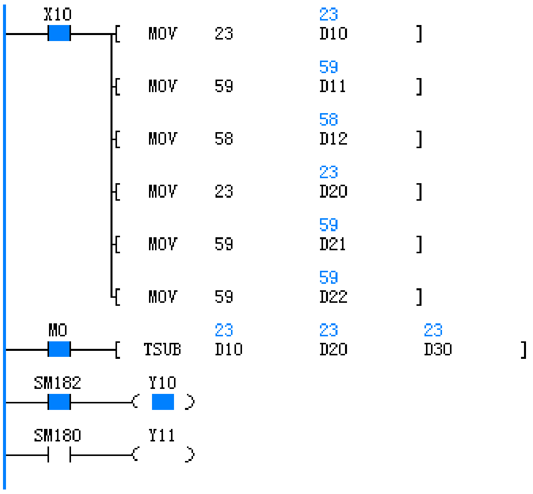
Hour: 0~23

Minute: 0~59

Second: 0~59

Example

S1			S2			D	
D10	23 (hour)		D20	23 (hour)		D30	23 (hour)
D11	59 (minute)	-	D21	59 (minute)	=	D31	59 (minute)
D12	58 (second)		D22	59 (second)		D32	59 (second)



```
LD   X10
MOV  23  D10
MOV  59  D11
MOV  58  D12
MOV  23  D20
MOV  59  D21
MOV  59  D22
LD   M0
TSUB    D10     D20
D30
LD   SM182
OUT  Y10
LD   SM180
OUT  Y11
```

1. When X10 is ON, send the time data to the 3 storage elements starting with D10 and the 3 storage elements starting with D20.

2. When M0 is ON, subtract the data starting with D20 from the data starting with D10, and store the result in the 3 storage elements starting with D30.

3. The carry flag (SM182) will be set to ON, and the zero flag (SM180) will be set to OFF.

6.9.5 HOUR: Timing list instruction

LAD: ├─┤ ├─[HOUR *(S)* *(D1)* *(D2)*]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	
IL: HOUR *(S)* *(D1)* *(D2)*	**Program steps**	**8**

Operand	Type	Applicable elements														Indexed addressing
		Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	INT								D				V		R	√
D2	BOOL			Y	M	S	LM									

■ **Operand description**

S: the hour comparison data. Range: 0~32767.

D1: time storage starting element. ***D1***: hour. ***D1***+1: second.

D2: alarm output address. When ***D1***≥***S***, the alarm point changes to ON, and generates output.

■ **Function description**

Make judgment on the time when the input contact is ON (unit: hour).

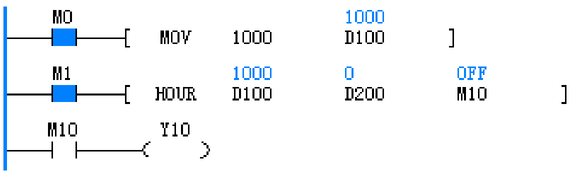
■ **Note**

1. To sustain the current data after power off, set D1 within the element saving range (see **Error! Reference source not found.Error! Reference source not found.**). Otherwise, the current data will be cleared upon PLC power off or when PLC changes from RUN to STOP.

2. The timing still continues even when the alarm output D2 is ON.

3. The hour data in this instruction is a 16-bit integer. It will restart from 0 after 32767.

■ **Example**



```
LD    M0
MOV   1000   D100
LD   M1
HOUR   D100   D200
M10
LD    M10
OUT   Y10
```

1. When M0 is ON, set the comparison data of HOUR instruction.

2. When M1 is ON, accumulate the time for the input contact.

3. M10 will be ON when the accumulated time≥1000.

6.9.6 DCMP: Compare date (=、<、>、<>、>=、<=) instruction

LAD: |——| |——[DCMP= (S1) (S2) (D)] |——| |——[DCMP< (S1) (S2) (D)] |——| |——[DCMP> (S1) (S2) (D)] |——| |——[DCMP<> (S1) (S2) (D)] |——| |——[DCMP>= (S1) (S2) (D)] |——| |——[DCMP<= (S1) (S2) (D)]	**Applicable to**	**EC10 EC10V EC20 EC20H**
	Influenced flag bit	
IL: **DCMP＝ (*S1*) (*S2*) (*D*)** **DCMP＜ (*S1*) (*S2*) (*D*)** **DCMP＞ (*S1*) (*S2*) (*D*)** **DCMP＜＞ (*S1*) (*S2*) (*D*)** **DCMP＞＝ (*S1*) (*S2*) (*D*)** **DCMP＜＝ (*S1*) (*S2*) (*D*)**	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT								D	SD			V		R	√
S2	INT								D	SD			V		R	√
D	BOOL			Y	M	S	LM				C	T				

Operand description

S1: date comparison data 1, which occupies the 3 word elements following ***S1***. The data must comply with the solar calendar format, or the system will report operand error.

S2: date comparison data 2, which occupies the 3 word elements following ***S2***. The data must comply with the solar calendar format, or the system will report operand error.

D: Comparison status output. When the data meet the comparison condition, ***D*** is set ON; otherwise, it is set OFF.

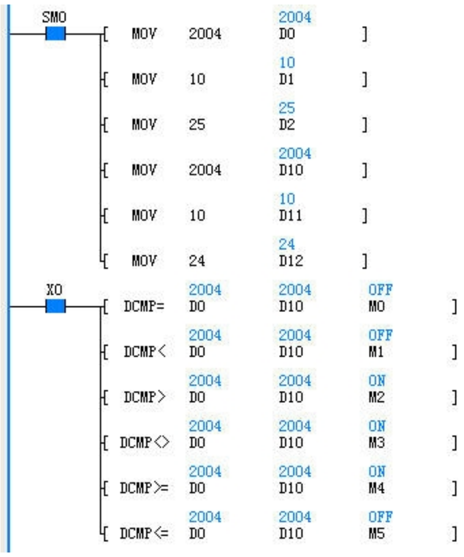
Function description

Conduct BIN comparison on the date data stored in the elements starting with S1 and S2, and assign the comparison result to D.

Note

The date data stored in the elements starting with S1 and S2 must comply with the solar calendar format, or the system will report operand error. For example, “2004, 9, 31” and “2003, 2, 29” are both illegal.

Example


SM0
MOV 2004 D0 2004
MOV 10 D1 10
MOV 25 D2 25
MOV 2004 D10 2004
MOV 10 D11 10
MOV 24 D12 24
X0
DCMP= D0 D10 M0 2004 2004 OFF
DCMP< D0 D10 M1 2004 2004 OFF
DCMP> D0 D10 M2 2004 2004 ON
DCMP<> D0 D10 M3 2004 2004 ON
DCMP>= D0 D10 M4 2004 2004 ON
DCMP<= D0 D10 M5 2004 2004 OFF

LD SM0
MOV 2004 D0
MOV 10 D1
MOV 25 D2
MOV 2004 D10
MOV 10 D11
MOV 24 D12
LD X0
DCMP= D0 D10 M0
DCMP< D0 D10 M1
DCMP> D0 D10 M2
DCMP<> D0 D10 M3
DCMP>= D0 D10 M4
DCMP<= D0 D10 M5

Conduct BIN comparison on the date data stored in the elements starting with D0 and D10, and assign the comparison result to M0.

6.9.7 TCMP: Compare time (=、<、>、<>、>=、<=) instruction

LAD: TCMP= (S1) (S2) (D) TCMP< (S1) (S2) (D) TCMP> (S1) (S2) (D) TCMP<> (S1) (S2) (D) TCMP>= (S1) (S2) (D) TCMP<= (S1) (S2) (D)	**Applicable to** **Influenced flag bit**	**EC10 EC10V EC20 EC20H**
IL: **TCMP＝ (*S1*) (*S2*) (*D*)** **TCMP＜ (*S1*) (*S2*) (*D*)** **TCMP＞ (*S1*) (*S2*) (*D*)** **TCMP＜＞ (*S1*) (*S2*) (*D*)** **TCMP＞＝ (*S1*) (*S2*) (*D*)** **TCMP＜＝ (*S1*) (*S2*) (*D*)**	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT								D	SD			V		R	√
S2	INT								D	SD			V		R	√
D	BOOL			Y	M	S	LM				C	T				

Operand description

S1: time comparison data 1, which occupies the 3 word elements following ***S1***. The data must comply with the 24-hour time format, or the system will report operand error.

S2: time comparison data 2, which occupies the 3 word elements following ***S2***. The data must comply with the 24-hour time format, or the system will report operand error.

D: comparison status output. When the data meet the comparison condition, ***D*** is set ON; otherwise, it is set OFF.

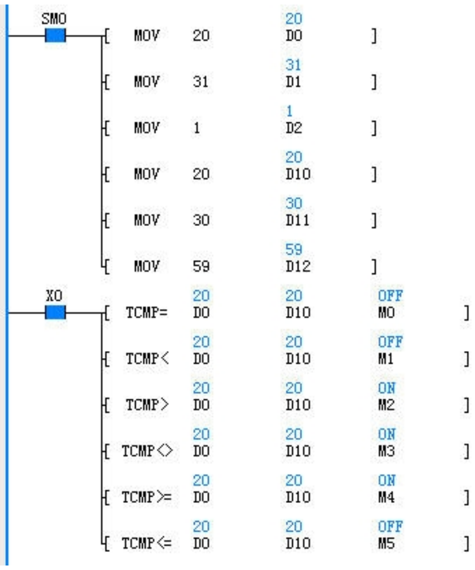
Function description

Conduct BIN comparison on the time data stored in the elements starting with S1 and S2, and assign the comparison result to D.

Note

The time data stored in the elements starting with S1 and S2 must comply with the 24-hour system, or the system will report operand error. For example, “24, 10, 31” and “13, 59, 60” are both illegal.

Example


SM0
MOV 20 20 D0
MOV 31 31 D1
MOV 1 1 D2
MOV 20 20 D10
MOV 30 30 D11
MOV 59 59 D12
X0
TCMP= 20 D0 20 D10 OFF M0
TCMP< 20 D0 20 D10 OFF M1
TCMP> 20 D0 20 D10 ON M2
TCMP<> 20 D0 20 D10 ON M3
TCMP>= 20 D0 20 D10 ON M4
TCMP<= 20 D0 20 D10 OFF M5

```
LD      SM0
MOV   20   D0
MOV   31   D1
MOV   1    D2
MOV   20   D10
MOV   30   D11
MOV   59   D12
LD      X0
TCMP=     D0      D10
M0
TCMP<     D0      D10
M1
TCMP>     D0      D10
M2
TCMP<>   D0      D10
M3
TCMP>=   D0      D10
M4
TCMP<=   D0      D10
M5
```

Conduct BIN comparison on the time data stored in the elements starting with D0 and D10, and assign the comparison result to M0.

6.9.8 HTOS: Time (hour, minute and second) to second instruction

LAD: ┤ ├──[HTOS (S) (D)]	Applicable to	EC20H
	Influenced flag bit	
IL: HTOS (S) (D)	Program steps	5

Operand	Type	Applicable elements													Indexed addressing
S	WORD		KnX	KnY	KnM	KnS	T	C	D	SD				R	√
D	WORD			KnY	KnM	KnS	T	C	D	SD				R	√

Operand description

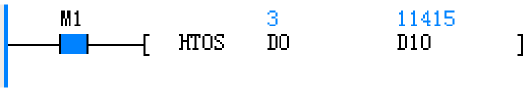
S: starting element of time data before storing conversion

D: starting element of time data after storing conversion

Function description

Convert time data of S-S+2 (hour, minute and second) to second, and store the result in D.

Example



LD M1

HTOS D0 D10

1. When M1 is ON, convert the starting hour, minute and second of D0 to second and store the result in D10. When D0=3, D1=10 and D2=15, D10=11415.

6.9.9 STOH: Second to time (hour, minute and second) instruction

LAD: ┤ ├──[STOH (S) (D)]	Applicable to	EC20H
	Influenced flag bit	
IL: STOH (S) (D)	Program steps	5

Operand	Type	Applicable elements													Indexed addressing
S	WORD		KnX	KnY	KnM	KnS	T	C	D	SD				R	√
D	WORD			KnY	KnM	KnS	T	C	D	SD				R	√

Operand description

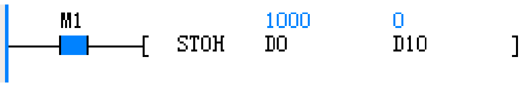
S: starting element of time data before storing conversion

D: starting element of time data after storing conversion

Function description

Convert second data to hour, minute and second, and store the result in D, D+1 and D+2.

Example



LD M1

STOH D0 D10

1. When M1 is ON, convert the second data of D0 to hour, minute and second and store the result in 3 units starting with D10. When D0=1000, D10=0, D11=16 and D12=40.

6.10 High-speed I/O instruction

6.10.1 HCNT: High-speed counter drive instruction

<table>
<tr><td colspan="9" rowspan="2"><b>LAD:</b>
┤ ├─[HCNT (D) (S)]</td><td colspan="3"><b>Applicable to</b></td><td colspan="5"><b>EC10 EC10A EC10V EC20 EC20H</b></td></tr>
<tr><td colspan="3"><b>Influenced flag bit</b></td><td colspan="5"></td></tr>
<tr><td colspan="9"><b>IL: HCNT (D) (S)</b></td><td colspan="3"><b>Program steps</b></td><td colspan="5"><b>7</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>D</td><td>DINT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>C</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>S</td><td>DINT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
</table>

Operand description

D: Counter SN. Range: C236~C255.

S: Comparison constant, a signed 32-bit data.

Range: -2147483648~2147483647.

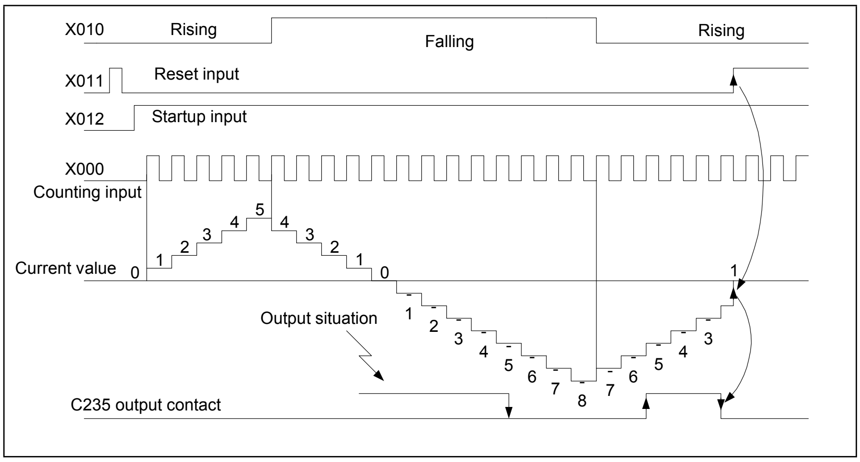
Function description

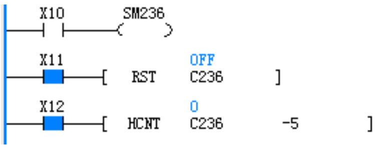
Drive the specified hardware high-speed counter. All high-speed counter must be driven to perform high-speed counting. Meanwhile, the NO contact action of the counter will be judged based on the S value.

Note

The HCNT instruction, SPD instruction, external input interrupt and pulse capture may have contradictory hardware demands. Pay attention to the preconditions of all system high-speed I/Os, and refer to the instruction description in actual practice.

Example





```
X10        SM236
┤ ├────────( )
X11                  OFF
──■────[ RST         C236         ]
X12                  0
──■────[ HCNT        C236    -5   ]
```

```
LD      X10
OUT     SM236
LD      X11
RST     C236
LD      X12
HCNT    C236   -5
```

1. When X12 changes from OFF to ON, the hardware counter C236 will be initialized. X0 is the pulse input point for C236, which counts the pulse input through X0. When X12 is OFF, X0 is a common input point, and C236 cannot count the external pulse of X0.

2. Contact actions: when the current value of the counter C236 increases from -6 to -5, the contact of C236 will be set. When the counter C236 decreases from -5 to -6, the contact of C236 will be reset.

3. When X11 is ON, the RST instruction will be executed, C236 will be cleared, and the C236 contact will be disconnected.

4. When PLC is powered off, the data of the high-speed counter and the contact status is set by the user in the system block through the AutoStation software.

6.10.2 DHSCS: High-speed counting compare set instruction

LAD: ┤├──[DHSCS *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DHSCS *(S1)* *(S2)* *(D)*	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT										C					
D	BOOL			Y	M	S										

Operand description

S1: a 32-bit DINT data, the one with which the high-speed counter will compare.

Range: -2147483648~2147483647.

S2: high-speed counter.

Range: C236~C255.

D: target bit element, including Y, M and S elements. They will be set or output immediately regardless of the scan cycle.

Note

1. The DHSCS instruction must work together with the HCNT instruction, because DHSCS is only applicable to the high-speed counters that is driven by HCNT.

2. The DHSCS instruction will be validated only by pulse input. You cannot validate the instruction by changing the counter value with instructions such as DMOV or MOV.

3. DHSCS (DHSCI, DHSCR, DHSZ, DHSP, DHST) can be used repeatedly. However, at most the first six such instructions can be driven at the same time.

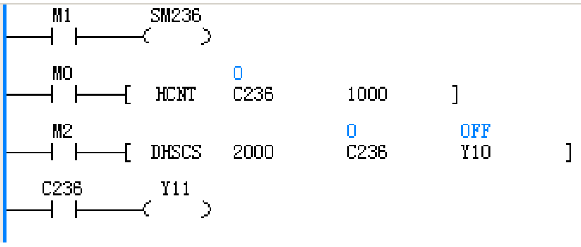
4. The maximum frequency supported by the PLC high-speed counters will be seriously affected by instructions like DHSCS, DHSCI, DHSCR, DHSZ, DHSP and DHST. For details, see **Error! Reference source not found.Error! Reference source not found.**.

Function description

1. A high-speed counter will count in the interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON. When high-speed counter counts to S1 in the DHSCS instruction, the bit element D will be set immediately, or, in the case of a Y element, the Y element will output immediately.

2. This instruction can be used when you want to set (and output, for Y elements) a certain bit element by comparing the counter value with a preset value.

Example



```
LD      M1
OUT     SM236
LD      M0
HCNT    C236   1000
LD      M2
DHSCS   2000   C236
Y10
LD      C236
OUT     Y11
```

1. When M0 is ON, C236 will count in the interrupt mode when X0 changes from OFF to ON (see **Error! Reference source not found.Error! Reference source not found.** for the description of the X0 input frequency). When C236 changes from 999 to 1000, the C236 contact will be set. When C236 changes from 1001 to 1000, the C236 contact will be reset. When the C236 contact drives Y11, the execution of Y11 is determined by the user program scan cycle.

2. When M2 is ON, and the DHSCS instruction meets the requirements stated in the preceding “Note”, Y10 will output immediately if C236 reaches 2000, regardless of the the scan cycle.

3. When M1 is ON, SM236 is driven, and the C236 counter counts down. When M1 is OFF, SM236 is not driven, and the C236 counter counts up.

6.10.3 DHSCI: High-speed counting interrupt trigger instruction

LAD: ┤├──[DHSCI *(S1)* *(S2)* *(S3)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DHSCI *(S1)* *(S2)* *(S3)*	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT										C					
S3	WORD	Constant														

■ **Operand description**

S1: a 32-bit DINT data, the one with which the high-speed counter will compare.

Range: -2147483648~2147483647.

S2: high-speed counter.

Range: C236~C255.

S3: interrupt SN. Range: 20~25.

■ **Function description**

A high-speed counter will count in the interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON. When the counter counts to S1, the S3 interrupt will start. You can write the interrupt according to your actual needs.

■ **Note**

1. The DHSCI instruction must work together with the HCNT instruction, because DHSCI is only applicable to the high-speed counters that is driven by HCNT.

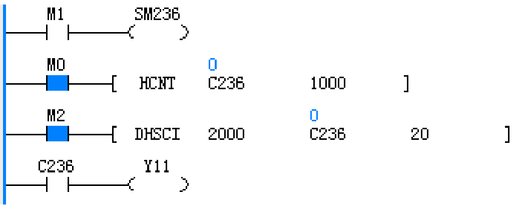
2. The DHSCI instruction will be validated only by pulse input. You cannot validate the instruction by changing the counter value with instructions such as DMOV or MOV.

3. DHSCI (DHSCS, DHSCR, DHSZ, DHSP, DHST) can be used repeatedly. However, at most the first six such instructions can be driven at the same time.

4. The maximum frequency supported by the PLC high-speed counters will be seriously affected by instructions like DHSCS, DHSCI, DHSCR, DHSZ, DHSP and DHST. For details, see **Error! Reference source not found.Error! Reference source not found.**.

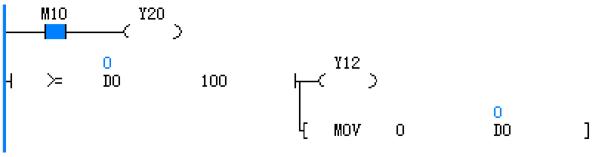
■ **Example**

Main user program:



```
LD      M1
OUT     SM236
LD      M0
DHSCI   2000    C236   20
LD      C236
OUT     Y11
```

Interrupt No.20:



```
LD      M10
OUT     Y20
LD>=    D0   100
OUT     Y12
MOV     0    D0
```

1. When M0 is ON, C236 will count in the interrupt mode when X0 changes from OFF to ON (see **Error! Reference source not found.Error! Reference source not found.** for the description of the X0 input frequency). When C236 changes from 999 to 1000, the C236 contact will be set. When C236 changes from 1001 to 1000, the C236 contact will be reset. When C236 contact drives Y11, the execution of Y11 will be determined by the user program scan cycle.

2. When M2 is ON, and the DHSCI instruction meets the requirements stated in the preceding "Note", interrupt No.20 will be executed immediately when C236 reaches 2000, regardless of the scan cycle.

3. When M1 is ON, SM236 is driven, and the C236 counter counts down. When M1 is OFF, SM236 is not driven, and the C236 counter counts up.

4. With pulse input, interrupt No.20 will be executed when C236 reaches 2000, and Y20 will be driven when M10 is ON. But, the output of Y20 is related to the scan cycle. Meanwhile, Y12 will be driven and D0 will be cleared when D0 is detected to be larger than 100.

6.10.4 DHSPI: High-speed output absolute position compare interrupt trigger instruction

LAD: ├──┤ ├──[DHSCI *(S1)* *(S2)* *(S3)*]	**Applicable to**	**EC20H**
	Influenced flag bit	
IL: DHSPI *(S1)* *(S2)* *(S3)*	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT									SD						
S3	WORD	Constant														

■ Operand description

S1: a 32-bit DINT data, the one with which the high-speed output position element will compare.

Range: -2147483648~2147483647.

S2: high-speed output position element.

Range: SD200,SD320,SD340,SD350,SD360 and SD370.

S3: interrupt SN. Range: 53,54,55,56,57 and 58.

■ Function description

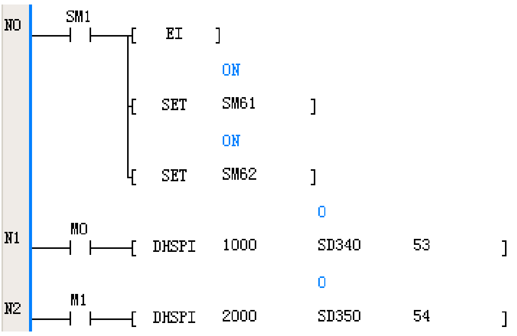
When the high-speed output position element is equal to S1 in DHSPI instruction, enter S3 interrupt subprogram; you can write the program that will be executed immediately in interrupt subprogram.

■ Note

1. Writing to SD element will not trigger position interrupt. After writing, passing the position that needs interrupt again will trigger position interrupt.

■ Example

Main user program:



You can select the interrupt SN to be 53 or other high-speed output postion interrupt sources, and then write the program executed when passing the postion in interrupt subprogram.

6.10.5 DHSCR: High-speed counting compare reset instruction

LAD: ├──┤ ├──[DHSCR *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DHSCR *(S1)* *(S2)* *(D)*	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT										C					
D	BOOL			Y	M	S					C					

■ **Operand description**

S1: a 32-bit DINT data, the one with which the high-speed counter will compare.

Range: -2147483648~2147483647.

S2: high-speed counter.

Range: C236~C255.

D: target bit element. The action on Y, M, S or C will be valid immediately regardless of the scan cycle. If ***D*** is a C element, it must be ***S2***.

■ **Function description**

A high-speed counter will count in the interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON. When the counter counts to S1, the D element will be reset (and output, for Y elements) immediately. You can use this instruction when you want to reset (and output, for Y elements) a certain bit element by comparing the counter value with a preset value.

■ **Note**

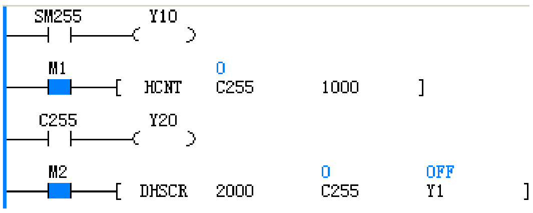
1. The DHSCR instruction must work together with the HCNT instruction, because DHSCR is only applicable to the high-speed counters that is driven by HCNT.

2. The DHSCR instruction will be validated only by pulse input. You cannot validate the instruction by changing the counter value with instructions such as DMOV or MOV.

3. DHSCR (DHSCI, DHSCS, DHSZ, DHSP, DHST) can be used repeatedly. However, at most the first six such instructions can be driven at the same time.

4. The maximum frequency supported by the PLC high-speed counters will be seriously affected by instructions like DHSCS, DHSCI, DHSCR, DHSZ, DHSP and DHST. For details, see **Error! Reference source not found.Error! Reference source not found.**.

■ **Example**



```
LD     SM255
OUT    Y10
LD     M1
HCNT   C255   1000
LD     C255
OUT    Y20
LD     M2
DHSCR   2000   C255
Y1
```

1. When M1 and X7 are both ON, C255 counts the phase difference of X3 and X4 in the interrupt mode. When C255 changes from 999 to 1000, C255 contact will be set, and reset when C255 changes from 1001 to 1000. When C255 contact drives Y20, the execution of Y20 will be determined by the user program scan cycle.

2. When M2 is ON, and the DHSCR instruction meets the requirements stated in the preceding "Note", Y1 will be output immediately when C255 reaches 2000, regardless of the the scan cycle.

3. When the X3 pulse input is ahead of X4, SM255 is ON. When the X4 pulse input is ahead of X3, SM255 is OFF.

4. When X7, the startup signal of C255, is OFF, C255 will not count.

5. When M1 and X7 are all ON, if X5 is ON, C255 will be cleared, and C255 auxiliary contact will be reset.

6.10.6 DHSZ: High-speed counting zone compare instruction

LAD: ├──┤ ├──[DHSZ *(S1)* *(S2)* *(S3)* *(D)*	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DHSZ *(S1)* *(S2)* *(S3)* *(D)*	**Program steps**	**13**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S3	DINT										C					
D	BOOL			Y	M	S										

■ **Operand description**

S1: a 32-bit DINT data, one of the two numbers with which the high-speed counter will compare.

Range: -2147483648~2147483647.

S2: a 32-bit DINT data, one of the two numbers with which the high-speed counter will compare.

Range: -2147483648~2147483647.

S3: high-speed counter.

Range: C236~C255.

D: target bit element. The action on Y, M or S will be valid immediately regardless of the scan cycle.

■ **Function description**

1. A high-speed counter will count in the interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON.

2. When the counter value is smaller than S1, the D element will be set. In addition, the D+1 and D+2 elements will be reset.

3. When the counter value ≥S1 and ≤S2, the D and D+2 elements will be reset, while the D+1 element will be set.

4. When the counter value is bigger than S2, the D and D+1 elements will be reset, while D+2 element will be set.

5. If D is a Y element, it will be output immediately regardless of the scan cycle.

■ **Note**

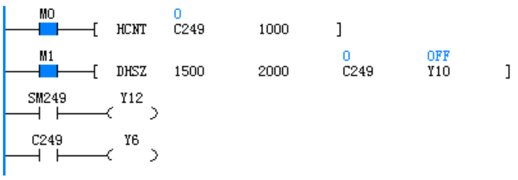
1. The DHSZ instruction must work together with the HCNT instruction, because DHSZ is only applicable to the high-speed counters that is driven by HCNT.

2. The DHSZ instruction will be validated only by pulse input. You cannot validate the instruction by changing the counter value with instructions such as DMOV or MOV.

3. DHSZ (DHSCI, DHSCS, DHSCR, DHSP, DHST) can be used repeatedly. However, at most the first six such instructions can be driven at the same time.

4. The maximum frequency supported by the PLC high-speed counters will be seriously affected by instructions like DHSCS, DHSCI, DHSCR, DHSZ, DHSP and DHST. For details, see **Error! Reference source not found.Error! Reference source not found.**.

■ **Example**



```
LD    M0
HCNT  C249  1000
LD    M1
DHSZ  1500  2000  C249
Y10
LD    SM249
OUT   Y12
LD    C249
OUT   Y6
```

1. When M0 and X6 are both ON, C249 will count up when X0 changes from OFF to ON, or count down when X1 changes from OFF to ON. When C249 changes from 999 to 1000, the C249 contact will be set; when C249 changes from 1001 to 1000, the C249 contact will be reset. When C249 contact drives Y6, the execution of Y6 will be determined by the user program scan cycle.

2. When M1 is ON, the DHSZ instruction meets the requirements stated in the preceding “Note”, the states of elements Y10~Y12 are as follows:

C249<1500: Y10: ON, Y11&Y12: OFF.

1500≤C249≤2000: Y10, Y12: OFF, Y11: ON.

C249>2000: Y10, Y11: OFF, Y12: ON.

The outputs of Y10, Y11 and Y12 are immediate, regardless of the scan cycle.

3. When M0 and X6 are ON at the same time, SM249 will be reset if X0 changes from OFF to ON and the counter counts up, and SM249 will be set if X1 changes from OFF to ON and the counter counts down.

4. When X6 is OFF, C249 stops counting.

5. When M0 and X6 are both ON, if X2 is ON, C249 will be cleared, and C249 auxiliary contact will be reset.

6.10.7 DHST: High-speed counting table compare instruction

LAD: ├──┤ ├──[DHST *(S1)* *(S2)* *(S3)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DHST *(S1) (S2) (S3)*	**Program steps**	**10**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT								D						R	
S2	INT	Constant														
S3	DINT										C					

Operand description

S1: the starting D element for table comparison. The following three D elements are the comparison data, SN of Y element and the output state. These four D elements form a record.

S2: the number of records for comparison. Range: 1~128.

S3: high-speed counter. Range: C236~C255.

Function description

1. A high-speed counter will count in the interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON.

2. When the counter value equates the comparison data of the present record, the corresponding Y element will be output.

3. The Y element specified in the present record will be output immediately, regardless of the scan cycle.

4. You can use the DHST instruction when you want to immediately output, according to certain comparison data, the Y elements specified in a certain table.

Note

1. The DHST instruction must work together with the HCNT instruction, because DHST is only applicable to the high-speed counters that is driven by HCNT.

2. The DHST instruction will be validated only by pulse input. You cannot validate the instruction by changing the counter value with instructions such as DMOV or MOV.

3. DHST (DHSCI, DHSCS, DHSCR, DHSP, DHSZ) can be used repeatedly. However, at most six such instructions can be driven at the same time.

4. In a user program, the DHSP and DHST instructions cannot be valid at the same time. That means a valid DHST (or DHSP) instruction will make the following DHSP (or DHST) instructions invalid.

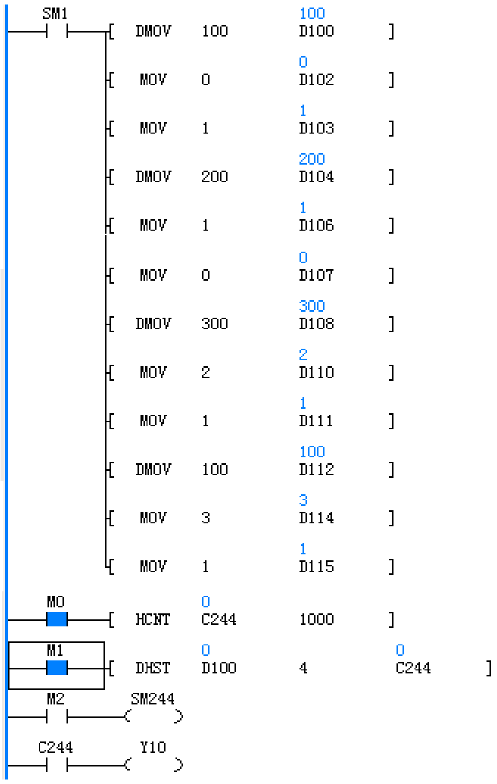
5. The maximum frequency supported by the PLC high-speed counters will be seriously affected by instructions like DHSCS, DHSCI, DHSCR, DHSZ, DHSP and DHST. For details, see **Error! Reference source not found.Error! Reference source not found.**.

Example

The table for comparison is shown below:

Comparison data		Y element	Set/Reset	Operation flow
MSB	LSB			
D100=0	D101=100	D102=0	D103=1	1 ↓
D104=0	D105=200	D106=1	D107=0	2 ↓
D108=0	D109=300	D110=2	D111=1	3 ↓
D112=0	D113=300	D114=3	D115=1	4 ↓ Return to 1

The following is the user program:



```
LD      SM1
DMOV    100   D100
MOV      0    D102
MOV      1    D103
DMOV    200   D104
MOV      1    D106
MOV      0    D107
DMOV    300   D108
MOV      2    D110
MOV      1    D111
DMOV     100 D112
MOV      3    D114
MOV      1    D115
LD       M0
HCNT    C244   1000
LD       M1
DHST     D100   4   C244
LD       M2
OUT       SM244
LD        C244
OUT        Y10
```

1. In the first user-program scan cycle, assign elements D100~D115 with values to generate the table for comparison.

2. When M0 and X6 are both ON, the C244 will count when X0 changes from OFF to ON (for the input frequency, see **Error! Reference source not found.Error! Reference source not found.**). When C244 changes from 999 to 1000, the C244 contact will be set; when C244 changes from 1001 to 1000, the C244 contact will be reset. When the C244 contact drives Y10, the execution of Y10 will be determined by the user program scan cycle.

3. When M1 is ON, and the DHST instruction meets the requirements in the preceding “Note”, the compare will start with the first record. The compare with the second record will not start until the first compare is over and the corresponding Y element has been output. After the compare with the last record is over, the compare with the first record will start again, and SM185 will be set. SD184 is the SN of the present record, and SD182&SD183 are the present data for comparison. The corresponding output will be immediate, regardless of the scan cycle.

4. When M2 is ON, SM244 is ON, and C244 will count down. If M2 is OFF, SM244 is OFF, and C244 will count up.

5. When X6 is OFF, C244 is invalid.

6. When M0 and X6 are both ON, if X2 is ON, C244 will be cleared, and C244 auxiliary contact will be reset.

6.10.8 DHSP: High-speed counting table compare pulse output instruction

LAD: ├─┤ ├─[DHSP *(S1)* *(S2)* *(S3)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: DHSP *(S1)* *(S2)* *(S3)*	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	DINT								D					R	
S2	INT	Constant													
S3	DINT										C				

■ **Operand description**

S1: the starting D element for table comparison. The following three D elements are the comparison data, and the data to output to SD180&SD181. These four D elements form a record.

S2: the number of records to be compared. Range: 1~128.

S3: high-speed counter. Range: C236~C255.

■ **Function description**

1. A high-speed counter will count in the interrupt mode only when it is driven by the HCNT instruction and the counting input changes from OFF to ON.

2. When the counter value equates the comparison data of the present record, the output data of the present record will become the values of SD180&SD181.

3. You can use the DHSP instruction when you want to control the high-speed output or assign values to certain parameters according to a table. For example, you can set the SD180&SD181 (double word) as the output frequency of the PLSY instruction, and the PLSY output frequency will be adjusted by the table compare result.

■ **Note**

1. The DHSP instruction must be used together with the HCNT instruction, because the DHST instruction cannot be executed unless the related high-speed counter is driven by the HCNT instruction.

2. When the DHSP instruction is used together with the PLSY instruction, the values assigned to SD180 and SD181 must meet the frequency output requirement of the PLSY instruction. For details, see the description of the PLSY instruction.

3. To stop the comparison at the last record, set the last output data of the table as 0. Under this situation, other DHST and DHSP instructions will be invalid. But at this time, the DHSP instruction is not regarded as a high-speed instruction when it comes to the number limit of high-speed instructions.

4. The DHSP instruction will be validated only by pulse input. You cannot validate the instruction by changing the counter value with instructions such as DMOV or MOV.

5. DHSP (DHSCI, DHSCS, DHSCR, DHST, DHSZ) can be used repeatedly. However, at most the first six such instructions can be driven at the same time.

6. In a user program, the DHSP and DHST instructions cannot be valid at the same time. That means a valid DHSP (or DHST) instruction will make the following DHST (or DHSP) instructions invalid.

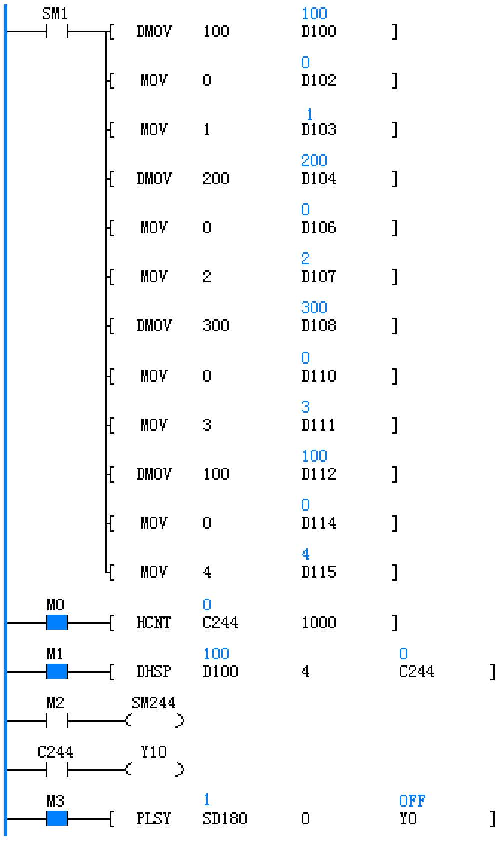
7. The maximum frequency supported by the PLC high-speed counters will be seriously affected by instructions like DHSCS, DHSCI, DHSCR, DHSZ, DHSP and DHST. For details, see **Error! Reference source not found.Error! Reference source not found.**.

■ **Example**

The table for comparison is shown below:

Comparison data		Output data (to SD180&SD181)		Operation flow
MSB	LSB	MSB	LSB	
D100=0	D101=100	D102=0	D103=1	1↓
D104=0	D105=200	D106=0	D107=2	2↓
D108=0	D109=300	D110=0	D111=3	3↓
D112=0	D113=100	D114=0	D115=4	4↓ Return to 1

The following is the user program:



```
LD      SM1
DMOV    100   D100
MOV     0     D102
MOV     1     D103
DMOV    200   D104
MOV     0     D106
MOV     2     D107
DMOV    300   D108
MOV     0     D110
MOV     3     D111
DMOV     100   D112
MOV     0     D114
MOV     4     D115
LD      M0
HCNT    C244   1000
LD      M1
DHSP     D100   4   C244
LD      M2
OUT      SM244
LD       C244
OUT       Y10
LD        M3
PLSY      SD180   0   Y0
```

1. In the first user-program scan cycle, assign elements D100~D115 with values to generate the table for comparison.

2. When M0 and X6 are both ON, C244 will count when X0 changes from OFF to ON (for the input frequency, see **Error! Reference source not found.Error! Reference source not found.**). When C244 changes from 999 to 1000, the C244 contact will be set; when C244 changes from 1001 to 1000, the C244 contact will be reset. When the C244 contact drives Y10, the execution of Y10 will be determined by the user program scan cycle.

3. When M1 is ON, and the DHSP instruction meets the requirements in the preceding “Note”, the compare will start with the first record. The compare with the second record will not start until the first compare is over and the output data has been output to SD180&SD181. After the compare with the last record is over, the compare with the first record will start again, and SM185 will be set. SD184 is the SN of the present record, and SD182&SD183 are the present data for comparison. The output data will be output to SD180&SD181 immediately, regardless of the scan cycle. If you want to stop the at the last record, set the output data of the last record to 0.

4. When M2 is ON, and SM244 is ON, C244 will count down. When M2 is OFF, and SM244 is OFF, C244 will count up.

5. When X6 is OFF, C244 is invalid.

6. When M0 and X6 are both ON, if X2 is ON, C244 will be cleared, and the C244 contact will be reset.

6.10.9 SPD: Pulse detection instruction

LAD: ┣━┫ ┣━[SPD *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: SPD *(S1)* *(S2)* *(D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	BOOL		X													
S2	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD								D				V		R	√

■ **Operand description**

S1: input point. Range: X0~X5.

S2: time unit for input point detection. Unit: ms. Operand S2>0.

D: the storage register for the counted pulse number, which will cause overflow when bigger than 65535.

■ **Function description**

To detect the number of pulses input through X0~X5 in the specified period of time (ms) and store the result in the designated storage register.

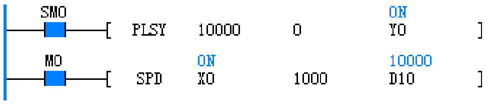
■ **Note**

1. SPD, HCNT, external input interrupt and pulse capture are contradictory in their occupation of hardware. For details, see **Error! Reference source not found.Error! Reference source not found.**.

2. For EC10 and EC20, the SPD instruction supports input points X0~X5. For EC20H, the SPD instruction supports input points X0~X7.

3. Maximum pulse input frequency: 10kHz. Detection may be faulty when frequency is higher than 10kHz.

■ **Example**



```
SM0                      ON
──┤├──[ PLSY  10000  0     Y0     ]
M0     ON                10000
──┤├──[ SPD   X0     1000  D10    ]
```

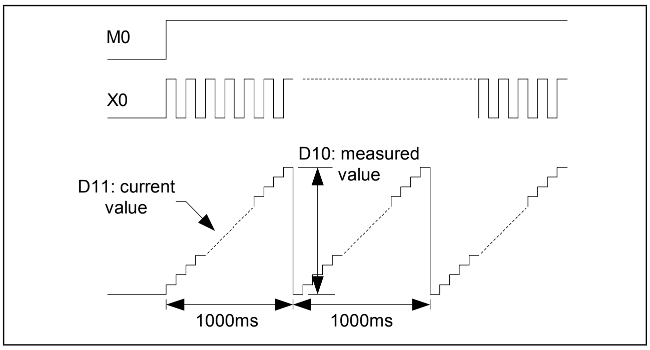
LD SM0

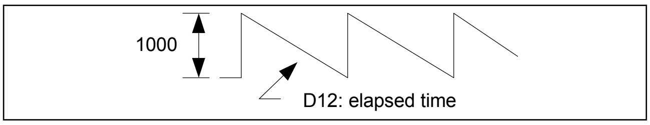
PLSY 10000 0 Y0

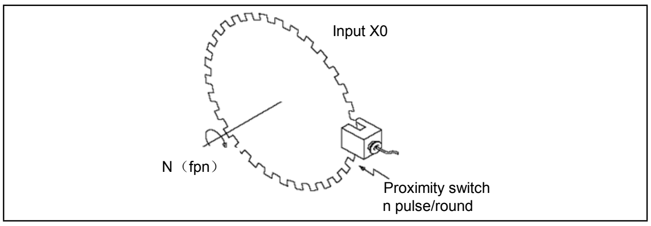
LD M0

SPD X0 1000 D10

The time sequence chart of the example program is shown below:







1. When M0 is ON, count the pulses input through X0 within 1000ms, and store the counting result in D10. D11 is the present counting value within the 1000ms, while D12 is the elapsed time within the 1000ms.

2. D10 is in positive proportion to the rotary speed of the plate in the preceding figure.

3. D10 counts whenever X0 changes from OFF to ON, and the counting value within the last 1000ms will be stored in D10.

6.10.10 PLSY: High-speed pulse output instruction

LAD: ┤├ ┤[PLSY *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC10 EC10A EC10V EC20 EC20H**
	Influenced flag bit	
IL: PLSY *(S1)* *(S2)* *(D)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D	BOOL			Y												

Operand description

S1: specified frequency (Hz).

Range: EC10, EC20: 1~100000(Hz); EC10V: Y0,Y1 can set 1~100000(Hz), Y2,Y3 can set 1~10000(Hz); EC20H: 1~200000(Hz). When ***S1*** is outside this range, the system will report instruction operand error, and no hardware resources will be occupied.

Change ***S1*** during the execution of the instruction will change the output frequency in real time.

S2: output pulse number (PLS).

Range: 0~2147483647. When ***S2*** is outside this range, the system will report instruction operand error, output no pulse, and no hardware resources will be occupied. When ***S2*** is 0, the pulse will output so long as the instruction is valid. If you change ***S2*** during the execution of the instruction, the change will be take effect in the next round.

D: high-speed pulse output point. Range: EC10, EC20: Y0, Y1; EC10V: Y0,Y1,Y2,Y3; EC20H: Y0, Y2, Y4, Y5, Y6, Y7.

Function description

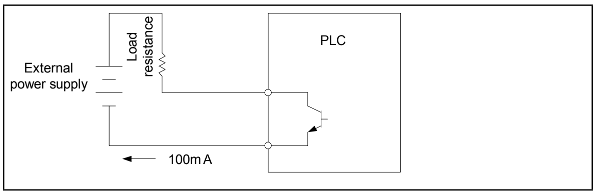
To output specified amount of high-speed pulses at the specified frequency. For that purpose, the load current on the PLC output transistor should be big, but below the rated load current.

Note

1. The PLC must use the transistor output mode.

2. When the PLC outputs high-frequency pulses, the following load current for the PLC output transistor must be used.

3. The output loop (transistor) for PLSY, PWM and PLSR is shown as follows:



4. With large load and current, the transistor off time is relatively longer. The PWM, PLSY and PLSR instructions require that the transistor output terminal be connected to their corresponding loads. When the output waveform does not conform to the instruction operand, increase the load current of the transistor (the transistor load current ≤100mA).

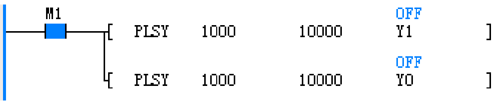
5. During or after the execution of the high-speed instruction, no other instructions can use the same port, unless the high-speed pulse output instruction is invalid.

6. Using multiple PLSY instructions can get independent pulse outputs. You can also use PWM or PLSR instructions to get independent pulse outputs at different output ports.

7. When multiple PWM, PLSY or PLSR instructions work on the same output point, the first valid instruction will control the state of the output point, and others will not affect the output point state.

8. Just like other high-speed instructions (DHSCS, DHSCR, DHSZ, DHSP, DHST and HCNT), the PLSY instruction must meet the system's requests on high-speed I/O.

Example

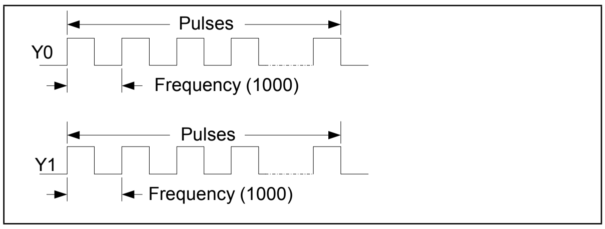


```
 M1                                       OFF
─┤ ├──┬─[ PLSY   1000      10000     Y1        ]
      │                                   OFF
      └─[ PLSY   1000      10000     Y0        ]
```

LD M1

PLSY 1000 10000 Y1

PLSY 1000 10000 Y0



1. When M1 is ON, 10,000 pulses will output through Y0 and Y1 at the frequency of 1000Hz. Then the pulse output will stop until M0 changes from OFF to ON when the next round of output will start. When M0 is OFF, there will be no output.

2. The duty cycle of the pulses is 50%. The output is handled in the interrupt mode, regardless of the scan cycle. For high frequency output, the output duty cycle at Y points is related to the load. The waveform at output points (Y0&Port 0, Y1&Port 1) is related to the load: so long as the current does not exceed the rated load current, the smaller the load is, the closer the output wave form is to the set operand.

3. SM80&SM81 controls the ON/OFF of the output at Y0 and Y1 respectively. When SM80 or SM81 is 1, the output is ON.

4. SM82&SM83 are the output monitors of Y0&Y1 respectively. SM82 or SM83 will be OFF after the output is complete.

5. SD50: the MSB of the output pulse number at Y0 for PLSY and PLSR instructions.

SD51: the LSB of the output pulse number at Y0 for PLSY and PLSR instructions.

SD52: the MSB of the output pulse number at Y1 for PLSY and PLSR instructions.

SD53: the LSB of the output pulse number at Y1 for PLSY and PLSR instructions.

SD54: the MSB of the total output pulse number at Y0 and Y1 for PLSY and PLSR instructions.

SD55: the LSB of the total output pulse number at Y0 and Y1 for PLSY and PLSR instructions.

6. SD50~SD55 can be changed with the instruction DMOV or MOV, or through the ConstrolStar software.

7. Refer to the DHSP instruction if you want to use the input pulse number to control the PLSY output pulse frequency.

Relevant elements:

Address	Name	Action and function	R/W
SM80	Y0 high-speed pulse output control	Y0 high-speed pulse output stop instruction	R/W
SM81	Y1 high-speed pulse output control	Y1 high-speed pulse output stop instruction	R/W
SM82	Y000 pulse output monitor (busy/ready)	Y0 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM83	Y001 pulse output monitor (busy/ready)	Y1 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM86	Y0 interrupt drive pulse ouptut valid	At ON, you can use PLSY instructions in interrupts and subprograms, and continuous and repeated drive with power flow in main programs	R/W
SM87	Y1 interrupt drive pulse ouptut valid	At ON, you can use PLSY instructions in interrupts and subprograms, and continuous and repeated drive with power flow in main programs	R/W
SM262	Y002 pulse output stop instruction	After setting, Y002 pulse will be disabled	R/W
SM264	Y004 pulse output stop instruction	After setting, Y004 pulse will be disabled	R/W
SM265	Y005 pulse output stop instruction	After setting, Y005 pulse will be disabled	R/W
SM266	Y006 pulse output stop instruction	After setting, Y006 pulse will be disabled	R/W
SM267	Y007 pulse output stop instruction	After setting, Y007 pulse will be disabled	R/W
SM272	Y002 pulse output monitor (busy/ready)	Y002 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM274	Y004 pulse output monitor (busy/ready)	Y004 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM275	Y005 pulse output monitor (busy/ready)	Y005 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM276	Y006 pulse output monitor (busy/ready)	Y006 high-speed pulse output monitor (ON: busy, OFF: ready))	R
SM277	Y007 pulse output monitor (busy/ready)	Y007 high-speed pulse output monitor (ON: busy, OFF: ready)	R

Address	Action and function	R/W
SD50	PLSY accumulated output Y0 total pulse number (MSB) (EC10V, EC20, EC20H)	R/W
SD51	PLSY accumulated output Y0 total pulse number (LSB) (EC10V, EC20, EC20H)	R/W
SD52	PLSY accumulated output Y1 total pulse number (MSB) (EC10V, EC20)	R/W

Address	Action and function	R/W
SD53	PLSY accumulated output Y1 total pulse number (LSB) (EC10V, EC20)	R/W
SD54	PLSY accumulated output Y1, Y0 total pulse number (MSB) (EC20)	R/W
SD55	PLSY accumulated output Y1, Y0 total pulse number (LSB) (EC20)	R/W
SD160	PLSY accumulated output Y2 total pulse number (MSB) (EC10V, EC20H)	R/W
SD161	PLSY accumulated output Y2 total pulse number (LSB) (EC10V, EC20H)	R/W
SD162	PLSY accumulated output Y3 total pulse number (MSB) (EC10V)	R/W
SD163	PLSY accumulated output Y3 total pulse number (LSB) (EC10V)	R/W
SD164	PLSY accumulated output Y4 total pulse number (MSB) (EC20H)	R/W
SD165	PLSY accumulated output Y4 total pulse number (LSB) (EC20H)	R/W
SD166	PLSY accumulated output Y5 total pulse number (MSB) (EC20H)	R/W
SD167	PLSY accumulated output Y5 total pulse number (LSB) (EC20H)	R/W
SD168	PLSY accumulated output Y6 total pulse number (MSB) (EC20H)	R/W
SD169	PLSY accumulated output Y6 total pulse number (LSB) (EC20H)	R/W
SD170	PLSY accumulated output Y7 total pulse number (MSB) (EC20H)	R/W
SD171	PLSY accumulated output Y7 total pulse number (LSB) (EC20H)	R/W
SD80	Current position of Y0 output locating instruction (MSB) (EC10)	R/W
SD81	Current position of Y0 output locating instruction (LSB) (EC10)	R/W
SD82	Current position of Y1 output locating instruction (MSB) (EC10)	R/W
SD83	Current position of Y1 output locating instruction (LSB) (EC10)	R/W
SD200	Current position of Y0 output locating instruction (MSB) (EC10V, EC20, EC20H)	R/W
SD201	Current position of Y0 output locating instruction (LSB) (EC10V, EC20, EC20H)	R/W
SD210	Current position of Y1 output locating instruction (MSB) (EC10V, EC20)	R/W
SD211	Current position of Y1 output locating instruction (LSB) (EC10V, EC20)	R/W
SD320	Current position of Y2 output locating instruction (MSB) (EC10V, EC20H)	R/W
SD321	Current position of Y2 output locating instruction (LSB) (EC10V, EC20H)	R/W
SD330	Current position of Y3 output locating instruction (MSB) (EC10V)	R/W
SD331	Current position of Y3output locating instruction (LSB) (EC10V)	R/W
SD340	Current position of Y4 output locating instruction (MSB) (EC20H)	R/W
SD341	Current position of Y4 output locating instruction (LSB) (EC20H)	R/W
SD350	Current position of Y5 output locating instruction (MSB) (EC20H)	R/W
SD351	Current position of Y5 output locating instruction (LSB) (EC20H)	R/W
SD360	Current position of Y6 output locating instruction (MSB) (EC20H)	R/W
SD361	Current position of Y6 output locating instruction (LSB) (EC20H)	R/W
SD370	Current position of Y7 output locating instruction (MSB) (EC20H)	R/W
SD371	Current position of Y7 output locating instruction (LSB) (EC20H)	R/W

6.10.11 PLSR: Count pulse with acceleration/deceleration output instruction

<table>
<tr><td colspan="3" rowspan="2"><b>LAD:</b>
PLSR (S1) (S2) (S3) (D)</td><td><b>Applicable to</b></td><td><b>EC10 EC10A EC10V EC20 EC20H</b></td></tr>
<tr><td><b>Influenced flag bit</b></td><td></td></tr>
<tr><td colspan="3"><b>IL: PLSR <i>(S1) (S2) (S3) (D)</i></b></td><td><b>Program steps</b></td><td><b>10</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="2">Applicable elements</td><td>Indexed addressing</td></tr>
</table>

S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S3	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	BOOL			Y												

■ Operand description

S1: maximum frequency. Range: 10~20,000(Hz), EC10V: Y2,Y3 range:10~10000(Hz). When ***S1*** is specified indirectly, and if the specified value is outside this setting range, it will be regarded as 10 or 20,000, depending on which limit it breaks. In that case, the system will report operand illegal, and the high-speed pulse output will be based on the default 10Hz or 20,000Hz.

S2: total output pulse number (PLS). Range: 110~2147483647. When ***S2*** is outside this range, the system will report instruction operand error, output no pulse, and no hardware resources will be occupied.

S3: acceleration or deceleration time (ms). If ***S1***×***S3***<100,000, ***S3*** will be regarded as 100000/***S1***. Meanwhile the system will report instruction operand error, and the acceleration or deceleration time will be uncertain.

If ***S1***×***S3***>***S2***×909, ***S3*** will be regarded as S2×909/***S1***. Meanwhile the system will report instruction operand error, and the acceleration or deceleration time will be uncertain.

🕮 **Note**

For EC10, the acceleration/deceleration time must not be smaller than 50ms.

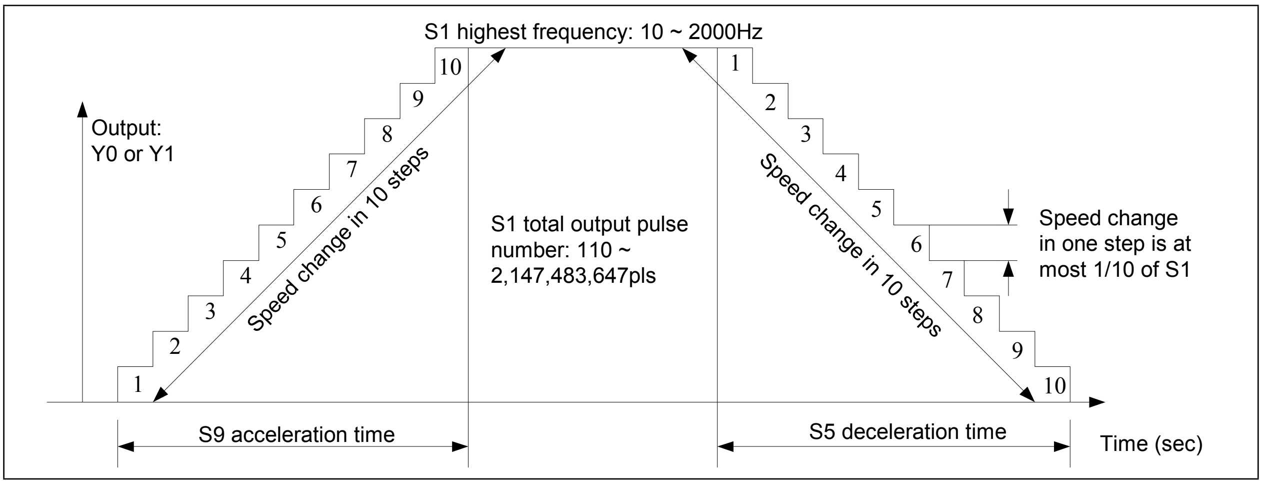
The speed change is evenly divided into 10 steps during the acceration or deceleration, each step being ***S1***/10.

D: high-speed pulse output point. Range: EC10, EC20: Y0, Y1; EC10V: Y0, Y1, Y2, Y3; EC20H: Y0, Y2, Y4, Y5, Y6, Y7.

■ Function description

The PLSR instruction is a high-speed pulse output instruction with acceleration/deceleration function. It is used for locating. Targeting at the specified maximum frequency, the pulse output will accelerate evenly. After the output pulse number reaches the preset value, the pulse output will decelerate evenly.

The operation process is shown in the following figure:


S1 highest frequency: 10 ~ 2000Hz
Output:
Y0 or Y1
1
2
3
4
5
6
7
8
9
10
Speed change in 10 steps
S1 total output pulse number: 110 ~ 2,147,483,647pls
Speed change in 10 steps
Speed change in one step is at most 1/10 of S1
S9 acceleration time
S5 deceleration time
Time (sec)

■ Note

1. The output frequency of this instruction is 10~20,000Hz. When the acceleration/deceleration rate exceeds that range, it will be automatically adjusted according to that range, regardless of the scan cycle.

2. Use the transistor output. During the high-speed pulse output, the output current must comply with the related regulations. The waveform at output points (Y0&Port 0, Y1&Port 1) is related to the load: so long as the current does not exceed the rated load current, the smaller the load is, the closer the output waveform is to the set operand.

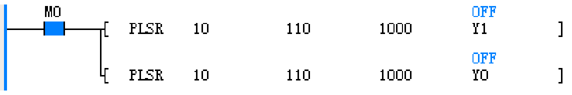
3. During the execution of the high-speed instruction, so long as the power flow is not OFF, no other instructions can use the same port, unless the high-speed pulse output instruction is invalid.

4. Using two PLSR instructions can output two independent pulses at Y0 and Y1. You can also use PLSR and the PWM (or PLSY) instruction to get independent pulse outputs at different output ports (Y0, Y1).

5. When multiple PWM, PLSY or PLSR instructions work on the same output point, the first valid instruction will control the state of the output point, and others will not affect the output point state.

6. Just like other high-speed instructions (DHSCS, DHSCR, DHSZ, DHSP, DHST and HCNT), the PLSR instruction must meet the system's requests on high-speed I/O.

■ **Example**



LD M0

PLSR 10 110 1000 Y1

PLSR 10 110 1000 Y0

1. When M0 is ON, Y0 and Y1 output 110 pulses respectively at set frequencies. When M0 changes from OFF to ON, pulses will be output again. When M0 is OFF, the output will stop.

2. The operand change during the execution of the instruction will not be valid until the next time this instruction is executed.

3. SM80&SM81 controls the ON/OFF of the output at Y0 and Y1 respectively. When SM80 or SM81 is 1, the output will stop.

4. SM82&SM83 are the output monitors of Y0&Y1 respectively. SM82 & SM83 will be ON when the output is going on, or OFF when the output is over.

5. SD50: the MSB of the output pulse number at Y0 for PLSY and PLSR instructions.

SD51: the LSB of the output pulse number at Y0 for PLSY and PLSR instructions.

SD52: the MSB of the output pulse number at Y1 for PLSY and PLSR instructions.

SD53: the LSB of the output pulse number at Y1 for PLSY and PLSR instructions.

SD54: the MSB of the total output pulse number at Y0 and Y1 for PLSY and PLSR instructions.

SD55: the LSB of the total output pulse number at Y0 and Y1 for PLSY and PLSR instructions.

6. SD50~SD55 can be changed with the instruction DMOV or MOV, or through the AutoStation software.

Relevant elements:

Address	Name	Action and function	R/W
SM80	Y0 high-speed pulse output control	Y0 high-speed pulse output stop instruction	R/W
SM81	Y1 high-speed pulse output control	Y1 high-speed pulse output stop instruction	R/W
SM82	Y0 high-speed pulse output monitor	Y0 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM83	Y1 high-speed pulse output monitor	Y1 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM262	Y002 pulse output stop instruction	After setting, Y002 pulse will be disabled	R/W
SM264	Y004 pulse output stop instruction	After setting, Y004 pulse will be disabled	R/W
SM265	Y005 pulse output stop instruction	After setting, Y005 pulse will be disabled	R/W
SM266	Y006 pulse output stop instruction	After setting, Y006 pulse will be disabled	R/W
SM267	Y007 pulse output stop instruction	After setting, Y007 pulse will be disabled	R/W
SM272	Y002 pulse output monitor (busy/ready)	Y002 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM274	Y004 pulse output monitor (busy/ready)	Y004 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM275	Y005 pulse output monitor (busy/ready)	Y005 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM276	Y006 pulse output monitor (busy/ready)	Y006 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM277	Y007 pulse output monitor (busy/ready)	Y007 high-speed pulse output monitor (ON: busy, OFF: ready)	R

Address	Action and function	R/W
SD50	PLSY accumulated output Y0 total pulse number (MSB) (EC10V, EC20, EC20H)	R/W
SD51	PLSY accumulated output Y0 total pulse number (LSB) (EC10V, EC20, EC20H)	R/W
SD52	PLSY accumulated output Y1 total pulse number (MSB) (EC10V, EC20)	R/W
SD53	PLSY accumulated output Y1 total pulse number (LSB) (EC10V, EC20)	R/W
SD54	PLSY accumulated output Y1, Y0 total pulse number (MSB) (EC20)	R/W
SD55	PLSY accumulated output Y1, Y0 total pulse number (LSB) (EC20)	R/W
SD160	PLSY accumulated output Y2 total pulse number (MSB) (EC10V, EC20H)	R/W
SD161	PLSY accumulated output Y2 total pulse number (LSB) (EC10V, EC20H)	R/W
SD162	PLSY accumulated output Y3 total pulse number (MSB) (EC10V)	R/W
SD163	PLSY accumulated output Y3 total pulse number (LSB) (EC10V)	R/W
SD164	PLSY accumulated output Y4 total pulse number (MSB) (EC20H)	R/W
SD165	PLSY accumulated output Y4 total pulse number (LSB) (EC20H)	R/W
SD166	PLSY accumulated output Y5 total pulse number (MSB) (EC20H)	R/W
SD167	PLSY accumulated output Y5 total pulse number (LSB) (EC20H)	R/W
SD168	PLSY accumulated output Y6 total pulse number (MSB) (EC20H)	R/W
SD169	PLSY accumulated output Y6 total pulse number (LSB) (EC20H)	R/W
SD170	PLSY accumulated output Y7 total pulse number (MSB) (EC20H)	R/W
SD171	PLSY accumulated output Y7 total pulse number (LSB) (EC20H)	R/W
SD80	Current position of Y0 output locating instruction (MSB) (EC10)	R/W
SD81	Current position of Y0 output locating instruction (LSB) (EC10)	R/W
SD82	Current position of Y1 output locating instruction (MSB) (EC10)	R/W
SD83	Current position of Y1 output locating instruction (LSB) (EC10)	R/W

Address	Action and function	R/W
SD200	Current position of Y0 output locating instruction (MSB) (EC10V ,EC20,EC20H)	R/W
SD201	Current position of Y0 output locating instruction (LSB) (EC10V ,EC20,EC20H)	R/W
SD210	Current position of Y1 output locating instruction (MSB) (EC10V ,EC20)	R/W
SD211	Current position of Y1 output locating instruction (LSB) (EC10V ,EC20)	R/W
SD320	Current position of Y2 output locating instruction (MSB) (EC10V ,EC20H)	R/W
SD321	Current position of Y2 output locating instruction (LSB) (EC10V ,EC20H)	R/W
SD330	Current position of Y3 output locating instruction (MSB) (EC10V)	R/W
SD331	Current position of Y3 output locating instruction (LSB) (EC10V)	R/W
SD340	Current position of Y4 output locating instruction (MSB) (EC20H)	R/W
SD341	Current position of Y4 output locating instruction (LSB) (EC20H)	R/W
SD350	Current position of Y5 output locating instruction (MSB) (EC20H)	R/W
SD351	Current position of Y5 output locating instruction (LSB) (EC20H)	R/W
SD360	Current position of Y6 output locating instruction (MSB) (EC20H)	R/W
SD361	Current position of Y6 output locating instruction (LSB) (EC20H)	R/W
SD370	Current position of Y7 output locating instruction (MSB) (EC20H)	R/W
SD371	Current position of Y7 output locating instruction (LSB) (EC20H)	R/W

6.10.12 PLS: Pulse output instruction of envelope

LAD: ├──┤ ├──[PLS *(S1)* *(S2)* *(D1)*]	**Applicable to**	**EC20 EC10 EC20H EC10V**
	Influenced flag bit	
IL: PLS *(S1) (S2) (D1)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												

Operand description

S1: the starting D element

S2: output section number. Range: 0~255

D: high-speed pulse output point. Range: EC10, EC20: Y0, Y1;EC10V: Y0,Y1,Y2,Y3; EC20H: Y0, Y2, Y4, Y5, Y6, Y7.

Function description

1. Use the ConstrolStar instruction wizzard to generate the PLS instruction, which can be called like subprograms. When the power flow is ON, the system will output corresponding pulses according to the configuration. You can control ON or OFF of the PG, and set frequency & pulse number.

2. There is no output when the section number is 0.

3. SM80 and SM81 can be used to stop the high-speed pulse output. Other flag bits are the same as other high-speed I/O instructions.

4. The subprogram PLS_SET generated by the AutoStation is as follows (n: D element addr. M: total section number):

LD SM0

DMOV section 1 step frequency Dn

DMOV section 1 step pulse number Dn+2

DMOV section 2 step frequency Dn+4

DMOV section 2 step pulse number Dn+6

DMOV section 3 step frequency Dn+8

DMOV section 3 step pulse number Dn+10

...

DMOV section M step frequency Dn+4M-4

DMOV section M step pulse number Dn+4M-2

DMOV max. speed Dn+4M

MOV min. speed Dn+4M+2

MOV acceleration time Dn+4M+3

MOV deceleration time Dn+4M+4

Note

1. It is recommended to use the PTO instruction wizard to generate PLS instruction. If you write the PLS instruction manually, note that the pulse number of the steps must not be too small. With set acceleration, the pulse number of each step must be bigger than the min. pulse number required by frequency transfer.

2. Use P to stand for the pulse number that a certain step outputs; F_N: frequency of section N; F_{ma}: the maximum

speed; $F_{\min}$: the minimum speed; T_{up}: the acceleration time; T_{down}: the deceleration time.

1) When the speed of step N is bigger than that of step N-1, the pulse number of step N must meet the following condition:

$$P \geq \frac{(F_N + F_{N-1}) \times (F_N - F_{N-1}) \times T_{up}}{2000 \times (F_{\max} - F_{\min})}$$

2) When the speed of step N is smaller than that of step N-1, the pulse number of step N must meet the following condition:

$$P \geq \frac{(F_N + F_{N-1}) \times (F_N - F_{N-1}) \times T_{down}}{2000 \times (F_{\max} - F_{\min})}$$

3. In particular,

1) When N=1, the frequency of step N-1 is used instead of $F_{\min}$ in the above format.

2) When all the step number is 1, that is to say, only one section, the pulse number must meet the following condition:

$$P \geq \frac{(F_1 + F_{\min}) \times (F_1 - F_{\min}) \times (T_{up} + T_{down})}{2000 \times (F_{\max} - F_{\min})}$$

3) The pulse number of the last step must meet the following format:

$$P \geq \frac{(F_M + F_{M-1}) \times (F_M - F_{M-1}) \times (T_{up} + T_{down})}{2000 \times (F_{\max} - F_{\min})}$$

4) The frequency set in every step must be within the range of maximum speed and minimum speed.

5) The maximum total pulse number of all steps is 999,999.

4. Use the transistor output. During the high-speed pulse output, the output current must comply with the related regulations. The waveform at output points (Y0&Port 0, Y1&Port 1) is related to the load: so long as the current does not exceed the rated load current, the smaller the load is, the closer the output waveform is to the set operand.

5. During the execution of the high-speed instruction, so long as the power flow is not OFF, no other instructions can use the same port, unless the high-speed pulse output instruction is invalid.

6. The PLSY, PLSR, PLS and locating instructions can output high-speed pulses through Y0 and Y1. Note that only one instruction can use one output port at one time.

Address	Name	Action and function	R/W
SM80	Y0 high-speed pulse output control	Y0 high-speed pulse output stop instruction	R/W
SM81	Y1 high-speed pulse output control	Y1 high-speed pulse output stop instruction	R/W
SM82	Y0 high-speed pulse output monitor	Y0 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM83	Y1 high-speed pulse output monitor	Y1 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM262	Y002 pulse output stop instruction	After setting, Y002 pulse will be disabled	R/W
SM264	Y004 pulse output stop instruction	After setting, Y004 pulse will be disabled	R/W
SM265	Y005 pulse output stop instruction	After setting, Y005 pulse will be disabled	R/W
SM266	Y006 pulse output stop instruction	After setting, Y006 pulse will be disabled	R/W
SM267	Y007 pulse output stop instruction	After setting, Y007 pulse will be disabled	R/W
SM272	Y002 pulse output monitor (busy/ready)	Y002 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM274	Y004 pulse output monitor (busy/ready)	Y004 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM275	Y005 pulse output monitor (busy/ready)	Y005 high-speed pulse output monitor (ON: busy, OFF: ready))	R
SM276	Y006 pulse output monitor (busy/ready)	Y006 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM277	Y007 pulse output monitor (busy/ready)	Y007 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM88	Evelope loop execution	Evelope loop execution at ON	R/W

Address	Action and function	R/W

Address	Action and function	R/W
SD56	Current output section number when Y000 envelope outputs	R
SD57	Current output section number when Y001 envelope outputs	R
SD88	Rising time of envelope (ms)	R/W
SD89	Falling time of envelope (ms)	R/W
SD252	Current output section number when Y002 envelope outputs	R
SD253	Current output section number when Y003 envelope outputs	R
SD254	Current output section number when Y004 envelope outputs	R
SD255	Current output section number when Y005 envelope outputs	R
SD256	Current output section number when Y006 envelope outputs	R
SD257	Current output section number when Y007 envelope outputs	R
SD50	PLSY accumulated output Y0 total pulse number (MSB)	R/W
SD51	PLSY accumulated output Y0 total pulse number (LSB)	R/W
SD52	PLSY accumulated output Y1 total pulse number (MSB)	R/W
SD53	PLSY accumulated output Y1 total pulse number (LSB)	R/W
SD54	PLSY accumulated output Y1, Y0 total pulse number (MSB)	R/W
SD55	PLSY accumulated output Y1, Y0 total pulse number (LSB)	R/W
SD160	PLSY accumulated output Y2 total pulse number (MSB)	R/W
SD161	PLSY accumulated output Y2 total pulse number (LSB)	R/W
SD162	PLSY accumulated output Y3 total pulse number (MSB)	R/W
SD163	PLSY accumulated output Y3 total pulse number (LSB)	R/W
SD164	PLSY accumulated output Y4 total pulse number (MSB)	R/W
SD165	PLSY accumulated output Y4 total pulse number (LSB)	R/W
SD166	PLSY accumulated output Y5 total pulse number (MSB)	R/W
SD167	PLSY accumulated output Y5 total pulse number (LSB)	R/W
SD168	PLSY accumulated output Y6 total pulse number (MSB)	R/W
SD169	PLSY accumulated output Y6 total pulse number (LSB)	R/W
SD170	PLSY accumulated output Y7 total pulse number (MSB)	R/W
SD171	PLSY accumulated output Y7 total pulse number (LSB)	R/W
SD80	Current position of Y0 output locating instruction (MSB) (EC10)	R/W
SD81	Current position of Y0 output locating instruction (LSB) (EC10)	R/W
SD82	Current position of Y1 output locating instruction (MSB)	R/W
SD83	Current position of Y1 output locating instruction (LSB)	R/W
SD200	Current position of Y0 output locating instruction (MSB) (EC20,EC20H)	R/W
SD201	Current position of Y0 output locating instruction (LSB) (EC20,EC20H)	R/W
SD210	Current position of Y1 output locating instruction (MSB)	R/W
SD211	Current position of Y1 output locating instruction (LSB)	R/W
SD320	Current position of Y2 output locating instruction (MSB)	R/W
SD321	Current position of Y2 output locating instruction (LSB)	R/W
SD330	Current position of Y3 output locating instruction (MSB)	R/W
SD331	Current position of Y3 output locating instruction (LSB)	R/W
SD340	Current position of Y4 output locating instruction (MSB)	R/W
SD341	Current position of Y4 output locating instruction (LSB)	R/W
SD350	Current position of Y5 output locating instruction (MSB)	R/W

Address	Action and function	R/W
SD351	Current position of Y5 output locating instruction (LSB)	R/W
SD360	Current position of Y6 output locating instruction (MSB)	R/W
SD361	Current position of Y6 output locating instruction (LSB)	R/W
SD370	Current position of Y7 output locating instruction (MSB)	R/W
SD371	Current position of Y7 output locating instruction (LSB)	R/W

6.10.13 PLSB: Count pulse with base frequency and acceleration/deceleration output instruction

LAD: ├─┤ ├──[*PLSB (S1) (S2) (S3) (S4) (D)*]	**Applicable to**	**EC10 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: PLSB *(S1) (S2) (S3) (S4) (D)*	**Program steps**	**12**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S3	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S4	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	BOOL			Y												

- **Operand description**

S1: base frequency (Hz). Range: 0~20000Hz. When ***S1*** is outside this range, the system will report instruction operand error, output no pulse, and no hardware resources will be occupied.

S2: maximum frequency (Hz). Range: 10~20000 (Hz). When ***S2*** is outside this range, the system will report instruction operand error, output no pulse, and no hardware resources will be occupied.

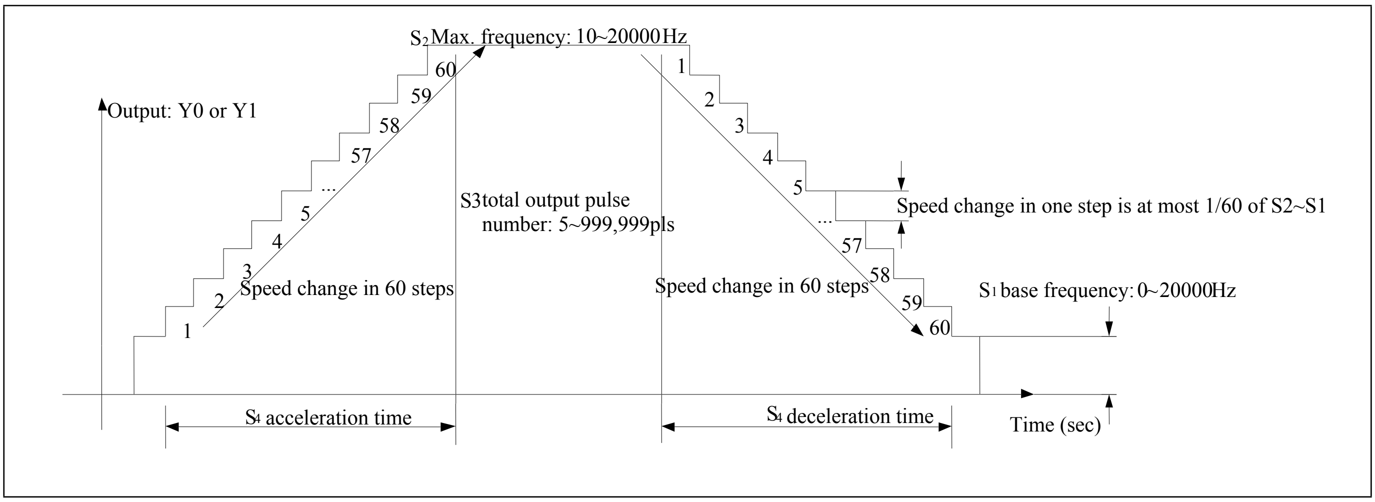
S3: total output pulse number (PLS). Range: 5~999999. When ***S3*** is outside this range, the system will report instruction operand error, output no pulse, and no hardware resources will be occupied.

S4: acceleration time (ms). Range: 0~10000. When ***S4*** is outside this range, the system will report instruction operand error, output no pulse, and no hardware resources will be occupied.

D: pulse output point. Range: EC10, EC20: Y0, Y1; EC20H: Y0, Y2, Y4, Y5, Y6, Y7.

- **Function description**

The PLSB instruction is a high-speed pulse output instruction with base frequency and acceleration/deceleration functions. It is used for locating. Targeting at the specified maximum frequency, the pulse output will accelerate evenly from base frequency. After the output pulse number reaches the preset value, the pulse output will decelerate evenly to base frequency. The operation process is shown in the following figure:


S2Max. frequency: 10~20000Hz
Output: Y0 or Y1
1
2
3
4
5
...
57
58
59
60
Speed change in 60 steps
S3total output pulse number: 5~999,999pls
1
2
3
4
5
...
57
58
59
60
Speed change in 60 steps
Speed change in one step is at most 1/60 of S2~S1
S1 base frequency: 0~20000Hz
S4 acceleration time
S4 deceleration time
Time (sec)

Note

1. The output frequency of this instruction is 10~20,000Hz. When the acceleration/deceleration rate exceeds that range, it will be automatically adjusted according to that range, regardless of the scan cycle.

2. When the operand is set improperly (for example, the pulse number is smaller than corresponding ACC/DEC time and frequency), accelerate or decelerate according to the set operand values and ensure the output pulse number is correct. The output frequency and ACC/DEC time may be smaller than the set operand values.

3. The instruction uses speed change in 60 steps. When the operand is set improperly or the pulse number is small, reduce the steps properly.

4. Use the transistor output. During the high-speed pulse output, the output current must comply with the related regulations. The waveform at output points (Y0&Port 0, Y1&Port 1) is related to the load: so long as the current does not exceed the rated load current, the smaller the load is, the closer the output waveform is to the set operand.

5. During the execution of the high-speed instruction, so long as the power flow is not OFF, no other instructions can use the same port, unless the high-speed pulse output instruction is invalid.

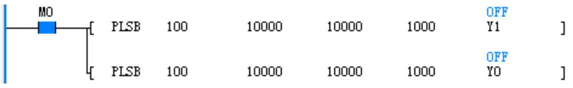
6. Using two PLSB instructions can output two independent pulses. You can also use PWM or PLSY instruction to get independent pulse outputs at different output ports.

7. When multiple PWM, PLSY or PLSB instructions work on the same output point, the first valid instruction will control the state of the output point, and others will not affect the output point state.

8. Just like other high-speed instructions (DHSCS, DHSCR, DHSZ, DHSP, DHST and HCNT), the PLSB instruction must meet the system’s requests on high-speed I/O.

9. The high-speed, envelope and locating instructions can output high-speed pulses through Y0 and Y1. Note that only one instruction can use one output port at one time.

Example



```
LD     M0
PLSR   100   10000   10000   1000   Y1
PLSR   100   10000   10000   1000   Y0
```

1. When M0 is ON, Y0 and Y1 output 10000 pulses respectively at set frequencies. When M0 changes from OFF to ON, pulses will be output again. When M0 is OFF, the output will stop.

2. The operand change during the execution of the instruction will not be valid until the next time this instruction is executed.

3. SM80&SM81 controls the ON/OFF of the output at Y0 and Y1 respectively. When SM80 or SM81 is 1, the output will stop.

4. SM82&SM83 are the output monitors of Y0&Y1 respectively. SM82&SM83 will be ON when the output is going on, or OFF when the output is over.

5. SD50: the MSB of the output pulse number at Y0 for PLSB instruction.

SD51: the LSB of the output pulse number at Y0 for PLSB instruction.

SD52: the MSB of the output pulse number at Y1 for PLSB instruction.

SD53: the LSB of the output pulse number at Y1 for PLSB instruction.

SD54: the MSB of the total output pulse number at Y0 and Y1 for PLSB instruction.

SD55: the LSB of the total output pulse number at Y0 and Y1 for PLSB instruction.

6. SD50~SD55 can be changed with the instruction DMOV or MOV, or through the AutoStation software.

Relevant elements:

Address	Name	Action and function	R/W
SM80	Y0 high-speed pulse output control	Y0 high-speed pulse output stop instruction	R/W
SM81	Y1 high-speed pulse output control	Y1 high-speed pulse output stop instruction	R/W
SM82	Y0 high-speed pulse output monitor	Y0 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM83	Y1 high-speed pulse output monitor	Y1 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM262	Y002 pulse output stop instruction	After setting, Y002 pulse will be disabled	R/W
SM264	Y004 pulse output stop instruction	After setting, Y004 pulse will be disabled	R/W
SM265	Y005 pulse output stop instruction	After setting, Y005 pulse will be disabled	R/W
SM266	Y006 pulse output stop instruction	After setting, Y006 pulse will be disabled	R/W
SM267	Y007 pulse output stop instruction	After setting, Y007 pulse will be disabled	R/W
SM272	Y002 pulse output monitor (busy/ready)	Y002 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM274	Y004 pulse output monitor (busy/ready)	Y004 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM275	Y005 pulse output monitor (busy/ready)	Y005 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM276	Y006 pulse output monitor (busy/ready)	Y006 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM277	Y007 pulse output monitor (busy/ready)	Y007 high-speed pulse output monitor (ON: busy, OFF: ready)	R

Address	Action and function	R/W

Address	Action and function	R/W
SD50	PLSY accumulated output Y0 total pulse number (MSB)	R/W
SD51	PLSY accumulated output Y0 total pulse number (LSB)	R/W
SD52	PLSY accumulated output Y1 total pulse number (MSB)	R/W
SD53	PLSY accumulated output Y1 total pulse number (LSB)	R/W
SD54	PLSY accumulated output Y1, Y0 total pulse number (MSB)	R/W
SD55	PLSY accumulated output Y1, Y0 total pulse number (LSB)	R/W
SD160	PLSY accumulated output Y2 total pulse number (MSB)	R/W
SD161	PLSY accumulated output Y2 total pulse number (LSB)	R/W
SD164	PLSY accumulated output Y4 total pulse number (MSB)	R/W
SD165	PLSY accumulated output Y4 total pulse number (LSB)	R/W
SD166	PLSY accumulated output Y5 total pulse number (MSB)	R/W
SD167	PLSY accumulated output Y5 total pulse number (LSB)	R/W
SD168	PLSY accumulated output Y6 total pulse number (MSB)	R/W
SD169	PLSY accumulated output Y6 total pulse number (LSB)	R/W
SD170	PLSY accumulated output Y7 total pulse number (MSB)	R/W
SD171	PLSY accumulated output Y7 total pulse number (LSB)	R/W
SD80	Current position of Y0 output locating instruction (MSB) (EC10)	R/W
SD81	Current position of Y0 output locating instruction (LSB) (EC10)	R/W
SD82	Current position of Y1 output locating instruction (MSB)	R/W
SD83	Current position of Y1 output locating instruction (LSB)	R/W
SD200	Current position of Y0 output locating instruction (MSB) (EC20,EC20H)	R/W
SD201	Current position of Y0 output locating instruction (LSB) (EC20,EC20H)	R/W
SD210	Current position of Y1 output locating instruction (MSB)	R/W
SD211	Current position of Y1 output locating instruction (LSB)	R/W
SD320	Current position of Y2 output locating instruction (MSB)	R/W
SD321	Current position of Y2 output locating instruction (LSB)	R/W
SD340	Current position of Y4 output locating instruction (MSB)	R/W
SD341	Current position of Y4 output locating instruction (LSB)	R/W
SD350	Current position of Y5 output locating instruction (MSB)	R/W
SD351	Current position of Y5 output locating instruction (LSB)	R/W
SD360	Current position of Y6 output locating instruction (MSB)	R/W
SD361	Current position of Y6 output locating instruction (LSB)	R/W
SD370	Current position of Y7 output locating instruction (MSB)	R/W
SD371	Current position of Y7 output locating instruction (LSB)	R/W

6.10.14 PWM: Pulse output instruction

LAD: ├─┤ ├──[PWM *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC20 EC10A EC10V EC10 EC20H**
	Influenced flag bit	
IL: PWM *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	BOOL			Y												

Operand description

S1: pulse width (ms/μs).

Range: 0~32767 (ms). When ***S1*** is bigger than 32767, the system will report illegal instruction operand, and no hardware resources will be occupied.

You can change the output pulses in real-time by changing ***S1*** during the execution of the instruction. When SM84=0, the unit of ***S1*** is ms; when SM84=1, the unit of ***S1*** is μs.

S2: pulse cycle (ms).

Range: 1~32767. When S2 is outside the range, the system will report illegal instruction operand, no pulse will be output, and no system resources will be occupied.

You can change the output pulses in real-time by changing ***S2*** during the execution of the instruction.

S2 must be bigger than ***S1***, or the system will report illegal instruction operand, no pulse will be output, and no system resources will be occupied.

D: high-speed pulse output point. Range: EC10, EC20: Y0, Y1; EC10V: Y0,Y1,Y2,Y3; EC20H: Y4, Y5, Y6, Y7.

Function description

Output PWM pulses with the width of S1 and cycle of S2 at the port designated by D.

Note

1. When S1 is 0, Y0 or Y1 output is OFF. When S1 is equates S2, Y0 or Y1 output is ON.

2. The waveform at output points (Y0&Port 0, Y1&Port 1) is related to the load: so long as the current does not exceed the rated load current, the smaller the load is, the closer the output waveform is to the set operand. Therefore, in order to output high-speed pulses, the load current at the PLC output transistor must be big, but smaller than the rated load current.

3. During the execution of the high-speed instruction, so long as the power flow is not OFF, no other instructions can use the same port, unless the high-speed pulse output instruction is invalid.

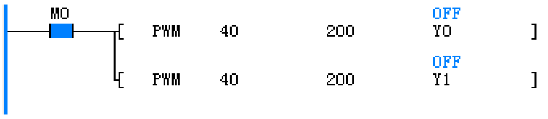
4. Using two PWM instructions can output two independent pulses at Y0 and Y1. You can also use the PLSY and PLSR instructions to get independent pulse outputs at different output ports (Y0, Y1).

5. When multiple PWM, PLSY or PLSR instructions work on the same output point, the first valid instruction will control the state of the output point, and others will not affect the output point state.

Relevant elements:

6. Just like other high-speed instructions (DHSCS, DHSCR, DHSZ, DHSP, DHST and HCNT), the PWM instruction must meet the system's requests on high-speed I/O.

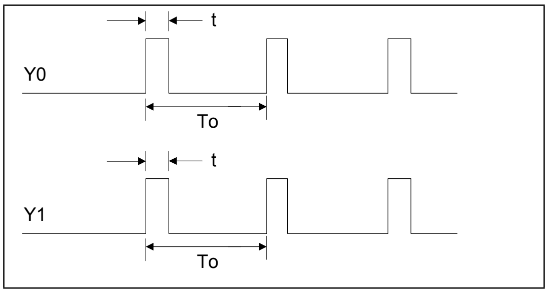
Example



LD M0

PWM 40 200 Y0

PWM 40 200 Y1



Where "t" is the pulse width and T0 is the pulse cycle.

1. When M0 is ON, Y0 and Y1 output PWM pulses with the width of 40ms and cycle of 200ms. When M0 is OFF, the output will stop, regardless of the scan cycle.

2. SM80 and SM81 control the output ON/OFF of Y0 and Y1 respectively. When SM80 and SM81 are ON, the output will stop.

3. SM82 and SM83 monitor the output of Y0 and Y1 respectively. When M0 is OFF, SM82 and SM83 are OFF.

Address	Name	Action and function	R/W
SM80	Y0 high-speed pulse output control	Y0 high-speed pulse output stop instruction	R/W
SM81	Y1 high-speed pulse output control	Y1 high-speed pulse output stop instruction	R/W
SM82	Y0 high-speed pulse output monitor	Y0 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM83	Y1 high-speed pulse output monitor	Y1 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM84	PWM time base unit	Microsecond at ON and millisecond at OFF	
SM264	Y004 pulse output stop instruction	After setting, Y004 pulse will be disabled	R/W
SM265	Y005 pulse output stop instruction	After setting, Y005 pulse will be disabled	R/W
SM266	Y006 pulse output stop instruction	After setting, Y006 pulse will be disabled	R/W
SM267	Y007 pulse output stop instruction	After setting, Y007 pulse will be disabled	R/W
SM274	Y004 pulse output monitor (busy/ready)	Y004 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM275	Y005 pulse output monitor	Y005 high-speed pulse output monitor (ON: busy, OFF:	R

	(busy/ready)	ready)	
SM276	Y006 pulse output monitor (busy/ready)	Y006 high-speed pulse output monitor (ON: busy, OFF: ready)	R
SM277	Y007 pulse output monitor (busy/ready)	Y007 high-speed pulse output monitor (ON: busy, OFF: ready)	R

6.11 Control calculation instruction

6.11.1 PID: PID instruction

LAD: ├─┤ ├──[PID (S1) (S2) (S3) (D)	**Applicable to**	**EC20 EC10A EC10V EC10 EC20H**
	Influenced flag bit	
IL: PID *(S1) (S2) (S3) (D)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D					R	√
S2	INT								D					R	√
S3	INT								D					R	√
D	INT								D					R	√

Operand description

D: calculation result output after the program is executed (MV)

S1: preset value (SV)

S2: current value (PV)

S3: sampling time (Ts). Range: 1~32767(ms). It must be set bigger than the calculation time.

S3+1: action, alarm and thresholds setting

Bit	Value and meaning	
	0	1
0	Forward	Reverse
1	Process value alarm disabled	Process value alarm enabled
2	Output value alarm disabled	Output value alarm enabled
3~4	Reserved	
5	Output threshold setting disabled	Output threshold setting enabled
6~15	Reserved	

S3+2: input filter constant (α). Range: 0~99 [%]. Zero means no input filtering function.

S3+3: proportional gain (Kp). Range: 1~32767 [%].

S3+4: integral time constant (TI). Range: 0~32767(×100ms). Zero means limit, or no integral.

S3+5: differential gain (KD). Range: 0~100[%]. Zero means no differential gain.

S3+6: differential time (TD). Range: 0~32767(×10ms). Zero means no differential processing.

S3+7~S3+14: internal data register for PID operation

S3+15: PID process value (positive change) alarm point. Range: 0~32767 (when BIT1 of ***S3+1*** is 1).

S3+16: PID process value (negative change) alarm point. 0~32767 (when BIT1 of ***S3+1*** is 1).

S3+17: PID output value (positive change) alarm point 0~32767 (when BIT2&BIT5 of ***S3+1*** are 1&0 respectively).

Output upper limit: -32768~32767 (when BIT2&BIT5 of ***S3+1*** are 0&1 respectively).

S3+18: PID output value (negative change) alarm point. Range: 0~32767 (when BIT2&BIT5 of ***S3+1*** are 1&0 respectively).

Output lower limit: -32768~32767 (when BIT2&BIT5 of ***S3+1*** are 0&1 respectively).

S3+19: PID alarm output

BIT0 process value (positive change) overflows

Bit1 process value (negative change) overflows

Bit2 output value (positive change) overflows

Bit3 output value (negative change) overflows

Where, ***S3~S3+6*** are the mandatory user set operands, while ***S3+15~S3+19*** are optional user set operands. You can set the operands through the PID instruction wizard of the AutoStation.

Function description

1. PID calculation will be carried out when the power flow is ON and it is the sampling time.

2. Multiple PID instructions can be executed simultaneously (no limit on the loop number). However, note that the elements used as S1, S2, S3 or D should be different.

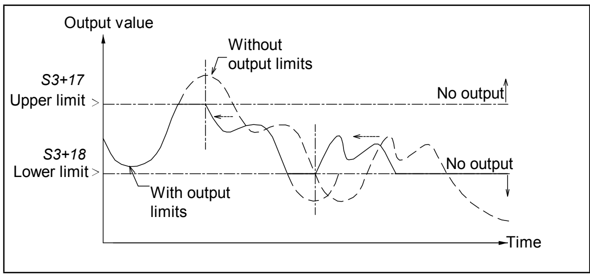
3. The PID instruction is applicable to timed interrupt subprograms, ordinary subprograms and the main program. Note that before using the PID instruction, confirm the operand settings and clear the internal data registers S3+7 first.

4. The input filtering constant can smooth the change of measured value.

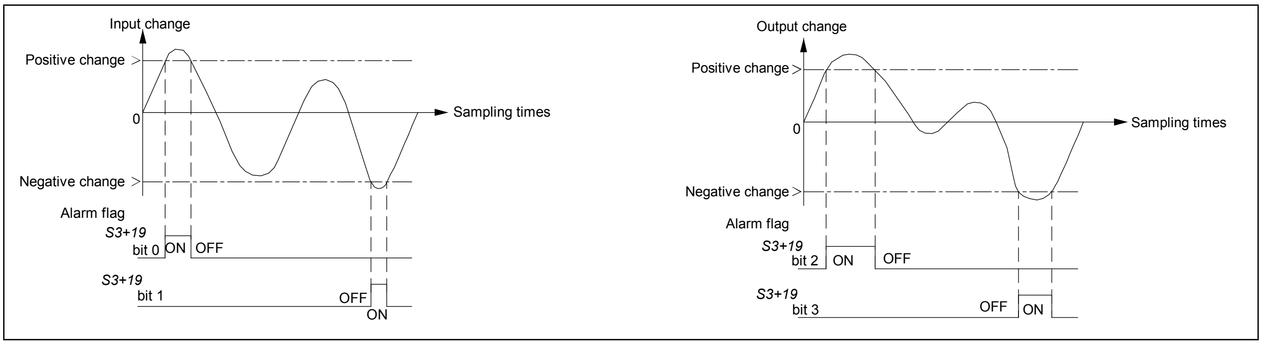
5. The differential gain can smooth the change of output value.

6. Action direction: BIT0 of S3+1 is used to set the forward (positive reactioin) and reverse (negative reaction) of the system.

7. Output thresholds: when the output threshold setting is enabled (BIT5&BIT2 of S3+1 are 1 and 0 respectively), the integral of PID can be controlled from becoming too big. The output value is shown as below:



8. Alarm setting: when the output thresholds are set valid (in S3+1, BIT1 is 1, BIT2 is 1 and BIT 5 is 0), the PID instruction will compare the current value with the preset value in S3+15~S3+18. If the current value is bigger than the preset value, PID will report alarm, and the corresponding BITs in S3+19 will be set. In this way, you can monitor the input change and output change. See the following figures.



9．Basic PID equations:

Direction	PID equations
Forward	$\Delta MV = KP\left\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I}EV_n + D_n\right\}$ $EV_n = PV_{nf-1} - SV$ $D_n = \frac{T_D}{T_S + \alpha_D * T_D}(PV_{nf} + PV_{nf-2} - 2PV_{nf-1}) + \frac{\alpha_D * T_D}{T_S + \alpha_D * T_D} * D_{n-1}$ $MV_n = \sum \Delta MV$
Reverse	$\Delta MV = KP\left\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I}EV_n + D_n\right\}$ $EV_n = SV - PV_{nf-1}$ $D_n = \frac{T_D}{T_S + \alpha_D * T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{\alpha_D * T_D}{T_S + \alpha_D * T_D} * D_{n-1}$ $MV_n = \sum \Delta MV$

Operand description:

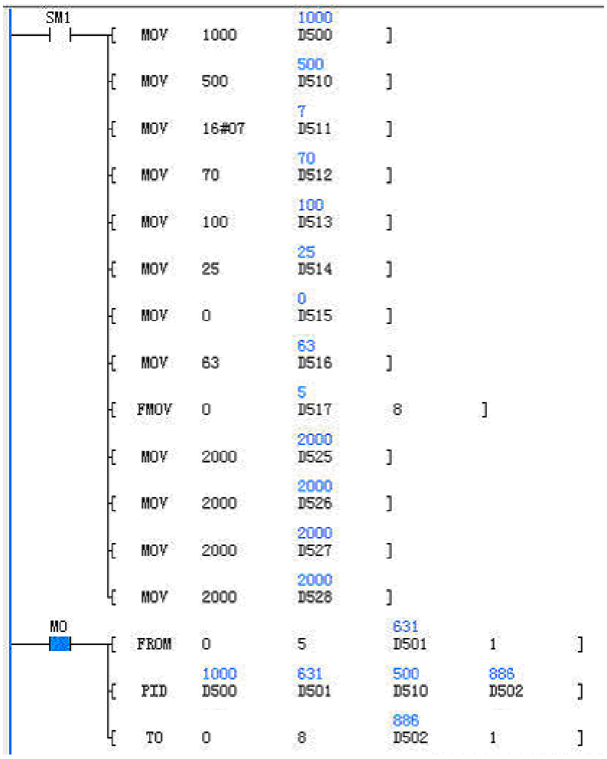
Symbol	Description		Symbol	Description
EV_n	The current error value		D_n	The current differential value
EV_{n-1}	The previous error value		D_{n-1}	The previous differential value
SV	The set point value		KP	The proportion gain

PV_{nf}	The calculated process value		T_S	The sampling time
PV_{nf-1}	The previsou process value		T_I	The integral time
PV_{nf-2}	The second previous Process Value		T_D	The differential time
ΔMV	The change in the output manipulation values		α_D	The differential gain
MV	The current output manipulation value			

Example

```
// PID initialization. If the control operands are the same, you can initialize the operands only once.
LD      SM1                     //Initialization, executed only once
MOV     1000     D500           //Setting target value
MOV     500      D510           //Sampling time(Ts)   Range: 1~32767(ms). It must be bigger than the
                                     //calculation time
MOV     7        D511                //Action direction
MOV     70       D512                //Input filtering constant (α) Range: 0~99[%]. Zero means no input filtering
MOV     100      D513                //Proportional gain (Kp) Range:1~32767[%]
MOV     25       D514                //Integral time (TI) Range: 0~32767(×100ms). Zero means limit, or no integral
MOV     0        D515                //Differential gain (KD) Range: 0~100[%]. Zero means no differential gain
MOV     63       D516                //Differential time (TD) Range: 0~32767 (×10ms). Zero means no differential
                                     // processing
FMOV         0        D517       8    //Clearing the memory for the transit data of PID calculation
MOV     2000     D525                //Process value (positive change) alarm setting 0~32767
MOV     2000     D526                //Process value (negative change) alarm setting 0~32767
MOV     2000     D527                //Output value (positive change) alarm setting 0~32767
MOV     2000     D528                //Output value (negative change) alarm setting 0~32767
//PID instruction execution
LD      M0                      //User-controlled PID calculation program
FROM         0    5   D501          1    //Input current measured value (users can input measured values
                                          //according to the actual situation)
PID     D500 D501 D510 D502      //PID instruction: PID S1 S2 S3 D
TO      0    8    D502          1    //PID calculation result is fed back to the controlled system (users can
                                          //handle the PID calculation result according to the actual situation)
```

The LAD of the above instructions is shown below:



```
 SM1                          1000
─┤ ├──┬[ MOV     1000         D500      ]
      │                       500
      ├[ MOV     500          D510      ]
      │                       7
      ├[ MOV     16#07        D511      ]
      │                       70
      ├[ MOV     70           D512      ]
      │                       100
      ├[ MOV     100          D513      ]
      │                       25
      ├[ MOV     25           D514      ]
      │                       0
      ├[ MOV     0            D515      ]
      │                       63
      ├[ MOV     63           D516      ]
      │                       5
      ├[ FMOV    0            D517      8         ]
      │                       2000
      ├[ MOV     2000         D525      ]
      │                       2000
      ├[ MOV     2000         D526      ]
      │                       2000
      ├[ MOV     2000         D527      ]
      │                       2000
      └[ MOV     2000         D528      ]
 M0                                     631
─┤ ├──┬[ FROM    0            5         D501      1         ]
      │         1000          631       500       886
      ├[ PID     D500         D501      D510      D502      ]
      │                                 886
      └[ TO      0            8         D502      1         ]
```

The PLC will initialize the PID operands only in the first scan cycle. When X2 is ON, the current measured value will be read from external A/D module (the actual situation could be different), assigned to the corresponding elements, and the PID calculation will be carried out. The calculation result will be converted into analog signals through the external D/A module (the actual situation could be different) and fed to the controlled system.

Note

1. The operand D should be a register outside of the Saving Range. Otherwise, it should be cleared (LD SM0 MOV 0 D****) in the first operation.

2. The PID instructions occupies 20 consecutive registers starting with S3.

3. The maximum error of sampling time (TS) is -(scan cycle +1ms)~+(scan cycle). When TS is small, the PID effect will be affected. It is recommended to use PID instruction in the timed interrupt.

4. When setting the PID output thresholds, if the upper limit is smaller than the lower limit, the system will report operand error, and no PID calculation will be carried out.

5. When the process value alarm and output value alarm are enabled, S3+15~S3+18 cannot be set negative, or the system will report operand error, and no PID calculation will be carried out.

6. Setting BIT2 and BIT5 of S3+1 ON at the same time will be regarded as invalid (essentially the same as setting BIT2 and BIT5 OFF), and there will be no limit, nor output value alarm.

7. When the PID control operands (S3~S3+6) are set outside their ranges, the system will report operand error, and no PID calculation will be carried out.

8. When the sampling time is smaller than the scan cycle, if there is data overflow or result overflow during the calculation, there will be no alarm, and the PID calculation continues.

9. The PID operands must be initialized before the PID instruction is executed the first time. If the operands remain the same during the operation, and the related operand elements will not be covered by other programs, you can initialize the PID operands only once. However, if the data in the transit data registers are changed during the PID calculation, the calculation result will be incorrect.

6.11.2 RAMP: Ramp wave signal output instruction

LAD: ├─┤ ├──[RAMP (S1) (S2) (D1) (S3) (D2)]	**Applicable to**	**EC20 EC10A EC10V EC10** **EC20H**
	Influenced flag bit	
IL: RAMP *(S1) (S2) (D1) (S3) (D2)*	**Program steps**	**12**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	INT								D				V		R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D2	BOOL			Y	M	S	LM				C	T				

■ **Operand description**

S1: starting value

S2: end value

D1: output value

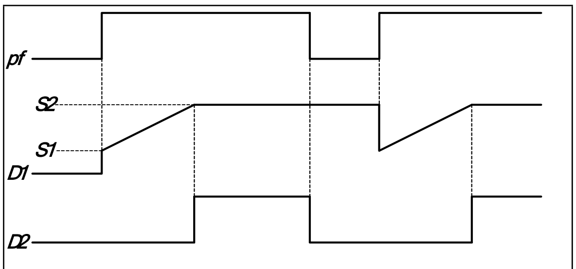
S3: step number (***S3***>0, or system will report operand error and do not execute the calculation)

D2: output state 0

■ **Function description**

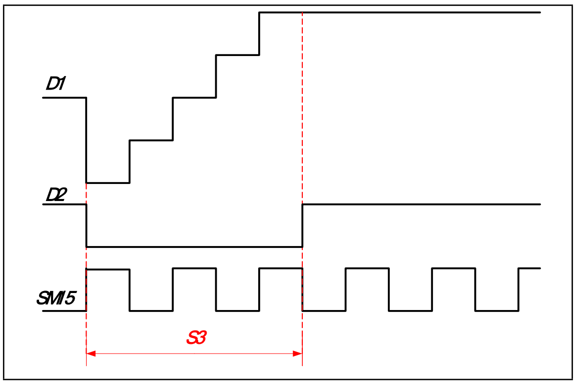
In each scan cycle, when the power flow is ON, this instruction can determine the increment and current output value D1 according to the ramp wave height and step number. When the output value D1 reaches S2, it will keep stable, and the output state D2 will be set ON. If the power flow falls, the output state D2 will be set OFF, but the output value D1 will not change, until the power flow rises again, when D1 will be initialized as S1, and continue to conduct the next ramp calculation.

See the following figure:



pf: state of the power flow

Analysis of the execution process of the ramp instruction is shown in the following figure (S3=5):



■ **Note**

1. If the result is not divisible when calculating the program steps, round off to the nearest whole number.

2. The instruction will generate one ramp data upon every rising edge.

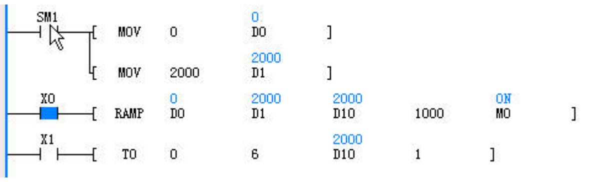
3. When S1=S2, D1=S2, D2=ON.

4. The total number of RAMP, HACKLE and TRIANGLE instructions in a program should not exceed 100.

■ **Example**

```
//Initialize registers upon the first scan cycle after the power-on
LD          SM1
MOV         0 D0
MOV         2000 D1
//Execute RAMP instruction when X0 is ON
LD          X0
RAMP            D0 D1 D10 1000 M0
//Output the ramp function result to external DA module when X1=ON to generate ramp wave form
LD          X1
TO          0 6 D10 1
```

The LAD of the preceding instructions is shown below:



1. When X0 is ON, D10 (in the first cycle, D10=D0=0) will increase by 2 (2000/1000) in every scan cycle. When D10= D1=2000, D10 will increase no more, and M0 will be ON. During the generation of the ramp function, if the power flow falls, the output state D2 will be OFF, the output value D1 will keep its current value until the next rising edge, when D10=D0 and a new ramp starts.

2. You can use an external special module to convert the data into analog waveform.

6.11.3 HACKLE: Hackle wave signal output instruction

LAD: HACKLE (S1) (S2) (D1) (S3) (D2)	**Applicable to**	**EC20 EC10A EC10V EC10** **EC20H**
	Influenced flag bit	
IL: HACKLE ***(S1) (S2) (D1) (S3) (D2)***	**Program steps**	**12**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	INT								D				V		R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D2	BOOL			Y	M	S	LM				C	T				

■ **Operand description**

S1: starting value

S2: end value

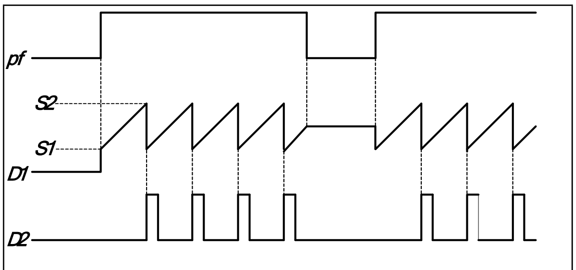
D1: output value

S3: step number (***S3***>0, or system will report operand error and do not execute the calculation)

D2: output state

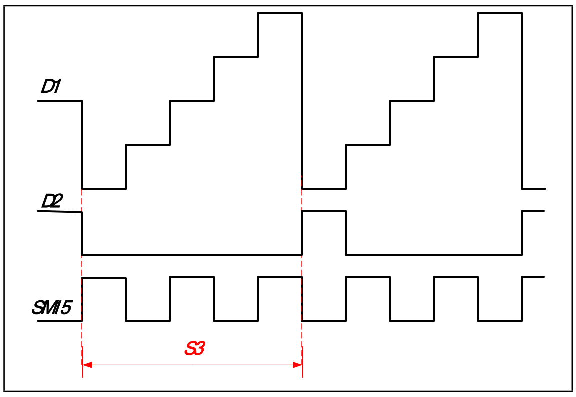
■ **Function description**

In each scan cycle, when the power flow is ON, this instruction can determine the increment and current output value D1 according to the hackle wave height and step number. When the output value reaches S2, it will be initialized as S1 and the state output D2 will be set ON. If the power flow in the next scan cycle is still ON, D2 will be set OFF to produce the next hackle wave. If the power flow falls, the output state D2 will be OFF, and the output value D1 will keep its current value until the next rising edge, when the output value D1 will be initialized as S1, and the next hackle wave will be created, as shown in the following figure.



pf: state of the power flow

The analysis of the hackle wave instruction is shown in the following figure (S3=5):



■ **Note**

1. If the result is not divisible when calculating the program steps, round off to the nearest whole number.

The instruction will generate a series of continuous hackle wave data so long as the power flow keep ON.

2. When S1=S2, D1=S2, D2=ON (no counting pulse).

3. The total number of RAMP, HACKLE and TRIANGLE instructions in a program should not exceed 100.

■ **Example**

//Initialize registers upon the first scan cycle after power-on

```
LD          SM1
MOV         0 D0
MOV2000 D1
```

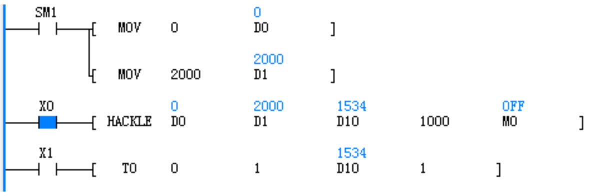
//Execute HACKLE instruction when X0 is ON

```
LD        X0
HACKLE    D0 D1 D10 1000 M0
```

//When X1 is ON, output the result of ramp function to external DA module to generate hackle waveform

```
LD        X1
TO        0 1 D10 1
```

The LAD for the preceding instruction is shown in the following figure:



1. When X0 is ON, D10 (in the first cycle, D10=D0=0) will increase by 2 (2000/1000) in every scan cycle. When D10=D1=2000, M0 will be ON. In the next scan cycle, if X0 keeps ON, D10=D0=0, and M0 is OFF, the next hackle wave will start. If the power flow falls, the output state D2 will be OFF, but the output value D1 will keep its current value until the next rising edge, when D1 will be initialized as S1, and a new hackle wave starts.

2. You can use an external special module to convert the data into analog waveform.

6.11.4 TRIANGLE: Triangle wave signal output instruction

LAD: TRIANGLE (S1) (S2) (D1) (S3) (D2)	Applicable to	EC20 EC20H	EC10A	EC10V	EC10
	Influenced flag bit				
IL: TRIANGLE *(S1) (S2) (D1) (S3) (D2)*	**Program steps**	**12**			

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D1	INT								D				V		R	√
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D2	BOOL			Y	M	S	LM				C	T				

■ **Operand description**

S1: starting value

S2: end value

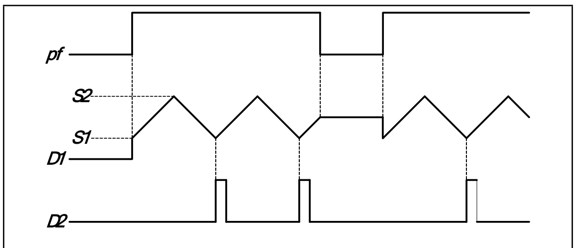
D1: output value

S3: step number (***S3***>0, or system will report operand error, and do not execute the calculation)

D2: output state

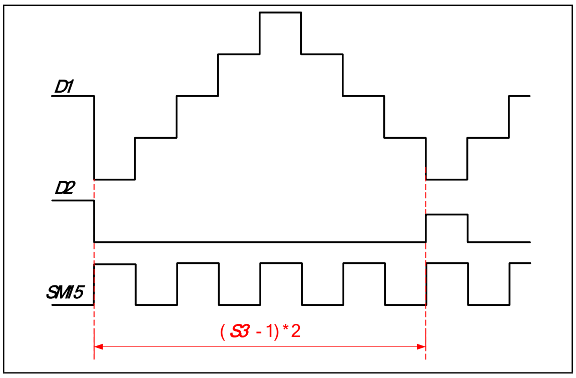
■ **Function description**

In each scan cycle, when the power flow is ON, this instruction can determine the increment and current output value D1 according to the triangle wave height and step number. When the output value reaches S2, the rising half of the triangle is complete, the increment direction of the output value will change and generate the falling half of the triangle. When the output value D1 reaches S1 again, the state output D2 will be set ON. In the next scan cycle, if the power flow keeps ON, the state output D2 will be set OFF and the next triangle will be generated. If the power flow falls, the output state D2 will be OFF, the output value D1 will keep its current value until the power flow rises again, when D1 will be initialized as S1, and a new triangle wave will start. See the following figure:



pf: state of the power flow

The analysis of the execution of the triangle instruction is shown in the following figure (S3=5):



■ **Note**

1. If the result is not divisible when calculating the program steps, round off to the nearest whole number.

2. The instruction will generate a series of continuous triangle wave data so long as the power flow keep ON

3. When S1=S2, D1=S2, D2=ON (no counting pulse).

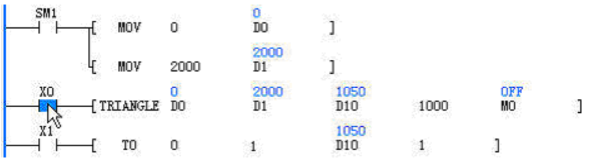
4. The cycle of the triangle wave is (S3-1)×2.

5. The total number of RAMP, HACKLE and TRIANGLE instructions in a program should not exceed 100.

■ **Example**

```
//Initialize registers upon the first scan cycle after power-on
LD      SM1
MOV     0 D0
MOV     2000 D1
//Execute TRIANGLE instruction when X0 is ON
LD      X0
TRIANGLE D0 D1 D10 1000 M0
//When X1 is ON, output the result of ramp function to external DA module to generate triangle waveform
LD      X1
TO      0 1 D10 1
```

The LAD of the preceding instruction is shown in the following figure:



```
 SM1                0         
─┤ ├──┬[ MOV   0     D0     ]
      │            2000
      └[ MOV   2000  D1     ]
 X0         0      2000    1050             OFF
─┤ ├───[TRIANGLE D0  D1    D10      1000    M0      ]
 X1                        1050
─┤ ├───[ TO    0     1     D10      1       ]
```

1. When X0 is ON, D10 (in the first cycle, D10=D0=0) will increase by 2 (2000/1000) in every scan cycle. When D10=D1=2000, the rising half of the triangle is complete, and D10 will decrease by 2 in every scan cycle that follows. When D10=D0=0, a complete triangle is complete, and M0 is ON. In the next scan cycle, if X0 keeps ON, and M0 is OFF, the next triangle wave will start. If the power flow falls, the output state D2 will be OFF, but the output value D1 will keep its current value until the next rising edge, when D1 will be initialized as S1, and a new triangle wave starts.

2. You can use an external special module to convert the data into analog waveform.

6.11.5 ABSD: Absolute drum control instruction

LAD: ├──┤ ├──[ABSD *(S1)* *(S2)* *(D)* *(S3)*]	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: ABSD *(S1)* *(S2)* *(D)* *(S3)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT		KnX	KnY	KnM	KnS			D		C	T		R	√
S2	INT										C				√
D	BOOL			Y	M	S									
S3	WORD	Constant													

■ **Operand description**

S1: starting element SN of table data (rising edge, falling edge). k=4.

S2: counter SN for monitoring the current value compared with table data

D: output starting elements SN

S3: rows of the table and points of output element; 1≤***S3***≤64

■ **Function description**

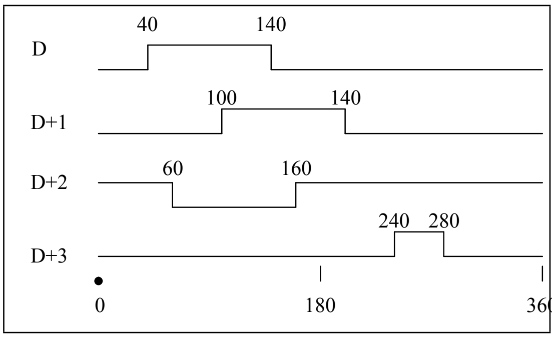
1. Compare n rows of table data starting with S1 (occupy n rows×2 points) with the current value S2 of the counter for ON/OFF control on D output of consecutive n points.

2. Change the rising points/falling points by changing the data of S1~S1+n×2.

3. Fill in the table from S1 to S1+2n+1 as follows:

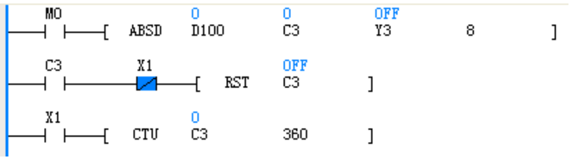
Rising point	Sample data	Falling point	Sample data	Output
S1	40	S1+1	140	D
S1+2	100	S1+3	200	D+1
S1+4	160	S1+5	60	D+2
S1+6	240	S1+7	280	D+3
…	…	…	…	…
S1+2n		S1+2n+1		D+n-1

4. When the instruction is ON, n points starting with D will change as follows:



■ **Example**

Rotate at a time (0~360°) to output ON/OFF (rotatry angle signal of 1 pulse per degree)



M0 is power flow input, X1 is the rotatry angle signal of 1 pulse per degree

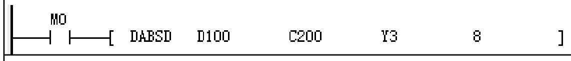
■ **Note**

1. When designating the element in S1, k=4.

2. When designating the counter SN in S2, the range is C0~C199.

3. The instruction is affected by scan cycle.

6.11.6 DABSD: Double word absolute drum control instruction

LAD: 	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DABSD *(S1)* *(S2)* *(D)* *(S3)*	**Program steps**	**11**

Operand	Type	Applicable elements													Indexed addressing
S1	DINT		KnX	KnY	KnM	KnS			D		C	T		R	√
S2	DINT										C				√
D	BOOL			Y	M	S									
S3	WORD	Constant													

■ **Operand description**

S1: starting element SN of table data (rising edge, falling edge). k=8.

S2: counter SN for monitoring the current value compared with table data. Range: C200~C255.

D: output starting elements SN

S3: rows of the table and points of output element; 1≤***S3***≤64

■ **Function description**

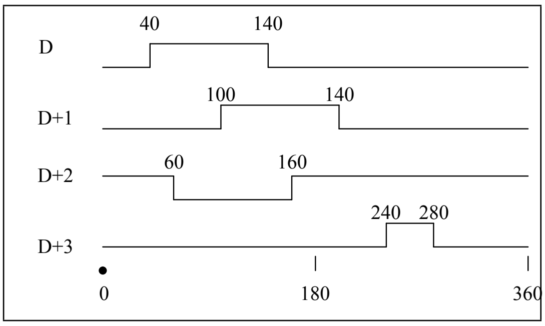
1. Compare n rows of table data starting with S1 (occupy n rows×4 points) with the current value S2 of the counter for ON/OFF control on D output of consecutive n points.

2. Change the rising points/falling points by changing the data of [S1+1, S1]~[S1+(n×2)+3, S1+(n×2)+2].

3. Fill in the table from [S1, S1+1] to [S1, S1+1]+4n+3 as follows:

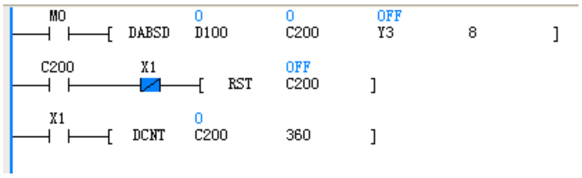
Rising point	Sample data	Falling point	Sample data	Output
[S1+1, S1]	40	[S1+3, S1+2]	140	D
[S1+5, S1+4]	100	[S1+7, S1+6]	200	D+1
[S1+9, S1+8]	160	[S1+11, S1+10]	60	D+2
[S1+13, S1+12]	240	[S1+15, S1+14]	280	D+3
…	…	…	…	…
[S1+4n+1, S1+4n]		[S1+4n+3, S1+4n+2]		D+n-1

4. When the instruction is ON, n points starting with D will change as follows:



■ **Example**

Rotate at a time (0~360°) to output ON/OFF (rotatry angle signal of 1 pulse per degree)



M0 is instruction input and X1 is the rotatry angle signal of 1 pulse per degree. SM200 is OFF by default, so DCNT instruction is counting up.

■ **Note**

1. When designating the element in S1, k=8.

2. When designating the counter SN in S2, the range is C200~C255.

3. The instruction is affected by scan cycle.

6.11.7 ALT: Alternate output instruction

LAD:	Applicable to	EC20 EC20H EC10V
├┤ ├[ALT (*D*)]	Influenced flag bit	Zero, carry, borrow
IL: ALT (*D*)	Program steps	11

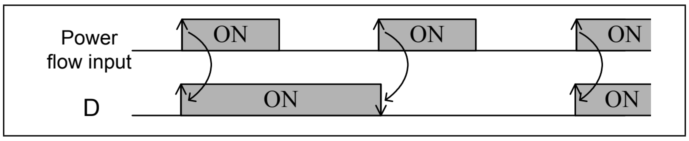
Operand	Type	Applicable elements													Indexed addressing
D	BOOL			Y	M	S									

Operand description

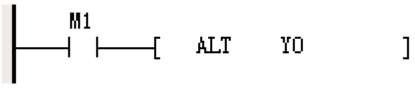
D: element address of alternate output

Function description

When the power flow is ON, the element will act reversely in each scan cycle, as shown below:



Example



6.12 Communication instruction

6.12.1 Modbus: Master station communication instruction

LAD:	**Applicable to**	**EC20 EC10 EC20H EC10V**
├┤ ├[MODBUS (*S1*) (*S2*) (*S3*)]	**Influenced flag bit**	
IL: Modbus (*S1*) (*S2*) (*S3*)	**Program steps**	**8**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	INT	D	V											R	
S3	INT	D												R	√

Operand description

S1: designated communication channel

S2: starting address of the data to be transmitted

S3: starting address of the data to be received

Function description

1. When being a master station and the input conditions are met, the system will transmit the data stored in the unit starting with S2, and then receive the data and store it to the unit starting with S3.

2. When being a slave station, the system needs no instruction control for transceiving data.

3. This instruction is executed upon the rising edge.

Note

1. Sending data through Modbus, whether the data is in RTU mode or ASCII mode, you only need to store the RTU-mode data into the unit starting with S2. You do not need to store the starting character, ending character and checksum, because they will be added to the data automatically in the sending process.

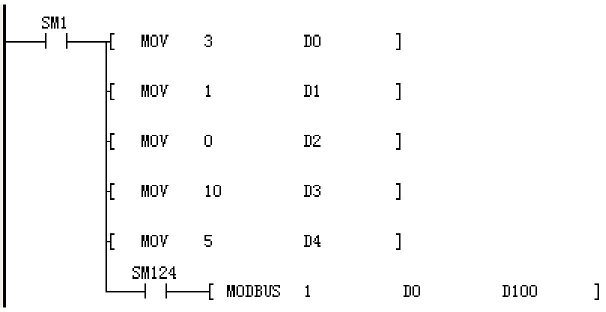
2. You do not need to set the length for the data to be sent. The system will set the length automatically based on the instruction.

S2	Slave address
S2+1	Function code
S2+2	Data 1
	...
S2+N+1	Data N

3. The data, when received through Modbus, will be stored in RTU-mode, regardless of whether you set it in RTU mode or ASCII mode. That is, when you set the data to ASCII mode, the system will automatically convert them to hexadecimal, remove the starting character and ending character, and store them to the data area starting with ***S3***.

4. The sent and received data are stored in the low bytes of the word element. High bytes are not used.

Example



LD SM1

MOV 3 D0

MOV 1 D1

MOV 0 D2

MOV 10 D3

MOV 5 D4

AND SM124

Modbus 1 D0 D100

1. Store the data sent through Modbus into the element starting with D0.

2. Store the data received in the elements starting with D100.

3. After receiving data through Modbus, the system will conduct CRC check, address check and instruction check. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

4. SM114 and SM124 are the flags of free serial port, and they can be also flags of MODBUS communication status.

The communication error codes are shown below:

Code	Description
0x01	Illegal instruction
0x02	Illegal register address
0x03	Wrong number of data
0x10	Communication timeout. The communication exceeds the preset communication time limit
0x11	Error in receiving data frame
0x12	Operand error. Operand (mode or master/slave) setting error
0x13	Error occurs because t he local station SN is the same as that set by the instruction

For the detailed application methods, see **Error! Reference source not found.Error! Reference source not found.**.

6.12.2 IVFWD: Inverter forward rotation instruction

LAD: EVFWD (S1) (S2)	**Applicable to**	**EC20 EC10 EC20H EC10V**	
	Influenced flag bit		
IL: IVFWD (*S1*) (*S2*)	**Program steps**	**6**	

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√

- **Operand description**

S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

- **Function description**

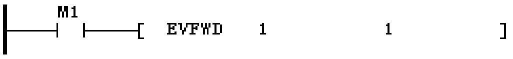
1. Control the inverter forward running through communication in the Modbus protocol.

2. This instruction is executed upon the rising edge.

- **Note**

The total number of the instructions for the communication between Modbus and inverter does not exceed 128.

- **Example**



LD M1

IVFWD 1 1

1. Set serial port 1, inverter address #1, and control the inverter forward running through communication in the Modbus protocol.

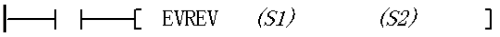
2. After the inverter receives the data, it will conduct CRC check, address check and instruction check and set the communication completion flag (SM135) after the

communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

The error codes in inverter instruction communication are listed below:

Error code	Description
0x1	Illegal instruction
0x2	Illegal register address
0x3	Data error. The data exceed the range
0x4	Slave operation failure, including the error caused by invalid data within the data range
0x5	Instruction valid, processing. It is used to store data to EEPROM.
0x6	Slave busy, please try again later. It is used to store data to EEPROM.
0x18	Information frame error, including the information length error and check error
0x20	The parameter cannot be modified
0x21	The parameter cannot be modified in the RUN state (only EV3100 supports this function)
0x22	The parameter is protected by password

6.12.3 IVREV: Inverter reverse rotation instruction

LAD:		
	**Applicable to**	**EC20 EC10 EC20H EC10V**
	Influenced flag bit	
IL: IVREV *(S1) (S2)*	**Program steps**	**6**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√

Operand description

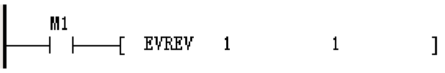
S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

Function description

1. Control the inverter reverse running through communication in the Modbus protocol.
2. This instruction is executed upon the rising edge.

Example

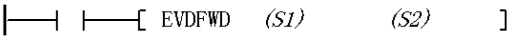


LD M1
IVREV 1 1

1. Set the serial port 1, inverter address #1, and control the inverter reverse running through communication in the Modbus protocol.

2. After the inverter receives the data, it will conduct CRC check, address check and instruction check, and set the communication completion flag (SM135) after the communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

6.12.4 IVDFWD: Inverter jogging forward rotation instruction

LAD:		
	**Applicable to**	**EC10 EC20 EC20H EC10V**
	Influenced flag bit	
IL: IVDFWD (*S1*) *(S2)*	**Program steps**	**6**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√

Operand description

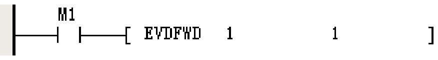
S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

Function description

1. Control the inverter jogging forward running through communication in the Modbus protocol.
2. This instruction is executed upon the rising edge.

Example



LD M1

IVDFWD 1
1

1. Set the serial port 1 and inverter address #1, and control the inverter jogging forward running through communication in the Modbus protocol.

2. After the inverter receives the data, it will conduct CRC check, address check and instruction check, and set the communication completion flag (SM135) after the communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

6.12.5 IVDREV: Inverter jogging reverse rotation instruction

LAD: [EVDREV *(S1)* *(S2)*]	**Applicable to**	**EC10 EC20 EC20H EC10V**
	Influenced flag bit	
IL: IVDREV (*S1*) *(S2)*	**Program steps**	**6**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√

Operand description

S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

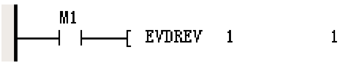
S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

Function description

1. Control the inverter jogging reverse running through communication in the Modbus protocol.

2. This instruction is executed upon the rising edge.

Example



LD M1
IVDREV 1 1

1. Set the serial port 1 and inverter address #1, and control the inverter jogging reverse running through communication in the Modbus protocol.

2. After the inverter receives the data, it will conduct CRC check, address check and instruction check, and set the communication completion flag (SM135) after the communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

6.12.6 IVSTOP: Inverter stop instruction

LAD: [EVSTOP *(S1)* *(S2)* *(S3)*]	**Applicable to**	**EC10 EC20 EC20H EC10V**
	Influenced flag bit	
IL: IVSTOP (*S1*) *(S2)* *(S3)*	**Program steps**	**8**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√
S3	WORD	Constant	D	V										R	√

Operand description

S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

S3: inverter stop mode. There are three stop modes: stop mode 0 (stop), stop mode 1 (free stop), stop mode 2 (JOG stop).

Function description

1. Control the inverter stop through communication in the Modbus protocol.

2. This instruction is executed upon the rising edge.

Example

M1
EVSTOP 1 1 0

LD M1

IVSTOP 1 1 0

1. Set the serial port 1, inverter address #1, and the inverter stop mode 0 (stop according to the set deceleration time), and control the inverter stop through communication in the Modbus protocol.

2. After the inverter receives the data, it will conduct CRC check, address check and instruction check, and set the communication completion flag (SM135) after the communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

6.12.7 IVFRQ: Inverter set frequency instruction

LAD: EVFRQ (S1) (S2) (S3)	**Applicable to**	**EC10 EC20 EC20H EC10V**
	Influenced flag bit	
IL: IVFRQ (*S1*) *(S2) (S3)*	**Program steps**	**8**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√
S3	WORD	Constant	D	V										R	√

Operand description

S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

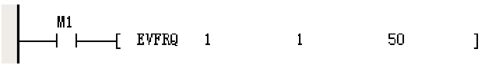
S3: frequency of the inverter

Function description

1. Control the running frequency of the inverter through communication in the Modbus protocol.

2. This instruction is executed upon the rising edge.

Example



LD M1

IVFRQ 1 1 50

1. Set the serial port 1, inverter address #1, and the running frequency of the inverter 50Hz, and control the running frequency of the inverter through communication in the Modbus protocol.

2. After the inverter receives the data, it will conduct CRC check, address check and instruction check, and set the communication completion flag (SM135) after the communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

6.12.8 IVWRT: Inverter write single register value instruction

LAD: EVWRT (S1) (S2) (S3) (S4)	**Applicable to**	**EC10 EC20 EC20H EC10V**
	Influenced flag bit	
IL: IVWRT (*S1*) *(S2) (S3) (S4)*	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√
S3	WORD	Constant	D	V										R	√
S4	WORD	Constant	D	V										R	√

Operand description

S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

S3: register address

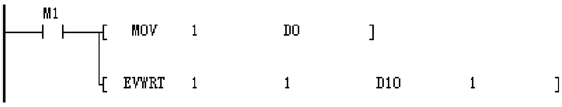
S4: register value

Function description

1. Write the single register value through communication in the Modbus protocol.

2. This instruction is executed upon the rising edge.

Example



LD M1

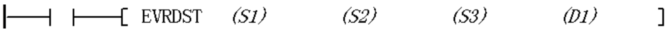
MOV 1 D0

IVWRT 1 1 D10 1

1. Set the serial port 1 and inverter address #1, input the register address 1 (digital frequency control) and register value 1 (disable frequency saving upon power-off), and write the value into the corresponding register through communication in the Modbus mode.

2. After the inverter receives the data, it will conduct CRC check, address check and instruction check, and set the communication completion flag (SM135) after the communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

6.12.9 IVRDST: Inverter read status instruction

LAD: 	**Applicable to**	**EC10 EC20 EC20H EC10V**
	Influenced flag bit	
IL: IVRDST *(S1) (S2) (S3) (D1)*	**Program steps**	**10**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√
S3	WORD	Constant	D	V										R	√
D1	WORD	D													√

Operand description

S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

S3: status information selection

0: Running status word 1: Actual operation value in the current main setting 2: Inverter model 3: Output current 4: Output voltage 5: Running speed 6: Operation fault information

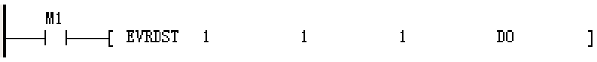
D1: storage address of the returned status information

Function description

1. Read the inverter status through communication in the Modbus protocol.

2. This instruction is executed upon the rising edge.

Example



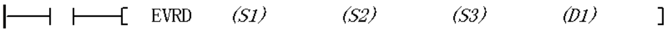
LD M1

IVRDST 1 1 1 D0

1. Set the serial port 1, inverter address#1, read status information selection 1 (actual running value in the current main setting) and set D0 as the storage register for the returned status information. Read the inverter status through communication in the Modbus protocol.

2. After the inverter receives the data, it will conduct CRC check, address check and instruction check, and set the communication completion flag (SM135) after the communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

6.12.10 IVRD: Inverter read single register value instruction

LAD: 	**Applicable to**	**EC10 EC20 EC20H EC10V**
	Influenced flag bit	
IL: IVRD *(S1) (S2) (S3) (D1)*	**Program steps**	**10**

Operand	Type	Applicable elements	Indexed addressing

S1	INT	Constant													
S2	WORD	Constant	D	V										R	√
S3	WORD	Constant	D	V										R	√
D1	WORD	D													√

Operand description

S1: designated communication channel (EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: inverter address. Broadcast mode. Broadcast address: 00. Slave address range: 1~247.

S3: address of the register to read

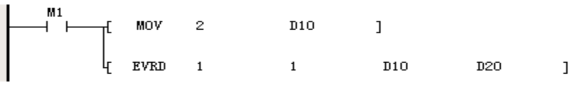
D1: storage address of the returned value

Function description

1. Read the single inverter register value through communication in the Modbus protocol.

2. This instruction is executed upon the rising edge.

Example



```
LD    M1
MOV   2   D10
IVRD 1 1 D10 D20
```

1. Set the serial port 1, inverter address#1, read register address 2 (initially set frequency of the inverter) and set D20 as the storage register for the returned value. Read a single inverter register through communication in the Modbus protocol.

2. After the inverter receives the data, it will conduct CRC check, address check and instruction check, and set the communication completion flag (SM135) after the communication. If there is any error, the error flag (SM136) will be set, and the error details will be recorded in the special register SD139.

6.12.11 XMT: Free port sending instruction

LAD: ├──┤ ├──[XMT *(S1)* *(S2)* *(S3)*]	Applicable to	EC20 EC10 EC10A EC20H EC10V	
	Influenced flag bit		
IL: XMT *(S1) (S2) (S3)*	Program steps	7	

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant														
S2	WORD	D	V												R	
S3	INT	Constant	KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	R	

Operand description

S1: designated communication channel. EC10, EC20: 0, 1; EC10V, EC20H: 0, 1, 2.

S2: starting address of the data to be sent

S3: number of bytes to be sent

Function description

When the power flow is valid, and the communication conditions are met, the designated data will be sent through the designated channel.

Note

1. Size of communication frame: depending on the element type (D or V) of the communication frame, the ending character of the frame does not exceed D7999 or V63.

2. In case of shutdown, the sending will stop.

Special register

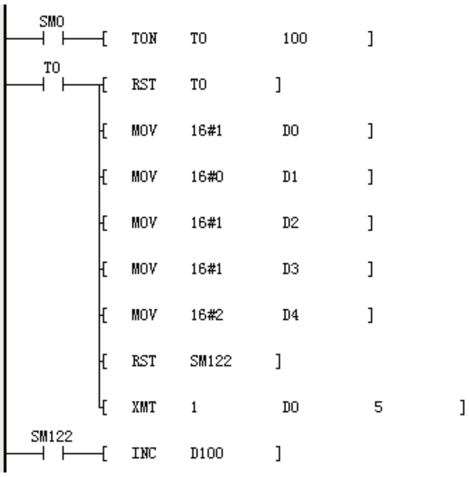
1. SM110/SM120: Sending enabled flag. It will be set when the XMT instruction is used and cleared when the sending is completed. When it is reset, the current sending stops.

2. SM112/SM122: Sending completed flag. When it is judged that the sending is completed, the sending completed flag will be set.

3. SM114/SM124: Idle flag. When the serial port has no communication task, it will be set, and it can be used as the checking bit for communication.

4. For detailed examples of the application, please refer to *Chapter 10* **Error! Reference source not found.**.

Example



LD SM0

TON T0 100

LD T0

RST T0

MOV 16# 1 D0

MOV 16#0 D1

MOV 16#1 D2

MOV 16#1 D3

MOV 16#2 D4

RST SM122

XMT 1 D0 5

LD SM122

INC D1OO

In this example, one data frame is sent in every 10s.

The following data will be sent through serial port 1.

01	00	01	01	02

1. Set port 1 in the system block as free port, and then set the baud rate, parity check, data bit and stop bit.

2. Write the data to be sent into the transmission buffer area. For EC20, only the low bytes of the word element will be sent.

3. Reset the sending completed flag (SM122) before sending the data.

4. When the sending is completed, set the sending completed flag (SM122).

6.12.12 RCV: Free port receiving instruction

LAD: RCV *(S1)* *(D)* *(S2)*	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: RCV *(S1)* *(D)* *(S2)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
D	WORD	D	V											R	
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	R	

Operand description

S1: designated communication channel. EC10, EC20: 0, 1; EC10V, EC20H: 0, 1, 2.

D: starting address for storing the received data

S2: maximum number of received bytes

Function description

When the power flow is valid, and the communication conditions are met, limited amount of data will be received through the designated channel to the designated registers.

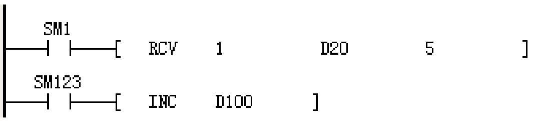
Note

1. Size of communication frame: depending on the element type (D or V) of the communication frame, the ending character of the frame does not exceed D7999 or V63.

2. The receiving stops upon shutdown.

3. The value range of S1: 0 and 1

Example



LD SM1

RCV 1 D20 5

LD SM123

INC D100

1. The instruction will be valid continuously as long as the power flow is valid. If you want to receive data only once, you can use a rising edge or special registers that are effective only once, such as SM1, to trigger the instruction.

2. For detailed application examples, refer to **Error! Reference source not found.Error! Reference source not found.**.

Special register

SM111 (SM121): Receiving enabled flag. It will be set when the RCV instruction is used and cleared when the sending is completed. When it is reset, the current receiving stops.

SM113 (SM123): Receiving completed flag. When the receiving is completed, the receiving completed flag will be set.

SM114 (SM124): Idle flag. When the serial port has no communication task, it will be set, and it can be used as the checking bit for communication.

SD111 (SD121): Starting character, which can be set in the system block

SD112 (SD122): Ending character, which can be set in the system block

SD113 (SD123): Character timeout time, i.e. the maximum receiving interval between the two characters, which can be set in the system block

SD114 (SD124): Frame timeout time, the time starting with the power flow and stops at the end of the receiving, which can be set in the system block

SD115 (SD125): receiving completion code. The definition of the data bit is shown as follows:

User end receiving flag	Designated ending word received flag	Max. number of characters received flag	Inter-character timeout flag	(Frame) reception timeout flag	Parity check error flag	Reserved
Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bits 6~15

SD116 (SD126): The characters currently received

SD117 (SD127): The character received previously

6.12.13 MODRW: MODBUS read/write instruction

LAD:	Applicable to	EC20 EC10 EC20H EC10V
├─┤ ├──[MODRW (S1) (S2) (S3) (S4) (S5) (D) (S6)]	Influenced flag bit	
IL: MODRW *(S1)(S2)(S3)(S4)(S5)(D)*	Program steps	14

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant													
S2	WORD	Constant	D	V										R	√
S3	WORD	Constant	D	V										R	√
S4	WORD	Constant	D	V										R	√
S5	WORD	Constant	D	V										R	√
D	WORD		D											R	√

Operand description

S1: designated communication channel(EC10,EC20: channel 1; EC10V,EC20H: channen1 and channel 2)

S2: address (slave address range: 1~247, broadcast address is applicable to write elements)

S3: function code. EC10 supports 01 (read coil), 02 (read discrete input), 03 (read register), 04 (read input register), 05 (write single coil), 06 (write single register), 15 (write multiple coils), 16 (write multiple registers)

S4: starting address of read/write elements

S5: number of read/write elements. The number of read/write elements for EC10 is limited by Max. RTU frame length (256), as shown below:

Code	Name	Number of elements	Number of D elements
01	Read coil	1~2000	(S5+15)/16
02	Read discrete input	1~2000	(S5+15)/16
03	Read register	1~125	S5
04	Read input register	1~125	S5
05	Write single coil	0 (fixed)	1
06	Write single register	0 (fixed)	1
15	Write multiple coils	1~1968	(S5+15)/16
16	Write multiple registers	1~123	S5

*The number of 05 and 06 must be 0 in ***S5***.

The number of read/write elements for EC20 and EC20H (***S5***≤16), the maximum number of word elements and bit elements is 16 and all bit elements are stored into a word.

D1: storage address of read/write elements. For the number of elements needed by EC10, refer to above table.

Function description

When the power flow is valid, send messages and receive the returned data.

Note

For EC20 and EC20H

1. The number of elements is 16 at most.
2. The bit elements read 16 at most, the small address is stored at low bit, and one byte stores 16 bits.
3. The returned abnormal code is Modbus instruction.

For EC10, V1.23 or higher version supports the instruction and needs to match with Programmer of V2.39 or higher version.

Example

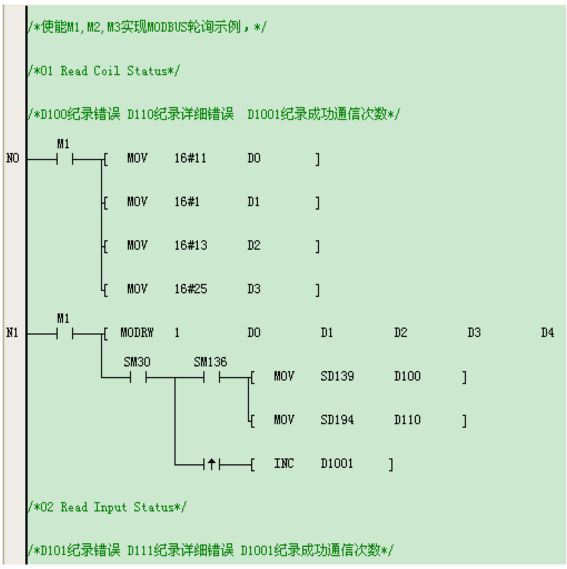
*The following example is only valid for EC10 series PLC.

1. Standard polling

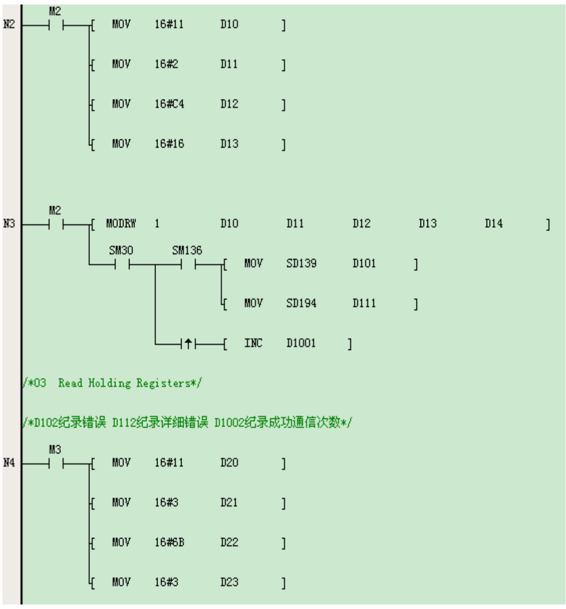
The example is the simple polling, M1, M2 and M3 are set, and three MODRW instructions access the device in turn.

In operation, reset any M element, the corresponding MODRW instruction will exit polling but other MODRW instructions will still execute by polling. For example, reset M2, MODRW instructions of M1 and M3 access the device.

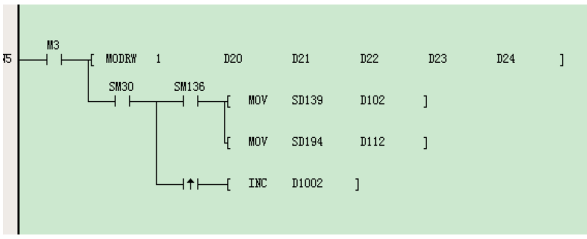
Similarly, one MODRW instruction can be inserted in operation. For example, set M2 and three MODRW instructions access the device in turn.



```
/*使能M1,M2,M3实现MODBUS轮询示例，*/
/*01 Read Coil Status*/
/*D100纪录错误 D110纪录详细错误  D1001纪录成功通信次数*/
N0   M1    [ MOV   16#11   D0    ]
           [ MOV   16#1    D1    ]
           [ MOV   16#13   D2    ]
           [ MOV   16#25   D3    ]
N1   M1    [ MODRW  1   D0   D1   D2   D3   D4 ]
        SM30   SM136   [ MOV   SD139   D100   ]
                       [ MOV   SD194   D110   ]
                ↑      [ INC   D1001   ]
/*02 Read Input Status*/
/*D101纪录错误 D111纪录详细错误 D1001纪录成功通信次数*/
```



```
N2   M2    [ MOV   16#11   D10   ]
           [ MOV   16#2    D11   ]
           [ MOV   16#C4   D12   ]
           [ MOV   16#16   D13   ]
N3   M2    [ MODRW  1   D10   D11   D12   D13   D14 ]
        SM30   SM136   [ MOV   SD139   D101   ]
                       [ MOV   SD194   D111   ]
                ↑      [ INC   D1001   ]
/*03  Read Holding Registers*/
/*D102纪录错误 D112纪录详细错误 D1002纪录成功通信次数*/
N4   M3    [ MOV   16#11   D20   ]
           [ MOV   16#3    D21   ]
           [ MOV   16#6B   D22   ]
           [ MOV   16#3    D23   ]
```



```
N5   M3    [ MODRW  1   D20   D21   D22   D23   D24 ]
        SM30   SM136   [ MOV   SD139   D102   ]
                       [ MOV   SD194   D112   ]
                ↑      [ INC   D1002   ]
```

In above program, SM30 shows whether the MODRW instruction is executed. After the MODRW instruction is executed, SM30 will be set. When the MODRW instruction enters the next execution, SM30 will be reset. The sequence charts in Example 2 illustrate the differences between SM30 and SM135.

When multiple MODRW instructions appear in the program, SM30 can be used to show the executive conditions of the instructions. The usage of multiple SM30 elements will not affect each other.

When the MODRW instruction has an error, SM136 will be set, SD139 and SD194 will indicate the error code. The values of SM136, SD139 and SD194 can be changed by other MODRW instructions, so record the conditions before executing the next MODRW instruction.

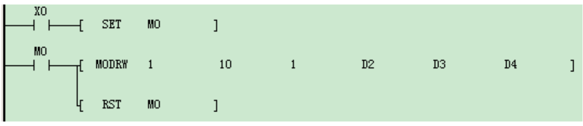
The error codes of MODRW instructions are shown below:

Code	Error name	Description
1	Illegal instruction	
2	Illegal address	
3	Illegal data	
4~15	Reserved	
16	Communication timeout	The communication exceeds the preset communication time limit
17	Reserved	

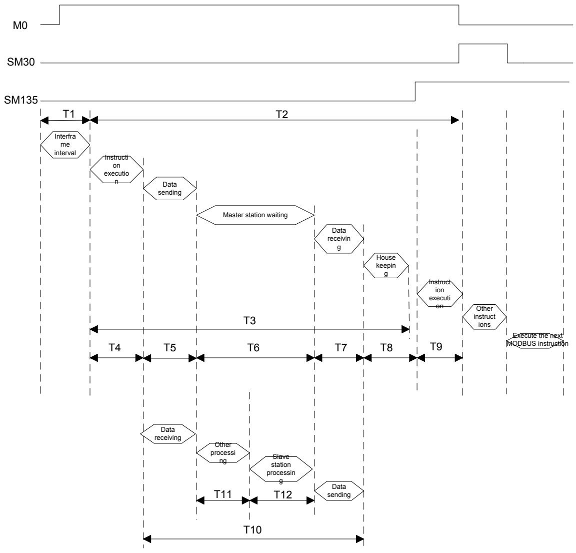
18	Set parameters error	Set parameters (mode or master/slave) error
19	S2 error, namely, slave address error	The station number itself is the same with the set station number, or the address is out of the range
20	D error, namely, element address overflow	The element address overflows (the received or sent data amount exceeds the memory space of the element)
21	Instruction execution failure	
22	Address does not match	The received slave address does not match with the requested slave address
23	Instruction does not match	The received instruction does not match with the requested instruction
24	Information frame error	The starting element address does not match
25	Data length does not match	The received data length does not comply with the protocol or the number of elements exceeds the maximum limit
26	CRC/LRC check error	
27	Reserved	
28	S3 error, namely, element address error	Error of starting element address setting
29	S4 error, namely, instruction error	Unavailable or illegal instruction setting
30	S5 error, namely, element number error	Error of element number setting
31	Reserved	
32	Parameters cannot be changed	Parameters cannot be changed
33	Parameters cannot be changed in running	Parameters cannot be changed in running (only available for EV3000)
34	Parameters under password protection	Parameters under password protection

2. Link time

The following LAD propram can realize communication between MODBUS master station and slave station. The time at each stage for a complete communication is shown below:



```
    X0
├──┤ ├──[  SET    M0       ]

    M0
├──┤ ├──[  MODRW  1        10       1        D2       D3       D4       ]
        │
        └[  RST    M0       ]
```



A complete MODBUS communication time (T_m) consists of T_1 and T_2, that is:

$$T_m = T_1 + T_2$$

According to MODBUS communitiaon protocol, the interval time among frames should be the time of 3.5 bytes at least.

Length of a character: starting bit (1 bit)+data length (7 bits)+check bit (0 bit or 1 bit)+stop bit (1 bit or 2 bits)

$$T_2 = (INT(\frac{T_3}{T_s}) + 1))T_s$$

T_s is the maximum scan cycle for PLC

$$T_3 = T_4 + T_5 + T_6 + T_7 + T_8 + T_9$$

The time for T_4, T_8 and T_9 is less than 1ms

$$T_5 = \frac{Bytes\ to\ be\ sent \times character\ length}{Baud\ rate\ (bps)} \times 1000(ms) + 1ms$$

T_6: the waiting time of master station is determined by salve station and the maximum value cannot exceed timeout time of the set main mode

$$T_7 = \frac{Bytes\ to\ be\ received \times characterlength}{Baud\ rate\,(bps)} \times 1000\,(ms) + 1ms$$

The processing time of slave station can be calculated according to the following formula:

$$T_{10} = T_5 + T_{11} + T_{12} + T_7$$

T_{11} is the maximum scan cycle

The time for T_{12} is less than 1ms

For example: The set communication specification is 19200, even check, 8 data bits, 1 stop bit, RTU transmission mode, send 10 characters and receive 20 characters. The processing time of master station is calculated as follows:

$$T_5 = \frac{10 \times 10}{19200} \times 1000 + 1 = 6.2ms$$

$$T_7 = \frac{20 \times 10}{19200} \times 1000 + 1 = 11.4ms$$

$$T_4 = T_8 = T_9 \approx 1ms$$

Suppose $T_6 = 35ms$, thus

$$T_3 = 1 + 6.2 + 35 + 11.4 + 1 + 1 = 55.6ms$$

Suppose the maximum scan cycle is 10ms, thus

$$T_2 = (INT(\frac{55.6}{15}) + 1))15 = 60ms$$

The processing time of slave station is:

$$T_{10} = 6.2 + 15 + 1 + 11.4 = 33.6ms$$

6.13 Data check instruction

6.13.1 CCITT: Check instruction

LAD: [CCITT (S1) (S2) (D)]	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: CCITT *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD								D				V		R	√

Operand description

S1: the starting element of the data to be checked

S2: the number of the data to be checked. ***S2***≥0, or the system will report operand error.

D: check result

Function description

1. Conduct CCITT check on the S2 data starting with S1, and assign the result to D.

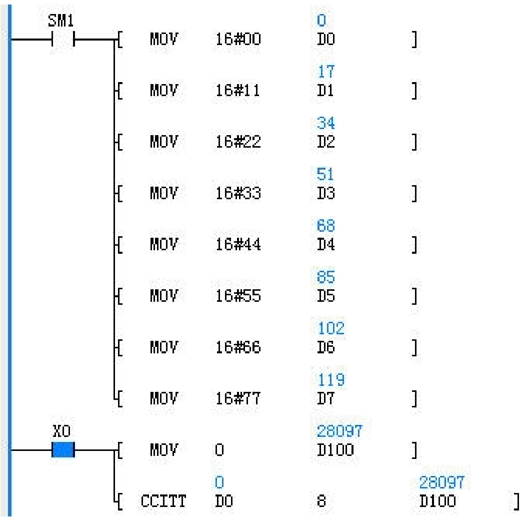
2. The expression for CCITT check algorithm is: X^16+X^12+X^5+1

Note

1. For the system will bring value of D into the operation each time the instruction is executed, make sure to clear D before executing the CCITT instruction.

2. The data within the checking data zone starting with S2 are stored in byte mode by default. That is, the high bytes are taken as 0, and the check result has 16 bits.

Example



```
LD      SM1
MOV     16#00 D0
MOV     16#11 D1
MOV     16#22 D2
MOV     16#33 D3
MOV     16#44 D4
MOV     16#55 D5
MOV     16#66 D6
MOV     16#77 D7
LD      X0
MOV   0   D100
CCITT    D0    8 D100
```

When X0 is ON, conduct CCITT check on the 8 data starting with D0, and the result is assigned to D100.

6.13.2 CRC16: Check instruction

LAD: [CRC16 (S1) (S2) (D)]	Applicable to	EC20 EC10 EC10A EC20H
	Influenced flag bit	
IL: CRC16 (*S1*) (*S2*) (*D*)	Program steps	7

Operand	Type	Applicable elements	Indexed addressing

S1	WORD								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD								D				V		R	√

Operand description

S1: the starting element of the data to be checked

S2: the number of the data to be checked; ***S2***≥0, or the system will report operand error

D: check result

Function description

1. Conduct CRC16 check on the S2 data starting with S1, and assign the result to D unit.

2. The expression for CRC16 check algorithm is: X^16+X^15+X^2+1

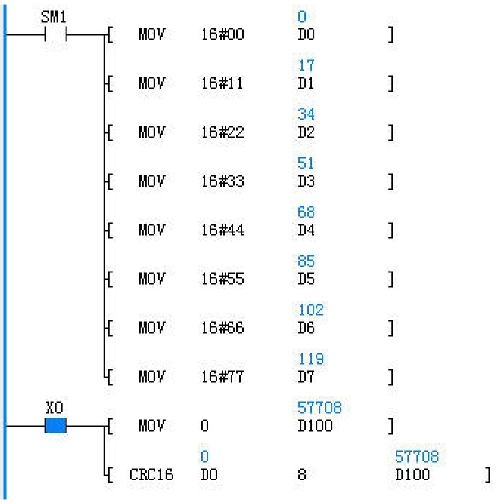
Note

1. For the system will bring value of D into the operation each time the instruction is executed, make sure to clear D before executing the CRC16 instruction.

2. The standard Modbus CRC check requires that the D element (checksum) be initialized as 16#FFFF, and the high/low byes (8 high, 8 low) shall be swapped.

3. The data within the checking data zone starting with S2 are stored in byte mode by default. That is, the high bits will be taken as 0, and the check result has 16 bits.

Example



```
LD     SM1
MOV    16#00 D0
MOV    16#11 D1
MOV    16#22 D2
MOV    16#33 D3
MOV    16#44 D4
MOV    16#55 D5
MOV    16#66 D6
MOV    16#77 D7
LD     X0
MOV  0   D100
CRC16 D0 8 D100
```

When X0 is ON, conduct CRC16 check on the 8 data starting with D0, and the result is assigned to D100.

6.13.3 LRC: Check instruction

LAD: ├──┤ ├──[LRC (S1) (S2) (D)]	Applicable to	EC20 EC10 EC10A EC20H EC10V
	Influenced flag bit	
IL: LRC (S1) (S2) (D)	Program steps	7

Operand	Type	Applicable elements														Indexed addressing
S1	WORD								D				V		R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	WORD								D				V		R	√

Operand description

S1: the starting element of the data to be checked

S2: the number of the data to be checked. ***S2***≥0, or the system will report operand error

D: check result

Function description

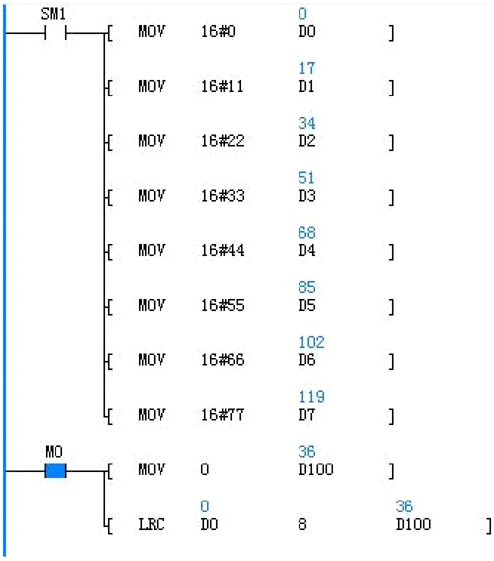
Conduct LRC check on the S2 data starting with the S1, and assign the result to D.

Note

1. For the system will bring value of D into the operation each time the instruction is executed, make sure to clear D before executing the LRC instruction.

2. The data within the checking data zone starting with S2 are stored in byte mode by default. That is, the high bytes are taken as 0, and the check result has 8 bits and is stored in the low bits of D.

Example



```
LD      SM1
MOV         16#00
D0
MOV         16#11
D1
MOV         16#22
D2
MOV         16#33
D3
MOV         16#44
D4
MOV         16#55
D5
MOV         16#66
D6
MOV         16#77
D7
LD      M0
MOV             0
D100
LRC    D0    8 D100
```

When X0 is ON, conduct LRC check on the 8 data starting with D0, and the result is assigned to D100.

6.14 Enhanced bit processing instruction

6.14.1 ZRST: Batch bit reset instruction

<table>
<tr><td colspan="9" rowspan="2">LAD:
ZRST (D) (S)]</td><td colspan="3">Applicable to</td><td colspan="5">EC20 EC10 EC10A EC20H EC10V</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="5"></td></tr>
<tr><td colspan="9">IL: ZRST (<i>D</i>) (<i>S</i>)</td><td colspan="3">Program steps</td><td colspan="5">5</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>D</td><td>BOOL</td><td></td><td></td><td>Y</td><td>M</td><td>S</td><td>LM</td><td></td><td></td><td></td><td>C</td><td>T</td><td></td><td></td><td></td><td>√</td></tr>
<tr><td>S</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

Operand description

D: destination operand

S: source operand

Function description

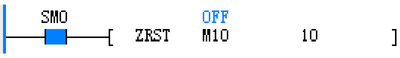
When the power flow is valid, reset S bit-elements starting with D.

Note

1. When a C element is reset, the counting value in it will also be cleared.

2. When a T element is reset, the timing value in it will also be cleared.

Example



```
LD      SM0
ZRST          M10
```

10 When SM0 is ON, the 10 units M10, M11, M12 ... M19 will be completely cleared.

6.14.2 ZSET: Set batch bit instruction

LAD: ┤├ ├─[ZSET *(D)* *(S)*]	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: ZSET (*D*) (*S*)	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
D	BOOL			Y	M	S	LM				C	T				√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

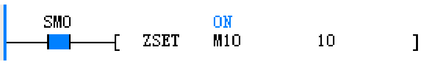
D: destination operand

S: source operand

Function description

When the power flow is valid, set S bit elements starting with D.

Example



```
LD     SM0
ZSET   M10   10
```

When SM0 is ON, the 10 units M10, M11, M12 … M19 will all be set to 1.

6.14.3 DECO: Decode instruction

LAD: ┤├ ├─[DECO *(S)* *(D)*]	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: DECO (*S*) (*D*)	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S	WORD	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

S: source operand

D: destination operand

Function description

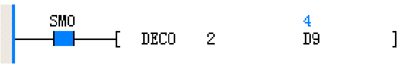
When the power flow is valid, set bit S in word element D to 1, and clear other bits.

Note

1. Range of S: 0 to 15.

2. If S is outside the range of 0~15, D will not be changed when the power flow is valid. Instead, the system will report operand error.

Example



```
LD     SM0
DECO   2   D9
```

When the power flow is valid, bit 2 in D9 will be set as 1, other bits will be cleared.

6.14.4 ENCO: Encode instruction

LAD: ┤├ ├─[ENCO *(S)* *(D)*]	Applicable to	EC20 EC10 EC10A EC20H EC10V
	Influenced flag bit	
IL: ENCO *(S)* *(D)*	Program steps	5

Operand	Type	Applicable elements	Indexed addressing

S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	Z	R	√

Operand description

S: source operand;

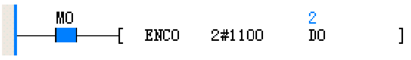
D: destination operand

Function description

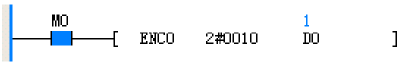
When the power flow is valid, assign the number of the bit whose value is 1 in word element S to D.

Note

When the value of multiple bits in S is 1, the smallest bit number will be written into D, as shown in the following figure:



Example



LD M0

ENCO 2#0010 D0

When the power flow is valid, operand 1 is 2#0010, bit 1 is 1, hence 1 is written into D0.

6.14.5 BITS: Counting ON bit in word instruction

<table>
<tr><td colspan="10" rowspan="2">LAD:
├─┤ ├─[BITS (S) (D)]</td><td colspan="3">Applicable to</td><td colspan="4">EC20 EC10 EC10A EC20H EC10V</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="4"></td></tr>
<tr><td colspan="10">IL: BITS (S) (D)</td><td colspan="3">Program steps</td><td colspan="4">5</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

Operand description

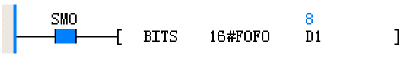
S: source operand

D: destination operand

Function description

When the power flow is valid, count how many bits in operand S is 1, and store the result into D.

Example



LD SM0

BITS 16#F0F0 D1

When the power flow is valid, it is counted that there are 8 bits whose value is 1 (ON status) in constant 16#F0F0, so 8 is stored into D1.

6.14.6 DBITS: Counting ON bit in double word instruction

<table>
<tr><td colspan="10" rowspan="2">LAD:
├─┤ ├─[DBITS (S) (D)]</td><td colspan="3">Applicable to</td><td colspan="4">EC20 EC10 EC10A EC20H EC10V</td></tr>
<tr><td colspan="3">Influenced flag bit</td><td colspan="4"></td></tr>
<tr><td colspan="10">IL: DBITS (S) (D)</td><td colspan="3">Program steps</td><td colspan="4">6</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S</td><td>DWORD</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>INT</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

Operand description

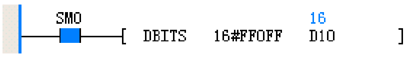
S: source operand

D: destination operand

Function description

When the power flow is valid, count how many bits in double word S is 1, and store the result into D.

Example



LD SM0

DBITS 16#FF0FF D10

When the power flow is valid, it is counted that there are 16 bits whose value is 1 (ON status) in constant 16#FF0FF, so 16 is stored into D10.

6.14.7 BON: Judging ON bit in word instruction

LAD: [BON (S1) (D) (S2)]	**Applicable to**	**EC20H**
	Influenced flag bit	
IL: BON*(S1) (D) (S2)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V			√
D	BOOL		Y	M	S											
S2	INT								D				V		R	

Operand description

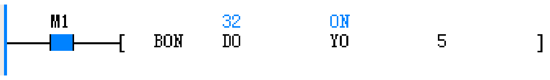
S: source operand

D: destination operand

Function description

When the power flow is valid, count the status of bit S2 in element S1, and store the result into D.

Example



LD M1

BON D0 Y0 5

When the power flow is valid, S1 in BON instruction is ConstantD0, the state of bit 5 is ON, and the result is stored into Y0.

6.15 Word contact instruction

6.15.1 BLD: Word bit contact LD instruction

LAD: BLD (S1) (S2) ( )	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: BLD *(S1) (S2)*	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

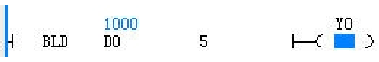
S1: source operand

S2: designated bit, 0≤***S2***≤15, or system will report operand error

Function description

Use the status of bit S2 in element S1 to drive the following operation.

Example



BLD D0

5

OUT Y0

Use the status of BIT5 (ON) in D0 (1000: 2#0000001111101000) to determine the status of Y0 in the following operation.

6.15.2 BLDI: Word bit contact LDI instruction

LAD: BLDI (S1) (S2)	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: BLDI *(S1)* *(S2)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

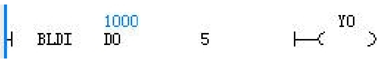
S1: source operand

S2: designated bit. 0≤***S2***≤15, or system will report operand error

Function description

Use the logic NOT of the status of bit S2 in element S1 to drive the following operation.

Example



```
BLDI   D0
5
OUT    Y0
```

Use the logic NOT of the status of BIT5 (ON) in D0 (1000: 2#0000001111101000), which is OFF, to determine the status of Y0 in the following operation.

6.15.3 BAND: Word bit contact AND instruction

LAD: BLD (S1) (S2) Note: because the logic relationship is visualized in the diagram, the BAND instruction is displayed in LAD as BLD	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: BAND *(S1)* *(S2)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

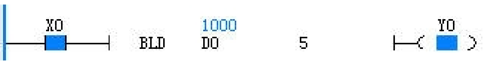
S1: source operand

S2: designated bit. 0≤***S2***≤15, or system will report operand error

Function description

Take the status of bit S2 in element S1 and use it in serial connection with other nodes to drive the operation of the following operation.

Example



```
LD     X0
BAND   D0 5
OUT    Y0
```

Take the status of BIT5 (ON) in element D0 (1000: 2#0000001111101000) and use it in serial connection with other nodes (X0: ON) to determine the status of Y0 in the following operation.

6.15.4 BANI: Word bit contact ANI instruction

LAD: BLDI (S1) (S2) Note: because the logic relationship is visualized in the diagram, the BANI instruction is displayed in LAD as BLDI	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: BANI *(S1)* *(S2)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

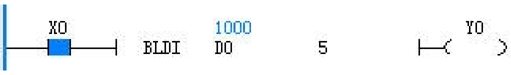
S1: source operand

S2: designated bit. 0≤***S2***≤15, or system will report operand error

Function description

Take the logic NOT of the status of bit S2 in element S1 and use it in serial connection with other nodes to drive the operation of the following instruction.

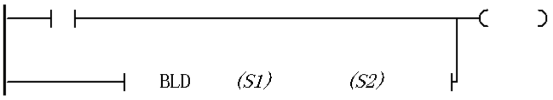
Example



```
LD     X0
BANI   D0   5
OUT    Y0
```

Take the logic NOT of the status of BIT5 (ON) in element D0 (1000: 2#0000001111101000), which is OFF, and use it in serial connection with other nodes (X0: ON) to determine the status of Y0 in the following operation.

6.15.5 BOR: Word bit contact OR instruction

LAD:  Note: because the logic relationship is visualized in the diagram, the BOR instruction is displayed in LAD as BLD	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: BOR *(S1)* *(S2)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

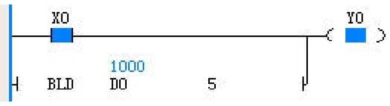
S1: source operand

S2: designated bit. 0≤S2≤15, or system will report operand error

Function description

Take the status of bit S2 in element S1 and use it in parallel connection with other nodes to drive the operation of the following instruction.

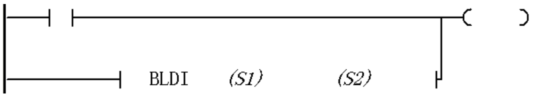
Example



```
LD     X0
BOR    D0   5
OUT    Y0
```

Take the status of BIT5 (ON) in element D0 (1000: 2#0000001111101000) and use it in parallel connection with other nodes (X0: ON) to determine the status of Y0 in the following operation.

6.15.6 BORI: Word bit contact ORI instruction

LAD:  Note: because the logic relationship is visualized in the diagram, the BORI instruction is displayed in LAD as BLDI	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: BORI *(S1)* *(S2)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S1	WORD		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

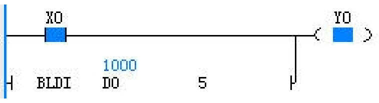
S1: source operand

S2: designated bit. 0≤S2≤15, or system will report operand error

Function description

Take the logic NOT of the status of bit S2 in element S1 and use it in parallel connection with other nodes to drive the operation of the following stage.

Example



LD X0
BORI D0 5
OUT Y0

Take the logic NOT of the status of BIT5 (ON) in element D0 (1000: 2#0000001111101000), which is OFF, and use it in parallel connection with other nodes (X0: ON) to determine the status of Y0 in the following operation.

6.15.7 BOUT: Word bit coil output instruction

<table>
<tr><td colspan="9" rowspan="2">**LAD:**
├──┤ ├──[BOUT (D) (S)]</td><td colspan="3">**Applicable to**</td><td colspan="5">**EC20 EC10 EC10A EC20H EC10V**</td></tr>
<tr><td colspan="3">**Influenced flag bit**</td><td colspan="5"></td></tr>
<tr><td colspan="9">**IL: BOUT (*D*) (*S*)**</td><td colspan="3">**Program steps**</td><td colspan="5">**5**</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>D</td><td>WORD</td><td></td><td></td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td></td><td>D</td><td></td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
<tr><td>S</td><td>INT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
</table>

Operand description

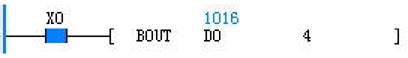
S1: source operand

S2: designated bit. 0≤S2≤15, or system will report operand error

Function description

Assign the current power flow status to bit S of element D.

Example



LD X0
BOUT D0 4

Assign the current power flow status (X0: ON) to BIT4 ofelement D0 (1000: 2#0000001111101000). After the execution, D0=1016 (2#0000001111111000).

6.15.8 BSET: Word bit coil set instruction

LAD: BSET (D) (S)	Applicable to	EC20 EC10 EC10A EC20H EC10V
	Influenced flag bit	
IL: BSET (*D*) (*S*)	Program steps	5

Operand	Type	Applicable elements														Indexed addressing
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

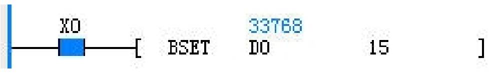
D: destination operand

S2: designated bit. 0≤S2≤15, or system will report operand error

Function description

Set bit S of element D.

Example



LD X0

BSET D0 15

When the power flow is valid, set BIT15 of element D0 (1000: 2#0000001111101000). After the execution, D0=33768 (2#1000001111101000).

6.15.9 BRST: Word bit coil reset instruction

LAD: BRST (D) (S)	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: BRST (*D*) (*S*)	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
D	WORD			KnY	KnM	KnS	KnLM		D		C	T	V	Z	R	√
S	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

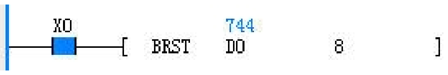
D: destination operand

S2: designated bit. 0≤S2≤15, or system will report operand error

Function description

Reset bit S of element D.

Example



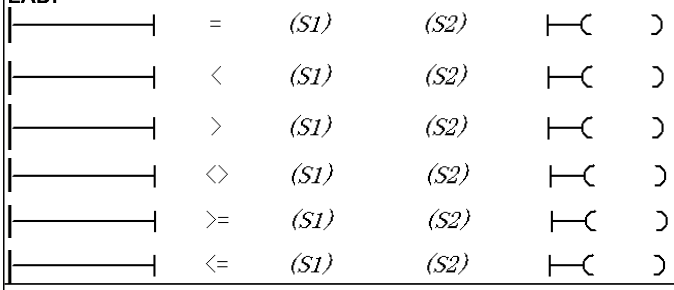
LD X0

BRST D0 8

When the power flow is valid, reset BIT8 of element D0 (1000: 2#0000001111101000). After the execution, D0=744 (2#0000001011101000).

6.16 Compare contact instruction

6.16.1 LD (=, <, >, <>, >=, <=): Compare integer LD※ instruction

LAD: 	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: LD= **(S1) (S2)** **LD< (S1) (S2)** **LD> (S1) (S2)** **LD<> (S1) (S2)** **LD>= (S1) (S2)** **LD<= (S1) (S2)**	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

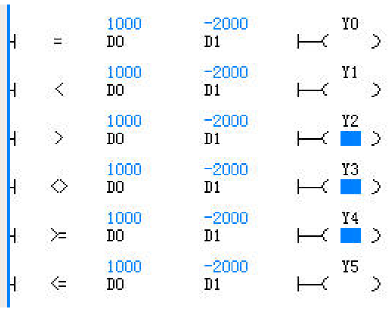
S1: comparison parameter 1

S2: comparison parameter 2

Function description

Conduct BIN comparison on elements S1 and S2, and use the comparison result to drive the following operation.

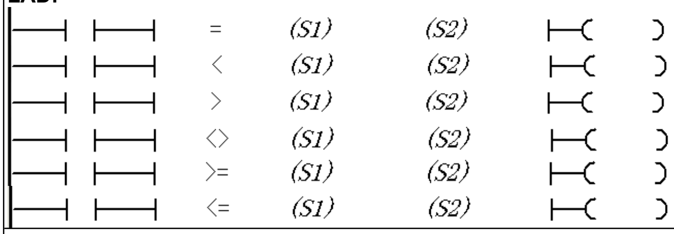
Example



```
LD=    D0 D1
OUT     Y0
LD<    D0 D1
OUT     Y1
LD>    D0 D1
OUT     2
LD<>   D0 D1
OUT     Y3
LD>=   D0 D1
OUT     Y4
LD<=   D0 D1
OUT     Y5
```

Conduct BIN comparison on the data of D0 and D1, and the comparison result is used to determine the output status of the following element.

6.16.2 AND(=, <, >, <>, >=, <=): Compare integer AND※ instruction

LAD: 	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: AND=　(S1)　(S2) **AND<　(S1)　(S2)** **AND>　(S1)　(S2)** **AND<>　(S1)　(S2)** **AND>=　(S1)　(S2)** **AND<=　(S1)　(S2)**	**Program steps**	**5**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

Operand description

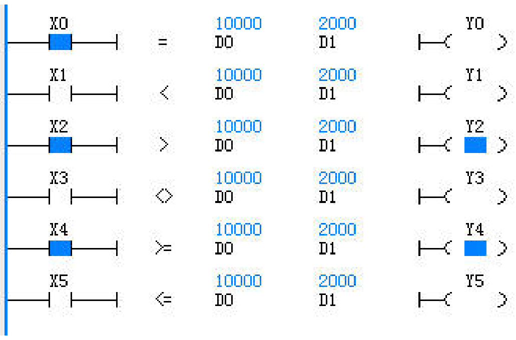
S1: comparison parameter 1

S2: comparison parameter 2

Function description

Conduct BIN comparison on elements S1 and S2, and use the comparison result to drive the following operation.

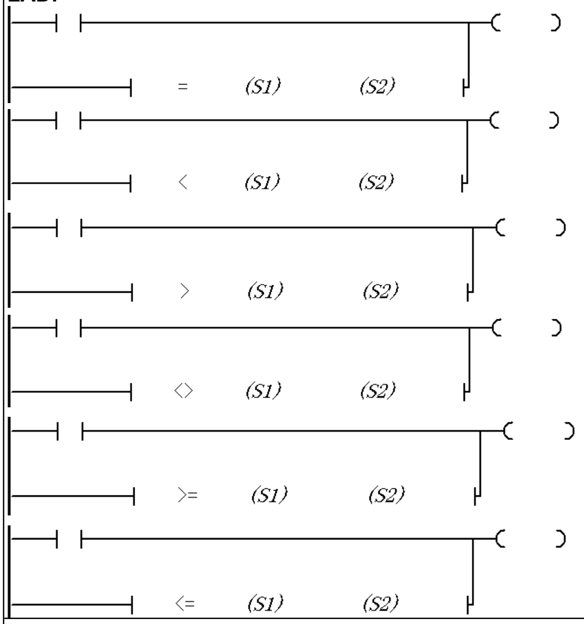
Example



```
LD      X0
AND=    D0 D1
OUT     Y0
LD      X1
AND<    D0 D1
OUT     Y1
LD      X2
AND>    D0 D1
OUT     Y2
LD      X3
AND<>   D0 D1
OUT     Y3
LD      X4
AND>=   D0 D1
OUT     Y4
LD      X5
AND<=   D0 D1
OUT     Y5
```

Conduct BIN comparison on the data of D0 and D1, and the comparison result is used to determine the output status of the following element.

6.16.3 OR(=, <, >, <>, >=, <=): Compare integer OR※ instruction

<table>
<tr><td rowspan="2"><b>LAD:</b>
</td><td><b>Applicable to</b></td><td><b>EC20 EC10 EC10A EC20H EC10V</b></td></tr>
<tr><td><b>Influenced flag bit</b></td><td></td></tr>
<tr><td><b>IL: OR＝ (S1) (S2)</b>
<b>OR＜ (S1) (S2)</b>
<b>OR＞ (S1) (S2)</b>
<b>OR＜＞ (S1) (S2)</b>
<b>OR＞＝ (S1) (S2)</b>
<b>OR＜＝ (S1) (S2)</b></td><td><b>Program steps</b></td><td><b>5</b></td></tr>
</table>

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√

- ■ **Operand description**

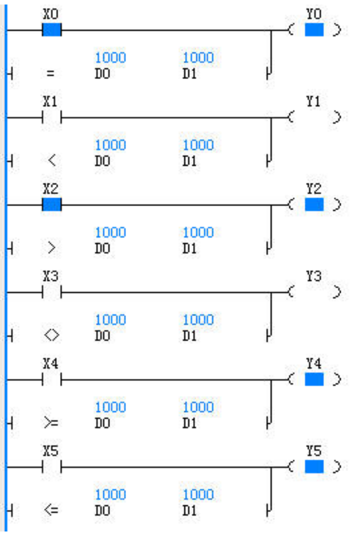
S1: comparison parameter 1

S2: comparison parameter 2

- ■ **Function description**

Compare elements S1 and S2, and use the comparison result in parallel connection with other nodes to drive the following operation.

- ■ **Example**



```
LD      X0
OR=     D0 D1
OUT     Y0
LD      X1
OR<     D0 D1
OUT     Y1
LD      X2
OR<>    D0 D1
OUT     Y2
LD      X3
OR>=    D0 D2
OUT     Y3
LD      X4
OR>=    D0 D1
OUT     Y4
LD      X5
OR<=    D0 D1
OUT     Y5
```

Compare elements D0 and D1, and use the comparison result in parallel connection with other nodes to determine the output status of the following element.

6.16.4 LDD(=, <, >, <>, >=, <=): Compare double integer LDD※ instruction

LAD: D= (S1) (S2) D< (S1) (S2) D> (S1) (S2) D<> (S1) (S2) D>= (S1) (S2) D<= (S1) (S2)	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: LDD＝ (S1) (S2) **LDD＜ (S1) (S2)** **LDD＞ (S1) (S2)** **LDD＜＞ (S1) (S2)** **LDD＞＝ (S1) (S2)** **LDD＜＝ (S1) (S2)**	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√

Operand description

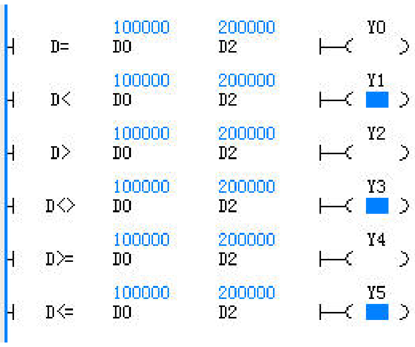
S1: comparison parameter 1

S2: comparison parameter 2

Function description

Compare elements S1 and S2, and use the comparison result to drive the following operation.

Example

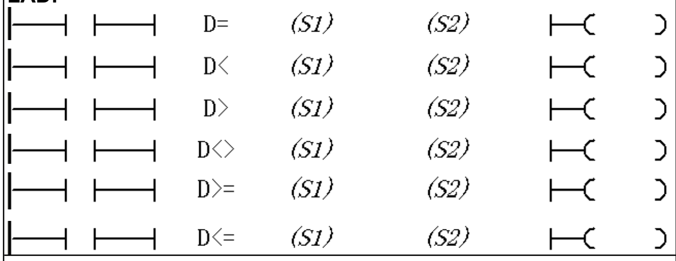


D= 100000 D0 200000 D2 Y0
D< 100000 D0 200000 D2 Y1
D> 100000 D0 200000 D2 Y2
D<> 100000 D0 200000 D2 Y3
D>= 100000 D0 200000 D2 Y4
D<= 100000 D0 200000 D2 Y5

```
LD= D0 D2
OUT Y0
LD<  D0 D2
OUT   Y1
LD<> D0 D2
OUT    Y2
LD>= D0 D2
OUT   Y3
LD>= D0 D2
OUT   Y4
LD<=D0 D2
OUT Y5
```

Compare (D0, D1) and (D2,D3), and use the comparison result to determine the output status of the following element.

6.16.5 ANDD(=, <, >, <>, >=, <=): Compare double integer ANDD※ instruction

LAD:  D= (S1) (S2) D< (S1) (S2) D> (S1) (S2) D<> (S1) (S2) D>= (S1) (S2) D<= (S1) (S2)	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: ANDD＝ (S1) (S2) **ANDD＜ (S1) (S2)** **ANDD＞ (S1) (S2)** **ANDD＜＞ (S1) (S2)** **ANDD＞＝ (S1) (S2)** **ANDD＜＝ (S1) (S2)**	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√

- **Operand description**

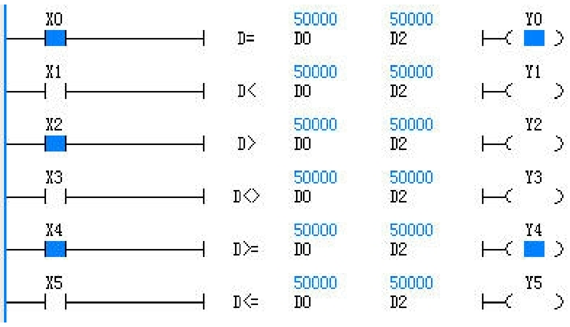
S1: comparison parameter 1

S2: comparison parameter 2

- **Function description**

Compare elements S1 and S2, and use the comparison result in serial connection with other nodes to drive the following operation.

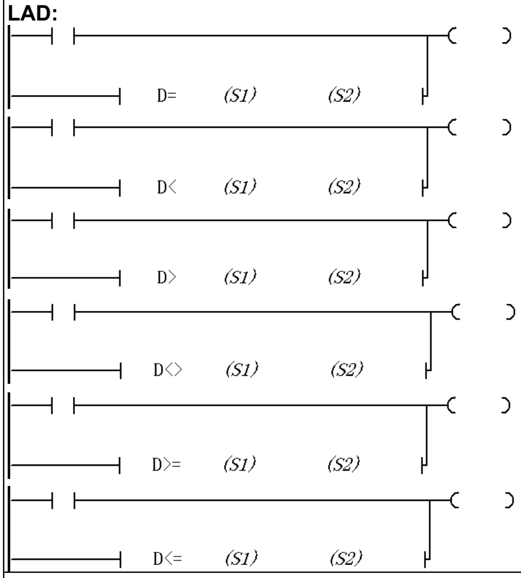
- **Example**



```
LD      X0
LDD=    D0 D2
OUT     Y0
LD      X1
LDD<    D0 D2
OUT     Y1
LD      X2
LDD<>   D0 D2
OUT     Y2
LD      X3
LDD<>   D0 D2
OUT     Y3
LD      X4
LDD>=   D0 D2
OUT     Y4
LD      X5
LDD<=   D0 D2
OUT     Y5
```

Compare (D0, D1) and (D2,D3), and use the comparison result in serial connection with other nodes to determine the output status of the following element.

6.16.6 ORD(=, <, >, <>, >=, <=): Compare double integer ORD※ instruction

LAD: 	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	
IL: ORD＝ (S1) (S2) **ORD＜ (S1) (S2)** **ORD＞ (S1) (S2)** **ORD＜＞ (S1) (S2)** **ORD＞＝ (S1) (S2)** **ORD＜＝ (S1) (S2)**	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√

- **Operand description**

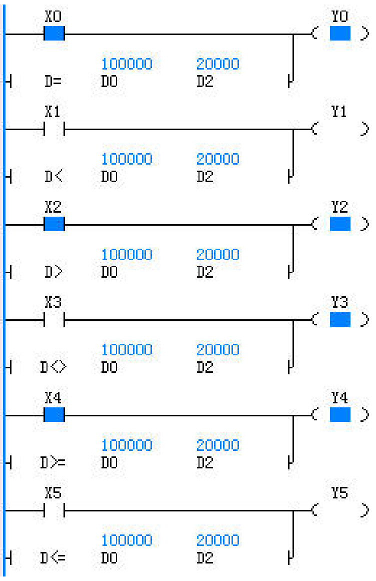
S1: comparison parameter 1

S2: comparison parameter 2

- **Function description**

Compare elements S1 and S2, and use the comparison result in parallel connection with other nodes to drive the following operation.

- **Example**



```
LD        X0
ORD=      D0 D2
OUT       Y0
LD        X1
ORD<      D0 D2
OUT       Y1
LD        X2
ORD<>     D0 D2
OUT       Y2
LD        X3
ORD>=     D0 D2
OUT       Y3
LD        X4
ORD>=     D0 D2
OUT       Y4
LD        X5
ORD<=     D0 D2
OUT       Y5
```

Compare (D0, D1) and (D2,D3), and use the comparison result in parallel connection with other nodes to determine the output status of the following element.

6.16.7 LDR: Compare floating point number instruction

LAD: R= (S1) (S2) ( ) R< (S1) (S2) ( ) R> (S1) (S2) ( ) R<> (S1) (S2) ( ) R>= (S1) (S2) ( ) R<= (S1) (S2) ( )	**Applicable to**	**EC20 EC10 EC20H EC10V**
	Influenced flag bit	
IL: LDR＝ (S1) (S2) **LDR＜ (S1) (S2)** **LDR＞ (S1) (S2)** **LDR＜＞ (S1) (S2)** **LDR＞＝ (S1) (S2)** **LDR＜＝ (S1) (S2)**	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	REAL	Constant							D				V		R	√
S2	RAEL	Constant							D				V		R	√

- **Operand description**

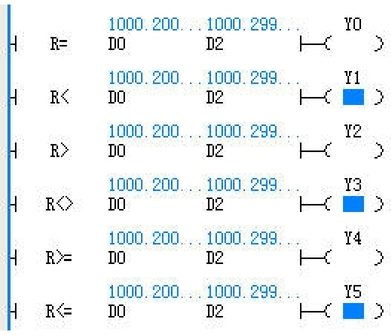
S1: comparison parameter 1

S2: comparison parameter 2

- **Function description**

Compare elements S1 and S2, and use the comparison result to drive the following operation.

- **Example**


1000.200... 1000.299... Y0
R= D0 D2
1000.200... 1000.299... Y1
R< D0 D2
1000.200... 1000.299... Y2
R> D0 D2
1000.200... 1000.299... Y3
R<> D0 D2
1000.200... 1000.299... Y4
R>= D0 D2
1000.200... 1000.299... Y5
R<= D0 D2

```
LDR=      D0 D2
OUT       Y0
LDR<      D0 D2
OUT        Y1
LDR>       D0 D2
OUT        Y2
LDR<>      D0
D2
OUT        Y3
LDR>=      D0
D2
OUT        Y4
LDR<=      D0 D2
OUT         Y5
```

Compare (D0, D1) and (D2,D3), and use the comparison result determine the output status of the following element.

6.16.8 ANDR: Compare floating point number instruction

LAD:	Applicable to	EC20 EC10 EC20H EC10V
R= (S1) (S2) R< (S1) (S2) R> (S1) (S2) R<> (S1) (S2) R>= (S1) (S2) R<= (S1) (S2)	Influenced flag bit	
IL: ANDR= (S1) (S2) ANDR< (S1) (S2) ANDR> (S1) (S2) ANDR<> (S1) (S2) ANDR>= (S1) (S2) ANDR<= (S1) (S2)	Program steps	7

Operand	Type	Applicable elements														Indexed addressing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√

Operand description

S1: comparison parameter 1

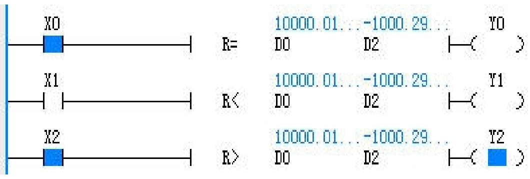
S2: comparison parameter 2

Function description

Compare elements S1 and S2, and use the comparison result in serial connection with other nodes to drive the following operation.

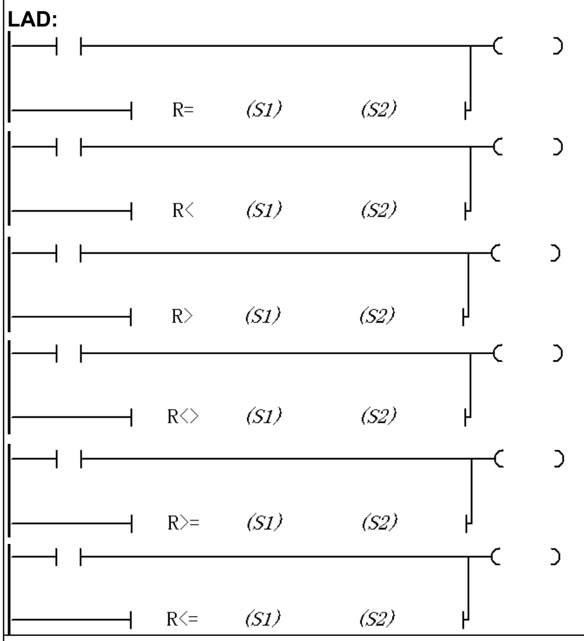
Example

Compare (D0, D1) and (D2,D3), and use the comparison result in serial connection with other nodes to determine the output status of the following element.



```
LD          X0
ANDR=     D0 D2
OUT          Y0
LD           X1
ANDR<    D0 D2
OUT          Y1
LD           X2
ANDR<>        D0
D2
OUT          Y2
LD            X3
ANDR<>       Y3
LD         X4
ANDR>=    D0 D2
OUT         Y4
LD            X5
ANDR<=    D0 D2
OUT          Y5
```

6.16.9 ORR: Compare floating point number instruction

LAD:  R= (S1) (S2) R< (S1) (S2) R> (S1) (S2) R<> (S1) (S2) R>= (S1) (S2) R<= (S1) (S2)	Applicable to	EC20 EC10 EC20H EC10V
	Influenced flag bit	
IL: ORR= (S1) (S2) ORR< (S1) (S2) ORR> (S1) (S2) ORR<> (S1) (S2) ORR>= (S1) (S2) ORR<= (S1) (S2)	Program steps	7

Operand	Type	Applicable elements														Indexed addressing
S1	REAL	Constant							D				V		R	√
S2	REAL	Constant							D				V		R	√

- **Operand description**

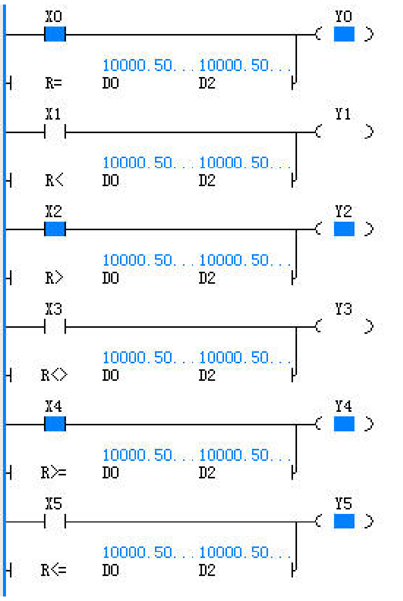
S1: comparison parameter 1

S2: comparison parameter 2

- **Function description**

Compare elements S1 and S2, and use the comparison result in parallel connection with other nodes to drive the following operation.

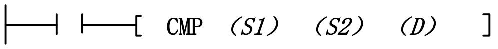
- **Example**



```
LD       X0
ORR=          D0
D2
OUT      Y0
LD       X1
ORR<          D0
D2
OUT      Y1
LD       X2
ORR>     D0 D2
OUT      Y2
LD       X3
ORR<>    D0 D2
OUT      Y3
LD       X4
ORR>=    D0 D2
OUT      Y4
LD       X5
ORR<=    D0 D2
OUT      Y5
```

Compare (D0, D1) and (D2, D3), and use the comparison result in parallel connection with other nodes to determine the output status of the following element.

6.16.10 CMP: Compare and set integer instruction

LAD: 	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	
IL: CMP *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements														Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	Z	R	√
D	BOOL			Y	M	S										

- **Operand description**

S1: data to be comparison values or element SN

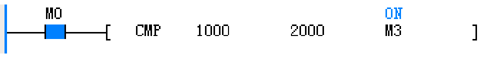
S2: data to be comparison source or element SN

D: starting element SN of output result

- **Function description**

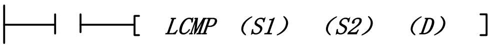
When the power flow is valid, execute the instruction and compare S1 and S2. Set one of (D)(D+1)(D+2) ON according to the result (<, =, >).

- **Example**



LD m0
CMP 1000
2000 M3

6.16.11 LCMP: Compare and set double integer instruction

LAD: 	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	
IL: LCMP *(S1) (S2) (D)*	**Program steps**	**9**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V	Z	R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V	Z	R	√
D	BOOL			Y	M	S										

- **Operand description**

S1: comparison parameter 1

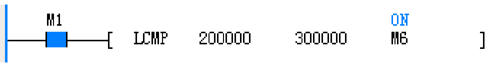
S2: comparison parameter 2

D: starting element SN of output result

- **Function description**

When the power flow is valid, execute the instruction and compare S1 and S2. Set one of (D)(D+1)(D+2) ON according to the result (<, =, >).

- **Example**



LD m1
LCMP 200000 300000
M6

6.16.12 RCMP: Compare and set floating point number instruction

LAD: ├──┤ ├──[RCMP *(S1)* *(S2)* *(D)*]	**Applicable to**	**EC20 EC20H**
	Influenced flag bit	
IL: RCMP *(S1)(S2)(D)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	REAL	Constant							D					R	√
S2	REAL	Constant							D					R	√
D	BOOL			Y	M	S									

- **Operand description**

S1: comparison parameter 1

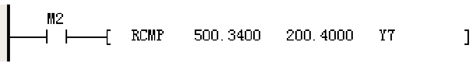
S2: comparison parameter 2

D: starting element SN of output result

- **Function description**

When the power flow is valid, execute the instruction and compare S1 and S2. Set one of (D)(D+1)(D+2) ON according to the result (<, =, >).

- **Example**



LD m2
RCMP 500.3400
200.4000 Y7

6.17 Batch data processing instruction

6.17.1 BKADD: Add batch data operation

LAD: ├──┤ ├──[BKADD *(S1)* *(S2)* *(D)* *(S3)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: BKADD *(S1)* *(S2)* *(D)* *(S3)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D	SD	C	T	V	R	√
S2	INT	Constant							D	SD	C	T	V	R	√
D	INT								D	SD	C	T	V	R	√
S3	INT	Constant							D				V	R	

- **Operand description**

S1: starting element SN for saving the data of add operation

S2: starting element SN for constant or saving the data of add operation

D: starting element SN for saving the result of add operation

S3: number of data

- **Function description**

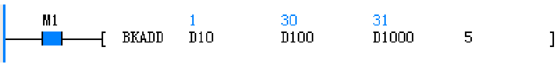
1. When the power flow is valid, execute the instruction, add S3 point 16bit data starting with S1 and S3 point 16bit data (BIN) starting with S2, and store the result in S3 point starting with D.

2. 16bit constant can be designated in S2. If S2 is constant, add S3 point 16bit data starting with S1 and S2, and store the result in S3 point starting with D.

- **Note**

When the operation result overflows, the carry flag will not be set ON.

- **Example**



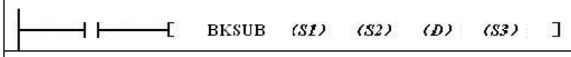
LD M1

BKADD D10 D100 D1000 5

When M1=ON, add the content of 5 units starting with D10 and 5 units starting with D100, and store the result in 5 units starting with D1000.

D1000=D10+D100, D1001=D11+D101,...,D1004=D14+D104.

6.17.2 BKSUB: Subtract batch data operation

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: BKSUB *(S1) (S2) (D) (S3)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D	SD	C	T	V	R	√
S2	INT	Constant							D	SD	C	T	V	R	√
D	INT								D	SD	C	T	V	R	√
S3	INT	Constant							D				V	R	

- **Operand description**

S1: starting element SN for saving the data of subtract operation

S2: starting element SN for constant or saving the data of subtract operation

D: starting element SN for saving the result of subtract operation

S3: number of data

- **Function description**

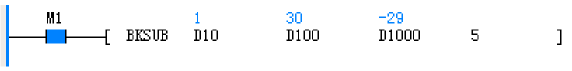
1. When the power flow is valid, execute the instruction, subtract S3 point 16bit data starting with S1 and S3 point 16bit data (BIN) starting with S2, and store the result in S3 point starting with D.

2. 16bit constant can be designated in S2. If S2 is constant, subtract S3 point 16bit data starting with S1 and S2, and store the result in S3 point starting with D.

- **Note**

When the operation result overflows, the carry flag will not be set ON.

- **Example**



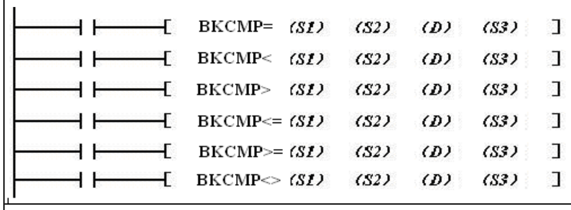
LD M1

BKSUB D10 D100 D1000 5

When M1=ON, subtract the content of 5 units starting with D10 and 5 units starting with D100, and store the result in 5 units starting with D1000.

D1000=D10-D100, D1001=D11-D101,…,D1004=D14-D104.

6.17.3 BKCMP=,>,<,<>,<=,>=: Compare batch data

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: BKCMP=,>,<,<>,<=,>= *(S1) (S2) (D) (S3)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant							D	SD	C	T	V	R	√
S2	INT								D	SD	C	T	V	R	√
D	BOOL		Y	M	S	LM	SM								
S3	INT	Constant							D				V	R	

- **Operand description**

S1: starting element SN for comparison value or stored data

S2: starting element SN to store comparison source data

D: starting element SN to store comparison result

S3: number of data

- **Function description**

1. After comparing S3 point 16bit data starting with S1 and S3 point 16bit data (BIN) starting with S2, store the result in S3 point starting with D.

2. 16bit constant can be designated in S1. If S1 is constant, compare S3 point 16bit data starting with S1 and S2, and store the result in S3 point starting with D.

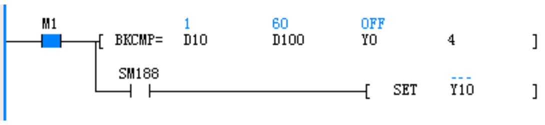
Ladder text in LAD figures: BKSUB (S1) (S2) (D) (S3); BKCMP= / BKCMP< / BKCMP> / BKCMP<= / BKCMP>= / BKCMP<> (S1) (S2) (D) (S3)

3. When the comparison results of S3 point starting with D are ON, set SM188.

- **Note**

When the operation result overflows, the carry flag will not be set ON.

- **Example**



LD M1

BKCMP= D10 D100 Y0 4

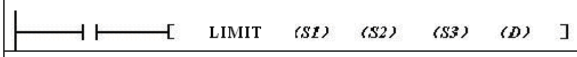
LD SM188

SET Y10

When M1=ON, compare the content of 4 units starting with D10 and 4 units starting with D100, and store the result in 4 units starting with Y0. Besides, when the comparison results are ON, Y10 is set ON.

6.18 Data table instruction

6.18.1 LIMIT: Upper/lower limit control

LAD:  LIMIT (S1) (S2) (S3) (D)	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: LIMIT (S1) (S2) (S3) (D)	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S3	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

- **Operand description**

S1: lower limit

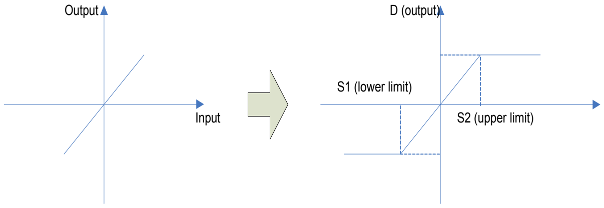
S2: upper limit

S3: input value controlled by upper/lower limit

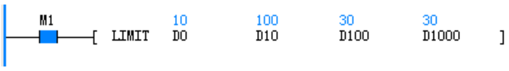
D: starting element SN for saving output value in upper/lower limit control

- **Function description**

Judge whether the input value designated in S3 is in the range of S1 and S2 to control and store the result in D. If S3<S1, D=S1; if S3>S2, D=S2; if S1<=S3<=S2, D=S2.



- **Example**



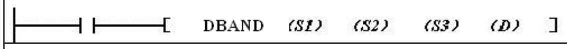
LD M1

LIMIT D0 D10 D100 D1000

When M1=ON, execute D0~D10 upper/lower limit control on the content of D100 and store the result in D1000.

D0(10)<=D100(30)<=D10(100), D1000=30.

6.18.2 DBAND: Dead band control

LAD:  DBAND (S1) (S2) (S3) (D)	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DBAND (S1) (S2) (S3) (D)	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√

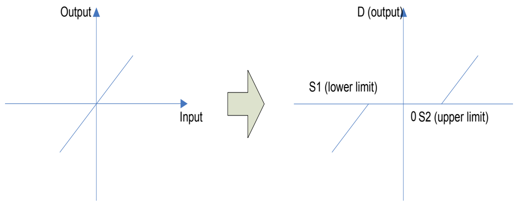
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S3	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

- **Operand description**

S1: lower limit of dead band

S2: upper limit of dead band

S3: input value in dead band control

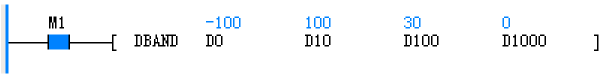


- **Example**

D: starting element SN for saving output value controlled by dead band

- **Function description**

Judge whether the input value designated in S3 is in the range of S1 and S2 to control and store the result in D. If S3<S1, D=S3-S1; if S3>S2, D= S3-S2; if S1<=S3<=S2, D=0.



```
M1            -100    100     30      0
──[ DBAND     D0      D10     D100    D1000    ]
```

LD M1

DBAND D0 D10 D100 D1000

When M1=ON, execute D0~D10 dead band control on the content of D100 and store the result in D1000.

D0 (-100)<D100(30)<D10(100), D1000=0

6.18.3 ZONE: Zone control

LAD: ZONE (S1) (S2) (S3) (D)]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: ZONE (S1) (S2) (S3) (D)	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S3	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

- **Operand description**

S1: negative deviation value to be added to input value

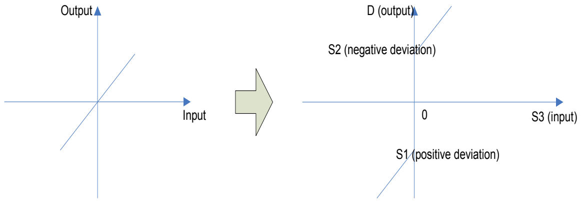
S2: positive deviation value to be added to input value

S3: input value controlled by zone

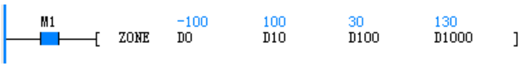
D: starting element SN for saving output value in zone control

- **Function description**

Judge the input value designated in S3 adds the deviation value in S1 or S2 to control and store the result in D. If S3<0, D=S3+S1; if S3>0, D=S3+S2; if S3=0, D=0.



- **Example**



```
M1          -100    100     30      130
──[ ZONE    D0      D10     D100    D1000    ]
```

LD M1

ZONE D0 D10 D100 D1000

When M1=ON, execute D0~D10 zone control on the content of D100 and store the result in D1000.

D100(30)>0, D1000=D100(30)+D10(100), D1000=130.

6.18.4 SCL: Locate coordinate

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: SCL *(S1) (S2) (S3) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT								D				V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

■ **Operand description**

S1: element SN to execute locating coordinate input value or store input value

S2: starting element SN of conversion table for locating coordinate

D: element SN to store output value in located coordinate control

■ **Function description**

1. According to specified conversion features, execute locating coordinate for input value in S1 and store the result in D.

2. The conversion for locating coordinate is executed according to the data table starting with element stored in S2. When the output value is not integer, it will be rounded off to the 1st decimal place.

3. Locating coordinate is set by conversion table:

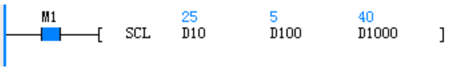
Coordinate point		S2
Point 1	X coordinate	S2+1
	Y coordinate	S2+2
Point 2	X coordinate	S2+3
	Y coordinate	S2+4
…	…	…
Point n (end)	X coordinate	S2+2n-1
	Y coordinate	S2+2n

■ **Note**

1. The data of X in the table should be in an ascending order. If part of data are not in an ascending order but detect from low bit, the operation before will still be executed.

2. S1 must be in the range set by the data table.

■ **Example**

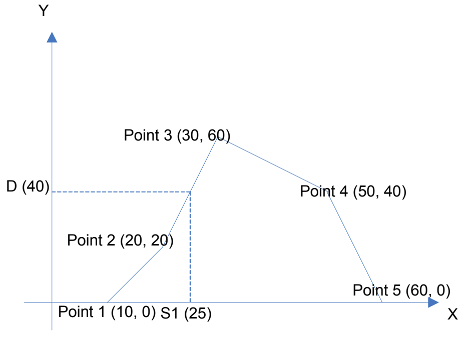


LD M1

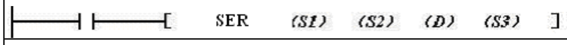
SCL D10 D100 D1000

When M1=ON, execute locating coordinate for the content of D10 and store the result in D1000.

Coordinate point		D100	5
Point 1	X coordinate	D101	10
	Y coordinate	D102	0
Point 2	X coordinate	D103	20
	Y coordinate	D104	20
Point 3	X coordinate	D105	30
	Y coordinate	D106	60
Point 4	X coordinate	D107	50
	Y coordinate	D108	40
Point 5	X coordinate	D109	60
	Y coordinate	D110	0



6.18.5 SER: Search data

LAD:  	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: SER *(S1) (S2) (S3) (D)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

S3	INT	Constant							D				V	R	

■ **Operand description**

S1: starting element SN to search the same data, Max. value and Min. value

S2: element SN to search reference values of the same data, Max. value and Min. value or store target elements

D: starting element SN to store the number after searching the same data, Max. value and Min. value

S3: number of the same search data, Max. value and Min. value (1≤S3≤256)

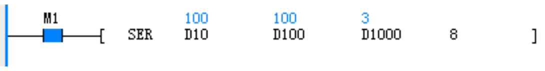
■ **Function description**

1. Search S3 data starting with S1 and the data the same with S2, and store the result in D-D+4.

2. Store the number of the same data and the locations of the initial/final value, Max. value and Min. value for 5 elements starting with D when there are same data.

3. Store 0 for starting 3 elements and other 2 elements as above when there are no same data..

■ **Example**



LD M1

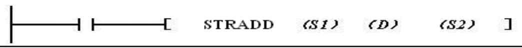
SER D0 D10 D100 D1000 8

When M1=ON, search the content of 8 units starting with D10 and store the result in 5 units starting with D1000.

Search element S1	Value	Comparison element value S2	Data location	Search result D	Value
D10	100	100	0	D1000	3
D11	78		1	D1001	0
D12	92		2	D1002	7
D13	100		3	D1003	5
D14	110		4	D1004	6
D15	-20		5		
D16	145		6		
D17	100		7		

6.19 String instruction

6.19.1 STRADD: Add string

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: STRADD *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	String	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
S2	INT	String	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

■ **Operand description**

S1: the 1st string unit

S2: the 2nd string unit

D: storage added string unit

■ **Function description**

1. When the power flow is valid, add string units starting with S1 and S2, and store the result in the element starting with D;

2. The instruction refers to add the first character of S2 to the end character of S1 and leave out the end flag of S1;

3. The valid data of string units are the data from the element designated by string units to the 1st detected ‘00H’;

4. When the number of strings after connection is odd, add ‘00H’ to the high byte of the end character element; when the number is even, add ‘0000H’ to the next element of the end character element.

■ **Note**

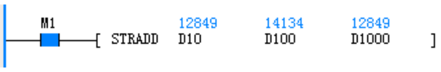
1. When designating strings, S1 and S2 allow 32 characters at most. The comma and double quotation marks indicate the delimiter in upper computer software, so it cannot be identified;

2. When the stored result in S1 and S2 is ‘00H’, add ‘0000H’ in D;

3. When the element addresses of S1 and D or S2 and D overlap, the system will report operand error;

4. When no ‘00H’ exists in the range of relevant elements of string units starting with S1 or S2, the system will report operand error.

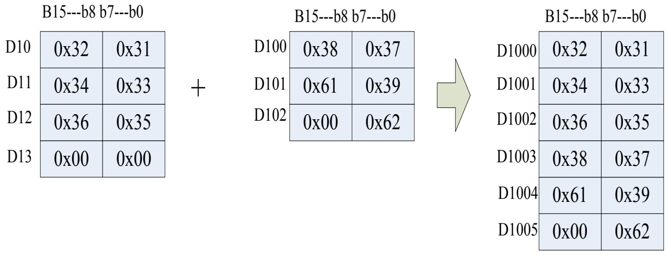
■ **Example**



LD M1

STRADD D10 D100 D1000

When M1=ON, add the string unit starting with D10 and the string unit starting with D100, and store the result in the unit starting with D1000.

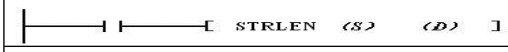


	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

	B15---b8	b7---b0
D100	0x38	0x37
D101	0x61	0x39
D102	0x00	0x62

	B15---b8	b7---b0
D1000	0x32	0x31
D1001	0x34	0x33
D1002	0x36	0x35
D1003	0x38	0x37
D1004	0x61	0x39
D1005	0x00	0x62

6.19.2 STRLEN: Detect string length

LAD: 	Applicable to	EC20H
	Influenced flag bit	Zero, carry, borrow
IL: STRLEN *(S) (D)*	Program steps	5

Operand	Type	Applicable elements													Indexed addressing
S	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

■ **Operand description**

S: string unit

D: string unit length

■ **Function description**

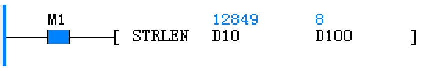
1. When the power flow is valid, detect the length of ***S*** and store the result in D.

2. The valid data of string units are the data from the element designated by string units to the 1st detected '00H'.

■ **Note**

When no '00H' exists in the range of relevant elements of string units starting with S, the system will report operand error.

■ **Example**



LD M1

STRLEN D10 D100

When M1=ON, detect the string unit length starting with D10, and store the result in D100.

6.19.3 STRRIGHT: Read string from the right

LAD:	Applicable to	EC20H
	Influenced flag bit	Zero, carry, borrow
IL: STRRIGHT *(S1) (D) (S2)*	Program steps	7

Operand	Type	Applicable elements													Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S2	INT	Constant							D				V	R	

■ **Operand description**

S1: string unit

D: storage read string unit

S2: number of read strings

■ **Function description**

1. When the power flow is valid, read S2 starting from the end valid character of S1 (except '00H'), and store the result in the element starting with D;

2. When S2=0, store '00H' in D;

3. When the number of read strings is odd, add '00H' to the high byte of the end character element; when the number is even, add '0000H' to the next element of the end character element;

4. The valid data of string units are the data from the element designated by string units to the 1st detected '00H'.

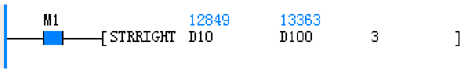
■ **Note**

1. When no '00H' exists in the range of relevant elements of string units starting with S1, the system will report operand error;

2. S2≥0;

3. S2≤the number of S1

- **Example**



LD M1

STRRIGHT D10 D100 3

When M1=ON, read 3 characters on the right of string units starting with D10, and store the result in D100.

	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

⇨

	B15---b8	b7---b0
D100	0x35	0x34
D101	0x00	0x36

6.19.4 STRLEFT: Read string from the left

LAD: [STRLEFT *(S1) (D) (S2)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: STRLEFT *(S1) (D) (S2)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S2	INT	Constant							D				V	R	√

- **Operand description**

S1: string unit

D: storage read string unit

S2: number of read strings

- **Function description**

1. When the power flow is valid, read S2 starting from the left of S1 (except ‘00H’), and store the result in the element starting with D;

2. When S2=0, store ‘00H’ in D;

3. When the number of read strings is odd, add ‘00H’ to the high byte of the end character element; when the number is even, add ‘0000H’ to the next element of the end character element;

4. The valid data of string units are the data from the element designated by string units to the 1st detected ‘00H’.

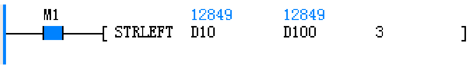
- **Note**

1. When no ‘00H’ exists in the range of relevant elements of string units starting with S1, the system will report operand error;

2. S2≥0;

3. S2≤the number of S1

- **Example**



LD M1

STRLEFT D10 D100 3

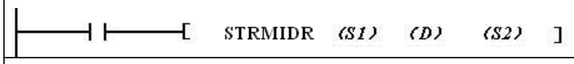
When M1=ON, read 3 characters on the left of string units starting with D10, and store the result in D100.

	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

⇨

	B15---b8	b7---b0
D100	0x32	0x31
D101	0x00	0x33

6.19.5 STRMIDR: Read any strings

LAD:  [STRMIDR *(S1) (D) (S2)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: STRMIDR *(S1) (D) (S2)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S2	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√

- **Operand description**

S1: string unit

D: storage read string unit

S2: initial location of read strings

S2+1 number of read strings n

- **Function description**

1. When the power flow is valid, read n characters starting from the S2 character for S1 string unit, and store the result in the element starting with D;

2. When the number of read strings is odd, add ‘00H’ to the high byte of the end character element; when the number is even, add ‘0000H’ to the next element of the end character element;

3. The valid data of string units are the data from the element designated by string units to the 1st detected ‘00H’;

4. When n=0, no execution;

5. When n=-1, read all the data of S1 and store the result in the element starting with D.

■ **Note**

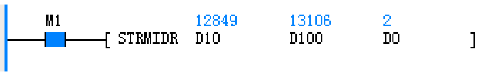
1. S2≤the number of S1;

2. n>-2;

3. S2≥1

4. When no ‘00H’ exists in the range of relevant elements of string units starting with S1, the system will report operand error.

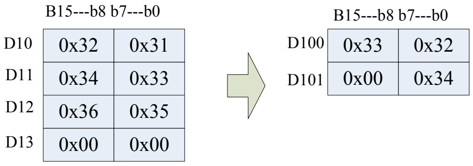
■ **Example**



LD M1

STRMIDR D10 D100 D0

When M1=ON, read D1(D1=3) from D0(D0=2) starting with D10, and store the result in D100.



	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

	B15---b8	b7---b0
D100	0x33	0x32
D101	0x00	0x34

6.19.6 STRMIDW: Replace any strings

LAD: ┤├──[STRMIDW *(S1)* *(D)* *(S2)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: STRMIDW*(S1) (D) (S2)*	**Program steps**	**7**

Operand	Type	Applicable elements												Indexed addressing
S1	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√
S2	INT		KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√

■ **Operand description**

S1: string unit for replacement

D: string unit to be replaced

S2: initial replace location

S2+1 number of replace strings n

■ **Function description**

1. When the power flow is valid, replace n characters from S2 in D with n characters of S1;

2. The valid data of string units are the data from the element designated by string units to the 1st detected ‘00H’;

4. When n=0, no execution;

5. When n=-1, store the content up to the end character designated by S1 after the element designated by D.

■ **Note**

1. S2≤the number of S1;

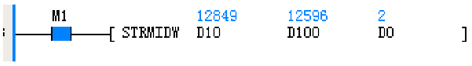
2. n>-2;

3. S2≥1

4. When the replaced characters exceed the end character of string unit starting with D, store the data up to the end character.

5. When no ‘00H’ exists in the range of relevant elements of string units starting with S1 and D, the system will report operand error.

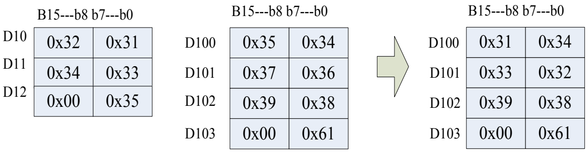
■ **Example**



LD M1

STRMIDW D10 D100 D0

When M1=ON, replace D1(D1=3) of string unit starting with D10 with D1(D1=3) after D0(D0=2) starting with D100.

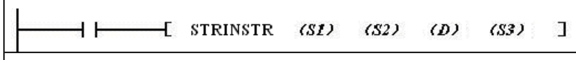


	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x00	0x35

	B15---b8	b7---b0
D100	0x35	0x34
D101	0x37	0x36
D102	0x39	0x38
D103	0x00	0x61

	B15---b8	b7---b0
D100	0x31	0x34
D101	0x33	0x32
D102	0x39	0x38
D103	0x00	0x61

6.19.7 STRINSTR: Search string

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: STRINSTR *(S1) (S2) (D) (S3)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	String						D	SD	C	T	V	R	√
S2	INT							D	SD	C	T	V	R	√
D	INT							D	SD	C	T	V	R	√
S3	INT	Constant						D				V	R	

■ **Operand description**

S1: string unit to be searched

S2: search source

D: search result

S3: initial search location

■ **Function description**

1. When the power flow is valid, search the strings the same with S1 from the S3 character of S2, and store the search result in D;

2. When the strings are not consistent, store "0" in D;

3. When S3 is negative or "0", no execution;

4. The valid data of string units are the data from the element designated by string units to the 1st detected '00H'.

■ **Note**

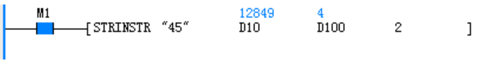
1. When no '00H' exists in the range of relevant elements of string units starting with S1 and S2, the system will report operand error;

2. S3≤the number of S2;

3. When designating strings, S1 allows 32 characters at most. The comma and double quotation marks indicate the delimiter in upper computer software, so it cannot be identified;

4. When S1 is the empty string ('00H'), the result will be the '00H' location of S2 (the 1st '00H' location if S2 is even).

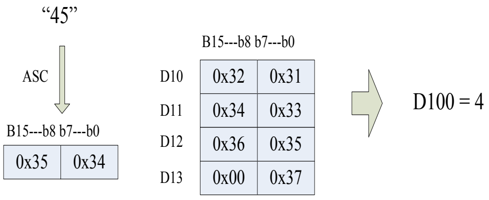
■ **Example**



LD M1

STRINSTR "45" D10 D100 2

When M1=ON, search the character the same with "45" from the 2nd character of the string unit starting with D10, and store the result in D100.`



6.19.8 STRMOV: Move string

LAD:	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: STRMOV *(S) (D)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S	INT	String	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T	V	R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C	T	V	R	√

■ **Operand description**

S: source string unit

D: target unit

■ **Function description**

1. Move all data of S including '00H' to the element unit starting with D;

2. The valid data of string units are the data from the element designated by string units to the 1st detected '00H'.

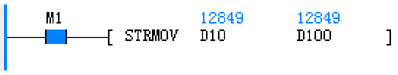
■ **Note**

1. When no '00H' exists in the range of relevant elements of string units starting with S, the system will report operand error;

2. When the number of S is even, '00H' will be stored in low byte while it can be stored in corresponding high byte in D;

3. When designating strings, S1 allows 32 characters at most. The comma and double quotation marks indicate the delimiter in upper computer software, so it cannot be identified.

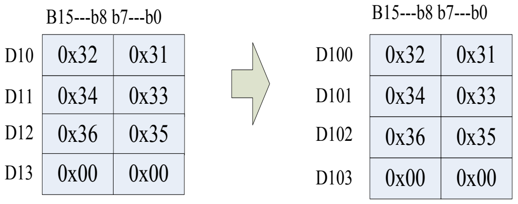
■ **Example**



LD M1

STRMOV D10 D100

When M1=ON, move the string starting with D10 to the unit starting with D100.

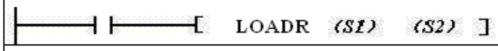


	B15---b8	b7---b0
D10	0x32	0x31
D11	0x34	0x33
D12	0x36	0x35
D13	0x00	0x00

	B15---b8	b7---b0
D100	0x32	0x31
D101	0x34	0x33
D102	0x36	0x35
D103	0x00	0x00

6.20 Extension file register instruction

6.20.1 LOADR: Read extension file register

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: LOADR *(S1) (S2)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S1	INT													R	√
S2	INT	Constant							D						

■ **Operand description**

S1: element unit of extension register for data storage

S2: number of read points (1≤***S2***≤1024)

■ **Function description**

Read S2 starting with S1 in extension file register stored in the memory into the element starting with S1 in the extension register.

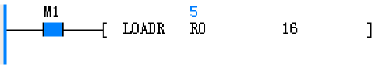
■ **Note**

1. When S2=0, no execution;

2. When the memory is not connected, the system will report no memory card;

3. When S2=1024, the execution time for the instruction will be about 80ms. In actual use, please set watchdog time correctly.

■ **Example**



LD M1

LOADR R0 16

When M1=ON, read 16 data starting with R0 unit in memory card R backup area, and store the result in 16 element units starting with R0.

6.20.2 SAVER: Write extension file register

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: SAVER *(S1) (S2) (D)*	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT													R	√
S2	INT	Constant													
D	INT								D						√

■ **Operand description**

S1: element unit of extension register for data storage (only designate the starting address of the stage)

S2: number of write points in each operation cycle (1≤***S2***≤2048)

D: stored number of write points

■ **Function description**

1. By 2048/S2 (aliquant, add 1 to the result) operation cycle, write the data in extension register from S1 to S1+2047 into the same units in the memory;

2. In write process, store the number of write points in D;

3. After the instruction is executed, set SM189 for the end of execution.

■ **Note**

1. When S2=2048, the execution time for the instruction will be about hundreds of milliseconds. In actual use, please set watchdog time correctly;

2. Before driving SAVER, execute INITER or INITR instruction and initialize the operation stage. If the write data in extension file register are inconsistent with those in extension register, report memory card operation error;

3. When the value designated in S2 is 0, execute the instruction according to S2=2048.

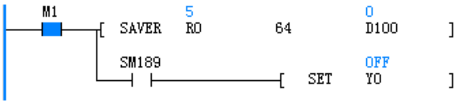
4. The value in S1 must be the starting element number of the stage, as shown below:

Stage No.	Starting element No.	Range of write memory card R backup area	Stage No.	Starting element No.	Range of write memory card R backup area
0	R0	R0~R2047	8	R16384	R16384~R18431
1	R2048	R2048~R4095	9	R18432	R18432~R20479
2	R4096	R4096~R6143	10	R20480	R20480~R22527
3	R6144	R6144~R8191	11	R22528	R22528~R24575
4	R8192	R8192~R10239	12	R24576	R24576~R26623
5	R10240	R10240~R12287	13	R26624	R26624~R28671
6	R12288	R12288~R14335	14	R28672	R28672~R30719
7	R14336	R14336~R16383	15	R30720	R30720~R32767

5. When the memory is not connected, the system will report no memory card;

6. Under memory hardware write protection, report memory card operation error.

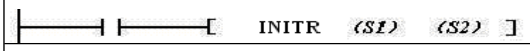
■ **Example**



```
LD   M1
SAVER   R0   64   D100
LD   SM189
SET   Y0
```

When M1=ON, store the data of 2048 units starting with R0 into 2048 units starting with ER0, and store the number of units in D100. After the instruction is executed, SM189=ON and Y0=ON.

6.20.3 INITR: Initialize extension register

<table>
<tr><td colspan="7">LAD:

INITR (S1) (S2)</td><td colspan="3"><b>Applicable to</b></td><td colspan="6"><b>EC20H</b></td></tr>
<tr><td colspan="7"></td><td colspan="3"><b>Influenced flag bit</b></td><td colspan="6"><b>Zero, carry, borrow</b></td></tr>
<tr><td colspan="7"><b>IL: INITR <i>(S1) (S2)</i></b></td><td colspan="3"><b>Program steps</b></td><td colspan="6"><b>5</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="13">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S1</td><td>INT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>INT</td><td>Constant</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

■ **Operand description**

S1: unit of extension register and extension file register for initialization (only designate the starting address of the stage)

S2: number of stages of extension register and extension file register for initialization (S2=1)

■ **Function description**

1. Initialize S2 stage extension register and extension file register starting with S1, initial value: 0xFFFF;

2. The initialization is executed in stages.

■ **Note**

1. The value in S1 must be the starting element number of the stage, as shown below:

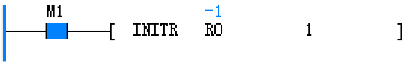
Stage No.	Starting element No.	Range of write memory card R backup area	Stage No.	Starting element No.	Range of write memory card R backup area
0	R0	R0~R2047	8	R16384	R16384~R18431
1	R2048	R2048~R4095	9	R18432	R18432~R20479
2	R4096	R4096~R6143	10	R20480	R20480~R22527
3	R6144	R6144~R8191	11	R22528	R22528~R24575
4	R8192	R8192~R10239	12	R24576	R24576~R26623
5	R10240	R10240~R12287	13	R26624	R26624~R28671
6	R12288	R12288~R14335	14	R28672	R28672~R30719
7	R14336	R14336~R16383	15	R30720	R30720~R32767

2. When the memory card is not used, the instruction will be not executed;

3. When connecting the memory card under memory hardware write protection, report memory card operation error;

4. The instruction can only initialize one stage at a time. If the memory card is used, the initialization time for each stage is about 100ms. In actual use, please set watchdog time correctly.

■ **Example**

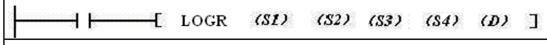


LD M1

INITR R0 1

When M1=ON, initialize extension register R0~R2047 at stage 0. If the memory card is used, ER0~ER2047 will be also initialized.

6.20.4 LOGR: Log in extension register

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: LOGR ***(S1) (S2) (S3) (S4) (D)***	**Program steps**	**11**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D		C	T			√
S2	INT	Constant							D						
S3	INT													R	
S4	INT	Constant													
D	INT								D						√

■ **Operand description**

S1: unit to execute log in

S2: number of units (1~1024)

S3: starting element address

S4: stage of elements (1~16)

D: number of logged data

■ **Function description**

1. When the power flow is valid, till the extension register starting with S3 and S4 extent of the extension file register are filled totally, log in S2 point starting with S1 all the time;

2. Log in at each operation cycle;

3. Store the number of logged data in D.

■ **Note**

1. When using the memory card, initialize the login stages. If the data for logging in extension file register are inconsistent with the starting data of S1, report memory card operation error. The initialization can be executed on the login stages by INITR or INITER instruction or on all ER elements by menu clear command from background memory card (need to select user program, global variable, data block and system block at the same time). In order to avoid data loss, please back up the content of the memory card by background software before initialization.

2. The value in S3 must be the starting element number of the stage, as shown below:

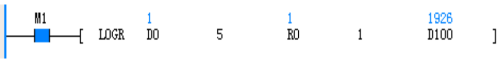
Stage No.	Starting element No.	Range of write memory card R backup area	Stage No.	Starting element No.	Range of write memory card R backup area
0	R0	R0~R2047	8	R16384	R16384~R18431
1	R2048	R2048~R4095	9	R18432	R18432~R20479
2	R4096	R4096~R6143	10	R20480	R20480~R22527
3	R6144	R6144~R8191	11	R22528	R22528~R24575
4	R8192	R8192~R10239	12	R24576	R24576~R26623
5	R10240	R10240~R12287	13	R26624	R26624~R28671
6	R12288	R12288~R14335	14	R28672	R28672~R30719
7	R14336	R14336~R16383	15	R30720	R30720~R32767

3. Under memory hardware write protection, report memory card operation error.

4. Format of login data

S3~S3+S2-1	Storage address of the 1st login data S3~S3+S2-1	D=S2	Data write area 1926 * S4	Storage area of login data
S3+S2~S3+2S2-1	Storage address of the 2nd login data S3~S3+S2-1	D=2S2		
S3+2S2~S3+3S2-1		D=3S2		
…………	…………	……		
S3+1926*S4-1 ~S3+2048*S4-1	Control area of write location Each time use 1 word write area, change ON to OFF in turn from 0 bit of S3+1926*S4-1. When all S3+1926*S4-1 become OFF, use next element S3+1926*S4.		Control area 122 * S4	

■ **Example**

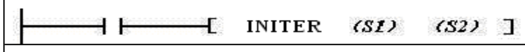


LD M1

LOGR D0 5 R0 1 D100

When M1=ON, log in the data of D0~D4 in R0~R2047 at stage 0, and record the login data number in D100.

6.20.5 INITER: Initialize extension file register

LAD: 	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: INITER *(S1) (S2)*	**Program steps**	**5**

Operand	Type	Applicable elements													Indexed addressing
S1	INT													R	√
S2	INT	Constant													

■ **Operand description**

S1: extension register unit in the same address as extension file register unit to be initialized (only designate the starting address of the stage)

S2: stage number of extension register and extension file register to be initialized (S2=1)

■ **Function description**

1. Initialize S2 stage extension file register starting with S1 in memory and write 0xFFFF;

2. The initialization is executed in stages.

■ **Note**

1. The value in S1 must be the starting element number of the stage, as shown below:

Stage No.	Starting element No.	Range of write memory card R backup area	Stage No.	Starting element No.	Range of write memory card R backup area
0	R0	R0~R2047	8	R16384	R16384~R18431
1	R2048	R2048~R4095	9	R18432	R18432~R20479
2	R4096	R4096~R6143	10	R20480	R20480~R22527
3	R6144	R6144~R8191	11	R22528	R22528~R24575
4	R8192	R8192~R10239	12	R24576	R24576~R26623

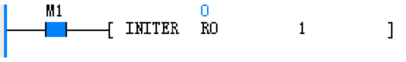
5	R10240	R10240~R12287	13	R26624	R26624~R28671
6	R12288	R12288~R14335	14	R28672	R28672~R30719
7	R14336	R14336~R16383	15	R30720	R30720~R32767

2. Under memory hardware write protection, report memory card operation error;

3. When the memory is not connected, the system will report no memory card;

4. The instruction can only initialize one stage at a time and the initialization time for each stage is about 100ms. In actual use, please set watchdog time correctly.

■ **Example**



LD M1

INITER R0 1

When M1=ON, initialize extension file register ER0~ER2047 at stage 0.

6.21 Locating instruction

6.21.1 ZRN: Regress to origin instruction

<table>
<tr><td colspan="11" rowspan="2"><b>LAD:</b>
ZRN (S1) (S2) (S3) (D)</td><td colspan="3"><b>Applicable to</b></td><td colspan="3"><b>EC20 EC10 EC20H EC10V</b></td></tr>
<tr><td colspan="3"><b>Influenced flag bit</b></td><td colspan="3"><b>Zero, carry, borrow</b></td></tr>
<tr><td colspan="11"><b>IL: ZRN (<i>S1</i>) (<i>S2</i>) (<i>S3</i>) (<i>D</i>)</b></td><td colspan="3"><b>Program steps</b></td><td colspan="3"><b>11</b></td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S1</td><td>DINT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>DINT</td><td>Constant</td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td></td><td>V</td><td></td><td>R</td><td>√</td></tr>
<tr><td>S3</td><td>BOOL</td><td></td><td>X</td><td>Y</td><td>M</td><td>S</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>D</td><td>BOOL</td><td></td><td></td><td>Y</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Operand description

S1: zero return speed, specifying the zero return start speed

32-bit instruction: EC10, EC20: 10~100000(Hz); EC10V: Y0, Y1 10~100000(Hz), Y2, Y3 10~10000(Hz); EC20H: 10~200000(Hz)

S2: crawling speed, specifying the relatively low speed when the proximity signal is ON

S3: proximity signal, specifying the X point for inputting proximity signal

If a non-X element is specified, the position offset of the zero point will increase due to the influence of the PLC calculation cycle.

D: starting address of the high-speed pulse output

EC10, EC20: Y0 or Y1; EC10V: Y0,Y1,Y2,Y3; EC20H: Y0, Y2, Y4, Y5, Y6, Y7

Function description

When SM85 clearing function is enabled, the CLR signals for high-speed pulse outputs Y0 and Y1 are output through Y2 and Y3 respectively. When SM85 is set, the CLR signals will be output to the servo amplifier through Y2 and Y3.

Note

Time sequence chart

1. Because the ZRN instruction is incapable of searching the proximity signal automatically, the zero return operation must start earlier than where the proximity sensor is located.

2. During the return to zero process, the value of the current value register will decrease.

3. The minimum value of actual output pulse frequency is determined by the following formula:

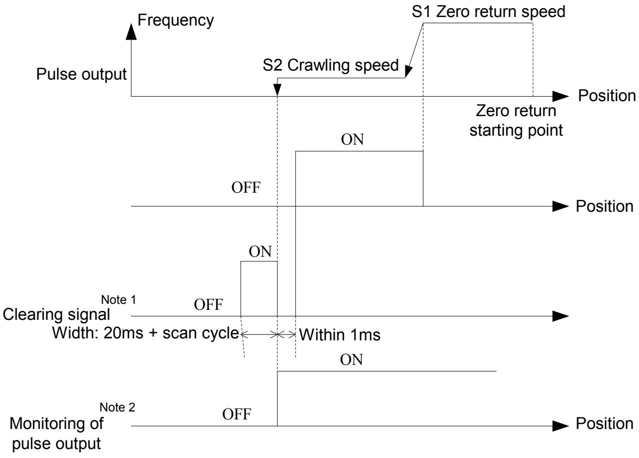
$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

$F_{\max}$ is the maximum speed set by SD85 and SD86, T is ACC/DEC time set by SD87 (unit: ms), $F_{\min\_acc}$ is the minimum output frequency limit.

4. The output pulse frequency will still output the frequency of the calculated result even if designated the value lower than the calculated result. The frequency of the initial acceleration and final deceleration cannot be lower than the calculated result. If the maximum speed is below the calculated result, there will be no pulse output.

5. 0<crawling speed<one tenth of the Max. speed

6. Refer to *Chapter 11 Using locating function*.



Note 1: When SM85 is set, the clearing function is valid
Note 2: SM82 & SM83 are the monitors of Y0 & Y1 pulse outputs respectively

6.21.2 PLSV: Variable speed pulse output instruction

LAD:	Applicable to	EC20 EC10 EC20H EC10V
├─┤ ├─[PLSV *(S)* *(D1)* *(D2)*]	Influenced flag bit	Zero, carry, borrow
IL: PLSV (*S*) (*D1*) (*D2*)	Program steps	8

Operand	Type	Applicable elements													Indexed addressing
S	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												
D2	BOOL			Y	M	S										

- **Operand description**

S: output pulse frequency(Hz)

32-bit instruction: EC10, EC20: 10~100000(Hz), -10~-100000(Hz);

EC10V: Y0,Y1 10~100000(Hz), -10~-100000(Hz), Y2,Y3 10~10000(Hz), -10~-10000(Hz);

EC20H: 10~200000(Hz), -10~-200000(Hz)

D1: starting address of the high-speed pulse output

EC10, EC20: Y0 or Y1; EC10V: Y0, Y1, Y2, Y3; EC20H: Y0, Y2, Y4, Y5, Y6, Y7

D2: starting address of rotating direction signal output. Its state is determined by ***S***:

When ***S*** is positive: ***D2*** is ON

When ***S*** is negative: ***D2*** is OFF

- **Function description**

1. You can change S even in the state of outputting high-speed pulses

2. Because there will be no acceleration or deceleration during the start & stop, if buffer is needed during the start or stop, it is recommended to use the RAMP instruction to change the value of pulse frequency S.

3. In the process of high-speed pulse output, when the power flow driven by the instruction turns OFF, the output will stop without deceleration.

4. If the corresponding high-speed pulse output monitor (SM82 or SM83) is ON, the power flow driven by the instruction will not be driven by the instruction again after the power flow turns OFF.

5. The direction is determined by the positive or negative nature of S.

- **Note**

1. The high-speed I/O instructions, PLS instruction and locating instructions can use Y0 or Y1 to output high-speed pulses. However, take care not to use more than one such instructions on Y0 or Y1 at one time.

2. Refer to *Chapter 11 Using locating function*.

6.21.3 DRVI: Relative position control instruction

LAD: |—| |—[DRVI *(S1)* *(S2)* *(D1)* *(D2)*]	**Applicable to**	**EC20 EC10 EC10A EC20H EC10V**
	Influenced flag bit	**Zero, carry, borrow**
IL: DRVI (*S1*) (*S2*) (*D1*) (*D2*)	**Program steps**	**11**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												
D2	BOOL			Y	M	S										

- **Operand description**

S1: output pulse number (relatively specified)

32-bit instruction: -999999~+999999

S2: output pulse frequency(Hz)

32-bit instruction: EC10, EC20: 10~100000(Hz); EC10V: Y0,Y1 10~100000(Hz),Y2,Y3 10~10000(Hz); EC20H: 10~200000(Hz)

D1: starting address of the high-speed pulse output

EC10, EC20: Y0 or Y1; EC10V: Y0,Y1,Y2,Y3; EC20H: Y0, Y2, Y4, Y5, Y6, Y7

D2: starting address of rotating direction signal output. Its state is determined by ***S1***:

When ***S1*** is positive: ***D2*** is ON

When ***S1*** is negative: ***D2*** is OFF

- **Function description**

1. S1 is stored in the following current registers:

Y0 output: SD80, SD81 (32-bit)

Y1 output: SD82, SD83 (32-bit)

2. When D2 is OFF, the value of the current value register will decrease.

3. The rotating direction is determined by the positive or negative nature of S1.

4. Changing the operands during the execution of the instruction will not take effect until the next cycle when the instruction is executed again.

5. During the execution of the instruction, the output will decelerate to stop when the driven contact turns OFF. The execution completion flag SM will not act then.

6. If the corresponding high-speed pulse output control (SM80 or SM81) is ON, the contact driven by the instruction will not be driven by the instruction again after the the power flow turns OFF.

- **Note**

1. The minimum value of actual output pulse frequency is determined by the following formula:

$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

$F_{\max}$ is the maximum speed set by SD85 and SD86, T is ACC/DEC time set by SD87 (unit: ms), $F_{\min\_acc}$ is the minimum output frequency limit.

2. The output pulse frequency will still output the frequency of the calculated result even if designated the value lower than the calculated result. The frequency of the initial acceleration and final deceleration cannot be lower than the calculated result. If the maximum speed is below the calculated result, there will be no pulse output.

3. 0<crawling speed<one tenth of the Max. speed

4. Refer to *Chapter 11 Using locating function*.

6.21.4 DRVA: Absolute position control instruction

LAD: ├──┤ ├──[DRVA (S1) (S2) (D1) (D2)]	**Applicable to**	**EC20 EC10 EC20H EC10V**
	Influenced flag bit	**Zero, carry, borrow**
IL: DRVA (*S1*) (*S2*) (*D1*) (*D2*)	**Program steps**	**11**

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												
D2	BOOL			Y	M	S										

- **Operand description**

S1: targe position (absolutely specified)

32-bit instruction: -999999~+999999

S2: output pulse frequency(Hz)

32-bit instruction: EC10, EC20: 10~100000(Hz); EC10V: Y0,Y1 10~100000(Hz),Y2,Y3 10~10000(Hz); EC20H: 10~200000(Hz)

D1: starting address of the high-speed pulse output

EC10, EC20: Y0 or Y1; EC10V: Y0,Y1,Y2,Y3; EC20H: Y0, Y2, Y4, Y5, Y6, Y7

PLC output must apply transistor output.

D2: starting address of rotating direction signal output

When the output pulse is positive, ***D2*** is ON

When the output pulse is negative, ***D2*** is OFF

- **Function description**

1. Output pulse in S1 corresponding to the current register as the relative position as follows:

SD80	Y0 outputs the current position of locating instruction (MSB)(EC10)
SD81	Y0 outputs the current position of locating instruction (LSB)(EC10)
SD82	Y1 outputs the current position of locating instruction (MSB) (EC10)
SD83	Y1 outputs the current position of locating instruction (LSB) (EC10)
SD200	Y0 outputs the current position of locating instruction (MSB) (EC20,EC10V,EC20H)
SD201	Y0 outputs the current position of locating instruction (LSB) (EC20, EC10V,EC20H)
SD210	Y1 outputs the current position of locating instruction (MSB) (EC20, EC10V)
SD211	Y1 outputs the current position of locating instruction (LSB) (EC20, EC10V)
SD320	Y2 outputs the current position of locating instruction (MSB) (EC10V,EC20H)
SD321	Y2 outputs the current position of locating instruction (LSB) (EC10V,EC20H)
SD330	Y3 outputs the current position of locating instruction (MSB) (EC10V)
SD331	Y3 outputs the current position of locating instruction (LSB) (EC10V)
SD340	Y4 outputs the current position of locating instruction (MSB) (EC20H)
SD341	Y4 outputs the current position of locating instruction (LSB) (EC20H)
SD350	Y5 outputs the current position of locating instruction (MSB) (EC20H)
SD351	Y5 outputs the current position of locating instruction (LSB) (EC20H)
SD360	Y6 outputs the current position of locating instruction (MSB) (EC20H)
SD361	Y6 outputs the current position of locating instruction (LSB) (EC20H)
SD370	Y7 outputs the current position of locating instruction (MSB) (EC20H)
SD371	Y7 outputs the current position of locating instruction (LSB) (EC20H)

2. When D2 is OFF, the value of the current value register will decrease.

3. The rotating direction is determined by the positive or negative nature of S1.

4. Changing the operands during the execution of the instruction will not take effect until the next cycle when the instruction is executed again.

5. During the execution of the instruction, the output will decelerate to stop when the driven contact turns OFF. The execution completion flag SM will not act then.

6. If the corresponding high-speed pulse output control (SM80 or SM81) is ON, the contact driven by the instruction will not be driven by the instruction again after the the power flow turns OFF.

- **Note**

1. The minimum value of actual output pulse frequency is determined by the following formula:

$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

$F_{\max}$ is the maximum speed set by SD85 and SD86, T is ACC/DEC time set by SD87 (unit: ms), $F_{\min\_acc}$ is the minimum output frequency limit.

2. The output pulse frequency will still output the frequency of the calculated result even if designated the value lower than the calculated result. The frequency of the initial acceleration and final deceleration cannot be lower than the calculated result. If the maximum speed is below the calculated result, there will be no pulse output.

3. 0<crawling speed<one tenth of the Max. speed

4. Refer to *Chapter 11 Using locating function*.

6.21.5 ABS: Read current value instruction

LAD: ├─┤ ├─[ABS (*S*) (*D1*) (*D2*)]															
Applicable to	**EC20 EC10 EC20H**														
Influenced flag bit	**Zero, carry, borrow**														
IL: ABS (*S*) (*D1*) (*D2*)	**Program steps**	**8**													
Operand	Type	Applicable elements													Indexed addressing
S	BOOL	X	Y	M	S										
D1	BOOL		Y	M	S										
D2	DINT		KnY	KnM	KnS					D	SD	C			R	√

- **Operand description**

S: the input point from servo

The input points occupies three consecutive Xs (***S***, ***S***+1 and ***S***+2) or other bit elements.

D1: output points to servo.

The output points occupies three consecutive Ys (***D1***, ***D1***+1 and ***D1***+2) or other bit elements

D2: the current value (32-bit) read from servo.

The current value occupies two word elements: ***D2*** (MSB) and ***D2***+1 (LSB). Because the read current value must be written into SD80 or SD82 (32-bit signed integer), you can directly specify SD80 or SD82 as ***D2***.

- **Function description**

1. You should power on the PLC and servo amplifier at the same time, or power on the servo amplifier first, in order to make sure that the servo amplifier is ON before the PLC enters the RUN state.

2. The read current value D2 can be stored in any word element, but the current value must be eventually stored in SD80 or SD82.

3. The power flow of the ABS instruction should be ON after the current value is read, otherwise the servo amplifier will turn OFF.

4. SM82 and SM83 are the output monitors of Y0 and Y1. The monitors will turn OFF after the output is complete.

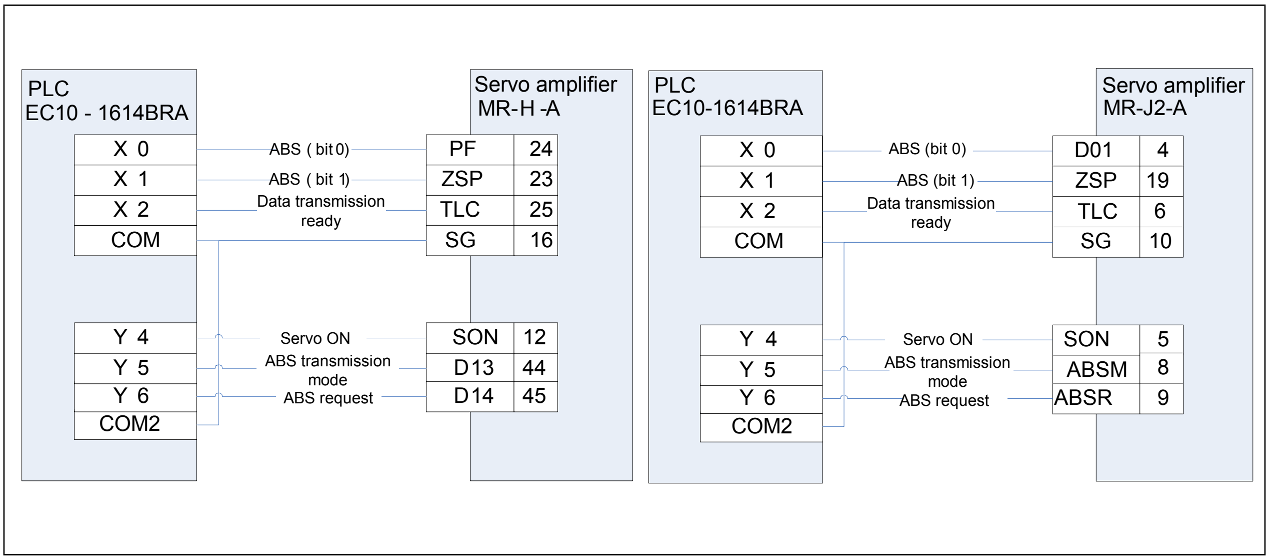
5. When the power flow is valid and the servo is ON, the ABS instruction will send the transmission mode signal.

6. When the data transmission ready signal and the ABS request signal coincide with each other, the (32+6)bit data communication will start.

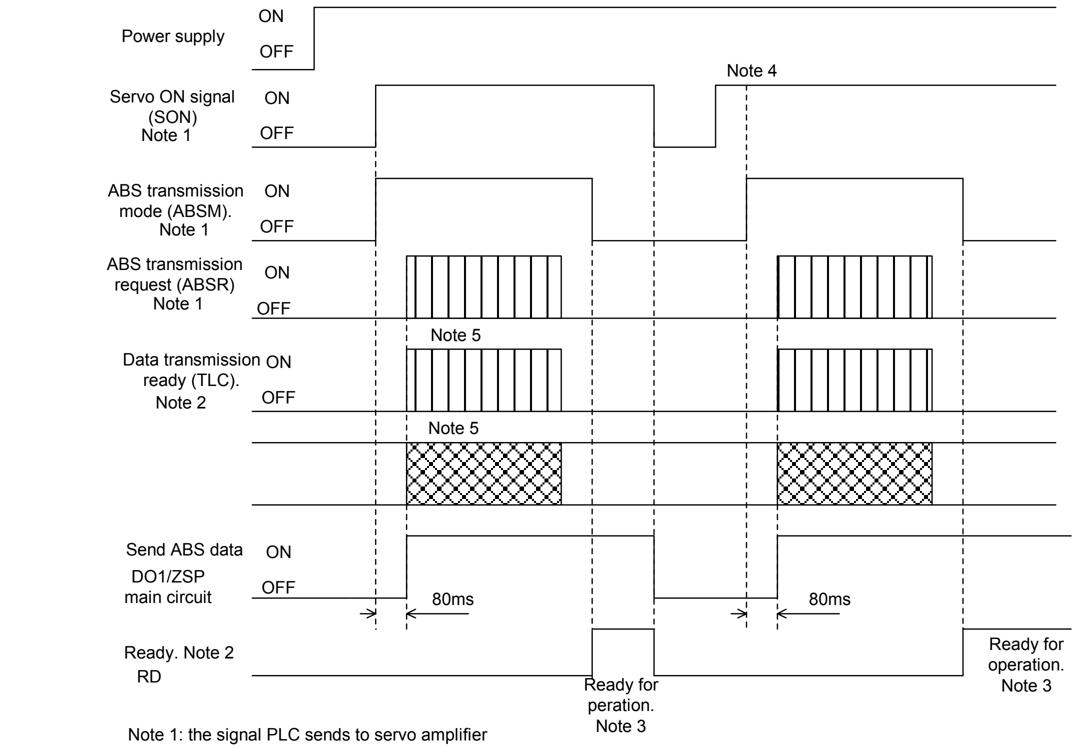
7. The data are tranmitted through the ABS 2-bit (bit0 & bit1) loop.

8. The system error code for ABS Data Read Timeout is 79; for ABS Data Read and Check Error, 80.

- **The wire connection for the I/O signals of the ABS instruction is as shown in the following figure:**



- **Time sequence chart**


Power supply
ON
OFF
Servo ON signal (SON) Note 1
ON
OFF
Note 4
ABS transmission mode (ABSM). Note 1
ON
OFF
ABS transmission request (ABSR) Note 1
ON
OFF
Note 5
Data transmission ready (TLC). Note 2
ON
OFF
Note 5
Send ABS data DO1/ZSP main circuit
ON
OFF
80ms
80ms
Ready. Note 2 RD
Ready for peration. Note 3
Ready for operation. Note 3

Note 1: the signal PLC sends to servo amplifier
Note 2: the signal servo amplifier sends
Note 3: system data transmission over, ready for normal operation. After RD is set, ABSM signal will not be accepted
Note 4: Here the SON signal is set before ABSM signal. Despite that, the main circuit will not be ON until ABSM is set ON. If transmission mode is interrupted with ABSM being set OFF during the ABS trasmission, the servo amplifier will report overtime alarm (AL.E5).
Note 5: These signal pins' definitions will change upon ABSM set/reset. See the related Mitsubishi product information.

- **Note**

The ABS instruction supports the Mitsubishi MR~J2 and MR~J2S servo amplifiers and use its specialized data transmission protocol to read the current value of absolute position. The ABS instruction is a dedicated 32-bit instruction. For the servo amplifiers of other brands, reading the current value of absolute position requires communication or other designated methods. When the ABS instruction is executed, the related I/O points will be processed accordingly. Therefore, the ABS instruction is applicable only to Mitsubishi servo amplifiers.

6.21.6 DSZR: Regress to origin with DOG search instruction

LAD: ├──┤ ├──[DSZR *(S1) (S2) (D1) (D2)*]	**Applicable to**	**EC20 EC20H EC10V**
	Influenced flag bit	**Zero, carry, borrow**
IL: DSZR *(S1) (S2) (D1) (D2)*	**Program steps**	**9**

Operand	Type	Applicable elements													Indexed addressing
S1	BOOL		X	Y	M	S									
S2	BOOL		X												
D1	BOOL			Y											
D2	BOOL			Y	M	S									

- **Operand description**

S1: element SN of near-point input signal (DOG). When designated to input the element, due to the influence from PLC operation cycle, the origin position offset will be enlarged.

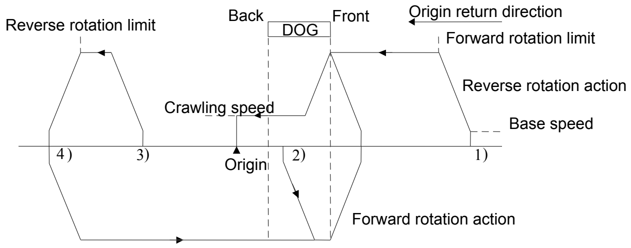
S2: element SN of zero-point input signal. Range: X0~X7

D1: pulse SN of output pulse. EC10, EC20: Y0 or Y1; EC10V: Y0,Y1,Y2,Y3; EC20H: Y0, Y2, Y4, Y5, Y6, Y7

D2: object SN of rotating direction signal output

- **Function description**

The near-point signal and zero-point signal are available. Forward rotation limit and reverse rotation limit are designed. The action of origin return will vary due to different positions. After the instruction is completed, send the clear signal.



1. The initial position before DOG:

1) By executing the instruction of origin return, start acting

2) Move in origin return direction at origin return speed

3) Once detecting the front of DOG, decrease to crawling speed

4) After detecting the back of DOG, stop when detecting the first zero-point signal

2. The initial position in DOG:

1) By executing the instruction of origin return, start acting

2) Move in the opposite direction of origin return at origin return speed

3) After detecting the front of DOG, decrease to stop (leave DOG)

4) Move in origin return direction at origin return speed (enter DOG again)

5) Once detecting the front of DOG, decrease to crawling speed

6) After detecting the back of DOG, stop when detecting the first zero-point signal

3. The initial position at near-point signal OFF (after DOG):

1) By executing the instruction of origin return, start acting

2) Move in origin return direction at origin return speed

3) When detecting reverse rotation limit, decrease to stop

4) Move in the opposite direction of origin return at origin return speed

5) After detecting the front of DOG, decrease to stop (leave DOG)

6) Move in origin return direction at origin return speed

7) Once detecting the front of DOG, decrease to crawling speed

8) After detecting the back of DOG, stop when detecting the first zero-point signal

4. The initial position at limit switch (forward rotation limit or reverse rotation limit):

1) By executing the instruction of origin return, start acting

2) Move in the opposite direction of origin return at origin return speed

3) After detecting the front of DOG, decrease to stop (leave DOG)

4) Move in origin return direction at origin return speed (enter DOG again)

5) Once detecting the front of DOG, decrease to crawling speed

6) After detecting the back of DOG, stop when detecting the first zero-point signal

■ **Note**

1. The PLSY, PLS and locating instructions can output high-speed pulses through Y0 and Y1. Note that only one instruction can use one output port at one time.

2. The minimum value of actual output pulse frequency is determined by the following formula:

$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

$F_{\max}$ is the maximum speed set by SD85 and SD86, T is ACC/DEC time set by SD87 (unit: ms), $F_{\min\_acc}$ is the minimum output frequency limit.

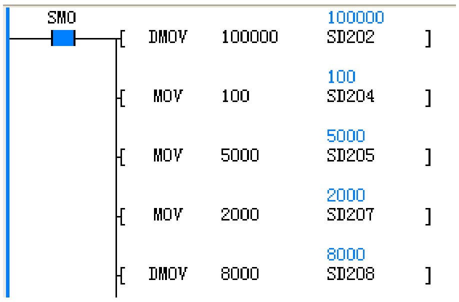
3. The output pulse frequency will still output the frequency of the calculated result even if designated the value lower than the calculated result. The frequency of the initial acceleration and final deceleration cannot be lower than the calculated result. If the maximum speed is below the calculated result, there will be no pulse output.

4. 0<crawling speed<one tenth of the Max. speed

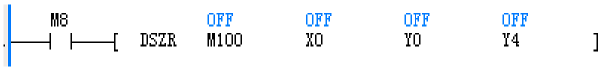
5. Refer to *Chapter 11 Using locating function*.

■ **Example**

The parameters of the Max. speed, base speed, ACC/DEC time, origin return speed and crawling speed can be set by default or by elements.


```
 SM0                          100000
--| |----+--[ DMOV   100000   SD202     ]
         |
         |                    100
         +--[ MOV    100      SD204     ]
         |
         |                    5000
         +--[ MOV    5000     SD205     ]
         |
         |                    2000
         +--[ MOV    2000     SD207     ]
         |
         |                    8000
         +--[ DMOV   8000     SD208     ]
```

Execute regress to origin with DOG search instruction:


```
   M8              OFF      OFF      OFF      OFF
--| |----[ DSZR    M100     X0       Y0       Y4       ]
```

6.21.7 DVIT: Interrupt locating

LAD: ├─┤ ├──[DVIT *(S1)* *(S2)* *(D1)* *(D2)*]	Applicable to	EC20 EC10V
	Influenced flag bit	Zero, carry, borrow
IL: DVIT *(S1)* *(S2)* *(D1)* *(D2)*	Program steps	11

Operand	Type	Applicable elements														Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		R	√
D1	BOOL			Y												
D2	BOOL			Y	M	S										

Operand description

S1: output pulses after interrupt (relative address)

S2: output pulse frequency. EC20: 10~100000(Hz); EC10V:Y0,Y1 10~100000(Hz),Y2,Y3 10~10000(Hz)

D1: SN of output pulse. EC20: Y0~Y1; EC10V: Y0,Y1,Y2,Y3.

D2: object SN of rotating direction signal output

Function description

1. Execute DVIT instruction. SM260 is valid interrupt input function, SD240 is the designation of interrupt input function, SM287 is logic inversion element of interrupt input signal. Logic inversion is to generate interrupts after confirming whether interrupt element is ON or OFF.

2. The designated method of interrupt input: SM260 is set ON.

3. Designate the SN of interrupt input (X0~X7) in SD240 or the user interrupt instruction element. The low 8bit of SD240 corresponds to interrupt input of pulse output Y0 and high 8bit corresponds to interrupt input of pulse output Y1.

Set value	Content
0	Designate X0 to interrupt input signal
1	Designate X1 to interrupt input signal
……	……
7	Designate X7 to interrupt input signal
8[*1]	Designate the user interrupt instruction element[*1] to interrupt input signal

	Pulse output element	User interrupt instruction element	
	Y0	SM289	
	Y1	SM299	

Note

1. The PLSY, PLS and locating instructions can output high-speed pulses through Y0 and Y1. Note that only one instruction can use one output port at one time.

2. The minimum value of actual output pulse frequency is determined by the following formula:

$$F_{\min\_acc}=\sqrt{\frac{F_{\max}\times 500}{T}}$$

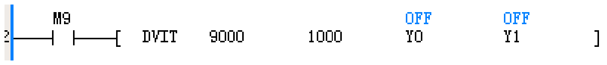
$F_{\max}$ is the maximum speed set by SD85 and SD86, T is ACC/DEC time set by SD87 (unit: ms), $F_{\min\_acc}$ is the minimum output frequency limit.

3. The output pulse frequency will still output the frequency of the calculated result even if designated the value lower than the calculated result. The frequency of the initial acceleration and final deceleration cannot be lower than the calculated result. If the maximum speed is below the calculated result, there will be no pulse output.

4. When the output pulses are less than the pulses needed by deceleration, deceleration frequency action can be completed.

5. Refer to *Chapter 11 Using locating function*.

Example



6.21.8 STOPDV: Pulse output stop instruction

LAD: ├┤ ├[STOPDV (S1) (S2) (S3) (D)]	Applicable to	EC20H
	Influenced flag bit	Zero, carry, borrow
IL: STOPDV (*S1*) (*S2*) (*S3*) (*D*)	Program steps	12

Operand	Type	Applicable elements													Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S3	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D	BOOL			Y											

Operand description

S1: output pulses after executing the instruction (relative address)

S2: base speed in deceleration

S3: time from the original output action decelerates to base speed

D: output point SN corresponding to high-speed pulse. Y0, Y2, Y4, Y5, Y6, Y7

Function description

1. Execute STOPDV instruction. Start in the process of PLSY, PLS and locating instructions and stop output action of the designated axis.

2. When the power flow is ON, stop after running the designated pulse. When the pulse is 0, stop output immediately; when the pulse is larger than 0, continue output, then decelerate to base speed, and stop output until reaching base speed.

3. The base speed and ACC/DEC time are set in the special data register of output axis, and the setting will be not changed when executing the instruction; the base speed and ACC/DEC time will be executed according to the setting in Operand rather than the setting in the special data register.

4. The direction signal of output axis needs no designation and it can identify the direction signal designated in original PLSY, PLS and locating instructions. Executing the instruction will not change the state of the direction signal.

Note

1. The minimum value of actual output pulse frequency is determined by the following formula:

$$F_{\min\_acc}=\sqrt{\frac{F_{\max}\times 500}{T}}$$

$F_{\max}$ is the maximum speed set by SD85 and SD86, T is ACC/DEC time set by SD87 (unit: ms), $F_{\min\_acc}$ is the minimum output frequency limit.

2. The output pulse frequency will still output the frequency of the calculated result even if designated the value lower than the calculated result. The frequency of the initial acceleration and final deceleration cannot be lower than the calculated result. If the maximum speed is below the calculated result, there will be no pulse output.

3. When the output pulses are less than the pulses needed by deceleration, deceleration frequency action can be completed.

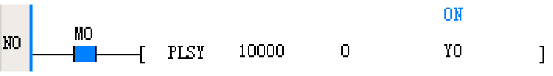
4. D can be designated to Y0, Y2, Y4, Y5, Y6 and Y7. Y1 and Y3 can be direction signal or negative pulse output signal to match with Y0 and Y2 separately.

5. If the output axis is in stop state, perform no operation. If the output axis is executing LIN, CW and CCW instructions, perform no operation.

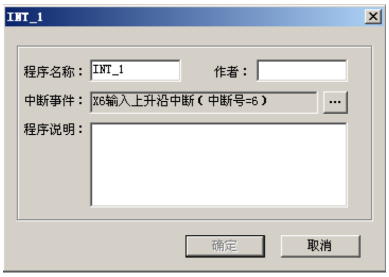
6. Refer to *Chapter 11 Using locating function*.

Example

In the main program, use PLSY instruction to drive Y0. The power flow is controlled by M0 element:



Set the interrupt source of interrupt subprogram, such as:

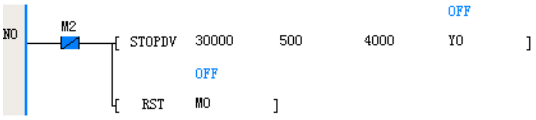


程序名称：INT_1 作者：

中断事件：X6输入上升沿中断（中断号=6）

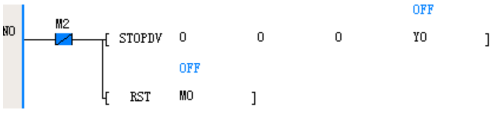
程序说明：

Add the following sentence in interrupt subprogram:



In above instruction, at the instant of setting X6 ON, Y0 will decelerate to stop after outputting pulses. Until Y0 stops completely, output 30000 pulses regardless of the scan cycle.

For STOPDV instruction in INT_1, Operand1 is 0, as shown below:



When the event of interrupt source generates (X6 rising edge), Y0 will stop immediately regardless of the scan cycle.

Note: When executing STOPDV instruction, disconnect relevant power flow of high-speed instruction for Y0 in primary function simultaneously to avoid scanning the instruction and restarting high-speed output after Y0 stops.

6.21.9 LIN: Linear trace interpolation

LAD: LIN *(S) (D1) (D2) (D3) (D4)*	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: LIN (*S*) (*D1*) (*D2*) (*D3*) (*D4*)	**Program steps**	**12**

Operand	Type	Applicable elements													Indexed addressing
S	DINT								D						
D1	BOOL			Y											
D2	BOOL			Y											
D3	BOOL			Y											
D4	BOOL			Y											

Operand description

S: starting address of storage area for parameter list

D1: corresponding output point SN of X axis pulse signal (or positive pulse signal). Range: Y0, Y4, Y5, Y6, Y7

D2: corresponding output point SN of X axis direction signal (or negative pulse signal)

D3: corresponding output point SN of Y axis pulse signal (or positive pulse signal). Range: Y2, Y4, Y5, Y6, Y7

D4: corresponding output point SN of Y axis direction signal (or negative pulse signal)

Function description

1. Move to the target position in linear trace at designated vector speed.

2. Definitions of parameter list

D element	Content

<table>
<tr><th>D element</th><th>Content</th></tr>
<tr><td>S</td><td>Reserved</td></tr>
<tr><td>S+1</td><td>Relation between action mode and output logic (configuration code in decimal)
<table>
<tr><th>Configuration</th><th>Mode</th><th>Output logic</th></tr>
<tr><td>00</td><td>Incremental</td><td>Pulse+direction (positive ON/ negative OFF)</td></tr>
<tr><td>01</td><td>Incremental</td><td>Positive pulse +negative pulse</td></tr>
<tr><td>10</td><td>Absolute</td><td>Pulse+direction (positive ON/</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>D element</th><th>Content</th></tr>
<tr><td></td><td><table><tr><td></td><td></td><td>negative OFF)</td></tr><tr><td>11</td><td>Absolute</td><td>Positive pulse +negative pulse</td></tr></table></td></tr>
<tr><td>S+2</td><td>Resultant speed, initial speed Fmin(Hz) (MSB)</td></tr>
<tr><td>S+3</td><td>Resultant speed, initial speed Fmin(Hz) (LSB)</td></tr>
<tr><td>S+4</td><td>Resultant speed, Max. speed Fmax(Hz) (MSB)</td></tr>
<tr><td>S+5</td><td>Resultant speed, Max. speed Fmax(Hz) (LSB)</td></tr>
<tr><td>S+6</td><td>ACC/DEC time T(ms) (MSB)</td></tr>
<tr><td>S+7</td><td>ACC/DEC time T(ms) (LSB)</td></tr>
<tr><td>S+8</td><td>X axis target position (move distance) (MSB)</td></tr>
<tr><td>S+9</td><td>X axis target position (move distance) (LSB)</td></tr>
<tr><td>S+10</td><td>Y axis target position (move distance) (MSB)</td></tr>
<tr><td>S+11</td><td>Y axis target position (move distance) (LSB)</td></tr>
</table>

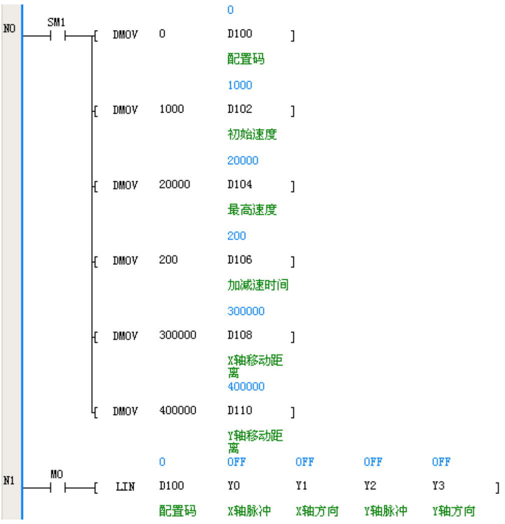
(1) In the incremental mode, the target trace adopts relative address, referring to the move distance of X and Y axes from the current position to the target.

(2) In the absolute mode, the target trace adopts absolute address, referring to the absolute coordinate of the target position on X and Y axes.

Note

1. D1 and D3 must be used in group. When the output group is Y0 and Y2, Y1 and Y3 can be direction signal or negative pulse output signal to match with Y0 and Y2 separately. D1 and D3 can select any two of Y4, Y5, Y6, Y7 for interpolation, and the corresponding direction signals are free to select.
2. Output group (Y0, Y2) can be designated to "pulse+direction" mode or "positive pulse+negative pulse" mode, the Max. speed 100k; Y4, Y5, Y6, Y7 for interpolation are output in "pulse+direction" mode, the Max. speed 20k.
3. The move distance of each axis should not exceed 16,777,215 pulses at a time.
4. Note that only one of PLSY, PLS or locating instructions can be used for one high-speed port at one time.

Example



```
                        0
N0   SM1
 ----| |----[ DMOV  0       D100     ]
                            配置码
                            1000
            [ DMOV  1000    D102     ]
                            初始速度
                            20000
            [ DMOV  20000   D104     ]
                            最高速度
                            200
            [ DMOV  200     D106     ]
                            加减速时间
                            300000
            [ DMOV  300000  D108     ]
                            X轴移动距
                            离
                            400000
            [ DMOV  400000  D110     ]
                            Y轴移动距
                            离
                    0       OFF      OFF      OFF      OFF
N1   M0
 ----| |----[ LIN   D100    Y0       Y1       Y2       Y3       ]
                    配置码  X轴脉冲  X轴方向  Y轴脉冲  Y轴方向
```

6.21.10 CW: Clockwise circular trace interpolation

LAD: CW (S) (D1) (D2) (D3) (D4)	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: CW (*S*) (*D1*) (*D2*) (*D3*) (*D4*)	**Program steps**	**12**

Operand	Type	Applicable elements													Indexed addressing
S	DINT								D						
D1	BOOL			Y											
D2	BOOL			Y											
D3	BOOL			Y											
D4	BOOL			Y											

■ **Operand description**

S: starting address of storage area for parameter list

D1: corresponding output point SN of X axis pulse signal (or positive pulse signal). Range: Y0, Y4, Y5, Y6, Y7

D2: corresponding output point SN of X axis direction signal (or negative pulse signal)

D3: corresponding output point SN of Y axis pulse signal (or positive pulse signal). Range: Y2, Y4, Y5, Y6, Y7

D4: corresponding output point SN of Y axis direction signal (or negative pulse signal)

■ **Function description**

1. Move to the target position in clockwise circular trace at designated linear speed.

2. Definitions of parameter list

<table>
<tr><th>D element</th><th>Content</th></tr>
<tr><td>S</td><td>Method to form circular arc
<table>
<tr><th>Configuration</th><th>Mode</th></tr>
<tr><td>0</td><td>Designate center of a circle</td></tr>
<tr><td>1</td><td>Designate passed position</td></tr>
</table></td></tr>
<tr><td>S+1</td><td>Relation between action mode and output logic (configuration code in decimal)
<table>
<tr><th>Configuration</th><th>Mode</th><th>Output logic</th></tr>
<tr><td>00</td><td>Incremental</td><td>Pulse+direction (positive ON/ negative OFF)</td></tr>
<tr><td>01</td><td>Incremental</td><td>Positive pulse +negative pulse</td></tr>
<tr><td>10</td><td>Absolute</td><td>Pulse+direction (positive ON/ negative OFF)</td></tr>
<tr><td>11</td><td>Absolute</td><td>Positive pulse +negative pulse</td></tr>
</table></td></tr>
<tr><td>S+2</td><td>(Reserved)</td></tr>
<tr><td>S+3</td><td>(Reserved)</td></tr>
<tr><td>S+4</td><td>Resultant speed (MSB)</td></tr>
<tr><td>S+5</td><td>Resultant speed (LSB)</td></tr>
<tr><td>S+6</td><td>(Reserved)</td></tr>
<tr><td>S+7</td><td>(Reserved)</td></tr>
<tr><td>S+8</td><td>X axis move distance (target position) (MSB)</td></tr>
<tr><td>S+9</td><td>X axis move distance (target position) (LSB)</td></tr>
<tr><td>S+10</td><td>Y axis move distance (target position) (MSB)</td></tr>
<tr><td>S+11</td><td>Y axis move distance (target position) (LSB)</td></tr>
<tr><td>S+12</td><td>Center of a circle of X axis/
X coordinate of passed point (MSB)</td></tr>
<tr><td>S+13</td><td>Center of a circle of X axis/
X coordinate of passed point (LSB)</td></tr>
<tr><td>S+14</td><td>Center of a circle of Y axis/
Y coordinate of passed point (MSB)</td></tr>
<tr><td>S+15</td><td>Center of a circle of Y axis/
Y coordinate of passed point (LSB)</td></tr>
</table>

(1) In the incremental mode, the target trace adopts relative address, referring to the move distance of X and Y axes from the current position to the target.

(2) In the absolute mode, the target trace adopts absolute address, referring to the absolute coordinate of the target position on X and Y axes.

■ **Note**

1. D1 and D3 must be used in group. When the output group is Y0 and Y2, Y1 and Y3 can be direction signal or negative pulse output signal to match with Y0 and Y2 separately. D1 and D3 can select any two of Y4, Y5, Y6, Y7 for interpolation, and the corresponding direction signals are free to select.

2. Output group (Y0, Y2) can be designated to "pulse+direction" mode or "positive pulse+negative pulse" mode, the Max. speed 100k; Y4, Y5, Y6, Y7 for interpolation are output in "pulse+direction" mode, the Max. speed 20k.

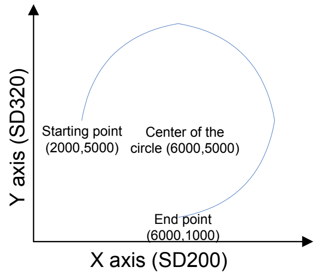
3. The move distance of each axis should not exceed 16,777,215 pulses at a time.

4. Note that only one of PLSY, PLS or locating instructions can be used for one high-speed port at one time.

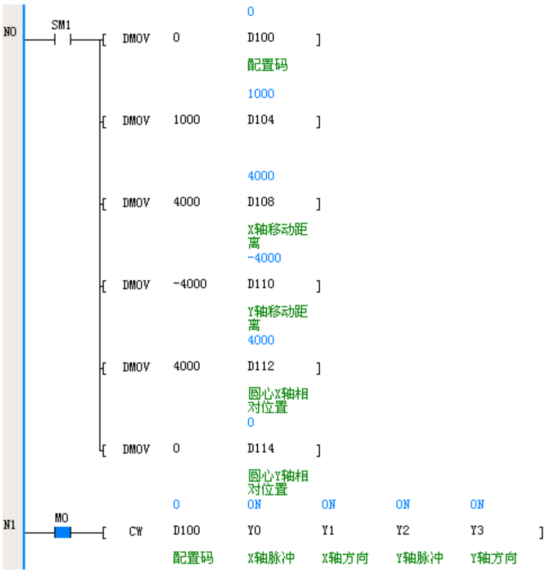
5. The coordinate of passed point designated in passed position mode refers to the position the whole circle passes while the path of user-defined trace may not pass.

■ **Example**

If the displacement of the current SD element is (2000, 5000), the circular arc will be drawn as follows:



The incremental mode can be used. The displacement of end point relative to starting point is (4000, -4000) and the displacement of center of the circle relative to starting point is (4000, 0). The programming can be:



```
                              0
N0    SM1
   ───┤ ├───┬[ DMOV    0      D100       ]
            │                 配置码
            │                 1000
            ├[ DMOV    1000   D104       ]

            │                 4000
            ├[ DMOV    4000   D108       ]
            │                 X轴移动距
            │                 离
            │                 -4000
            ├[ DMOV    -4000  D110       ]
            │                 Y轴移动距
            │                 离
            │                 4000
            ├[ DMOV    4000   D112       ]
            │                 圆心X轴相
            │                 对位置
            │                 0
            └[ DMOV    0      D114       ]
                              圆心Y轴相
                              对位置
                       0      ON       ON       ON       ON
N1    M0
   ───█───[ CW      D100   Y0       Y1       Y2       Y3       ]
                       配置码  X轴脉冲  X轴方向  Y轴脉冲  Y轴方向
```

6.21.11 CCW: Counterclockwise circular trace interpolation

LAD: ┤├───[CCW (S) (D1) (D2) (D3) (D4)]		**Applicable to**	**EC20H**
		Influenced flag bit	**Zero, carry, borrow**
IL: CCW (*S*) (*D1*) (*D2*) (*D3*) (*D4*)		**Program steps**	**12**

Operand	Type	Applicable elements													Indexed addressing
S	DINT								D						
D1	BOOL			Y											
D2	BOOL			Y											
D3	BOOL			Y											
D4	BOOL			Y											

■ Operand description

S: starting address of storage area for parameter list

D1: corresponding output point SN of X axis pulse signal (or positive pulse signal). Range: Y0, Y4, Y5, Y6, Y7

D2: corresponding output point SN of X axis direction signal (or negative pulse signal)

D3: corresponding output point SN of Y axis pulse signal (or positive pulse signal). Range: Y2, Y4, Y5, Y6, Y7

D4: corresponding output point SN of Y axis direction signal (or negative pulse signal)

■ Function description

1. Move to the target position in counterclockwise circular trace at designated linear speed.

2. Definitions of parameter list

D element	Content

<table>
<tr><td>S</td><td>Method to form circular arc<table><tr><th>Configuration</th><th>Mode</th></tr><tr><td>0</td><td>Designate center of a circle</td></tr><tr><td>1</td><td>Designate passed position</td></tr></table></td></tr>
<tr><td>S+1</td><td>Relation between action mode and output logic (configuration code in decimal)<table><tr><th>Configuration</th><th>Mode</th><th>Output logic</th></tr><tr><td>00</td><td>Incremental</td><td>Pulse+direction (positive ON/ negative OFF)</td></tr><tr><td>01</td><td>Incremental</td><td>Positive pulse +negative pulse</td></tr><tr><td>10</td><td>Absolute</td><td>Pulse+direction (positive ON/ negative OFF)</td></tr><tr><td>11</td><td>Absolute</td><td>Positive pulse +negative pulse</td></tr></table></td></tr>
<tr><td>S+2</td><td>(Reserved)</td></tr>
<tr><td>S+3</td><td>(Reserved)</td></tr>
<tr><td>S+4</td><td>Resultant speed (MSB)</td></tr>
<tr><td>S+5</td><td>Resultant speed (LSB)</td></tr>
<tr><td>S+6</td><td>(Reserved)</td></tr>
<tr><td>S+7</td><td>(Reserved)</td></tr>
<tr><td>S+8</td><td>X axis move distance (target position) (MSB)</td></tr>
<tr><td>S+9</td><td>X axis move distance (target position) (LSB)</td></tr>
<tr><td>S+10</td><td>Y axis move distance (target position) (MSB)</td></tr>
<tr><td>S+11</td><td>Y axis move distance (target position) (LSB)</td></tr>
<tr><td>S+12</td><td>Center of a circle of X axis/
X coordinate of passed point (MSB)</td></tr>
<tr><td>S+13</td><td>Center of a circle of X axis/
X coordinate of passed point (LSB)</td></tr>
<tr><td>S+14</td><td>Center of a circle of Y axis/
Y coordinate of passed point (MSB)</td></tr>
<tr><td>S+15</td><td>Center of a circle of Y axis/
Y coordinate of passed point (LSB)</td></tr>
</table>

(1) In the incremental mode, the target trace adopts relative address, referring to the move distance of X and Y axes from the current position to the target.

(2) In the absolute mode, the target trace adopts absolute address, referring to the absolute coordinate of the target position on X and Y axes.

■ Note

1. D1 and D3 must be used in group. When the output group is Y0 and Y2, Y1 and Y3 can be direction signal or negative pulse output signal to match with Y0 and Y2 separately. D1 and D3 can select any two of Y4, Y5, Y6, Y7 for interpolation, and the corresponding direction signals are free to select.

2. Output group (Y0, Y2) can be designated to “pulse+direction” mode or “positive pulse+negative pulse” mode, the Max. speed 100k; Y4, Y5, Y6, Y7 for interpolation are output in “pulse+direction” mode, the Max. speed 20k.

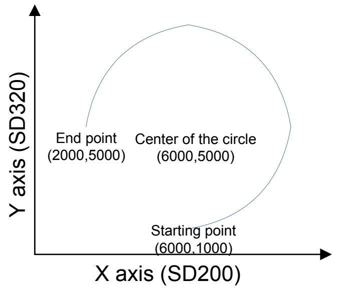
3. The move distance of each axis should not exceed 16,777,215 pulses at a time.

4. Note that only one of PLSY, PLS or locating instructions can be used for one high-speed port at one time.

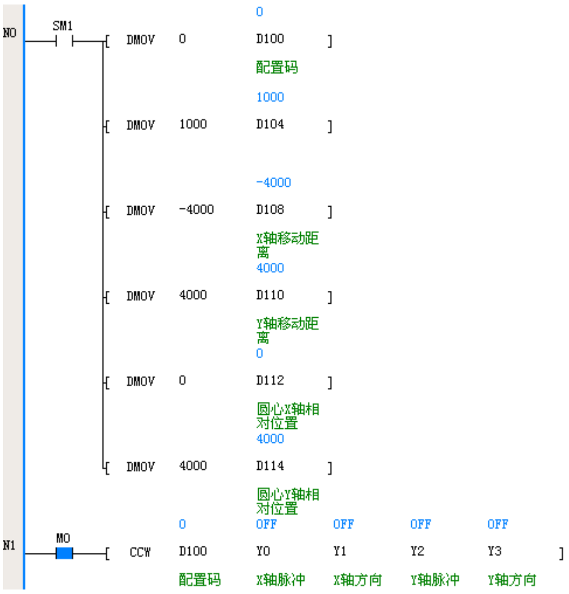
5. The coordinate of passed point designated in passed position mode refers to the position the whole circle passes while the path of user-defined trace may not pass.

■ Example

If the displacement of the current SD element is (6000, 1000), the circular arc will be drawn as follows:



The incremental mode can be used. The displacement of end point relative to starting point is (-4000, 4000) and the displacement of center of the circle relative to starting point is (0, 4000). The programming can be:



```
                         0
N0    SM1
   ──┤ ├──┬[ DMOV   0       D100    ]
          │                 配置码
          │                 1000
          ├[ DMOV   1000    D104    ]

          │                 -4000
          ├[ DMOV   -4000   D108    ]
          │                 X轴移动距
          │                 离
          │                 4000
          ├[ DMOV   4000    D110    ]
          │                 Y轴移动距
          │                 离
          │                 0
          ├[ DMOV   0       D112    ]
          │                 圆心X轴相
          │                 对位置
          │                 4000
          └[ DMOV   4000    D114    ]
                            圆心Y轴相
                            对位置
                    0       OFF      OFF      OFF      OFF
N1    M0
   ──┤ ├──[ CCW     D100    Y0       Y1       Y2       Y3      ]
                    配置码  X轴脉冲  X轴方向  Y轴脉冲  Y轴方向
```

6.21.12 MOVELINK: Synchronous control instruction

LAD: ┤├──[MOVELINK *(S1) (S2) (S3) (S4) (S5) (S6)*]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: MOVELINK(*S1*)(*S2*)(*S3*)(*S4*)(*S5*)(*S6*)	**Program steps**	**17**

Operand	Type	Applicable elements													Indexed addressing
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
S3	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		
S4	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		
S5	BOOL			Y											
S6	BOOL			Y							C				

■ **Operand description**

S1: total pulses of secondary axis in the following process

S2: total pulses of primary axis in the followed process

S3: pulses of primary axis before reaching constant speed synchronization

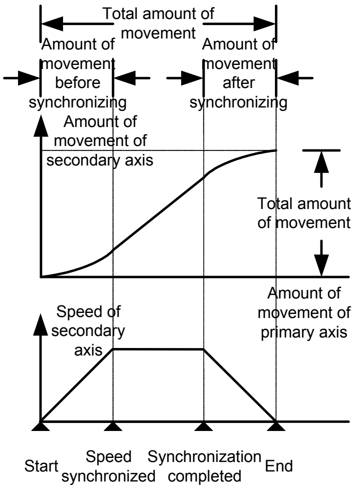
S4: pulses of primary axis after completing constant speed synchronization

S5: corresponding output point SN of secondary axis pulse signal. Range: Y4, Y5

S6: corresponding output point SN or high-speed counter SN of primary axis pulse signal. Range: Y0~Y7, C236~C255, C301~C306

■ **Function description**

1. The instruction realizes simple synchronous functions of two axes. The secondary axis measures and follows the speed and position of the primary axis. The primary axis can be high-speed output port of the module or high-speed input port.


Total amount of movement
Amount of movement before synchronizing
Amount of movement after synchronizing
Amount of movement of secondary axis
Total amount of movement
Amount of movement of primary axis
Speed of secondary axis
Start
Speed synchronized
Synchronization completed
End

2. As shown above, after the instruction starts, the secondary axis begins from static state and follows the primary axis to run after ACC and DEC process. When the primary axis outputs S3, two axes will be consistent in speed and start constant speed synchronization. When the pulses of the primary axis from the target position is S4, constant speed synchronization will be over and the secondary axis will start decelerating. When the primary axis arrives at the target position, the secondary axis will stop.

3. When the primary axis is defined as output axis, S6 will be Y0~Y7; when it is defined as input axis, S6 is C236~C255 or C301~C306.

4. When the power flow is ON, the secondary axis will run. The instruction supports calling methods of main programs, subprograms and interrupt programs.

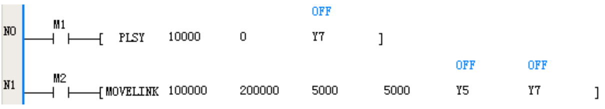
5. The direction signal of output axis does not need designation and its ON/OFF state will not change in the following process.

■ **Note**

1. Note that only one of PLSY, PLS or locating instructions can be used for one high-speed port at one time.

2. The output point SN D corresponding to high-speed pulse can be: Y0, Y2, Y4, Y5, Y6, Y7. Y1 and Y3 can be direction signal or negative pulse output signal to match with Y0 and Y2 separately.

■ **Example**


```
                                      OFF
      M1
N0 ──┤ ├──[  PLSY     10000     0      Y7          ]
                                                         OFF        OFF
      M2
N1 ──┤ ├──[MOVELINK  100000   200000   5000    5000    Y5         Y7        ]
```

6.21.13 GEARBOX: Electronic gear instruction

LAD:	Applicable to	EC20H
─┤├─[GEARBOX (D1) (S1) (D2) (S2)]	Influenced flag bit	Zero, carry, borrow
IL: GEARBOX (D1) (S1) (D2) (S2)	Program steps	9

Operand	Type	Applicable elements													Indexed addressing
D1	BOOL			Y											
S1	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√
D2	BOOL			Y											
S2	DINT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C		V		√

- **Operand description**

D1: corresponding output point SN of primary axis pulse signal. Range: Y0~Y7

S1: electronic gear ratio determined by the operand and S2

D2: corresponding output point SN of secondary axis pulse signal. Range: Y4, Y5

S2: electronic gear ratio determined by the operand and S1

Range of electronic gear ratio (***S1***/***S2***): 1/10000~10000

- **Function description**

1. The secondary axis follows the primary axis by electronic gear ratio (S1/S2). When the primary axis sends N pulses, the secondary axis will send N×S1/S2 pulses; when the primary axis outputs F pulses, the secondary axis will output F×S1/S2 pulses;

2. When the power flow is ON, the secondary axis will run.

3. The direction signal of output axis does not need designation and its ON/OFF state will not change in the following process.

4. When the electronic gear ratio<0, namely, either S1 or S2 is negative or positive, SD element of the secondary axis will decrease along with pulse output.

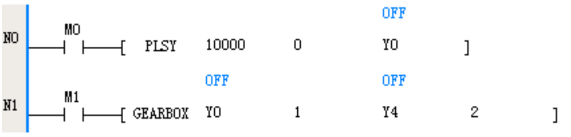
- **Note**

1. Note that only one of PLSY, PLS or locating instructions can be used for one high-speed port at one time.

2. If the electronic gear ratio exceeds the designated range, there will be no pulse output.

3. The output point SN D corresponding to high-speed pulse can be: Y0, Y2, Y4, Y5, Y6, Y7. Y1 and Y3 can be direction signal or negative pulse output signal to match with Y0 and Y2 separately.

- **Example**



```
                                          OFF
N0   M0
   ──┤ ├──[ PLSY     10000     0          Y0         ]
                     OFF                  OFF
N1   M1
   ──┤ ├──[ GEARBOX  Y0        1          Y4     2        ]
```

6.22 Data processing instruction

6.22.1 MEAN: Mean instruction

LAD: |—| |—[MEAN (S1) (D) (S2)]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: MEAN (*S1*) (*D*) (*S2*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT	Constant	KnX	KnY	KnM	KnS	KnLM	KnSM	D	SD	C	T		R	√
S2	INT	Constant							D					R	√
D	INT			KnY	KnM	KnS	KnLM		D	SD	C			R	√

■ **Operand description**

S1: starting element SN to store the required mean values

S2: mean values (1~64)

D: word element SN to store the achieved mean values

■ **Function description**

1. Store S2 16bit mean values starting with S1 to D, and round down remainder.

■ **Example**

M1
32
10
MEAN
D0
D10
4

LD M1

MEAN D0 D10 4

When M1=ON, calculate the mean value of 4 units starting with D0 and store the result to D10. When D0=32, D1=10, D2=15 and D3=-14, D10=10.

6.22.2 WTOB: Data separation instruction for byte unit

LAD: |—| |—[WTOB (S1) (D) (S2)]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: WTOB (*S1*) (*D*) (*S2*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D	SD	C	T		R	√
S2	INT								D	SD	C	T		R	√
D	INT	Constant							D					R	√

■ **Operand description**

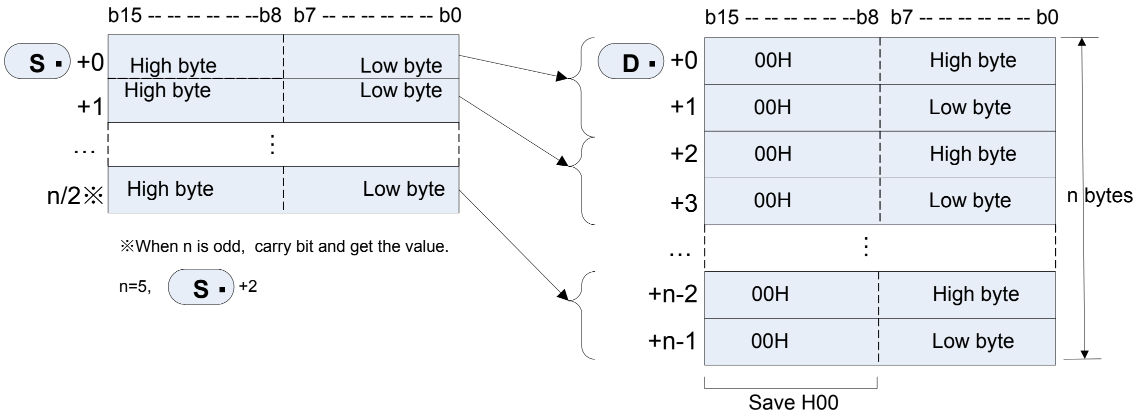
S1: starting element SN to store the data which will be separated in byte unit

S2: data number to be separated (S2≥0)

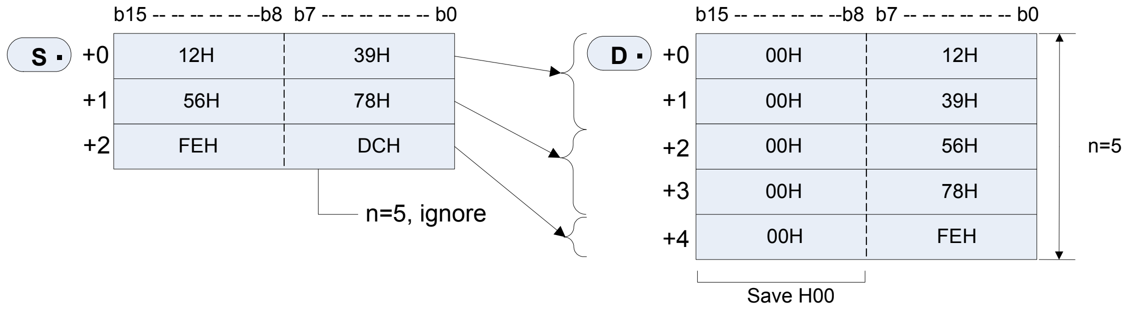
D: starting element SN to store the data which has been separated in byte unit

■ **Function description**

1. Separate the 16bit data stored in S2/S elements starting with S1 into S2 bytes, store the result into the low bytes of S2 elements starting with D and clear the high bytes.



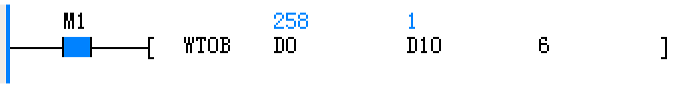
2. When S2 is odd, only the high bytes (8bit) in the last data of separation source are the object data. For example, n=5, the data of low bytes of S~S+2 will be stored in D~D+4.


b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
S · +0
12H
39H
+1
56H
78H
+2
FEH
DCH
n=5, ignore
b15 -- -- -- -- -- --b8 b7 -- -- -- -- -- -- b0
D · +0
00H
12H
+1
00H
39H
+2
00H
56H
+3
00H
78H
+4
00H
FEH
n=5
Save H00

3. S2=0, the instruction will not be executed.

4. Source operand and destination operand cannot be overlapped.

■ **Example**



```
 M1                 258       1
─┤├─────[ WTOB      D0        D10       6        ]
```

LD M1

WTOB D0 D10 6

When M1=ON, separate 3 unit data starting with D0 into 6 units according to high/low bytes, and store the result into 6 units starting with D10. When D0=0x102, D1=0x304 and D2=0x506, D10=0x01, D11=0x02, D12=0x03, D13=0x04, D14=0x05 and D15=0x06.

6.22.3 BTOW: Data combination instruction for byte unit

LAD: ├──┤ ├──[BTOW (S1) (D) (S2)]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: BTOW (*S1*) (*D*) (*S2*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D	SD	C	T		R	√
S2	INT								D	SD	C	T		R	√
D	INT	Constant							D					R	√

Operand description

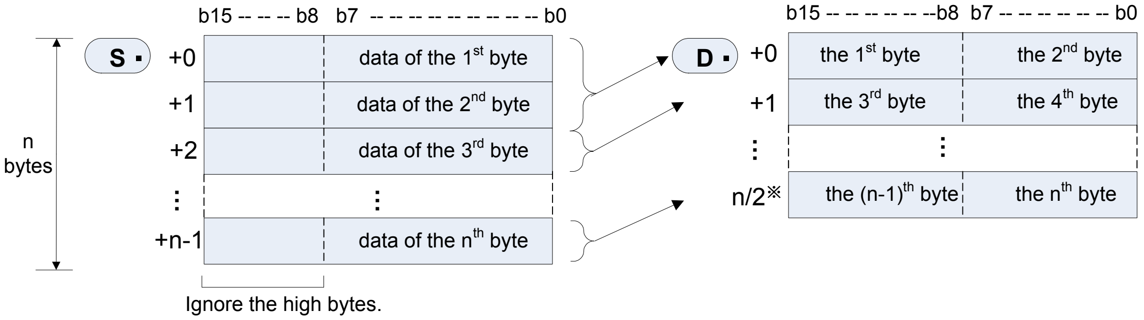
S1: starting element SN to store the data which will be combined in byte unit

S2: data number to be combined (S2≥0)

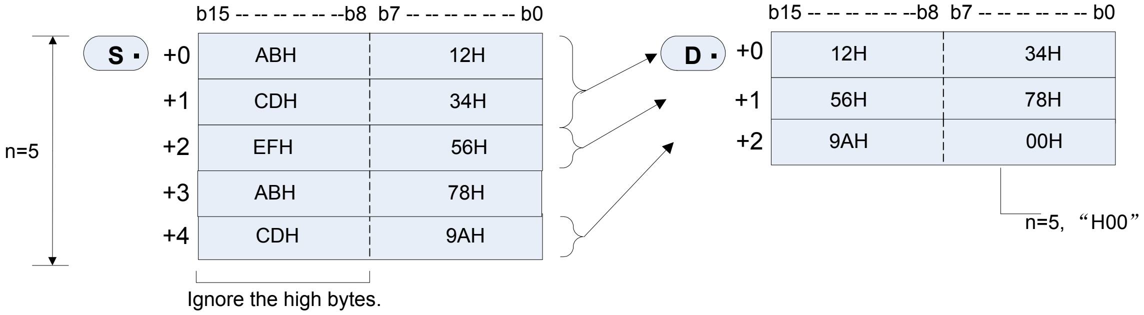
D: starting element SN to store the data which has been combined in byte unit

Function description

1. Store the data after the combination of low bytes (8bit) of S2 16bit data starting with S1 into S2/2 elements starting with D. Ignore (8bit) high bytes of 16bit data (after S1) of combination source.



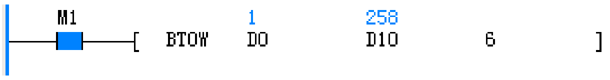
2. When S2 is odd, the low bytes combined at last will be cleared.



3. S2=0, the instruction will not be executed.

4. Source operand and destination operand cannot be overlapped.

Example



LD M1

BTOW D0 D10 6

When M1=ON, store 3 unit data after the combination of 6 unit data starting with D0 into 3 units starting with D10. When D0=0x01, D1=0x02, D2=0x03, D3=0x04, D4=0x05 and D5=0x06, D10=0x102, D11=0x304 and D12=0x506.

6.22.4 UNI: 4bit combination instruction for 16bit data

LAD: ├──┤ ├──[UNI (S1) (D) (S2)]	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: UNI (*S1*) (*D*) (*S2*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D	SD	C	T		R	√
S2	INT								D	SD	C	T		R	√
D	INT	Constant							D					R	√

■ **Operand description**

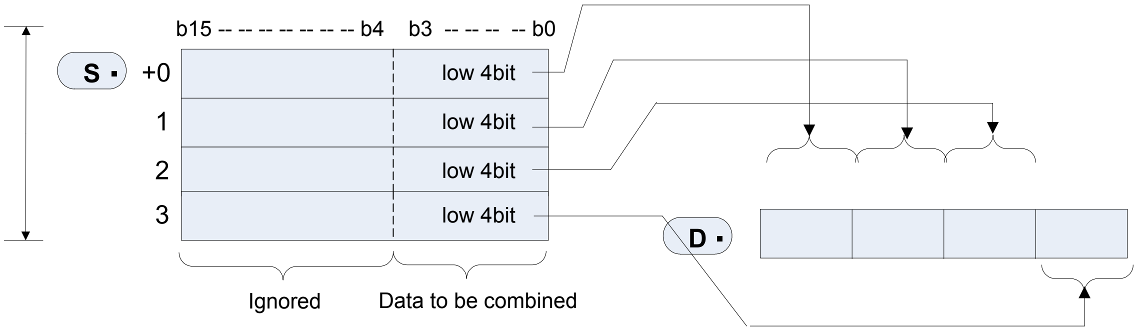
S1: starting element SN to store the data which will be combined

S2: combined data (0-4, S2=0, no processing)

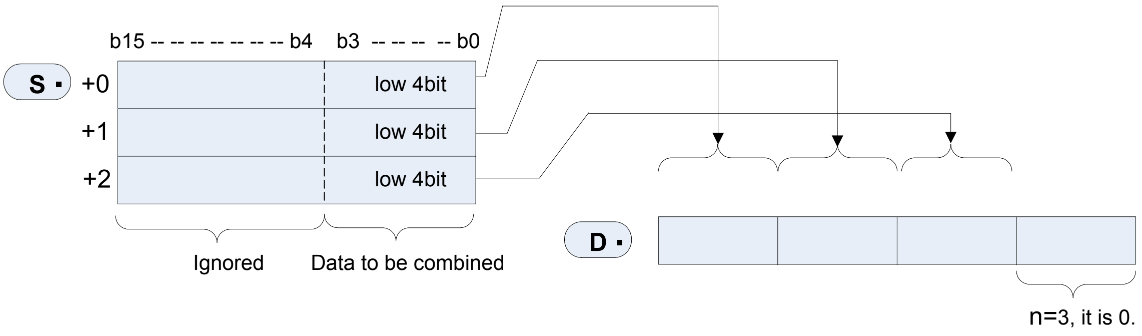
D: element SN to store the data which has been combined

■ **Function description**

1. Store S2 16bit data starting with S1 into S2 elements starting with D.



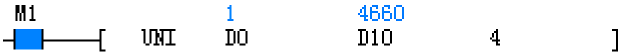
2. When S2 is 1~3, the unit of low bit {4×(4-S2)} in D is zero.



3. When the range of S2 is 1-4 and S2=0, the instruction will not be executed.

4. Source operand and destination operand cannot be overlapped.

■ **Example**



LD M1

UNI D0 D10 4

When M1=ON, combine low 4bit of 4 unit data starting with D0 and store the result into D10. When D0=0x01, D1=0x02, D2=0x03 and D3=0x04, D10=0x1234.

6.22.5 DIS: 4bit separation instruction for 16bit data

LAD:	**Applicable to**	**EC20H**
	Influenced flag bit	**Zero, carry, borrow**
IL: DIS (*S1*) (*D*) (*S2*)	**Program steps**	**7**

Operand	Type	Applicable elements													Indexed addressing
S1	INT								D	SD	C	T		R	√
S2	INT								D	SD	C	T		R	√
D	INT	Constant							D					R	√

■ **Operand description**

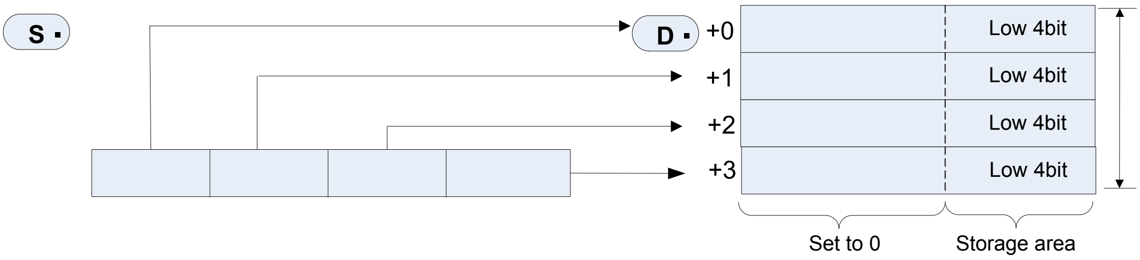
S1: starting element SN to store the data which will be separated

S2: separated data (0-4, S2=0, no processing)

D: element SN to store the data which has been separated

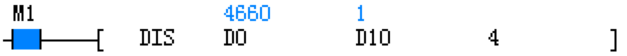
■ **Function description**

1. Store S2 16bit data starting with S1 into S2 elements starting with D.



2. The valid range of S2 is 1-4. The instruction will not be executed for other data.
3. The high 12bit in S2 elements starting with D is cleared.
4. Source operand and destination operand cannot be overlapped.

■ **Example**

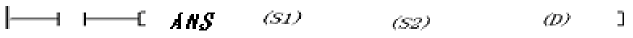


LD M1

DIS D0 D10 4

When M1=ON, separate D0 unit data every 4 bits, and store the result into 4 units starting with D10. When D0=0x1234, D10=0x01, D11=0x02, D12=0x03 and D13=0x04.

6.22.6 ANS: Signal alarm set instruction

LAD: 	Applicable to	EC20H
	Influenced flag bit	Zero, carry, borrow
IL: ANS (*S1*) (*S2*) (*D*)	Program steps	7

Operand	Type	Applicable elements													Indexed addressing
S1	INT											T			√
S2	INT	Constant							D					R	√
D	BOOL			S											√

Operand description

S1: timer SN to judge time, only applicable to 100ms timer, T0-T209

S2: data to judge time (1-32767)

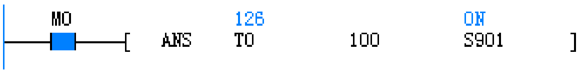
D: element to set signal alarm, S900-S999

Function description

1. When the hold time of power flow>S2, set D; when the hold time of power flow<S2, reset S1 and do not set D; when the power flow is invalid, reset S1.

Address No.	Name	Function
SM400	Signal alarm is valid	Set SM400 after ON, SM401 and SD401 works
SM401	Signal alarm acts	Any of S900-S999 acts, set SM401 after ON
SD401	Min. No. at On	Store the Min. No. in S900-S999

Example



LD M0

ANS T0 100 S901

When the power flow is valid, not disconnected in 10s, S901 will be set.

6.22.7 ANR: Signal alarm reset instruction

LAD:	Applicable to	EC20H
	Influenced flag bit	Zero, carry, borrow
IL: ANR	Program steps	1

Operand	Type	Applicable elements	Indexed addressing

- **Operand description**

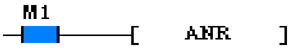
No operand

- **Function description**

1. When the power flow is valid, reset the running state of signal alarm S900-S999; if there are multiple state actions, reset the one with the Min. No., when the power flow becomes valid again, reset the next one with the Min. No..

Address No.	Name	Function
SM400	Signal alarm is valid	Set SM400 after ON, SM401 and SD401 works
SM401	Signal alarm acts	Any of S900-S999 acts, set SM401 after ON
SD401	Min. No. at On	Store the Min. No. in S900-S999

- **Example**



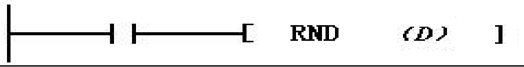
LD M1

ANR

When the power flow is valid, if there are multiple S set by ANS, reset the one with the Min. No..

6.23 Other instructions

6.23.1 RND: Generate random number instruction

<table>
<tr><td colspan="10">LAD:
 </td><td colspan="3">Applicable to</td><td colspan="4">EC20H</td></tr>
<tr><td colspan="10"></td><td colspan="3">Influenced flag bit</td><td colspan="4">Zero flag</td></tr>
<tr><td colspan="10">IL: RND <i>(D)</i></td><td colspan="3">Program steps</td><td colspan="4">3</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="14">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>D</td><td>WORD</td><td></td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td></td><td>Z</td><td>R</td><td>√</td></tr>
</table>

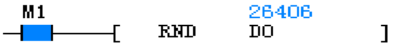
■ **Operand description**

D: starting element SN to store random number

■ **Function description**

1. Generate 0-32767 random number and store the value into D; if the random number is 0, set zero flag SM180.

■ **Example**

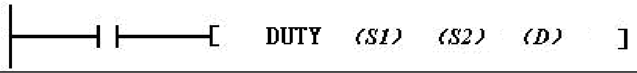


LD X0

RND D0

When M1=ON, generate the random number and store it into D0, D0=26406.

6.23.2 DUTY: Generate timing pulse instruction

<table>
<tr><td colspan="11">LAD:
 </td><td colspan="3">Applicable to</td><td colspan="4">EC20H</td></tr>
<tr><td colspan="11"></td><td colspan="3">Influenced flag bit</td><td colspan="4"></td></tr>
<tr><td colspan="11">IL: DUTY <i>(S1)</i> <i>(S2)</i> <i>(D)</i></td><td colspan="3">Program steps</td><td colspan="4">7</td></tr>
<tr><td>Operand</td><td>Type</td><td colspan="15">Applicable elements</td><td>Indexed addressing</td></tr>
<tr><td>S1</td><td>WORD</td><td>Constant</td><td></td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
<tr><td>S2</td><td>INT</td><td>Constant</td><td></td><td>KnX</td><td>KnY</td><td>KnM</td><td>KnS</td><td>KnLM</td><td>KnSM</td><td>D</td><td>SD</td><td>C</td><td>T</td><td>V</td><td>Z</td><td>R</td><td>√</td></tr>
<tr><td>D</td><td>BOOL</td><td>SM</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

■ **Operand description**

S1: scan times of ON

S2: scan times of OFF

D: always target address of timing output

■ **Function description**

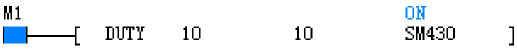
1. The timing pulse output unit D scans ON S1 times and OFF S2 times;

2. SM unit, SM430-SM434

Target address of timing output	Element for counting scan times
SM430	SD430
SM431	SD431
SM432	SD432
SM433	SD433
SM434	SD434

3. The instruction can be used 5 times while multiple DUTY instructions cannot use the same target address of timing output.

■ **Example**



When M1=ON, there will be 10 scan times for ON and 10 scan times for OFF in SM430. Simultaneously, store the count value of scan times in SD430.

■ **Note**

Act when the instruction is at rising edge, continue even if the power flow is OFF, and stop at STOP or power off.

Chapter 7 SFC tutor

This chapter introduces the basic concepts and programming methods of Sequential Function Chart (SFC). In addition, the points to note during the programming is also introduced.

7.1 Introduction to SFC

7.1.1 What is SFC

The Sequential Function Chart, or SFC, is a programming language developed and got popular in recent years. SFC can turn a PLC program into a structured flow chart. By using standard programming symbols and grammar compliant with IEC61131-3, the SFC can divide a complicated operation process into sequential procedures that are linked together with conditioned transfers, so as to realize sequence control.

The SFC edited programs are direct and sequential. Each procedure and transfer condition are relatively simple program sections, ideal for the sequential control application. These advantages explain why it is finding wider application.

7.1.2 What is SFC of EC series PLC

The SFC of EC series PLC is a programming language used by COMPANY EC series PLCs. Besides standard SFC functions, the SFC of EC series PLC can provide multiple nested LAD program blocks.

The program edited with SFC of EC series PLC can be converted into LAD or IL program.

The SFC of EC series PLC can also support up to 20 independent procedures. The independent procedures can run independently, that is to say, the steps within different independent procedures are scanned and executed separately. However, jumping among independent procedures is enabled.

7.1.3 Basic concepts of SFC

The SFC has the following two basic concepts: step and transfer. Other concepts, like jump, branch and multiple independent procedures, all evolve from the two basic concepts.

- **Steps**

1. Definition

A step is actually a program section, representing a work state or move in the sequence control process. Putting multiple steps together in a organic way can form a complete SFC program.

2. Execution of steps

In a SFC program, each step is represented by a fixed S element.

A step is valid when it is being executed. For a valid step, its corresponding S element is ON, and the PLC will scan and execute its instructions. While a step not being executed is invalid. For a invalid step, its corresponding S element is OFF, and the PLC will not scan and execute its instructions.

- **Transfer**

The sequence control process is actually a series of step transfers. A PLC executing a certain step will, if certain logic conditions are met, leave the current step to enter and execute a new step. That transition is called the step transfer.

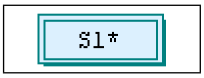
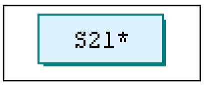
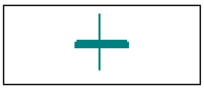
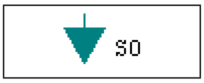
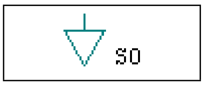
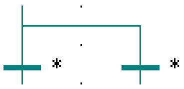
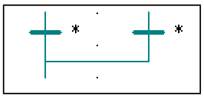
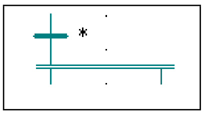
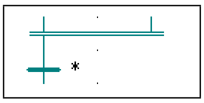
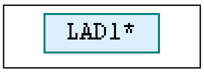
The logic condition that triggers the transfer is called the transfer condition.

7.1.4 Programming symbols and their usage

- **Programming symbol**

The EC series PLC SFC programming language consists of the following programming symbols:

Table 7-1 Programming symbols

Symbol name	Symbol	Description
Initial step		A initial step of SFC, numbered as Sn. The “n” must not repeat. The execution of a SFC program must start with an initial step, whose S element range is S0~S19
Normal step		A normal step, numbered as Sn. The “n” must not repeat. The S element range for the normal step is S20--S991
Transfer		A transfer. It can be built-in with a transfer condition (embeded LAD). You can compile the transfer condition so that the S element connected with this transfer will be set when the condition is met and enter the next step. The transfer must be used between steps.
Jump		A jump, used after the transfer. It can set the specified S element to ON when the transfer conditions are met. It is used to cycle or jump among steps
Reset		A reset, used after the transfer. It can set the specified S element to OFF when the transfer conditions are met. It is used to end the SFC program
Selection branch		Multiple independent transfer conditions, used after a step. When the transfer condition of one branch is met, the last step will end and the next step of the corresponding branch will start. After that, no other parallel branch will be selected
Selection merge		A merge of selection branches. When the transfer condition of one branch is met, the last step will end and the next step will start
Parallel branch		Connected after a step, the parallel branches share the same transfer conditions. When the transfer conditions are met, the parallel branches are validated and executed at the same time
Parallel merge		A merge of parallel branches. The next step will start only after all the parallel steps are finished and the transfer conditions are met
Ladder chart block		The LAD block presents LAD instructions for operations besides the SFC flow, such as starting the initial step and other general operations

- **Usage of programming symbols**

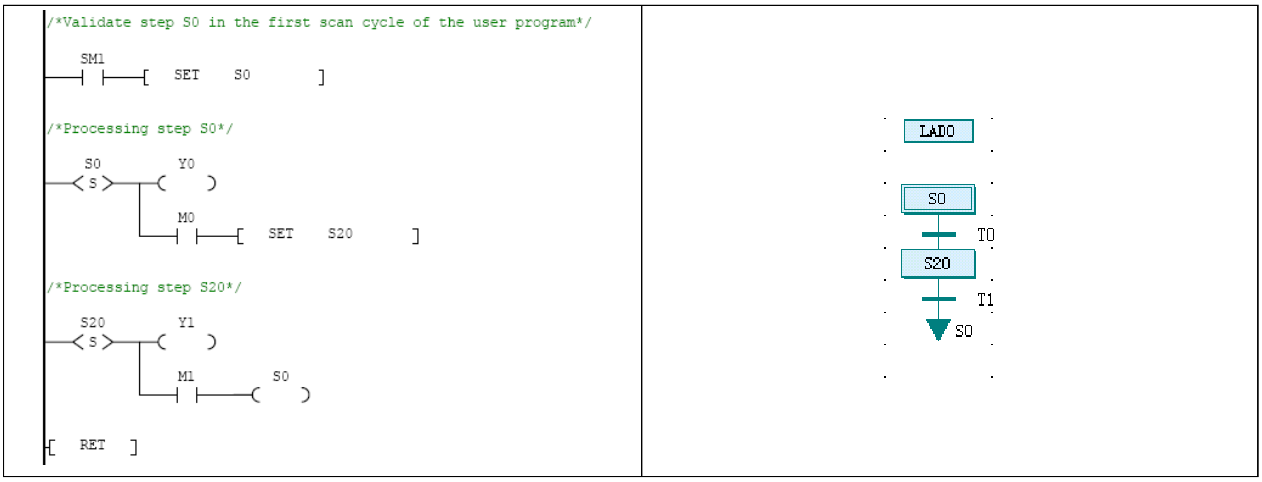
1. The initial step can be used alone. If you connect it with other symbols, you must use it at the start of you SFC program, and use a transfer condition symbol after it.
2. However, you cannot connet the LAD step with other symbols.
3. You must connect an normal stepwith transfer condition symbols, for the ordinary steps cannot be used alone.
4. The reset and jump should both be preceded by a transfer and followed by nothing.
5. Neither the transfer nor the jump can exist alone in a program.

7.1.5 SFC program structure

The structure of a SFC program is classified into three types: simple sequential structure, selection branch structure and parallel structure. Besides, the jump structure is also a special form of the selection branch structure.

- **Simple sequential structure**

Error! Reference source not found. shows a simple structured SFC program and its LAD counterpart.



In a simple structured SFC program, when the step transfer conditions are met, the program will run from the current step to the next step in a linear flow. At the last step, when the transfer conditions are met, the SFC program section will exit end or transfer to the initial step.

1. Ladder chart block

The ladder chart block is used to start SFC program section. To be specific, to set the S element of the initial step to ON. In the preceding figure, the program uses the power-on startup mode.

The ladder chart block can also be used as other general program sections besides the SFC program.

2. Initial step

As shown in **Error! Reference source not found.**, the initial step is started by a ladder chart block. The range of S elements for initial-step is 0~19.

3. Normal step

The normal step is the main component of the program. The range of S elements for normal-step is 20~991 (for EC20 series) or 20~1023 (for EC10 series).

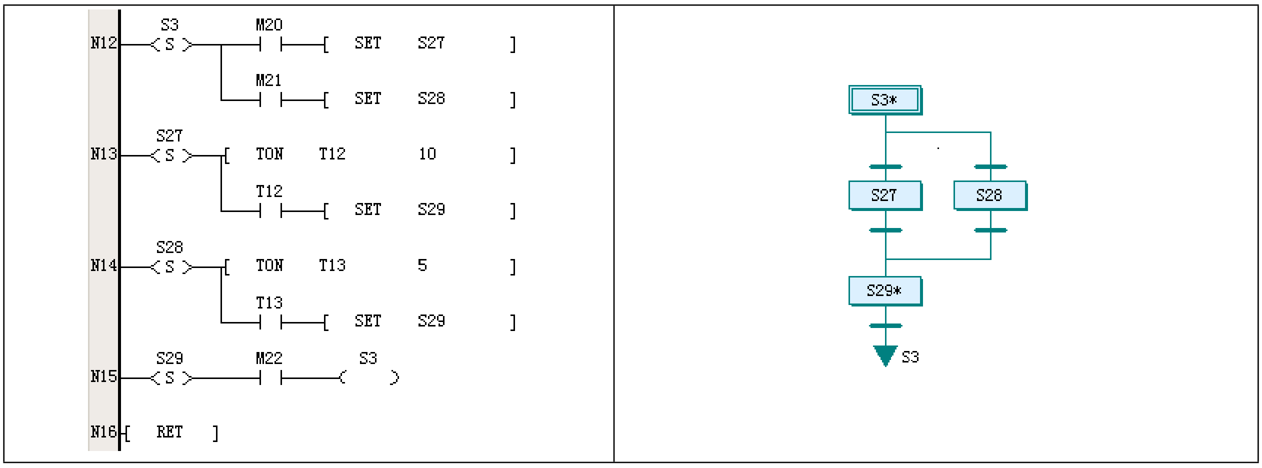
4. Transfer or reset

The program shown in **Error! Reference source not found.** is ended with a jump, which leads the program to the initial step. This is a cyclic program.

However, the program can also be ended with a reset, which can reset the status of the last step, end a program, and wait for the next round of execution.

- **Selection branch structure**

The selection branch structure is shown in the following figure, with LAD on the left and SFC on the right.



1. Selection branch

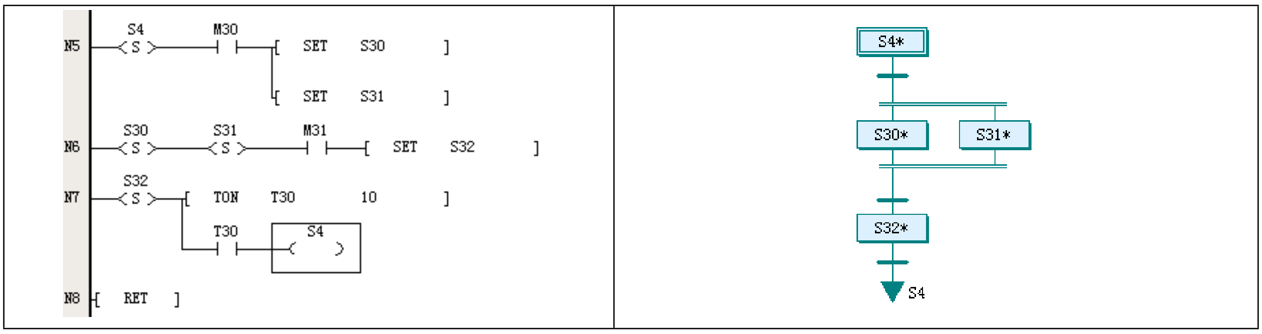
A branch step is validated when its corresponding transfer conditions are met. You must ensure that the transfer conditions of different branches are all exclusive, so as to make sure that each time only one branch will be selected. As shown in the preceding figure, steps S27 and S28 in row N12 of LAD program are transferred from conditions M20 and M21 respectively. The conditions M20 and M21 must not be met at the same time in order to ensure that S27 and S28 will not be selected at the same time.

2. Selection merge

The selection merge is the structure where all selection branches merge to the same step. The transfer conditions are set respectively. As shown in the preceding figure, the transfer condition in the branch of S27 is that time is up for T12, while that for the branch of S28 is that time is up for T13. However, the results are the same: step S29 starts.

- **Parallel branch structure**

The parallel branch structure is shown in the following figure, with LAD on the left and SFC on the right.



1. Parallel branch

When the transfer conditions are met for the parallel branches, all branch steps will be validated at the same time. This enables the PLC to process multiple procedures at the same time, a quite usual sequential control process. As shown in the preceding figure, in program row N5, the steps S30 and S31 will be validated at the same time when condition M30 is met.

2. Parallel merge

The parallel merge is the structure where all parallel branches merge to the same step by invalidating all branch steps at the same time. As shown in the preceding figure, in program row N6, when the program is running both S30 and S31 at the same time, if condition M31 is met, the program will start S32 and end S30&S31.

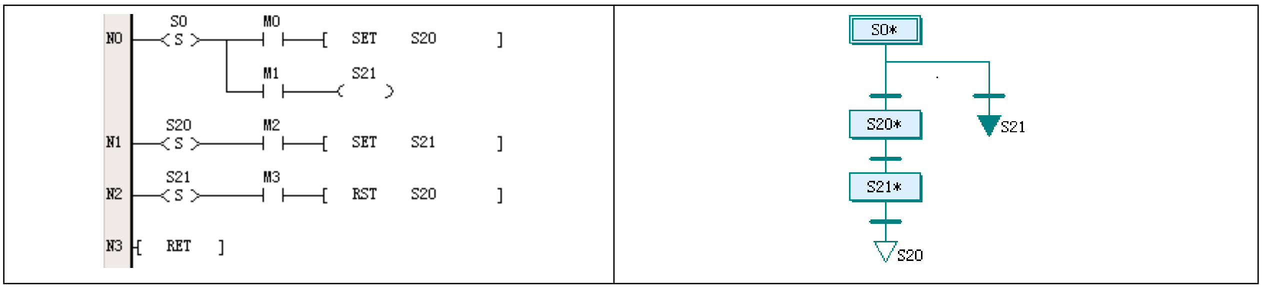
The sequential control behind the parallel merge structure is that no next step can be executed unless all the parallel steps are finished.

- **Jump structure**

The applications of jumps include: to omit certain steps, to recycle by returning to the initial step or a normal step, and to jump to another independent procedure.

1. Omitting certain steps

In a procedure, when certain steps are unnecessary under certain conditions, the program can jump directly to the needed step and omit the unnecessary steps, as shown in the following figure, with LAD on and left and SFC on the right.



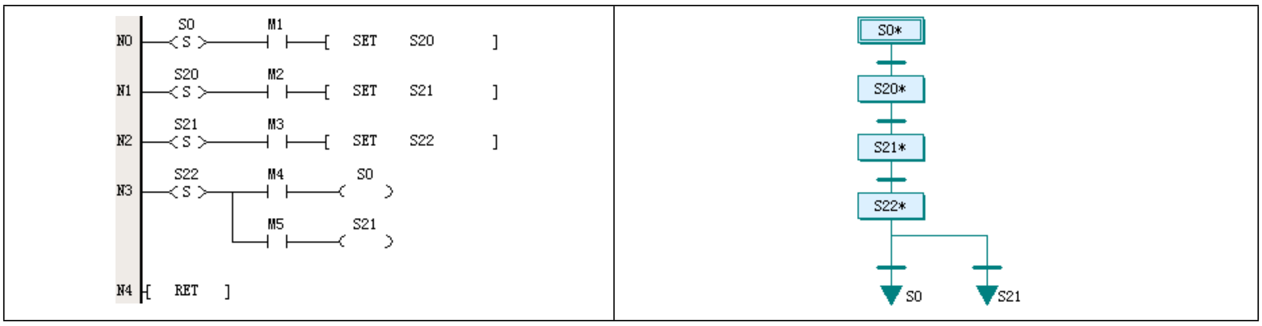
In the SFC program shown in the preceding figure, S21 is used as the jump, while step S20 is omitted. The jump is actually a selection branch.

While in the LAD counterpart, the second branch in row N0 is the jump instruction, which uses the OUT coil instead of the SET instruction in the transfer. When step S0 is valid, and if M1 is ON, the program will jump to step S21.

2. Recycling

In a procedure, when it is necessary to recycle a part or all of the steps under certain conditions, you can use the jump function. you can recycle a part of the steps if you jump to a previous normal step, or all the step if you jump to the initial step.

Shown in the following figure is a program that can realize the above two recycles, with LAD on the left and SFC on the right.



In the SFC, when step S22 is valid, the program may jump to step S21 to recycle S21 and S22, or jump to the initial step S0 to recycle all the steps. Which recycle will be selected is determined by a selection branch structure.

While in the LAD, the two kinds of jumps are realized in row N3, where you can see the OUT coil.

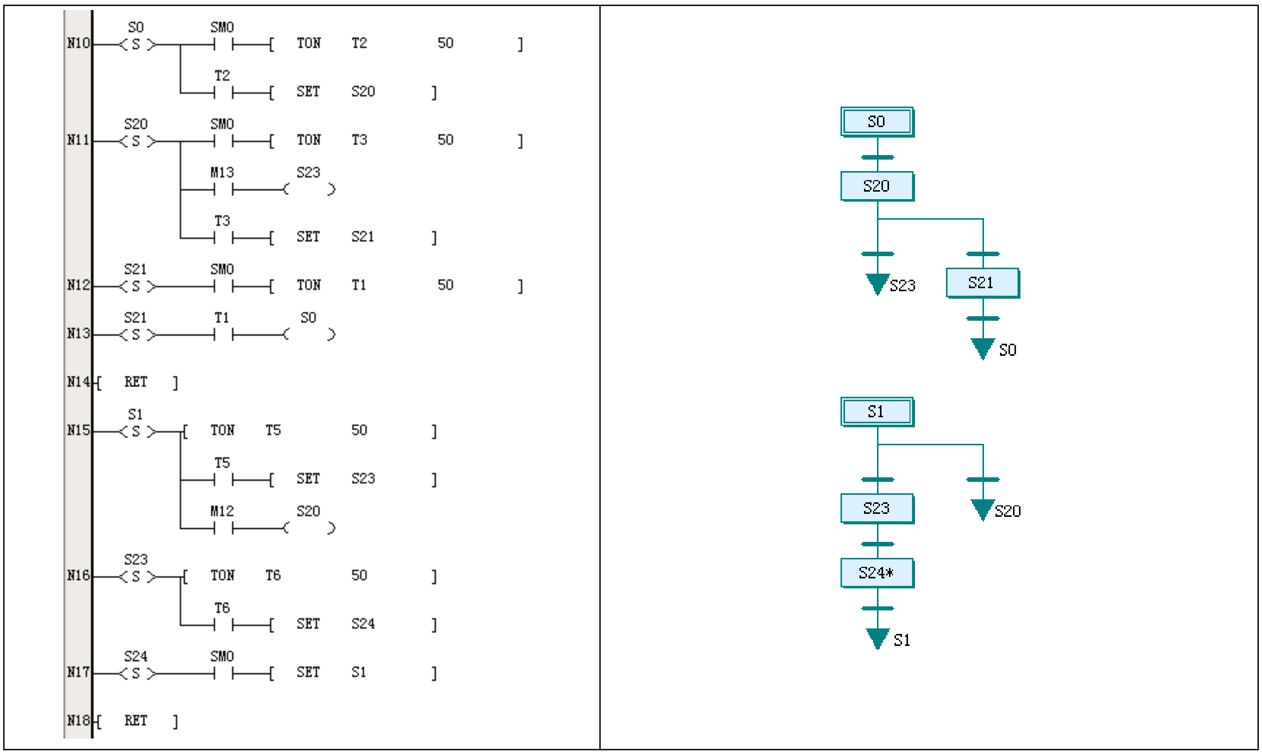
3. Jumping to another independent procedure

The SFC of EC series PLC supports multiple independent procedures and jumping among these procedures is allowed. You can set certain transfer conditions in an independent procedure for jumping to a random step (initial or normal) of another independent procedure.

📖 **Note**

Jumping among multiple independent procedures complicates the program. Use it with prudence.

Shown in the following figure is a jump from one independent procedure to another, with LAD on the left and SFC on the right.



In the SFC, when the S0 in the first procedure is valid, the program can jump to step S23 in the second procedure under certain conditions; while in the second procedure, the program can also jump to step S20 in the first procedure under certain conditions.

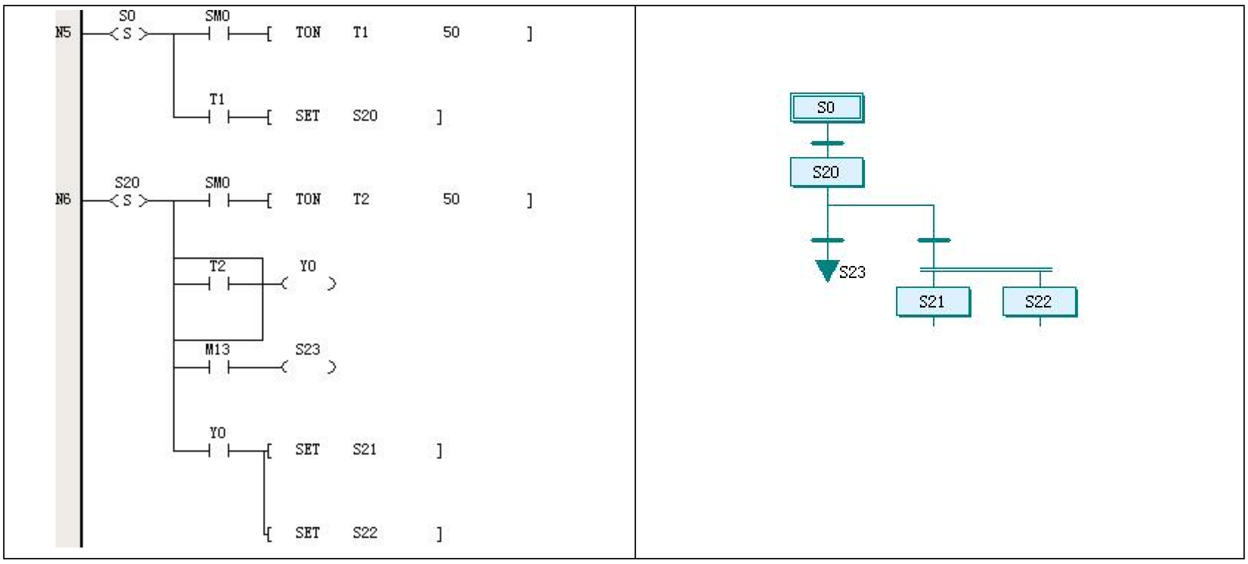
As shown in the preceding figure, the jump is based on a selection branch structure. When the program jumps to another procedure, all the steps in the original procedure will become invalid. As the example shows, if the program jumps to step S23 in the second procedure from step S20 in the first procedure, step S20 and all the other steps in the first procedure will become invalid.

7.1.6 Execution of SFC program

The similarity between the execution of a SFC program and that of a LAD program is that they both carry out cyclic scanning from up to down and from left to right.

On the other hand, their difference lies in that in a SFC program, the steps' validity will change according to certain conditions, and only valid steps can be executed. While in a LAD main program, the whole program will be scanned and executed in each scan cycle.

As shown in the following figure, the program on the right is the LAD counterpart of the SFC program on the left. When step S20 is valid, the T2 timer will be scanned and start timing. Steps S21 and S22 will not be executed before T2 counter reaches the preset value, and S23 will not be executed when M13 is OFF.



The S elements state will switch between ON and OFF according to the transfer conditions, thus making the program transfer from one step to another. When a S element changes from ON to OFF, the output elements of the corresponding step will be cleared or reset. For details, see **Error! Reference source not found.Error! Reference source not found.**.

Note

1. The SFC program of EC series PLC usually contains LAD program blocks that are used to handle operations besides the flow, including starting the SFC. The LAD program blocks are not controlled by the S elements and will be executed in every scan cycle.

2. Because the state change of the S element will affect the embedded instructions of the corresponding step, and the switch-over between two steps takes some time, it is necessary to observe certain rules during the SFC programming. For details, see *7.4Points to note in SFC programming*.

7.2 Relationship between SFC program and LAD program

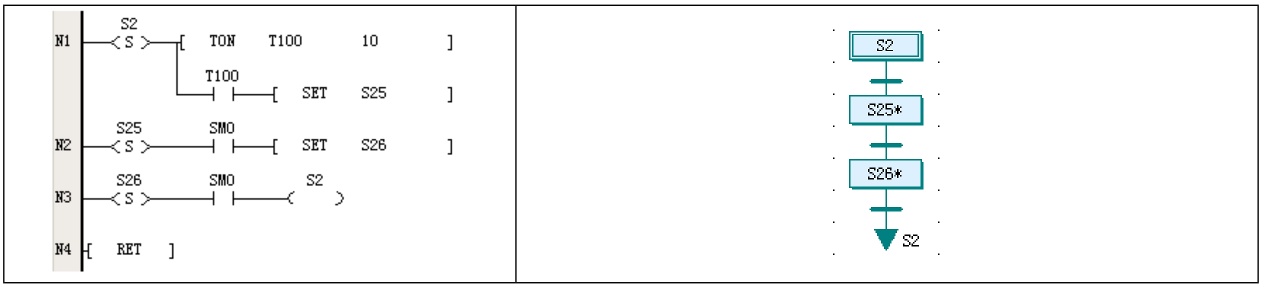
A SFC program can take the form of a LAD program, which can help understand the SFC program structure.

In the LAD program, the SFC symbols are replaced with various SFC instructions, while the procedures are represented by various structures.

7.2.1 STL instruction and step

All SFC steps are represented by S elements. In a LAD program, a step is started by a STL instruction.

Shown in the following left figure is the LAD program of a simple sequential structure, and the right figure is its corresponding SFC program.



As shown in the LAD program, the S2 step starts with a STL instruction, and the following TON instruction is the internal instruction of S2. A step can be made up of multiple instructions. A SFC step is actually a relatively complete program section, almost consistent with the LAD counterpart.

The difference between an initial step and a normal step is that they use different S elements.

For detailed information about the STL instruction, see **Error! Reference source not found.Error! Reference source not found.**. Note that when the step changes from ON to OFF, the destination operands of its internal instructions will be cleared. Such instructions include OUT, TON, TOF, PWM, HCNT, PLSY, PLSR, DHSCS, SPD, DHSCI, DHSCR, DHSZ, DHST, DHSP and BOUT.

🕮 **Note**

Because the PLC runs in continuous scan cycles, after a step transition, the instructions of the original step will not be affected by the change of ON to OFF until the next scan cycle. See *7.4.1Common programming errors*.

7.2.2 SFC transfter instruction

As shown in the preceding figure, the transfer symbols in the SFC program on the right are realized through the SET instructions in the LAD program on the left.

The transfer conditions consist of the NO contacts before the SET instruction. The NO contacts are controlled by internal instructions or through external operation.

When the power flow of the SET instruction is valid, the specified step becomes valid, and the current valid step is invalidated. A step transfer is thus complete.

7.2.3 RET instruction and SFC program section

As shown in the preceding figure, the SFC program on the right starts with a S2 initial step symbol, and returns to S2 after two ordinary steps. While in the LAD program, the SFC program section must end with the RET instruction.

The RET instruction can be only used in a main program.

7.2.4 SFC jump and reset instruction

As shown in the preceding figure, the jump to S2 is realized in LAD program by the N3 row, which uses an OUT instruction. The destination operand of the OUT instruction (jump) can be in any independent procedure.

If the reset S26 is used, line N3 in the LAD program will be a RST instruction, which can reset the last step S26.

7.2.5 SFC selection branch, parallel branch and merge

See ***Error! Reference source not found.*** in *7.1.5SFC program structure* for the LAD counterpart of SFC selection branches.

See ***Error! Reference source not found.***in *7.1.5SFC program structure* for the LAD counterpart of SFC parallel branches.

7.3 How to program with SFC

1. Analyse the work flow and determine the program structure

The structure of a SFC program is classified into three types: simple sequential structure, selection branch structure and parallel branch structure. Besides, the jump structure is also a special form of the selection branch structure.

To program with SFC, the first thing to do is to determine the structure of the flow. For example, a single object passing through a sequential flow is a simple sequential structure. Multiple objects with different parameters to be processed asynchronously needs a selection branch structure. While a cooperation of multiple independent mechanical elements may need a parallel branch structure.

2. Determine the major procedures and transfer conditions to draw a draft flow chart

After determining the strucuture, you need to figure out the major procedures and transfer conditions. By deviding the work flow into smaller operation stages, you can get the procedures. End each procedure with a transfer condition, and you can get the draft of the work flow.

3. Make a SFC program according to the draft flow chart

Use the SFC programming language in AutoStation to make a SFC program out of the draft flow chart. By now you have got an executable PLC program, but you still need to refine it.

4. Make a list of input and output points, and determine the objects of each procedure and the transfer conditions

Generally, the input points are transfer conditions, while the output points are the operation objects. In addition, with the list, you can further modify the SFC.

5. Input the steps and transfer conditions

In the SFC program you just made, right click a SFC symbol and select **Embedded Ladder Chart** in the shortcut menu. You are then able to edit the step or transfer condition through the LAD programming language.

6. Add functional program sections to the program

Do remember to add program sections that provide general functions, such as start, stop and alarm functions. Such program sections should all be put in LAD blocks.

Note

The start and stop operations are crucial for personal and equipment safety. Considering the special features of SFC program, make sure that all outputs that should be stopped are shut down when the PLC is stopped.

7.4 Points to note in SFC programming

The STL instruction has some special characteristics, and the PLC scans instructions cyclically by their display order. Because of these reasons, there are some points to note during SFC programming.

7.4.1 Common programming errors

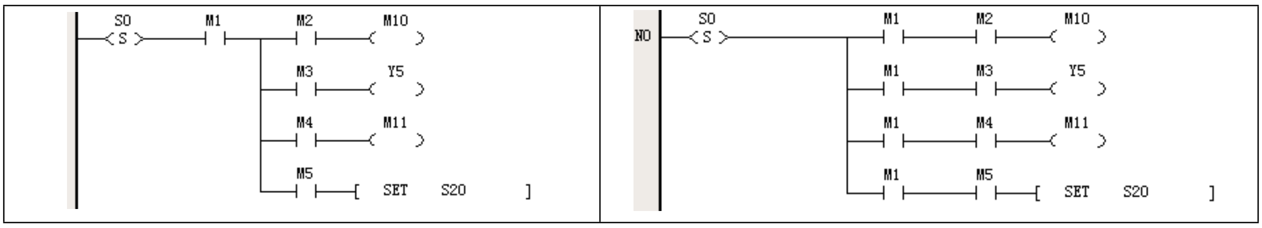
1. Reusing steps

In the same PLC program, each step corresponds to a unique S element and cannot be reused.

Note this when editing a SFC program using the LAD editor.

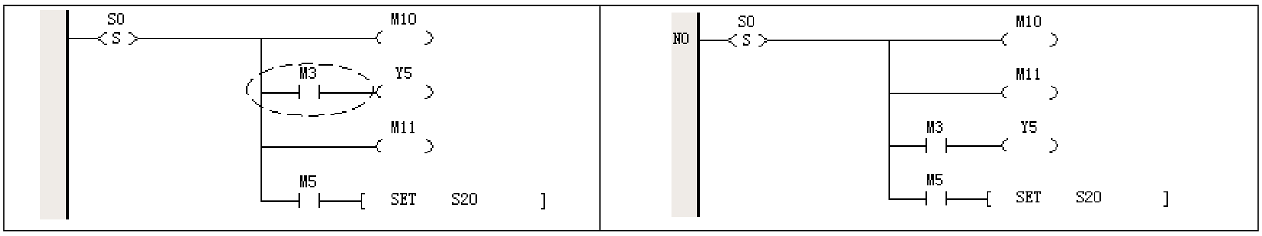
2. Setting branches after a transfer condition

Setting conditioned branches after a transfer condition is prohibited in SFC programming, as shown in the left figure below. Instead, you should change it into the right figure below.



3. Connecting output coils to internal bus after a NC or NO contact instruction

Connecting output coils to the internal bus after a NO or NC contact instruction in a branch is prohibited, as shown in the left figure below. Instead, you should change it into the right figure below.

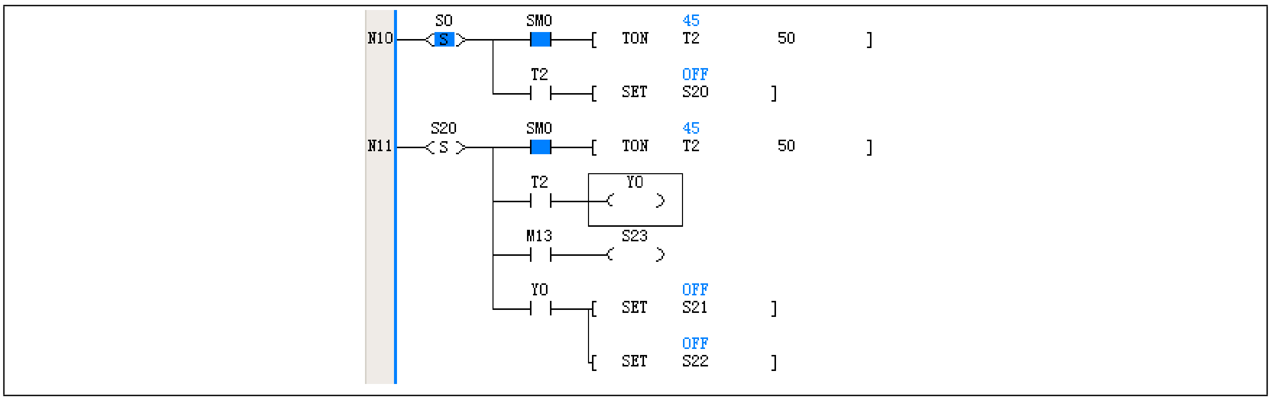


4. Reusing the same element in neighboring steps

The PLC scans instructions by their display order. The scanning of the current step and that of the next step are closely joined together.

Therefore, after a STL instruction is executed, although certain elements of the instruction will be reset (see **Error! Reference source not found.Error! Reference source not found.**), the reset will not be carried out until the next scan cycle. That means, at the moment of the transfer, the elements of the last step retains their states and values until the step is scanned in the next cycle.

As shown in the following figure, the two neighboring steps use the same timer: T2. When the S0-S20 transfer occurs, the T2 will retain its value and state, rendering the step S20 unable to perform as it is designed. The program will jump directly to S21 and S22. Therefore, it should be noted that, although reusing elements in a program is not prohibited, you should avoid reusing them in neighboring steps, or accidents may occur.

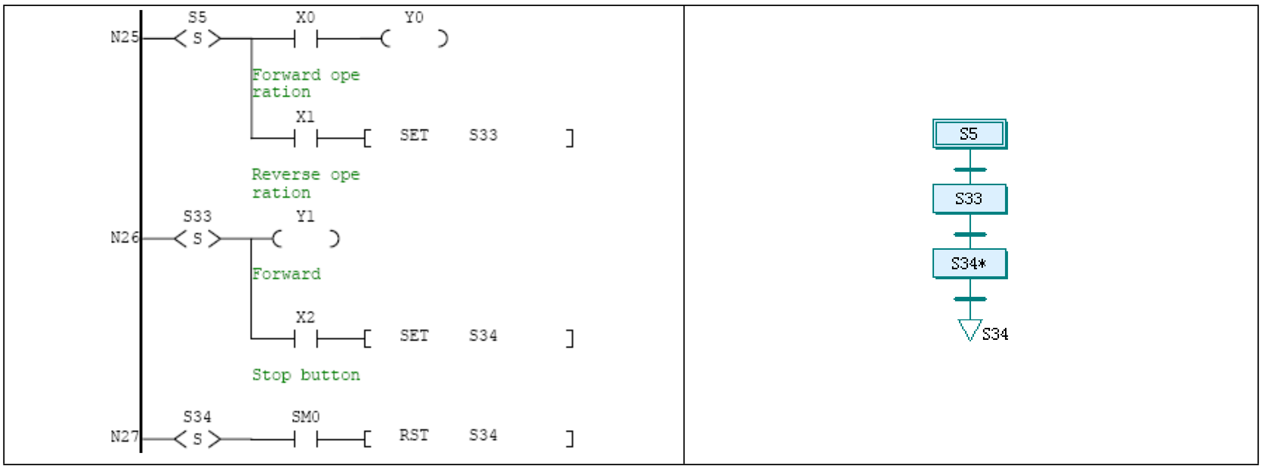


5. Failing to inter-lock elements

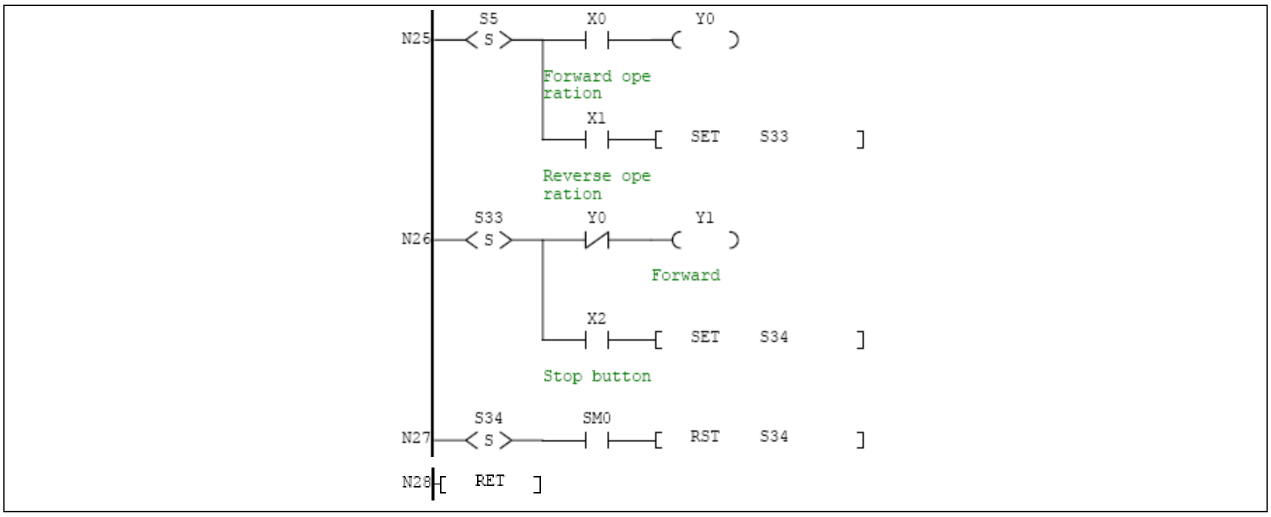
During SFC programming, certain elements may become contradictary to each other under some special transfer conditions. Inter-locking is then necessary.

Take the following forward&backward operation program as an example, where Y0 and Y1 are respectively forward and backward output. X0 is forward operation, X1 is backward operation, and X2 the is stop button. Y0 and Y1 should be inter-locked, that is to say, they should not be ON at the same time.

However, in this example, when Y0 is ON, if X1 is ON and the S33 is validated, Y1 will be also ON, within the same scan cycle with Y0.



Therefore, you need to add an interlock to the program by adding a Y0 NC contact before the Y1 output coil, as shown in the following figure.

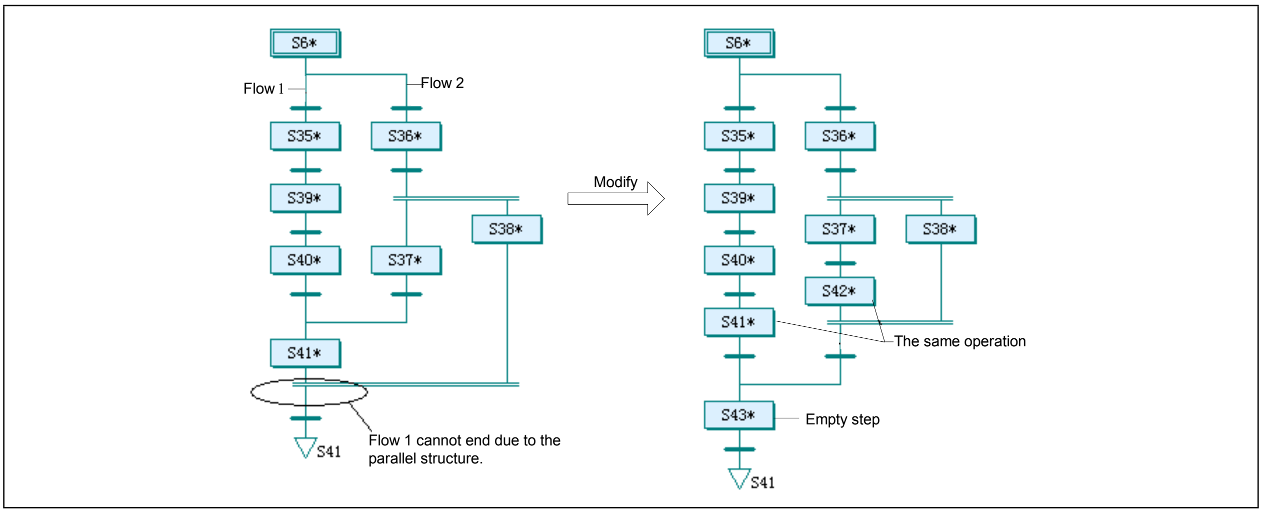


6. Confusing jumps with transfers

Jumps are used between different procedures or non-neighboring steps, while transfers are used between neighboring steps. It is prohibited to change an output coil into a SET instruction where a jump should be used, or change a output coil into a SET instruction where a transfer should be used.

7. Using parallel merge for selection branches

In a selection branch structure, only one selection is valid. However, when it is mixed with a parallel branch structure, the selection branch structure may never end. As shown in the following figure, in the left part, when flow 1 runs to step S41, it meets the transfer condition of a parallel merge. But the system will never run flow 2. Therefore the transfer will never occur, making flow 1 unable to end.



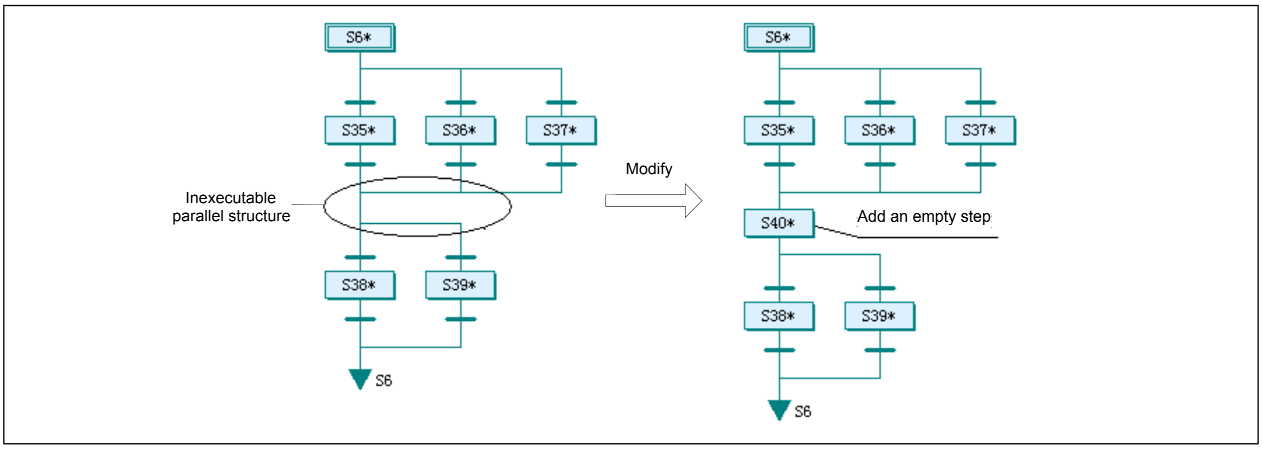
As shown in the right part, to correct it, you need to add a step S42 whose function is the same as S41. Then add an empty step S43 that serves as a structural block without actual function. Design the transfer conditions for S38, S41 and S43 according to the actual situation.

7.4.2 Programming skills

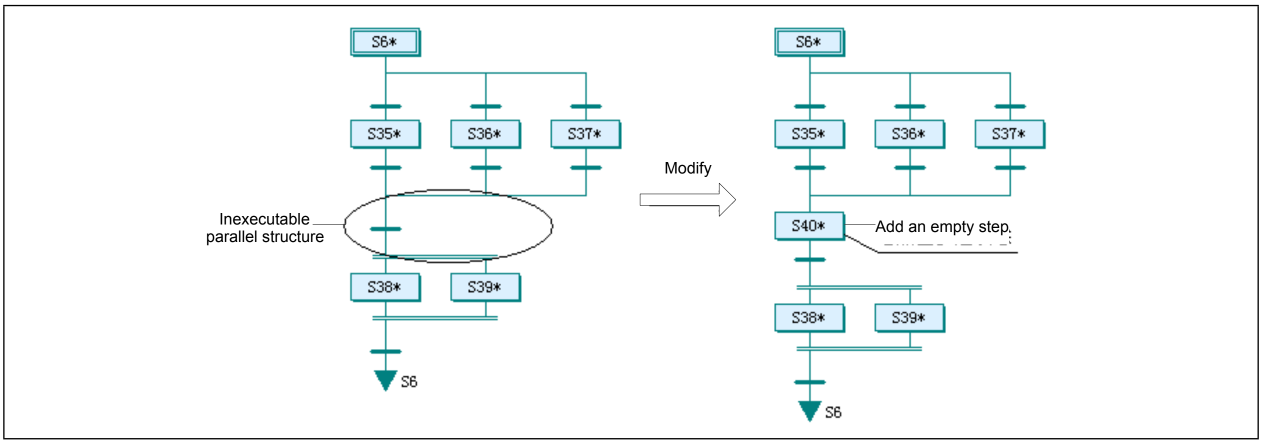
1.Making use of empty steps

You may need empty steps to deal with the branches with grammatical problems. The empty steps do not provide actual operation, but a necessary node in structure before the next transfer. See the following example.

In the left figure below, the selection merge is connected immediately with another selection branch structure. That is prohibited. You can change it as the right figure shows: add an empty step.



In the left figure below, the selection merge is connected immediately with a parallel branch structure. That is prohibited too. You can also change it as the right figure shows: add an empty step.

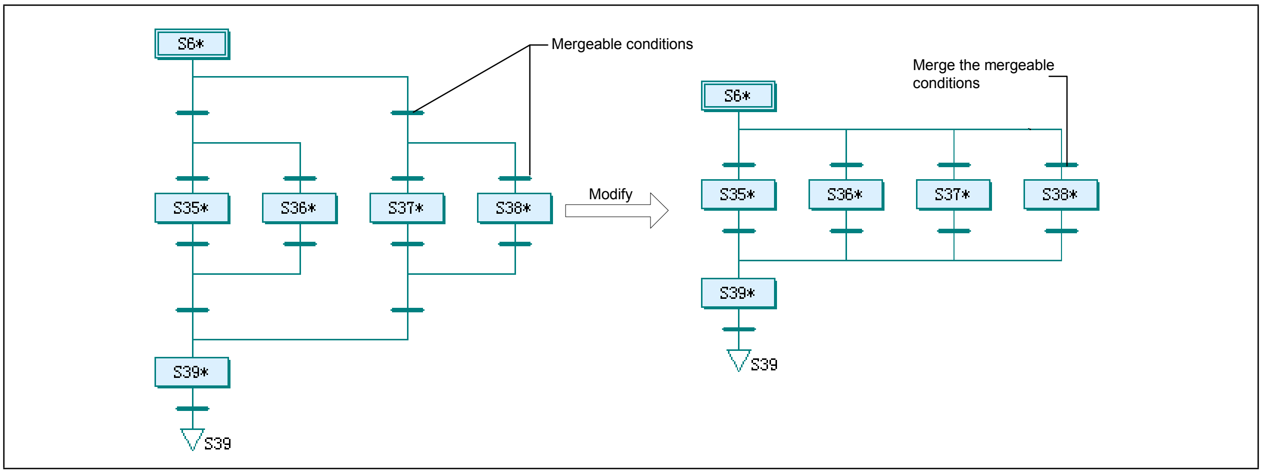


You can address other structures, such as parallel merge connected with parallel branches, or parallel branches connected with selection branches, by adding an empty step.

2.Merging branches and transfer conditions

Some seemingly complicated branches are the result of bad design. You can simplify them by merging some branches.

As shown below, the designer set a selection branch first, following it by two selection branches. However, simply four selection branches will achieve the same. The original two-level transfer conditions become one level transfer condition.



3.Making use of battery backup function

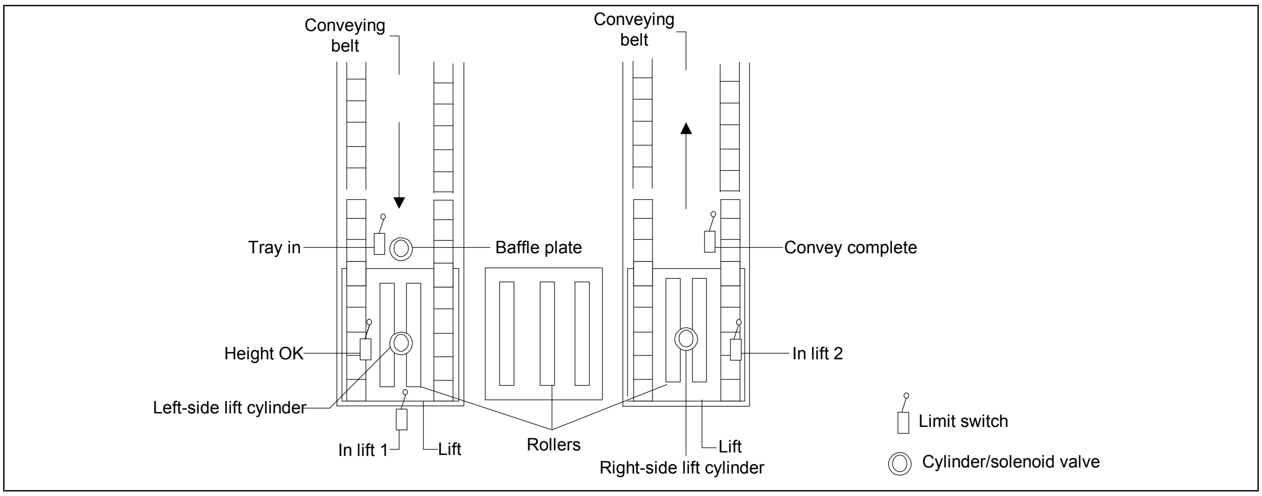
The S elements can be saved upon power failure by the battery. In this way, the program can resume from the step when the power failure occurred.

7.5 Examples of SFC programming

The examples in this section are just illustrations of SFC programming, with simplified operations and conditions. The equipment configuration is conceptual and for study only. Do not apply the example programs to actual use.

7.5.1 Simple sequential structure

The following example is an object lifting and conveying machine. This machine uses cylinder lifting devices and rollers to convey the object tray from one conveying belt to another. The following figure is a top view of the machine.



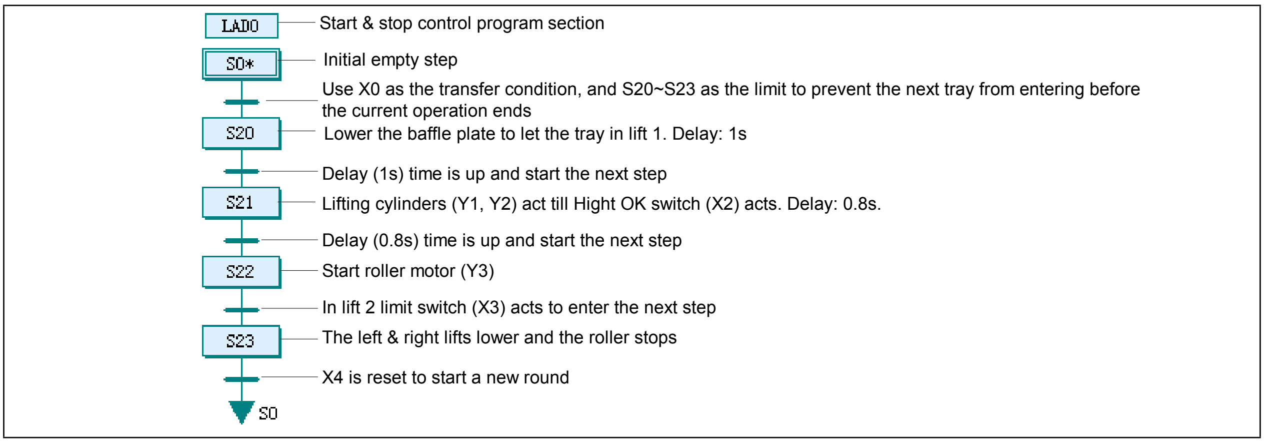
After the machine is started, the object tray will be conveyed to the entrance of the machine at the left side and trigger the "Tray in" limit switch. If no other tray is occupying the machine, the "Baffle plate" will lower down to let the object tray enter the machine. When the tray is completely into the lift when it triggers the "In lift 1" limit switch, the lift will raise the tray until the "Height OK" limit switch is triggered. The rollers will then act to convey the tray to the lift on the right side until the "In lift 2" limit switch is triggered. The lift will then lower to put the tray to the conveying belt on the right. When the "Convey complete" limit switch is reset, a complete lift and convey process is over and the machine is ready for the next round.

The input and output points are listed in the following table.

SN	Address	Monitored object		SN	Address	Monitored object
1	X0	Tray in limit switch		8	Y0	Cylinder solenoid valve for the baffle plate
2	X1	In lift 1 limit switch		9	Y1	Cylinder solenoid valve for the left lift
3	X2	Height OK limit switch		10	Y2	Cylinder solenoid valve for the right lift
4	X3	In lift 2 limit switch		11	Y3	Roller motor contactor
5	X4	Convey complete		12	Y4	Motor contactor for the left conveying belt
6	X5	Start switch		13	Y5	Motor contactor for the right conveying belt
7	X6	Auxiliary signal of emergency switch				

This is a simple sequential flow. The procedures are linear, without any selection or parallel procedures. Writing the program with SFC would be faster and clearer than the conventional logic design method.

See the following figure for the SFC program and its LAD counterpart.



Start & stop control program section

Initial empty step

Use X0 as the transfer condition, and S20~S23 as the limit to prevent the next tray from entering before the current operation ends

Lower the baffle plate to let the tray in lift 1. Delay: 1s

Delay (1s) time is up and start the next step

Lifting cylinders (Y1, Y2) act till Hight OK switch (X2) acts. Delay: 0.8s.

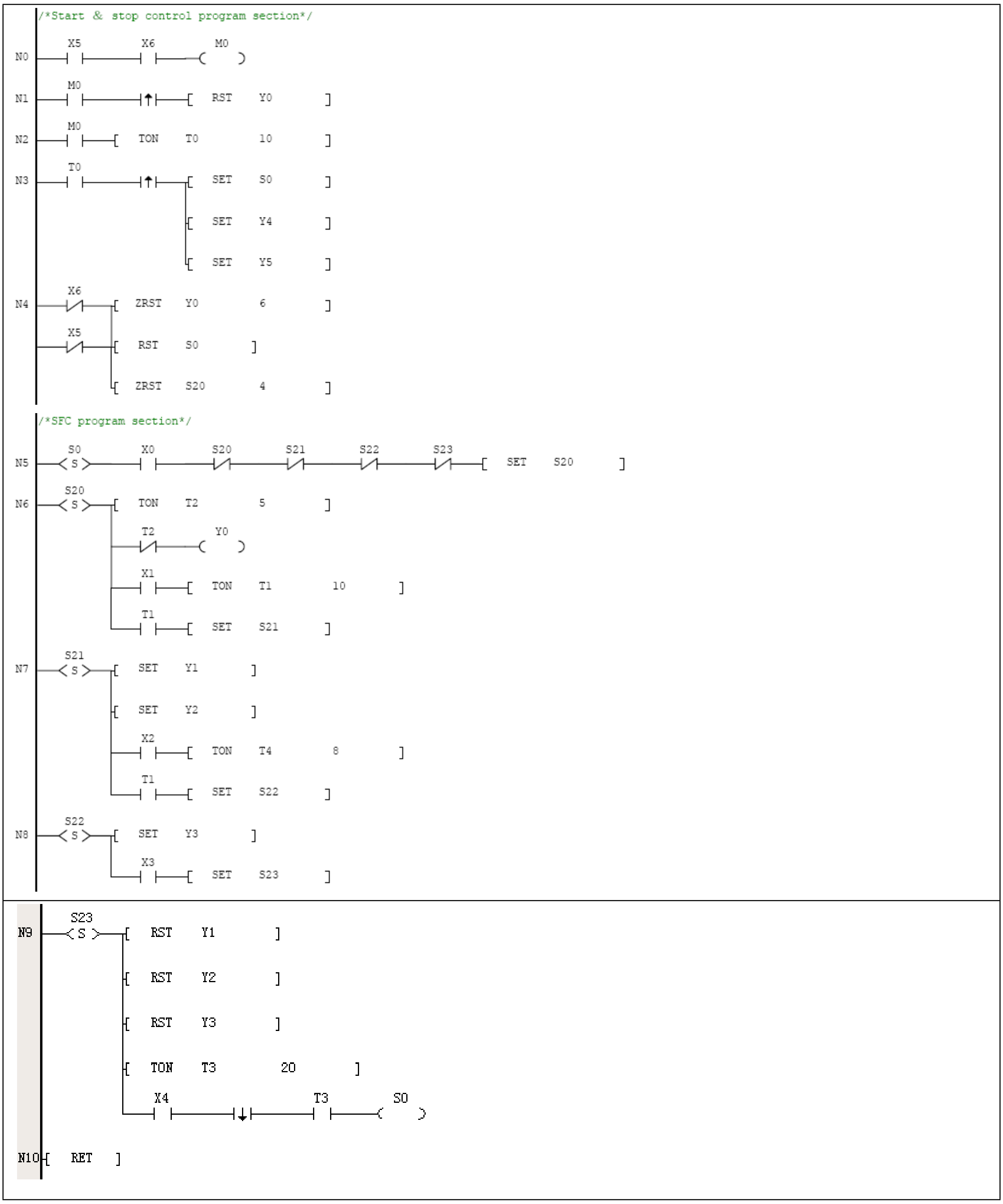
Delay (0.8s) time is up and start the next step

Start roller motor (Y3)

In lift 2 limit switch (X3) acts to enter the next step

The left & right lifts lower and the roller stops

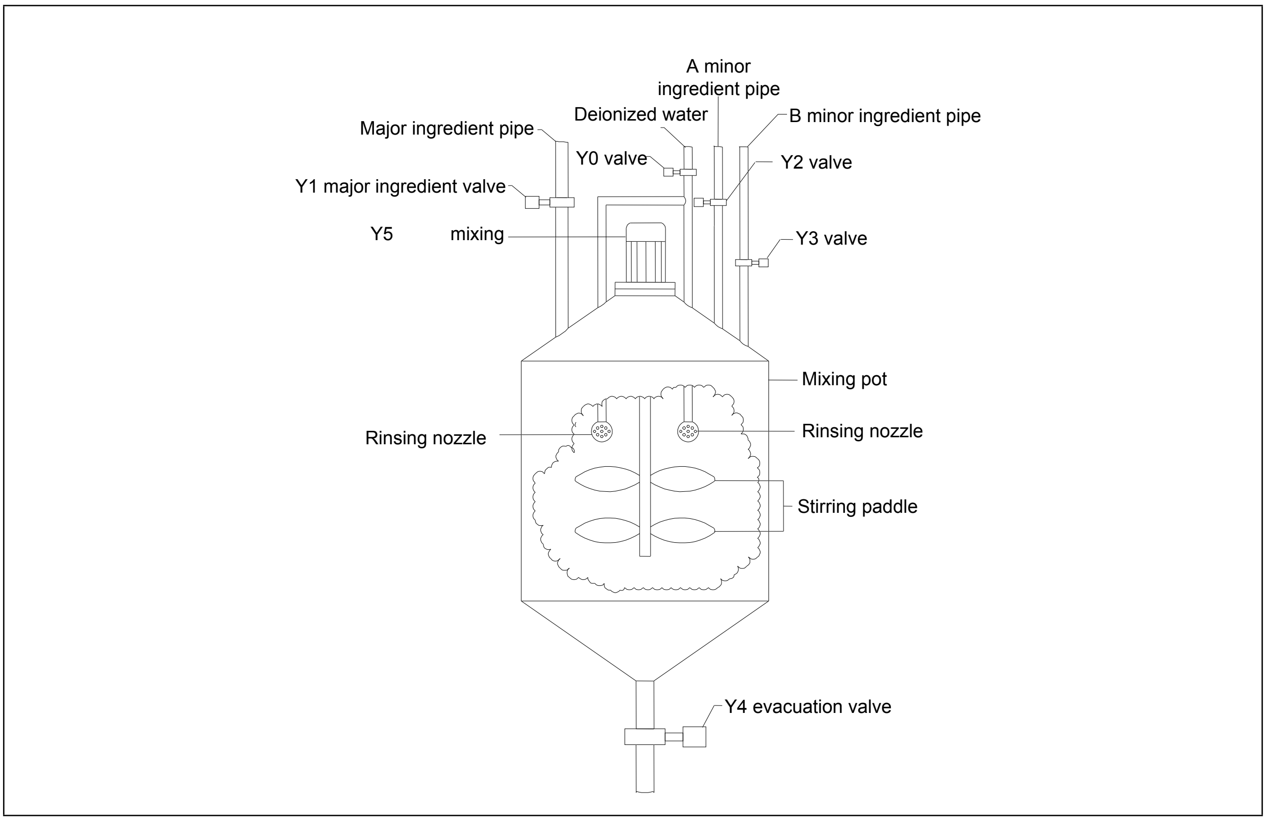
X4 is reset to start a new round



```
/*Start & stop control program section*/
N0   X5  X6  ( M0 )
N1   M0  ↑  [ RST  Y0 ]
N2   M0  [ TON  T0  10 ]
N3   T0  ↑  [ SET  S0 ]
            [ SET  Y4 ]
            [ SET  Y5 ]
N4   X6  [ ZRST  Y0  6 ]
     X5  [ RST  S0 ]
         [ ZRST  S20  4 ]
/*SFC program section*/
N5   S0 <S>  X0  S20  S21  S22  S23  [ SET  S20 ]
N6   S20 <S>  [ TON  T2  5 ]
     T2  ( Y0 )
     X1  [ TON  T1  10 ]
     T1  [ SET  S21 ]
N7   S21 <S>  [ SET  Y1 ]
              [ SET  Y2 ]
     X2  [ TON  T4  8 ]
     T1  [ SET  S22 ]
N8   S22 <S>  [ SET  Y3 ]
     X3  [ SET  S23 ]
N9   S23 <S>  [ RST  Y1 ]
              [ RST  Y2 ]
              [ RST  Y3 ]
              [ TON  T3  20 ]
     X4  ↓  T3  ( S0 )
N10  [ RET ]
```

7.5.2 Selection branch structure

The following example is a material mixing flow. Through this flow, two kinds of products, namely A and B, are produced. See the following figure for the illustration of the manufacturing device.



To start the operation, the operator should select through the touch screen the product type, A or B, for the next batch of product. As the second step, the major ingredient wil be added until the added ingredient reaches 2000kg. As the third step, minor ingredient, A for type A product or B for type B product, will be added until the added minor ingredient reaches 500kg. As the forth step, the ingredients will be mixed round for 20 minutes. As the fifth step, the material will be evacuated until the left material is less than 20kg and the delay is over. Then the machine is ready for the next round.

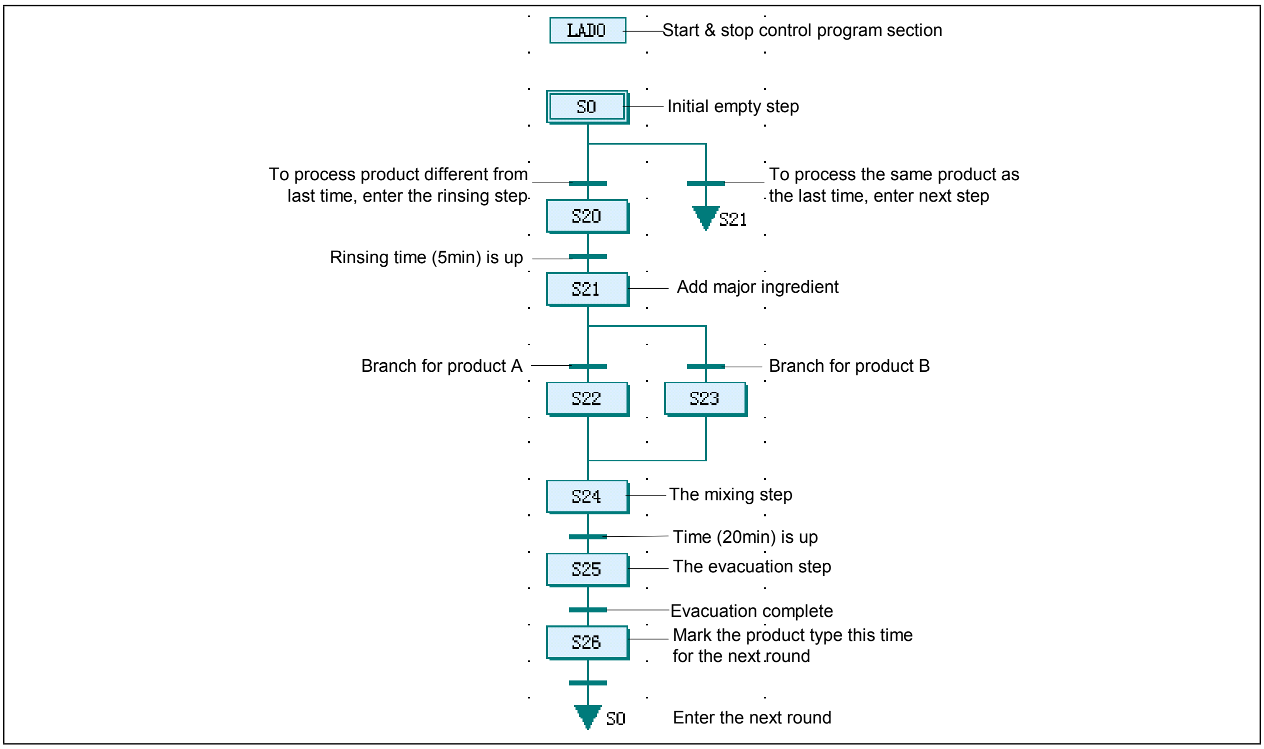
If the machine is brand new, or the product type produced last time is different from what is going to be produced, you need to open the deionized water valve and evacuation valve to rinse the machine for 5 minutes before the operation.

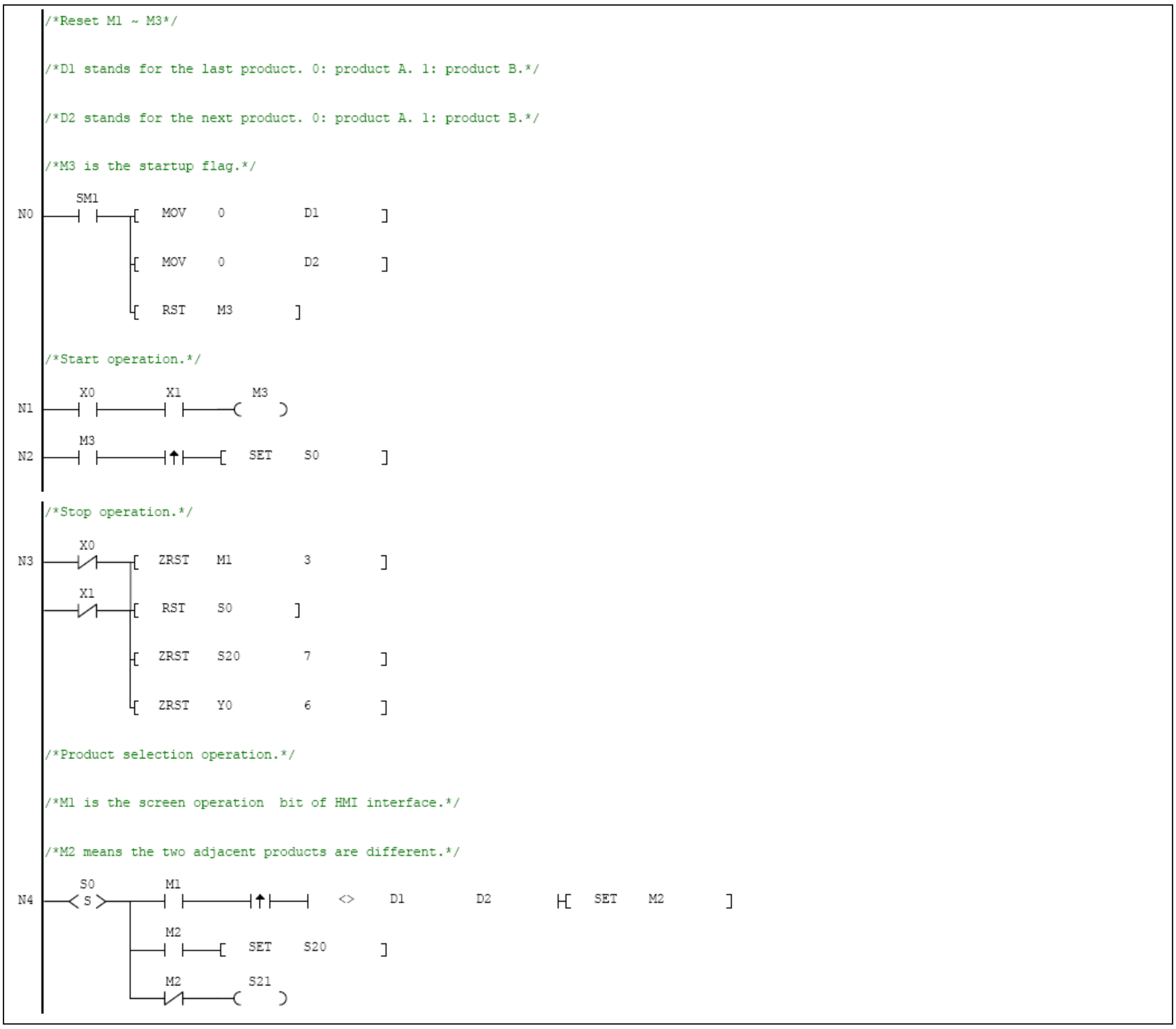
The input and output points are listed in the following table.

SN	Address	Monitored object		SN	Address	Monitored object
1	X0	Deionized water valve open		10	X11	Evacuation valve open
2	X1	Deionized water valve closed		11	X12	Evacuation valve closed
3	X2	Major ingredient valve open		12	Y0	Solenoid valve for deionized water
4	X3	Major ingredient valve closed		13	Y1	Solenoid valve for major ingredient
5	X4	Minor ingredient A valve open		14	Y2	Solenoid valve for minor ingredient A
6	X5	Minor ingredient A valve closed		15	Y3	Solenoid valve for minor ingredient B
7	X6	Minor ingredient B valve open		16	Y4	Solenoid valve for evacuation
8	X7	Minor ingredient B valve closed		17	Y5	Mixing motor contactor
9	X10	Mixing motor running				

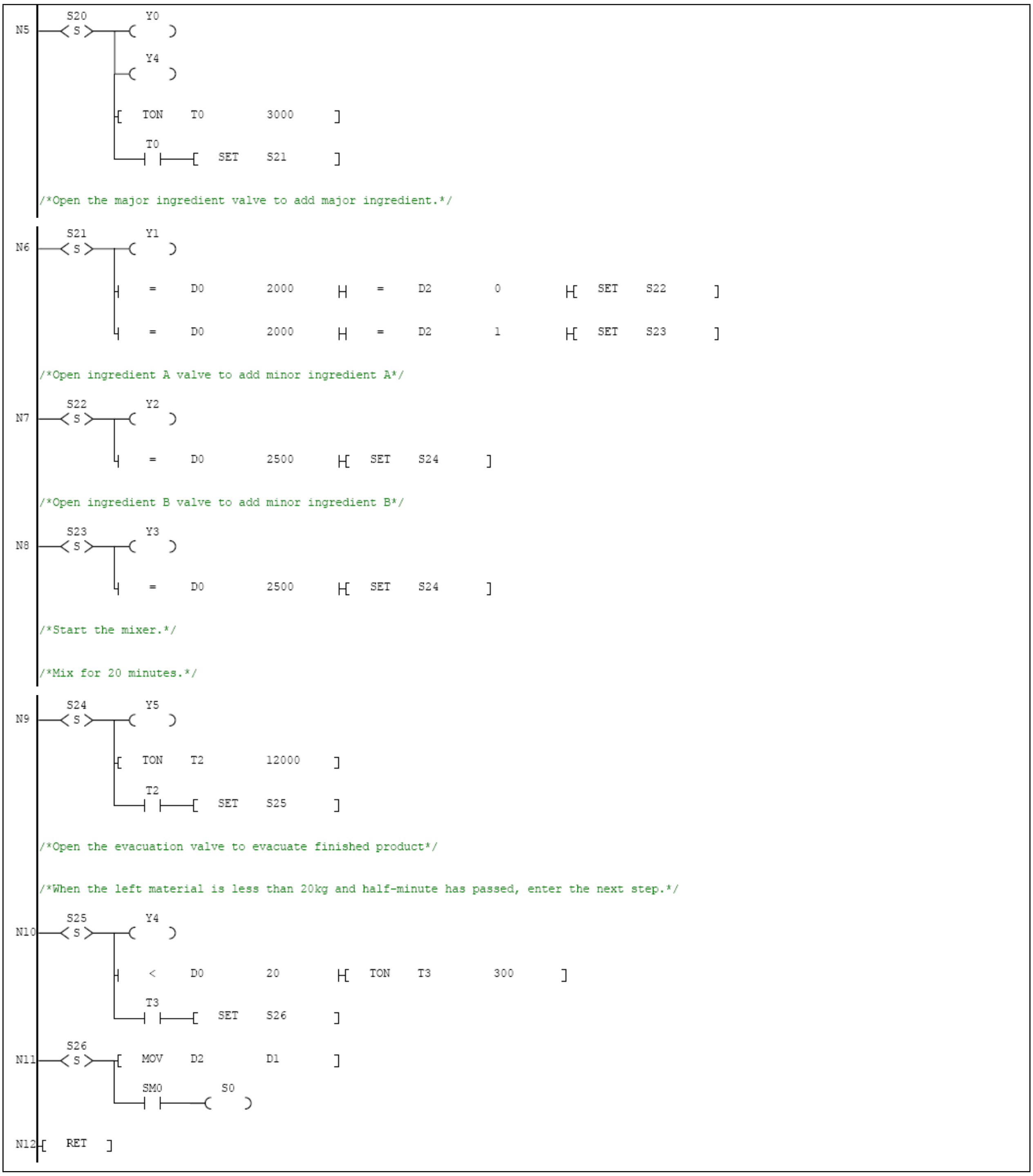
Obviously this is a selection branch structured flow. You can select only one type of product , A or B, in a round. Meanwhile, the flow has a selection and jump structure: the rinsing procedure.

The following figures are the corresponding SFC program and its LAD counterpart.


LAD0
Start & stop control program section
S0
Initial empty step
To process product different from last time, enter the rinsing step
To process the same product as the last time, enter next step
S20
S21
Rinsing time (5min) is up
S21
Add major ingredient
Branch for product A
Branch for product B
S22
S23
S24
The mixing step
Time (20min) is up
S25
The evacuation step
Evacuation complete
S26
Mark the product type this time for the next round
S0
Enter the next round



```
/*Reset M1 ~ M3*/
/*D1 stands for the last product. 0: product A. 1: product B.*/
/*D2 stands for the next product. 0: product A. 1: product B.*/
/*M3 is the startup flag.*/
      SM1
N0  --| |--+-[ MOV   0        D1       ]
           |
           +-[ MOV   0        D2       ]
           |
           +-[ RST   M3    ]
/*Start operation.*/
      X0       X1        M3
N1  --| |------| |------(    )
      M3
N2  --| |------|↑|--[ SET   S0       ]
/*Stop operation.*/
      X0
N3  --|/|--+-[ ZRST  M1       3        ]
      X1   |
    --|/|--+-[ RST   S0    ]
           |
           +-[ ZRST  S20      7        ]
           |
           +-[ ZRST  Y0       6        ]
/*Product selection operation.*/
/*M1 is the screen operation  bit of HMI interface.*/
/*M2 means the two adjacent products are different.*/
      S0        M1
N4  --< S >--+--| |------|↑|--|  <>   D1      D2     |-[ SET   M2      ]
             |  M2
             +--| |--[ SET   S20      ]
             |  M2         S21
             +--|/|------(    )
```

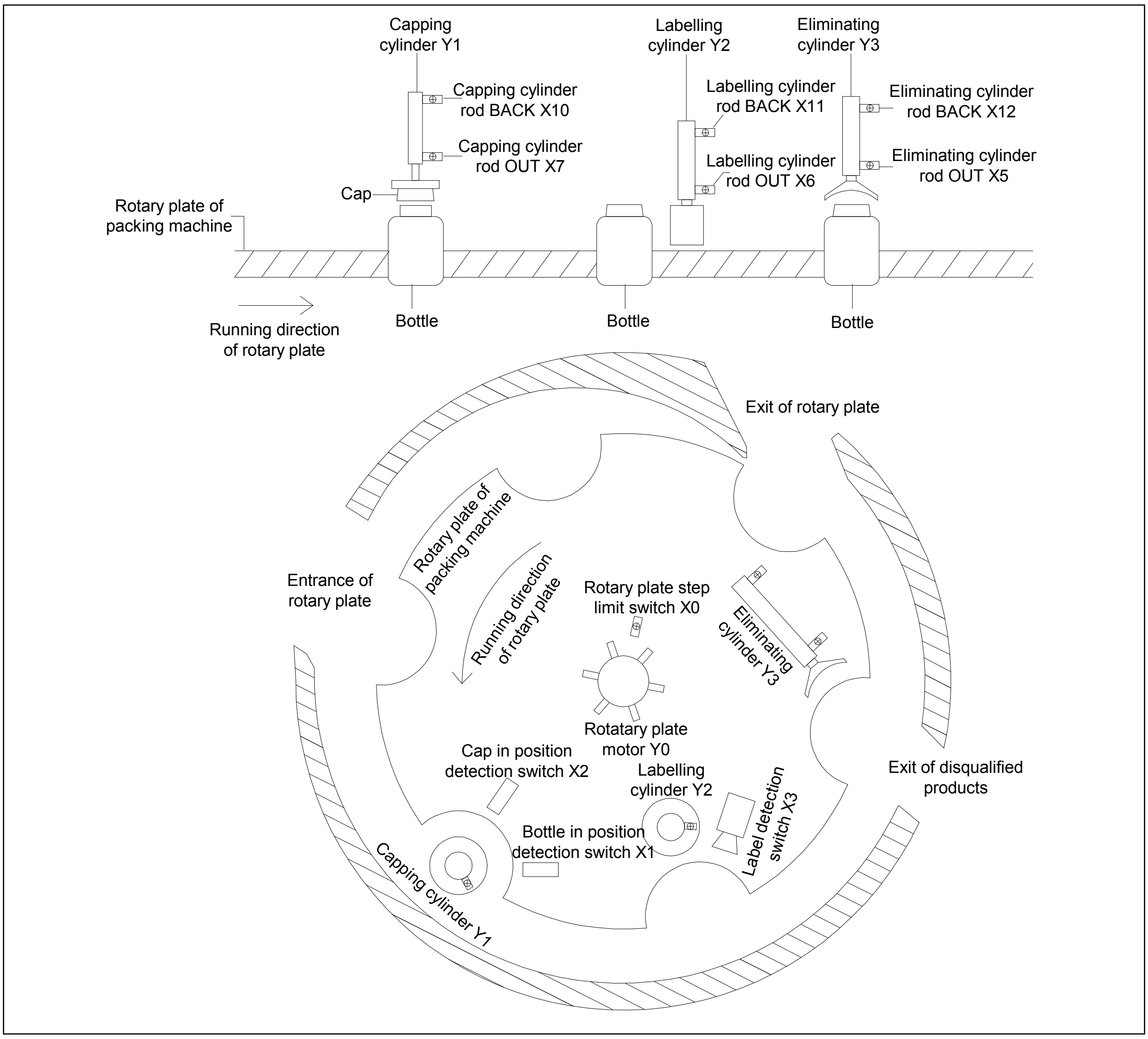


```
     S20        Y0
N5 ──< S >──┬─(     )
            │   Y4
            ├─(     )
            │
            ├[ TON   T0        3000    ]
            │   T0
            └──┤ ├──[ SET   S21     ]

/*Open the major ingredient valve to add major ingredient.*/
     S21        Y1
N6 ──< S >──┬─(     )
            │
            ├┤  =   D0        2000    ├┤  =   D2        0        ├[ SET   S22     ]
            │
            └┤  =   D0        2000    ├┤  =   D2        1        ├[ SET   S23     ]

/*Open ingredient A valve to add minor ingredient A*/
     S22        Y2
N7 ──< S >──┬─(     )
            │
            └┤  =   D0        2500    ├[ SET   S24     ]

/*Open ingredient B valve to add minor ingredient B*/
     S23        Y3
N8 ──< S >──┬─(     )
            │
            └┤  =   D0        2500    ├[ SET   S24     ]

/*Start the mixer.*/

/*Mix for 20 minutes.*/
     S24        Y5
N9 ──< S >──┬─(     )
            │
            ├[ TON   T2        12000   ]
            │   T2
            └──┤ ├──[ SET   S25     ]

/*Open the evacuation valve to evacuate finished product*/

/*When the left material is less than 20kg and half-minute has passed, enter the next step.*/
     S25        Y4
N10──< S >──┬─(     )
            │
            ├┤  <   D0        20      ├[ TON   T3        300     ]
            │   T3
            └──┤ ├──[ SET   S26     ]
     S26
N11──< S >──┬[ MOV   D2        D1      ]
            │   SM0         S0
            └──┤ ├──────(     )

N12┤[ RET  ]
```

7.5.3 Parallel branch structure

The next example is a bottle packager. The packager seals the bottles and sticks labels to them. Meanwhile, it will examine the bottle cap and label, so that the flawed products will be eliminated in the third procedure, while the qualified products will continue to the next work flow.

If no bottle is sent from the last work flow, the packager will not conduct any sealing or labelling. The three procedures are carried out at the same time, and each bottle moves from one position to another each time the rotary plate rotates. See the following figure for the illustration of the packager.


Capping cylinder Y1
Labelling cylinder Y2
Eliminating cylinder Y3
Capping cylinder rod BACK X10
Capping cylinder rod OUT X7
Labelling cylinder rod BACK X11
Labelling cylinder rod OUT X6
Eliminating cylinder rod BACK X12
Eliminating cylinder rod OUT X5
Cap
Rotary plate of packing machine
Running direction of rotary plate
Bottle
Bottle
Bottle
Exit of rotary plate
Rotary plate of packing machine
Entrance of rotary plate
Running direction of rotary plate
Rotary plate step limit switch X0
Eliminating cylinder Y3
Rotatary plate motor Y0
Cap in position detection switch X2
Labelling cylinder Y2
Label detection switch X3
Exit of disqualified products
Bottle in position detection switch X1
Capping cylinder Y1

During the operation, the rotary plate rotates one step each time, which is detected by the X0 limit switch. The rotary plate will stay at each step long enough for all the three procedures, driven by cylinders, are finished. The cylinder rod OUT signal and cylinder rod BACK signal are monitored respectively.

The input and output points arel isted in the following table.

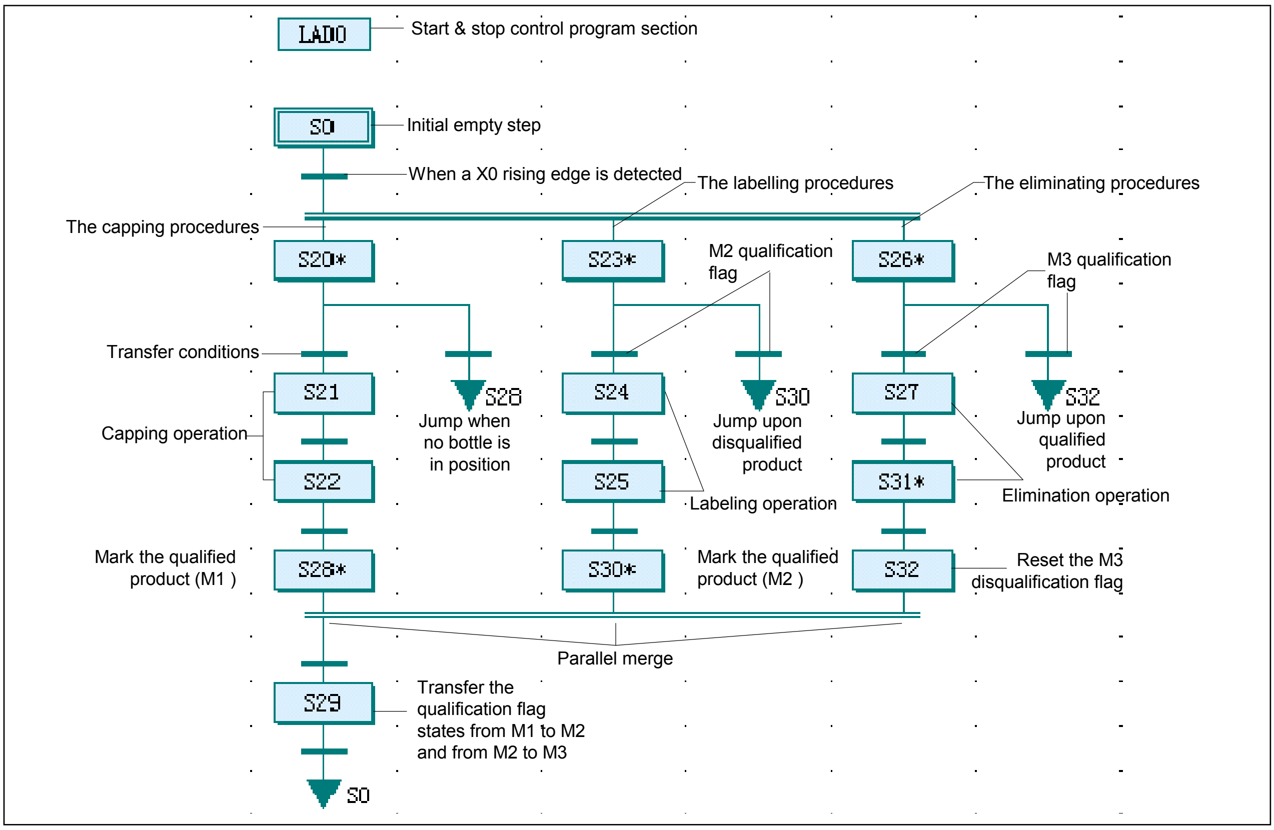
SN	Address	Monitored object		SN	Address	Monitored object
1	X0	Rotary plate step limit switch		8	X10	Capping cylinder rod BACK
2	X1	Bottle in position detection switch		9	X11	Labelling cylinder rod BACK
3	X2	Cap in position detection switch		10	X12	Eliminatiing cylinder rod BACK
4	X3	Label detection switch		11	Y0	Rotary plate motor
5	X5	Eliminating cylinder rod OUT		12	Y1	Capping cylinder
6	X6	Labelling cylinder rod OUT		13	Y2	Labelling cylinder
7	X7	Capping cylinder rod OUT		14	Y3	Eliminating cylinder

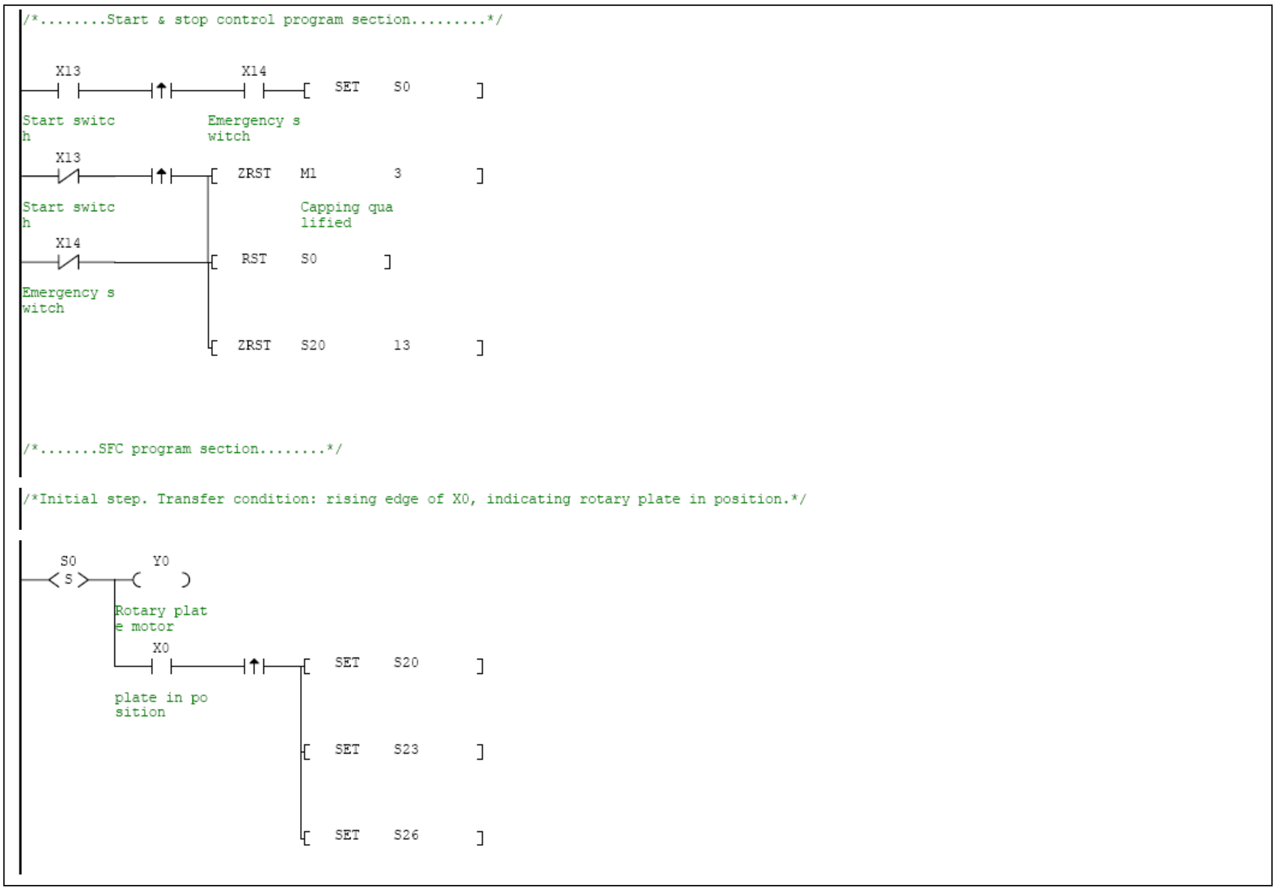
It is obvious that this is a parallel branch structured flow. With every step that the rotary plate makes, all the tree procedures are carried out at the same time. Then, when the three procedures are finished, the rotary plate will rotate one step again. See the following figure for the corresponding SFC program and its LAD counterpart.

In the program, M1~M3 are the qualification flags for the procedures of capping, labeling and eliminating respectively.

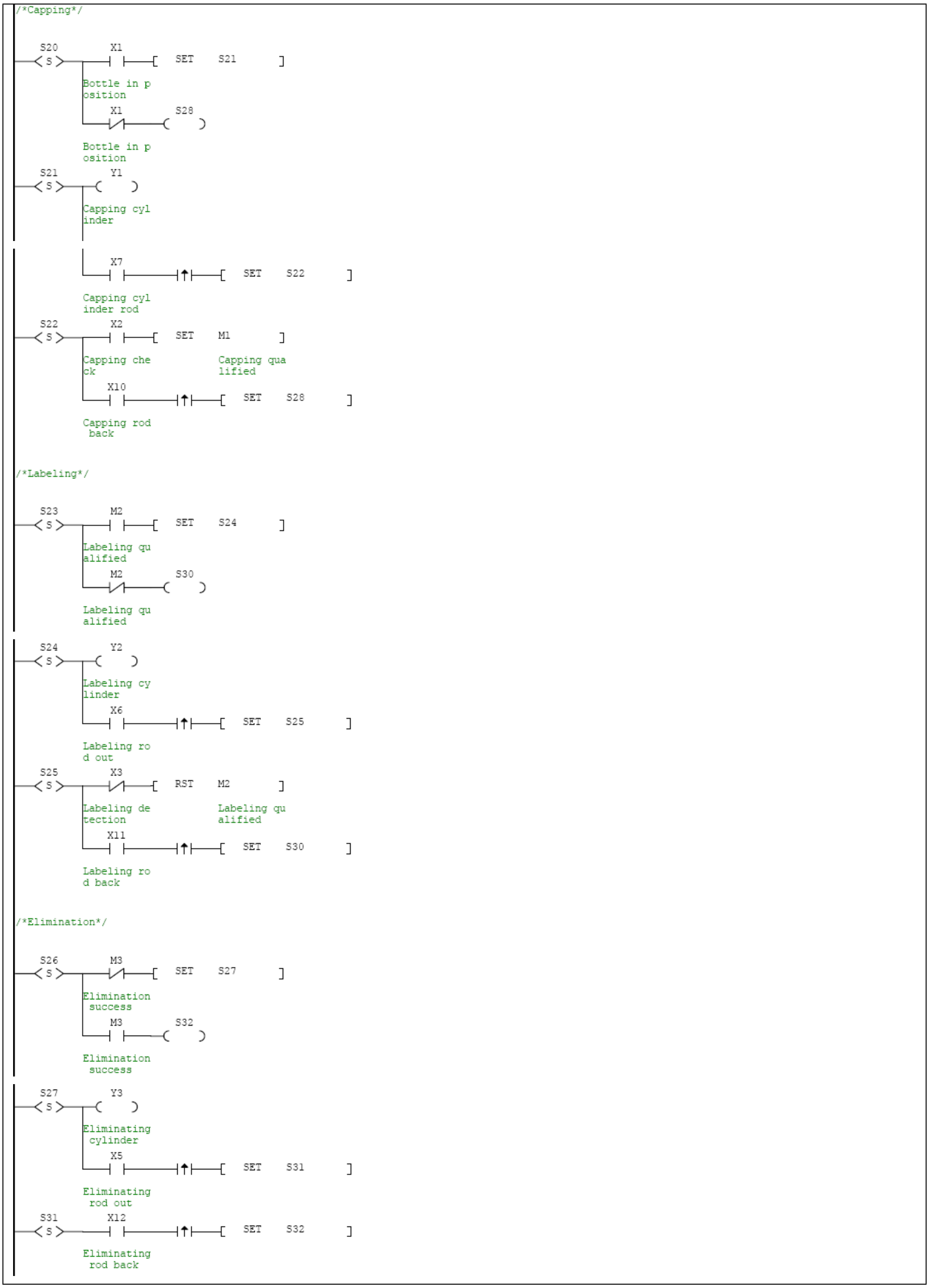
When the capping procedure runs to S22, X2 will check whether the capping is qualified or not. If yes, the corresponding qualification flag M1 will be set. When the labelling procedure runs to S25, X3 will check whether the labelling is qualified or not. If not, M2 will be reset. After all the procedures are complete, at step S29, the M2 state will be transferred to M3, and M1 state will be transferred to M2.

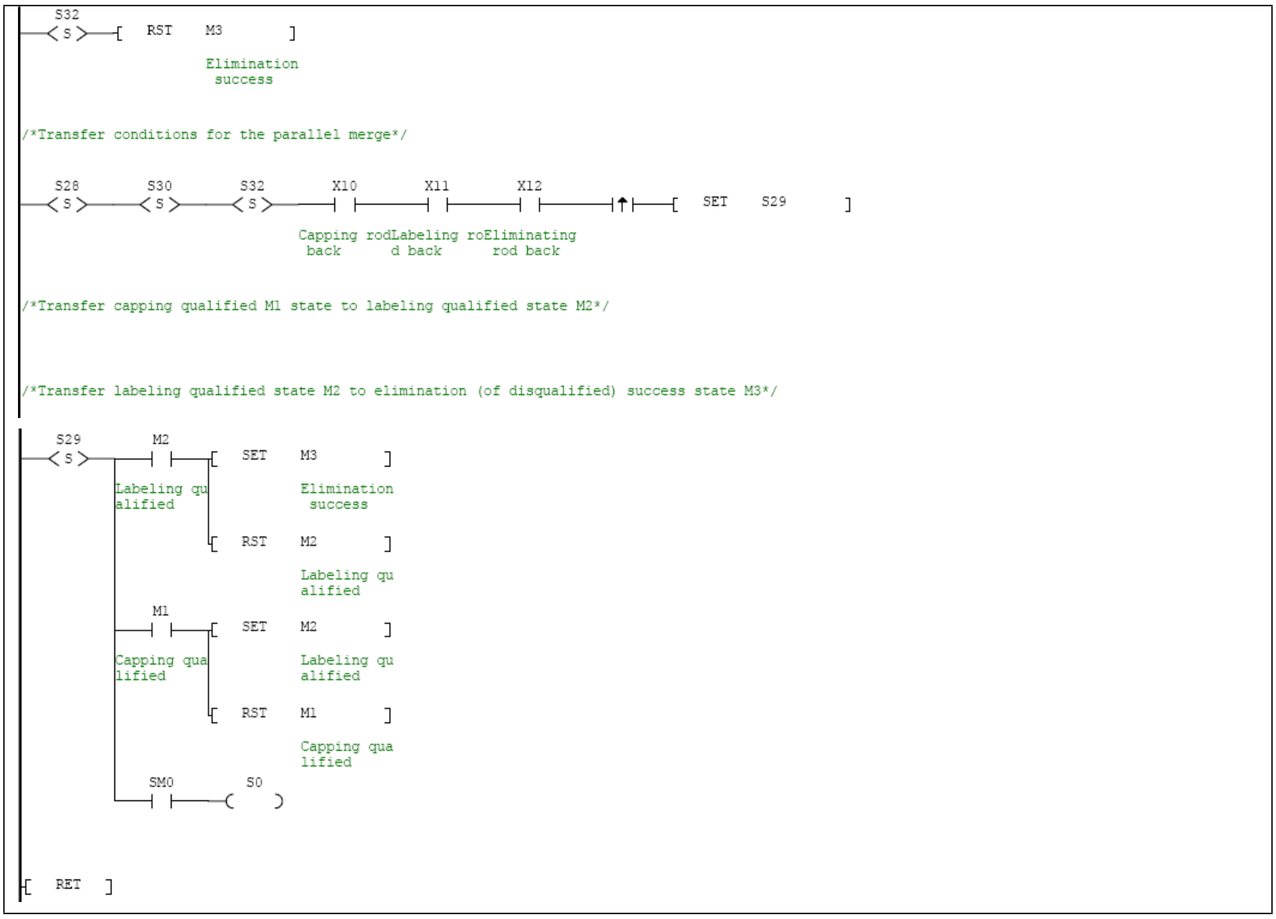
The capping procedure will act according to X1 state. If X1 indicates no bottle is in position, the capping will not proceed. The labelling procedure will act according to M2 state. If M2 is OFF, it indicates that the bottle in position is disqualified, and the labelling will not proceed. The eliminating procedure will act according to M3. The elimination will not be conducted when M3 is ON, which indicates that the bottle is qualified, or the elimination will be conducted otherwise. In both cases, M3 will be reset in S32 to prepare for the next procedure.


LAD0
Start & stop control program section
S0
Initial empty step
When a X0 rising edge is detected
The labelling procedures
The eliminating procedures
The capping procedures
S20*
S23*
M2 qualification flag
S26*
M3 qualification flag
Transfer conditions
S21
S28
Jump when no bottle is in position
S24
S30
Jump upon disqualified product
S27
S32
Jump upon qualified product
Capping operation
S22
S25
Labeling operation
S31*
Elimination operation
Mark the qualified product (M1)
S28*
S30*
Mark the qualified product (M2)
S32
Reset the M3 disqualification flag
Parallel merge
S29
Transfer the qualification flag states from M1 to M2 and from M2 to M3
S0



```
/*........Start & stop control program section.........*/

  X13                      X14
--| |--------|↑|-----------| |----[  SET    S0       ]
Start switc               Emergency s
h                         witch
  X13
--|/|--------|↑|----+-[  ZRST   M1       3        ]
Start switc         |             Capping qua
h                   |             lified
  X14               |
--|/|---------------+-[  RST    S0       ]
Emergency s         |
witch               |
                    |
                    +-[  ZRST   S20      13       ]


/*.......SFC program section........*/

/*Initial step. Transfer condition: rising edge of X0, indicating rotary plate in position.*/

  S0          Y0
--< S >--+--(     )
         | Rotary plat
         | e motor
         |    X0
         +---| |--------|↑|--+-[  SET    S20      ]
           plate in po       |
           sition            |
                             +-[  SET    S23      ]
                             |
                             +-[  SET    S26      ]
```


/*Capping*/
S20
X1
SET S21
Bottle in position
X1
S28
Bottle in position
S21
Y1
Capping cylinder
X7
SET S22
Capping cylinder rod
S22
X2
SET M1
Capping check
Capping qualified
X10
SET S28
Capping rod back
/*Labeling*/
S23
M2
SET S24
Labeling qualified
M2
S30
Labeling qualified
S24
Y2
Labeling cylinder
X6
SET S25
Labeling rod out
S25
X3
RST M2
Labeling detection
Labeling qualified
X11
SET S30
Labeling rod back
/*Elimination*/
S26
M3
SET S27
Elimination success
M3
S32
Elimination success
S27
Y3
Eliminating cylinder
X5
SET S31
Eliminating rod out
S31
X12
SET S32
Eliminating rod back



/*Transfer conditions for the parallel merge*/

/*Transfer capping qualified M1 state to labeling qualified state M2*/

/*Transfer labeling qualified state M2 to elimination (of disqualified) success state M3*/

Chapter 8 Using high-speed input functions

This chapter presents the usage and notes about the high-speed input functions, including high-speed counter and external pulse capture function.

8.1 High-speed counter

8.1.1 Configuration

The built-in high-speed counter for EC series small PLCs are configured as follows:

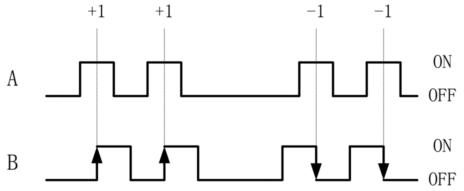
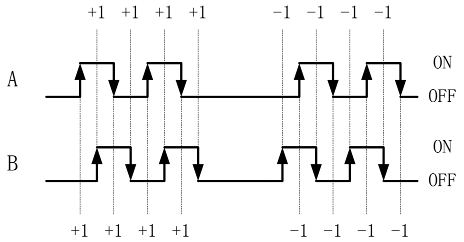
Table 8-1 High-speed counter configuration

Counter	Input point	X0	X1	X2	X3	X4	X5	X6	X7	Max. frequency(kHz)			
										EC20H	EC20	EC10	EC10V
Single phase single point input mode	C236	Up/down								100	100		
	C237		Up/down										
	C238			Up/down							10		
	C239				Up/down								
	C240					Up/down							
	C241						Up/down						
	C301							Up/down			/		
	C302								Up/down				
	C242	Up/down		Reset							10		
	C243				Up/down		Reset						
	C244	Up/down		Reset				Start					
	C245				Up/down		Reset		Start				
Single phase up/ down input mode	C246	Up	Down							100	50		
	C247	Up	Down	Reset							10		
	C303				Up	Down					/		
	C248				Up	Down	Reset				10		
	C249	Up	Down	Reset				Start					
	C250				Up	Down	Reset		Start				
Double phase input mode	C251	Ph A	Ph B							50	30		
	C304			Ph A	Ph B						/		
	C305					Ph A	Ph B						
	C306							Ph A	Ph B				
	C252	Ph A	Ph B	Reset							5		
	C253				Ph A	Ph B	Reset						
	C254	Ph A	Ph B	Reset				Start					
	C255				Ph A	Ph B	Reset		Start				
SPD instruction		Input point	Input point	Input point	Input point	Input point	Input point	Input point	Input point	100	10		
Pulse capture function		Input point	Input point	Input point	Input point	Input point	Input point	Input point	Input point	/	/		
External interrupt No. (rising/falling edge)		0/10	1/11	2/12	3/13	4/14	5/15	6/16	7/17	/	/		

In the modes listed in the preceding table, the high-speed counters will act according to certain input and handle high-speed action according to interrupts. The counting practice is unrelated to the PLC scan cycle.

All the high-speed counters are of the 32-bit up/down type. According to their different up/down switchover methods, they fall into the following four categories:

Counting method	Action
Single phase single point input	C236~C245 and C301~C302 are down counters when SM236~SM245 and SM301~SM302 are ON, and up counters when SM236~SM245 and SM301~SM302 are OFF

Counting method	Action
Single phase up/down input	Corresponding to actions of up/down input, C246~C250 and C303 automatically count up/down, SM246~SM250 and SM303 determine the current direction of corresponding counters, up input when SM elements are OFF and down input when SM elements are ON
Double phase input	When SM100~SM104 are set OFF, C251~C255 C304~C306 will conduct up/down counting according to double phase input. SM251~SM255 and SM304~SM306 determine the current direction of corresponding counters, up input when SM elements are OFF and down input when SM elements are ON The counting directions are shown as follows: 
Double phase fourfold frequency input	When SM100~SM104 are set ON, C251~C255 C304~C306 will conduct fourfold frequency up/down counting according to double phase input. SM251~SM255 and SM304~SM306 determine the current direction of corresponding counters, up input when SM elements are OFF and down input when SM elements are ON The counting directions are shown as follows:  +1 +1 +1 +1 −1 −1 −1 −1 A ON OFF B ON OFF +1 +1 +1 +1 −1 −1 −1 −1

8.1.2 Relationship between high-speed counter and SM auxiliary relay

Special auxiliary relay for controlling counting direction

Type	Counter SN	Up/down control
Single phase single point input	C236	SM236
	C237	SM237
	C238	SM238
	C239	SM239
	C240	SM240
	C241	SM241
	C242	SM242
	C243	SM243
	C244	SM244
	C245	SM245
	C301	SM301
	C302	SM302

Special auxiliary relay for controlling fourfold frequency

Type	Counter SN	Fourfold frequency control
Double phase input	C251	SM100
	C252	SM100
	C253	SM102
	C254	SM100
	C255	SM102
	C304	SM101
	C305	SM103
	C306	SM104

Special auxiliary relay for monitoring counting direction

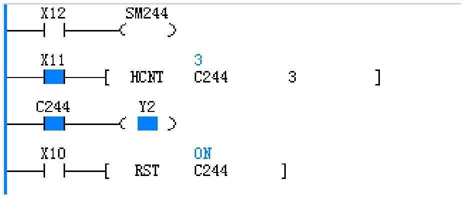
Type	Counter SN	Up/down monitor
Single phase up/down input	C246	SM246
	C247	SM247
	C248	SM248
	C249	SM249
	C250	SM250
	C303	SM303
Double phase input	C251	SM251
	C252	SM252
	C253	SM253
	C254	SM254
	C255	SM255
	C304	SM304
	C305	SM305
	C306	SM306

8.1.3 Usage of high-speed counter

■ Usage of single phase single point input high-speed counter

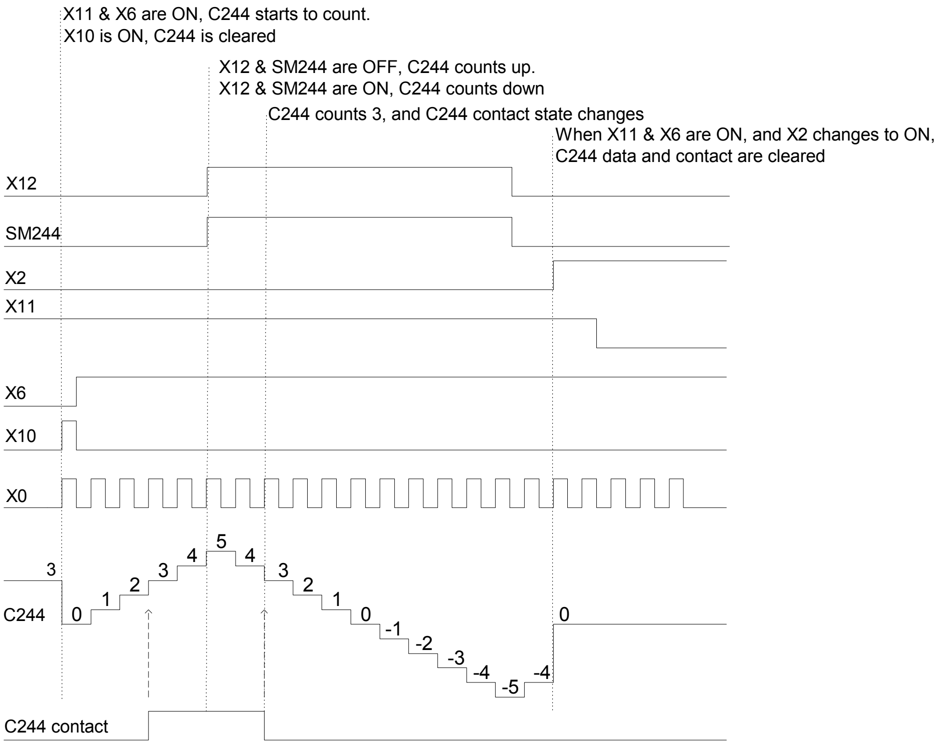
The single phase single point input high-speed counter starts to count only when the pulse input changes from OFF to ON, with the counting direction determined by its corresponding SM element.

Example:



```
X12      SM244
─┤ ├────(     )

X11                3
─┤■├───[ HCNT   C244      3        ]

C244     Y2
─┤■├────( ■ )

X10                ON
─┤ ├───[ RST    C244      ]
```

The time sequence chart of the contacts action in the program is shown in the following figure:



X11 & X6 are ON, C244 starts to count.
X10 is ON, C244 is cleared

X12 & SM244 are OFF, C244 counts up.
X12 & SM244 are ON, C244 counts down

C244 counts 3, and C244 contact state changes

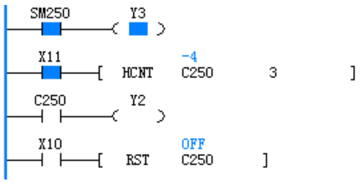
When X11 & X6 are ON, and X2 changes to ON,
C244 data and contact are cleared

Note:
1. Counter input point: X0.
2. High speed counters, when used in instructions DHSCS, DHSCR, DHSZ, DHSP and DHST, can trigger operations free from the scan cycle.

■ Usage of single phase up/down input high-speed counter

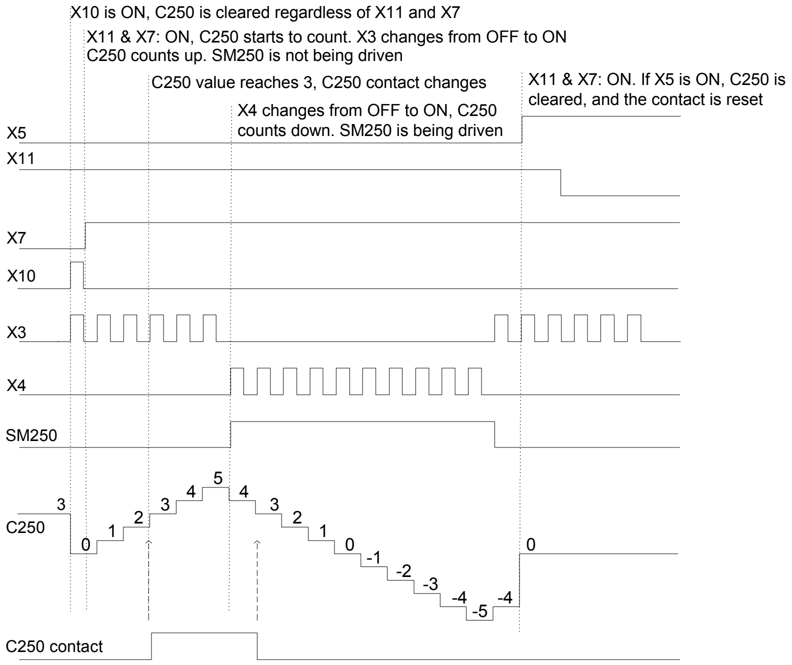
The single phase up/down input high-speed counter starts to count only when the pulse input changes from OFF to ON. The two input points determine its counting direction, which is monitored by its corresponding SM element.

Example:



```
SM250    Y3
─┤■├────( ■ )

X11                -4
─┤■├───[ HCNT   C250      3        ]

C250     Y2
─┤ ├────(     )

X10                OFF
─┤ ├───[ RST    C250      ]
```

The time sequence chart of the contacts action in the program is shown in the following figure:

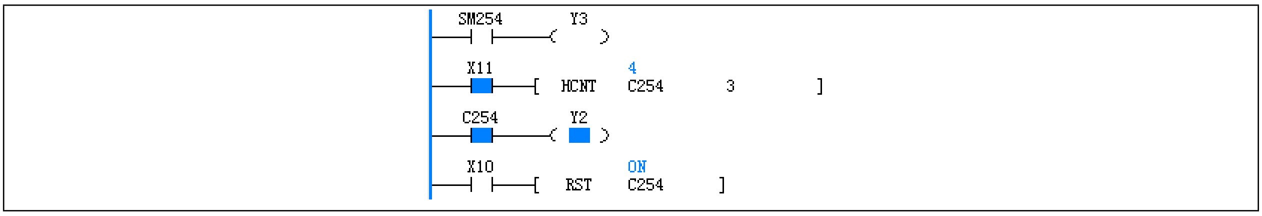


Note:
1. Counter input points: X3 & X4.
2. High speed counters, when used in instructions DHSCS, DHSCR, DHSZ, DHSP and DHST, can trigger operations free from the scan cycle.

■ Usage of double phase input high-speed counter

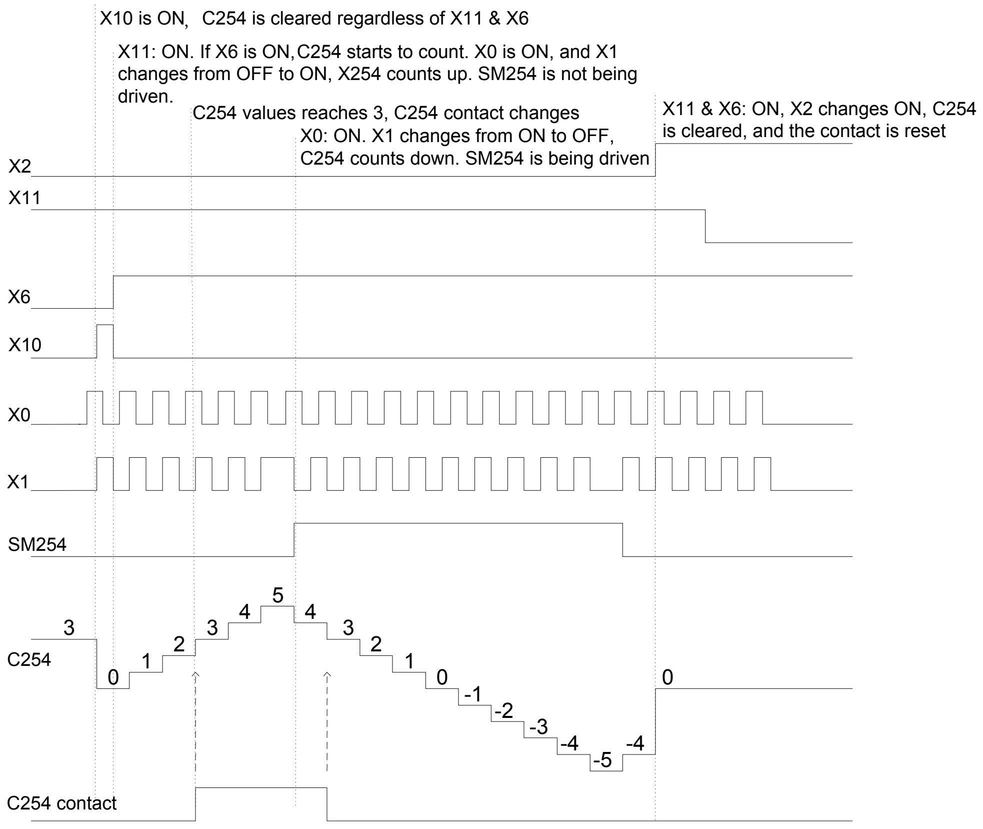
The double phase input high-speed counter starts to count only when the pulse input changes from OFF to ON. The phase difference of the two pulse inputs determines the counting direction, which is monitored by the corresponding SM element.

Example:



```
 SM254         Y3
──┤ ├────────(    )

 X11               4
──┤█├────────[ HCNT  C254       3        ]

 C254          Y2
──┤█├────────(  █  )

 X10               ON
──┤ ├────────[ RST   C254      ]
```

The time sequence chart of the contacts action in the program is shown in the following figure:



X10 is ON, C254 is cleared regardless of X11 & X6

X11: ON. If X6 is ON,C254 starts to count. X0 is ON, and X1 changes from OFF to ON, X254 counts up. SM254 is not being driven.

C254 values reaches 3, C254 contact changes

X0: ON. X1 changes from ON to OFF, C254 counts down. SM254 is being driven

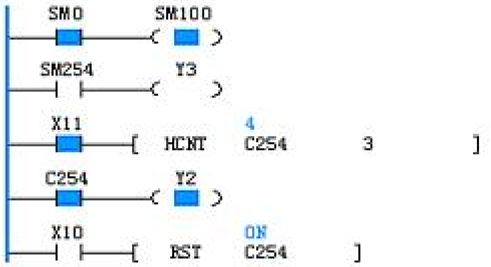
X11 & X6: ON, X2 changes ON, C254 is cleared, and the contact is reset

Note:
1. Counter input points: X0 & X1.
2. High-speed counters, when used in instructions DHSCS, DHSCR, DHSZ, DHSP and DHST, can trigger operations free from the scan cycle.

■ **Usage of double phase fourfold frequency input high-speed counter**

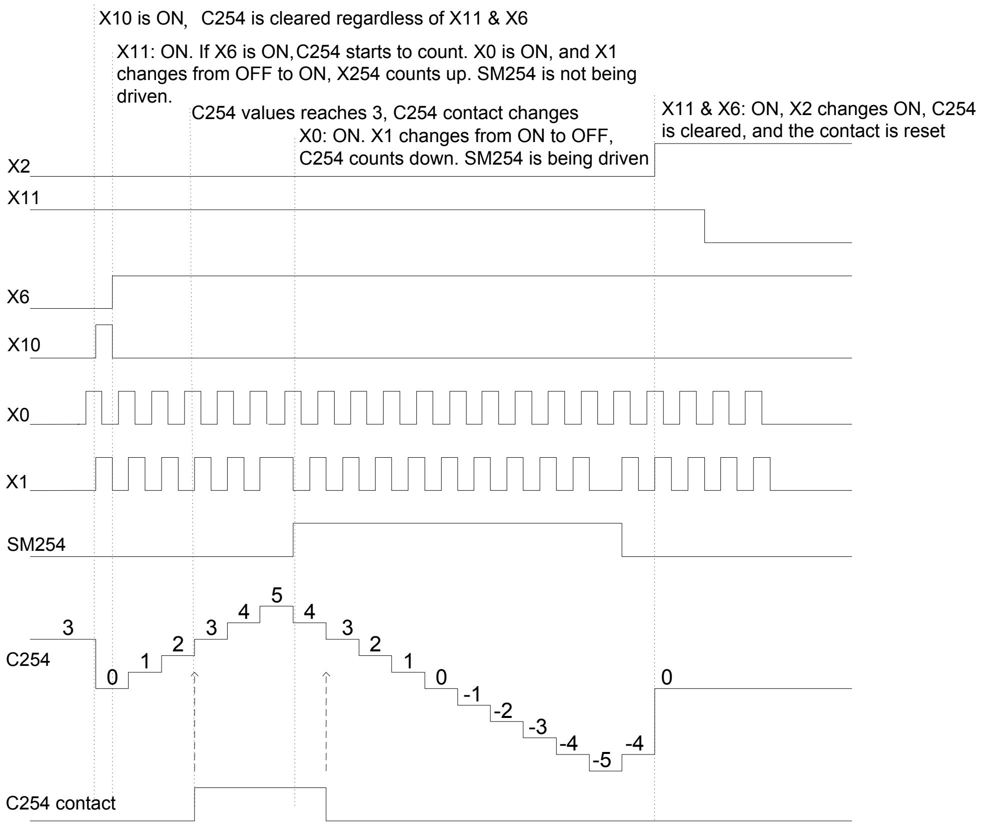
The double phase fourfold frequency input high-speed counter starts to count only when the pulse two inputs change from OFF to ON and ON to OFF. The phase difference of the two pulse inputs determines the counting direction, which is monitored by the corresponding SM element.

Example:



```
 SM0        SM100
-| |-------( )
 SM254      Y3
-| |-------( )
 X11                 4
-| |-------[ HCNT   C254      3        ]
 C254       Y2
-| |-------( )
 X10                 ON
-| |-------[ RST    C254      ]
```

The time sequence chart of the contacts action in the program is shown in the following figure:


X10 is ON, C254 is cleared regardless of X11 & X6
X11: ON. If X6 is ON,C254 starts to count. X0 is ON, and X1 changes from OFF to ON, X254 counts up. SM254 is not being driven.
C254 values reaches 3, C254 contact changes
X0: ON. X1 changes from ON to OFF, C254 counts down. SM254 is being driven
X11 & X6: ON, X2 changes ON, C254 is cleared, and the contact is reset
X2
X11
X6
X10
X0
X1
SM254
C254
3
0
1
2
3
4
5
4
3
2
1
0
-1
-2
-3
-4
-5
-4
0
C254 contact

Note:
1. Counter input points: X0 & X1.
2. High-speed counters, when used in instructions DHSCS, DHSCR, DHSZ, DHSP and DHST, can trigger operations free from the scan cycle.

8.1.4 Points to note about high-speed counters in EC20 and EC10 series

- **Classification**

C236, C237, C246 and C251 can be used as both hardware counters and software counters, depending on the modes in which they are used. All the other high-speed counters are software counters.

- **Maximum combined frequency**

1. The maximum combined frequency, or the sum of frequencies of all signals input at any time, should not exceed 80kHz on the following two occasions:

When multiple high-speed counters (hardware counting mode) are used simultaneously.

When the high-speed counters (hardware counting mode) and the SPD instruction are used at the same time.

2. The maximum combined frequency when multiple software high-speed counters, or when high-speed counters and the SPD instruction are used at the same time, is shown in the following table:

Conditions	Maximum combined frequency
Instructions DHSCS, DHSCR, DHSCI, DHSZ, DHSP and DHST are not used	≤200kHz
Instructions DHSCS, DHSCR, DHSCI, DHSP and DHST are used	≤30kHz
Instruction DHSZ is used	≤20kHz

- **Maximum frequency of hardware counter**

Counters C236, C237, C246 and C251 are the only four potential hardware counters. Among which:

C236, C237 and C246 are single phase counters. Their maximum counting frequency is 50kHz.

C251 is a double phase counter. Its maximum counting frequency is 30kHz.

- **Maximum frequency of software counter**

The high-speed counters used in instructions DHSCS, DHSCR, DHSCI, DHSP or DHST are all in software counting mode. The maximum input frequency for the single phase counters is 10kHz; for double phase counters: 5kHz.

When used in the DHSZ instruction, the maximum frequency for the single phase counters is 5kHz; for double phase counters: 4kHz.

8.2 External pulse capture function

The input points that provides the external pulse capture function are X0~X7. The corresponding SM elements are listed below:

Input point	Corresponding SM element
X0	SM90
X1	SM91
X2	SM92
X3	SM93
X4	SM94
X5	SM95
X6	SM96
X7	SM97

Note

1. When the output input point changes from OFF to ON, the SM element of the corresponding terminal will be set to ON.

2. SM90~SM97 will be cleared when the user program starts.

3. When using pulse capture, satisfy the maximum combined frequency of PLC series or abnormalities may occur.

4. If high-speed counters or SPD instructions are used on the same input point, the pulse capture function will become invalid after the first scan cycle, regardless of the validity of the instructions.

8.3 Points to note about high-speed input application

X0~X7 are input signals in the functions of high-speed counters, SPD instructions, pulse capture and external interrupts. Because different functions may use one or multiple input points, the functions cannot be used at the same time. During PLC programming, corresponding various functions of each input point can be applied one. If X0~X7 are used repeatedly in user program, the program cannot pass compiling.

The following table illustrates the functions of X0~X7 in high-speed counters, SPD instructions, pulse capture and external interrupts.

Counter \ Input point		X0	X1	X2	X3	X4	X5	X6	X7	Max. frequency (kHz)			
										EC20H	EC20	EC10	EC10V
Single phase single point input mode	C236	Up/down								100	100		
	C237		Up/down										
	C238			Up/down							10		
	C239				Up/down								
	C240					Up/down							
	C241						Up/down						
	C301							Up/down			/		
	C302								Up/down				
	C242	Up/down		Reset							10		
	C243				Up/down		Reset						
	C244	Up/down		Reset				Start					
	C245				Up/down		Reset		Start				
Single phase up/ down input mode	C246	Up	Down							100	50		
	C247	Up	Down	Reset							10		
	C303				Up	Down					/		
	C248				Up	Down	Reset				10		
	C249	Up	Down	Reset				Start					
	C250				Up	Down	Reset		Start				
Double	C251	Ph A	Ph B							50	30		

Counter	Input point	X0	X1	X2	X3	X4	X5	X6	X7	Max. frequency (kHz)			
										EC20H	EC20	EC10	EC10V
phase input mode	C304			Ph A	Ph B						/		
	C305					Ph A	Ph B						
	C306							Ph A	Ph B				
	C252	Ph A	Ph B	Reset							5		
	C253				Ph A	Ph B	Reset						
	C254	Ph A	Ph B	Reset				Start					
	C255				Ph A	Ph B	Reset		Start				
SPD instruction		Input point	Input point	Input point	Input point	Input point	Input point	Input point	Input point	100	10		
Pulse capture function		Input point	Input point	Input point	Input point	Input point	Input point	Input point	Input point	/	/		
External interrupt No. (rising/falling edge)		0/10	1/11	2/12	3/13	4/14	5/15	6/16	7/17	/	/		

Chapter 9 Using interrupts

This chapter details the mechanism, processing procedures and usage of various interrupts.

9.1 Interrupt program

When an interrupt event occurs, the normal scan cycle will be interrupted and the interrupt program will be executed, which is called the interrupt mechanism. For the event-triggered control tasks that requires priority, you often need to use this special mechanism.

The system provides many kinds of programmable interrupt resources. Each kind of interrupt resource can trigger a type of interrupt events, and each type of interrupt event are independently numbered.

In order to deal with a certain interrupt event, you must compile a processing program, that is, an interrupt program, which is an independent POU in the user program.

An event number must be designated for each interrupt program in order to link the interrupt program with the interrupt event designated with the event SN. When responding to the interrupt request of the interrupt event, the system will call the corresponding interrupt program based on the interrupt event number.

The following are the interrupt resources provided by EC series small PLC:

Event No.	Interrupt event	Enabling SM	Event No.	Interrupt event	Enabling SM
0	X0 input rising edge interrupt	SM40	28	Timer interrupt 2	Setting: SD68 Enabling: SM68
1	X1 input rising edge interrupt	SM41	29	Power failure interrupt	SM56
2	X2 input rising edge interrupt	SM42	30	Character sending interrupt of communication port 0	SM48
3	X3 input rising edge interrupt	SM43	31	Character receiving interrupt of communication port 0	SM49
4	X4 input rising edge interrupt	SM44	32	Frame sending interrupt of communication port 0	SM50
5	X5 input rising edge interrupt	SM45	33	Frame receiving interrupt of communication port 0	SM51
6	X6 input rising edge interrupt	SM46	34	Character sending interrupt of communication port 1	SM52
7	X7 input rising edge interrupt	SM47	35	Character receiving interrupt of communication port 1	SM53
8	Frame sending interrupt of COM2	SM59	36	Frame sending interrupt of communication port 1	SM54
9	Frame receiving interrupt of COM2	SM60	37	Frame receiving interrupt of communication port 1	SM55
10	X0 input falling edge interrupt	SM40	38	Character sending interrupt of COM2	SM57
11	X1 input falling edge interrupt	SM41	39	Character receiving interrupt of COM2	SM58
12	X2 input falling edge interrupt	SM42	40	High-speed counter interrupt 6	SM65
13	X3 input falling edge interrupt	SM43	41	High-speed counter interrupt 7	SM65
14	X4 input falling edge interrupt	SM44	42	PTO (Y2) output completion interrupt	SM72
			43	PTO (Y3) output completion interrupt	SM73
15	X5 input falling edge interrupt	SM45	44	PTO (Y4) output completion interrupt	SM74
16	X6 input falling edge interrupt	SM46	45	PTO (Y5) output completion interrupt	SM75

Event No.	Interrupt event	Enabling SM	Event No.	Interrupt event	Enabling SM
17	X7 input falling edge interrupt	SM47	46	PTO (Y6) output completion interrupt	SM76
18	PTO (Y0) output completion interrupt	SM63	47	PTO (Y7) output completion interrupt	SM77
19	PTO (Y1) output completion interrupt	SM64	50	Interpolation completion interrupt 1 (Y0,Y2)	SM69
20	High-speed counter interrupt 0	SM65	51	Interpolation completion interrupt 2 (Y4,Y5)	SM78
21	High-speed counter interrupt 1	SM65	52	Interpolation completion interrupt 3 (Y6,Y7)	SM79
22	High-speed counter interrupt 2	SM65	53	High-speed output passed position interrupt 1	SM61
23	High-speed counter interrupt 3	SM65	54	High-speed output passed position interrupt 2	SM62
24	High-speed counter interrupt 4	SM65	55	High-speed output passed position interrupt 3	SM105
25	High-speed counter interrupt 5	SM65	56	High-speed output passed position interrupt 4	SM106
26	Timer interrupt 0	Setting: SD66 Enabling: SM66	57	High-speed output passed position interrupt 5	SM107
27	Timer interrupt 1	Setting: SD67 Enabling: SM67	58	High-speed output passed position interrupt 6	SM108

9.2 Processing interrupt event

1. When a certain interrupt event occurs, if it is enabled, its corresponding event number will be added to the interrupt request queue, which is 8-record long and FIFO.

2. Processing of the interrupt request by system:

1) If the system detects that any request in the interrupt queue, it will stop the normal execution of user program.

2) The system will read in the request queue the head record, which is the number of the first interrupt event. The interrupt program corresponding to the event number will be called and executed.

3) When the interrupt program is finished, the corresponding head record of the request queue will be deleted, and all the following records will take one step foward.

4) The system will repeat these procedures until the queue is empty.

5) When the interrupt request queue is null, the system will continue to execute the interrupted main program.

3. The system can handle only one interrupt request at one time. When the system is processing an interrupt request, a new interrupt event will be added to the interrupt request queue rather than being responded immediately. The system will process it after all the requests ahead of it in the queue are processed.

4. When there are 8 records in the interrupt request queue, the system will automatically mask the new interrupt event so that no new requests will be added to the queue. The mask will not be cancelled until all the requests in the queue are processed and the interrupted main program is executed.

Note

1. The interrupts should be brief, or abnormalities may occur, including the mask of other interrupt events (missing of interrupt requests), system scan overtime and low execution efficiency of main program.

2. It is prohibited to call other subprograms in the interrupt program.

3. If you want to refresh I/O immediately during the interrupt, use the REF instruction. Note that the execution time of REF is related to the number of the I/Os to be refreshed.

4. An interrupt event can generate an interrupt request only when the corresponding interrupt event is enabled (which requires setting the corresponding SM element ON), and the global interrupt enable flag shall be on.

5. When an interrupt request with no corresponding interrupt program in the user program is generated, the request will be responded to, but the response is empty.

9.3 Timer interrupt

■ Description

The timer interrupt is the interrupt event generated by the system from time to time based on the user setting.

The timer interrupt program is applicable to the situation that requires timing and immediate processing by the system, such as the timing sampling of analog signals, and timing updating analog output according to certain waveform.

You can set the intervals (unit: ms) for the timer interrupts by setting the corresponding SD elements. The system will generate the interrupt event when the set time interval is reached (recommended minimum interval: > 4ms).

The ON/OFF status of certain SM elements can enable/disable the corresponding timer interrupts.

The system provides 3 kinds of timer interrupt resources.

Table 9-1 Timer interrupt resource list

Timer interrupt	Interrupt event No.	Intervals of timer interrupt (SD)	Enable control (SM)
0	26	SD66	SM66
1	27	SD67	SM67
2	28	SD68	SM68

🕮 Note

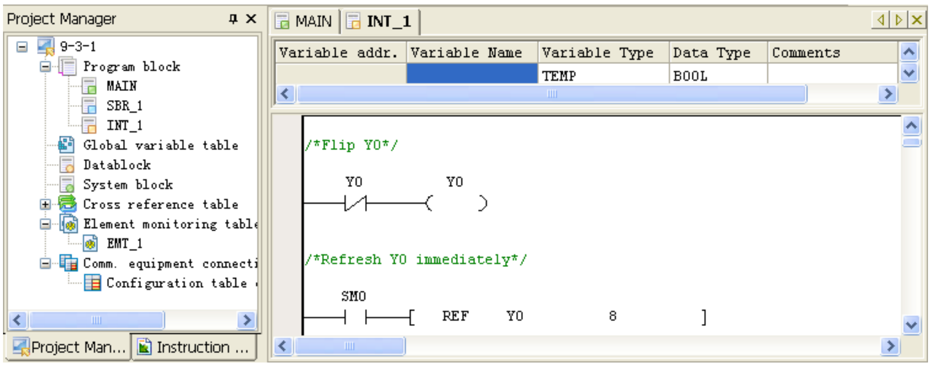
1. Setting of enable control elements cannot affect the exection of the timed interrupts in the interrupt request queue.
2. The timing for a re-enabled interrupt will start from zero.

To change the interval of the timer interrupt when the program is running, it is recommended to follow the following procedures: disable the timer interrupt, change the interval and enable the timer interrupt.

■ Example

This example uses timer interrupt 0 to flip the Y0 output once a second, which makes Y0 flash.

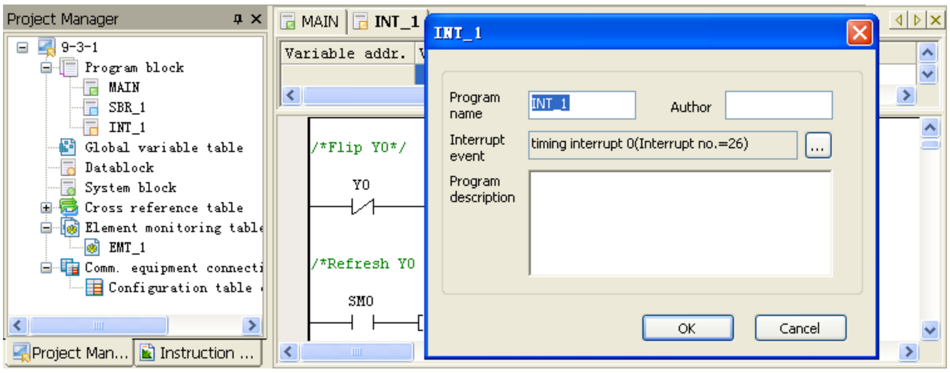
1. Compile an interrupt program for the interrupt event.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	

```
/*Flip Y0*/
   Y0          Y0
──┤/├───────(    )

/*Refresh Y0 immediately*/
   SM0
──┤ ├─────[  REF    Y0        8       ]
```

2. Specify an interrupt event number for the interrupt program:



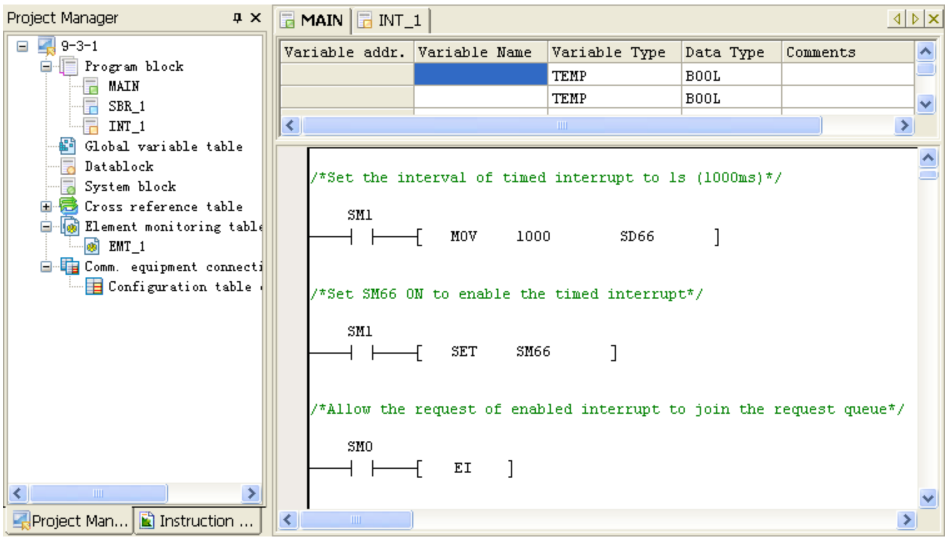
Program name: INT_1

Author:

Interrupt event: timing interrupt 0(Interrupt no.=26)

Program description:

3. Set the interval for the timer interrupt and enable the timer interrupt in the main program.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	

```
/*Set the interval of timed interrupt to 1s (1000ms)*/
    SM1
──┤ ├──[ MOV    1000      SD66      ]

/*Set SM66 ON to enable the timed interrupt*/
    SM1
──┤ ├──[ SET    SM66      ]

/*Allow the request of enabled interrupt to join the request queue*/
    SM0
──┤ ├──[ EI    ]
```

9.4 External interrupt

■ Description

The external interrupt is related to the actual PLC input points. It is classified into input rising edge interrupt and input falling edge interrupt. In the user program, add the actions related to external event to the external interrupt program. The highest response frequency of the system to the external event is 1K. The external events over 1K may be lost. The rising edge interrupt and falling edge interrupt cannot be used on the same port simultaneously. All the external interrupts are only valid when the global interrupt control EI and corresponding enabling SM are valid.

The detailed relationship is as follows:

Interrupt number	Enabling element		Interrupt number	Enabling element
0 or 10	SM40		4 or 14	SM44
1 or 11	SM41		5 or 15	SM45
2 or 12	SM42		6 or 16	SM46
3 or 13	SM43		7 or 17	SM47

The external interrupts are numbered as follows:

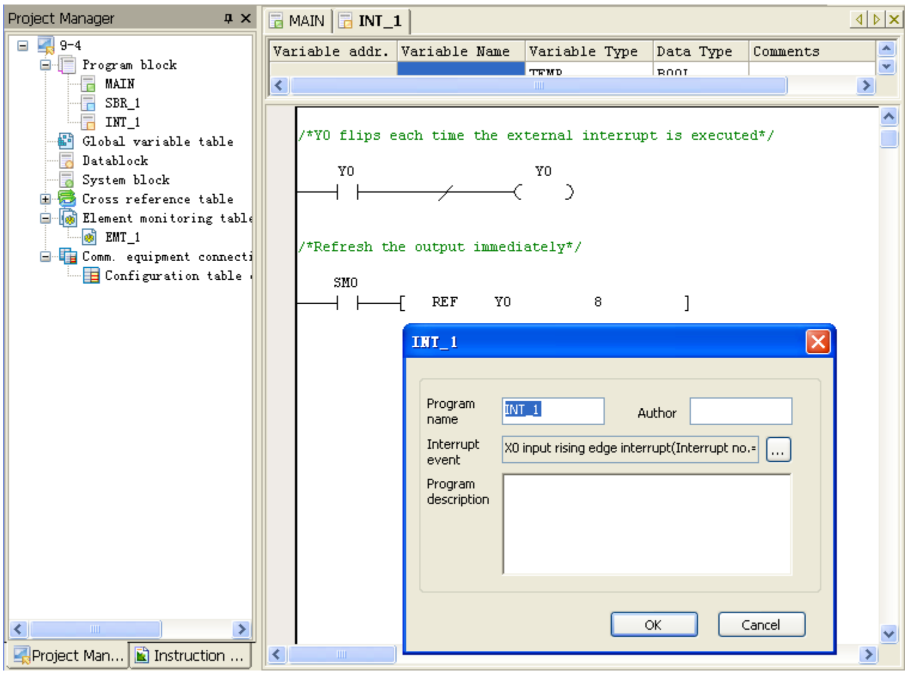
Interrupt number	Interrupt source		Interrupt number	Interrupt source
0	X0 input rising edge interrupt		9	Reserved
1	X1 input rising edge interrupt		10	X0 input falling edge interrupt
2	X2 input rising edge interrupt		11	X1 input falling edge interrupt
3	X3 input rising edge interrupt		12	X2 input falling edge interrupt
4	X4 input rising edge interrupt		13	X3 input falling edge interrupt
5	X5 input rising edge interrupt		14	X4 input falling edge interrupt
6	X6 input rising edge interrupt		15	X5 input falling edge interrupt
7	X7 input rising edge interrupt		16	X6 input falling edge interrupt
8	Reserved		17	X7 input falling edge interrupt

The single input pulse frequency of X0-X7 is less than 200Hz.

■ Example

In the example, the system upsets the output of Y0 based on the corresponding external interrupt 0 function and rising edge input event of X0.

1. Compile the interrupt program to flip Y0 status once upon every interrupt and output immediately. To use an interrupt, you should select its corresponding interrupt number. See the following figure for the specific operation.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	

```
/*Y0 flips each time the external interrupt is executed*/

 Y0                        Y0
-| |-----------/---------(    )

/*Refresh the output immediately*/

 SM0
-| |---[ REF    Y0        8        ]
```

INT_1

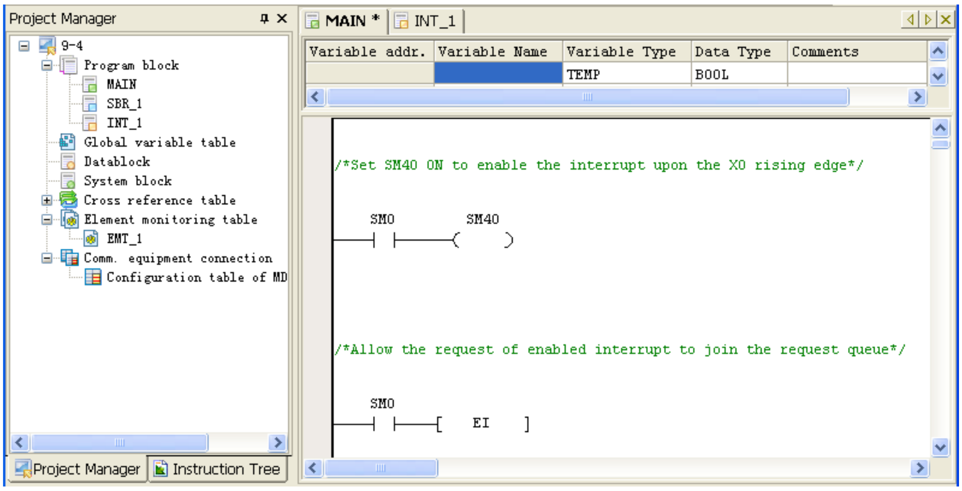
Program name: INT_1

Author:

Interrupt event: X0 input rising edge interrupt(Interrupt no.=

Program description:

2. Write EI instruction in the main program, and set SM40, the interrupt enabling flag of X0 input rising edge interrupt, valid.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	

```
/*Set SM40 ON to enable the interrupt upon the X0 rising edge*/

 SM0       SM40
-| |------(    )

/*Allow the request of enabled interrupt to join the request queue*/

 SM0
-| |---[  EI   ]
```

9.5 High-speed counter interrupt

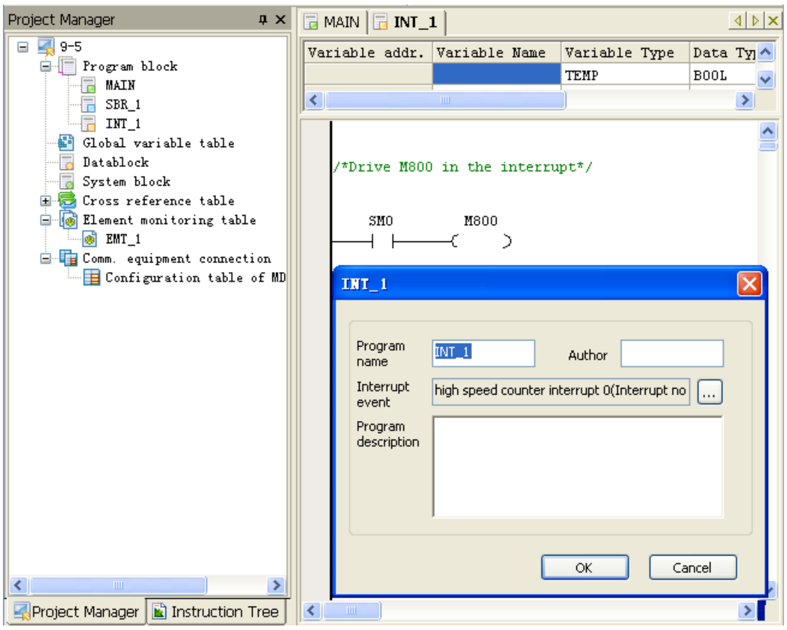
■ Description

The high-speed counter interrupt must be used together with the HCNT instruction or DHSCI instruction, and generates high-speed counter interrupt based on the value of the high-speed counter. You can compile programs related to external pulse input in the high-speed interrupt program. The high-speed counter interrupts (20~25) are valid only when the EI instruction and corresponding interrupt enable flag are valid.

■ Example

This example uses the high-speed counter function of X0 to call the interrupt program (number 20) when the external counter C236 reaches the value specified through the DHSCI instruction.

1. Compile interrupt program, choose an interrupt number for each interrupt subprogram. See the following figure for the specific operation.



Variable addr.	Variable Name	Variable Type	Data Ty
		TEMP	BOOL

```
/*Drive M800 in the interrupt*/

  SM0         M800
--| |--------(    )
```

INT_1

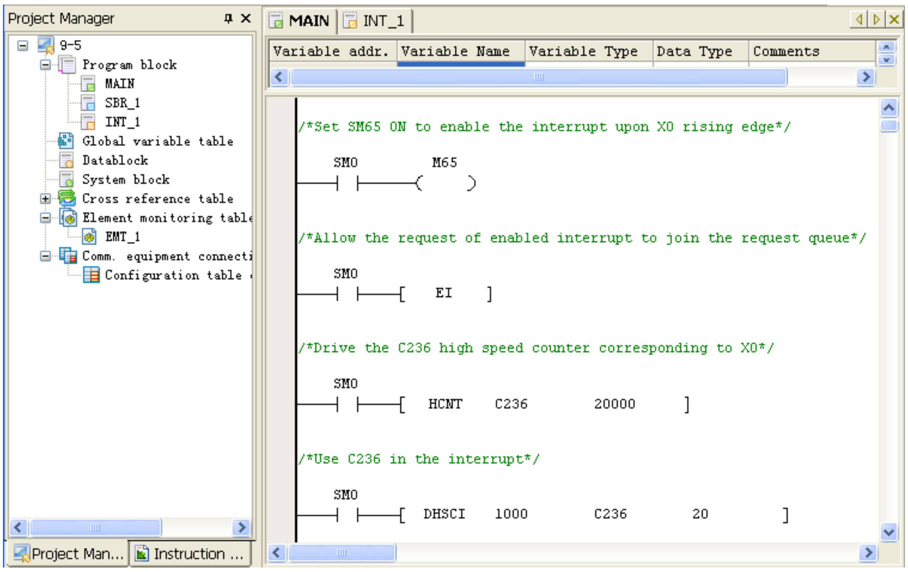
Program name: INT_1

Author:

Interrupt event: high speed counter interrupt 0(Interrupt no

Program description:

2. Write EI instruction in the main program, and set SM65, the interrupt enabling flag of high-speed counter interrupt, valid. Drive the high-speed counter C236 and high-speed counter interrupt instruction.



Variable addr.	Variable Name	Variable Type	Data Type	Comments

```
/*Set SM65 ON to enable the interrupt upon X0 rising edge*/

  SM0          M65
--| |--------(    )

/*Allow the request of enabled interrupt to join the request queue*/

  SM0
--| |-----[   EI   ]

/*Drive the C236 high speed counter corresponding to X0*/

  SM0
--| |-----[  HCNT     C236        20000      ]

/*Use C236 in the interrupt*/

  SM0
--| |-----[  DHSCI    1000        C236        20          ]
```

9.6 PTO output completion interrupt

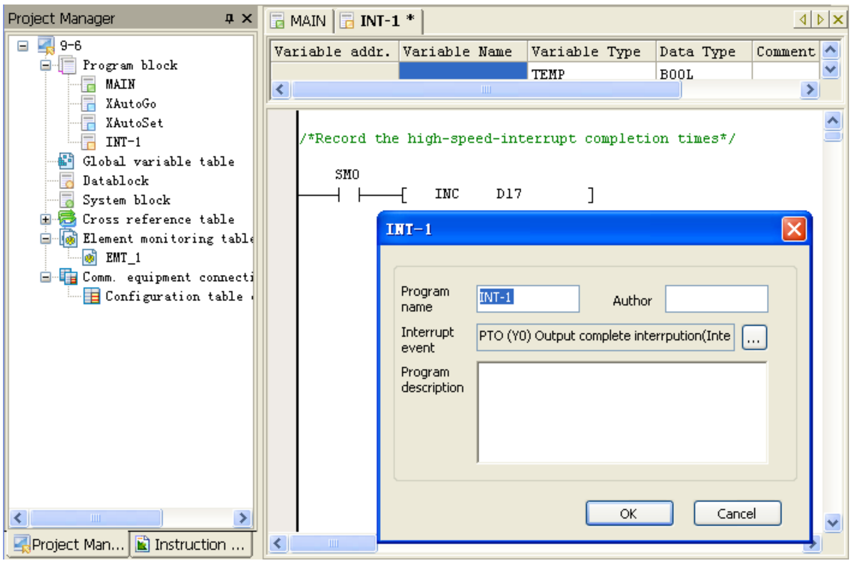
■ Description

The PTO output completion interrupt is triggered when enable flag (SM63 or SM64) is set and the high-speed pulse output at Y0 or Y1 is finished. You can carry out the relevant processing in the interrupt subprogram. This function is applicable only to EC10 series PLC.

■ Example

This example uses the high-speed pulse output of Y0 to call the interrupt program (number 18) after Y0 high-speed pulse output is finished.

1. Code function in interrupt program (INT_1): Compile program for the interrupt code to realize the control. Choose the corresponding interrupt number for each interrupt. See INT_1 for the specific operation.



Variable addr.	Variable Name	Variable Type	Data Type	Comment
		TEMP	BOOL	

```
/*Record the high-speed-interrupt completion times*/

 SM0
──┤ ├──[ INC    D17       ]
```

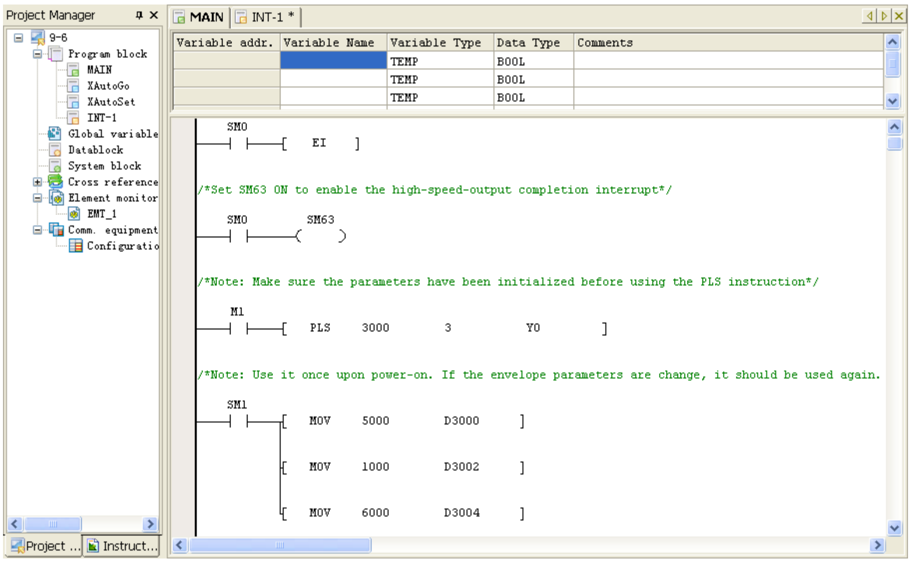
INT-1

Program name: INT-1   Author:

Interrupt event: PTO (Y0) Output complete interrpution(Inte

Program description:

2. Code function in main program: Enable the global interrupt of the system and the enable flag SM63 of PTO output interrupt. Use PLS instruction.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	
		TEMP	BOOL	

```
 SM0
──┤ ├──[ EI   ]

/*Set SM63 ON to enable the high-speed-output completion interrupt*/

 SM0        SM63
──┤ ├──────(    )

/*Note: Make sure the parameters have been initialized before using the PLS instruction*/

 M1
──┤ ├──[ PLS    3000      3        Y0       ]

/*Note: Use it once upon power-on. If the envelope parameters are change, it should be used again.

 SM1
──┤ ├──[ MOV    5000      D3000     ]
      │
      [ MOV    1000      D3002     ]
      │
      [ MOV    6000      D3004     ]
```

9.7 Power failure interrupt

When the enable flag of SM56 is set and the main module has detected the power failure, the power failure interrupt will be triggered and the user can carry out the relevant processing in the interrupt subprogram. This function is applicable only to EC10 series PLC.

As the power failure interrupt subprogram is executed when the system has no external power supply, the execution duration of power failure interrupt subprogram shall not be over 5ms. Otherwise, the power failure retention component cannot be completely saved.

9.8 Serial port interrupt

■ Description

Serial port interrupt: Under the free port protocol mode of serial port, the system will generate interrupt event based on the sending or receiving events of serial port.

For each serial port, the system supports 4 interrupt resources for the user. The interrupt program of serial port is mainly used when special processing is required for the receiving and sending of character/frame at the serial port and timely processing is requested. It is able to respond to the processing of completing character/frame XMT/RCV without being influenced by scanning time.

Set the ON/OFF status of SM component and the serial port interrupt can be enabled or disabled. When the serial port interrupt is disabled, the ones that have been added to the interrupt queue will continue to be executed.

Do not call the XMT instruction of serial port in the processing subprogram of character sending interrupt when the power flow is normally on. Otherwise, it may lead to interrupt subprogram nesting which blocks the execution of user program.

Interrupt of frame receiving and sending refers to the interrupt event that is delivered after the XMT and RCV instructions of the serial port are executed.

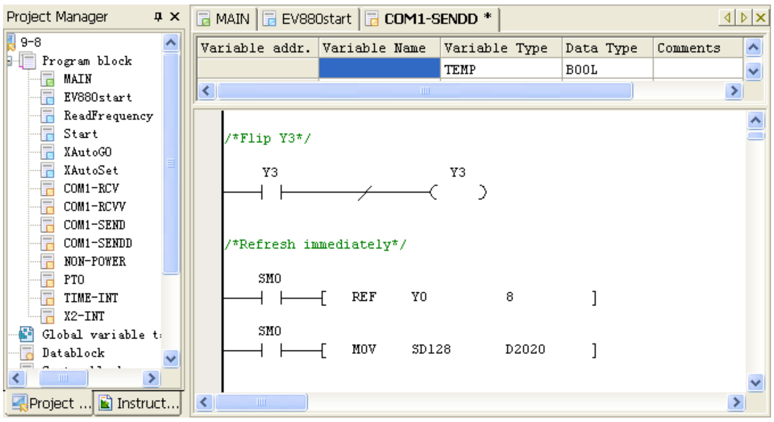
Serial port interrupt resource list:

Event No.	Corresponding interrupt event	Interrupt enabling SM
30	Character sending interrupt of communication port 0	SM48
31	Character receiving interrupt of communication port 0	SM49
32	Frame sending interrupt of communication port 0	SM50
33	Frame receiving interrupt of communication port 0	SM51
34	Character sending interrupt of communication port 1	SM52
35	Character receiving interrupt of communication port 1	SM53
36	Frame sending interrupt of communication port 1	SM54
37	Frame receiving interrupt of communication port 1	SM55
8	Character sending interrupt of communication port 2	SM59
9	Character receiving interrupt of communication port 2	SM60
38	Frame sending interrupt of communication port 2	SM57
39	Frame receiving interrupt of communication port 2	SM58

■ **Example**

In the example, with the sending interrupt function of serial port frame, the system will flip Y3 output once when a frame is sent out and generate flashing effect based on the frequency of the character sending frame.

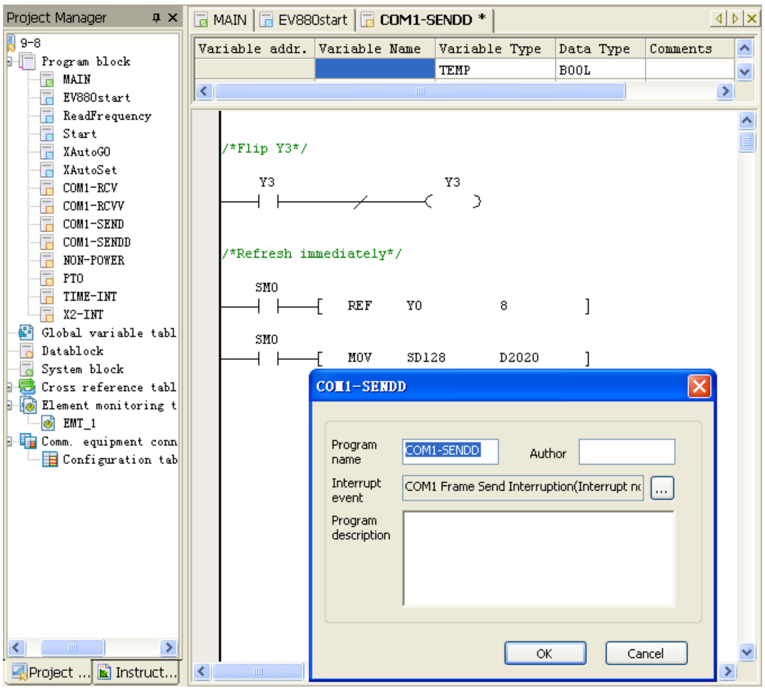
1. Compile interrupt program and the processing code when the serial port sending frame is completed and the interrupt is triggered.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	

```
/*Flip Y3*/
  Y3                        Y3
──┤ ├──────────/──────────(    )

/*Refresh immediately*/
  SM0
──┤ ├───[  REF    Y0        8        ]

  SM0
──┤ ├───[  MOV    SD128     D2020    ]
```

2. Specify interrupt event number for the interrupt program:



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	

```
/*Flip Y3*/
   Y3                        Y3
---| |--------/--------(     )

/*Refresh immediately*/
   SM0
---| |---[  REF    Y0         8        ]
   SM0
---| |---[  MOV    SD128      D2020    ]
```

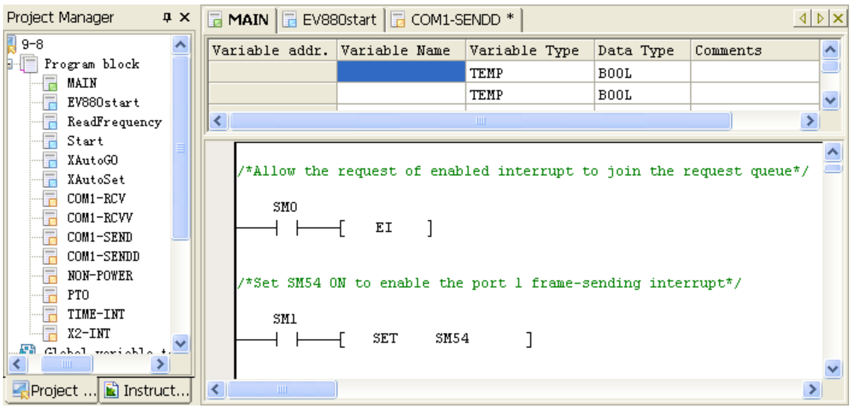
COM1-SENDD

Program name: COM1-SENDD   Author:

Interrupt event: COM1 Frame Send Interruption(Interrupt n

Program description:

3. Compile the code of the sending frame interrupt of enable serial port in the main program.



Variable addr.	Variable Name	Variable Type	Data Type	Comments
		TEMP	BOOL	
		TEMP	BOOL	

```
/*Allow the request of enabled interrupt to join the request queue*/
   SM0
---| |---[  EI   ]

/*Set SM54 ON to enable the port 1 frame-sending interrupt*/
   SM1
---| |---[  SET    SM54     ]
```

For the detailed example of serial port interrupt, please refer to *Chapter 10 Using communication function*.

9.9 Measure short time pulse

This function is applicable only to EC10 series PLC.

The high-speed ring counter offers the counting function of high accuracy in 0.1ms, which can match with input interrupts to measure short time pulse width.

The relevant elements are as follows:

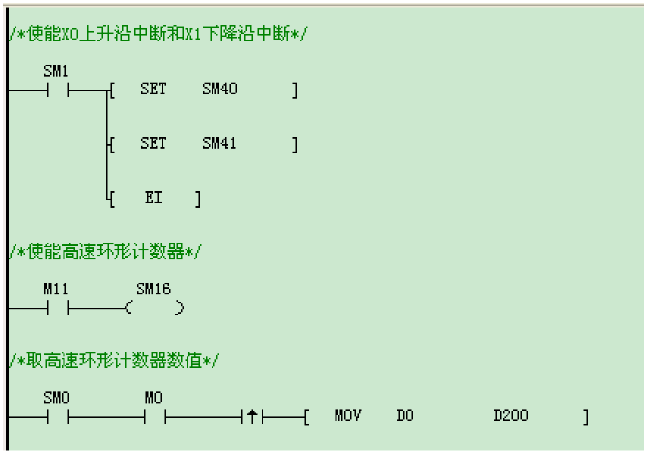
	Name	Function	Attribute	Range
SM16	Enabling flag of high-speed ring counter	Unit: 0.1ms, 16 bits Setting: High-speed ring counter starts counting Clearing: High-speed ring counter stops counting	R/W	
SD16	High-speed ring counter	0-20971 (Unit: 0.1ms, 16 bits) up counting. The high-speed ring counter will count up for 0.1m clock next operation cycle after SM16 is set, and restart from zero when over 20971. The error depends on the time for executing a single instruction.	R/W	0-20971

Before using high-speed ring counter, reset SD16.

The following is an example of measuring pulse width. Connect the signals that need measurement to X0 and X1 terminals, and set the interrupt optimal setting of X0 and X1 to the high level.

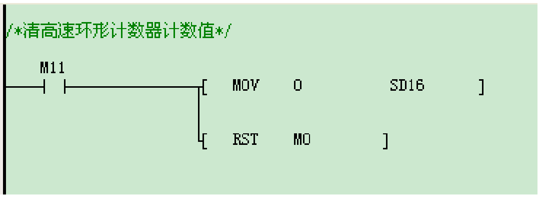
D200 specifies the measured pulse width (Unit: 0.1ms)

Main program:



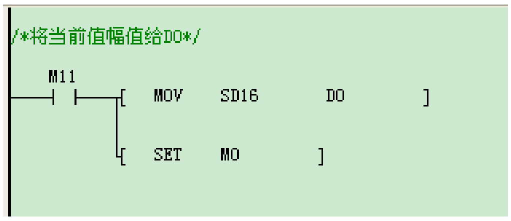
```
/*使能X0上升沿中断和X1下降沿中断*/
  SM1
──┤ ├──┬[ SET   SM40   ]
       ├[ SET   SM41   ]
       └[ EI  ]
/*使能高速环形计数器*/
  M11       SM16
──┤ ├────(      )
/*取高速环形计数器数值*/
  SM0       M0
──┤ ├──────┤ ├──────┤↑├──[ MOV   D0      D200   ]
```

X0 rising edge interrupt



```
/*清高速环形计数器计数值*/
  M11
──┤ ├──────────────┬[ MOV   0       SD16   ]
                   └[ RST   M0      ]
```

X1 falling edge interrupt



```
/*将当前值幅值给D0*/
  M11
──┤ ├──┬[ MOV   SD16     D0     ]
       └[ SET   M0      ]
```

Chapter 10 Using communication function

This chapter introduces the communication function of EC series small PLC, including the communication resources and communication protocols, and uses examples to illustrate.

Communication resource

The baud rates applicable to EC series small PLC are listed in the following table:

Communication protocol	Available baud rate
Free port protocol, Modbus communication protocol	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200
N:N communication protocol	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200

The communication protocols that EC series small PLC supports are listed in the following table:

Main module	Communication port	Port type	Available protocol
EC20	Communication port 0	RS232	Programming port protocol, free port protocol, Modbus communication protocol (slave station), N:N bus communication protocol (master station, slave station)
	Communication port 1	RS232 or RS485	Free port protocol, Modbus communication protocol (master station, slave station), N:N bus communication protocol (master station, slave station)
EC10	Communication port 0	RS232	Programming port protocol, free port protocol, Modbus communication protocol (slave station), N:N bus communication protocol (master station, slave station)
	Communication port 1	RS232 or RS485	Free port protocol, Modbus communication protocol (master station, slave station), N:N bus communication protocol (master station, slave station)
EC10V	Communication port 0	RS232	Programming port protocol, free port protocol, Modbus communication protocol (slave station), N:N bus communication protocol (master station, slave station)
	Communication port 1	RS485	Free port protocol, Modbus communication protocol (master station, slave station), N:N bus communication protocol (master station, slave station)
	Communication port 2	RS485	Free port protocol, Modbus communication protocol (master station, slave station), N:N bus communication protocol (master station, slave station)
EC20H	Communication port 0	RS232	Programming port protocol, free port protocol, Modbus communication protocol (slave station), N:N bus communication protocol (master station, slave station)
	Communication port 1	RS485	Free port protocol, Modbus communication protocol (master station, slave station), N:N bus communication protocol (master station, slave station)
	Communication port 2	RS485	Free port protocol, Modbus communication protocol (master station, slave station), N:N bus communication protocol (master station, slave station)

You can also set the mode selection switch of EC series PLC to TM to transfer port 0 to programming port protocol.

10.2 Programming port protocol

The programming port protocol is an internal protocol dedicated to the communication between Programmer and main module.

10.3 Free port communication protocol

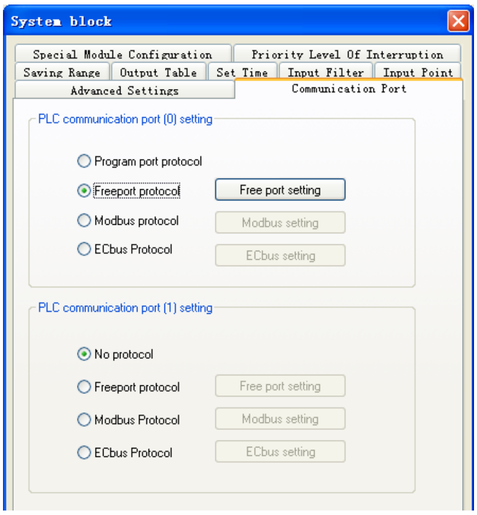
10.3.1 Introduction

The free port protocol is a communication mode with user-defined data file format. It supports two data formats: ASCII&binary. The free port protocol realizes data communication through instructions and can only be used when PLC is in RUN state.

The free port communication instructions include XMT (sending instruction) and RCV (receiving instruction).

10.3.2 Parameter setting

Select **Communication Port** in the **System block** dialogue box, and select **Freeport protocol** in port 0 or port 1 setting area to enable the **Freeport setting** button as follows:



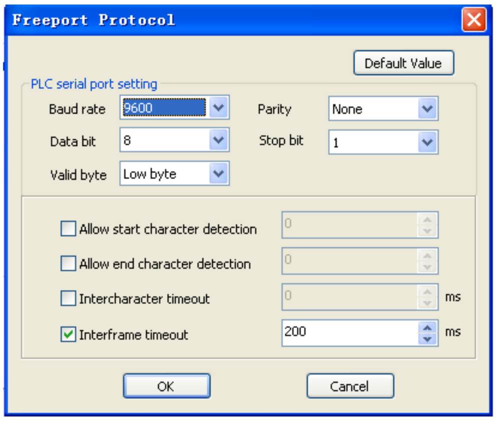
PLC communication port (0) setting

○ Program port protocol
● Freeport protocol
○ Modbus protocol
○ ECbus Protocol

PLC communication port (1) setting

● No protocol
○ Freeport protocol
○ Modbus Protocol
○ ECbus Protocol

The parameter setting of free port is as follows:



PLC serial port setting

Baud rate: 9600
Parity: None
Data bit: 8
Stop bit: 1
Valid byte: Low byte

□ Allow start character detection 0
□ Allow end character detection 0
□ Intercharacter timeout 0 ms
☑ Interframe timeout 200 ms

Configurable items are listed in the following table:

Item	Setting	Remark
Baud rate	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200, default: 9600	—
Data bit	7 or 8, default: 8	—
Parity	No check, odd, even, default: no check	—
Stop bit	1 or 2, default: 1	—
Allow start character detection	Enabled or disabled, default: disabled	—
Start character detection	0~255 (corresponding to 00~FF)	Start receiving after the designated start character is detected. Save the received characters (including the start character) to the designated BFM
Allow end character detection	Enabled or disabled, default: disabled	—
End character detection	0~255 (corresponding to 00~FF)	Stop receiving after the preset end character is received, and save the end character to the BFM
Intercharacter timeout enabling	Enabled or disabled, default: disabled	—
Intercharacter timeout	0~65535ms	Stop receiving if the interval between two received characters is longer than the timeout setting
Interframe timeout enabling	Enabled or disabled, default: disabled	—
Interframe timeout	0~65535ms	When the RCV power flow is valid and the communication conditions are met, the timing will start as soon as the communication serial port starts to receive. If a frame is not received completely when the set time is up, the reception ends

10.3.3 Free port instruction

- **Points to note**

The free port instructions XMT and RCV can be used to send/receive data to/from the designated communication port. For the usage of the free port instruction, refer to **Error! Reference source not found.Error! Reference source not found.** and **Error! Reference source not found.Error! Reference source not found.**.

Note that to use free port instruction on a certain port, you need to set the free port protocol and communication parameter for the communication port through the system block of AutoStation. In addition, you need to download the system setting to the PLC and restart it.

- **Example**

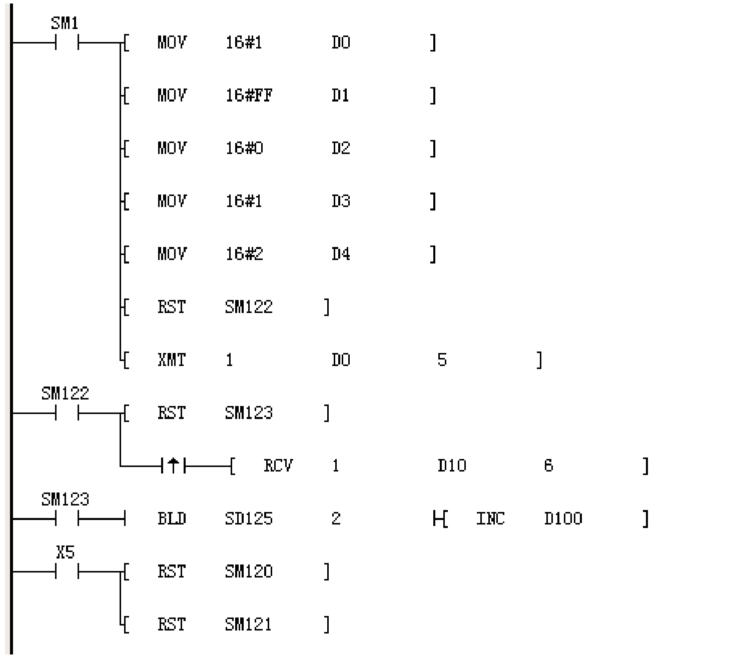
Example 1: Send a 5-byte data and then receive a 6-byte data through communication port 1.

The data to be sent:	01	FF	00	01	02

The data to be received:	01	FF	02	03	05	FE

Save the received data to D elements starting with D10. Each byte occupies one D element, as shown below:

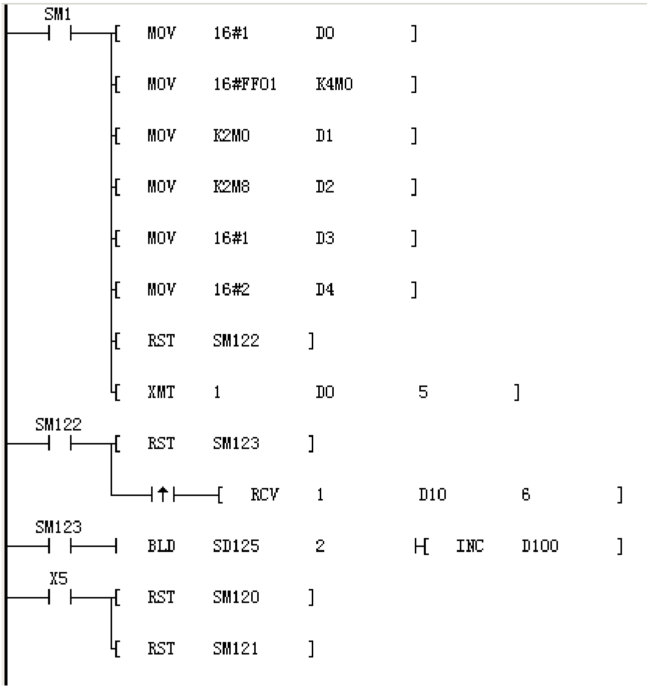
01	FF	02	03	05	FE
D10	D11	D12	D13	D14	D15



```
SM1
--| |--+-[ MOV   16#1    D0    ]
       +-[ MOV   16#FF   D1    ]
       +-[ MOV   16#0    D2    ]
       +-[ MOV   16#1    D3    ]
       +-[ MOV   16#2    D4    ]
       +-[ RST   SM122   ]
       +-[ XMT   1       D0     5     ]
SM122
--| |--+-[ RST   SM123   ]
       +--|↑|--[ RCV   1      D10     6     ]
SM123
--| |----  BLD   SD125   2      -[ INC   D100    ]
X5
--| |--+-[ RST   SM120   ]
       +-[ RST   SM121   ]
```

1. Change the setting of communication port in the system block to free port communication and set the related parameters.
2. When the power flow of SM1 is valid, save the to-be-sent data to the communication BFM starting with D0. Send data with XMT instruction and reset SM122 (transmission complete flag bit) before the sending.
3. Set SM122 after the transmission, and begin to receive data upon the rising edge. The maximum length for the received characters is 6.
4. Set SM123 after the data is received, and perform the corresponding operation based on the receiving completion information register (SD125).
5. Use X5 as the enable bit for interrupting the sending and receiving.

Example 2: Send and receive data through communication port 1.

 SM1 [MOV 16#1 D0] [MOV 16#FF01 K4M0] [MOV K2M0 D1] [MOV K2M8 D2] [MOV 16#1 D3] [MOV 16#2 D4] [RST SM122] [XMT 1 D0 5] SM122 [RST SM123] |↑|—[RCV 1 D10 6] SM123 BLD SD125 2 H[INC D100] X5 [RST SM120] [RST SM121]	Different from “Example 1”, when sending the high&low bytes of a word element, the element must be divided into high&low byte parts. For instance, if you want to send the content of D2, you can store its high byte and low byte separately in D3 and D4, and then send D3 and D4. You can also store the data in a K4MX (such as K4M0 of in this example) element. Take K2M0 as high byte and K2M8 as low byte.

10.4 Modbus communication protocol

10.4.1 Introduction

For the serial port communication of EC series small PLC, Modbus communication protocol is available. Two communication modes: ASCII and RTU (EC10 only supports RTU mode) are supported. The PLC can be set as the master or slave station.

10.4.2 Characteristics of links

1. Physical layer: RS232, RS485

2. Link layer: asynchronous transfer mode

1) Data bit: 7 bits (ASCII) or 8 bits (RTU)

2) Transfer rate: 1200, 2400, 4800, 9600, 19200, 38400

3) Check method: even check, odd check or no check

4) Stop bit: 1 or 2 stop bits

3. Networking configuration: up to 31 sets of equipment. Address range: 1~31. Broadcast is supported.

10.4.3 RTU transfer mode

1. Hexadecimal data.

2. The interval between two characters shall not be less than the time of 1.5 characters.

3. There is no frame head or tail, and the interval between two frames is at least the time of 3.5 characters.

4. Use CRC16 check.

5. The maximum length of RTU frame is 256 bytes and the frame structure is as follows:

Structure of frame	Address	Function code	Data	CRC
Number of bytes	1	1	0~252	2

6. Calculation of interval among characters:

If the communication baud rate is 19200, the interval of 1.5 characters is 1/19200×11×1.5×1000=0.86ms.

The interval of 3.5 characters is 1/19200×11×3.5×1000=2ms.

10.4.4 ASCII transfer mode

1. Use ASCII data communication.
2. The frame takes ": (3A)" as the head, and CRLF (0D 0A) as the tail.
3. The allowed interval among characters is 1s.
4. Use LRC check.
5. The frame of ASCII is longer than that of RTU. It is required two character codes for transferring one byte (HEX) in ASCII mode. The maximum length for data field (2×252) of ASCII is twice of RTU data field (252). The maximum length of ASCII frame is 513 characters and the structure of frame is as follows:

Structure of frame	Head	Address	Function code	Data	LRC	Tail
Number of bytes	1	2	2	0~2*252	2	2

10.4.5 Available Modbus function code

Available Modbus function codes include 01, 02, 03, 04, 05, 06, 15 and 16.

Note: 04 function code is only available for version 1.23 of EC10.

10.4.6 Addressing mode of PLC element

1. Relationship between read-write element function code and the element:

Function code	Name	Modicon data address	Type of operational element	Remark
01	Read coil	0[Note1]:xxxx	Y, X, M, SM, S, T, C	Bit read
02	Read discrete input	1[Note2]:xxxx	X	Bit read
03	Read register	4[Note3]:xxxx [注4]	D, SD, Z, T, C	Word read
05	Write single coil	0:xxxx	Y, M, SM, S, T, C	Bit write
06	Write single register	4:xxxx	D, SD, Z, T, C	Word write
15	Write multiple coils	0:xxxx	Y, M, SM, S, T, C	Bit write
16	Write multiple registers	4:xxxx	D, SD, Z, T, C	Word write

Note:
1. 0 means "coil".
2. 1 means "discrete input".
3. 4 means "register".
4. xxxx means range "1~9999". Each type has an independent logic address range of 1 to 9999 (protocol address starts from 0).
5. 0, 1 and 4 do not have the physical meaning and are not involved in actual addressing.
6. Users shall not write X element with function codes 05 and 15; otherwise, the system will not feed back the error information if the written operand and data are correct, but the system will not perform any operation on the write instruction.

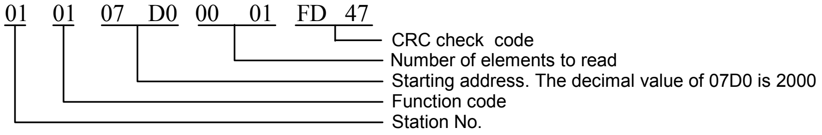
2. Relationship between PLC Element and Modbus communication protocol address:

Element	Type	Physical element	Protocol address	Available function code	Remark
Y	Bit	Y0~Y377 (octal code) 256 points in total	0000~0255	01, 05, 15	Output status, element code: Y0 toY7, Y10 toY17
X	Bit	X0~X377 (octal code) 256 points in total	1200~01455 0000~0255	01, 05, 15 02	Input status, it supports two kinds of address, the element code is same as above
M	Bit	M0~M2047 M2048~M10240	2000~4047 12000-20191	01, 05, 15	
SM	Bit	SM0~SM255 SM256~SM511	4400~4655 30000-30255	01, 05, 15	
S	Bit	S0~S1023 S1024~S4095	6000-7023 31000-34071	01, 05, 15	
T	Bit	T0~T255 T256~T511	8000~8255 11000-11255	01, 05, 15	Status of T element
C	Bit	C0~C255 C256~C306	9200~9455 10000-10050	01, 05, 15	Status of C element
D	Word	D0~D7999	0000~7999	03, 06, 16	

Element	Type	Physical element	Protocol address	Available function code	Remark
SD	Word	SD0~SD255 SD256~SD511	8000~8255 12000-12255	03, 06, 16	
Z	Word	Z0~Z15	8500~8515	03, 06, 16	
T	Word	T0~T255 T256~T511	9000~9255 11000-11255	03, 06, 16	Current value of T element
C	Word	C0~C199	9500~9699	03, 06, 16	Current value of C element (WORD)
C	Double word	C200~C255	9700~9811	03, 16	Current value of C element (DWORD)
C	Double word	C256~C306	10000-10101	03, 16	Current value of C element (DWORD)
R	Word	R0~R32767	13000-45767	03, 06, 16	

Note:

The protocol address is the address used on data transfer and corresponds with the logic address of Modicon data. The protocol address starts from 0 and the logic address of Modicon data begins with 1, that is, protocol address+1=logic address of Modicon data. For example, if M0 protocol address is 2000, and its corresponding logic address of Modicon data will be 0:2001. In practice, the read and write of M0 is completed through the protocol address, e.g.: read M0 element frame (sent from the master station):


01 01 07 D0 00 01 FD 47
CRC check code
Number of elements to read
Starting address. The decimal value of 07D0 is 2000
Function code
Station No.

10.4.7 Modbus slave

Modbus slave responds to the master station according to the received message of local address rather than send out message actively. The slave only supports Modbus function codes 01, 02, 03, 05, 06, 08, 15 and 16. The other codes are illegal function codes (except broadcast frame).

10.4.8 Read/write elements

All the function codes supported by EC20, except 08 are used for read and write elements. In principle, in one frame, there are 2000 bits and 125 words for reading, 1968 bits and 120 words for writing at most. However, the actual protocol addresses for elements of different types are different and discontinuous (e.g.: Y377's protocol address is 255, X0’s protocol address is 1200). Therefore, when reading or writing an element, the element read for one time can only be the same type, and the maximum number of the read elements depends on the elements of this type that are actually defined. For example, when reading element Y (Y0~Y377, 256 points in total), the protocol address ranges from 0 to 255, the corresponding logic address of Modicon data is from 1 to 256, and the maximum number of elements Y that can be read is 256.

The examples are as follows:

1. XMT from master station: 01 01 00 00 01 00 3D 9A

01- address; 01-function code; 00 00-starting address; 01 00-number of elements to read; 3D 9A-check

Response of slave station: provide correct response

2. XMT from master station: 01 01 00 00 01 01 FC 5A

The starting address for the reading of master station is 0000. 01 01 (257) elements are read, which is beyond the defined number of elements Y.

Response of slave station: 01 81 03 00 51

The data from the slave station response is illegal, because 257>256, and 256 is the allowed maximum number of elements Y.

3. XMT from master station: 01 01 00 64 00 A0 7D AD

The starting address for the reading of master station: 00 64 (decimal 100)

Number of elements read: 00 A0 (decimal 160)

Slave station response: 01 81 02 C1 91

The slave station responds with illegal data address, because there are only 156 elements Y starting with the protocol address 100, but 160>156, 160 is illegal.

4. XMT from master station: 01 04 00 02 00 0A D1 CD

The frame of XMT function code 04 of master station

Response of slave station: 01 84 01 82 C0

The slave station responds with illegal function code. 04 is not supported by EC20.

Note

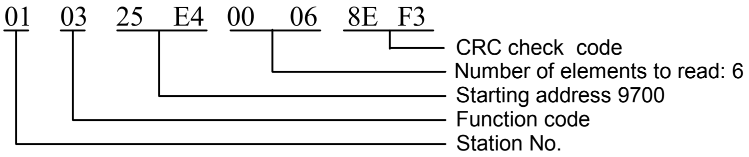
1. Element X does not support write operation (that is, the write of element X is invalid). For the writable properties of elements SM and SD, refer to Appendix 1Special auxiliary relay and Appendix 2Special data register (if the element is un-writable, the write operation is invalid).

2. The address of the slave station is 01, the last two bytes are CRC check code and the second byte is function code.

10.4.9 Handle double word

The current count value of C element is word or double word. The value from C200 to C255 are double words, which are read and written through the function code (03, 16) of the register. Every two registers correspond to a C double word. Only the pair can be read and written from/into register upon reading or writing.

For example, read the RTU frame of three C double word elements from C200 to C202:



In the returned data, 9700 and 9701 are two addresses for the content of C200. 9700 is the high 16 bits and 9701 is the low 16 bits.

When reading the double word, if the starting address read is not even number, then the system will respond with error code of illegal address; if the read number of registers is not an even number, the system will respond with error code of illegal data.

For example:

XMT from master station: 01 03 25 E5 00 04 5E F2

The starting address for the reading of master station : 4 word elements of 25 E5 (decimal 9701)

Response of slave station: 01 83 02 C0 F1

Response of slave station: illegal data address

XTM from slave station: 01 03 25 E4 00 05 CE F2

The starting address for master station read: 5 word elements of 25 E5

Response of slave station: 01 83 03 01 31

The data sent back from slave station is illegal.

10.4.10 Handle LONG INT

A LONG INT data can be saved in two D elements. For example, if a LONG INT data is saved in D3 and D4 of EC series PLC, the high 16 bits will be stored in D3 and the low 16 bits will be stored in D4. This is also true when the master station reads LONG INT data through Modbus and reorganize the data into 32 bits.

The storage principle for FLOAT is the same as the storage principle for LONG INT data.

10.4.11 Diagnostic function code

Diagnostic function code is used for test the communication between the master station and slave station, or the internal error of the slave station. The supported diagnostic subfunction codes are as follows:

Function code	Subfunction code	Name of subfunction code	Function code	Subfunction code	Name of subfunction code
08	00	Return query data	08	12	Return bus communication error count
08	01	Restart communication option	08	13	Return bus exceptional error count
08	04	Forced listen only mode	08	14	Return slave message count
08	10	Clear the counter	08	15	Return salve no response count
08	11	Return bus message count	08	18	Return bus character overrun count

Applicable to EC20/EC20H

10.4.12 Error code

For the XMT of master station, the slave station returns data or statistic value in the data field under the normal response status. But in the abnormal response status, the server will return error code in the data field. Refer to the following table for error codes:

Error code	Meaning of error code
0x01	Illegal function code
0x02	Illegal register address
0x03	Illegal data

In addition, if the slave station receives data under the following situations, no message will be returned:

1.Error in broadcast frame, e.g. data error, address error;

2.Characters overrun, e.g. RTU frame over 256 bytes;

3.Under RTU transfer mode, interval between two characters time out, which is the same as receiving error frame, and no message will be returned;

4.Listen-only mode of slave station;

5.The slave station received ASCII error frame, including frame tail error, character range error.

📖 **Note**

Read station is equipped with compulsory element. What is read is the value run by the program, which may be inconsistent with the compulsory value.

10.4.13 Modbus parameter setting

■ **Set communication port in system block**

There are two serial ports (serial port 0 and 1) on the communication port interface. Communication port 0 only supports Modbus slave station while communication port 1 supports both master and slave stations.

■ **Set Modbus communication parameters**

There is a button of default value on Modbus operand interface. The default value is the communication setting recommended by Modbus communication protocol. For the parameter setting items, refer to the table below.

Item	Setting
Station No.	0~31
Baud rate	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200
Data bit	Set to 7 or 8 bits; 7 for ASCII mode, 8 for RTU mode
Parity check bit	Set to no check, odd check and even check
Stop bit	Set to 1 or 2; set to 1 for odd or even check; set to 2 for no check status
Modbus master/slave	It can be set to master or slave station; communication port 1 can be set to master/slave station, communiation port 0 can only be set to slave station
Transfer mode	Select RTU mode or ASCII mode
Main mode timeout	The time for waiting the slave response by master is over the set value.
Note: After the operand is set and downloaded in the system block, it will be valid only after one operation.	

10.4.14 Modbus instruction

When PLC is used as Modbus master station, the Modbus data frame can be sent/received through Modbus instruction provided by system. For the detailed use of Modbus instruction, refer to **Error! Reference source not found.Error! Reference source not found.**.

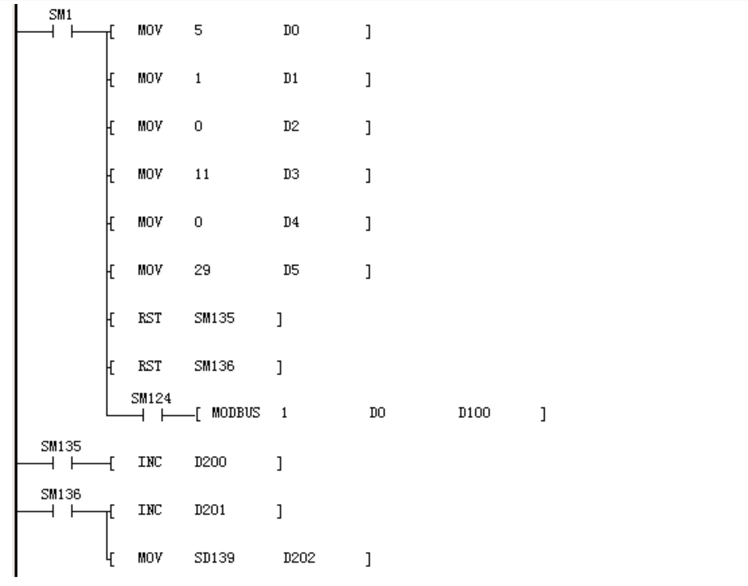
If PLC is set to master station, there is a timeout item in main mode when setting Modbus parameter in the system block. To ensure the correctness of the received data, the timeout period shall be longer than a scan cycle of Modbus slave station and with reasonable margin. For example, if EC20 is the slave station and a scan cycle of EC20 is 300ms, the main mode timeout of the master station shall be over 300ms. It is proper to set the timeout to 350ms.

■ **Application program**

Example 1: When EC20 PLC is Modbus master station as well as slave station, read bit status of No.5 station. The protocol address of slave station read by master station is the bit value ranging from 11 to 39. Assuming that the read data are as follows, the storage location for the received data starts from D100, save the address to D100, function code to D101 and number of registers in D102. Save the read bit value in the units beginning with D103.

42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27
X	X	X	0	0	1	1	0	0	0	0	1	0	1	1	0
D106								D105							
26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
1	1	0	0	0	1	1	0	1	1	1	0	0	0	0	1
D104								D103							

If the read number of the registers is not the times of 8, add 0 to the high bits. In the above example, it has added 0 to 3 high bits (40, 41 and 42) in D106.



```
SM1
-| |--[ MOV    5       D0      ]
     [ MOV    1       D1      ]
     [ MOV    0       D2      ]
     [ MOV    11      D3      ]
     [ MOV    0       D4      ]
     [ MOV    29      D5      ]
     [ RST    SM135   ]
     [ RST    SM136   ]
      SM124
     -| |--[ MODBUS  1      D0      D100    ]
SM135
-| |--[ INC    D200    ]
SM136
-| |--[ INC    D201    ]
     [ MOV    SD139   D202    ]
```

1. Designate 5 as the address of the slave station to be accessed (save to D0).
2. Designate 1 as the function code (save to D1).
3. The address of the register to be read is 11 (Save to D2/D3 according to high and low bytes).
4. The number of registers to be read is 29 (Save to D4/D5 according to high bits and low bits).
5. The received data is saved to D100.
6. If the receive is completed (set SM135), add 1 to D200.
7. If the communication fails (set SM136), add 1 to D201 and save the error code to D202.
8. SM124 is the idle flag of the communication port.

Note

1. When logic address is used for addressing the bit element of EC20 PLC, the logic address 1 is the protocol address 0. In the above example, reading the value of 11~39 bits (protocol address) in the slave station, the logic address shall start from 12.

2. The failure of this communication will not affect the next communication, that is, if there are two Modbus XMT instructions in one user program, the first communication fails and has error code, it will not influence the data sending of the second Modbus instruction. Thus, in the example, we placed the error code of SD139 in D202, which can be observed through D202.

3. For the message sending of the slave station, if the master station is in listen-only mode, there will be no data to be returned and the system will display error flag. Therefore, when using Modbus of EC20, if EC20 is the master station, the user shall clearly know which PLC slave station is under listen-only mode, so as to ensure that the failure of the communication is not caused by the listen-only mode of the slave station.

Example 2: EC20 is the Modbus master station, the slave station is also an EC20. Read the status of bit elements (protocol address: 2000~2017) in No.5 station.

The read data are as follows:

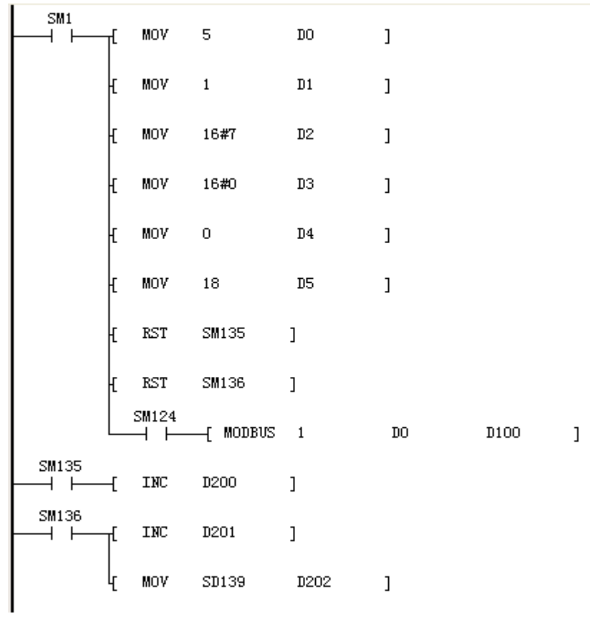
The received frame starts from D100.

D100 is for saving address

D101 is for saving function code

D102 is for saving the number of registers

Units beginning with D103 are for saving the read value of bit element



```
SM1
--| |--+-[ MOV   5      D0   ]
       |
       +-[ MOV   1      D1   ]
       |
       +-[ MOV   16#7   D2   ]
       |
       +-[ MOV   16#0   D3   ]
       |
       +-[ MOV   0      D4   ]
       |
       +-[ MOV   18     D5   ]
       |
       +-[ RST   SM135  ]
       |
       +-[ RST   SM136  ]
       |  SM124
       +--| |--[ MODBUS  1    D0    D100   ]
SM135
--| |----[ INC   D200   ]
SM136
--| |--+-[ INC   D201   ]
       |
       +-[ MOV   SD139  D202  ]
```

1. The program has designated 5 as the address of the slave station to be accessed (save to D0).
2. The program has designated 1 as function code (save to D1).
3. The starting address of the register to be read is 07D0 (hexadecimal, save to D2/D3 according to high bits and low bits).
4. The number of registers to be read is 18 (Save to D4/D5 according to high bits and low bits).
5. The received data is saved to D100.
6. If the receive is completed (set SM135), add 1 to D200.
7. If the communication fails (set SM136), add 1 to D201 and save the error code to D202.
8. SM124 is the idle flag of the communication port.

Example 3: EC20 is the Modbus master station as well as the slave station. Read the status of the bit element with the protocol address ranging from 40 to 43 of No.5 station.

The read data are as follows:

The received frame starts from D100.

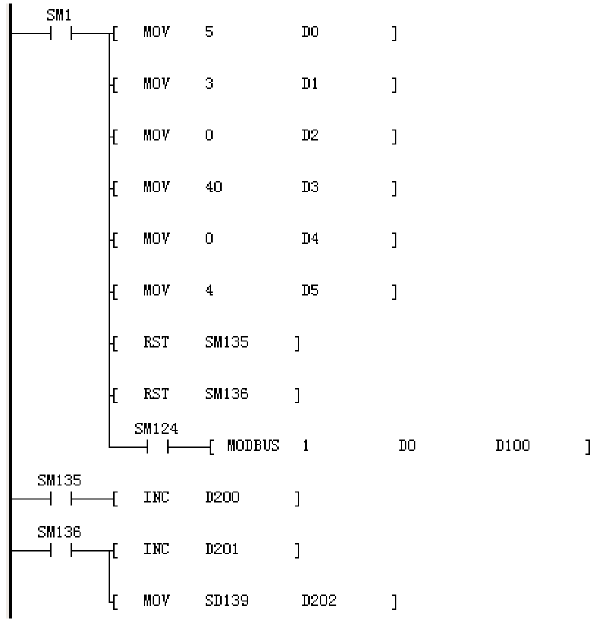
D100 is for saving address

D101 is for saving function code

D102 is for saving the number of registers

Units beginning with D103 are for saving the read value of bit element

40 element MSB	40 element LSB	41 element MSB	41 element LSB	42 element MSB	42 element LSB	43 element MSB	43 element LSB
D103	D104	D105	D106	D107	D108	D109	D110



```
SM1
--| |--+-[ MOV   5      D0   ]
       |
       +-[ MOV   3      D1   ]
       |
       +-[ MOV   0      D2   ]
       |
       +-[ MOV   40     D3   ]
       |
       +-[ MOV   0      D4   ]
       |
       +-[ MOV   4      D5   ]
       |
       +-[ RST   SM135  ]
       |
       +-[ RST   SM136  ]
       |  SM124
       +--| |--[ MODBUS  1    D0    D100   ]
SM135
--| |----[ INC   D200   ]
SM136
--| |--+-[ INC   D201   ]
       |
       +-[ MOV   SD139  D202  ]
```

1. The program has designated 5 as the address of the slave station to be accessed (save to D0).
2. The program has designated 3 as function code (save to D1).
3. The starting address of the register to be read is 40 (save to D2/D3 according to high bits and low bits).
4. The number of registers to be read is 4 (Save to D4/D5 according to high bits and low bits).
5. The received data is saved to D100.
6. If the receiving is completed (set SM135), add 1 to D200.
7. If the communication fails (set SM136), add 1 to D201 and save the error code to D202.
8. SM124 is the idle flag of the communication port.

10.5 N:N bus communication protocol

10.5.1 Introduction

N:N bus is a small PLC network developed by Control Technology Co., Ltd. The physical layer of N:N bus uses RS485, so the PLC can be directly connected through communication port 1 or connected through communication port 0 by RS232/RS485 converter. The connected PLC of N:N bus can automatically exchange the values between D elements and M elements , which makes the access to the other PLC elements on the network as convenient as accessing its own element. In N:N bus, the data access between PLCs is completely equivalent (N:N communication network).

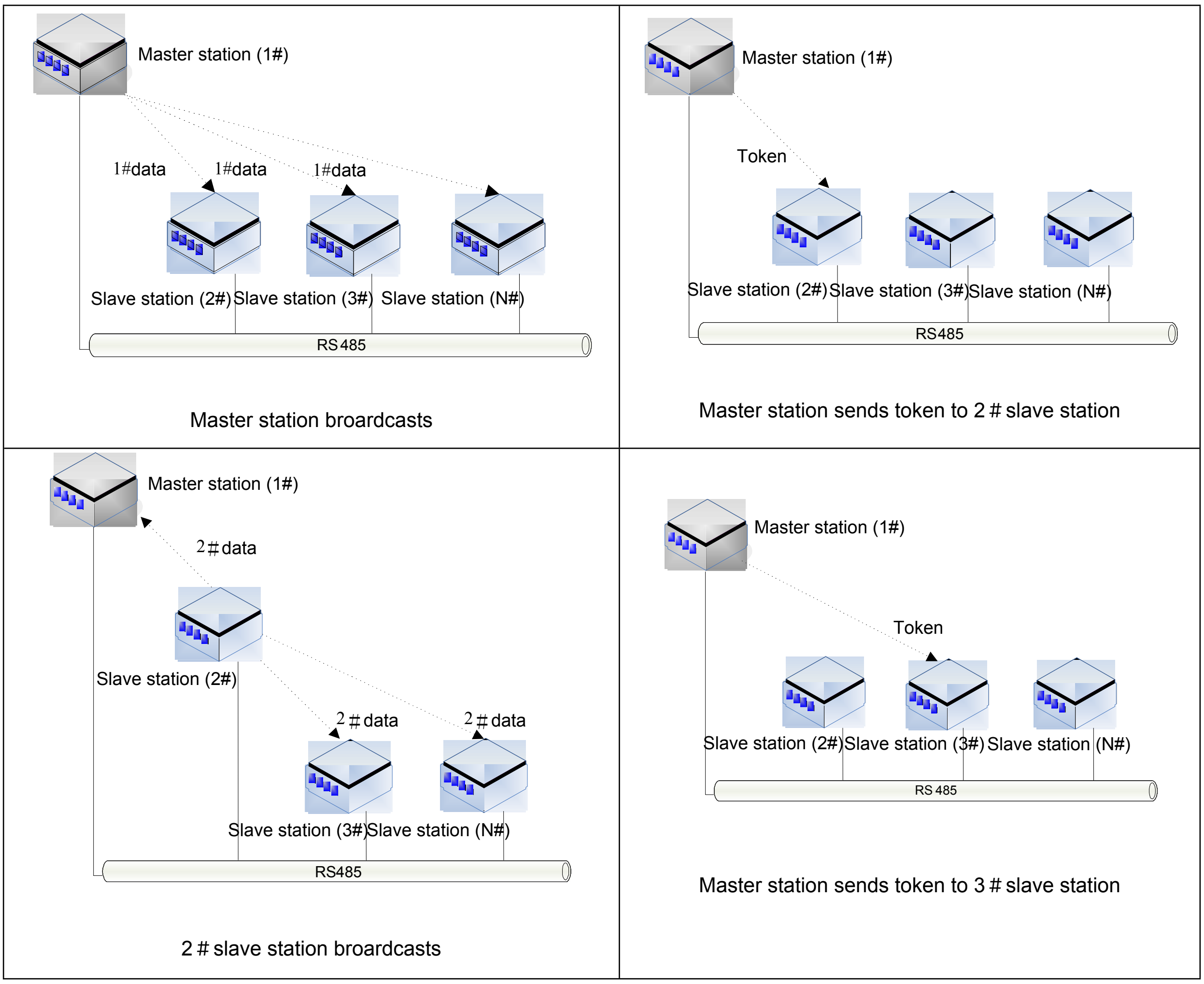
It is convenient to configure N:N bus. Most parameters of N:N bus only need to be configured on No.0 PLC. In addition, N:N bus supports online modification of the network parameters, and is able to detect the newly added PLC automatically. If any PLC is disconnected from the network, the other PLCs will continue to exchange the data. It is also able to monitor the communication status of the whole network through the relevant SM element of any PLC in N:N bus.

10.5.2 N:N bus data transfer mode

N:N bus has two types of messages: token sent by the master station; broadcast of PLCs on data.

The token is sent by the master station. At first, the master station holds the token. After data broadcast, it will send the token to each slave station in cycle and sequence. Only the slave station receiving the token can broadcast other PLCs (including master station).

Figure 10-1~10-5 show the main process of network communication. 1＃station is the master station. It is necessary to note that generally 0＃station is the master station by default and 1＃station is the standby master station (shift to the master station when communication fault or power failure to the master station occurs).

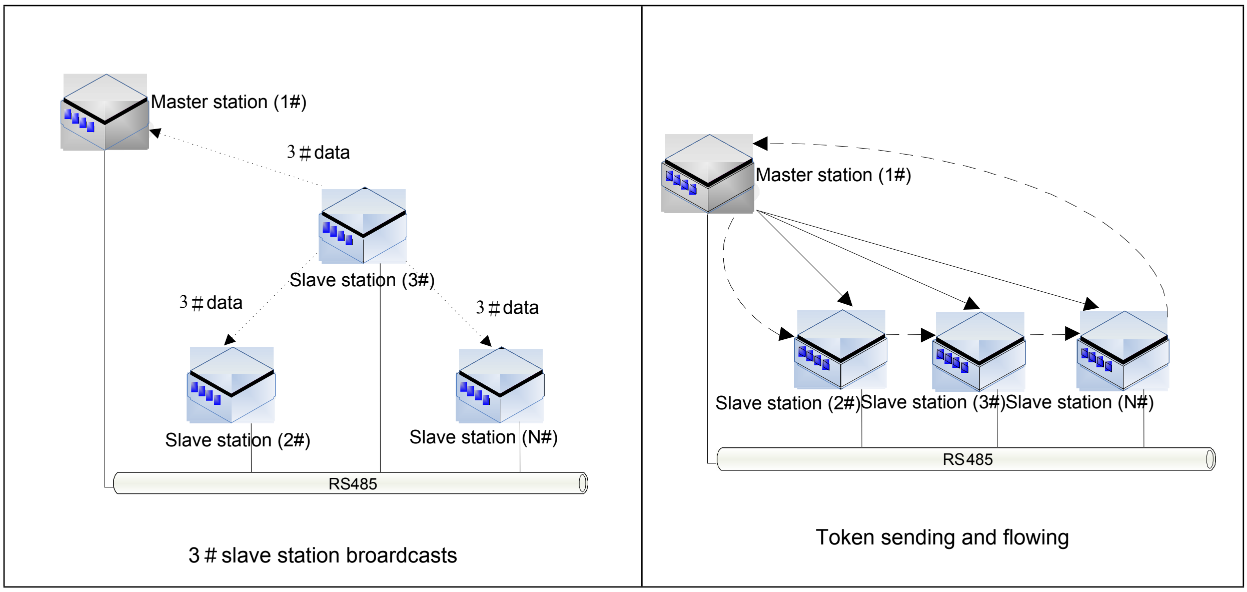


Master station broadcasts

Master station sends token to 2＃slave station

2＃slave station broadcasts

Master station sends token to 3＃slave station



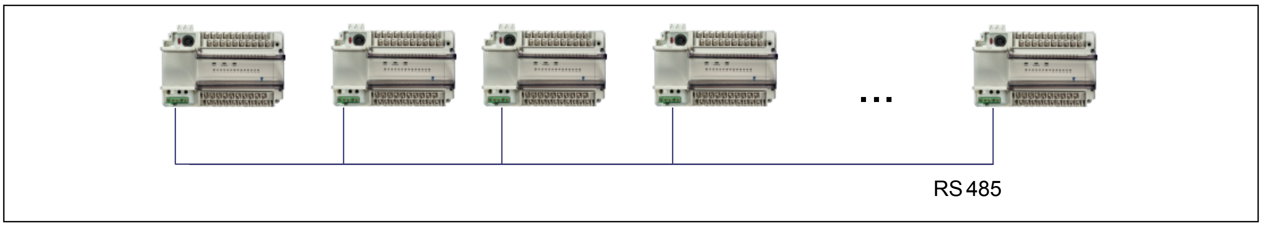
3＃slave station broadcasts

Token sending and flowing

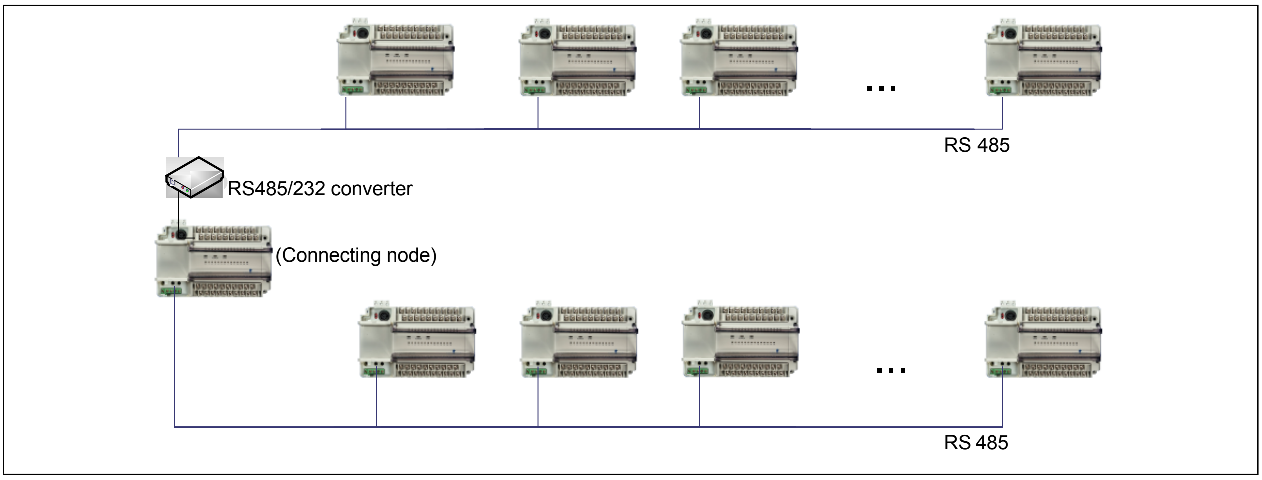
Figure 10-6 shows the sequence in which the token flows. The bold solid lines indicate the process of sending the token and the dotted lines indicate the sequence of holding the token and broadcasting. It is necessary to note that the token is not sent from one slave station (2#PLC) to another slave station (3#PLC) but from the master station to 2#PLC and then the master station to 3#PLC.

10.5.3 N:N bus network structure

N:N bus supports two kinds of network: single-layer network and multiple-layer network (as shown in the following figures):



N:N bus single-layer network



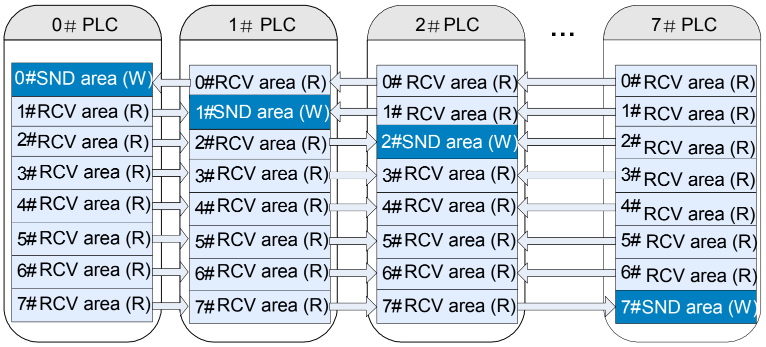
N:N bus multiple-layer network

In the single-layer network, each PLC only connects to N:N bus through 1 communication port. In the multiple-layer network, the layer-to-layer PLC (intermediate node) shall be connected, and the two communication ports of PLC shall be connected to different layers. The single-layer network can support up to 32 PLCs , while each layer of multiple-layer network can support 16 PLCs at most.

10.5.4 N:N bus refresh mode

The PLCs connected to N:N bus can automatically realize the exchange between parts of D elements and M elements in the network. The quantity and numbering of elements D and M are fixed, and the elements are called “Elements Sharing Area”. If

PLC uses N:N bus, the value of the Elements Sharing Area will keep refreshing automatically, so as to keep the value consistency of the Elements Sharing Area for each PLC in the network.



Note:
SND area: sending area
RCV area: receiving area
W: write
R: read

As shown in the above figure, each PLC with N:N bus connected has a writable sending area in the Elements Sharing Area. N:N bus will automatically send the information (values of designated elements D and M) of the writable sending area to other PLCs, receive the information from other PLCs and save it to the read-only sending area.

The element number in the Elements Sharing Area is fixed (64 D elements and 512 M elements can be shared) and these elements are distributed to more than one PLC. Therefore, the less PLCs are connected to the network, the more elements can be distributed to each PLC. The relationship is defined by N:N bus refresh mode:

■ **Distribution of D element on N:N bus single-layer network:**

Distribution of D element in sending area	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
D7700~D7701	#0	#0	#0	#0	#0
D7702~D7703	#1				
D7704~D7705	#2	#1			
D7706~D7707	#3				
D7708~D7709	#4	#2	#1		
D7710~D7711	#5				
D7712~D7713	#6	#3			
D7714~D7715	#7				
D7716~D7717	#8	#4	#2	#1	
D7718~D7719	#9				
D7720~D7721	#10	#5			
D7722~D7723	#11				
D7724~D7725	#12	#6	#3		
D7726~D7727	#13				
D7728~D7729	#14	#7			
D7730~D7731	#15				
D7732~D7733	#16	#8	#4	#2	#1
D7734~D7735	#17				
D7736~D7737	#18	#9			
D7738~D7739	#19				
D7740~D7741	#20	#10	#5		
D7742~D7743	#21				
D7744~D7745	#22	#11			
D7746~D7747	#23				
D7748~D7749	#24	#12	#6	#3	
D7750~D7751	#25				
D7752~D7753	#26	#13			
D7754~D7755	#27				
D7756~D7757	#28	#14	#7		
D7758~D7759	#29				
D7760~D7761	#30	#15			
D7762~D7763	#31				

Explanation:

1) In mode 1, the D elements distributed to the sending area by 0# station are D7700 and D7701. D7700 and D7701 can be written by the PLC of 0# station, and directly read by other stations (1#--31#).

2) In mode 2, the D elements distributed to the sending area by 0# station are D7700, D7701, D7701 and D7703. The elements can be written by the PLC of 0# station and directly read by other stations (1#--15#).

- **Distribution of M element on N:N bus single-layer network:**

Distribution of M element in sending area	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
M1400~M1415	#0	#0	#0	#0	#0
M1416~M1431	#1				
M1432~M1447	#2	#1			
M1448~M1463	#3				
M1464~M1479	#4	#2	#1		
M1480~M1495	#5				
M1496~M1511	#6	#3			
M1512~M1527	#7				
M1528~M1543	#8	#4	#2	#1	
M1544~M1559	#9				
M1560~M1575	#10	#5			
M1576~M1591	#11				
M1592~M1607	#12	#6	#3		#0
M1608~M1623	#13				
M1624~M1639	#14	#7			
M1640~M1655	#15				
M1656~M1671	#16	#8	#4	#2	#1
M1672~M1687	#17				
M1688~M1703	#18	#9			
M1704~M1719	#19				
M1720~M1735	#20	#10	#5		
M1736~M1751	#21				
M1752~M1767	#22	#11			
M1768~M1783	#23				
M1784~M1799	#24	#12	#6	#3	
M1800~M1815	#25				
M1816~M1831	#26	#13			
M1832~M1847	#27				
M1848~M1863	#28	#14	#7		
M1864~M1879	#29				
M1880~M1895	#30	#15			
M1896~M1911	#31				

Explanation:

1) In mode 1, the M elements distributed to the sending area by 0# station range from M1400 to M1415. The elements can be written by the PLC of 0# station and directly read by other stations (1#--31#).

2) In mode 2, the M elements distributed to the sending area by 0# station range from M1400 to M1431. The elements can be written by the PLC of 0# station and directly read by other stations (1#--31#).

- **Distribution of D element on N:N bus multiple-layer network (layer 0):**

Distribution of D element in sending area	Mode 6	Mode 7	Mode 8	Mode 9
D7700~D7701	#0	#0	#0	#0
D7702~D7703	#1			
D7704~D7705	#2	#1		
D7706~D7707	#3			
D7708~D7709	#4	#2	#1	
D7710~D7711	#5			
D7712~D7713	#6	#3		
D7714~D7715	#7			
D7716~D7717	#8	#4	#2	#1
D7718~D7719	#9			
D7720~D7721	#10	#5		
D7722~D7723	#11			
D7724~D7725	#12	#6	#3	

Distribution of D element in sending area	Mode 6	Mode 7	Mode 8	Mode 9
D7726~D7727	#13			
D7728~D7729	#14	#7		
D7730~D7731	#15			

Explanation:

In mode 6, D7700 and D7701 are distributed to the sending area by 0# station (layer 0). They can be written by the PLC of 0# station and directly read by the other stations (1#--15#).

- **Distribution of D element on N:N bus multiple-layer network (layer 1):**

Distribution of D element in sending area	Mode 10	Mode 11	Mode 12	Mode 13
D7732~D7733	#0	#0	#0	#0
D7734~D7735	#1			
D7736~D7737	#2	#1		
D7738~D7739	#3			
D7740~D7741	#4	#2	#1	
D7742~D7743	#5			
D7744~D7745	#6	#3		
D7746~D7747	#7			
D7748~D7749	#8	#4	#2	#1
D7750~D7751	#9			
D7752~D7753	#10	#5		
D7754~D7755	#11			
D7756~D7757	#12	#6	#3	
D7758~D7759	#13			
D7760~D7761	#14	#7		
D7762~D7763	#15			

Explanation:

In mode 10, D7732 and D7733 are distributed to the sending area by 0# station (layer 0). They can be written by the PLC of 0# station and directly read by the other stations (1#--15#).

- **Distribution of M element on N:N bus multiple-layer network (layer 0):**

Distribution of M element in sending area	Mode 6	Mode 7	Mode 8	Mode 9
M1400~M1415	#0	#0	#0	#0
M1416~M1431	#1			
M1432~M1447	#2	#1		
M1448~M1463	#3			
M1464~M1479	#4	#2	#1	
M1480~M1495	#5			
M1496~M1511	#6	#3		
M1512~M1527	#7			
M1528~M1543	#8	#4	#2	#1
M1544~M1559	#9			
M1560~M1575	#10	#5		
M1576~M1591	#11			
M1592~M1607	#12	#6	#3	
M1608~M1623	#13			
M1624~M1639	#14	#7		
M1640~M1655	#15			

Explanation:

In mode 6, the M elements distributed to the sending area by 0# station (layer 0) range from M1400 to M1415. The elements can be written by the PLC of 0# station and directly read by other stations (1#--15#).

- **Distribution of M element on N:N bus multiple-layer network (layer 1):**

Distribution of M element in sending area	Mode 10	Mode 11	Mode 12	Mode 13
M1656~M1671	#0	#0	#0	#0

Distribution of M element in sending area	Mode 10	Mode 11	Mode 12	Mode 13
M1672~M1687	#1			
M1688~M1703	#2	#1		
M1704~M1719	#3			
M1720~M1735	#4	#2	#1	
M1736~M1751	#5			
M1752~M1767	#6	#3		
M1768~M1783	#7			
M1784~M1799	#8	#4	#2	#1
M1800~M1815	#9			
M1816~M1831	#10	#5		
M1832~M1847	#11			
M1848~M1863	#12	#6	#3	
M1864~M1879	#13			
M1880~M1895	#14	#7		
M1896~M1911	#15			

Explanation:

In mode 10, the M elements distributed to the sending area by 0# station (layer 1) range from M1656 to M1671. The elements can be written by the PLC of 0# station and directly read by other stations (1#--15#).

🕮 **Note**

Once PLC is configured with N:N bus communication protocol, D7700~D7763 and M1400~M1911 will become the public resource for data exchange on the network. Please pay attention to these elements when using them in the program.

10.5.5 Enhanced refresh mode

To support share among more elements, EC series PLC provides mode 14~18. The modes are only applicable to the structure of single layer and the share among more elements. M element and D element enlarge on the original basis (M1400-M1911, D7500~D7755).

M element area (512):

Distribution of M element	Mode 14	Mode 15	Mode 16	Mode 17	Mode 18
M1400-M1415	#0	#0	#0	#0	#0
M1416-M1431	#1				
M1432-M1447	#2	#1			
M1448-M1463	#3				
M1464-M1479	#4	#2	#1		
M1480-M1495	#5				
M1496-M1511	#6	#3			
M1512-M1527	#7				
M1528-M1543	#8	#4	#2	#1	
M1544-M1559	#9				
M1560-M1575	#10	#5			
M1576-M1591	#11				
M1592-M1607	#12	#6	#3		
M1608-M1623	#13				
M1624-M1639	#14	#7			
M1640-M1655	#15				
M1656-M1671	#16	#8	#4	#2	#1
M1672-M1687	#17				
M1688-M1703	#18	#9			
M1704-M1719	#19				
M1720-M1735	#20	#10	#5		
M1736-M1751	#21				
M1752-M1767	#22	#11			
M1768-M1783	#23				
M1784-M1799	#24	#12	#6	#3	
M1800-M1815	#25				
M1816-M1831	#26	#13			

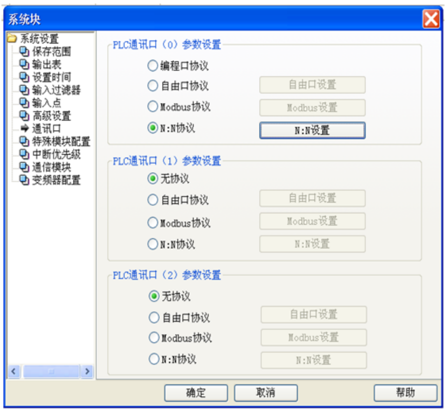
M1832-M1847	#27				
M1848-M1863	#28	#14	#7		
M1864-M1879	#29				
M1880-M1895	#30	#15			
M1896-M1911	#31				

D element area (256):

Distribution of D element	Mode 14	Mode 15	Mode 16	Mode 17	Mode 18
D7500~D7507	#0	#0	#0	#0	#0
D7508~D7515	#1				
D7516~D7523	#2	#1			
D7524~D7531	#3				
D7532~D7539	#4	#2	#1		
D7540~D7547	#5				
D7548~D7555	#6	#3			
D7556~D7563	#7				
D7564~D7571	#8	#4	#2	#1	
D7572~D7579	#9				
D7580~D7587	#10	#5			
D7588~D7595	#11				
D7596~D7603	#12	#6	#3		
D7604~D7611	#13				
D7612~D7619	#14	#7			
D7620~D7627	#15				
D7628~D7635	#16	#8	#4	#2	#1
D7636~D7643	#17				
D7644~D7651	#18	#9			
D7652~D7659	#19				
D7660~D7667	#20	#10	#5		
D7668~D7675	#21				
D7676~D7683	#22	#11			
D7684~D7691	#23				
D7692~D7699	#24	#12	#6	#3	
D7700~D7707	#25				
D7708~D7715	#26	#13			
D7716~D7723	#27				
D7724~D7731	#28	#14	#7		
D7732~D7739	#29				
D7740~D7747	#30	#15			
D7748~D7755	#31				

10.5.6 N:N bus parameter setting

Select **Communication Port** in the **System block** dialogue box, and select **N:N bus protocol** in the port 0 or port 1 setting area to enable the **N:N bus setting** button as follows:



PLC通讯口（0）参数设置

○ 编程口协议
○ 自由口协议
○ Modbus协议
● N:N协议

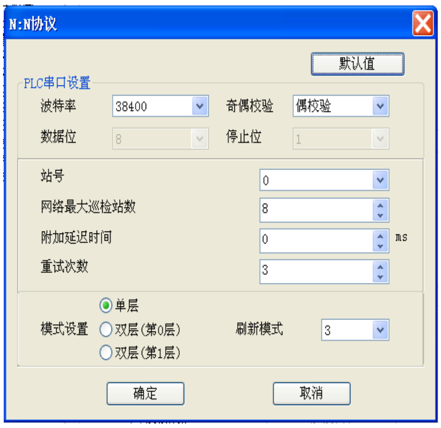
PLC通讯口（1）参数设置

● 无协议
○ 自由口协议
○ Modbus协议
○ N:N协议

PLC通讯口（2）参数设置

● 无协议
○ 自由口协议
○ Modbus协议
○ N:N协议

Click the **N:N bus setting** button to enter the **N:N bus protocol** setting dialogue box as shown below:



N:N协议

默认值

PLC串口设置

波特率 38400

奇偶校验 偶校验

数据位 8

停止位 1

站号 0

网络最大巡检站数 8

附加延迟时间 0 ms

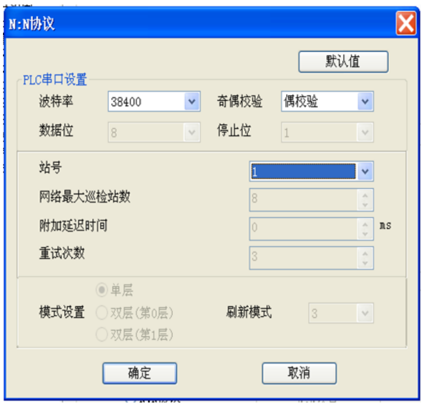
重试次数 3

模式设置 ● 单层 ○ 双层(第0层) ○ 双层(第1层)

刷新模式 3

确定 取消

As shown in the preceding figure, the N:N bus parameters are set through the system block. The **Station No.** shall begin with 0#. Several PLCs cannot share the same station number. 0# station is used for starting and setting the whole network. The setting of **Max number of sites**, **Additional delay time**, **Retry times** and **Mode** can be realized through 0# station. For the stations with other station numbers, except that the **Baud rate** and **Parity check** shall be consistent with those of 0# station, they only need to set their own **Station No.**, as shown in the following figure:



N:N协议

默认值

PLC串口设置

波特率 38400

奇偶校验 偶校验

数据位 8

停止位 1

站号 1

网络最大巡检站数 8

附加延迟时间 0 ms

重试次数 3

模式设置 ● 单层 ○ 双层(第0层) ○ 双层(第1层)

刷新模式 3

确定 取消

The **Max number of sites** refers to the total number of PLCs used in the network. If 6 PLCs are used in total, the value shall be set to 6 and the station number of the 6 PLCs ranges from 0 to 5. If you want to add another two PLCs to the network later without any interruption of the network, you can set the **Max number of sites** to 8. The numbers of the newly added PLCs are 6# and 7#. When 6# and 7# are connected to the network, they will be automatically detected by N:N bus within one second and included into the data exchange with 0#-5#.

10.6 Control strategies

10.6.1 Master station confirmation

No. 0 station is the master station by default and only the station can initialize and start the whole network. The relevant settings of N:N, such as refresh mode, additional delay time and retry times, must be configured by No. 0 station. When the station modifies the relevant configuration online and downloads system block, the standby master station will control the network. After finishing system block downloading, No. 0 station will become the master station instead of the standby one.

The strategy of master station in network: The station with the minimum No. will be the master station.

10.6.2 Max. number of inspection stations

It is recommended to set the Max. number of inspection stations to the total number of PLCs in the actual network and compile the station No. from 0 in sequence. When the Max. number of inspection stations is N, the network will only control No. 0~N-1 stations. Specially, if the Max. number of inspection stations is wrong, that is, the number is smaller than the number of PLCs in 485 network, the station No. larger than or equal to the Max. number of inspection stations cannot broadcast data but can receive broadcast data from the station No. smaller than the Max. number of inspection stations.

10.6.3 Multiple master-slave (M:N)

N:N can be applied to network multiple master-slave structure. The meanings of master and slave: The master means the PLC can write its own M and D elements and read M and D elements of other stations; the slave means the PLC can only read M and D elements of other stations. In the Max. number of inspection stations (also limited by refresh mode), the PLC with smaller station No. can be master while larger station No. can be slave. The slave stations can only read the relevant M and D elements of master station which correspond to each master station according to refresh mode. You can refer to N:N share M and D element list.

10.6.4 Examples of N:N

There are 5 PLCs in total, the refresh mode is 3 and station No. is 0#~4#. Store the sum of D100 of 0#PLC and D305 of 2#PLC into D500 of 4#PLC.

0# programming: MOV D100 D7700

2# programming: MOV D305 D7716

4# programming: ADD D7700 D7716 D500

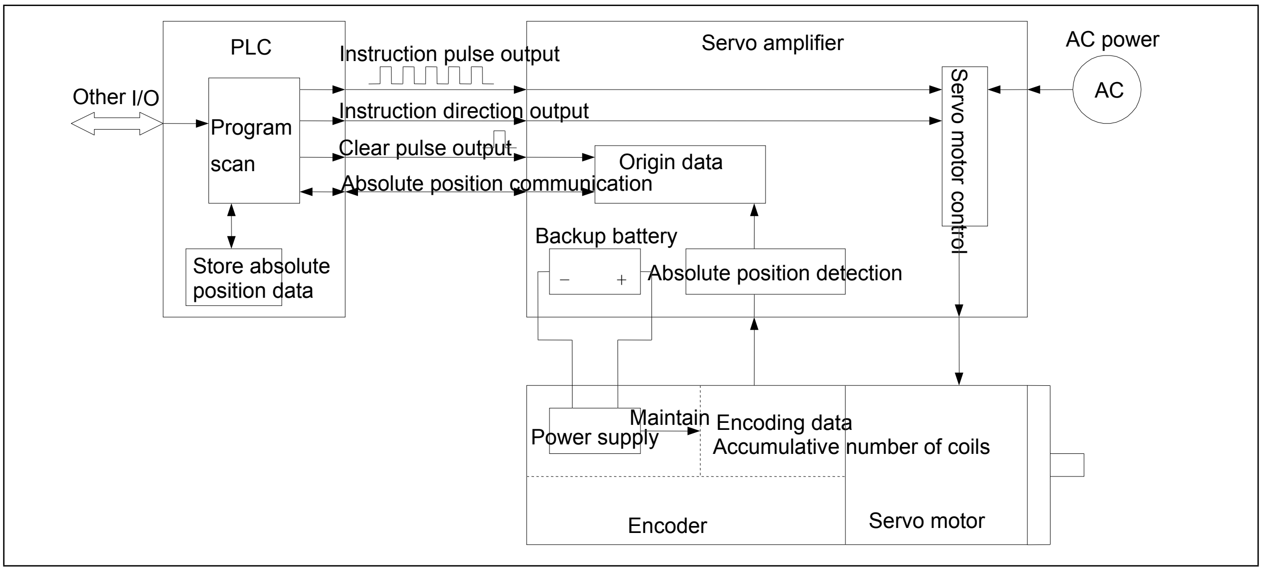
Instruction: The example is N:N single layer network of 5 PLC stations and 3 refresh mode: each station can distribute 8 D elements and 64 M elements. The distributed D elements of 0#station are D7700~D7707, 2#station are D7716~D7723 and 4#station are D7732~D7739. Store D100 of 0#station to the network to distribute in write area D7700, D305 of 2#station to the network to distribute in write area D7716 and the sum of D7700 and D7716 of 4#station to the local element D500.

Chapter 11 Using positioning function

Positioning control system

Absolute position system

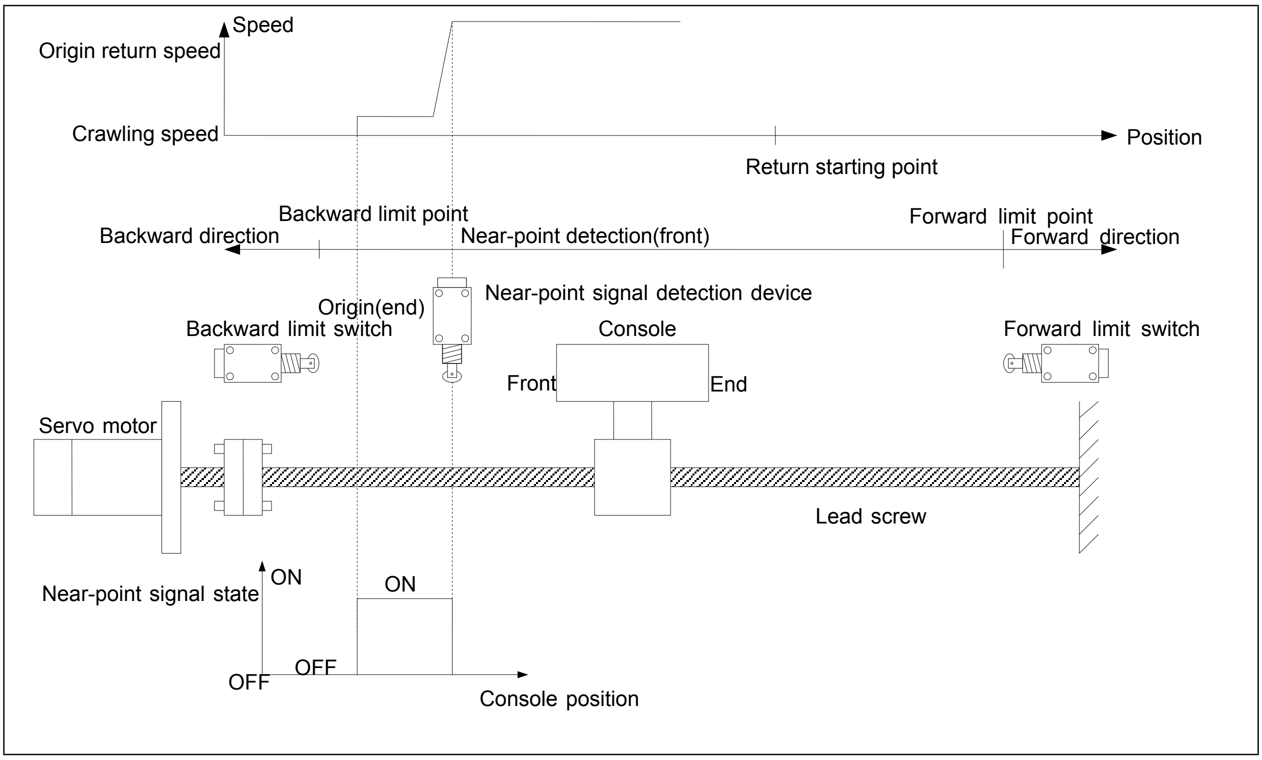
By detecting the current encoding position and the total number of running coils of servo motor, the absolute position system achieves the absolute position data of servo motor on travel. According to the principle, an absolute coordinate system can be established on mechanical travel. The following is the block diagram of the absolute position system.



In above figure, unlike the common incremental encoder, the current encoding position and the total number of running coils for the absolute position system can be maintained by power supply of a backup battery. Even if the power is off, the servo drive can power on again and achieve the current absolute position data.

After PLC powers on, by communication or other special methods, the absolute position data will be obtained to confirm travel coordinate position. PLC can control servo drive and motor through locating instruction, realize accurate positioning on travel, and conduct automatic increasing/decreasing and refresh on absolute position data. Therefore, a work system can be structured on basis of the absolute position coordinate.

The following is a mechanical simple diagram of absolute position system based on EC series PLC locating instruction.

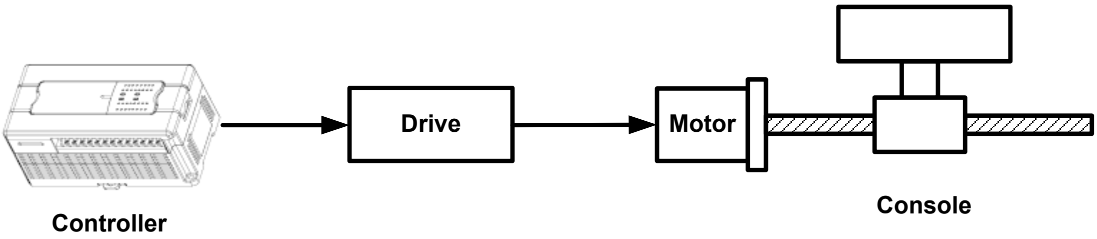


In the system, the servo motor drives lead screw to control the operation of console. The console's position in travel is detected by absolute encoder. The servo motor will decelerate to crawling speed when the near-point signal device detects the front of console (set) to be origin return signal; PLC will stop high-speed pulse output when the near-point signal device detects the end of console (reset) to be origin arrival signal. The forward limit switch and backward limit switch must be set. When using ZRN instruction, without automatic near-point signal searching, you must operate at the distance farther than the near-point signal device. By design and programming, you can adopt jogging operation and adjust the position manually.

11.1.2 Positioning control system

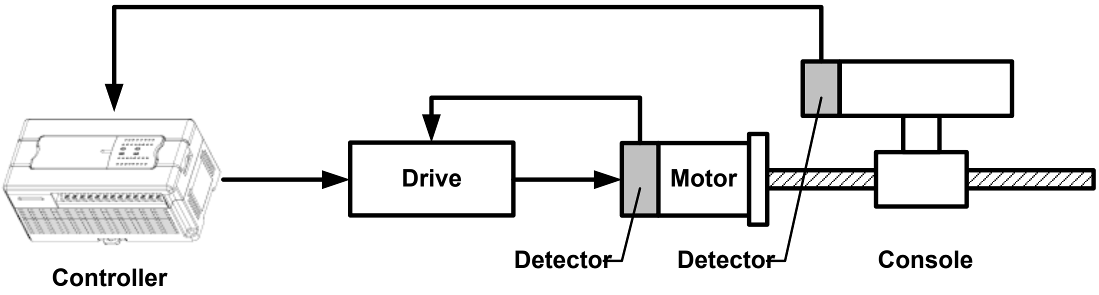
According to different control methods, the positioning control system can be divided into open loop control system, half close loop control system and close loop control system.

The open loop control system does not accept feedback control and only controls output, also called no feedback control system. The system consists of controller, stepping drive and stepping motor. The controller sends pulse instructions to the stepping drive and thus the stepping motor drives the console to move a certain distance. The system is simple, stable and easy to use, but it cannot detect or correct errors, has poor control accuracy and anti-interference performance, and is sensitive to the changes of system parameters. Therefore, it is only applicable to the cases regardless of outside influence or requiring small inertia or low accuracy.



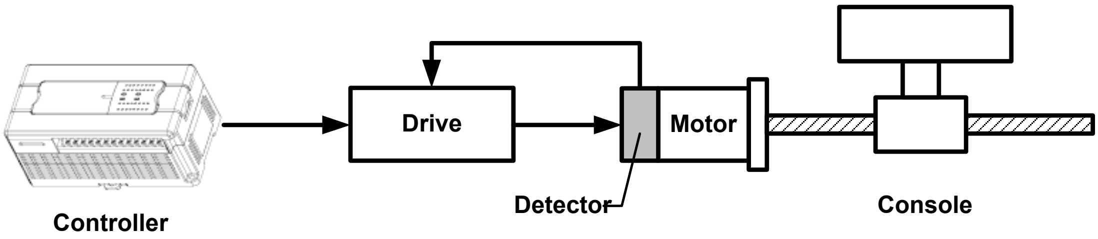
Open loop control system

The close loop control system is the automatic control system made up of signal positive channel and feedback channel, also called feedback control system. The system consists of controller, servo drive, servo motor and detector. It performs automatic detection on the actual displacement of the console and feedbacks to the controller for close loop control. The system has high positioning accuracy, but it is complicated, hard to debug and maintain and expensive. Therefore, it is mainly applicable to the cases requiring high accuracy and large numerical control machine tools.



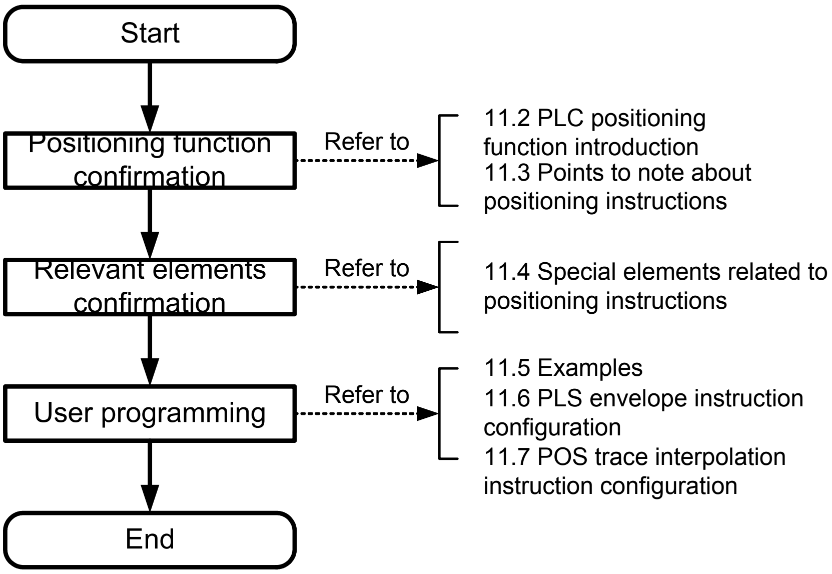
Close loop control system

The working principle of half close control system is similar to the close control system. However, its detector is not installed on the console but on the axis of the servo motor. The system is more excellent than open loop control system in accuracy, speed and dynamic property, less complicated and expensive than close loop control system, and it is mainly applicable to the cases requiring medium accuracy and medium or small numerical control machine tools.



Half close loop control system

11.1.3 Positioning control procedures


Start
Positioning function confirmation
Refer to
11.2 PLC positioning function introduction
11.3 Points to note about positioning instructions
Relevant elements confirmation
Refer to
11.4 Special elements related to positioning instructions
User programming
Refer to
11.5 Examples
11.6 PLS envelope instruction configuration
11.7 POS trace interpolation instruction configuration
End

11.2 EC series PLC positioning function introduction

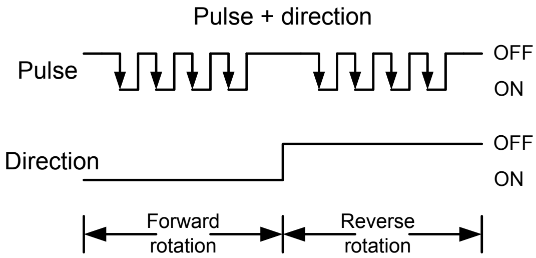
The positioning functions EC series PLC support includes pulse output positioning, linear and circular trace interpolation of two axes and synchronous motion control intershafts. The functions can be used widely to control stepping and servo drives of various brands. The absolute position data can be obtained by the way the relevant servo drive provides.

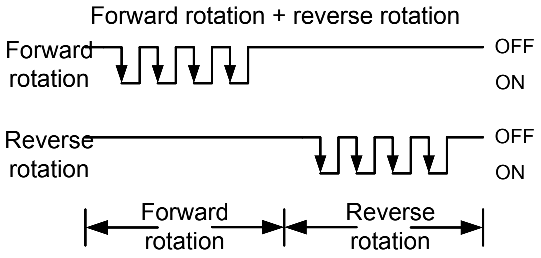
Positioning functions of EC series PLC main module

Name	EC20H	EC20	EC10V	EC10	EC10A
Control axis	6 axes/4 axes	2 axes	4 axes	2 axes	2 axes
Max. output frequency	200kHz	100kHz	100kHz	100kHz	50kHz
Pulse output type	Open collector	Open collector	Open collector	Open collector	Open collector

Name	EC20H	EC20	EC10V	EC10	EC10A
Pulse output mode	Pulse+direction, forward rotation+reverse rotation	Pulse+direction	Pulse+direction	Pulse+direction	Pulse+direction
ACC/DEC processing	Ladder ACC/DEC, triangle prevention	Ladder ACC/DEC	Ladder ACC/DEC	Ladder ACC/DEC	Ladder ACC/DEC
Interpolation function	2-axis linear interpolation, circular arc interpolation	—	—	—	—
Synchronizing function	Position synchronization, electronic gear	—	—	—	—
Absolute postion detection	Read ABS instruction		—	Read ABS instruction	—
Positioning range	-2, 147, 483, 648~+2, 147, 483, 647 (pulse)				

When connecting to the servo, set the input signal of servo amplifier to negative logic mode and define the pulse output mode as follows:

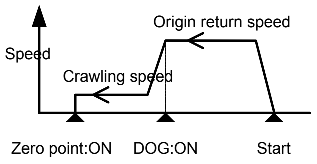
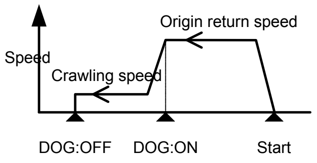
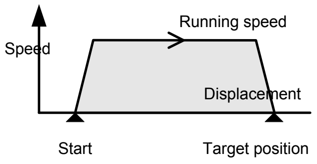
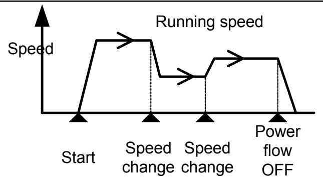


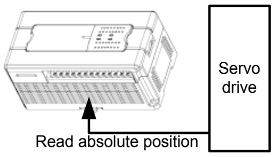
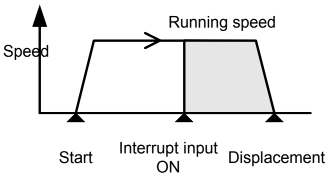
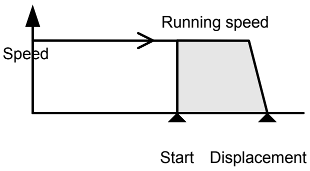
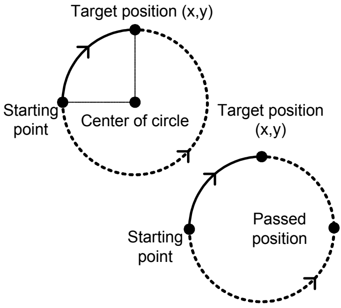
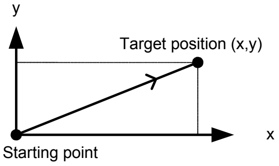
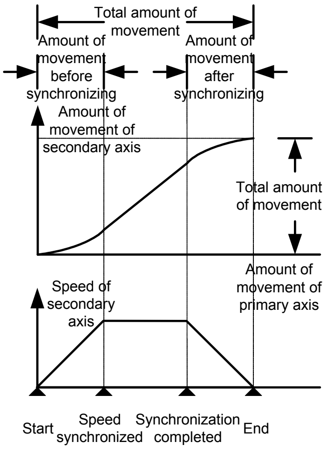
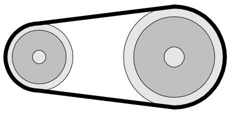


Note: The high-speed IO instructions can also be pulse output, only pulse signal output control rather than direction signal control. When using the instructions, conduct positive and accumulative processing on corresponding SD elements; when the servo drive is running forward, set the servo direction signal to ON, whereas set the servo direction signal to OFF.

The positioning functions EC series PLC support are shown as follows:

Positioning functions of EC series PLC main module

Name	Action	Content	EC20H	EC20	EC10	EC10V
DSZR	 Origin return speed Speed Crawling speed Zero point:ON DOG:ON Start	Act according to the set origin return speed and search DOG signal automatically. After detecting DOG signal (DOG sensor is ON), decelerate to crawling speed. Stop at zero point input and finish origin return.	●	●		●
ZRN	 Origin return speed Speed Crawling speed DOG:OFF DOG:ON Start	Act according to the set origin return speed. After detecting DOG signal (DOG sensor is ON), decelerate to crawling speed. Stop when DOG sensor is OFF and finish origin return.	●	●	●	●
DRVI	 Running speed Speed Displacement Start Target position	Act according to the set running speed and stop at the target position. The position adopts relative coordinate.	●	●	●	●
DRVA		Act according to the set running speed and stop at the target position. The position adopts absolute coordinate.	●	●	●	●
PLSV	 Running speed Speed Start Speed change Speed change Power flow OFF	Act according to the set running speed. If the running speed changes, run at new speed; if the power flow is invalid, pulse output will stop. During action of ACC/DEC and speed change, execute ACC/DEC.	●	●	●	●

Name	Action	Content	EC20H	EC20	EC10	EC10V
ABS		Read the current absolute position data from servo drive.	●	●	●	●
DVIT	 Running speed Speed Start Interrupt input ON Displacement	Act according to the set running speed. If the interrupt input is ON, decelerate to stop after running the set pulses.		●		●
STOPDV		When operating a certain positioning, if the instruction starts, decelerate to stop after running the set pulses.	●			
CW	 Target position (x,y) Starting point Center of circle Target position (x,y) Passed position Starting point	Move to the target position in clockwise circular trace at designated linear speed.	●			
CCW		Move to the target position in counterclockwise circular trace at designated linear speed.	●			
LIN		Move to the target position in linear trace at designated vector speed.	●			
MOVELINK	 Total amount of movement Amount of movement before synchronizing Amount of movement after synchronizing Amount of movement of secondary axis Total amount of movement Amount of movement of primary axis Speed of secondary axis Start Speed synchronized Synchronization completed End	The secondary axis follows the movement of the primary axis and keeps synchronous speed with the primary axis in a designated range. Support ACC/DEC control in the transient process before and after synchronizing.	●			
GEARBOX		Control the secondary axis to follow the movement of the primary axis according to a certain electronic gear ratio.	●			

The locating instructions and high-speed instructions output controllable pulses according to the set high-speed ports, regardless of user program scan cycle. For using methods of the instructions, please refer to *6.10High-speed I/O instruction.*

In the program, executing different locating instructions or high-speed instructions according to different output ports can output independent high-speed pulses.

11.3 Points to note about locating instructions

When the locating instructions or high-speed instructions are valid (including output completion), other operations on the same port will be invalid. Only when the high-speed pulse output instructions are invalid can other instructions output correctly.

If there are multiple locating instructions or high-speed instructions at the same port, the instruction first valid will occupy the output port and the instruction last valid will not occupy the port.

- **Transistor output**

Use EC series PLC with transistor output.

- **Requirements of locating instructions in programming**

The locating instructions can be used repeatedly in programs, but it is necessary to note that:

1. Do not execute other locating instructions or high-speed pulse instructions at the same high-speed pulse output point. At any time, only one locating instruction (high-speed pulse instruction) drives one high-speed pulse output point.

2. After the power flow of one locating instruction is OFF, it can be connected and driven again by 1 or above PLC scan cycle.

- **Points to apply locating instructions and high-speed instructions simultaneously**

From the point of functions, the locating instructions are recommended to replace the high-speed pulse instructions (PLSY, PLSR, PLS) and they can finish automatic updating of absolute position SD elements.

Absolute position SD elements can be used to store and update the current absolute position after executing the locating instructions. Automatic increasing/decreasing of SD elements depends on the accumulated SD changes when outputting pulses and the running direction when calling locating instructions. Do not write SD elements when executing the locating instructions; otherwise, a mess of data may occur.

If the locating instructions and high-speed pulse instructions (PLSY, PLSR, PLS) need to be used at the same time, it is necessary to program PLC to update the data in SD elements of absolute position registers correctly.

- **Limits to the actual output frequency of locating instructions**

When executing the locating instructions, the minimum frequency of the actual output pulse is limited by:

$$F_{\min\_acc} = \sqrt{\frac{F_{\max} \times 500}{T}}$$

In above formula, $F_{\max}$ is the Max. speed set by SD85 and SD86, T is ACC/DEC time set by SD87 (unit: ms), and $F_{\min\_acc}$ is the limit value of the Min. output frequency.

If the locating instruction sets the output frequency to F, there will be 3 situations for the actual output frequency.

- F< base frequency or F> $F_{\max}$, no actual output
- F< $F_{\min\_acc}$, the actual output is $F_{\min\_acc}$
- $F_{\min\_acc}$ ≤ F ≤ $F_{\max}$, the actual output is F

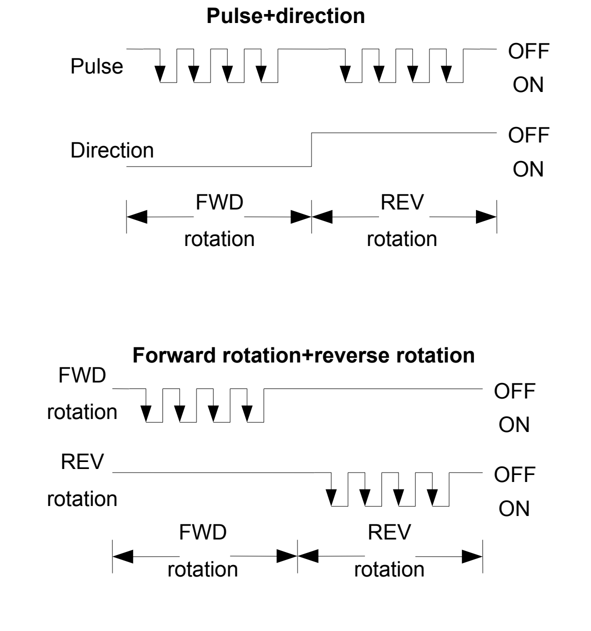
11.4 Special elements related to locating instructions

Elements related to locating instructions of EC20H series

Definitions and distribution of output axes of EC20H series are shown in the following tables.

- **Definitions of output axes of EC20H-1616MAT6**

Output axis	Available mode	Definition of output point		Definition of output mode
0	Pulse+direction	Pulse	Y0	
		Direction	Y1	
	Forward rotation +reverse rotation	FWD rotation	Y0	
		REV rotation	Y1	
1	Pulse+direction	Pulse	Y2	
		Direction	Y3	

Output axis	Available mode	Definition of output point		Definition of output mode
	Forward rotation +reverse rotation	FWD rotation	Y2	 Pulse+direction Pulse OFF ON Direction OFF ON FWD rotation REV rotation Forward rotation+reverse rotation FWD rotation OFF ON REV rotation OFF ON FWD rotation REV rotation
		REV rotation	Y3	
2	Pulse+direction	Pulse	Y4	
		Direction	Unlimited except Y4	
3	Pulse+direction	Pulse	Y5	
		Direction	Unlimited except Y5	
4	Pulse+direction	Pulse	Y6	
		Direction	Unlimited except Y6	
5	Pulse+direction	Pulse	Y7	
		Direction	Unlimited except Y7	

- **Definitions of output axes of EC20H-1616MAT4**

Output axis	Available mode	Definition of output point		Definition of output mode
0	Pulse+direction	Pulse	Y0	Refer to table 11-3 for definitions of output modes
		Direction	Y1	
	Forward rotation +reverse rotation	FWD rotation	Y0	
		REV rotation	Y1	
1	Pulse+direction	Pulse	Y2	
		Direction	Y3	
	Forward rotation +reverse rotation	FWD rotation	Y2	
		REV rotation	Y3	
2	Pulse+direction	Pulse	Y4	
		Direction	Unlimited except Y4	
3	Pulse+direction	Pulse	Y5	
		Direction	Unlimited except Y5	

🕮 **Note**

When using any output axis to connect servo, it is necessary to consider matching the output points. All output axes can adopt “pulse+direction” mode; the output axis 0 and 1 can adopt “forward rotation+reverse rotation” mode. In the mode of “pulse+direction”, the corresponding output points of the pulse and direction of the output axis 0 and 1 are fixed. You can choose direction signals for the output axis 2-5, but note that the output points cannot be used in other ways at the same time. For example, the pulse or direction signals of the output axis 2-5 and other axes cannot be defined on the same output point. In the mode of “forward rotation+reverse rotation”, the corresponding output points of FWD rotation and REV rotation signals are fixed. As for the output axis 0 and 1, whichever mode they are set, as long as the axis uses the locating instruction or high-speed I/O instruction, its corresponding two output points cannot be used in other ways.

- **Control and monitor output channels of output axis 0 (corresponding to Y0)**

Address	Name	Function	R/W
SM80	Pulse output stop control	Disable Y0 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM82	Pulse output monitor	Used to monitor the state of Y0 high-speed pulse output channel, ON when busy, OFF when ready	R
SM63	Pulse output completion	Enable Y0 high-speed pulse output completion interrupt when setting; disable	R/W

Address	Name	Function	R/W
	interrupt enabling control	pulse output completion interrupt when resetting	
SM280	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y0: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM281	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y0: using corresponding Y element Y(N) of N in SD206 means clear signal when setting; set Y10 to clear signal according to the default value when resetting	R/W
SM282	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y0: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM283	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y0: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM284	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y0: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM285	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y0: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM286	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y0: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM288	Locating instruction in drive	Used to monitor the state of Y0 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W
Note: When SM280 is set, the corresponding default clear signal Y10 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

- **Special data registers of output axis 0 (corresponding to Y0)**

Address	Name	Function	R/W
SD50	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD200~SD201 according to SD50~SD51 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD200.	R/W
SD51	Accumulated pulse total number (LSB)		R/W
SD200	Current absolute position (MSB)		R/W
SD201	Current absolute position (LSB)		R/W
SD202	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~200000, unit: pulse	R/W
SD203	Max. speed (LSB)		R/W
SD204	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD205	ACC/DEC time	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD206	Clear signal element designation	When SM201 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD207	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD208	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD209	Origin return speed (LSB)		R/W
SD56	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R
Note: 1. SD202~SD205 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.			

- **Control and monitor output channels of output axis 1 (corresponding to Y2)**

Address	Name	Function	R/W
SM262	Pulse output stop control	Disable Y2 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM272	Pulse output monitor	Used to monitor the state of Y2 high-speed pulse output channel, ON when busy, OFF when ready	R

Address	Name	Function	R/W
SM72	Pulse output completion interrupt enabling control	Enable Y2 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM320	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y2: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM321	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y2: using corresponding Y element Y(N) of N in SD326 means clear signal when setting; set Y12 to clear signal according to the default value when resetting	R/W
SM322	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y2: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM323	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y2: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM324	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y2: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM325	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y2: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM326	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y2: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM328	Locating instruction in drive	Used to monitor the state of Y2 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W

Note: When SM320 is set, the corresponding default clear signal Y12 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.

- **Special data registers of output axis 1 (corresponding to Y2)**

Address	Name	Function	R/W
SD160	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD320~SD321 according to SD160~SD161 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD320.	R/W
SD161	Accumulated pulse total number (LSB)		R/W
SD320	Current absolute position (MSB)		R/W
SD321	Current absolute position (LSB)		R/W
SD322	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~200000, unit: pulse	R/W
SD323	Max. speed (LSB)		R/W
SD324	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD325	ACC/DEC time	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD326	Clear signal element designation	When SM321 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD327	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD328	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD329	Origin return speed (LSB)		R/W
SD252	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R

Note:
1. SD322~SD325 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution.
2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.

- **Control and monitor output channels of output axis 2 (corresponding to Y4)**

Address	Name	Function	R/W
SM264	Pulse output stop control	Disable Y4 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM274	Pulse output monitor	Used to monitor the state of Y4 high-speed pulse output channel, ON when busy, OFF when ready	R

Address	Name	Function	R/W
SM74	Pulse output completion interrupt enabling control	Enable Y4 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM340	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y4: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM341	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y4: using corresponding Y element Y(N) of N in SD346 means clear signal when setting; set Y14 to clear signal according to the default value when resetting	R/W
SM342	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y4: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM343	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y4: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM344	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y4: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM345	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y4: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM346	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y4: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM348	Locating instruction in drive	Used to monitor the state of Y4 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W

Note: When SM340 is set, the corresponding default clear signal Y14 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.

■ Special data registers of output axis 2 (corresponding to Y4)

Address	Name	Function	R/W
SD164	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD340~SD341 according to SD164~SD165 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD340.	R/W
SD165	Accumulated pulse total number (LSB)		R/W
SD340	Current absolute position (MSB)		R/W
SD341	Current absolute position (LSB)		R/W
SD342	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~200000, unit: pulse	R/W
SD343	Max. speed (LSB)		R/W
SD344	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD345	ACC/DEC time	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD346	Clear signal element designation	When SM341 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD347	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD348	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD349	Origin return speed (LSB)		R/W
SD254	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R

Note:

1. SD342~SD345 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution.
2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.

■ Control and monitor output channels of output axis 3 (corresponding to Y5)

Address	Name	Function	R/W
SM265	Pulse output stop control	Disable Y5 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM275	Pulse output monitor	Used to monitor the state of Y5 high-speed pulse output channel, ON when busy, OFF when ready	R

Address	Name	Function	R/W
SM75	Pulse output completion interrupt enabling control	Enable Y5 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM350	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y5: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM351	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y5: using corresponding Y element Y(N) of N in SD356 means clear signal when setting; set Y15 to clear signal according to the default value when resetting	R/W
SM352	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y5: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM353	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y5: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM354	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y5: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM355	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y5: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM356	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y5: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM358	Locating instruction in drive	Used to monitor the state of Y5 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W
Note: When SM350 is set, the corresponding default clear signal Y15 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

- **Special data registers of output axis 3 (corresponding to Y5)**

Address	Name	Function	R/W
SD166	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD350~SD351 according to SD166~SD167 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD350.	R/W
SD167	Accumulated pulse total number (LSB)		R/W
SD350	Current absolute position (MSB)		R/W
SD351	Current absolute position (LSB)		R/W
SD352	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~100000, unit: pulse	R/W
SD353	Max. speed (LSB)		R/W
SD354	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD355	ACC/DEC time	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD356	Clear signal element designation	When SM351 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD357	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD358	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD359	Origin return speed (LSB)		R/W
SD255	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R
Note: 1. SD352~SD355 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.			

- **Control and monitor output channels of output axis 4 (corresponding to Y6)**

Address	Name	Function	R/W
SM266	Pulse output stop control	Disable Y6 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM276	Pulse output monitor	Used to monitor the state of Y6 high-speed pulse output channel, ON when busy, OFF when ready	R

Address	Name	Function	R/W
SM76	Pulse output completion interrupt enabling control	Enable Y6 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM360	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y6: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM361	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y6: using corresponding Y element Y(N) of N in SD366 means clear signal when setting; set Y16 to clear signal according to the default value when resetting	R/W
SM362	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y6: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM363	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y6: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM364	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y6: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM365	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y6: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM366	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y6: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM368	Locating instruction in drive	Used to monitor the state of Y6 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W
Note: When SM360 is set, the corresponding default clear signal Y16 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

- **Special data registers of output axis 4 (corresponding to Y6)**

Address	Name	Function	R/W
SD168	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD360~SD361 according to SD168~SD169 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD360.	R/W
SD169	Accumulated pulse total number (LSB)		R/W
SD360	Current absolute position (MSB)		R/W
SD361	Current absolute position (LSB)		R/W
SD362	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~100000, unit: pulse	R/W
SD363	Max. speed (LSB)		R/W
SD364	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD365	ACC/DEC time	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD366	Clear signal element designation	When SM361 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD367	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD368	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD369	Origin return speed (LSB)		R/W
SD256	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R
Note: 1. SD362~SD365 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.			

- **Control and monitor output channels of output axis 5 (corresponding to Y7)**

Address	Name	Function	R/W
SM267	Pulse output stop control	Disable Y7 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM277	Pulse output monitor	Used to monitor the state of Y7 high-speed pulse output channel, ON when busy, OFF when ready	R

Address	Name	Function	R/W
SM77	Pulse output completion interrupt enabling control	Enable Y7 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM370	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y7: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM371	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y7: using corresponding Y element Y(N) of N in SD376 means clear signal when setting; set Y17 to clear signal according to the default value when resetting	R/W
SM372	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y7: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM373	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y7: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM374	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y7: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM375	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y7: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM376	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y7: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM378	Locating instruction in drive	Used to monitor the state of Y7 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W
Note: When SM370 is set, the corresponding default clear signal Y17 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

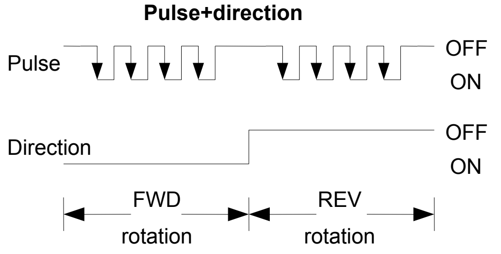
- **Special data registers of output axis 5 (corresponding to Y7)**

Address	Name	Function	R/W
SD170	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD370~SD371 according to SD170~SD171 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD370.	R/W
SD171	Accumulated pulse total number (LSB)		R/W
SD370	Current absolute position (MSB)		R/W
SD371	Current absolute position (LSB)		R/W
SD372	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~100000, unit: pulse	R/W
SD373	Max. speed (LSB)		R/W
SD374	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD375	ACC/DEC time	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD376	Clear signal element designation	When SM371 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD377	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD378	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD379	Origin return speed (LSB)		R/W
SD257	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R
Note: 1. SD372~SD375 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.			

11.4.2 Elements related to locating instructions of EC20 series

Definitions and distribution of output axes of EC20 series are shown in the following tables.

- **Definitions of output axes of EC20 series**

Output axis	Available mode	Definition of output point		Definition of output mode
0	Pulse+direction	Pulse	Y0	 Pulse+direction Pulse OFF ON Direction OFF ON FWD rotation REV rotation
		Direction	Unlimited except Y0	
1	Pulse+direction	Pulse	Y1	
		Direction	Unlimited except Y1	

- **Control and monitor output channels of output axis 0 (corresponding to Y0)**

Address	Name	Function	R/W
SM80	Pulse output stop control	Disable Y0 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM82	Pulse output monitor	Used to monitor the state of Y0 high-speed pulse output channel, ON when busy, OFF when ready	R
SM63	Pulse output completion interrupt enabling control	Enable Y0 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM280	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y0: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM281	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y0: using corresponding Y element Y(N) of N in SD206 means clear signal when setting; set Y10 to clear signal according to the default value when resetting	R/W
SM282	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y0: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM283	Forward rotation limit	Applicable to DSZR/DVIT, function on the corresponding axis of Y0: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM284	Reverse rotation limit	Applicable to DSZR/DVIT, function on the corresponding axis of Y0: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM285	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y0: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM286	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y0: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM287	Logic reverse rotation of interrupt signal	Applicable to DVIT, function on the corresponding axis of Y0: negative logic processing when setting (interrupt signal is ON when inputting OFF); positive logic processing when resetting (interrupt signal signal is ON when inputting ON)	R/W
SM288	Locating instruction in drive	Used to monitor the state of Y0 high-speed pulse output channel when executing DSZR/DVIT, ON when busy, OFF when ready	R/W
SM289	Y0 interrupt signal element valid	Applicable to DVIT, function on the corresponding axis of Y0: set X(N) in SD240 to interrupt input signal when setting; set X0 to interrupt input signal according to the default value when resetting	R/W
SM260	Module interrupt signal element valid	Applicable to DVIT, function on the corresponding axes of Y0 and Y1: use SM289, SM299 and SD240 to set interrupt input signal when setting; disable interrupt input signal setting when resetting	R/W
Note: When SM280 is set, the corresponding default clear signal Y10 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

- **Special data registers of output axis 0 (corresponding to Y0)**

Address	Name	Function	R/W
SD50	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD200~SD201 according to SD50~SD51 and direction signals. When starting the machine and reading	R/W

Address	Name	Function	R/W
SD51	Accumulated pulse total number (LSB)	absolute position data from servo drive, store the acquired data (32-bit long) into SD200.	R/W
SD200	Current absolute position (MSB)		R/W
SD201	Current absolute position (LSB)		R/W
SD202	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~100000, unit: pulse	R/W
SD203	Max. speed (LSB)		R/W
SD204	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD205	ACC/DEC time	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD207	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD208	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD209	Origin return speed (LSB)		R/W
SD220	Y0 clear signal element designation	When SM281 is set, using the corresponding Y element Y(N) of N in the element means clear signal	R/W
SD240	Y0 interrupt signal element designation	When SM289 is set, using the corresponding Y element Y(N) of N in the element means interrupt signal	R/W

Note:

1. SD202~SD205 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution.

2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, the base speed will assign 1/10 of the Max. speed. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output.

- **Control and monitor output channels of output axis 1 (corresponding to Y1)**

Address	Name	Function	R/W
SM81	Pulse output stop control	Disable Y1 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM83	Pulse output monitor	Used to monitor the state of Y1 high-speed pulse output channel, ON when busy, OFF when ready	R
SM64	Pulse output completion interrupt enabling control	Enable Y1 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM290	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y1: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM291	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y1: using corresponding Y element Y(N) of N in SD230 means clear signal when setting; set Y11 to clear signal according to the default value when resetting	R/W
SM292	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y1: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM293	Forward rotation limit	Applicable to DSZR/DVIT, function on the corresponding axis of Y1: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM294	Reverse rotation limit	Applicable to DSZR/DVIT, function on the corresponding axis of Y1: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM295	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y1: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM296	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y1: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM297	Logic reverse rotation of interrupt signal	Applicable to DVIT, function on the corresponding axis of Y1: negative logic processing when setting (interrupt signal is ON when inputting OFF); positive logic processing when resetting (interrupt signal signal is ON when inputting ON)	R/W
SM298	Locating instruction in drive	Used to monitor the state of Y1 high-speed pulse output channel when executing DSZR/DVIT, ON when busy, OFF when ready	R/W
SM299	Y1 interrupt signal element valid	Applicable to DVIT, function on the corresponding axis of Y1: set X(N) in SD240 to interrupt input signal when setting; set X1 to interrupt input signal according to the default value when resetting	R/W

Address	Name	Function	R/W
SM260	Module interrupt signal element valid	Applicable to DVIT, function on the corresponding axes of Y0 and Y1: use SM289, SM299 and SD240 to set interrupt input signal when setting; disable interrupt input signal setting when resetting	R/W
Note: When SM290 is set, the corresponding default clear signal Y11 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

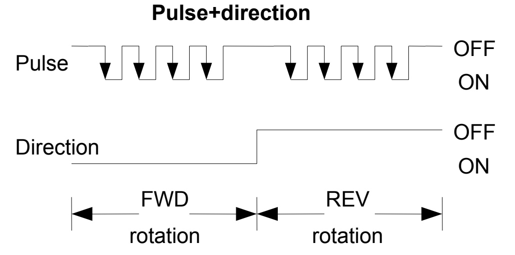
- **Special data registers of output axis 1 (corresponding to Y1)**

Address	Name	Function	R/W
SD52	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD210~SD211 according to SD52~SD53 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD210.	R/W
SD53	Accumulated pulse total number (LSB)		R/W
SD210	Current absolute position (MSB)		R/W
SD211	Current absolute position (LSB)		R/W
SD212	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~100000, unit: pulse	R/W
SD213	Max. speed (LSB)		R/W
SD214	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD215	ACC/DEC time	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD217	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD218	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD219	Origin return speed (LSB)		R/W
SD230	Clear signal element designation	When SM291 is set, using the corresponding Y element Y(N) of N in the element means clear signal	R/W
SD240	Interrupt signal element designation	When SM299 is set, using the corresponding Y element Y(N) of N in the element means interrupt signal	R/W
Note: 1. SD212~SD215 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, the base speed will assign 1/10 of the Max. speed. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output.			

11.4.3 Elements related to locating instructions of EC10V series

Definitions and distribution of output axes of EC10V series are shown in the following tables.

- **Definitions of output axes of EC10V**

Output axis	Available mode	Definition of output point		Definition of output mode
0	Pulse+direction	Pulse	Y0	**Pulse+direction**  Pulse OFF ON Direction OFF ON FWD rotation REV rotation
		Direction	Unlimited except Y0	
1	Pulse+direction	Pulse	Y1	
		Direction	Unlimited except Y1	
2	Pulse+direction	Pulse	Y2	
		Direction	Unlimited except Y2	
3	Pulse+direction	Pulse	Y3	
		Direction	Unlimited except Y3	

- **Control and monitor output channels of output axis 0 (corresponding to Y0)**

Address	Name	Function	R/W
SM80	Y0 pulse output stop control	Disable Y0 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM82	pulse output monitor	Used to monitor the state of Y0 high-speed pulse output channel, ON when	R

Address	Name	Function	R/W
		busy, OFF when ready	
SM63	pulse output completion interrupt enabling control	Enable Y0 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM280	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y0: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM281	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y0: using corresponding Y element Y(N) of N in SD206 means clear signal when setting; set Y10 to clear signal according to the default value when resetting	R/W
SM282	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y0: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM283	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y0: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM284	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y0: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM285	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y0: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM286	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y0: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM288	Locating instruction in drive	Used to monitor the state of Y0 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W
Note: When SM280 is set, the corresponding default clear signal Y10 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

- **Special data registers of output axis 0 (corresponding to Y0)**

Address	Name	Function	R/W
SD50	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD200~SD201 according to SD50~SD51 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD200.	R/W
SD51	Accumulated pulse total number (LSB)		R/W
SD200	Current absolute position (MSB)		R/W
SD201	Current absolute position (LSB)		R/W
SD202	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~200000, unit: pulse	R/W
SD203	Max. speed (LSB)		R/W
SD204	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD205	ACC time	The ACC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD260	DEC time	The DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD206	Clear signal element designation	When SM201 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD207	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD208	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD209	Origin return speed (LSB)		R/W
SD56	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R
Note: 1. SD202~SD205 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.			

- **Control and monitor output channels of output axis 1 (corresponding to Y1)**

Address	Name	Function	R/W

Address	Name	Function	R/W
SM81	Pulse output stop control	Disable Y1 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM83	Pulse output monitor	Used to monitor the state of Y1 high-speed pulse output channel, ON when busy, OFF when ready	R
SM64	Pulse output completion interrupt enabling control	Enable Y1 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM290	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y1: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM291	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y1: using corresponding Y element Y(N) of N in SD230 means clear signal when setting; set Y11 to clear signal according to the default value when resetting	R/W
SM292	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y1: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM293	Forward rotation limit	Applicable to DSZR/DVIT, function on the corresponding axis of Y1: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM294	Reverse rotation limit	Applicable to DSZR/DVIT, function on the corresponding axis of Y1: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM295	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y1: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM296	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y1: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM297	Logic reverse rotation of interrupt signal	Applicable to DVIT, function on the corresponding axis of Y1: negative logic processing when setting (interrupt signal is ON when inputing OFF); positive logic processing when resetting (interrupt signal signal is ON when inputing ON)	R/W
SM298	Locating instruction in drive	Used to monitor the state of Y1 high-speed pulse output channel when executing DSZR/DVIT, ON when busy, OFF when ready	R/W
SM299	Y1 interrupt signal element valid	Applicable to DVIT, function on the corresponding axis of Y1: set X(N) in SD240 to interrupt input signal when setting; set X1 to interrupt input signal according to the default value when resetting	R/W
SM260	Module interrupt signal element valid	Applicable to DVIT, function on the corresponding axes of Y0 and Y1: use SM289, SM299 and SD240 to set interrupt input signal when setting; disable interrupt input signal setting when resetting	R/W
Note: When SM290 is set, the corresponding default clear signal Y11 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

- **Special data registers of output axis 1 (corresponding to Y1)**

Address	Name	Function	R/W
SD52	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD210~SD211 according to SD52~SD53 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD210.	R/W
SD53	Accumulated pulse total number (LSB)		R/W
SD210	Current absolute position (MSB)		R/W
SD211	Current absolute position (LSB)		R/W
SD212	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~100000, unit: pulse	R/W
SD213	Max. speed (LSB)		R/W
SD214	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD215	ACC time	The ACC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD261	DEC time	The DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD217	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD218	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the	R/W

Address	Name	Function	R/W
SD219	Origin return speed (LSB)	instruction	R/W
SD230	Clear signal element designation	When SM291 is set, using the corresponding Y element Y(N) of N in the element means clear signal	R/W
SD240	Interrupt signal element designation	When SM299 is set, using the corresponding Y element Y(N) of N in the element means interrupt signal	R/W
Note: 1. SD212~SD215 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, the base speed will assign 1/10 of the Max. speed. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output.			

- **Control and monitor output channels of output axis 2 (corresponding to Y2)**

Address	Name	Function	R/W
SM262	Pulse output stop control	Disable Y2 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM272	Pulse output monitor	Used to monitor the state of Y2 high-speed pulse output channel, ON when busy, OFF when ready	R
SM72	Pulse output completion interrupt enabling control	Enable Y2 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM320	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y2: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM321	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y2: using corresponding Y element Y(N) of N in SD326 means clear signal when setting; set Y12 to clear signal according to the default value when resetting	R/W
SM322	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y2: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM323	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y2: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM324	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y2: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM325	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y2: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM326	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y2: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM328	Locating instruction in drive	Used to monitor the state of Y2 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W
Note: When SM320 is set, the corresponding default clear signal Y12 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

- **Special data registers of output axis 2 (corresponding to Y2)**

Address	Name	Function	R/W
SD160	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD320~SD321 according to SD160~SD161 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD320.	R/W
SD161	Accumulated pulse total number (LSB)		R/W
SD320	Current absolute position (MSB)		R/W
SD321	Current absolute position (LSB)		R/W
SD322	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~200000, unit: pulse	R/W
SD323	Max. speed (LSB)		R/W
SD324	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD325	ACC time	The ACC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD262	DEC time	The DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W

Address	Name	Function	R/W
SD326	Clear signal element designation	When SM321 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD327	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD328	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD329	Origin return speed (LSB)		R/W
SD252	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R
Note: 1. SD322~SD325 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.			

- **Control and monitor output channels of output axis 3 (corresponding to Y3)**

Address	Name	Function	R/W
SM263	Pulse output stop control	Disable Y3 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM273	Pulse output monitor	Used to monitor the state of Y3 high-speed pulse output channel, ON when busy, OFF when ready	R
SM73	Pulse output completion interrupt enabling control	Enable Y4 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM330	Clear function valid	Applicable to DSZR/ZRN, function on the corresponding axis of Y3: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM331	Clear signal designated element valid	Applicable to DSZR, function on the corresponding axis of Y3: using corresponding Y element Y(N) of N in SD346 means clear signal when setting; set Y14 to clear signal according to the default value when resetting	R/W
SM332	Origin return direction	Applicable to DSZR, function on the corresponding axis of Y3: origin return direction is forward rotation direction when setting; origin return direction is reverse rotation direction when resetting	R/W
SM333	Forward rotation limit	Applicable to DSZR, function on the corresponding axis of Y3: forward rotation limit arrival when setting; resetting means forward rotation limit non-arrival when resetting	R/W
SM334	Reverse rotation limit	Applicable to DSZR, function on the corresponding axis of Y3: reverse rotation limit arrival when setting; reverse rotation limit non-arrival when resetting	R/W
SM335	Logic reverse rotation of near-point signal	Applicable to DSZR, function on the corresponding axis of Y3: negative logic processing when setting (near-point signal is ON when inputting OFF); positive logic processing when resetting (near-point signal is ON when inputting ON)	R/W
SM336	Logic reverse rotation of zero-point signal	Applicable to DSZR, function on the corresponding axis of Y3: negative logic processing when setting (zero-point signal is ON when inputting OFF); positive logic processing when resetting (zero-point signal is ON when inputting ON)	R/W
SM338	Locating instruction in drive	Used to monitor the state of Y3 high-speed pulse output channel when executing DSZR, ON when busy, OFF when ready	R/W
Note: When SM330 is set, the corresponding default clear signal Y13 of the output axis will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If the default clear signal is used in other ways, reset valid clear signal to disable the function.			

- **Special data registers of output axis 3 (corresponding to Y3)**

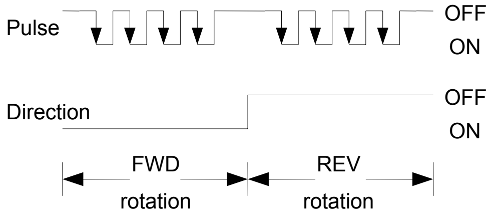
Address	Name	Function	R/W
SD162	Accumulated pulse total number (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD330~SD331 according to SD162~SD163 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD330.	R/W
SD163	Accumulated pulse total number (LSB)		R/W
SD330	Current absolute position (MSB)		R/W
SD331	Current absolute position (LSB)		R/W
SD332	Max. speed (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~200000, unit: pulse	R/W
SD333	Max. speed (LSB)		R/W
SD334	Base speed	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD335	ACC time	The ACC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W

Address	Name	Function	R/W
SD263	DEC time	The DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD336	Clear signal element designation	When SM331 is set, the corresponding Y element Y(N) of N in the element is clear signal	R/W
SD337	Crawling speed	Applicable to DSZR, function as the crawling speed when executing the instruction	R/W
SD338	Origin return speed (MSB)	Applicable to DSZR, function as the origin return speed when executing the instruction	R/W
SD339	Origin return speed (LSB)		R/W
SD253	Envelope output stage	Applicable to PLS, used to detect the envelope output stage	R
Note: 1. SD332~SD335 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.			

11.4.4 Elements related to locating instructions of EC10 series

Definitions and distribution of output axes of EC10 series are shown in the following tables.

- **Definitions of output axes of EC10**

Output axis	Available mode	Definition of output point		Definition of output mode
0	Pulse+direction	Pulse	Y0	**Pulse+direction**  Pulse OFF ON Direction OFF ON FWD rotation REV rotation
		Direction	Unlimited except Y0	
1	Pulse+direction	Pulse	Y1	
		Direction	Unlimited except Y1	

- **Control and monitor of output channels**

Address	Name	Function	R/W
SM80	Y0 pulse output stop control	Disable Y0 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM81	Y1 pulse output stop control	Disable Y1 high-speed pulse output function when setting; enable the output function when resetting	R/W
SM82	Y0 pulse output monitor	Used to monitor the state of Y0 high-speed pulse output channel, ON when busy, OFF when ready	R
SM83	Y1 pulse output monitor	Used to monitor the state of Y1 high-speed pulse output channel, ON when busy, OFF when ready	R
SM63	Y0 pulse output completion interrupt enabling control	Enable Y0 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM64	Y1 pulse output completion interrupt enabling control	Enable Y1 high-speed pulse output completion interrupt when setting; disable pulse output completion interrupt when resetting	R/W
SM85	Clear function valid	Applicable to ZRN, function on the corresponding axes of Y0 and Y1: CLR signal output of origin return instruction is valid when setting; no CLR signal output when resetting	R/W
SM86	Y0 interrupt drive pulse output valid	ON: PLSY instructions can be called in interrupt programs and subprograms, and driven continuously and repeatedly with power flow in main programs	R/W
SM87	Y1 interrupt drive pulse output valid	ON: PLSY instructions can be called in interrupt programs and subprograms, and driven continuously and repeatedly with power flow in main programs	R/W
SM89	PLSV gradual frequency conversion	ON: frequency changes gradually	R/W
Note: When SM85 is set, Y2 or Y3 will send 1 CLR pulse output at origin return arrival and the pulse width is 20ms+1 scan cycle. If Y2 or Y3 is used in other ways, resetting SM85 will disable the function.			

- **Special data registers of output channels**

Address	Name	Function	R/W
SD50	Accumulated pulse total number of output axis 0 (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD80~SD81 according to SD50~SD51 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD80.	R/W
SD51	Accumulated pulse total number of output axis 0 (LSB)		R/W
SD80	Current absolute position of output axis 0 (MSB)		R/W
SD81	Current absolute position of output axis 0 (LSB)		R/W
SD52	Accumulated pulse total number of output axis 1 (MSB)	Used to calculate and store absolute position. Each time the position instruction is executed, calculate and update SD82~SD83 according to SD52~SD53 and direction signals. When starting the machine and reading absolute position data from servo drive, store the acquired data (32-bit long) into SD82.	R/W
SD53	Accumulated pulse total number of output axis 1 (LSB)		R/W
SD82	Current absolute position of output axis 1 (MSB)		R/W
SD83	Current absolute position of output axis 1 (LSB)		R/W
SD84	Base speed of output axes 0 and 1	The base speed when the output axis executes the locating instruction (below 1/10 of the Max. speed)	R/W
SD85	Max. speed of output axes 0 and 1 (MSB)	The Max. speed when the output axis executes the locating instruction, range: 10~100000, unit: pulse	R/W
SD86	Max. speed of output axes 0 and 1 (LSB)		R/W
SD87	ACC/DEC time of output axes 0 and 1	The ACC/DEC time when the output axis executes the locating instruction, range: 50~5000, unit: ms	R/W
SD56	Envelope output stage of output axis 0	Applicable to PLS, used to detect the envelope output stage	R
SD57	Envelope output stage of output axis 1	Applicable to PLS, used to detect the envelope output stage	R
Note: 1. SD84~SD87 can be changed by users according to needs. Please ensure to assign the value before driving the locating instruction. Changing above parameters when executing the locating instruction may affect correct execution. 2. The base speed must be smaller than 1/10 of the Max. speed; otherwise, no pulse output. If the speed of the locating instruction is lower than the base speed or higher than the Max. speed, no pulse output, either.			

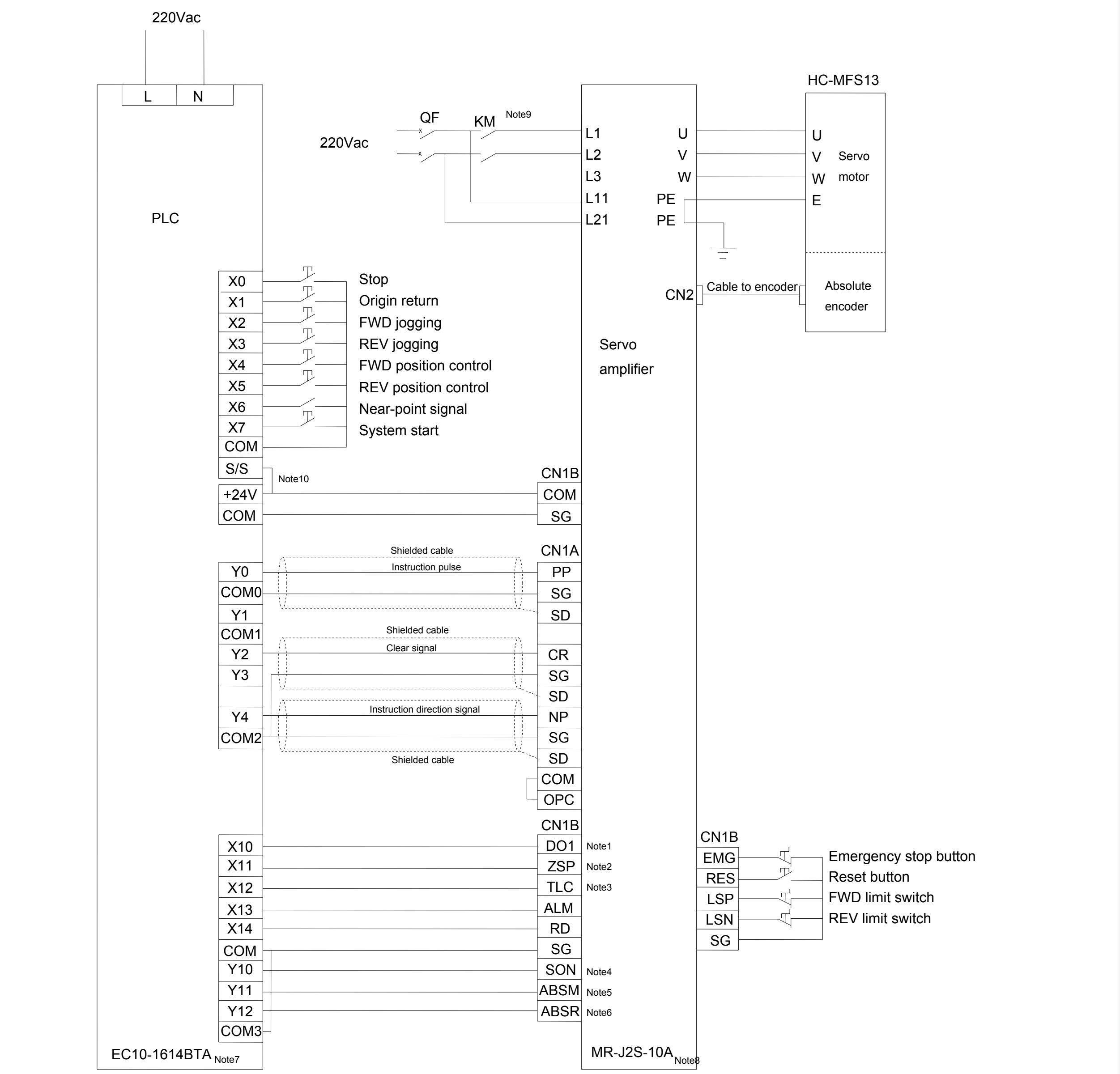
11.5 Examples

Example of pulse output program

- **Mechanical diagram**

Refer to the schematic diagram in 6.17.3. The example is about the absolute coordinate system of single-shaft lead screw.

- **Wiring diagram**



Note1: When ABSM is set to ON, it is ABS bit1 data line; when ABSM is set to OFF, it is positioning complete signal.
Note2: When ABSM is set to ON, it is ABS bit2 data line; when ABSM is set to OFF, it is zero speed signal.
Note3: When ABSM is set to ON, it is transfer data ready signal; when ABSM is set to OFF, it is torque limit signal.
Note4: Servo enabling signal, set before ABS is executed.
Note5: ABS transfer mode signal.
Note6: ABS transfer request signal.
Note7: PLC main module which must adopt transistor output.
Note8: Please install according to installation instructions of servo amplifier in relevant models; the plugs of multiple ports are totally the same in shape,, so be careful not to plug improperly.
Note9: In alarm or emergency situations, need program to control KM and disconnect the power supply.
Note10: In the example, PLC adopts sink input, short circuit +24V and S/S terminals.

- **Program examples**

The program realizes the following functions:

1. When PLC enters the running state, read abosute position data of servo drive via ABS instruction or communication mode (servo drive is required to power on before or at the same time PLC runs).

2. Enter the running state, set SM85 and clear output function. Each time origin returns, Y2 will output clear pulse.

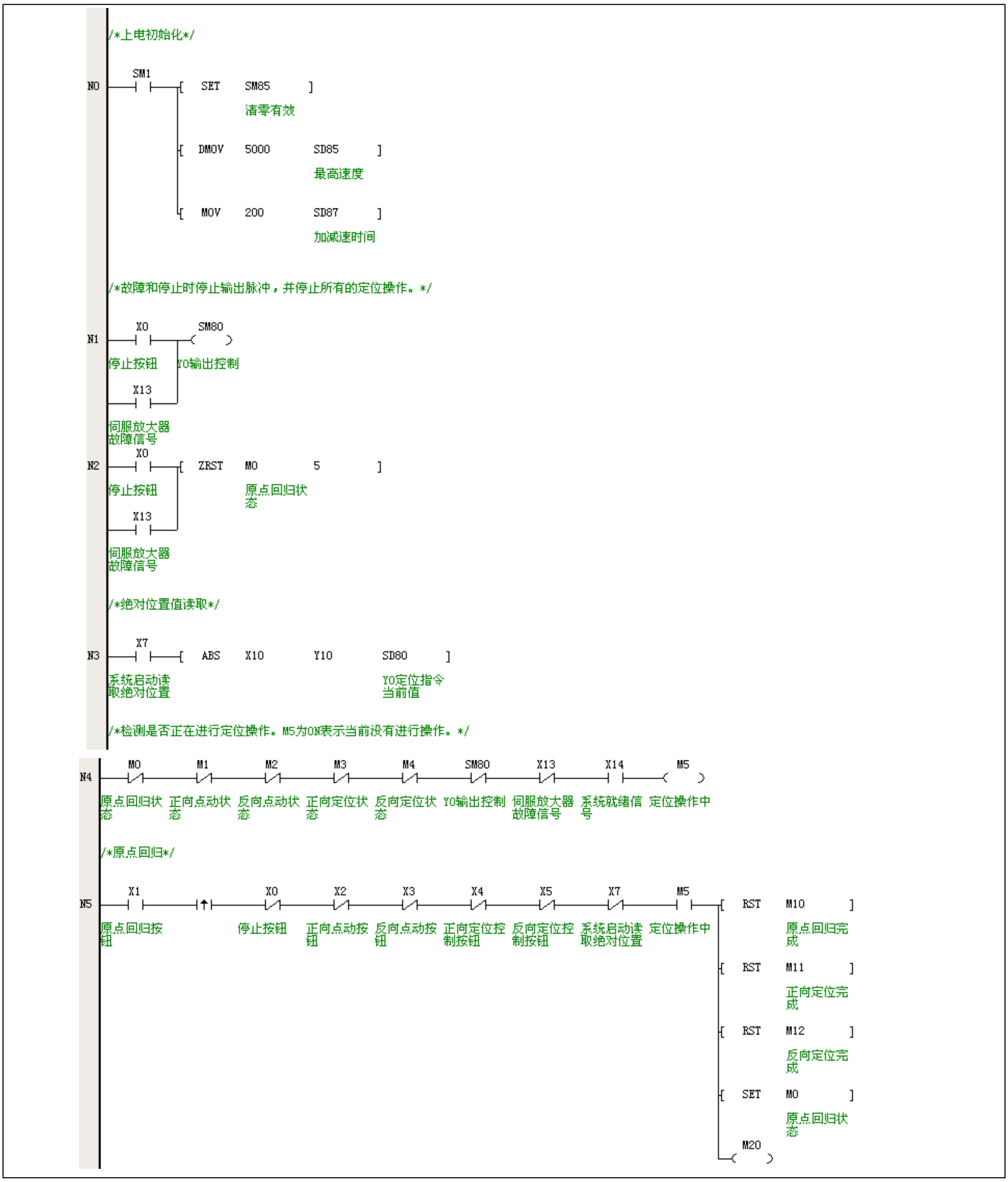
3. Operate the JOG+ button for jogging in forward direction.

4. Operate the JOG- button for jogging in reverse direction.

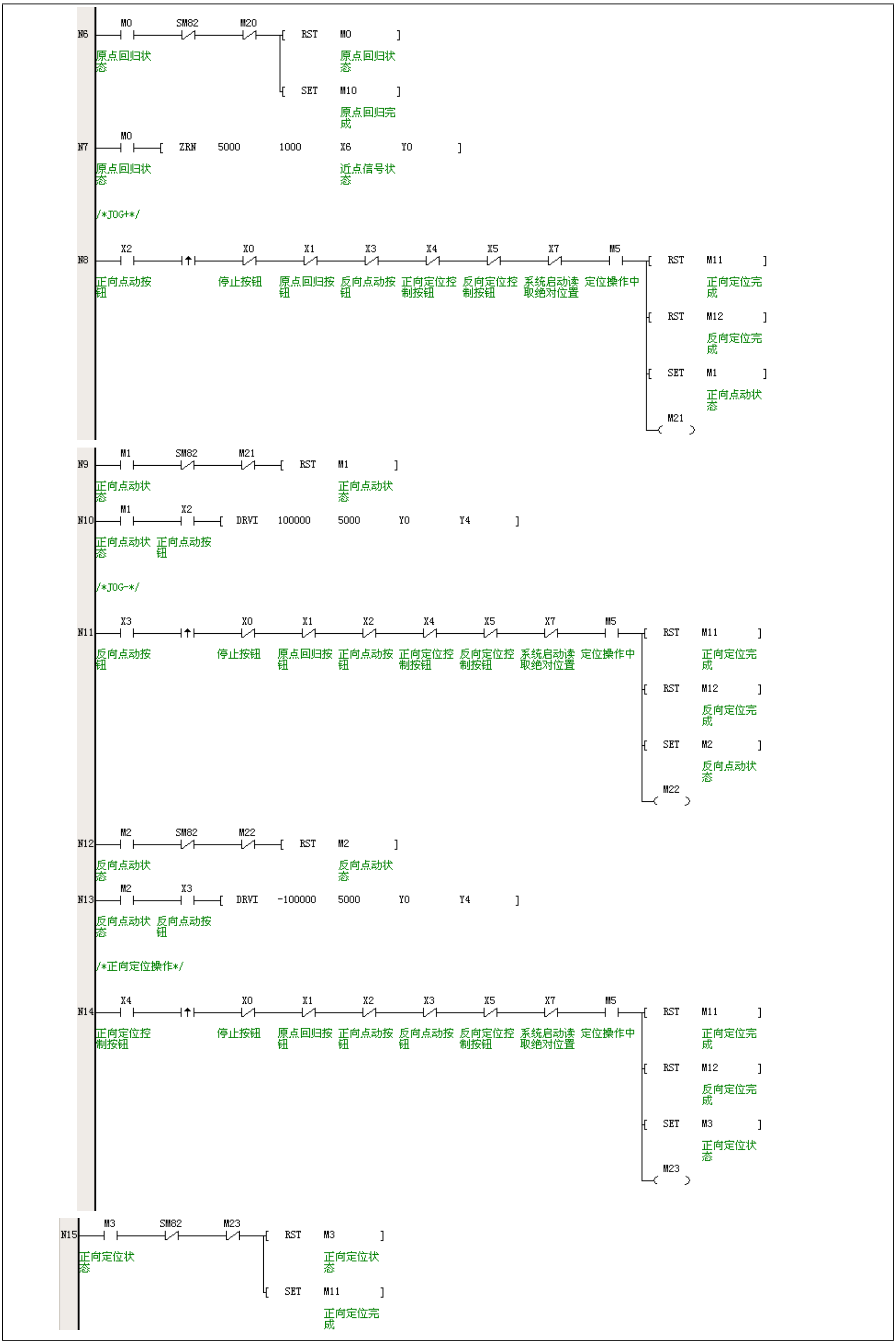
5. When the console locates farther than the front of near-point signal, operate the origin return button for origin return action in backward direction.

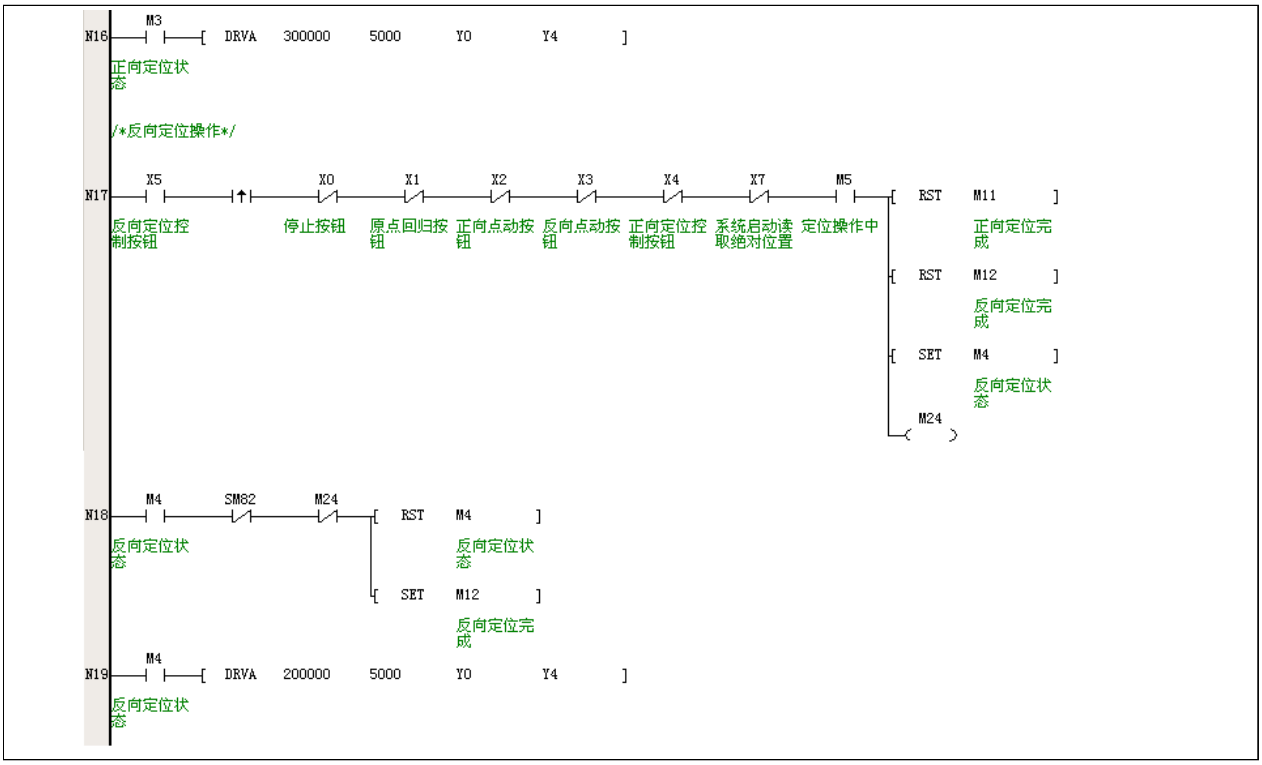
6. Press the stop button when the console is running and the console will stop running.

7. Operate the forward position control and reverse position control buttons for console positioning.



```
/*上电初始化*/

      SM1
N0  ──┤ ├──┬─[ SET    SM85        ]
           │         清零有效
           ├─[ DMOV   5000   SD85        ]
           │                 最高速度
           └─[ MOV    200    SD87        ]
                             加减速时间

/*故障和停止时停止输出脉冲，并停止所有的定位操作。*/

      X0          SM80
N1  ──┤ ├──┬──( )
    停止按钮 │ Y0输出控制
      X13   │
    ──┤ ├──┘
    伺服放大器
    故障信号
      X0
N2  ──┤ ├──┬─[ ZRST   M0     5           ]
    停止按钮 │         原点回归状
           │         态
      X13   │
    ──┤ ├──┘
    伺服放大器
    故障信号

/*绝对位置值读取*/

      X7
N3  ──┤ ├────[ ABS    X10    Y10    SD80        ]
    系统启动读                       Y0定位指令
    取绝对位置                       当前值

/*检测是否正在进行定位操作。M5为ON表示当前没有进行操作。*/

      M0       M1       M2       M3       M4       SM80     X13      X14      M5
N4  ──┤/├──────┤/├──────┤/├──────┤/├──────┤/├──────┤/├──────┤/├──────┤ ├──────( )
    原点回归状 正向点动状 反向点动状 正向定位状 反向定位状 Y0输出控制 伺服放大器 系统就绪信 定位操作中
    态        态        态        态        态                  故障信号   号

/*原点回归*/

      X1                X0       X2       X3       X4       X5       X7       M5
N5  ──┤ ├──────┤↑├──────┤/├──────┤/├──────┤/├──────┤/├──────┤/├──────┤/├──────┤ ├──┬─[ RST    M10        ]
    原点回归按           停止按钮   正向点动按 反向点动按 正向定位控 反向定位控 系统启动读 定位操作中 │         原点回归完
    钮                            钮        钮        制按钮    制按钮    取绝对位置         │         成
                                                                                      ├─[ RST    M11        ]
                                                                                      │         正向定位完
                                                                                      │         成
                                                                                      ├─[ RST    M12        ]
                                                                                      │         反向定位完
                                                                                      │         成
                                                                                      ├─[ SET    M0         ]
                                                                                      │         原点回归状
                                                                                      │         态
                                                                                      │   M20
                                                                                      └──( )
```

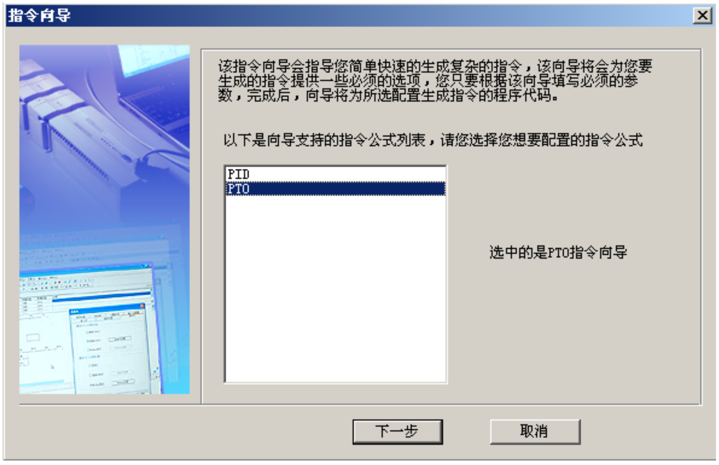

N6
M0
SM82
M20
RST M0
原点回归状态
原点回归状态
SET M10
原点回归完成
N7
M0
ZRN 5000 1000 X6 Y0
原点回归状态
近点信号状态
/*JOG+*/
N8
X2
X0
X1
X3
X4
X5
X7
M5
正向点动按钮
停止按钮
原点回归按钮
反向点动按钮
正向定位控制按钮
反向定位控制按钮
系统启动读取绝对位置
定位操作中
RST M11
正向定位完成
RST M12
反向定位完成
SET M1
正向点动状态
M21
N9
M1
SM82
M21
RST M1
正向点动状态
正向点动状态
N10
M1
X2
DRVI 100000 5000 Y0 Y4
正向点动状态
正向点动按钮
/*JOG-*/
N11
X3
X0
X1
X2
X4
X5
X7
M5
反向点动按钮
停止按钮
原点回归按钮
正向点动按钮
正向定位控制按钮
反向定位控制按钮
系统启动读取绝对位置
定位操作中
RST M11
正向定位完成
RST M12
反向定位完成
SET M2
反向点动状态
M22
N12
M2
SM82
M22
RST M2
反向点动状态
反向点动状态
N13
M2
X3
DRVI -100000 5000 Y0 Y4
反向点动状态
反向点动按钮
/*正向定位操作*/
N14
X4
X0
X1
X2
X3
X5
X7
M5
正向定位控制按钮
停止按钮
原点回归按钮
正向点动按钮
反向点动按钮
反向定位控制按钮
系统启动读取绝对位置
定位操作中
RST M11
正向定位完成
RST M12
反向定位完成
SET M3
正向定位状态
M23
N15
M3
SM82
M23
RST M3
正向定位状态
正向定位状态
SET M11
正向定位完成



```
N16  M3
     ─┤├─[ DRVA  300000  5000  Y0  Y4 ]
     正向定位状态

/*反向定位操作*/

N17  X5         X0       X1         X2         X3         X4          X7              M5
     ─┤├─┤↑├─┤/├─┤/├─┤/├─┤/├─┤/├─┤/├─┤├─┬─[ RST  M11 ]
     反向定位控制按钮  停止按钮  原点回归按钮  正向点动按钮  反向点动按钮  正向定位控制按钮  系统启动读取绝对位置  定位操作中  │  正向定位完成
                                                                                        ├─[ RST  M12 ]
                                                                                        │  反向定位完成
                                                                                        ├─[ SET  M4 ]
                                                                                        │  反向定位状态
                                                                                        └─( M24 )

N18  M4      SM82    M24
     ─┤├──┤/├──┤/├─┬─[ RST  M4 ]
     反向定位状态          │  反向定位状态
                       └─[ SET  M12 ]
                          反向定位完成

N19  M4
     ─┤├─[ DRVA  200000  5000  Y0  Y4 ]
     反向定位状态
```

- **Configuration of PLS envelope instruction**

PLS envelope instruction is generated by PTO instruction wizard. In Programmer, select **Tool->Instruction Wizard…**, to carry out the relevant configuration of PLS instruction, as shown below, select PTO.



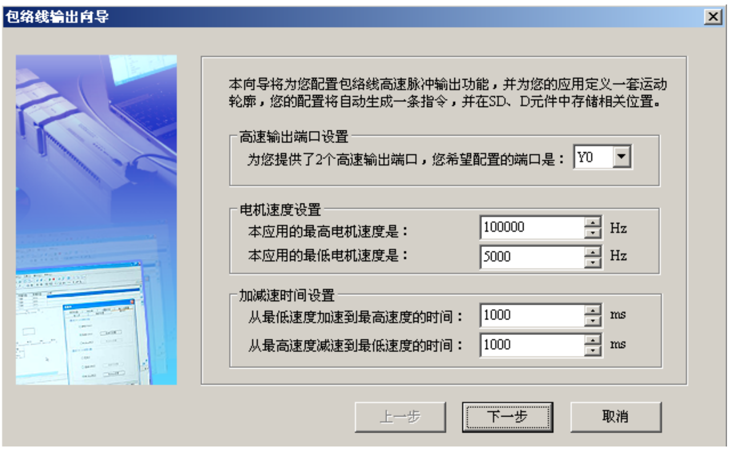
该指令向导会指导您简单快速的生成复杂的指令，该向导将会为您要生成的指令提供一些必须的选项，您只要根据该向导填写必须的参数，完成后，向导将为所选配置生成指令的程序代码。

以下是向导支持的指令公式列表，请您选择您想要配置的指令公式

PID
PTO

选中的是PTO指令向导

Click **Next** to enter the interface of high-speed pulse output point, Max. and Min. frequency of high-speed pulse output, and ACC/DEC time, as shown below:



本向导将为您配置包络线高速脉冲输出功能，并为您的应用定义一套运动轮廓，您的配置将自动生成一条指令，并在SD、D元件中存储相关位置。

高速输出端口设置
为您提供了2个高速输出端口，您希望配置的端口是：Y0

电机速度设置
本应用的最高电机速度是：100000 Hz
本应用的最低电机速度是：5000 Hz

加减速时间设置
从最低速度加速到最高速度的时间：1000 ms
从最高速度减速到最低速度的时间：1000 ms

The accelerated speed for all stages of envelope in acceleration/deceleration is definite. For example, if the setting goes as above figure, thus the ACC time required by the motor accelerating from 20000Hz to 50000Hz is:

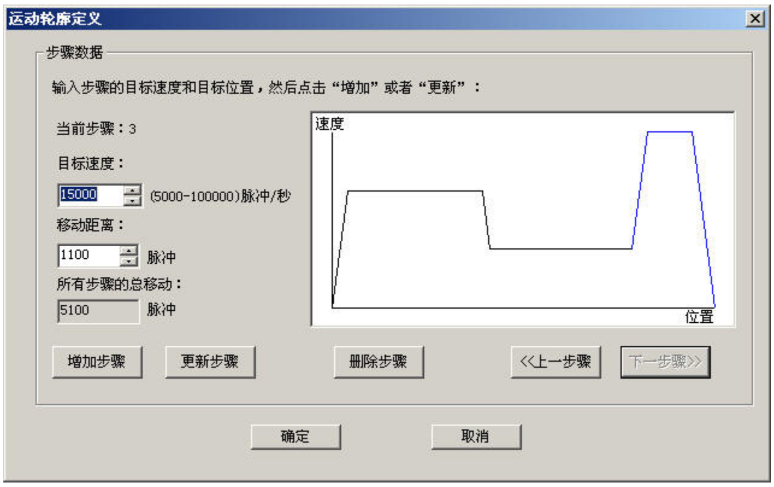
1000×(50000-20000)÷(100000-5000)=316(ms)=0.316(s)

During ACC time, the total output pulse number can be achieved by trapezoid area calculation formula:

(20000+50000)×0.316÷2=11060(total pulse number)

Therefore, if the ACC/DEC time or pulse number is required, please make relevant calcultation and then set the Max. speed, Min. speed and ACC/DEC time.

Click **Next** in above figure and then enter **Motion Contour Definition** in below figure. Input the target speed and move distance of the first step and click **Add Step**. Input the target speed and move distance of the second step and click **Add Step**, and so on. Finally, click **Finish**.



运动轮廓定义

步骤数据

输入步骤的目标速度和目标位置，然后点击“增加”或者“更新”：

当前步骤：3

目标速度：

15000 (5000-100000)脉冲/秒

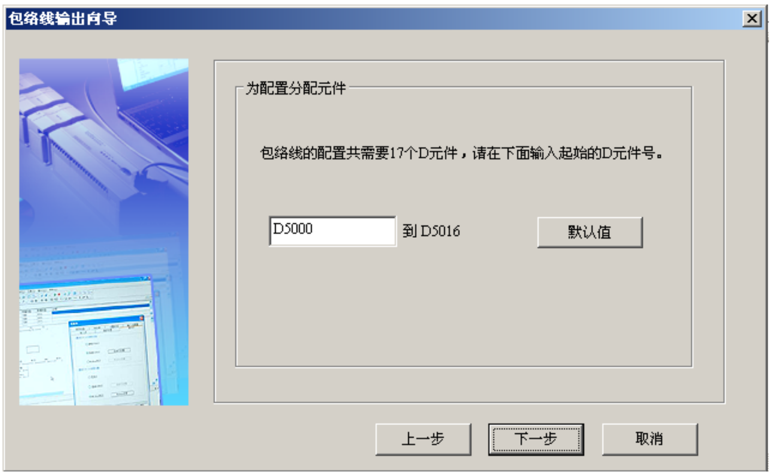
移动距离：

1100 脉冲

所有步骤的总移动：

5100 脉冲

The above configuration will be stored in D elements. You can choose which D elements are used to store the settings.



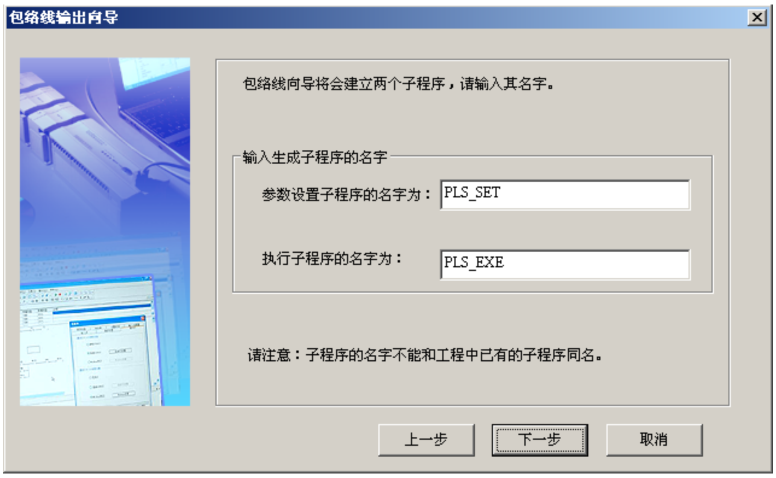
包络线输出向导

为配置分配元件

包络线的配置共需要17个D元件，请在下面输入起始的D元件号。

D5000 到 D5016

Set up two subprograms for envelope output wizard, one is parameter setting subprogram and the other is PLS execution subprogram, as shown below. When programming the main program, ensure to call the PLS execution subprogram after properly calling and executing the parameter setting subprogram (relevant D elements will be assigned).



包络线输出向导

包络线向导将会建立两个子程序，请输入其名字。

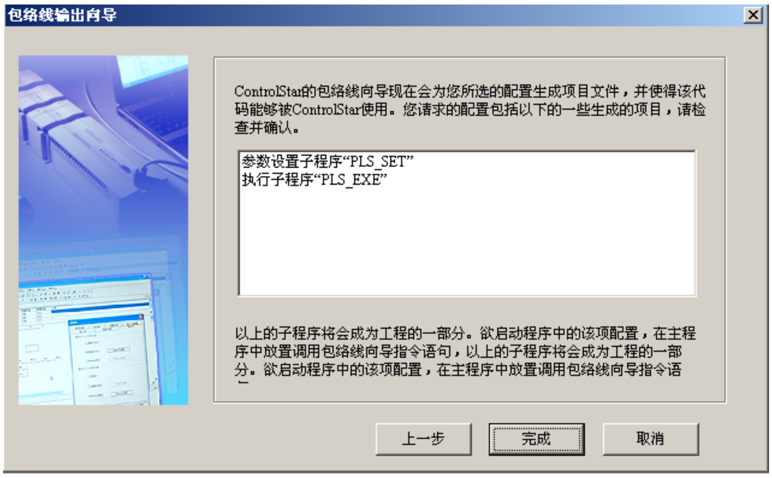
输入生成子程序的名字

参数设置子程序的名字为：PLS_SET

执行子程序的名字为：PLS_EXE

请注意：子程序的名字不能和工程中已有的子程序同名。

So far, all configuration is completed, as shown below, click **Finish** to end PTO configuration.



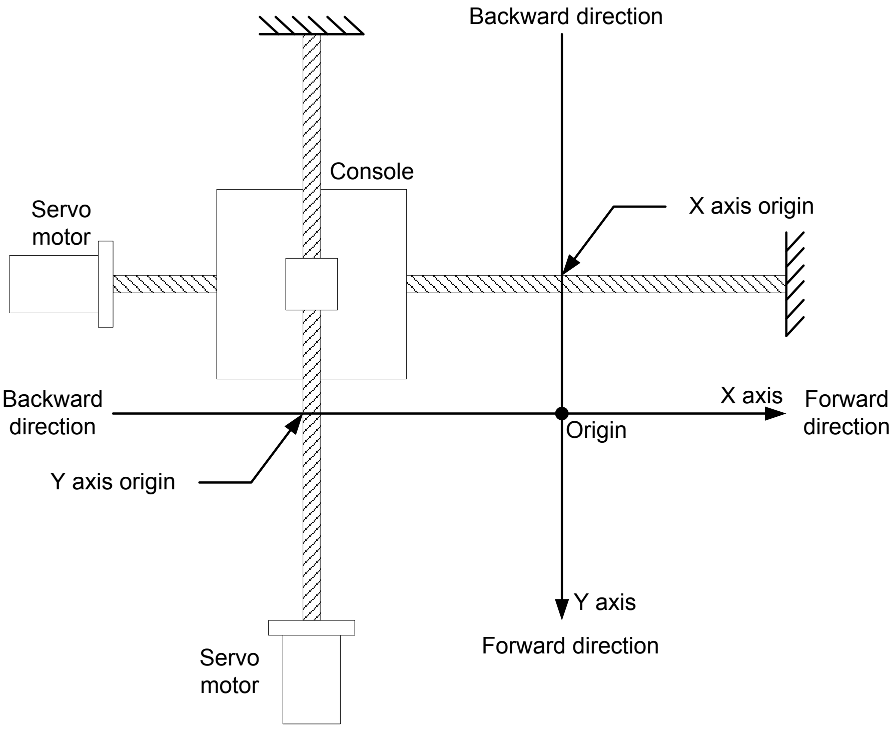
ControlStar的包络线向导现在会为您所选的配置生成项目文件，并使得该代码能够被ControlStar使用。您请求的配置包括以下的一些生成的项目，请检查并确认。

参数设置子程序“PLS_SET”
执行子程序“PLS_EXE”

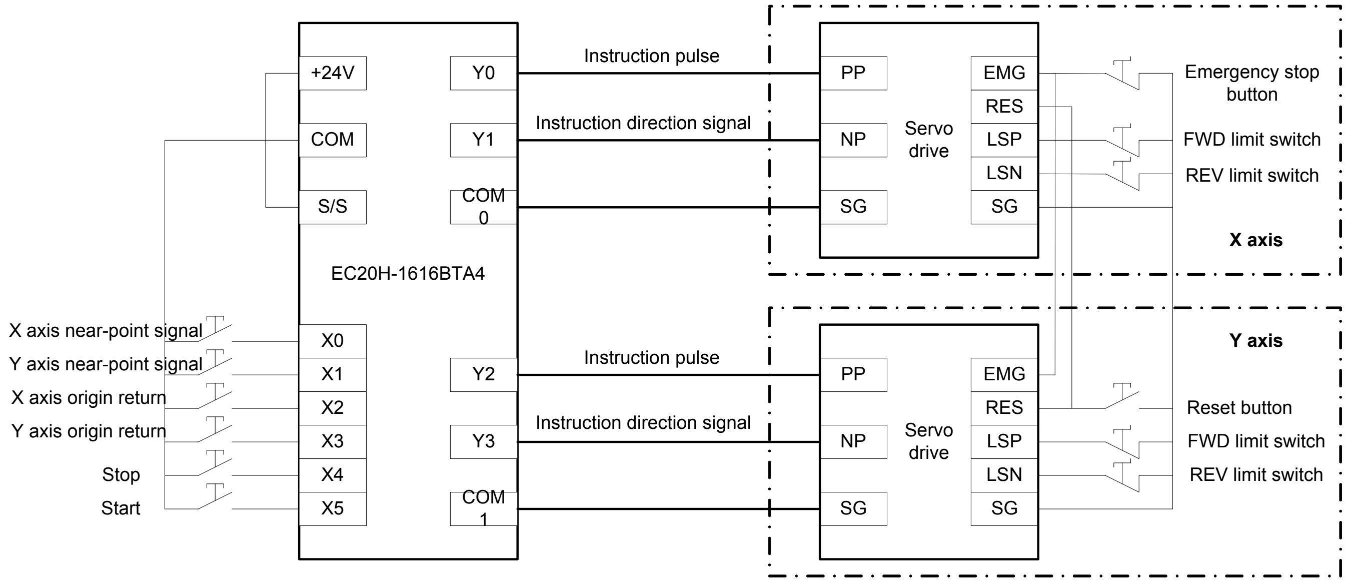
以上的子程序将会成为工程的一部分。欲启动程序中的该项配置，在主程序中放置调用包络线向导指令语句，以上的子程序将会成为工程的一部分。欲启动程序中的该项配置，在主程序中放置调用包络线向导指令语

11.5.2 Example of trace interpolation program

- **Mechanical diagram**



- **Wiring diagram**



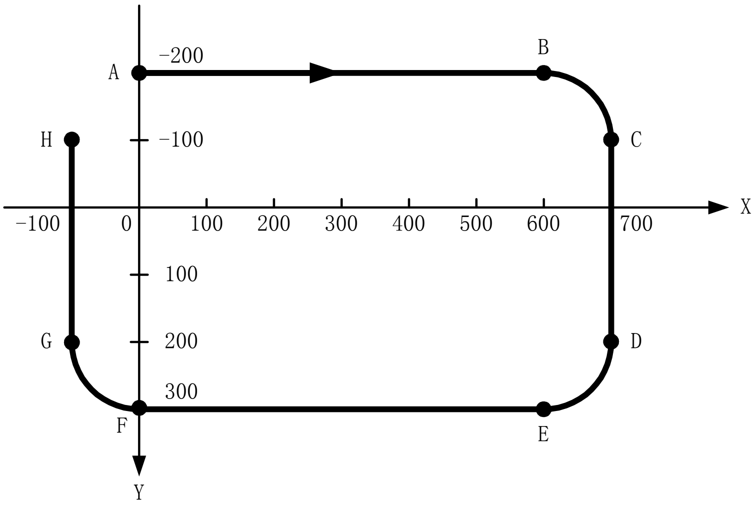
- **Program examples**

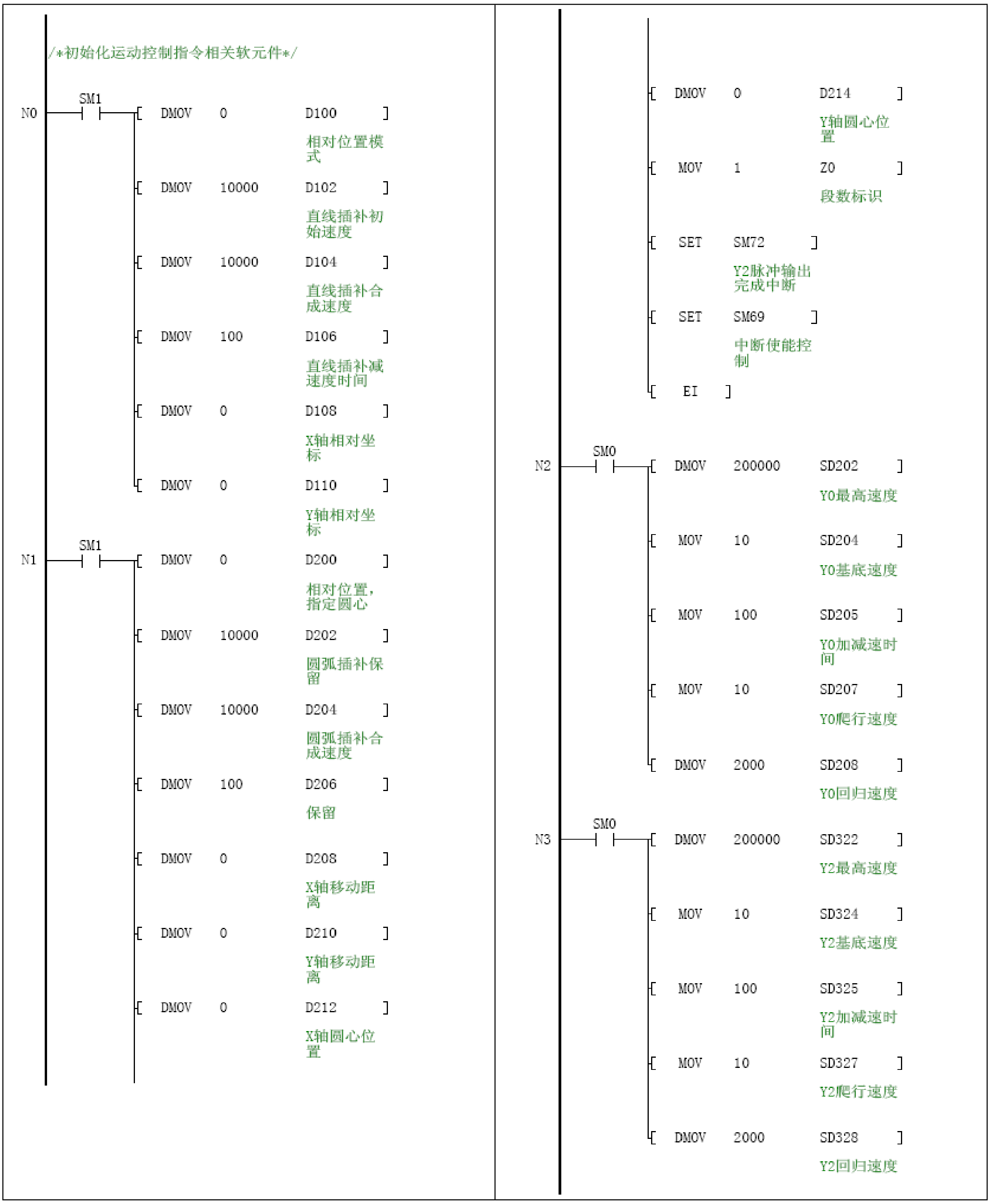
The program realizes the following functions:

1. Operate the origin return button, the console will perform origin automatic search and stop running when reaching the origin.

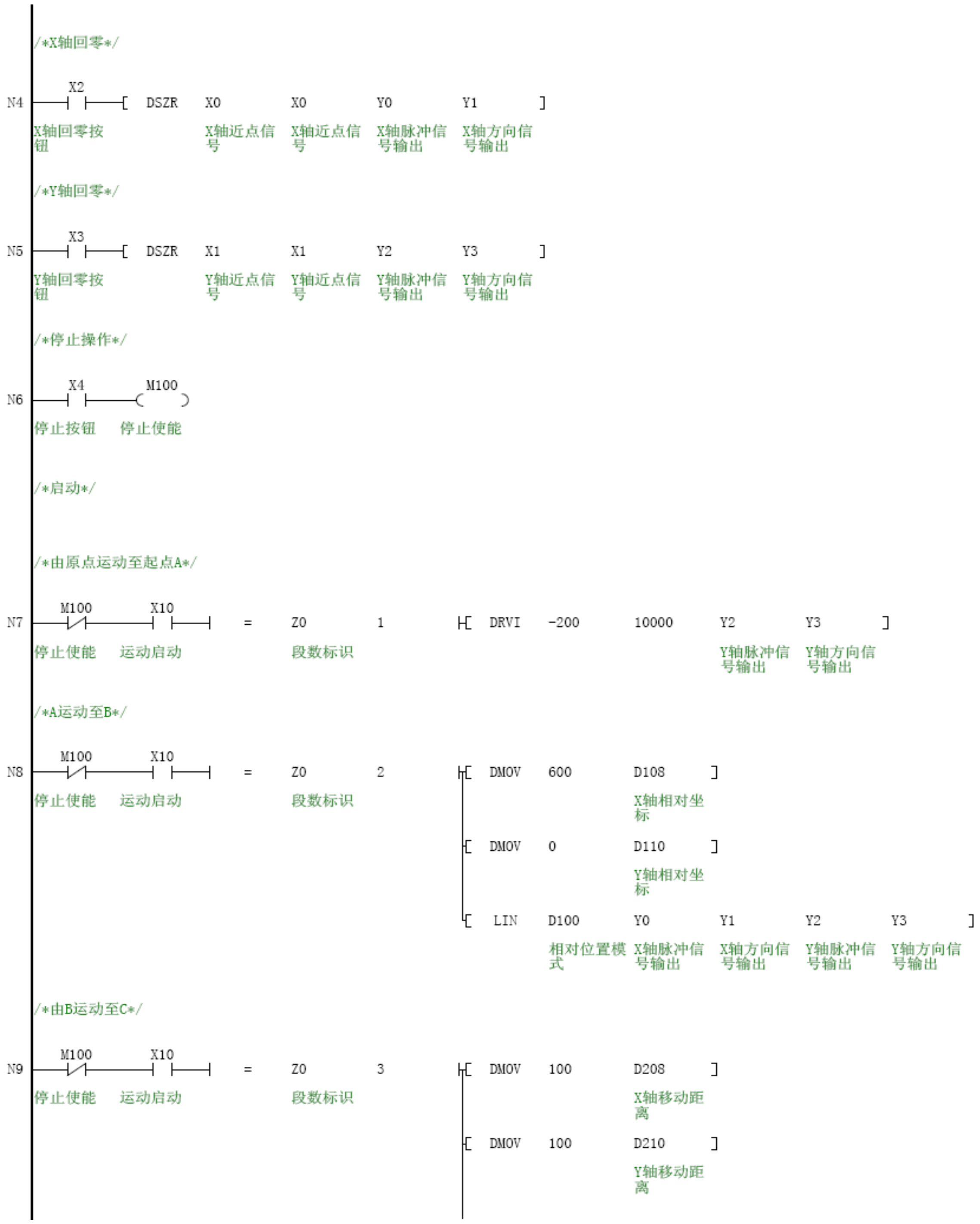
2. Enter the running state, operate the start button and run continuously in the trace shown in the diagram.

3. Press the stop button when the console is running and the console will stop running.

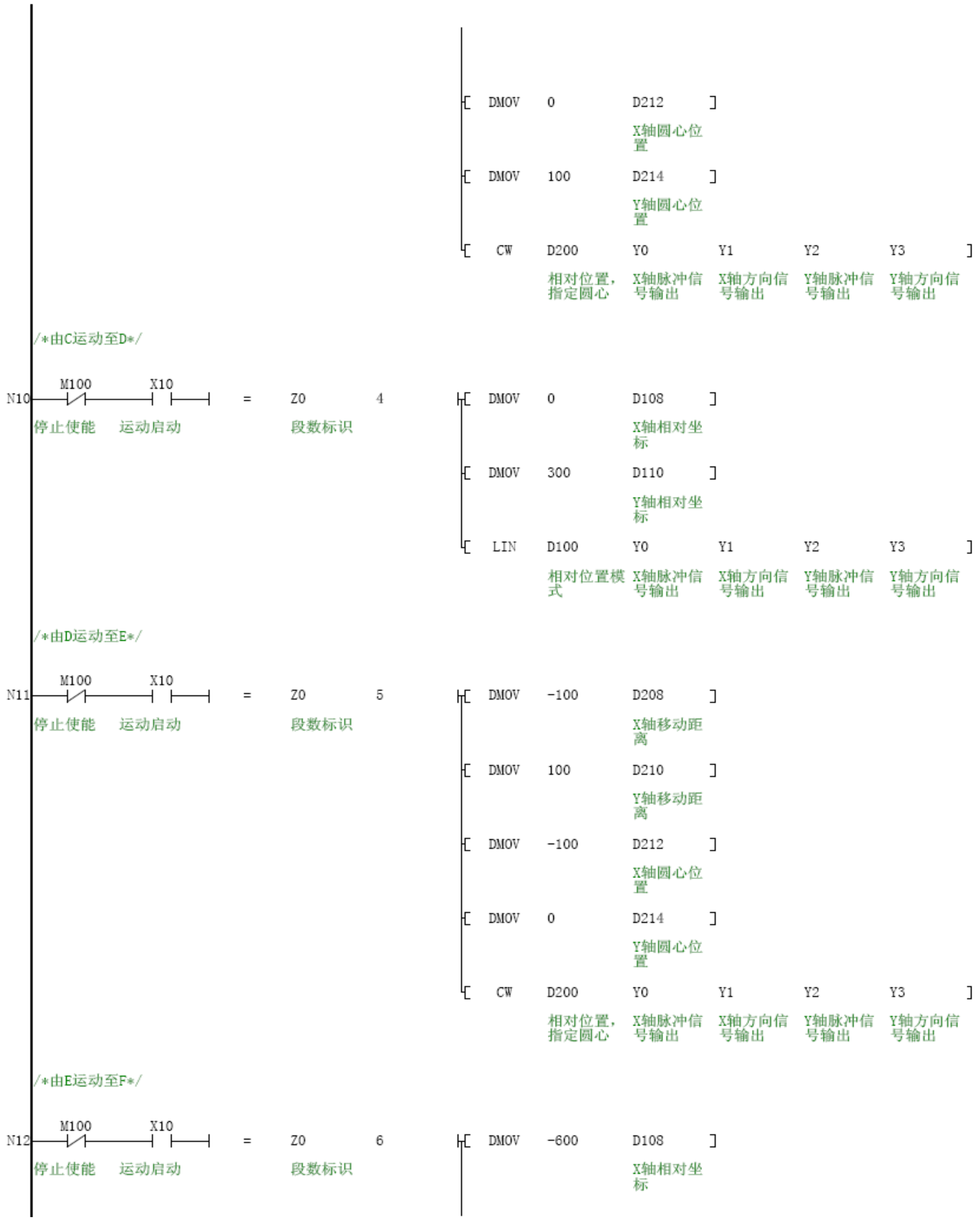




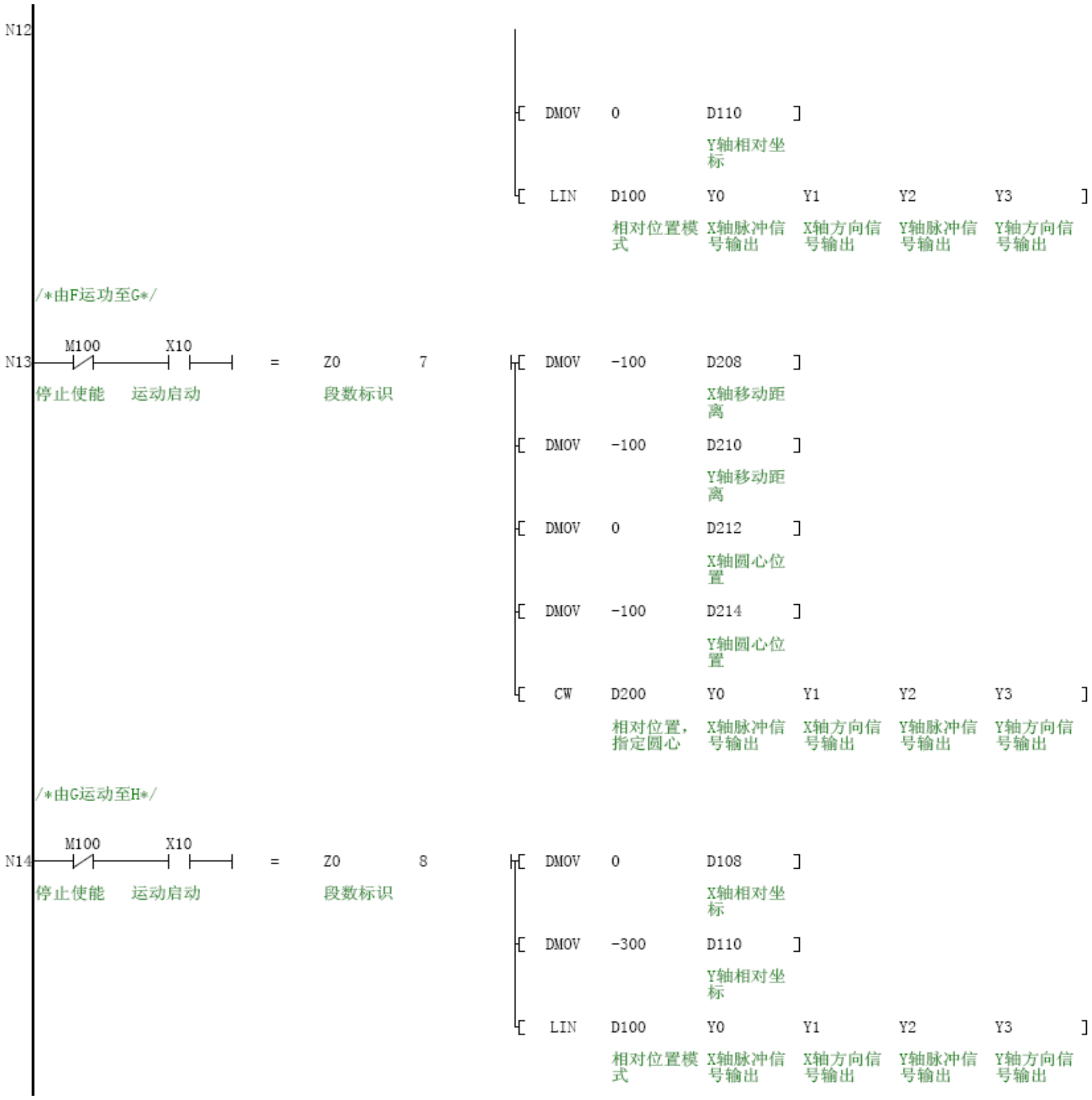
```
/*初始化运动控制指令相关软元件*/

N0   SM1
  ---| |---[ DMOV   0        D100   ]
                             相对位置模式
           [ DMOV   10000    D102   ]
                             直线插补初始速度
           [ DMOV   10000    D104   ]
                             直线插补合成速度
           [ DMOV   100      D106   ]
                             直线插补减速度时间
           [ DMOV   0        D108   ]
                             X轴相对坐标
           [ DMOV   0        D110   ]
                             Y轴相对坐标
N1   SM1
  ---| |---[ DMOV   0        D200   ]
                             相对位置，指定圆心
           [ DMOV   10000    D202   ]
                             圆弧插补保留
           [ DMOV   10000    D204   ]
                             圆弧插补合成速度
           [ DMOV   100      D206   ]
                             保留
           [ DMOV   0        D208   ]
                             X轴移动距离
           [ DMOV   0        D210   ]
                             Y轴移动距离
           [ DMOV   0        D212   ]
                             X轴圆心位置

           [ DMOV   0        D214   ]
                             Y轴圆心位置
           [ MOV    1        Z0     ]
                             段数标识
           [ SET    SM72   ]
                    Y2脉冲输出完成中断
           [ SET    SM69   ]
                    中断使能控制
           [ EI   ]
N2   SM0
  ---| |---[ DMOV   200000   SD202  ]
                             Y0最高速度
           [ MOV    10       SD204  ]
                             Y0基底速度
           [ MOV    100      SD205  ]
                             Y0加减速时间
           [ MOV    10       SD207  ]
                             Y0爬行速度
           [ DMOV   2000     SD208  ]
                             Y0回归速度
N3   SM0
  ---| |---[ DMOV   200000   SD322  ]
                             Y2最高速度
           [ MOV    10       SD324  ]
                             Y2基底速度
           [ MOV    100      SD325  ]
                             Y2加减速时间
           [ MOV    10       SD327  ]
                             Y2爬行速度
           [ DMOV   2000     SD328  ]
                             Y2回归速度
```



```
/*X轴回零*/
N4   X2 (X轴回零按钮)  [DSZR  X0  X0  Y0  Y1]
     X0 X轴近点信号   X0 X轴近点信号   Y0 X轴脉冲信号输出   Y1 X轴方向信号输出

/*Y轴回零*/
N5   X3 (Y轴回零按钮)  [DSZR  X1  X1  Y2  Y3]
     X1 Y轴近点信号   X1 Y轴近点信号   Y2 Y轴脉冲信号输出   Y3 Y轴方向信号输出

/*停止操作*/
N6   X4 (停止按钮)  (M100) 停止使能

/*启动*/

/*由原点运动至起点A*/
N7   M100/ (停止使能)  X10 (运动启动)  [=  Z0 (段数标识)  1]  [DRVI  -200  10000  Y2  Y3]
     Y2 Y轴脉冲信号输出   Y3 Y轴方向信号输出

/*A运动至B*/
N8   M100/ (停止使能)  X10 (运动启动)  [=  Z0 (段数标识)  2]
        [DMOV  600  D108]   D108 X轴相对坐标
        [DMOV  0    D110]   D110 Y轴相对坐标
        [LIN   D100  Y0  Y1  Y2  Y3]
     D100 相对位置模式   Y0 X轴脉冲信号输出   Y1 X轴方向信号输出   Y2 Y轴脉冲信号输出   Y3 Y轴方向信号输出

/*由B运动至C*/
N9   M100/ (停止使能)  X10 (运动启动)  [=  Z0 (段数标识)  3]
        [DMOV  100  D208]   D208 X轴移动距离
        [DMOV  100  D210]   D210 Y轴移动距离
```



```
                                          [ DMOV   0          D212       ]
                                                              X轴圆心位置
                                          [ DMOV   100        D214       ]
                                                              Y轴圆心位置
                                          [ CW     D200       Y0         Y1         Y2         Y3         ]
                                                   相对位置，  X轴脉冲信   X轴方向信   Y轴脉冲信   Y轴方向信
                                                   指定圆心    号输出      号输出      号输出      号输出
/*由C运动至D*/
     M100      X10
N10──┤/├───────┤ ├──┤   =    Z0     4   [ DMOV   0          D108       ]
停止使能   运动启动          段数标识                          X轴相对坐标
                                          [ DMOV   300        D110       ]
                                                              Y轴相对坐标
                                          [ LIN    D100       Y0         Y1         Y2         Y3         ]
                                                   相对位置模  X轴脉冲信   X轴方向信   Y轴脉冲信   Y轴方向信
                                                   式          号输出      号输出      号输出      号输出
/*由D运动至E*/
     M100      X10
N11──┤/├───────┤ ├──┤   =    Z0     5   [ DMOV   -100       D208       ]
停止使能   运动启动          段数标识                          X轴移动距离
                                          [ DMOV   100        D210       ]
                                                              Y轴移动距离
                                          [ DMOV   -100       D212       ]
                                                              X轴圆心位置
                                          [ DMOV   0          D214       ]
                                                              Y轴圆心位置
                                          [ CW     D200       Y0         Y1         Y2         Y3         ]
                                                   相对位置，  X轴脉冲信   X轴方向信   Y轴脉冲信   Y轴方向信
                                                   指定圆心    号输出      号输出      号输出      号输出
/*由E运动至F*/
     M100      X10
N12──┤/├───────┤ ├──┤   =    Z0     6   [ DMOV   -600       D108       ]
停止使能   运动启动          段数标识                          X轴相对坐标
```



```
N12
                                      [ DMOV   0          D110       ]
                                                          Y轴相对坐标
                                      [ LIN    D100       Y0         Y1         Y2         Y3        ]
                                               相对位置模式 X轴脉冲信号输出 X轴方向信号输出 Y轴脉冲信号输出 Y轴方向信号输出
/*由F运功至G*/
     M100      X10
N13──┤/├───────┤ ├──┤ =    Z0      7   ┤[ DMOV   -100       D208       ]
  停止使能   运动启动       段数标识                       X轴移动距离
                                      [ DMOV   -100       D210       ]
                                                          Y轴移动距离
                                      [ DMOV   0          D212       ]
                                                          X轴圆心位置
                                      [ DMOV   -100       D214       ]
                                                          Y轴圆心位置
                                      [ CW     D200       Y0         Y1         Y2         Y3        ]
                                               相对位置，指定圆心 X轴脉冲信号输出 X轴方向信号输出 Y轴脉冲信号输出 Y轴方向信号输出
/*由G运动至H*/
     M100      X10
N14──┤/├───────┤ ├──┤ =    Z0      8   ┤[ DMOV   0          D108       ]
  停止使能   运动启动       段数标识                       X轴相对坐标
                                      [ DMOV   -300       D110       ]
                                                          Y轴相对坐标
                                      [ LIN    D100       Y0         Y1         Y2         Y3        ]
                                               相对位置模式 X轴脉冲信号输出 X轴方向信号输出 Y轴脉冲信号输出 Y轴方向信号输出
```

Appendix 1 Special auxiliary relay

All the special auxiliary relays are initialized when the PLC changes from STOP to RUN. Those that have been set in system setting will be set to the preset value after that initialization.

Note

The reserved SD and SM elements are not listed in the table. The reserved SM elements are by default read only (R).

1. PLC work state flag

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM0	Monitoring run bit	This bit is high in the RUN state, and zero in the STOP state	R	√	√	√	√
SM1	Initial run pulse bit	This bit is set high when PLC changes from STOP to RUN, and set low after a scan cycle	R	√	√	√	√
SM2	Power on flag bit	This bit is set high after system power-on, and set low after a scan cycle	R	√	√	√	√
SM3	System error	This bit is set when system error occurs after power-on or after PLC changes from STOP to RUN, or reset if no system error occurs	R	√	√	√	√
SM4	Battery voltage low	This bit is set when the battery voltage is too low, or reset if the battery voltage is detected higher than 2.4V	R	√		√	√
SM5	AC power failure detection bit	This bit is set when PLC detects AC power off (detecting time 40ms). If the power is on after the delay of power off detecting time (set in SD05), the bit will be reset	R	√	√	√	√
SM6	24Vdc power failure	This bit is set when PLC detects the 24Vdc power failure (detecting time 50ms). If within the following 50ms the power is detected to be back, this bit will be reset	R	√	√	√	√
SM7	No battery work mode	If this bit is set as 1, the battery backup data lost error and the forced-table lost error (code: 44) will not be reported upon system battery failure (configurable only through system block)	R	√		√	√
SM8	Constant scan mode	Set this bit, and the scan time will be constant (configurable only through system block)	R	√	√	√	√
SM9	Input point startup mode	Set this bit, and the PLC can change from STOP to RUN when the designated X input point is ON (configurable only through system block)	R	√	√	√	√

2. Clock running bit

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM10	10ms clock	Crystal oscillation (period: 10ms). Reverse every half period. The first half period is 0 when the user program starts	R	√	√	√	√
SM11	100ms clock	Crystal oscillation (period: 100ms). Reverse every half period. The first half period is 0 when the user program starts	R	√	√	√	√
SM12	1s clock	Crystal oscillation (period: 1s). Reverse every half period. The first half period is 0 when the user program starts	R	√	√	√	√
SM13	1min clock	Crystal oscillation (period: 1min). Reverse every half period. The first half period is 0 when the user program starts	R	√	√	√	√
SM14	1hour clock	Crystal oscillation (period: 1 hour). Reverse every half period. The first half period is 0 when the user program starts	R	√	√	√	√
SM15	Scan cycle oscillation bit	This bit reverses once every scan cycle (The first period is 0 when the user program starts)	R	√	√	√	√
SM16	High-speed ring counter enabling flag	Unit: 0.1ms, 16 bits Set: high-speed ring counter starts counting Reset: high-speed ring counter stops counting	R/W		√		√

3. User program execution error

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM20	Instruction execution error	This bit is set upon instruction execution error. Ths corresponding error type code is written into SD20. This bit is cleared after the execution succeeds	R	√	√	√	√
SM21	Instruction register number subscript overflow	This bit is set upon instruction execution error. The corresponding error type code is written into SD20	R	√	√	√	√
SM22	Instruction parameter illegal	This bit is set upon instruction execution error. The corresponding error type code is written into SD20. This bit is cleared after the execution succeeds	R	√	√	√	√
SM30	Instruction execution end flag	This bit is connected at the end of MODRW instruction execution	R		√		√

4. Interrupt control

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM40	X0 input rising/falling edge interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM41	X1 input rising/falling edge interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM42	X2 input rising/falling edge interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM43	X3 input rising/falling edge interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM44	X4 input rising/falling edge interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM45	X5 input rising/falling edge interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM46	X6 input rising/falling edge interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM47	X7 input rising/falling edge interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM48	COM 0 character transmission interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM49	COM 0 character reception interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM50	COM 0 frame transmission interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM51	COM 0 frame reception interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM52	COM 1 character transmission interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM53	COM 1 character reception interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM54	COM 1 frame transmission interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM55	COM 1 frame reception interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM56	Power failure interrupt	Enable when set as 1	R/W	√	√	√	√
SM57	COM 2 character transmission interrupt enabling flag bit	Enable when set as 1	R/W	√		√	√
SM58	COM 2 character reception interrupt enabling flag bit	Enable when set as 1	R/W	√		√	√
SM59	COM 2 frame transmission interrupt enabling flag bit	Enable when set as 1	R/W	√		√	√
SM60	COM 2 frame reception interrupt enabling flag bit	Enable when set as 1	R/W	√		√	√
SM61	Positioning instruction passed position interrupt 1 enabling flag bit	Enable when set as 1	R/W				
SM62	Positioning instruction passed position interrupt 2 enabling flag bit	Enable when set as 1	R/W				
SM63	PTO (Y0) output finish interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM64	PTO (Y1) output finish interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM65	High-speed counter interrupt enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM66	Timed interrupt 0 enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM67	Timed interrupt 1 enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM68	Timed interrupt 2 enabling flag bit	Enable when set as 1	R/W	√	√	√	√
SM69	Interpolation complete interrupt 1 enabling flag bit	Enable when set as 1	R/W			√	
SM72	High-speed output 2 complete interrupt enabling flag bit	Enable when set as 1	R/W			√	√
SM73	High-speed output 3 complete interrupt enabling flag bit	Enable when set as 1	R/W				√
SM74	High-speed output 4 complete interrupt enabling flag bit	Enable when set as 1	R/W			√	
SM75	High-speed output 5 complete interrupt enabling flag bit	Enable when set as 1	R/W			√	
SM76	High-speed output 6 complete interrupt enabling flag bit	Enable when set as 1	R/W			√	
SM77	High-speed output 7 complete interrupt enabling flag bit	Enable when set as 1	R/W			√	
SM78	Interpolation complete interrupt 2 enabling flag bit	Enable when set as 1	R/W			√	

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM106	Positioning instruction passed position interrupt 4 enabling flag bit	Enable when set as 1	R/W			√	
SM107	Positioning instruction passed position interrupt 5 enabling flag bit	Enable when set as 1	R/W			√	
SM108	Positioning instruction passed position interrupt 6 enabling flag bit	Enable when set as 1	R/W			√	

5. External instruction

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM70	Print mode selection	1-16 characters when set as 1, 8 characters when set as 0	R/W	√		√	
SM71	Print in progress	In printing when set as 1	R	√		√	

6. High-speed pulse output control

Addr.	Name	Function	R/W	EC20	EC10	EC20H	EC10V
SM80	Y0 high-speed pulse output control	Y0 high-speed pulse output stop instruction	R/W	√	√	√	√
SM81	Y1 high-speed pulse output control	Y1 high-speed pulse output stop instruction	R/W	√	√		√
SM82	Y0 high-speed pulse output monitor	Y0 high-speed pulse output mointor (ON: busy. OFF: ready)	R	√	√	√	√
SM83	Y1 high-speed pulse output monitor	Y1 high-speed pulse output mointor (ON: busy. OFF: ready)	R	√	√	√	√
SM84	PWM time base unit	ON: microsecond. OFF: millisecond	R/W		√		√
SM85	Reset function valid	Output of CLR signal for ZRN instruction enabled	R/W		√		
SM86	Y0 interrupt drive pulse output valid	ON: PLSY instructions can be called in interrupt programs and subprograms, and driven continuously and repeatedly with power flow in main programs	R/W		√	√	√
SM87	Y1 interrupt drive pulse output valid	ON: PLSY instructions can be called in interrupt programs and subprograms, and driven continuously and repeatedly with power flow in main programs	R/W		√	√	√
SM88	Envelope loop execution	ON: envelope loop execution	R/W			√	
SM89	PLSV gradual frequency conversion	ON: frequency changes gradually	R/W		√	√	√
SM382	Y2 interrupt drive pulse output valid	ON: PLSY instructions can be called in interrupt programs and subprograms, and driven continuously and repeatedly with power flow in main programs	R/W				√
SM383	Y3 interrupt drive pulse output valid	ON: PLSY instructions can be called in interrupt programs and subprograms, and driven continuously and repeatedly with power flow in main programs	R/W				√

7. Pulse capture monitoring bit

Addr.	Name	Function	R/W	EC20	EC10	EC20H	EC10V
SM90	Input X0 pulse capture monitoring bit	Capture rising edge pulse at input X0	R/W	√	√		√
SM91	Input X1 pulse capture monitoring bit	Capture rising edge pulse at input X1	R/W	√	√		√
SM92	Input X2 pulse capture monitoring bit	Capture rising edge pulse at input X2	R/W	√	√		√
SM93	Input X3 pulse capture monitoring bit	Capture rising edge pulse at input X3	R/W	√	√		√
SM94	Input X4 pulse capture monitoring bit	Capture rising edge pulse at input X4	R/W	√	√		√
SM95	Input X5 pulse capture monitoring bit	Capture rising edge pulse at input X5	R/W	√	√		√
SM96	Input X6 pulse capture monitoring bit	Capture rising edge pulse at input X6	R/W	√	√		√
SM97	Input X7 pulse capture monitoring bit	Capture rising edge pulse at input X7	R/W	√	√		√
Note: 1. All the elements in this table are cleared when the PLC changes from STOP to RUN. The pulse capture will fail when the HCNT or SPD instruction is being executed at the same input point. For details, see ***Error! Reference source not found.Error! Reference source not***							

found. and ***Error! Reference source not found.Error! Reference source not found.***.

2. For hardware counters, the total pulse frequency input through X0~X7 (using pulse capture, SPD instruction or HCNT instructions, but not high-speed compare instructions) is ≤80k. For software counters, that frequency (using instructions DHSCS, DHSCI, DHSZ, DHSP or DHST for driven high-speed counters) is ≤30k.

8.Fourfold frequency

Addr.	Name	function	R/W	EC20	EC10	EC20H	EC10V
SM100	AB ph of X0 and X1 input once/fourfold switching	Clear when STOP—>RUN	R/W	√	√	√	√
SM101	AB ph of X2 and X3 input once/fourfold switching		R/W			√	
SM102	AB ph of X3 and X4 input once/fourfold switching		R/W	√	√	√	√
SM103	AB ph of X4 and X5 input once/fourfold switching		R/W			√	
SM104	AB ph of X6 and X7 input once/fourfold switching		R/W			√	

9.Free port (port 0)

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM110	Port 0 transmission enabling flag bit	This bit is set when XMT instruction is used, and is cleared after the transmission is over. You can manually clear this bit to halt the current transmission at Port 0. The transmission can resume when power flow is on again	R/W	√	√	√	√
SM111	Port 0 reception enabling flag bit	This bit is set when RCV instruction is used, and is cleared after the transmission is over. You can manually clear this bit to halt the current transmission at Port 0. The transmission can resume when power flow is on again	R/W	√	√	√	√
SM112	Port 0 transmission complete flag bit	This bit is set after the transmission is over	R/W	√	√	√	√
SM113	Port 0 reception complete flag bit	This bit is set after the reception is over	R/W	√	√	√	√
SM114	Port 0 idle flag bit	This bit is set when the port is idle	R	√	√	√	√

Note

SM112~SM114 are the flags for the reception, complete and idle states in all communication protocols that are supported by PORT 0. For example, the PORT 0 of EC10 PLC supports N:N bus, Modbus and Freeport. No matter which protocol is used, the functions of SM112~SM114 remain the same.

10. Free port (port 1)

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM120	Port 1 transmission enabling flag bit	This bit is set when XMT instruction is used, and is cleared after the transmission is over. You can manually clear this bit to halt the current transmission at Port 1. The transmission can resume when power flow is on again	R/W	√	√	√	√
SM121	Port 1 reception enabling flag bit	This bit is set when RCV instruction is used, and is cleared after the transmission is over. You can manually clear this bit to halt the current transmission at Port 1. The transmission can resume when power flow is on again	R/W	√	√	√	√
SM122	Port 1 transmission	This bit is set after the transmission is over	R/W	√	√	√	√

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
	complete flag bit						
SM123	Port 1 reception complete flag bit	This bit is set after the reception is over	R/W	√	√	√	√
SM124	Port 1 idle flag bit	This bit is set when the port is idle	R	√	√	√	√

📖 **Note**

SM122~SM124 are the flags for the reception, complete and idle states in all communication protocols that are supported by Port 1 . For example, the Port 1 of EC10 PLC supports N:N bus, Modbus and Freeport. No matter which protocol is used, the functions of SM122~SM124 remain the same.

11. Extension free port (port 2)

Addr.	Name	Action and function	R/W	EC20	EC20H	EC10V
SM130	Port 2 transmission enabling flag bit	This bit is set when XMT instruction is used, and is cleared after the transmission is over. You can manually clear this bit to halt the current transmission at Port 2. The transmission can resume when power flow is on again	R/W	√	√	√
SM131	Port 2 reception enabling flag bit	This bit is set when RCV instruction is used, and is cleared after the transmission is over. You can manually clear this bit to halt the current transmission at Port 2. The transmission can resume when power flow is on again	R/W	√	√	√
SM132	Port 2 transmission complete flag bit	This bit is set after the transmission is over	R/W	√	√	√
SM133	Port 2 reception complete flag bit	This bit is set after the reception is over	R/W	√	√	√
SM134	Port 2 idle flag bit	This bit is set when the port is idle	R	√	√	√

Note

SM132~SM134 are the flags for the reception, complete and idle states in all communication protocols that are supported by Port 2 . For example, the Port 2 of EC10 PLC supports N:N bus, Modbus and Freeport. No matter which protocol is used, the functions of SM132~SM134 remain the same.

12. Modbus communication

Addr.	Name	Action and function	R/W	EC20	EC10	EC10V	EC20H
SM135	Port 1 Modbus communication complete	This bit is set after the communication is over	R/W	√	√	√	√
SM136	Port 1 Modbus communication error	This bit is set upon communication error	R/W	√	√	√	√
SM137	Port 2 Modbus communication complete	This bit is set after the communication is over	R/W	√		√	√
SM138	Port 2 Modbus communication error	This bit is set upon communication error	R/W	√		√	√

13. N:N bus communication

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM140	Station 0 communication error flag		R	√	√	√	√
SM141	Station 1 communication error flag		R	√	√	√	√
SM142	Station 2 communication error flag		R	√	√	√	√
SM143	Station 3 communication error flag		R	√	√	√	√
SM144	Station 4 communication error flag		R	√	√	√	√

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM145	Station 5 communication error flag		R	√	√	√	√
SM146	Station 6 communication error flag		R	√	√	√	√
SM147	Station 7 communication error flag		R	√	√	√	√
SM148	Station 8 communication error flag		R	√	√	√	√
SM149	Station 9 communication error flag		R	√	√	√	√
SM150	Station 10 communication error flag		R	√	√	√	√
SM151	Station 11 communication error flag		R	√	√	√	√
SM152	Station 12 communication error flag		R	√	√	√	√
SM153	Station 13 communication error flag		R	√	√	√	√
SM154	Station 14 communication error flag		R	√	√	√	√
SM155	Station 15 communication error flag		R	√	√	√	√
SM156	Station 16 communication error flag		R	√	√	√	√
SM157	Station 17 communication error flag		R	√	√	√	√
SM158	Station 18 communication error flag		R	√	√	√	√
SM159	Station 19 communication error flag		R	√	√	√	√
SM160	Station 20 communication error flag		R	√	√	√	√
SM161	Station 21 communication error flag		R	√	√	√	√
SM162	Station 22 communication error flag		R	√	√	√	√
SM163	Station 23 communication error flag		R	√	√	√	√
SM164	Station 24 communication error flag		R	√	√	√	√
SM165	Station 25 communication error flag		R	√	√	√	√
SM166	Station 26 communication error flag		R	√	√	√	√
SM167	Station 27 communication error flag		R	√	√	√	√
SM168	Station 28 communication error flag		R	√	√	√	√
SM169	Station 29 communication error flag		R	√	√	√	√
SM170	Station 30 communication error flag		R	√	√	√	√
SM171	Station 31 communication error flag		R	√	√	√	√

14. Enabling flag of integrated analog channel

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM172	Enabling flag of AD channel 1	Sampling at AD channel 1 is enabled when this bit is set to 1	R/W		√		√
SM173	Enabling flag of AD channel 2	Sampling at AD channel 2 is enabled when this bit is set to 1	R/W		√		√
SM174	Voltage/current enabling flag of AD channel 1	1 for current input and 0 for voltage input	R/W		√		√
SM175	Voltage/current enabling flag of AD channel 2	1 for current input and 0 for voltage input	R/W		√		√
SM176	Enabling flag of AD channel 3	Sampling at AD channel 3 is enabled when this bit is set to 1	R/W		√		√
SM177	Voltage/current enabling flag of AD channel 3	1 for current input and 0 for voltage input	R/W		√		√
SM34	Enabling flag of AD channel 4	Sampling at AD channel 4 is enabled when this bit is set to 1	R/W		√		√
SM35	Voltage/current enabling flag of AD channel 4	1 for current input and 0 for voltage input	R/W		√		√
SM36	Enabling flag of AD channel 5	Sampling at AD channel 5 is enabled when this bit is set to 1	R/W		√		√
SM37	Voltage/current enabling flag of AD channel 5	1 for current input and 0 for voltage input	R/W		√		√
SM38	Enabling flag of AD channel 6	Sampling at AD channel 6 is enabled when this bit is set to 1	R/W		√		√

SM39	Voltage/current enabling flag of AD channel 6	1 for current input and 0 for voltage input	R/W		√		√
SM178	Enabling flag of DA channel 0	Output at DA channel 0 is enabled when this bit is set to 1	R/W		√		√

15. Operation flag bit

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM180	Zero flag bit	This bit is set when the related calculation result is zero. You can clear or set this bit manually	R/W	√	√	√	√
SM181	Carry/overflow flag bit	This bit is set when the result of the related calculation is a carry. You can clear or set this bit manually	R/W	√	√	√	√
SM182	Borrow flag bit	This bit is set when the result of the related calculation is a borrow. You can clear or set this bit manually	R/W	√	√	√	√
SM185	Table comparison flag	This bit is set when the whole table is completed	R/W	√	√	√	√
SM188	Data block comparison set	This bit is set when the data block comparison result is 1	R/W			√	

16. ASCII code conversion instruction flag

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM186	ASC instruction storing mode flag	0: the most and least significant bytes of every word are stored with one ASCII code 1: the least significant byte of every word is stored with one ASCII code	R/W	√	√	√	√
SM187	Instruction execution completion	Set ON after MTR initial circulation	R				

17. System bus error flag

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM190	Main module bus error flag bit	1. Clear when power on correct addressing 2. Clear when no STOP→RUN error 3. Clear when downloading new programs 4. The bit causes stop of the system	R	√	√	√	√
SM191	General module bus error flag bit	1. This bit is set and the system raises an alarm when a general module bus operation error occurs 2. This bit is reset automatically when the system error is removed	R	√	√	√	√
SM192	Special module bus error flag bit	1. This bit is set and the system raises an alarm when a special module bus operation error occurs 2. This bit is reset automatically when the system error is removed	R	√	√	√	√

18. Real-time clock error flag

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM193	R/W of real-time clock error	This bit is set upon real-time clock error This bit is automatically cleared if system fault is removed	R	√	√	√	√

19. Memory card existing flag

Addr.	Name	Action and function	R/W	EC20	EC10	EC10V	EC20H
SM197	Memory card is connected or not	This bit is set as 1 when memory card connected and reset when no memory card	R			√	√

20. Counting direction of bi-directional counters

Addr.	Counter addr.	Function	R/W	EC20	EC10	EC20H	EC10V
SM200	C200	When SM2 _ _ is of high level, the corresponding C2_ _ becomes down counter When SM2 _ _ is of low level, the corresponding C2_ _ becomes up counter	R/W	√	√	√	√
SM201	C201		R/W	√	√	√	√
SM202	C202		R/W	√	√	√	√
SM203	C203		R/W	√	√	√	√
SM204	C204		R/W	√	√	√	√
SM205	C205		R/W	√	√	√	√
SM206	C206		R/W	√	√	√	√
SM207	C207		R/W	√	√	√	√
SM208	C208		R/W	√	√	√	√
SM209	C209		R/W	√	√	√	√
SM210	C210		R/W	√	√	√	√
SM211	C211		R/W	√	√	√	√
SM212	C212		R/W	√	√	√	√
SM213	C213		R/W	√	√	√	√
SM214	C214		R/W	√	√	√	√
SM215	C215		R/W	√	√	√	√
SM216	C216		R/W	√	√	√	√
SM217	C217		R/W	√	√	√	√
SM218	C218		R/W	√	√	√	√
SM219	C219		R/W	√	√	√	√
SM220	C220		R/W	√	√	√	√
SM221	C221		R/W	√	√	√	√
SM222	C222		R/W	√	√	√	√
SM223	C223		R/W	√	√	√	√
SM224	C224		R/W	√	√	√	√
SM225	C225		R/W	√	√	√	√
SM226	C226		R/W	√	√	√	√
SM227	C227		R/W	√	√	√	√
SM228	C228		R/W	√	√	√	√
SM229	C229		R/W	√	√	√	√
SM230	C230		R/W	√	√	√	√
SM231	C231		R/W	√	√	√	√
SM232	C232		R/W	√	√	√	√
SM233	C233		R/W	√	√	√	√
SM234	C234		R/W	√	√	√	√
SM235	C235		R/W	√	√	√	√

21. Counting direction and monitoring of high-speed counter

Type	Addr.	Name	Register content	R/W	EC20	EC10	EC20H	EC10V
Single phase single point counting input	SM236	C236	The high&low level of SM2 _ _ corresponds to the counting down&up of the counter respectively	R/W	√	√	√	√
	SM237	C237		R/W	√	√	√	√
	SM238	C238		R/W	√	√	√	√
	SM239	C239		R/W	√	√	√	√
	SM240	C240		R/W	√	√	√	√
	SM241	C241		R/W	√	√	√	√
	SM301	C301		R/W			√	

Type	Addr.	Name	Register content	R/W	EC20	EC10	EC20H	EC10V
	SM302	C302		R/W			√	
	SM242	C242		R/W	√	√	√	√
	SM243	C243		R/W	√	√	√	√
	SM244	C244		R/W	√	√	√	√
Single phase bi-directional counting input	SM245	C245	When the single phase bi-directional counter and 2-phase counter C2 _ _ is in the down counting mode, the corresponding SM2 _ _ becomes high level; when in up counting mode, the corresponding SM2 _ _ becomes low level	R/W	√	√	√	√
	SM246	C246		R/W	√	√	√	√
	SM247	C247		R/W	√	√	√	√
	SM303	C303		R/W			√	
	SM248	C248		R/W	√	√	√	√
	SM249	C249		R/W	√	√	√	√
AB phase counting input	SM250	C250		R/W	√	√	√	√
	SM251	C251		R/W	√	√	√	√
	SM304	C304		R/W			√	
	SM305	C305		R/W			√	
	SM306	C306		R/W			√	
	SM252	C252		R/W	√	√	√	√
	SM253	C253		R/W	√	√	√	√
	SM254	C254		R/W	√	√	√	√
	SM255	C255		R/W	√	√	√	√

22. Enhanced positioning

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM260	Interrupt input function designation valid	Y0 and Y1 are applicable to DVIT. When the function is unused, Y0 will correspond to X0 interrupt and Y1 to X1 interrupt. When this bit is set, each 4bit of SD240 will correspond to input of each output (Y)	R/W	√			√
SM262	Y002 pulse output stop instruction	After this bit is set, Y002 pulse will be disabled	R/W			√	√
SM262	Y003 pulse output stop instruction	After this bit is set, Y003 pulse will be disabled	R/W				√
SM264	Y004 pulse output stop instruction	After this bit is set, Y004 pulse will be disabled	R/W			√	
SM265	Y005 pulse output stop instruction	After this bit is set, Y005 pulse will be disabled	R/W			√	
SM266	Y006 pulse output stop instruction	After this bit is set, Y006 pulse will be disabled	R/W			√	
SM267	Y007 pulse output stop instruction	After this bit is set, Y007 pulse will be disabled	R/W			√	
SM272	Y002 pulse output monitor (busy/ready)	Setting at Y002 pulse output	R			√	√
SM272	Y003 pulse output monitor (busy/ready)	Setting at Y003 pulse output	R				√
SM274	Y004 pulse output monitor (busy/ready)	Setting at Y004 pulse output	R			√	
SM275	Y005 pulse output monitor (busy/ready)	Setting at Y005 pulse output	R			√	
SM276	Y006 pulse output monitor (busy/ready)	Setting at Y006 pulse output	R			√	
SM277	Y007 pulse output monitor (busy/ready)	Setting at Y007 pulse output	R			√	
SM280	Clear function valid	Applicable to DSZR/ZRN, CLR signal output function valid (Y0) when positioning instruction origin return (ZRN)	R/W	√		√	√
SM281	Clear signal designated element valid	Applicable to DSZR, the corresponding Y(N) in SD230 is clear signal; Y10 is clear signal of Y0 when no designation	R/W	√		√	√
SM282	Origin return direction	Y0 is applicable to DSZR	R/W	√		√	√

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM283	Forward rotation limit	Y0 is applicable to DSZR/DVIT	R/W	√		√	√
SM284	Reverse rotation limit	Y0 is applicable to DSZR/DVIT	R/W	√		√	√
SM285	Logic reverse rotation of near-point signal	Y0 is applicable to DSZR	R/W	√		√	√
SM286	Logic reverse rotation of zero-point signal	Y0 is applicable to DSZR	R/W	√		√	√
SM287	Logic reverse rotation of interrupt signal	Y0 is applicable to DVIT, not applicable to user interrupt input instruction	R/W	√		√	√
SM288	Positioning instruction in drive	Y0 is applicable to DSZR/DVIT	R/W	√		√	√
SM289	User interrupt input instruction	Y0 is applicable to DVIT	R/W	√			√
SM290	Clear function valid	Applicable to DSZR/ZRN, CLR signal output function valid (Y1) when positioning instruction origin return (ZRN)	R/W	√			√
SM291	Clear signal designated element valid	Applicable to DSZR, the corresponding Y(N) in SD230 is clear signal; Y11 is clear signal of Y1 when no designation	R/W	√			√
SM292	Origin return direction	Y1 is applicable to DSZR	R/W	√			√
SM293	Forward rotation limit	Y1 is applicable to DSZR/DVIT	R/W	√			√
SM294	Reverse rotation limit	Y1 is applicable to DSZR/DVIT	R/W	√			√
SM295	Logic reverse rotation of near-point signal	Y1 is applicable to DSZR	R/W	√			√
SM296	Logic reverse rotation of zero-point signal	Y1 is applicable to DSZR	R/W	√			√
SM297	Logic reverse rotation of interrupt signal	Y1 is applicable to DVIT	R/W	√			√
SM298	Positioning instruction in drive	Y1 is applicable to DSZR/DVIT	R/W	√			√
SM299	User interrupt input instruction	Y1 is applicable to DVIT	R/W				√
SM320	Clear function valid	Y2 is applicable to DSZR/ZRN	R/W			√	√
SM321	Clear signal designated element valid	Applicable to DSZR, the corresponding Y(N) in SD230 is clear signal; Y2 is Y12 when no designation (EC20 is Y3)	R/W			√	√
SM322	Origin return direction	Y2 is applicable to DSZR	R/W			√	√
SM323	Forward rotation limit	Y2 is applicable to DSZR/DVIT	R/W			√	√
SM324	Reverse rotation limit	Y2 is applicable to DSZR/DVIT	R/W			√	√
SM325	Logic reverse rotation of near-point signal	Y2 is applicable to DSZR	R/W			√	√
SM326	Logic reverse rotation of zero-point signal	Y2 is applicable to DSZR	R/W			√	√
SM327	Logic reverse rotation of interrupt signal	Y2 is applicable to DVIT	R/W				√
SM328	Positioning instruction in drive	Y2 is applicable to DSZR/DVIT	R/W			√	√
SM329	User interrupt input instruction	Y2 is applicable to DVIT	R/W				√
SM330	Clear function valid	Y3 is applicable to DSZR/ZRN	R/W				√
SM331	Clear signal designated element valid	Applicable to DSZR, the corresponding Y(N) in SD230 is clear signal; Y2 is Y12 when no designation (EC20 is Y3)	R/W				√
SM332	Origin return direction	Y3 is applicable to DSZR	R/W				√
SM333	Forward rotation limit	Y3 is applicable to DSZR/DVIT	R/W				√
SM334	Reverse rotation limit	Y3 is applicable to DSZR/DVIT	R/W				√
SM335	Logic reverse rotation of near-point signal	Y3 is applicable to DSZR	R/W				√
SM336	Logic reverse rotation of zero-point signal	Y3 is applicable to DSZR	R/W				√

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM337	Logic reverse rotation of interrupt signal	Y3 is applicable to DVIT	R/W				√
SM338	Positioning instruction in drive	Y3 is applicable to DSZR/DVIT	R/W				
SM339	User interrupt input instruction	Y3 is applicable to DVIT	R/W				√
SM341	Clear signal designated element valid	Applicable to DSZR, the corresponding Y(N) in SD230 is clear signal; Y4 is Y14 when no designation	R/W			√	
SM342	Origin return direction	Y4 is applicable to DSZR	R/W			√	
SM343	Forward rotation limit	Y4 is applicable to DSZR/DVIT	R/W			√	
SM344	Reverse rotation limit	Y4 is applicable to DSZR/DVIT	R/W			√	
SM345	Logic reverse rotation of near-point signal	Y4 is applicable to DSZR	R/W			√	
SM346	Logic reverse rotation of zero-point signal	Y4 is applicable to DSZR	R/W			√	
SM347	Logic reverse rotation of interrupt signal	Y4 is applicable to DVIT	R/W			√	
SM348	Positioning instruction in drive	Y4 is applicable to DSZR/DVIT	R/W			√	
SM350	Clear function valid	Y5 is applicable to DSZR/ZRN	R/W			√	
SM351	Clear signal designated element valid	Applicable to DSZR, the corresponding Y(N) in SD is clear signal; Y5 is Y15 when no designation	R/W			√	
SM352	Origin return direction	Y5 is applicable to DSZR	R/W			√	
SM353	Forward rotation limit	Y5 is applicable to DSZR/DVIT	R/W			√	
SM354	Reverse rotation limit	Y5 is applicable to DSZR/DVIT	R/W			√	
SM355	Logic reverse rotation of near-point signal	Y5 is applicable to DSZR	R/W			√	
SM356	Logic reverse rotation of zero-point signal	Y5 is applicable to DSZR	R/W			√	
SM357	Logic reverse rotation of interrupt signal	Y5 is applicable to DVIT	R/W				
SM358	Positioning instruction in drive	Y5 is applicable to DSZR/DVIT	R/W			√	
SM360	Clear function valid	Y6 is applicable to DSZR/ZRN	R/W			√	
SM361	Clear signal designated element valid	Applicable to DSZR, the corresponding Y(N) in SD230 is clear signal; Y6 is Y16 when no designation	R/W			√	
SM362	Origin return direction	Y6 is applicable to DSZR	R/W			√	
SM363	Forward rotation limit	Y6 is applicable to DSZR/DVIT	R/W			√	
SM364	Reverse rotation limit	Y6 is applicable to DSZR/DVIT	R/W			√	
SM365	Logic reverse rotation of near-point signal	Y6 is applicable to DSZR	R/W			√	
SM366	Logic reverse rotation of zero-point signal	Y6 is applicable to DSZR	R/W			√	
SM370	Clear function valid	Y7 is applicable to DSZR/ZRN	R/W			√	
SM371	Clear signal designated element valid	Applicable to DSZR, the corresponding Y(N) in SD230 is clear signal; Y7 is Y17 when no designation (EC20 is Y3)	R/W			√	
SM372	Origin return direction	Y7 is applicable to DSZR	R/W			√	
SM373	Forward rotation limit	Y7 is applicable to DSZR/DVIT	R/W			√	
SM374	Reverse rotation limit	Y7 is applicable to DSZR/DVIT	R/W			√	
SM375	Logic reverse rotation of near-point signal	Y7 is applicable to DSZR	R/W			√	
SM376	Logic reverse rotation of zero-point signal	Y7 is applicable to DSZR	R/W			√	

23. Signal alarm

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SM400	Signal alarm valid	Set SM400 after ON, SM401 and SD401 works	R/W			√	
SM401	Signal alarm action	Any of S900-S999 acts, set SM401 to ON	R/W			√	

24. Time output instruction

Addr.	Name	Action and function	R/W	EC20	EC10	EC20 H	EC10V
SM430	Timer clock output 1	For DUTY instruction	R/W			√	
SM431	Timer clock output 1	For DUTY instruction	R/W			√	
SM432	Timer clock output 1	For DUTY instruction	R/W			√	
SM433	Timer clock output 1	For DUTY instruction	R/W			√	
SM434	Timer clock output 1	For DUTY instruction	R/W			√	

25. CANopen instruction

Addr.	Name	Action and function	R/W	EC20	EC1 0	EC20H	EC10V
SM440	CANopen instruction finish					√	
SM441	CANopen instruction error					√	
SM442	CANopen instruction is being executed					√	

Appendix 2 Special data register

Note

1. All special data registers except SD50~SD55 will be initialized when the PLC changes from STOP to RUN.
2. The reserved SD and SM elements are not listed in the table. The reserved SD elements are by default read only.

1. PLC work state data

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V	Range
SD00	PLC type	11 means EC10, 20 means EC20 110 means EC10A, 112 means EC10V 111 means EVC-UPRG 28 means EC20H	R	√	√	√		
SD01	Version	For example, 100 means V1.00	R	√	√	√	√	
SD02	Capacity of user program	For example, 8 means an 8k step program	R	√	√	√	√	
SD03	System error code	Store the code of occured system error	R	√	√	√	√	
SD04	Battery voltage	Unit: 0.1V, for example 36 means 3.6V	R	√		√	√	
SD05	Setting of AC power failure detection delay	Configurable only through system block. Any setting smaller than 10ms or bigger than 100ms will be regarded as 10ms or 100ms respectively	R	√		√	√	10~100ms
SD07	Number of extension I/O module		R	√	√	√	√	
SD08	Number of special module		R	√	√	√	√	
SD09	Setting the input points for operation control. Decimal (X0 is displayed as 0; X10, 8. Maximum:15). Configurable through system block		R	√	√	√	√	0-15
SD10	Number of main module I/O points	High byte: input Low byte: output	R	√	√	√	√	
SD11	Number of extension module I/O points	High byte: input Low byte: output	R	√	√	√	√	
SD12	Number of main module analog I/O points	High byte: input Low byte: output	R		√	√	√	
SD16	High-speed ring counter	0-20971 (Unit: 0.1ms, 16 bits) up ring counter	R/W		√			0-32767

2. Operation error code FIFO area

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V	Range
SD20	Reserved operation error code 0	In the order of arrival, the latest five operation error codes are reserved. SD20 always stores the latest error codes	R	√	√	√	√	
SD21	Reserved operation error code 1		R	√	√	√	√	
SD22	Reserved operation error code 2		R	√	√	√	√	
SD23	Reserved operation error code 3		R	√	√	√	√	
SD24	Reserved operation error code 4		R	√	√	√	√	

3. FROM/TO error

Addr.	Name	R/W	EC20	EC10V	EC10	EC20H	Range

SD25	Special modules' numbering is wrong (starting with 0) when using FROM/TO instruction	R	√	√	√	√	Initial value: 255
SD26	The I/O chips' numbering is wrong (starting with 0) when refreshing I/O	R	√	√	√	√	Initial value: 255

4. Scan time

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V	Range
SD30	Current scan value	Current scan time (unit: ms)	R	√	√	√	√	
SD31	Min. scan time	Min. scan time (uint: ms)	R	√	√	√	√	
SD32	Max. scan time	Max. scan time (unit: ms)	R	√	√	√	√	
SD33	Constant scan time	Default: 0ms. Unit: 1ms. When the constant scan time is longer than the user monitoring overtime setting, user program overtime alarm will be raised. When a scan cycle of user program is longer than the constant scan time, the cycle constant scan mode is invalid automatically and no alarm will be raised. SD33 is regarded as 1000ms when it is set bigger than 1000ms (configurable only through the system block)	R	√	√	√	√	0~1000ms
SD34	User program overtime	Default: 100ms. Unit: 1ms. Any setting smaller than 100 or bigger than 1000 will be regarded as 100 or 1000 respectively. Configurable only through system block	R	√	√	√	√	100~1000 ms

Note

1. The error tolerance of SD30, SD31 and SD32 is 1ms.

2. It is recommended to set the user program overtime (SD34) at least 5ms bigger than the constant scan time (SD33). Otherwise, due to the influence of system operation and user program, the system is apt to report user program overtime error.

5. Input filtering constant setting

Addr.	Name	Action and function	R/W	EC20	EC10	EC20H	EC10V
SD35	Input filtering constant	Configurable only through system block	R	0-60ms	0,2,4,8,16,32,64 ms	0-60ms	0,2,4,8,16,32,64 ms
SD36	Input filtering constant	Configurable only through system block	R		0,2,4,8,16,32,64 ms		0,2,4,8,16,32,64 ms

Note

1. EC20 has only one group of filtering SD35, available input X0~X17.

2. EC10 and EC10A have two groups of filtering: SD35, available input X0~X3; SD36, available input X4~X7. Range: 0, 2, 4, 8, 16, 32, 64ms.

3. EC10 and EC10A have two groups of filtering: SD35, available input X0~X3; SD36, available input X4~X7. Range: 0~64ms.

6. High-speed pulse output monitoring

Addr.	Name	R/W	EC20	EC10	EC20H	EC10V	Range
SD50	PLSY accumulated output Y0 total pulse number (MSB)	R/W	√	√	√	√	
SD51	PLSY accumulated output Y0 total pulse number (LSB)	R/W	√	√	√	√	
SD52	PLSY accumulated output Y1 total pulse number (MSB)	R/W	√	√		√	
SD53	PLSY accumulated output Y1 total pulse number (LSB)	R/W	√	√		√	

Addr.	Name	R/W	EC20	EC10	EC20H	EC10V	Range
SD54	PLSY accumulated output Y1, Y0 total pulse number (MSB)	R/W	√	√			
SD55	PLSY accumulated output Y1, Y0 total pulse number (LSB)	R/W	√	√			
SD56	Current section of the PLS instruction that outputs Y0	R		√	√	√	
SD57	Current section of the PLS instruction that outputs Y1	R		√		√	
SD160	PLSR/PLSY accumulated output Y2 total pulse number (MSB)	R/W			√	√	
SD161	PLSR/PLSY accumulated output Y2 total pulse number (LSB)	R/W			√	√	
SD162	PLSR/PLSY accumulated output Y3 total pulse number (MSB)	R/W				√	
SD163	PLSR/PLSY accumulated output Y3 total pulse number (LSB)	R/W				√	
SD164	PLSR/PLSY accumulated output Y4 total pulse number (MSB)	R/W			√		
SD165	PLSR/PLSY accumulated output Y4 total pulse number (LSB)	R/W			√		
SD166	PLSR/PLSY accumulated output Y5 total pulse number (MSB)	R/W			√		
SD167	PLSR/PLSY accumulated output Y5 total pulse number (LSB)	R/W			√		
SD168	PLSR/PLSY accumulated output Y6 total pulse number (MSB)	R/W			√		
SD169	PLSR/PLSY accumulated output Y6 total pulse number (LSB)	R/W			√		
SD252	Current section of the PLS instruction that outputs Y2	R			√	√	
SD253	Current section of the PLS instruction that outputs Y3	R				√	
SD254	Current section of the PLS instruction that outputs Y4	R			√		
SD255	Current section of the PLS instruction that outputs Y5	R			√		
SD256	Current section of the PLS instruction that outputs Y6	R			√		
SD257	Current section of the PLS instruction that outputs Y7	R			√		

7. Timed interrupt cycle

Addr.	Name	Register content	R/W	EC20	EC10	EC20H	EC10V	Range
SD66	Cycle of timed interrupt 0	The interrupt will not occur when the value is not within 1~32767	R/W	√	√	√	√	1~32767ms
SD67	Cycle of timed interrupt 1	The interrupt will not occur when the value is not within 1~32767	R/W	√	√	√	√	1~32767ms
SD68	Cycle of timed interrupt 2	The interrupt will not occur when the value is not within 1~32767	R/W	√	√	√	√	1~32767ms

Note:

An error of ±1ms may occur when the system processes a user timed interrupt. To ensure the normal operation of the interrupt, it is recommended to set the cycle of timed interrupts to be bigger or equal to 5ms.

8. Positioning instruction

Addr.	Name	Initial value	Function	R/W	EC20	EC10	EC20H	EC10V
SD80	The current value of Y0 output positioning instruction (MSB)	0	As the current values of Y000 output positioning instruction for data register	R/W		√		
SD81	The current value of Y0 output positioning instruction (LSB)			R/W		√		
SD82	The current value of Y1 output positioning instruction (MSB)	0	As the current values of Y001 output positioning instruction for data register	R/W		√		

Addr.	Name	Initial value	Function	R/W	EC20	EC10	EC20H	EC10V
SD83	The current value of Y1 output positioning instruction (LSB)			R/W		√		
SD84	Base speed of executing instructions ZRN, DRVI and DRVA	5000	Base speed when executing instructions ZRN, PLSV, DRVI and DRVA (below 1/10 of the Max. speed)	R/W		√		
SD85	Max. speed of executing instructions ZRN, DRVI and DRVA (MSB)	100000	Max. speed when executing instructions ZRN, PLSV, DRVI and DRVA (10-100000)	R/W		√		
SD86	Max. speed of executing instructions ZRN, DRVI and DRVA (LSB)			R/W		√		
SD87	ACC/DEC time of executing instructions ZRN, DRVI and DRVA	1000	ACC/DEC time when raising from the base speed (SD84) to the Max. speed (SD85, SD86) at executing instructions ZRN, DRVI and DRV (50ms-5000ms)	R/W		√		
SD240	Interrupt input designation	0	Interrupt input designation	R/W	√			√

9. Real-time clock

Addr.	Name	Register content	R/W	EC20	EC10	EC20H	EC10V	Range
SD100	Year	For real-time clock	R	√	√	√	√	2000~2099
SD101	Month	For real-time clock	R	√	√	√	√	1~12 months
SD102	Day	For real-time clock	R	√	√	√	√	1~31 days
SD103	Hour	For real-time clock	R	√	√	√	√	0~23 hours
SD104	Minute	For real-time clock	R	√	√	√	√	0~59 minutes
SD105	Second	For real-time clock	R	√	√	√	√	0~59 seconds
SD106	Week	For real-time clock	R	√	√	√	√	0 (Sunday)~6 (Saturday)
Note: You can set these elements only with the TWR instruction or through the upper computer.								

10. Reception control and state of free port (Port 0)

Addr.	Name		Register content	R/W	EC20	EC10	EC20H	EC10V	Range
SD110	Free port0 mode state word	SD110.0~SD110.2 port baud rate	b2, b1, b0 000=38,400 001=19,200 010=9,600 011=4,800 100=2,400 101=1,200 110=57,600 111=115,200	R	√	√	√	√	
		SD110.3 stop bit	0=1 stop bit 1=2 stop bits						
		SD110.4 parity check	0=even parity 1=odd parity						
		SD110.5 parity check	0=no check						

Addr.	Name		Register content	R/W	EC20	EC10	EC20H	EC10V	Range
		enabling	1=check						
		SD110.6 character data bit	Data bit of every character 0=8 bits 1=7 bits						
		SD110.7 free port receiving start character start mode	1=start character specified 0=start character unspecified						
		SD110.8 free port receiving end character mode	1=end character specified 0=end character unspecified						
		SD110.9 free port word overtime enabling	1: word overtime enabled 0: word overtime disabled						
		SD110.10 free port frame overtime enabling	1=frame overtime enabled 0=frame overtime disabled						
		SD110.11	Reserved						
		SD110.12 high/low byte valid	0: word element valid at low byte 1: word element valid at high/ low byte						
		SD110.13~SD110.15	Reserved						
SD111	Start character			R/W	√	√	√	√	
SD112	End character			R/W	√	√	√	√	
SD113	Word overtime setting		Default: 0ms (word overtime omitted)	R/W	√	√	√	√	1~32767ms
SD114	Frame overtime setting		Default: 0ms (frame overtime omitted)	R/W	√	√	√	√	1~32767ms
SD115	Receiving completion message code		Bit 0: set when receiving ends Bit 1: set when specified end character is received Bit 2: set when Max. character number is received Bit 3: set upon word overtime Bit 4: set upon frame overtime Bit 5: set upon parity check error Bits 6~15: reserved	R	√	√	√	√	
SD116	Current received characters			R	√	√	√	√	
SD117	Total number of current received characters			R	√	√	√	√	
SD118	Current sent characters			R		√	√	√	

11. Reception control and state of free port (Port 1)

Addr.	Name		Register content	R/W	EC20	EC10	EC20H	EC10V	Range
SD120	Free port1 mode state word	SD120.0~SD120.2 port baud rate	b2, b1, b0 000=38,400 001=19,200 010=9,600 011=4,800 100=2,400 101=1,200 110=57,600 111=115,200	R/W	√	√	√	√	
		SD120.3 stop bit	0=1 stop bit 1=2 stop bits						
		SD120.4 parity check	0=even parity						

Addr.	Name		Register content	R/W	EC20	EC10	EC20H	EC10V	Range
			1=odd parity						
		SD120.5 parity check enabling	0=disabled 1=enabled						
		SD120.6 character data bit	Data bit of every character 0=8 bits 1=7 bits						
		SD120.7 free port receiving start character mode	1: start character specified 0: start character unspecified						
		SD120.8 free port receiving end character mode	1: end character specified 0: end character unspecified						
		SD120.9 free port word overtime enabling	1: word overtime enabled 0: word overtime disabled						
		SD120.10 free port frame overtime enabling	1: frame overtime enabled 0: frame overtime disabled						
		SD120.11	Reserved						
		SD120.12 high/low byte valid	0: word element valid at low byte 1: word element valid at high/ low byte						
		SD120.13~SD120.15	Reserved						
SD121	Start character			R/W	√	√	√	√	
SD122	End character			R/W	√	√	√	√	
SD123	Word overtime setting		Default: 0ms (word overtime omitted)	R/W	√	√	√	√	0~32767ms
SD124	Frame overtime setting		Default: 0ms (frame overtime omitted)	R/W	√	√	√	√	0~32767ms
SD125	Receiving completion message code		Bit 0: set when receiving ends Bit 1: set when specified end character is received Bit 2: set when Max. character number is received Bit 3: set upon word overtime Bit 4: set upon frame overtime Bit 5: set upon parity check error Bits 6~15: reserved	R	√	√	√	√	
SD126	Current received characters			R	√	√	√	√	
SD127	Total number of current received characters			R	√	√	√	√	
SD128	Current sent characters			R	√	√	√	√	

12. Modbus/N:N bus setting

Addr.	Name	R/W	EC20	EC10	EC20H	EC10V	Range
SD130	Local station No. (Port 0)	R/W	√	√	√	√	MOD (1~32) , EMR (0~31)
SD131	Max. timeout time of Port 0 (post-sending and pre-receiving) / N:N bus extra delay	R/W		√	√	√	
SD132	Port 0 retry times	R/W	√	√	√	√	
SD133	N:N bus network update mode (Port 0)	R/W	√	√	√	√	1~13
SD134	Error code of Modbus master Code(COM0)	R					

SD135	Local station No. (Port 1)	R/W	√	√	√	√	MOD (1~32), EMR (0~31)
SD136	Max. timeout time of Port 1 (post-sending and pre-receiving) /N:N bus extra delay	R/W	√	√	√	√	
SD137	Port 1 retry times	R/W	√	√	√	√	0~100
SD138	N:N bus network update mode (Port 1)	R/W	√	√	√	√	1~13

13. Reception control and state of extension free port (Port 2)

Addr.	Name		Register content	R/W	EC20	EC10	EC20H	EC10V	Range
SD140	Free port1 mode state word	SD140.0~SD140.2 port baud rate	b2, b1, b0 000=38,400 001=19,200 010=9,600 011=4,800 100=2,400 101=1,200 110=57,600 111=115,200	R/W			√	√	
		SD140.3 stop bit	0=1 stop bit 1=2 stop bits						
		SD140.4 parity check	0=even parity 1=odd parity						
		SD140.5 parity check enabling	0=disabled 1=enabled						
		SD140.6 character data bit	Data bit of every character 0=8 bits 1=7 bits						
		SD140.7 free port receiving start character mode	1: start character specified 0: start character unspecified						
		SD140.8 free port receiving end character mode	1: end character specified 0: end character unspecified						
		SD140.9 free port word overtime enabling	1: word overtime enabled 0: word overtime disabled						
		SD140.10 free port frame overtime enabling	1: frame overtime enabled 0: frame overtime disabled						
SD140		SD140.11	Reserved	R/W			√	√	
		SD140.12 high/low byte valid	0: word element valid at low byte 1: word element valid at high/ low byte						
		SD140.13~SD140.15	Reserved						
SD141	Start character			R/W			√	√	
SD142	End character			R/W			√	√	
SD143	Word overtime setting		Default: 0ms (word overtime omitted)	R/W			√	√	0~32767ms
SD144	Frame overtime setting		Default: 0ms (frame overtime omitted)	R/W			√	√	0~32767ms
SD145	Receiving completion message code		Bit 0: set when receiving ends Bit 1: set when specified end character is received Bit 2: set when Max. character number is received Bit 3: set upon word	R			√	√	

Addr.	Name	Register content	R/W	EC20	EC10	EC20H	EC10V	Range
		overtime Bit 4: set upon frame overtime Bit 5: set upon parity check error Bits 6~15: reserved						
SD146	Current received characters		R			√	√	
SD147	Total number of current received characters		R			√	√	
SD148	Current sent characters		R			√	√	

14. Modbus setting (Port 2)

Addr.	Name	R/W	EC20	EC10	EC20H	EC10V	Range
SD150	Local station No. (Port 2)	R/W			√	√	MOD (1~32)
SD151	Max. timeout time of Port 2 (post-sending and pre-receiving)	R/W			√	√	
SD152	Port 2 retry times	R/W			√	√	0~100
SD153	N:N bus COM2 update mode	R			√	√	1~18 (default 3)
SD154	N:N bus COM0 polling cycle	R			√	√	
SD155	N:N bus COM1 polling mode	R			√	√	
SD156	N:N bus COM2 polling cycle	R			√	√	
SD159	Error code of Modbus master (Port 2)	R			√	√	

15. Setting and reading of integrated analog signal channel

Addr.	Name	R/W	EC20	EC10	EC20H	EC10V	Range
SD172	Average sample value of A/D CH1	R		√		√	-10000~+10000
SD173	Sampling times of A/D CH1	R/W		√		√	1~1000
SD174	Average sample value of A/D CH2	R		√		√	-10000~+10000
SD175	Sampling times of A/D CH2	R/W		√		√	1~1000
SD176	Average sample value of A/D CH3	R		√		√	-10000~+10000
SD177	Sampling times of A/D CH3	R/W		√		√	1~1000
SD185	Average sample value of A/D CH4	R		√		√	-10000~+10000
SD186	Sampling times of A/D CH4	R/W		√		√	1~1000
SD187	Average sample value of A/D CH5	R		√		√	-10000~+10000
SD188	Sampling times of A/D CH5	R/W		√		√	1~1000
SD189	Average sample value of A/D CH6	R		√		√	-10000~+10000
SD190	Sampling times of A/D CH6	R/W		√		√	1~1000
SD178	Output value of D/A CH1	R/W		√		√	-10000~+10000
Note: The default value of SD173, SD175, SD177, SD186, SD188 and SD190 is 8.							

16. Usage of DHSP and DHST instructions

Addr.	Name	R/W	EC20	EC10	EC20H	EC10V	Range
SD180	MSB of DHSP table comparison output data	R/W		√	√	√	
SD181	LSB of DHSP table comparison output data	R/W		√	√	√	
SD182	MSB of DHST or DHSP table comparison data	R/W	√	√	√	√	
SD183	LSB of DHST or DHSP table comparison data	R/W	√	√	√	√	
SD184	Record No. of the table being executed	R/W	√	√	√	√	

17. Error flag

Addr.	Name	R/W	EC20	EC10	EC20H	EC10V	Range
SD191	No. of the module where bus error occured	R	√	√	√	√	
SD192	No. of the special module where bus error occured	R	√	√	√	√	
SD193	MODBUS detailed error (COM0)	R	√	√	√	√	
SD194	MODBUS detailed error (COM1)	R	√	√		√	
SD195	MODBUS detailed error (COM2)	R				√	

18. Enhanced positioning instruction

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD200	Current data register of Y000 output locating instruction	0	R/W		√	√	√
SD201							
SD202	Max. speed when Y0 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (10~100000)	100000	R/W		√	√	√
SD203							
SD204	Base speed when Y0 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (below 1/10 of Max. speed)	5000	R/W		√	√	√
SD205	ACC/DEC time (50ms~5000ms) for base speed (SD204) rising to the Max. speed (SD202, SD203) when Y0 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W		√	√	√
SD206	Y0 clear signal element designation		R/W		√	√	√
SD207	Y0 crawling speed, applicable to DSZR	1000	R/W		√	√	√
SD208	Y0 origin return speed, applicable to DSZR	50000	R/W		√	√	√
SD209							
SD220	Y0 clear signal element designation				√		
SD260	DEC time (50ms~5000ms) for the Max. speed reducing to the base speed when Y0 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W				√

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD210	Current data register of Y001 output locating instruction	0	R/W		√		√
SD211							
SD212	Max. speed when Y1 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (10~100000)	100000	R/W		√		√
SD213							
SD214	Base speed when Y1 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (below 1/10 of Max. speed)	5000	R/W		√		√
SD215	ACC/DEC time (50ms~5000ms) for base speed (SD204) rising to the Max. speed (SD202, SD203) when Y1 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W		√		√
SD216	Y1 clear signal element designation		R/W		√		√
SD217	Y1 crawling speed, applicable to DSZR	1000	R/W		√		√
SD218	Y1 origin return speed, applicable to DSZR	50000	R/W		√		√

SD219							
SD230	Y1 clear signal element designation	0			√		
SD261	DEC time (50ms~5000ms) for the Max. speed reducing to the base speed when Y1 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W				√

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD320	Current data register of Y002 output locating instruction	0	R/W			√	√
SD321							
SD322	Max. speed when Y2 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (10~100000)	100000	R/W			√	√
SD323							
SD324	Base speed when Y2 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (below 1/10 of Max. speed)	5000	R/W			√	√
SD325	ACC/DEC time (50ms~5000ms) for base speed (SD204) rising to the Max. speed (SD202, SD203) when Y2 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W			√	√
SD326	Y2 clear signal element designation		R/W			√	√
SD327	Y2 crawling speed, applicable to DSZR	1000	R/W			√	√
SD328	Y2 origin return speed, applicable to DSZR	50000	R/W			√	√
SD329							
SD262	DEC time (50ms~5000ms) for the Max. speed reducing to the base speed when Y2 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W				√

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD330	Current data register of Y003 output locating instruction	0	R/W				√
SD331							
SD332	Max. speed when Y3 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (10~100000)	100000	R/W				√
SD333							
SD334	Base speed when Y3 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (below 1/10 of Max. speed)	5000	R/W				√
SD335	ACC/DEC time (50ms~5000ms) for base speed (SD204) rising to the Max. speed (SD202, SD203) when Y3 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W				√
SD336	Y3 clear signal element designation		R/W				√
SD337	Y3 crawling speed, applicable to DSZR	1000	R/W				√
SD338	Y3 origin return speed, applicable to DSZR	50000	R/W				√
SD339							
SD263	DEC time (50ms~5000ms) for the Max. speed reducing to the base speed when Y3 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W				√

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD340	Current data register of Y004 output locating instruction	0	R/W			√	

SD341							
SD342	Max. speed when Y4 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (10~100000)	100000	R/W			√	
SD343							
SD344	Base speed when Y4 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (below 1/10 of Max. speed)	5000	R/W			√	
SD345	ACC/DEC time (50ms~5000ms) for base speed (SD204) rising to the Max. speed (SD202, SD203) when Y4 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W			√	
SD346	Y4 clear signal element designation		R/W			√	
SD347	Y4 crawling speed, applicable to DSZR	1000	R/W			√	
SD348	Y4 origin return speed, applicable to DSZR	50000	R/W			√	
SD349							

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD350	Current data register of Y005 output locating instruction	0	R/W			√	
SD351							
SD352	Max. speed when Y5 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (10~100000)	100000	R/W			√	
SD353							
SD354	Base speed when Y5 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (below 1/10 of Max. speed)	5000	R/W			√	
SD355	ACC/DEC time (50ms~5000ms) for base speed (SD204) rising to the Max. speed (SD202, SD203) when Y5 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W			√	
SD356	Y5 clear signal element designation		R/W			√	
SD357	Y5 crawling speed, applicable to DSZR	1000	R/W			√	
SD358	Y5 origin return speed, applicable to DSZR	50000	R/W			√	
SD359							

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD360	Current data register of Y006 output locating instruction	0	R/W			√	
SD361							
SD362	Max. speed when Y6 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (10~100000)	100000	R/W			√	
SD363							
SD364	Base speed when Y6 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (below 1/10 of Max. speed)	5000	R/W			√	
SD365	ACC/DEC time (50ms~5000ms) for base speed (SD204) rising to the Max. speed (SD202, SD203) when Y6 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W			√	
SD366	Y6 clear signal element designation		R/W			√	
SD367	Y6 crawling speed, applicable to DSZR	1000	R/W			√	
SD368	Y6 origin return speed, applicable to DSZR	50000	R/W			√	
SD369							

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V

SD370	Current data register of Y007 output locating instruction	0	R/W			√	
SD371							
SD372	Max. speed when Y7 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (10~100000)	100000	R/W			√	
SD373							
SD374	Base speed when Y7 executes ZRN, PLSV, DRVI, DRVA, DSZR and DVIT instructions (below 1/10 of Max. speed)	5000	R/W			√	
SD375	ACC/DEC time (50ms~5000ms) for base speed (SD204) rising to the Max. speed (SD202, SD203) when Y7 executes ZRN, DRVI, DRVA, DSZR and DVIT instructions	1000	R/W			√	
SD376	Y7 clear signal element designation		R/W			√	
SD377	Y7 crawling speed, applicable to DSZR	1000	R/W			√	
SD378	Y7 origin return speed, applicable to DSZR	50000	R/W			√	
SD379							

19. Signal alarm instruction

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD401	Store the Min. No. in S900-S999	0	R/W			√	

20. Timing output instruction

Addr.	Name	Initial value	R/W	EC10	EC20	EC20H	EC10V
SD430	Scan times of timing clock output 1		R/W			√	
SD431	Scan times of timing clock output 2		R/W			√	
SD432	Scan times of timing clock output 3		R/W			√	
SD433	Scan times of timing clock output 4					√	
SD434	Scan times of timing clock output 5					√	

Appendix 3 Reserved elements

Description	EC10/EC10V/EC20		EC10A	
Buffer area for transmission of inverter instructions	D7940	D7969	D3940	D3969
Buffer area for reception of inverter instructions	D7970	D7999	—	—
N:N bus network shared area	D7700	D7763	—	—
N:N enhanced mode (mode14-18) network shared area	D7500	D7755	—	—
N:N network shared area	M1400	M1911	—	—
EROMWR instruction operation area	D6000	D6999	—	—

Appendix 4 Modbus communication error code

Error codes	Description
0x01	Illegal function code
0x02	Illegal register address
0x03	Data number error
0x10	Communication overtime (longer than the preset maximum communication time)
0x11	Data frame reception error
0x12	Parameter error (mode or master/slave parameter setting error)
0x13	Error occurs because the local station number coincides with the station number set by the instruction
0x14	Element address overflow (the data received or sent is too much for the storing area)
0x15	Instruction execution failure
0x16	The received slave address does not match with the requested slave address, the specific error code element stores the received slave address
0x17	The received function code does not match with the requested function code, the specific error code element stores the received function code
0x18	Information frame error: only refer to the starting element address does not match, the specific error code element stores the received starting element address
0x19	The received data length does not conform to the protocol or the number of elements exceeds the maximum limit of the specified function code
0x20	CRC/LRC check error
0x21	Reserved
0x22	Starting element address setting error for instruction parameter
0x23	Unavailable function code or illegal function code set by instruction parameter
0x24	Number of elements setting error for instruction parameter
0x25	Reserved
0x26	Parameter unchangeable during operation (only applicable to EV3000)
0x27	Parameters under password protection

Appendix 5 Inverter instruction error code

Error code	Description
0x1	Illegal function code
0x2	Illegal register address
0x3	Data error (data outside the range)
0x4	Slave operation failure (including the error due to invalid data, although the data is in the range)
0x5	Valid instruction. Processing. Mainly used to save data to EEPROM
0x6	Slave busy. Please try again later. Mainly used to save data to EEPROM
0x18	Information fram error, including information length error and parity check error
0x20	Parameter unchangeable
0x21	Parameter unchangeable during operation (only applicable to EV3100)
0x22	Parameters under password protection

Appendix 6 System error code

Error code	Description	Error type	Description	EC10	EC20
0	No error			√	√
1~9	Reserved			√	√
System hardware error					
10	SRAM error	System error	User program stops, and ERR indicator turns on. To remove this fault, power off the PLC and check the hardware	√	
11	FLASH error	System error	User program stops, and ERR indicator turns on. To remove this fault, power off the PLC and check the hardware	√	
12	Communication port error	System error	User program stops, and ERR indicator turns on. To remove this fault, power off the PLC and check the hardware	√	
13	Real-time clock error	System error	User program stops, and ERR indicator turns on. To remove this fault, power off the PLC and check the hardware	√	
14	I2C error	System error	User program stops, and ERR indicator turns on. To remove this fault, power off the PLC and check the hardware	√	
External setting error (20~23)					
20	Serious local I/O error	System error	User program stops, and ERR indicator turns on. To remove this fault, power off the PLC and check the hardware	√	
21	Serious extension I/O error	System error	ERR indicator blinks. This alarm is cleared automatically upon the removal of the fault	√	
22	Serious special module error	System error	ERR indicator blinks. This alarm is cleared automatically upon the removal of the fault	√	
23	Update error of real-time clock (incorrect time is read during system update)	System error	ERR indicator blinks. This alarm is cleared automatically upon the removal of the fault	√	
24	EEPROM write/read operation error	System error	ERR indicator blinks. This alarm is cleared automatically upon the removal of the fault	√	
25	Local analog signal error	System error	ERR indicator blinks. This alarm is cleared automatically upon the removal of the fault	√	
26	System special module configuration error	System error	ERR indicator blinks. This alarm is cleared automatically upon the removal of the fault	√	
Storage error (40~45)					
40	User program file error	System error	User program stops, and ERR indicator turns on. To remove this fault, download new program or format the disk	√	√
41	System configuration file error	System error	User program stops, and ERR indicator turns on. To remove this fault, download new system configuration files or format the disk	√	√
42	Data block file error	System error	User program stops, and ERR indicator turns on. To remove this fault, download new data block file or format the disk	√	√
43	Battery backup data lost	System error	User program keeps running ERR indicator blinks. To remove this fault, clear the register, or format the disk, or reset	√	√
44	Forced-table lost	System error	User program keeps running. ERR indicator blinks. To remove this fault, clear the register, or force, or format the disk, or reset	√	√
45	User information file error	System error	User program keeps running ERR indicator is off. To remove this fault, download new program and data block files, or format the disk	√	√
46~59	Reserved				

Error code	Description	Error type	Description	EC10	EC20
Instruction execution error (60~75)					
60	User program compilation error	Execution error	User program stops, and ERR indicator turns on.	√	√
61	User program operation overtime error	Execution error	User program stops, and ERR indicator turns on.	√	√
62	Illegal user program instruction execution error	Execution error	User program stops, and ERR indicator turns on.	√	√
63	Illegal element type of instruciton operand	Execution error	User program stops, and ERR indicator turns on.	√	√
64	Illegal instruction operand value	Execution error	User program keeps running, ERR indicator keeps off. The corresponding error code will be prompted in SD20	√	√
65	Outside instruction element range	Execution error		√	√
66	Subprogram stack overflow	Execution error		√	√
67	User interrupt request queue overflow	Execution error		√	√
68	Illegal label jump or subprogram call	Execution error		√	√
69	Divided by 0 error	Execution error		√	√
70	Definition error of stack operand	Execution error	When stack size, or stack elements are smaller than zero, or stack element number exceeds the limit of stack size	√	√
71	Reserved			√	√
72	Undefined user subprogram or interrupt subprogram	Execution error		√	√
73	Special module address invalid	Execution error			√
74	Special module access error	Execution error			√
75	I/O error when using REF instruction	Execution error		√	√
76	Cannot set real time clock time using TWR	Execution error		√	√
77	Parameter 3 of PLSR instruction inappropriate under constant scan	Execution error		√	√
78	BFM unit of accessed special module exceeds range	Execution error			√
79	ABS data read timeout	Execution error		√	
80	ABS data read and check error	Execution error		√	

Appendix 7 Modbus communication protocol (EC10, EC20 series)

1.Modbus communication protocol overview

EC series small PLC has two communication ports: Port 0 (also the programming port), which supports Modbus slave station, and Port 1 , which supports Modbus master station and slave station (configurable through ConstrolStar). Their features include:

1. Using RS485 or RS232 port, with RS232 3-line system as the physical interface.
2. Supportive of RTU mode and ASCII mode, but not of the change of the ASCII ending character.
3. Being the Modbus slave station, the addresses range from 1 to 31.
4. Supportive of broadcast mode. The broadcast is effective for write and sub-function codes of diagnosis.
5. Suppoting baud rates including 38,400 bps, 19,200 bps, 9,600 bps, 4,800 bps, 2,400 bps and 1,200 bps. (Default: 19200, 8 bits, 1 stop bit, even check)
6.Supportive of data field 2×252 bytes (ASII mode) or 252 bytes (RTU mode).

2.Supported Modbus function code and element addressing

The slave station supports function codes 01, 02, 03, 05, 06, 08, 15, 16 (decimal).

Pay attention to the following points during the reading:

Relationship between read-write element function code and the element

Function code	Name of function code	Modicon data address	Type of operational element	Remark
01	read coil status	0$^{\text{Note 1}}$:xxxx	Y, X, M, SM, S, T, C	Bit read
02	read discrete input status	1$^{\text{Note 2}}$:xxxx	X	Bit read
03	read register status	4$^{\text{Note 3}}$:xxxx$^{\text{Note 4}}$	D, SD, Z, T, C	Word read
05	write single coil status	0:xxxx	Y, M, SM, S, T, C	Bit write
06	write single register status	4:xxxx	D, SD, Z, T, C	Word write
15	write multiple coils status	0:xxxx	Y, M, SM, S, T, C	Bit write
16	write multiple registers status	4:xxxx	D, SD, Z, T, C	Word write

Note:

1. 0 means “coil”.
2. 1 means “discrete input”.
3. 4 means “register”.
4. xxxx means range “1~9999”. Each type has an independent logic address range of 1 to 9999 (protocol address starts from 0).
5. 0, 1 and 4 do not have the physical meaning and are not involved in actual addressing.
6. Users shall not write X element with function codes 05 and 15; otherwise, the system will not feed back the error information if the written operand and data are correct, but the system will not perform any operation on the write instruction.

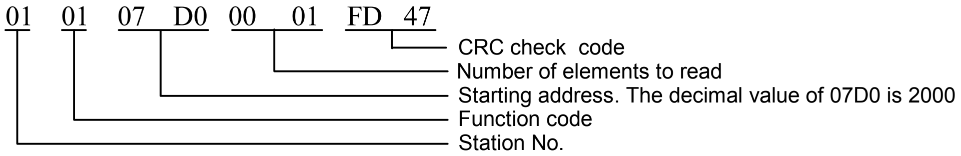
Relationship between PLC element and Modbus communication protocol address

Element	Type	Physical element	Protocol address	Supported function code	Notes
Y	bit	Y0 to Y377 (octal code) 256 points in total	0000~0255	01, 05, 15	output status, element code: Y0~Y7, Y10~Y17
X	bit	X0 to X377	1200~01455	01, 05, 15	input status, it supports

Element	Type	Physical element	Protocol address	Supported function code	Notes
		(octal code) 256 points in total	0000~0255	02	two kinds of address, the element code is same as above
M	bit	M0 to M1999	2000~3999	01, 05, 15	
SM	bit	SM0 to SM255	4400~4655	01, 05, 15	
S	bit	S0~S991	6000~6991	01, 05, 15	
T	bit	T0~T255	8000~8255	01, 05, 15	status of T element
C	bit	C0~C255	9200~9455	01, 05, 15	status of C element
D	word	D0~D7999	0000~7999	03, 06, 16	
SD	word	SD0~SD255	8000~8255	03, 06, 16	
Z	word	Z0~Z15	8500~8515	03, 06, 16	
T	word	T0~T255	9000~9255	03, 06, 16	current value of T element
C	word	C0~C199	9500~9699	03, 06, 16	current value of C element (WORD)
C	double word	C200~C255	9700~9811	03, 16	current value of C element (DWORD)
C	double word	C256~C306	10000-10101	03, 16	current value of C element (WORD)
R	word	R0~R32767	13000-45767	03, 06, 16	

Note:

The protocol address is the address used on data transfer and corresponds with the logic address of Modicon data. The protocol address starts from 0 and the logic address of Modicon data begins with 1, that is, protocol address+1=logic address of Modicon data. For example, if M0 protocol address is 2000, and its corresponding logic address of Modicon data will be 0:2001. In practice, the read and write of M0 is completed through the protocol address, for example: read M0 element frame (sent from the master):


01 01 07 D0 00 01 FD 47
CRC check code
Number of elements to read
Starting address. The decimal value of 07D0 is 2000
Function code
Station No.

Abnormal response description:

Abnormal code	Definition
0x01	Illegal function code
0x02	Illegal register address
0x03	Illegal data

Notes:

1. Elements X and Y use octal system. There are 256 points in total from X0 to X377, 256 points from Y0 to Y377, with the combinations of Y0~Y7, Y10~Y17 and Y20~Y27. etc.

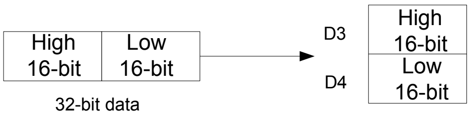
2. Two addressing methods are available for element X. One is the protocol address of 1200-1455 with corresponding function codes of 01, 05 and 15; the other is the protocol address of 0-255 with function code 02.

3. Processing of double-word element: C element is a counter. It has status and current value. C200~C255 are 32-bit elements, but each C element in the range will get two protocol addresses during the protocol address compiling. For example:

The protocol address of C200 is 9700~9701. When reading the elements though Modbus, both the starting protocol address and the number of the read elements shall be even number.

4. For most SM, SD elements, the real value cannot be written through Modbus, but PLC salve station will still return "OK" to indicate the completion of write operation, which is allowable.

5. In addition, the Modbus communication protocol of EC20 supports the read and write of double word element, LONG INT variable and floating point number. In the PLC of EC20, 32-bit data are stored with high bits at high address. For example, a 32-bit data is stored in two D elements (D3 and D4), with 16 high bits in D3 and 16 low bits in D4, as shown in the following figure: (Refer to the description for the specific example)



3.Modbus function code description

3.1 Read coil status (0x01)

Up to 256 bit-element can be read in EC series PLC.

1. Request frame

Address	Function code (01H)	Starting address		Number of elements		Check code (CRC or LRC)
		H	L	H	L	

2. Response frame

If the read address is not the times of 8, the remaining bits will be filled with 0 (starting with the high bits).

Address	Function code (01H)	Number of bytes read (n)	Read data No.1	......	Read data No.n	Check code (CRC or LRC

B7	B6	B5	B4	B3	B2	B1	B0

3.2 Read discrete input status (0x02)

In the PLC of GCM series, it specially refers to X element. The function code only supports the read function of X element with the maximum read number of 256.

1. Request frame

Address	Function code (02H)	Starting address		Number of elements		Check code (CRC or LRC)
		H	L	H	L	

2. Response frame

If the read address is not the times of 8, the remaining bits will be filled with 0 (starting with the high bits).

Address	Function code (02H)	Number of bytes read (n)	Read data No.1	...	Read data No.n	Check code (CRC or LRC

B7	B6	B5	B4	B3	B2	B1	B0

3.3 Read holding registers (0x03)

It refers to reading the value of data (word) register at the slave station, with the maximum number of registers of 125 read each time. It does not support broadcast.

1. Request frame

Address	Function Code (03H)	Starting address		Number of elements		Check code (CRC or LRC)
		H	L	H	L	

2. Response frame

Address	Function code (03H)	Number of bytes read (n)	Read data No.1		...	Read data No.n		Check code (CRC or LRC)
			H	L		H	L	

3.4 Force (write) single coil (0x05)

Force (Write) single coil writes bit element value to the slave station and supports broadcast, i.e. writing the same element to all slave stations. It supports 1 bit element at most.

1. Request frame

Address	Function code (05H)	Starting address		Written element value		Check code (CRC or LRC)
		H	L	H	L	

Note: The written value of the element is 0xFF00 (ON, 1) or 0x0000 (OFF, 0).

2. Response frame

Response frame is the repeat of request frame.

Address	Function code (05H)	Starting address		Written element value		Check code (CRC or LRC)
		H	L	H	L	

3.5 Preset (write) single register (0x06)

Preset (Write) single register writes word element value to the slave station and supports broadcast, i.e. writing the same element to all slave stations. It supports 1 bit element at most.

1. Request frame

Address	Function code (06H)	Starting address		Written element value		Check code (CRC or LRC)
		H	L	H	L	

2. Response frame

Response frame is the repeat of request frame.

Address	Function code (06H)	Starting address		Written element value		Check code (CRC or LRC)
		H	L	H	L	

3.6 Return diagnostic check (0x08)

Diagnostic register and communication error information can be obtained through returning diagnostic check.

Diagnostic code	Description
0x00	Return Request frame
0x01	Restart Comm Option
0x04	Listen Only Mode of Slave Station
0x0a	Clear Ctrs and Diagnostic Reg
0x0b	Return Bus Message Count
0x0c	Return Bus CRC Error Count
0x0d	Return Bus Exception Error Cnt
0x0e	Return Slave Message Count
0x0f	Return Slave No Response Cnt
0x12	Return Bus Char. Overrun Cnt

The frame description of sub-function code is as follows.

- **Return request frame (0x00):**

1. Request frame

Address	Function code (0x08H)	Function word	Any character	Check code

		(0x00H)	(0x00H)	H	L	(CRC or LRC)

2. Response frame

Return request frame intact.

Address	Function code (0x08H)	Function word		Any character		Check code
		(0x00H)	(0x00H)	H	L	(CRC or LRC)

- **Restart communication option (0x01):**
- **After receiving the frame, PLC will exit from Listen Only mode (Broadcast frame is supported).**

1. Request frame

The normal Data field is 0x00 00 or 0xff 00.

Address	Function code (0x08H)	Function word		Data field		Check code
		0x00H	0x01H	H	L	(CRC or LRC)

2. Response frame

Address	Function code (0x08H)	Function word		Data field		Check code
		0x00H	0x01H	H	L	(CRC or LRC)

- **Listen only mode of slave station (0x04):**
- **Slave station enters Listen Only mode. None of the instructions will be executed or responded. The slave station can only recognize the restart communication option instruction and enters the online mode after receiving the instruction (Broadcast frame is supported).**

1. Request frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x04H)	0x00H	0x00H	(CRC or LRC)

2. Response frame

No return

- **Clear counter and diagnostic register (0x0A):**
- **Clear all counters (Broadcast frame is supported).**

1. Request frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0AH)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0AH)	0x00H	0x00H	(CRC or LRC)

- **Return bus message count (0x0B):**
- **Record the total number of the messages to all master stations from the slave stations since the last starting, clearing and power-on of counter, which excludes the message of CRC error.**

1. Request frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0BH)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0BH)	H	L	(CRC or LRC)

- **CRC error count (0x0C):**
- **Record the number of CRC errors received by slave station since the last starting, clearing and power-on of counter.**

1. Request frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0CH)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0CH)	H	L	(CRC or LRC)

- **Return slave exception error count (0x0D):**
- **Record the number of the exception error that detected by slave station since the last starting, clearing and power-on of counter, which includes the error detected in the broadcast message.**

1. Request frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0DH)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0DH)	H	L	(CRC or LRC)

- **Return slave message count (0x0E)**
- **Record the number of the addressing messages received by the slave station since the last starting, clearing and power-on of counter.**

1. Request frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0EH)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0EH)	H	L	(CRC or LRC)

- **Return slave no response count (0x0F)**
- **Record the number of messages that have not returned to the slave station since the last starting, clearing and power-on of counter.**

1. Request frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0FH)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x0FH)	H	L	(CRC or LRC)

- **Return bus character overrun count (0x12)**
- **Record the number of the messages that cannot be addressed due to the character overrun since the last starting, clearing and power-on of counter.**

1. Request frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x12H)	0x00H	0x00H	(CRC or LRC)

2. Response frame

Address	Function code (0x08H)	Function word		Data field		Check code
		(0x00H)	(0x12H)	H	L	(CRC or LRC)

3.7 Force (write) multiple coils (0x0F)

At most 1968 bit elements (0x07b0) can be written and the number is changeable according to the defined range.

1. Request frame

Address	Function code (0FH)	Starting address		Number of elements		Number of bytes(n)	Written element value No.1	...	Written element value No.N	Check code (CRC or LRC)
		H	L	H	L					

B7	B6	B5	B4	B3	B2	B1	B0

2. Response frame

Address	Function code (0FH)	Starting address		Number of elements		Check code (CRC or LRC)
		H	L	H	L	

3.8 Preset (write) multiple registers (0x10 Hex)

At most 120 registers (0x78) can be written

1. Request frame

Address	Function code (0x10H)	Starting address		Number of elements		Number of bytes (n)	Written element value No.1		...	Written element value No.N		Check code (CRC or LRC)
		H	L	H	L		H	L		H	L	

2. Response frame

Address	Function code (0x10H)	Starting address		Number of elements		Check code (CRC or LRC)
		H	L	H	L	

3.9 Faulty response frame (0x80+function code)

Response Frame:

Address	Function code	Error code (see above)	Check code (CRC or LRC)

Function code refers to the function code of the captured request frame + 0x80

3.10 Points to note

1. Refer to the address classification of elements, the elements read each time shall be of the same type. For example, elements X and Y cannot be read in one frame.

2. The address and data range of the element shall be within the range specified by the protocol. For example:

For Y element, the protocol address range is 0000~0255 (Y0-Y377):

- If the read starting address is 1 and 256 elements are read, address error will be returned (error code 02), because there are only 255 Y elements that start with 1.
- If the read starting address is 0 and 257 elements are read, data error will be returned (error code 03), because the actual defined number of Y elements is only 256.
- If the read starting address is 0 and 256 elements are read, the status of 256 elements will be returned.

In other words, the read number of the elements must be within the actually defined range. It is true for read/write of bit/word elements.

4. Example of Modbus communication control

Rather than transmitting any message actively, the Modbus slave station only decides whether to respond to the message from the master station based on the specific situation after receiving the message for the local station. The slave station only supports Modbus function codes 01, 02, 03, 05, 06, 08, 15 and 16. The rest will be responded with illegal function code (except broadcast frame).

- **Read and write of element:**

Except function code 08, the other supported function codes can read and write element. In principle, one frame can read up to 2000 bit elements 125 word elements, and write 1968 bit elements and 120 word elements at most. However, the real protocol addresses are separate and discontinuous for different elements, therefore, when reading or writing an element, the elements read at one time can only be the same type and the maximum number of the read or written elements is related to the actually defined number of the elements. For example, when reading Y element (Y0-Y377), the protocol address ranges from 0 to 255, the logic address of the corresponding Modicon data is 1-256 and the maximum number of elements can be read is 256. See the following examples:

Note: The address of the slave station is 01, the last two bytes are CRC check code and the second byte is function code.

1. XMT from master station: 01 01 00 00 01 00 3D 9A

01 address; 01 function code; 00 00 starting address; 01 00 number of read elements; 3D 9A check

Slave station response: return correct response

2. XMT from master station: 01 01 00 00 01 01 FC 5A

The master station reads 01 01 elements (257), which is over the defined range of Y elements.

Slave station response: 01 81 03 00 51

The response of the slave station is illegal data, because 257>256, 256 is the allowed maximum number of Y elements.

3. XMT from master station: 01 01 00 64 00 A0 7D AD

00 64(decimal 100) is the starting address for master station to read, 00 A0 (decimal 160) is the number of the elements.

Slave station response: 01 81 02 C1 91

The response of the slave station is illegal address, because there are only 156 Y elements which are defined to start from 100 and 160 Y elements have exceeded the number.

4. XMT from master station: 01 01 01 2C 00 0A 7C 38

The master station reads 10 bit elements of 01 2C (decimal 300).

Slave station response: 01 81 02 C1 91

The response of the slave station is illegal address, because protocol address 300 has no definition of bit element.

5. XMT from master station: 01 04 00 02 00 0A D1 CD

The mater station sends the frame of function code 04.

Slave station response: 01 84 01 82 C0

The response of the slave station is illegal function code.

6. XMT from master station: 01 02 00 00 00 0A F8 0D

Master station reads input element (X element), 10 (X0-X9) from the starting address 00 00.

Slave station response: 01 02 02 00 00 B9 B8

The slave station responds with correct information, which has 02 bytes, and the content is 00 00.

7. XMT from master station: 01 01 04 B0 00 0A BC DA

Master station reads 10 bit elements(X0-X9) starting with 04 B0 (decimal 1200).

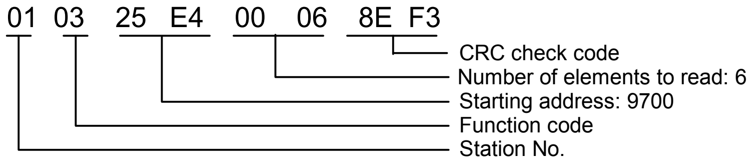
Slave station response: 01 01 02 00 00 B9 FC

The slave station responds with 02 bytes, and the content is 00 00.

Note

1. The slave station address is 01, the last two bytes are CRC check codes and the second byte is function code.
2. X element does not support write.

- **Processing of double-word elements**
- **1. The current value of C element is word element or double word element. The values from C200 to C255 are double word elements, which are read and written through the function codes (03, 16) of read/write register. The address of every two registers corresponds to one C double word element, and the registers can only be read or written in pair. For example, read the RTU fame of three C double word elements (C200-C202):**



- **In the returned data, 9700 and 9701 are the two addresses representing the content of C200. 9700 is the high 16 bits and 9701 is the low 16 bits.**
- **2. When reading the double word element, if the starting address for the reading is not an even number, the error code of illegal address will be returned. For example:**
- **XMT from master station: 01 03 25 E5 00 04 5E F2**
- **The starting address for the reading sent by the master is 25 E5 (four word elements, decimal 9701).**
- **Slave station response: 01 83 02 C0 F1**
- **Slave station response: illegal data address**
- **3. If the number of the read elements is not an even number, the error code of illegal data will be returned. For example:**
- **XMT from master station: 01 03 25 E4 00 05 CE F2**
- **25 E4: The starting address for master station reading, 5 word elements**
- **Slave response: 01 83 03 01 31**
- **Slave station returns illegal data.**
- **Processing of LONG INT data:**
- **Based on the storage method of PLC in GCM, one LONG INT data can be saved in two D elements. For example: Store one LONG INT data in D3 and D4, D3 is used for storing high 16 bits, D4 is used for storing low 16 bits in COMPANY PLC. If master station reads LONG INT data through Modbus, the 32-bit data shall be regrouped based on the LONG INT storage principle of COMPANY PLC after reading the data.**
- **Storage principle of FLOAT is the same as the storage principle of LONG INT.**

5. Description of broadcast

The slave station supports broadcast but not all the function codes. The slave station supports function codes 01, 02, 03, 05, 06, 08, 15 and 16 (decimal). Wherein, 01, 02 and 03 can read element but do not support broadcast, no response will be gotten after sending out the broadcast; 05, 06, 15 and 16 can write element and support broadcast, no response will be gotten after sending out the broadcast, but slave station will process the received data; 08 is the diagnostic function code, it does not support the broadcast except its sub-function codes 0x01, 0x04 and 0x0A (Hexadecimal).

Appendix 8 ASCII code table

ASCII HEX		High 3-bit							
		0	1	2	3	4	5	6	7
Low 4-bit	0	NUL	DLE	SPACE	0	@	P	、	p
	1	SOH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	’	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L		l	l
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	~
	F	SI	US	/	?	O	_	o	DEL

Appendix 9 Instruction index

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
A	ABS	Read current value instruction	8	Zero, carry, borrow	√	√		√
	ACOS	Floating point number ACOS instruction	7	Zero, carry, borrow				√
	ADD	Add integer instruction	7	Zero, carry, borrow	√	√	√	√
	ANB	Power flow block and instruction	1		√	√	√	√
	AND	NO contact power-flow and	1		√	√	√	√
	AND<	Compare integer AND< instruction	5		√	√	√	√
	AND<=	Compare integer AND<= instruction	5		√	√	√	√
	AND<>	Compare integer AND<> instruction	5		√	√	√	√
	AND=	Compare integer AND= instruction	5		√	√	√	√
	AND>	Compare integer AND> instruction	5		√	√	√	√
	AND>=	Compare integer AND>= instruction	5		√	√	√	√
	ANDD<	Compare double integer AND< instruction	7		√	√	√	√
	ANDD<=	Compare double integer AND<= instruction	7		√	√	√	√
	ANDD<>	Compare double integer AND<> instruction	7		√	√	√	√
	ANDD=	Compare double integer AND= instruction	7		√	√	√	√
	ANDD>	Compare double integer AND> instruction	7		√	√	√	√
	ANDD>=	Compare double integer AND>= instruction	7		√	√	√	√
	ANDR<	Compare floating point number AND< instruction	7		√	√	√	√
	ANDR<=	Compare floating point number AND<= instruction	7		√	√	√	√
	ANDR<>	Compare floating point number AND<> instruction	7		√	√	√	√
	ANDR=	Compare floating point number AND= instruction	7		√	√	√	√
	ANDR>	Compare floating point number AND> instruction	7		√	√	√	√
	ANDR>=	Compare floating point number AND>= instruction	7		√	√	√	√
	ANI	NC contact power-flow and	1		√	√	√	√
	ANR	Signal alarm reset instruction	1	Zero, carry, borrow				√
	ANS	Signal alarm set instruction	7	Zero, carry, borrow				√
	ASC	ASCII code conversion instruction	19	Zero, carry, borrow	√	√	√	√
	ASIN	Floating point number ASIN instruction	7	Zero, carry, borrow				√
	ATI	ASCII code-hexadecimal number conversion instruction	7	Zero, carry, borrow	√	√	√	√
	ATAN	Floating point number ATAN instruction	7	Zero, carry, borrow				√
	ALT	Alternate output instruction	11	Zero, carry, borrow	√		√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	ABSD	Absolute drum control instruction	9	Zero, carry, borrow	√			√
B	BAND	Word bit contact AND instruction	5		√	√	√	√
	BANI	Word bit contact ANI instruction	5		√	√	√	√
	BCD	Word to 16-bit BCD instruction	5	Zero, carry, borrow	√	√	√	√
	BIN	16-bit BCD to word instruction	5	Zero, carry, borrow	√	√	√	√
	BITS	Counting ON bit in word instruction	5		√	√	√	√
	BKADD	Add batch data operation	9	Zero, carry, borrow				√
	BKCMP=,>,<,<>,<=,>=	Compare batch data	9					√
	BKSUB	Subtract batch data operation	9	Zero, carry, borrow				√
	BLD	Word bit contact LD instruction	5		√	√	√	√
	BLDI	Word bit contact LDI instruction	5		√	√	√	√
	BMOV	Move data block transmission instruction	7		√	√	√	√
	BON	Judging ON bit in word instruction	7					√
	BOR	Word bit contact OR instruction	5		√	√	√	√
	BORI	Word bit contact ORI instruction	5		√	√	√	√
	BOUT	Word bit coil output instruction	5		√	√	√	√
B	BRST	Word bit coil reset instruction	5		√	√	√	√
	BSET	Word bit coil set instruction	5		√	√	√	√
	BTOW	Data combination instruction for byte unit	7	Zero, carry, borrow				√
C	CALL	Calling a subprogram	Depend on the program		√	√	√	√
	CCITT	CCITT check instruction	7		√	√	√	√
	CCW	Counterclockwise circular trace interpolation	12	Zero, carry, borrow				√
	CFEND	Conditional end from user main program	1		√	√	√	√
	CIRET	Conditional return from user interrupt subprogram	1		√	√	√	√
	CJ	Conditional jump	3		√	√	√	√
	COS	Floating point number COS instruction	7	Zero, carry, borrow	√	√	√	√
	CRC16	CRC16 check instruction	7		√	√	√	√
	CSRET	Conditional return from user subprogram	1		√	√	√	√
	CTR	16-bit counter loop cycle counting instruction	5		√	√	√	√
	CTU	16-bit counter counting up instruction	5		√	√	√	√
	CMP	Compare and set integer instruction	7		√*			√
	CW	Clockwise circular trace interpolation	12	Zero, carry, borrow				√
D	DADD	Add double integer instruction	10	Zero, carry, borrow	√	√	√	√
	DBAND	Dead band control	9	Zero, carry, borrow				√
	DBCD	Double word to 32-bit BCD instruction	7	Zero, carry, borrow	√	√	√	√
	DBIN	32-bit BCD to double word instruction	7		√	√	√	√
	DBITS	Counting ON bit in double word instruction	6		√	√	√	√
	DCMP<	Compare date< instruction	7		√	√		√
	DCMP<=	Compare date<= instruction	7		√	√		√
	DCMP<>	Compare date<> instruction	7		√	√		√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	DCMP=	Compare date= instruction	7		√	√		√
	DCMP>	Compare date> instruction	7		√	√		√
	DCMP>=	Compare date>= instruction	7		√	√		√
	DCNT	32-bit counting instruction	7		√	√	√	√
	DDEC	Decrement double integer instruction	4	Zero, carry, borrow	√	√	√	√
	DDIV	Divide double integer instruction	10	Zero, carry, borrow	√	√	√	√
	DEC	Decrement integer instruction	3	Zero, carry, borrow	√	√	√	√
	DEG	Floating point number rad->angle	7	Zero, carry, borrow				√
	DECO	Decode instruction	5		√	√	√	√
	DFLT	Double integer to floating point number instruction	7	Zero, carry, borrow	√	√		√
	DFMOV	Fill data block double word instruction	9		√	√	√	√
	DFROM	Read double word from special module buffer register instruction	10		√			√
	DGBIN	32-bit Gray code to double word instruction	7	Zero, carry, borrow	√	√	√	√
	DGRY	Double word to 32-bit Gray code instruction	7	Zero, carry, borrow	√	√	√	√
	DHSCI	High-speed counting interrupt trigger instruction	10		√	√	√	√
	DHSCR	High-speed counting compare reset instruction	10		√	√	√	√
	DHSCS	High-speed counting compare set instruction	10		√	√	√	√
	DHSP	High-speed counting table compare pulse output instruction	10		√	√	√	√
	DHSPI	High-speed output absolute position compare interrupt trigger instruction	10					√
	DHST	High-speed counting table compare instruction	10		√	√	√	√
	DHSZ	High-speed counting zone compare instruction	13		√	√	√	√
	DI	Disable interrupt instruction	1		√	√	√	√
	DINC	Increment double integer instruction	4	Zero, carry, borrow	√	√	√	√
	DINT	Floating point number to double integer instruction	7	Zero, carry, borrow	√	√	√	√
D	DIS	4bit separation instruction for 16bit data	7	Zero, carry, borrow				√
	DIV	Divide integer instruction	7		√	√	√	√
	DMOV	Move double word data transmission instruction	7		√	√	√	√
	DMUL	Multiply double integer instruction	10	Zero, carry, borrow	√	√	√	√
	DNEG	Negative double integer instruction	7	Zero, carry, borrow	√	√	√	√
	DRCL	32-bit carry circular shift left instruction	9	Carry	√	√	√	√
	DRCR	32-bit carry circular shift right instruction	9	Carry	√	√	√	√
	DROL	32-bit circular shift left instruction	9	Carry	√	√	√	√
	DROR	32-bit circular shift right instruction	9	Carry	√	√	√	√
	DRVA	Absolute position control instruction	11	Zero, carry, borrow	√*	√	√	√
	DRVI	Relative position control instruction	11	Zero, carry, borrow	√*	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	DSHL	32-bit shift left instruction	9		√	√	√	√
	DSHR	32-bit shift right instruction	9		√	√	√	√
	DSQT	Square root double integer instruction	7	Zero, carry, borrow	√	√	√	√
	DSUB	Subtract double integer instruction	10	Zero, carry, borrow	√	√	√	√
	DSUM	Sum double integer instruction	9	Zero, carry, borrow	√	√	√	√
	DTI	Double integer to integer instruction	6	Zero, carry, borrow	√	√	√	√
	DTO	Write double word to special module buffer register instruction	10		√			√
	DUTY	Generate timing pulse instruction	7					√
	DVABS	Double integer absolute value instruction	7	Zero, carry, borrow	√	√	√	√
	DWAND	AND double word instruction	10		√	√	√	√
	DWINV	NOT double word instruction	10		√	√	√	√
	DWOR	OR double word instruction	10		√	√	√	√
	DWXOR	Exclusive-OR double word instruction	10		√	√	√	√
	DXCH	Exchange double word instruction	7		√	√	√	√
	DABSD	Double word absolute drum control instruction	11	Zero, carry, borrow	√			√
	DSZR	Regress to origin with DOG search instruction	9	Zero, carry, borrow	√		√	√
	DVIT	Interrupt locating	11	Zero, carry, borrow	√		√	√
E	ED	Power flow falling edge detection	1		√	√	√	√
	EI	Enable interrupt instruction	1		√	√	√	√
	ENCO	Encode instruction	5		√	√	√	√
	EROMWR	EEPROM write instruction	7		√	√		√
	EU	Power flow rising edge detection	2		√	√	√	√
	IVDFWD	Inverter jogging forward rotation instruction	6		√*	√	√	√
	IVDREV	Inverter jogging reverse rotation instruction	6		√*	√	√	√
	IVFRQ	Inverter set frequency instruction	8		√*	√	√	√
	IVFWD	Inverter forward rotation instruction	6		√*	√		√
	IVRD	Inverter read single register value instruction	10		√*	√	√	√
	IVRDST	Inverter read status instruction	10		√*	√	√	√
	IVREV	Inverter reverse rotation instruction	6		√*	√	√	√
	IVSTOP	Inverter stop instruction	8		√*	√	√	√
	IVWRT	Inverter write single register value instruction	10		√*	√	√	√
	EXP	Floating point number EXP instruction	7	Zero, carry, borrow	√	√	√	√
F	FIFO	First-in-first-out instruction	7		√	√	√	√
	FLT	Integer to floating point number instruction	6	Zero, carry, borrow	√	√	√	√
	FMOV	Fill data block instruction	7		√	√	√	√
	FOR	Cycle instruction	3		√	√	√	√
	FROM	Read word from special module buffer register instruction	9		√			√
G	GBIN	16-bit Gray code to word instruction	5	Zero, carry, borrow	√	√	√	√
	GRY	Word to 16-bit Gray code instruction	5	Zero, carry, borrow	√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
H	HACKLE	Hackle wave signal output instruction	12		√	√	√	√
	HCNT	High-speed counter drive instruction	7		√	√	√	√
	HOUR	Timing list instruction	8		√	√	√	√
	HTOS	Time (hour, minute and second) to second instruction	5					√
I	INC	Increment integer instruction	3	Zero, carry, borrow	√	√	√	√
	INITR	Initialize extension register	5	Zero, carry, borrow				√
	INT	Floating point number to integer instruction	6	Zero, carry, borrow	√	√	√	√
	INV	Power flow block inverse	1		√	√	√	√
	ITA	Hexadecimal number-ASCII code conversion instruction	7	Zero, carry, borrow	√	√	√	√
	ITD	Integer to double integer instruction	6	Zero, carry, borrow	√	√	√	√
L	LBL	Jump label definition	3		√	√	√	√
	LCNV	Engineering conversion instruction	9	Zero, carry, borrow	√			√
	LD	NO contact power-flow loading	1		√	√	√	√
	LD<	Compare integer LD< instruction	5		√	√	√	√
	LD<=	Compare integer LD<= instruction	5		√	√	√	√
	LD<>	Compare integer LD<> instruction	5		√	√	√	√
	LD=	Compare integer LD= instruction	5		√	√	√	√
	LD>	Compare integer LD= instruction	5		√	√	√	√
	LD>=	Compare integer LD>= instruction	5		√	√	√	√
	LDD<	Compare double integer LD< instruction	7		√	√	√	√
	LDD<=	Compare double integer LD<= instruction	7		√	√	√	√
	LDD<>	Compare double integer LD<> instruction	7		√	√	√	√
	LDD=	Compare double integer LD= instruction	7		√	√	√	√
	LDD>	Compare double integer LD> instruction	7		√	√	√	√
	LDD>=	Compare double integer LD>= instruction	7		√	√	√	√
	LDI	NC contact power-flow loading	1		√	√	√	√
	LDR<	Compare floating point number LD< instruction	7		√	√	√	√
	LDR<=	Compare floating point number LD<= instruction	7		√	√	√	√
	LDR<>	Compare floating point number LD<> instruction	7		√	√	√	√
	LDR=	Compare floating point number LD= instruction	7		√	√	√	√
	LDR>	Compare floating point number LD> instruction	7		√	√	√	√
	LDR>=	Compare floating point number LD>= instruction	7		√	√	√	√
	LIFO	Last-in-first-out instruction	7		√	√	√	√
	LIMIT	Upper/lower limit control	9	Zero, carry, borrow				√
	LIN	Linear trace interpolation	12	Zero, carry, borrow				√
	LN	Floating point number LN instruction	7	Zero, carry, borrow	√	√	√	√
	LOADR	Read extension file register	5	Zero, carry, borrow				√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	LOG	Floating point number LOG instruction	7	Zero, carry, borrow				√
	LOGR	Log in extension register	11	Zero, carry, borrow				√
	LRC	LRC check instruction	7		√	√	√	√
M	MC	Main control	3		√	√	√	√
	MCR	Main control remove	1		√	√	√	√
	MEAN	Mean instruction	7	Zero, carry, borrow				√
	Modbus	Master station communication instruction	8		√	√	√	√
	MOV	Move word data transmission instruction	5		√	√	√	√
	MOVLINK	Synchronous control instruction	17	Zero, carry, borrow				√
	MPP	Output power-flow stack pop off	1		√	√	√	√
	MPS	Output power-flow input stack	1		√	√	√	√
	MRD	Read output power-flow stack top value	1		√	√	√	√
	MUL	Multiply integer instruction	8	Zero, carry, borrow	√	√	√	√
	MODRW	MODBUS read/write instruction	14		√*	√	√	√
N	NEG	Negative integer instruction	5	Zero, carry, borrow	√	√	√	√
	NEXT	Return from cycle	1		√	√	√	√
	NOP	No operation	1		√	√	√	√
O	OR	NO contact power-flow or	1		√	√	√	√
	OR<	Compare integer OR< instruction	5		√	√	√	√
	OR<=	Compare integer OR<= instruction	5		√	√	√	√
	OR<>	Compare integer OR<> instruction	5		√	√	√	√
	OR=	Compare integer OR= instruction	5		√	√	√	√
	OR>	Compare integer OR> instruction	5		√	√	√	√
	OR>=	Compare integer OR>= instruction	5		√	√	√	√
	ORB	Power flow or instruction	1		√	√	√	√
	ORD<	Compare double integer OR< instruction	7		√	√	√	√
	ORD<=	Compare double integer OR<= instruction	7		√	√	√	√
	ORD<>	Compare double integer OR<> instruction	7		√	√	√	√
	ORD=	Compare double integer OR= instruction	7		√	√	√	√
	ORD>	Compare double integer OR> instruction	7		√	√	√	√
	ORD>=	Compare double integer OR>= instruction	7		√	√	√	√
	ORI	NC contact power-flow or	1		√	√	√	√
	ORR<	Compare floating point number OR< instruction	7		√	√	√	√
	ORR<=	Compare floating point number OR<= instruction	7		√	√	√	√
	ORR<>	Compare floating point number OR<> instruction	7		√	√	√	√
	ORR=	Compare floating point number OR= instruction	7		√	√	√	√
	ORR>	Compare floating point number OR> instruction	7		√	√	√	√
	ORR>=	Compare floating point number OR>= instruction	7		√	√	√	√
	OUT	Coil output instruction	1		√	√	√	√
	OUT Sxx	SFC state jump	3		√	√	√	√
P	PID	PID instruction	9		√	√	√	√
	PLS	Pulse output instruction of envelope	7		√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	PLSR	Count pulse with ACC/DEC output instruction	10		√	√	√	√
	PLSV	Variable speed pulse output instruction	8	Zero, carry, borrow	√	√	√	√
	PLSY	High-speed pulse output instruction	9		√	√	√	√
	POWER	Floating point number exponentiation instruction	10	Zero, carry, borrow	√	√	√	√
	PUSH	Push instruction	7		√	√	√	√
	PWM	Pulse output instruction	7		√	√	√	√
	PLSB	Count pulse with base frequency and ACC/DEC output instruction	12	Zero, carry, borrow		√		√
R	RAD	Floating point number angle->rad	7	Zero, carry, borrow				√
	RADD	Add floating point number instruction	10	Zero, carry, borrow	√	√	√	√
	RAMP	Ramp wave signal output instruction	12		√	√	√	√
	RCL	16-bit carry circular shift left instruction	7	Carry	√	√	√	√
	RCR	16-bit carry circular shift right instruction	7	Carry	√	√	√	√
	RCV	Free port receiving instruction	7		√	√	√	√
	RDIV	Divide floating point number instruction	10	Zero, carry, borrow	√	√		√
	REF	Instant refresh I/O instruction	5		√	√	√	√
	REFF	Set input filtering constant instruction	3		√	√	√	√
	RET	SFC program end	1		√	√	√	√
	RLCNV	Floating point engineering conversion instruction	12	Zero, carry, borrow	√			√
	RMOV	Move floating point number data transmission	7		√	√	√	√
	RMUL	Multiply floating point number instruction	10	Zero, carry, borrow	√	√	√	√
	RND	Generate random number instruction	3	Zero				√
	RNEG	Negative floating point number instruction	7	Zero, carry, borrow	√	√	√	√
	ROL	16-bit circular shift left instruction	7	Carry	√	√	√	√
	ROR	16-bit circular shift right instruction	7	Carry	√	√	√	√
	RSQT	Square root floating point number instruction	7	Zero, carry, borrow	√	√	√	√
	RST	Coil reset instruction	1		√	√	√	√
R	RST Sxx	SFC state reset	3		√	√	√	√
	RSUB	Substract floating point number instruction	10	Zero, carry, borrow	√	√	√	√
	RSUM	Sum floating point number instruction	9	Zero, carry, borrow	√	√	√	√
	RVABS	Floating point number absolute value instruction	7	Zero, carry, borrow	√	√	√	√
	RCMP	Compare and set floating point number instruction	9		√*			√
S	SAVER	Write extension file register	7	Zero, carry, borrow				√
	SCL	Locate coordinate	7	Zero, carry, borrow				√
	SEG	Word to 7-segment code instruction	5	Zero, carry, borrow	√	√	√	√
	SER	Search data	9	Zero, carry, borrow				√
	SET	Coil set instruction	1		√	√	√	√
	SET Sxx	SFC state transfer	3		√	√	√	√
	SFTL	Shift left byte instruction	9		√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	SFTR	Shift right byte instruction	9		√	√	√	√
	SHL	16-bit shift left instruction	7		√	√	√	√
	SHR	16-bit shift right instruction	7		√	√	√	√
	SIN	Floating point number SIN instruction	7	Zero, carry, borrow	√	√	√	√
	SPD	Pulse detection instruction	7		√	√	√	√
	SQT	Square root integer instructions	5	Zero, carry, borrow	√	√	√	√
	STL	SFC state load instruction	3		√	√	√	√
	STOH	Second to time (hour, minute and second) instruction	5					√
	STOP	User program stop	1		√	√	√	√
	STRADD	Add string	7	Zero, carry, borrow				√
	STRINSTR	Search string	9	Zero, carry, borrow				√
	STRLEFT	Read string from the left	7	Zero, carry, borrow				√
	STRLEN	Detect string length	5	Zero, carry, borrow				√
	STRMIDR	Read any strings	7	Zero, carry, borrow				√
	STRMIDW	Replace any strings	7	Zero, carry, borrow				√
	STRMOV	Move string	5	Zero, carry, borrow				√
	STRRIGHT	Read string from the right	7	Zero, carry, borrow				√
	SUB	Subtract integer instruction	7	Zero, carry, borrow	√	√	√	√
	SUM	Sum integer instruction	8	Zero, carry, borrow	√	√	√	√
	SWAP	Swap bytes	3		√	√	√	√
T	TADD	Add clock instruction	7	Zero, carry	√	√	√	√
	TAN	Floating point number TAN instruction	7	Zero, carry, borrow	√	√	√	√
	TCMP<	Compare time< instruction	7		√	√	√	√
	TCMP<=	Compare time<= instruction	7		√	√	√	√
	TCMP<>	Compare time<> instruction	7		√	√	√	√
	TCMP=	Compare time= instruction	7		√	√	√	√
	TCMP>	Compare time> instruction	7		√	√	√	√
	TCMP>=	Compare time>= instruction	7		√	√	√	√
	TKY	EEPROM write instruction	7					√
	TMON	Monostable timing instruction	5		√	√	√	√
	TO	Write word to special module buffer register instruction	9		√			√
	TOF	Off-delay timing instruction	5		√	√	√	√
	TON	On-delay timing instruction	5		√	√	√	√
	TONR	On-delay remember timing instruction	5		√	√	√	√
	TRD	Read real-time clock instruction	3		√	√	√	√
	TRIANGLE	Triangle wave signal output instruction	12		√	√	√	√
	TSUB	Subtract clock instruction	7	Zero, borrow	√	√	√	√
	TWR	Write real-time clock instruction	3		√	√	√	√
U	UNI	4bit combination instruction for 16bit data	7	Zero, carry, borrow				√
V	VABS	Integer absolute value instruction	5	Zero, carry, borrow	√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	VRRD	Read analog potentiometer value instruction	5		√	√		
W	WAND	AND word instruction	7		√	√	√	√
	WDT	User program watchdog reset	1		√	√	√	√
	WINV	NOT word instruction	5		√	√	√	√
	WOR	OR word instruction	7		√	√	√	√
	WSFL	Shift left word instruction	9	Zero, carry, borrow	√	√	√	√
	WSFR	Shift right word instruction	9	Zero, carry, borrow	√	√	√	√
	WTOB	Data separation instruction for byte unit	7	Zero, carry, borrow				√
	WXOR	Exclusive-OR word instruction	7		√	√	√	√
X	XCH	Exchange word	5		√	√	√	√
	XMT	Free port sending instruction	7		√	√	√	√
Z	ZONE	Zone control	9	Zero, carry, borrow				√
	ZRN	Regress to origin instruction	11	Zero, carry, borrow	√	√		√
	ZRST	Batch bit reset instruction	5		√	√	√	√
	ZSET	Set batch bit instruction	5		√	√	√	√
Note: * only applies to EC20.								

Appendix 10 Classified instruction index

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
Basic instruction	LD	NO contact power-flow loading	1		√	√	√	√
	LDI	NC contact power-flow loading	1		√	√	√	√
	AND	NO contact power-flow and	1		√	√	√	√
	ANI	NC contact power-flow and	1		√	√	√	√
	OR	NO contact power-flow or	1		√	√	√	√
	ORI	NC contact power-flow or	1		√	√	√	√
	OUT	Coil output instruction	1		√	√	√	√
	SET	Coil set instruction	1		√	√	√	√
	RST	Coil reset instruction	1		√	√	√	√
	ANB	Power flow block and instruction	1		√	√	√	√
	ORB	Power flow or instruction	1		√	√	√	√
	INV	Power flow block inverse	1		√	√	√	√
	NOP	No operation	1		√	√	√	√
	MPS	Output power-flow input stack	1		√	√	√	√
	MRD	Read output power-flow stack top value	1		√	√	√	√
	MPP	Output power-flow stack pop off	1		√	√	√	√
	MC	Main control	3		√	√	√	√
	MCR	Main control remove	1		√	√	√	√
	EU	Power flow rising edge detection	2		√	√	√	√
	ED	Power flow falling edge detection	2		√	√	√	√
	TON	On-delay timing instruction	5		√	√	√	√
	TOF	Off-delay timing instruction	5		√	√	√	√
	TMON	Monostable timing instruction	5		√	√	√	√
	TONR	On-delay remember timing instruction	5		√	√	√	√
	CTU	16-bit counter counting up instruction	5		√	√	√	√
	CTR	16-bit counter loop cycle counting instruction	5		√	√	√	√
	DCNT	32-bit counting instruction	7		√	√	√	√
Program control instruction	LBL	Jump label definition	3		√	√	√	√
	CJ	Conditional jump	3		√	√	√	√
	CALL	Calling a subprogram	Depend on the program		√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	CSRET	Conditional return from user subprogram	1		√	√	√	√
	CFEND	Conditional end from user main program	1		√	√	√	√
	CIRET	Conditional return from user interrupt subprogram	1		√	√	√	√
	FOR	Cycle instruction	3		√	√	√	√
	NEXT	Return from cycle	1		√	√	√	√
	WDT	User program watchdog reset	1		√	√	√	√
	STOP	User program stop	1		√	√	√	√
	EI	Enable interrupt instruction	1		√	√	√	√
	DI	Disable interrupt instruction	1		√	√	√	√
SFC instruction	STL	SFC state load instruction	3		√	√	√	√
	SET Sxx	SFC state transfer	3		√	√	√	√
	OUT Sxx	SFC state jump	3		√	√	√	√
	RST Sxx	SFC state reset	3		√	√	√	√
	RET	SFC program end	1		√	√	√	√
Data transmission instruction	MOV	Move word data transmission instruction	5		√	√	√	√
	DMOV	Move double word data transmission instruction	7		√	√	√	√
	RMOV	Move floating point number data transmission	7		√	√	√	
	BMOV	Move data block transmission instruction	7		√	√	√	√
	SWAP	Swap bytes	3		√	√	√	√
Data flow instruction	XCH	Exchange word	5		√	√	√	√
	DXCH	Exchange double word instruction	7		√	√	√	√
	FMOV	Fill data block instruction	7		√	√	√	√
	DFMOV	Fill data block double word instruction	9		√	√	√	√
	WSFR	Shift right word instruction	9	Zero, carry, borrow	√	√	√	√
	WSFL	Shift left word instruction	9	Zero, carry, borrow	√	√	√	√
	PUSH	Push instruction	7		√	√	√	√
	FIFO	First-in-first-out instruction	7		√	√	√	√
	LIFO	Last-in-first-out instruction	7		√	√	√	√
Integer/ double integer math instruction	ADD	Add integer instruction	7	Zero, carry, borrow	√	√	√	√
	DADD	Add double integer instruction	10	Zero, carry, borrow	√	√	√	√
	SUB	Subtract integer instruction	7	Zero, carry, borrow	√	√	√	√
	DSUB	Subtract double integer instruction	10	Zero, carry, borrow	√	√	√	√
	INC	Increment integer instruction	3	Zero, carry, borrow	√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	DINC	Increment double integer instruction	4	Zero, carry, borrow	√	√	√	√
	DEC	Decrement integer instruction	3	Zero, carry, borrow	√	√	√	√
	DDEC	Decrement double integer instruction	4	Zero, carry, borrow	√	√	√	√
	MUL	Multiply integer instruction	8	Zero, carry, borrow	√	√	√	√
	DMUL	Multiply double integer instruction	10	Zero, carry, borrow	√	√	√	√
	DIV	Divide integer instruction	7		√	√	√	√
	DDIV	Divide double integer instruction	10	Zero, carry, borrow	√	√	√	√
	VABS	Integer absolute value instruction	5	Zero, carry, borrow	√	√	√	√
	DVABS	Double integer absolute value instruction	7	Zero, carry, borrow	√	√	√	√
	NEG	Negative integer instruction	5	Zero, carry, borrow	√	√	√	√
	DNEG	Negative double integer instruction	7	Zero, carry, borrow	√	√	√	√
	SQT	Square root integer instruction	5	Zero, carry, borrow	√	√	√	√
	DSQT	Square root double integer instruction	7	Zero, carry, borrow	√	√	√	√
	SUM	Sum integer instruction	8	Zero, carry, borrow	√	√	√	√
	DSUM	Sum double integer instruction	9	Zero, carry, borrow	√	√	√	√
Floating point number math instruction	RADD	Add floating point number instruction	10	Zero, carry, borrow	√	√	√	√
	RSUB	Substract floating point number instruction	10	Zero, carry, borrow	√	√	√	√
	RMUL	Multiply floating point number instruction	10	Zero, carry, borrow	√	√	√	√
	RDIV	Divide floating point number instruction	10	Zero, carry, borrow	√	√	√	√
	RVABS	Floating point number absolute value instruction	7	Zero, carry, borrow	√	√	√	√
	RNEG	Negative floating point number instruction	7	Zero, carry, borrow	√	√	√	√
	RSQT	Square root floating point number instruction	7	Zero, carry, borrow	√	√	√	√
	SIN	Floating point number SIN instruction	7	Zero, carry, borrow	√	√	√	√
	COS	Floating point number COS instruction	7	Zero, carry, borrow	√	√	√	√
	TAN	Floating point number TAN instruction	7	Zero, carry, borrow	√	√	√	√
	LN	Floating point number LN instruction	7	Zero, carry, borrow	√	√	√	√
	EXP	Floating point number EXP instruction	7	Zero, carry, borrow	√	√	√	√
	POWER	Floating point number exponentiation instruction	10	Zero, carry, borrow	√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
Floating point math instruction	RSUM	Sum floating point number instruction	9	Zero, carry, borrow	√	√	√	√
	ASIN	Floating point number ASIN instruction	7	Zero, carry, borrow				√
	ACOS	Floating point number ACOS instruction	7	Zero, carry, borrow				√
	ATAN	Floating point number ATAN instruction	7	Zero, carry, borrow				√
	RAD	Floating point number angle->rad	7	Zero, carry, borrow				√
	DEG	Floating point number rad->angle	7	Zero, carry, borrow				√
	LOG	Floating point number LOG instruction	7	Zero, carry, borrow				√
Word/ double word logic instruction	WAND	AND word instruction	7		√	√	√	√
	DWAND	AND double word instruction	10		√	√	√	√
	WOR	OR word instruction	7		√	√	√	√
	DWOR	OR double word instruction	10		√	√	√	√
	WXOR	Exclusive-OR word instruction	7		√	√	√	√
	DWXOR	Exclusive-OR double word instruction	10		√	√	√	√
	WINV	NOT word instruction	5		√	√	√	√
	DWINV	NOT double word instruction	7		√	√	√	√
Shift/ rotate instruction	ROR	16-bit circular shift right instruction	7	Carry	√	√	√	√
	DROR	32-bit circular shift right instruction	9	Carry	√	√	√	√
	ROL	16-bit circular shift left instruction	7	Carry	√	√	√	√
	DROL	32-bit circular shift left instruction	9	Carry	√	√	√	√
	RCR	16-bit carry circular shift right instruction	7	Carry	√	√	√	√
	DRCR	32-bit carry circular shift right instruction	9	Carry	√	√	√	√
	RCL	16-bit carry circular shift left instruction	7	Carry	√	√	√	√
	DRCL	32-bit carry circular shift left instruction	9	Carry	√	√	√	√
	SHR	16-bit shift right instruction	7		√	√	√	√
	DSHR	32-bit shift right instruction	9		√	√	√	√
	SHL	16-bit shift left instruction	7		√	√	√	√
	DSHL	32-bit shift left instruction	9		√	√	√	√
	SFTL	Shift left byte instruction	9		√	√	√	√
	SFTR	Shift right byte instruction	9		√	√	√	√
Enhanced bit logic instruction	DECO	Decode instruction	5		√	√	√	√
	ENCO	Encode instruction	5		√	√	√	√
	BITS	Counting ON bit in word instruction	5		√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	DBITS	Counting ON bit in double word instruction	6		√	√	√	√
	ZRST	Batch bit reset instruction	5		√	√	√	√
	ZSET	Set batch bit instruction	5		√	√	√	√
	BON	Judging ON bit in word instruction	7					√
High-speed I/O instruction	HCNT	High-speed counter drive instruction	7		√	√	√	√
	DHSCS	High-speed counting compare set instruction	10		√	√	√	√
	DHSCR	High-speed counting compare reset instruction	10		√	√	√	√
	DHSCI	High-speed counting interrupt trigger instruction	10		√	√	√	√
	DHSZ	High-speed counting zone compare instruction	13		√	√	√	√
	DHST	High-speed counting table compare instruction	10		√	√	√	√
	DHSP	High-speed counting table compare pulse output instruction	10		√	√	√	√
	SPD	Pulse detection instruction	7		√	√	√	√
	PLSY	High-speed pulse output instruction	9		√	√	√	√
	PLSR	Count pulse with ACC/DEC output instruction	10		√	√	√	√
	PLSB	Count pulse with base frequency and ACC/DEC output instruction	12	Zero, carry, borrow		√		√
	PWM	Pulse output instruction	7		√	√	√	√
	PLS	Pulse output instruction of envelope	7		√	√	√	√
Control calculation instruction	PID	PID instruction	9		√	√	√	√
	RAMP	Ramp wave signal output instruction	12		√	√	√	√
	TRIANGLE	Triangle wave signal output instruction	12		√	√	√	√
	HACKLE	Hackle wave signal output instruction	12		√	√	√	√
	ABSD	Absolute drum control instruction	9	Zero, carry, borrow	√			√
	DABSD	Double word absolute drum control instruction	11	Zero, carry, borrow	√			√
	ALT	Alternate output instruction	3	Zero, carry, borrow	√		√	√
External equipment instruction	FROM	Read word from special module buffer register instruction	9		√			√
	DFROM	Read double word from special module buffer register instruction	10		√			√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	TO	Write word to special module buffer register instruction	9		√			√
	DTO	Write double word to special module buffer register instruction	10		√			√
	VRRD	Read analog potentiometer value instruction	5		√	√		
	REFF	Set input filtering constant instruction	3		√	√	√	√
	REF	Instant refresh I/O instruction	5		√	√	√	√
	EROMWR	EEPROM write instruction	7		√	√		√
	PR	Print instruction	5		√*			√
	TKY	Numeric key input instruction	7					√
Locating instruction	ABS	Read current value instruction	8	Zero, carry, borrow	√	√		√
	ZRN	Regress to origin instruction	11	Zero, carry, borrow	√	√	√	√
	PLSV	Variable speed pulse output instruction	8	Zero, carry, borrow	√	√	√	√
	DRVI	Relative position control instruction	11	Zero, carry, borrow	√*	√	√	√
	DRVA	Absolute position control instruction	11	Zero, carry, borrow	√*	√	√	√
	DSZR	Regress to origin with DOG search instruction	9	Zero, carry, borrow	√		√	√
	DVIT	Interrupt locating	11	Zero, carry, borrow	√		√	√
	LIN	Linear trace interpolation	12	Zero, carry, borrow				√
	CW	Clockwise circular trace interpolation	12	Zero, carry, borrow				√
	CCW	Counterclockwise circular trace interpolation	12	Zero, carry, borrow				√
	MOVLINK	Synchronous control instruction	17	Zero, carry, borrow				√
Real-time clock instruction	TRD	Read real-time clock instruction	3		√	√	√	√
	TWR	Write real-time clock instruction	3		√	√	√	√
	TADD	Add clock instruction	7	Zero, carry	√	√	√	√
	TSUB	Subtract clock instruction	7	Zero, borrow	√	√	√	√
	HOUR	Timing list instruction	8		√	√	√	√
	HTOS	Time (hour, minute and second) to second instruction	5					√
	STOH	Second to time (hour, minute and second) instruction	5					√
Compare contact instruction	LD=	Compare integer LD= instruction	5		√	√	√	√
	LDD=	Compare double integer LD= instruction	7		√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	LDR=	Compare floating point number LD= instruction	7		√	√	√	√
	LD>	Compare integer LD> instruction	5		√	√	√	√
	LDD>	Compare double integer LD> instruction	7		√	√	√	√
	LDR>	Compare floating point number LD> instruction	7		√	√	√	√
	LD>=	Compare integer LD>= instruction	5		√	√	√	√
	LDD>=	Compare double integer LD>= instruction	7		√	√	√	√
Compare contact instruction	LD<	Compare integer LD< instruction	5		√	√	√	√
	LDD<	Compare double integer LD< instruction	7		√	√	√	√
	LDR<	Compare floating point number LD< instruction	7		√	√	√	√
	LD<=	Compare integer LD<= instruction	5		√	√	√	√
	LDD<=	Compare double integer LD<= instruction	7		√	√	√	√
	LDR<=	Compare floating point number LD<= instruction	7		√	√	√	√
	LD<>	Compare integer LD<> instruction	5		√	√	√	√
	LDD<>	Compare double integer LD<> instruction	7		√	√	√	√
	LDR<>	Compare floating point number LD<> instruction	7		√	√	√	√
	AND=	Compare integer AND= instruction	5		√	√	√	√
	ANDD=	Compare double integer AND= instruction	7		√	√	√	√
	ANDR=	Compare floating point number AND= instruction	7		√	√	√	√
	AND>	Compare integer AND> instruction	5		√	√	√	√
	ANDD>	Compare double integer AND> instruction	7		√	√	√	√
	ANDR>	Compare floating point number AND> instruction	7		√	√	√	√
	AND>=	Compare integer AND>= instruction	5		√	√	√	√
	ANDD>=	Compare double integer AND>= instruction	7		√	√	√	√
	ANDR>=	Compare floating point number AND>= instruction	7		√	√	√	√
	AND<	Compare integer AND< instruction	5		√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	ANDD<	Compare double integer AND< instruction	7		√	√	√	√
	ANDR<	Compare floating point number AND< instruction	7		√	√	√	√
	AND<=	Compare integer AND<= instruction	5		√	√	√	√
	ANDD<=	Compare double integer AND<= instruction	7		√	√	√	√
	ANDR<=	Compare floating point number AND<= instruction	7		√	√	√	√
Compare contact instruction	AND<>	Compare integer AND<> instruction	5		√	√	√	√
	ANDD<>	Compare double integer AND<> instruction	7		√	√	√	√
	ANDR<>	Compare floating point number AND<> instruction	7		√	√	√	√
	OR=	Compare integer OR= instruction	5		√	√	√	√
	ORD=	Compare double integer OR= instruction	7		√	√	√	√
	ORR=	Compare floating point number OR= instruction	7		√	√	√	√
	OR>	Compare integer OR> instruction	5		√	√	√	√
	ORD>	Compare double integer OR> instruction	7		√	√	√	√
	ORR>	Compare floating point number OR> instruction	7		√	√	√	√
	OR>=	Compare integer OR>= instruction	5		√	√	√	√
	ORD>=	Compare double integer OR>= instruction	7		√	√	√	√
	ORR>=	Compare floating point number OR>= instruction	7		√	√		√
	OR<	Compare integer OR< instruction	5		√	√	√	√
	ORD<	Compare double integer OR< instruction	7		√	√	√	√
	ORR<	Compare floating point number OR< instruction	7		√	√	√	√
	OR<=	Compare integer OR<= instruction	5		√	√	√	√
	ORD<=	Compare double integer OR<= instruction	7		√	√	√	√
	ORR<=	Compare floating point number OR<= instruction	7		√	√		√
	OR<>	Compare integer OR<> instruction	5		√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	ORD<>	Compare double integer OR<> instruction	7		√	√	√	√
	ORR<>	Compare floating point number OR<> instruction	7		√	√	√	√
	CMP	Compare and set integer instruction	7		√*			√
	LCMP	Compare and set double integer instruction	9		√*			√
	RCMP	Compare and set floating point number instruction	9		√*			√
Data converting instruction	ITD	Integer to double integer instruction	6	Zero, carry, borrow	√	√	√	√
	DTI	Double integer to integer instruction	6	Zero, carry, borrow	√	√	√	√
	FLT	Integer to floating point number instruction	6	Zero, carry, borrow	√	√	√	√
	DFLT	Double integer to floating point number instruction	7	Zero, carry, borrow	√	√	√	√
	INT	Floating point number to integer instruction	6	Zero, carry, borrow	√	√	√	√
	DINT	Floating point number to double integer instruction	7	Zero, carry, borrow	√	√	√	√
	BCD	Word to 16-bit BCD instruction	5	Zero, carry, borrow	√	√	√	√
	DBCD	Double word to 32-bit BCD instruction	7	Zero, carry, borrow	√	√	√	√
	BIN	16-bit BCD to word instruction	5	Zero, carry, borrow	√	√	√	√
	DBIN	32-bit BCD to double word instruction	7	Zero, carry, borrow	√	√	√	√
	GRY	Word to 16-bit Gray code instruction	5	Zero, carry, borrow	√	√	√	√
	DGRY	Double word to 32-bit Gray code instruction	7	Zero, carry, borrow	√	√	√	√
	GBIN	16-bit Gray code to word instruction	5	Zero, carry, borrow	√	√	√	√
	DGBIN	32-bit Gray code to double word instruction	7	Zero, carry, borrow	√	√	√	√
	SEG	Word to 7-segment code instruction	5	Zero, carry, borrow	√	√	√	√
	ASC	ASCII code conversion instruction	19	Zero, carry, borrow	√	√	√	√
	ITA	Hexadecimal number-ASCII code conversion instruction	7	Zero, carry, borrow	√	√	√	√
	ATI	ASCII code-hexadecimal number conversion instruction	7	Zero, carry, borrow	√	√	√	√
	LCNV	Engineering conversion instruction	9	Zero, carry, borrow	√			√
	RLCNV	Floating point engineering conversion instruction	12	Zero, carry, borrow	√			√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
Word contact instruction	BLD	Word bit contact LD instruction	5		√	√	√	√
	BLDI	Word bit contact LDI instruction	5		√	√	√	√
	BAND	Word bit contact AND instruction	5		√	√	√	√
	BANI	Word bit contact ANI instruction	5		√	√	√	√
	BOR	Word bit contact OR instruction	5		√	√	√	√
	BORI	Word bit contact ORI instruction	5		√	√	√	√
	BSET	Word bit coil set instruction	5		√	√	√	√
	BRST	Word bit coil reset instruction	5		√	√	√	√
	BOUT	Word bit coil output instruction	5		√	√	√	√
Communication instruction	Modbus	Master station communication instruction	8		√	√	√	√
	XMT	Free port sending instruction	7		√	√	√	√
	RCV	Free port receiving instruction	7		√	√	√	√
	IVFWD	Inverter forward rotation instruction	6		√*	√	√	√
	IVREV	Inverter reverse rotation instruction	6		√*	√	√	√
	IVDFWD	Inverter jogging forward rotation instruction	6		√*	√	√	√
	IVDREV	Inverter jogging reverse rotation instruction	6		√*	√	√	√
	IVSTOP	Inverter stop instruction	8		√*	√	√	√
	IVFRQ	Inverter set frequency instruction	8		√*	√	√	√
	IVWRT	Inverter write single register value instruction	10		√*	√	√	√
	IVRDST	Inverter read status instruction	10		√*	√	√	√
	IVRD	Inverter read single register value instruction	10		√*	√	√	√
	MODRW	MODBUS read/write instruction	14		√*	√	√	√
Data check instruction	CCITT	CCITT check instruction	7		√	√	√	√
	CRC16	CRC16 check instruction	7		√	√	√	√
	LRC	LRC check instruction	7		√	√	√	√
Compare date instruction	DCMP=	Compare date= instruction	7		√	√	√	√
	DCMP>	Compare date> instruction	7		√	√	√	√
	DCMP<	Compare date< instruction	7		√	√	√	√
	DCMP>=	Compare date>= instruction	7		√	√	√	√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
	DCMP<=	Compare date<= instruction	7		√	√	√	√
	DCMP<>	Compare date<> instruction	7		√	√	√	√
Compare time instruction	TCMP=	Compare time= instruction	7		√	√	√	√
	TCMP>	Compare time> instruction	7		√	√	√	√
	TCMP<	Compare time< instruction	7		√	√	√	√
	TCMP>=	Compare time>= instruction	7		√	√	√	√
	TCMP<=	Compare time<= instruction	7		√	√	√	√
	TCMP<>	Compare time<> instruction	7		√	√	√	√
Data processing instruction	MEAN	Mean instruction	7	Zero, carry, borrow				√
	WTOB	Data separation instruction for byte unit	7	Zero, carry, borrow				√
	BTOW	Data combination instruction for byte unit	7	Zero, carry, borrow				√
	UNI	4bit combination instruction for 16bit data	7	Zero, carry, borrow				√
	DIS	4bit separation instruction for 16bit data	7	Zero, carry, borrow				√
	ANS	Signal alarm set instruction	7	Zero, carry, borrow				√
	ANR	Signal alarm reset instruction	1	Zero, carry, borrow				√
Data block processing instruction	BKADD	Add batch data operation	9	Zero, carry, borrow				√
	BKSUB	Subtract batch data operation	9	Zero, carry, borrow				√
	BKCMP=,>,<,<>,<=,>=	Compare batch data	9	Zero, carry, borrow				√
Data table processing instruction	LIMIT	Upper/lower limit control	9	Zero, carry, borrow				√
	DBAND	Dead band control	9	Zero, carry, borrow				√
	ZONE	Zone control	9	Zero, carry, borrow				√
	SCL	Locate coordinate	7	Zero, carry, borrow				√
	SER	Search data	9	Zero, carry, borrow				√
String processing instruction	STRADD	Add string	7	Zero, carry, borrow				√
	STRLEN	Detect string length	5	Zero, carry, borrow				√
	STRRIGHT	Read string from the right	7	Zero, carry, borrow				√
	STRLEFT	Read string from the left	7	Zero, carry, borrow				√
	STRMIDR	Read any strings	7	Zero, carry, borrow				√
	STRMIDW	Replace any strings	7	Zero, carry, borrow				√
	STRINSTR	Search string	9	Zero, carry, borrow				√
	STRMOV	Move string	5	Zero, carry, borrow				√

Instruction		Instruction function	Program steps	Influenced flag bit	EC20	EC10	EC10V	EC20H
Extension file register instruction	LOADR	Read extension file register	5	Zero, carry, borrow				√
	SAVER	Write extension file register	7	Zero, carry, borrow				√
	INITR	Initialize extension register	5	Zero, carry, borrow				√
	LOGR	Log in extension register	11	Zero, carry, borrow				√
	INITER	Initialize extension file register	5	Zero, carry, borrow				√
Others	RND	Generate random number instruction	3	Zero				√
	DUTY	Generate timing pulse instruction	7					√
Note: * only applies to EC20.								